Fast Methods II

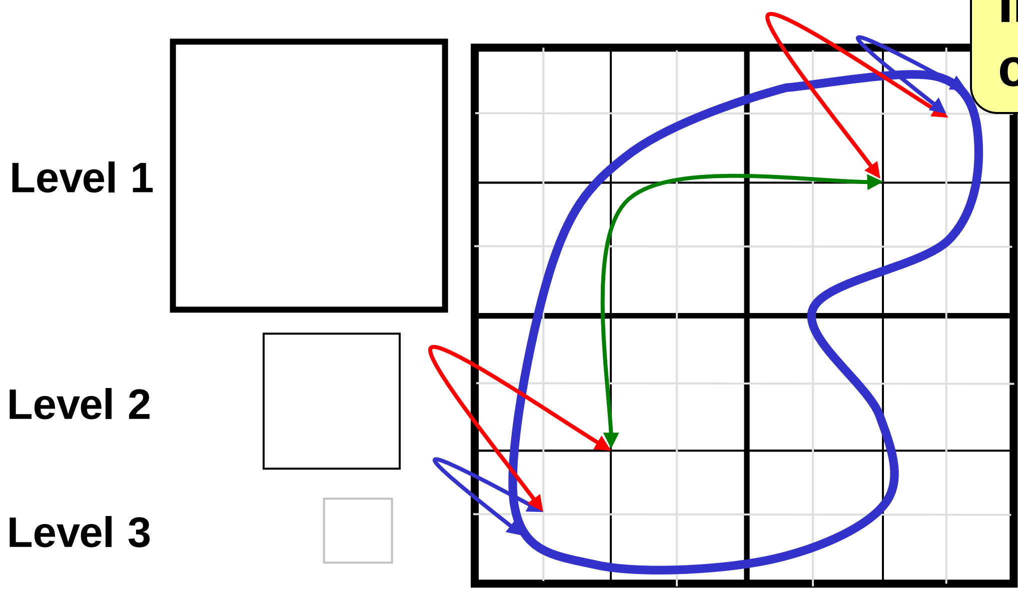
Donald R. Wilton

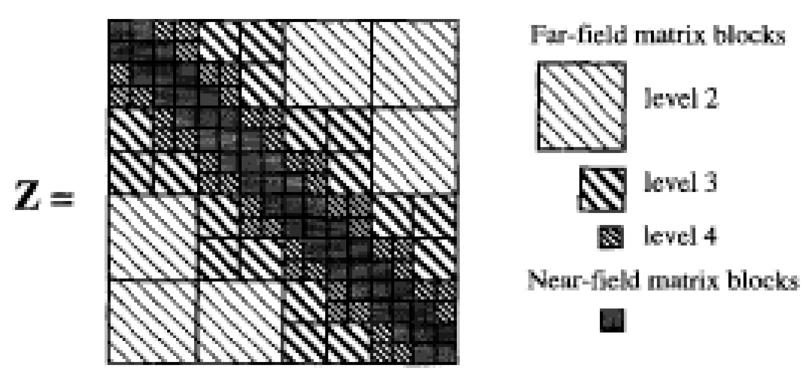
Vikram Jandhyala

Central Fast Method Ideas

- ***Fast*** **methods all employ a form of matrix or Green's function rank-reduced separability**
- ***Multi-level*** **schemes gain additional efficiency by a** ***hierarchical*** **grouping scheme.**

Information used at a lower level can be reused at a higher level





Examples of Separable Expansions of Green's Function

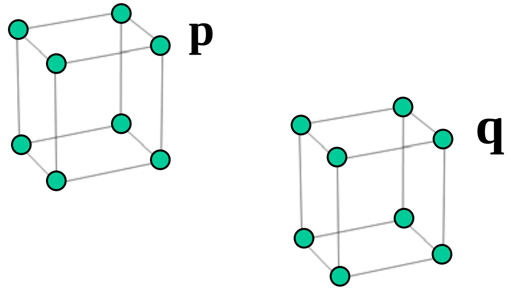
- **Taylor Series (elegant but difficult to apply):**

$$G(\mathbf{r},\mathbf{r}') \approx \sum_{p=0}^{P}\sum_{q=0}^{Q}\frac{1}{p!q!}\left[\left(\mathbf{r}-\mathbf{r}_0\right)\cdot\nabla\right]^p\left[\left(\mathbf{r}'-\mathbf{r}_S\right)\cdot\nabla'\right]^q G(\mathbf{r},\mathbf{r}')\Big|_{\mathbf{r}=\mathbf{r}_0,\mathbf{r}'=\mathbf{r}_S}$$

- **Products of terms like** $\left(\mathbf{r}-\mathbf{r}_0\right)^p\left(\mathbf{r}'-\mathbf{r}_S\right)^q$, **where** $\mathbf{r}_0$, $\mathbf{r}_S$ **are centered in an obs. & source group, rsp.**
- **Works best for asymptotically smooth Green's functions, e.g. quasi-statics**
- **Dynamic case limited by wavelength**

Separable Expansions of Green's Fn, Cont'd

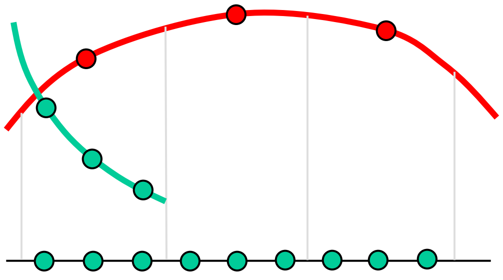
Polynomial Interpolation:



$$G(\mathbf{r},\mathbf{r}') \approx \sum_{\mathbf{p}} \sum_{\mathbf{q}} \overbrace{G(\mathbf{r}^{(\mathbf{p})},\mathbf{r}'^{(\mathbf{q})})}^{G_{\mathbf{p},\mathbf{q}}} L_{\mathbf{p}}(\mathbf{r}) L_{\mathbf{q}}(\mathbf{r}'),$$

$$\mathbf{p} = (p_x, p_y, p_z), \;\; \mathbf{q} = (q_x, q_y, q_z)$$

- **More accuracy simply implies using high order interpolation**
- **Wavelength limited**
- **Hierarchical principle: Lagrange polynomials** $L_{\mathbf{p}}(\mathbf{r}) = L_{p_1}(x) L_{p_2}(y) L_{p_3}(z)$ **at low levels (coarse discretization) are represented in terms of those at higher levels (fine discretization).**



Separable Expansions of Green's Fn, Cont'd

CGFFT, AIM, Pre-Corrected FFT:

$$\sum_{\mathbf{n}} <\Lambda_{\mathbf{m}}, G(\mathbf{r}-\mathbf{r}'), \Lambda_{\mathbf{n}}> I_{\mathbf{n}}, \quad \underset{\text{CGFFT}}{\Rightarrow} \quad \underbrace{\left[Z'_{\mathbf{m-n}}\right]\left[I'_{\mathbf{n}}\right]}_{\text{extended to circulant form}}$$

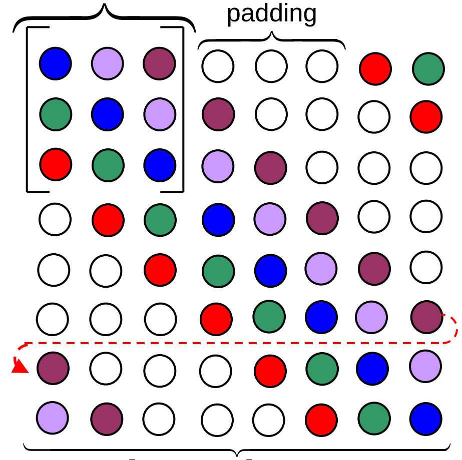
$$\sum_{\mathbf{n}} \overbrace{<\Lambda_{\mathbf{m}}, L_{\mathbf{p}}>}^{\text{precompute}} G_{\mathbf{p-q}} <L_{\mathbf{q}}, \Lambda_{\mathbf{n}}> I_{\mathbf{n}} \quad \underset{\text{AIM}}{\Rightarrow}$$

$$\underbrace{Z'_{\mathbf{m}}}_{\text{1st col. of } Z'} = \sum_{\mathbf{p}} \tilde{Z}'_{\mathbf{p}} e^{-j\mathbf{m}\cdot\mathbf{k_p}} = \text{DFT}(\tilde{Z}'_{\mathbf{p}}) \Rightarrow Z'_{\mathbf{m-n}} = \sum_{\mathbf{p}} \tilde{Z}'_{\mathbf{p}} e^{-j\mathbf{m}\cdot\mathbf{k_p}} e^{j\mathbf{n}\cdot\mathbf{k_p}}$$

$$\mathbf{m} = (m_x\hat{\mathbf{x}} + m_y\hat{\mathbf{y}} + m_z\hat{\mathbf{z}}), \qquad \mathbf{k_p} = \frac{2\pi p_x}{N_x}\hat{\mathbf{x}} + \frac{2\pi p_y}{N_y}\hat{\mathbf{y}} + \frac{2\pi p_z}{N_z}\hat{\mathbf{z}}$$

Store only the first column

$\left[Z_{\mathbf{m-n}}\right]_{3\times 3}$ zero padding



$\left[Z'_{\mathbf{m-n}}\right]_{2^p\times 2^p}$

For efficient FFT, the augmented array $Z'_{\mathbf{m-n}}$ should be dim $2^P \times 2^P$

- **Separability follows from DFT representation; FFT automatically provides hierarchical scheme**
- **Green's function must be convolutional**
- **Requires space-filling, regular grid**

Separable Expansions of Green's Fn, Cont'd

FMM, MLFMA:

$$G(\mathbf{r},\mathbf{r}') \approx \oiint_{\hat{\mathbf{k}}} e^{j\mathbf{k}\cdot(\mathbf{r}-\mathbf{r}_l)} T(\hat{\mathbf{k}}\cdot\mathbf{R}_{l,l'}) e^{j\mathbf{k}\cdot(\mathbf{r}'-\mathbf{r}_{l'})} d\hat{\mathbf{k}}^2$$

$$\approx \sum_p \sum_q e^{j\mathbf{k}_{pq})\cdot(\mathbf{r}-\mathbf{r}_l)} T(\hat{\mathbf{k}}_{pq}\cdot\mathbf{R}_{l,l'}) e^{j\mathbf{k}_{pq}\cdot(\mathbf{r}'-\mathbf{r}_{l'})} \sin\theta_p \,\Delta\theta\,\Delta\phi$$

Translation operator :

$$T(\hat{\mathbf{k}}_{pq}\cdot\mathbf{R}_{l,l'}) \equiv \left[T_{pq}\right]_{l,l'} = \sum_p \sum_q \sum_{\ell=0}^{L} (-j)^{\ell} (2\ell+1) P_\ell\left(\hat{\mathbf{k}}_{pq}\cdot\mathbf{R}_{l,l'}\right)$$

- **Hierarchy provided by successive translation between (multi)-levels with interpolation ($\left[I_{pq}\right]_{l'-1}$) and anterpolation ($\left[I_{pq}\right]_{l-1}^{t}$) of translation operator:**

$\leftarrow$increasing levels, decreasing interpolation density

$$\left[T_{pq}\right]_{l,l'} = \left[T_{pq}\right]_{l,l-1} \left[I_{pq}\right]_{l-1}^{t} \left[T_{pq}\right]_{l-1,l-2} \left[I_{pq}\right]_{l-2}^{t}$$

$$\cdots\times \left[T_{pq}\right]_{3,2} \left[I_{pq}\right]_{2}^{t} \left[T_{pq}\right]_{2,2} \left[I_{pq}\right]_{2} \left[T_{pq}\right]_{2,3}$$

$$\cdots\times \left[I_{pq}\right]_{l'-2} \left[T_{pq}\right]_{l'-2,l'-1} \left[I_{pq}\right]_{l'-1} \left[T_{pq}\right]_{l'-1,l'}$$

increasing levels, decreasing interpolation density$\rightarrow$

$\left[I_{pq}\right]_{l'-1} = \hat{\mathbf{k}}$ **- space interpolation operator at level** l

Examples of Direct Methods

SVD:

- **Singular value decomposition can be used to directly obtain**

$$\mathbf{A} \approx \mathbf{U\Sigma V}^{\dagger}$$

where

$$\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_r]$$

$$\mathbf{U} = [\mathbf{u}_1,\ \mathbf{u}_2, \ldots,\ \mathbf{u}_r]$$

$\mathbf{\Sigma}_r = \mathbf{diag}(\sigma_1, \sigma_2, \ldots, \sigma_{\mathrm{r}})$ **are the singular values**

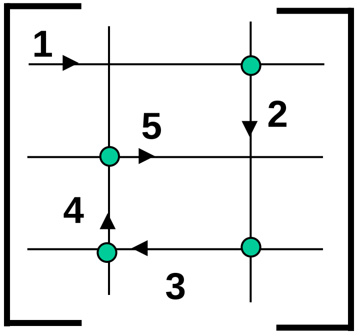
- **Method needs all of the original matrix block A and is inefficient**

Direct Methods, Cont'd

ACA:

- **Adaptive Cross Approximation builds a block by adding products** $\mathbf{u}_{r+1}\mathbf{v}_{r+1}^{t}$ **that are essentially rows and columns of the residual matrix, rsp.:**

$$\mathbf{A} \approx \mathbf{U}_r \mathbf{V}_r^{\dagger}$$



where

$$\mathbf{U}_r = \left[\mathbf{u}_1,\ \mathbf{u}_2, \ldots,\ \mathbf{u}_r\right]$$

$$\mathbf{V}_r = \left[\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_r\right]$$

- **Simple to apply**
- **Only necessary to compute rows and cols of A needed to form** $\mathbf{U}_r, \mathbf{V}_r$
- **Appears to work best for statics, moderate freqs.**

ACA Method, Cont'd

ACA Algorithm: (S.Kurz,O. Rain, and S. Rjasanow, "The Adaptive Cross-Approximation Technique for the 3-D Boundary-Element Method" IEEE TRANS. MAG., **38**, MAR. 2002.

1) $e_{i_{k+1}}^T R_k = e_{i_{k+1}}^T A - \sum_{l=1}^{k} (u_l)_{i_{k+1}} v_l^T$

2) $j_{k+1}\colon |(R_k)_{i_{k+1}, j_{k+1}}| = \max_j |(R_k)_{i_{k+1}, j}|$

3) $v_{k+1} = e_{i_{k+1}}^T R_k / (R_k)_{i_{k+1}, j_{k+1}}$

4) $u_{k+1} = A e_{j_{k+1}} - \sum_{l=1}^{k} (v_l)_{j_{k+1}} u_l$

5) $i_{k+2}\colon |(u_{k+1})_{i_{k+2}}| = \max_{i \neq i_{k+1}} |(u_{k+1})_i|$

6) $S_{k+1} = S_k + u_{k+1} v_{k+1}^T.$

$$e_i \equiv \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \leftarrow i \textbf{ th row}$$

Stopping criterion: $\|u_k\|_F \|v_k\|_F \leq \varepsilon \|S_k\|_F.$

with recursive norm computation,

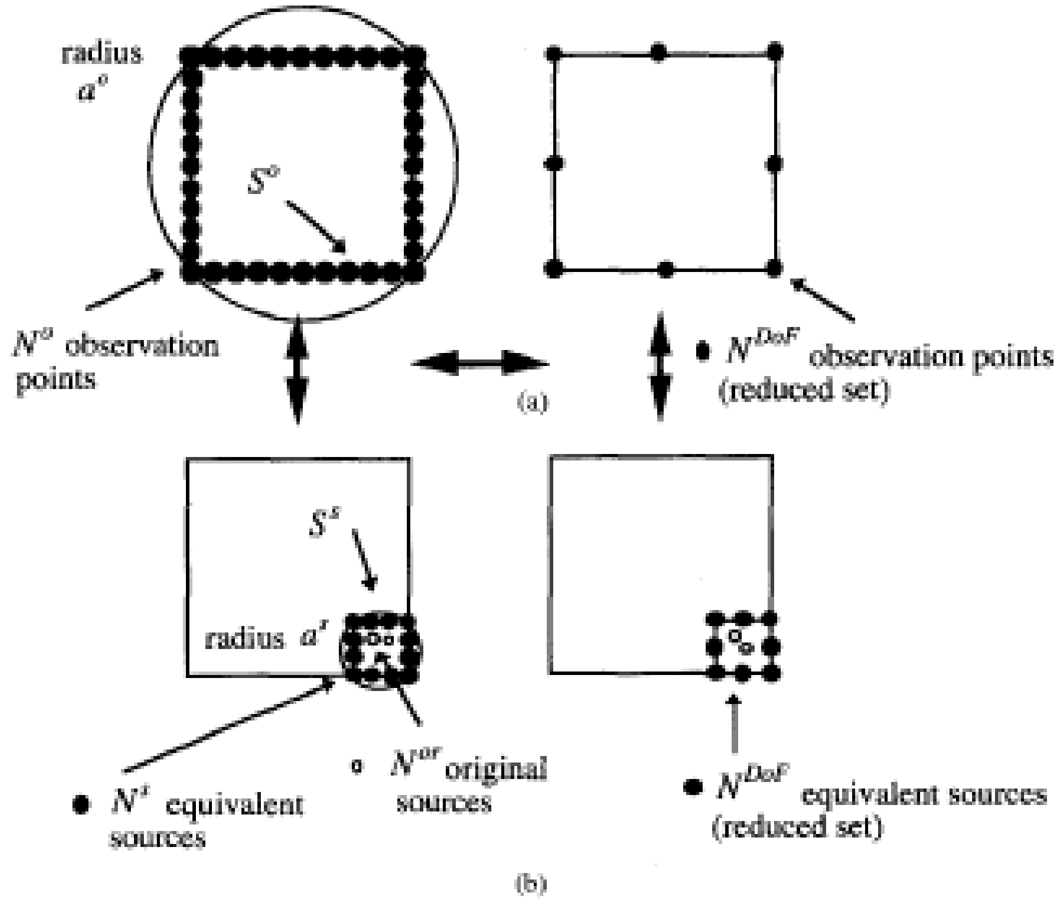
$$\|S_k\|_F^2 = \|S_{k-1}\|_F^2 + 2 \sum_{j=1}^{k-1} (u_j, u_k)(v_j, v_k) + \|u_k\|_F^2 \|v_k\|_F^2$$

Direct Methods, Cont'd

MLMDA

- **Multilevel Matrix Decomposition Algorithm (Michielssen, Boag)**

- **Uses equivalence principle and far-field DoF concepts for heirarchical representation**



Direct Methods, Cont'd

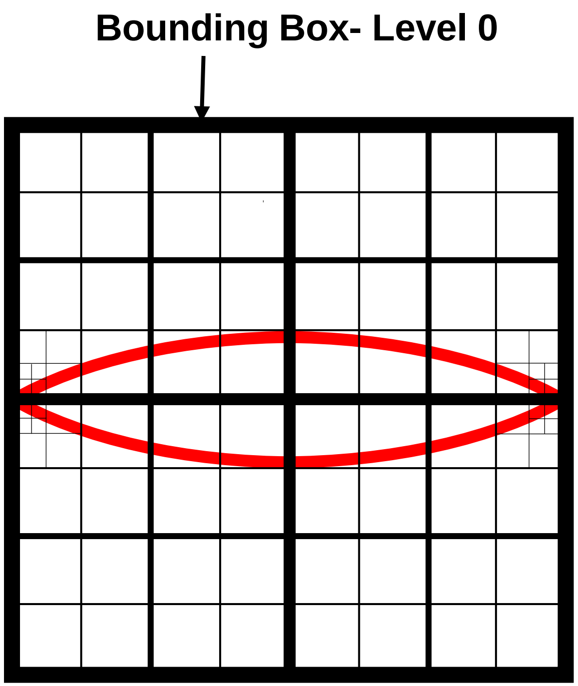
Rank-revealing QR decomposition

- **Columns of block are taken as bases for representing matrix through modified Gram-Schmidt orthogonalization to produce Q; $R=Q^tA$ (since $QQ^t = I$ implies $A=QR$)**

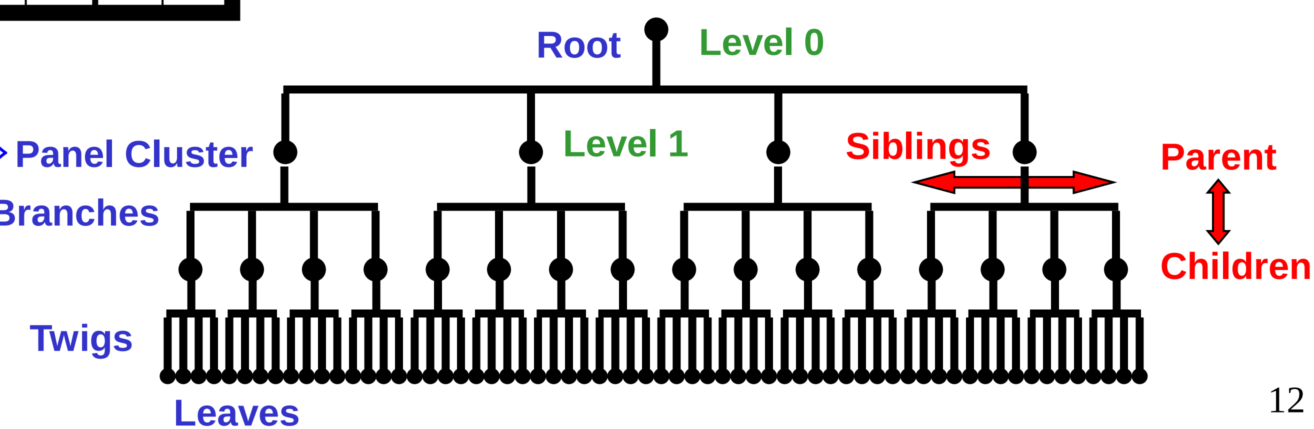
- **In principle, a low frequency method, but has been successfully applied to objects about 20 wavelengths in size**

- **Very efficient when combined with PILOT algorithm (Jandhyala)**

Problem Domains Are Generally Partitioned to Find Compressible Matrix Blocks



- Object bounding box is recursively subdivided into cells to form quad-tree (2D) or oct-tree (3D)
- No information stored for empty cells (panels)
- Roughly equal number of DoFs per cell
- Interactions between elements are now between groups of elements in different cells



Tree structures are often described using either botanical and/or genetic terminology!

Definitions of Sibling, Nearest Neighbor Shell, and Interaction Shell Sets

Define :

C_i^ℓ **- ith cell at ℓth level**

$P_{C_i^\ell}$ **- parent cell of cell C_i^ℓ**

$S_{C_j^{\ell+1}}$ ***- Sibling Set :***

$$S_{C_j^{\ell+1}} = \left\{ C_k^{\ell+1} \forall k \mid P_{C_k^{\ell+1}} = P_{C_j^{\ell+1}} \right\}$$

$=$ ***set of cells with the same parent***

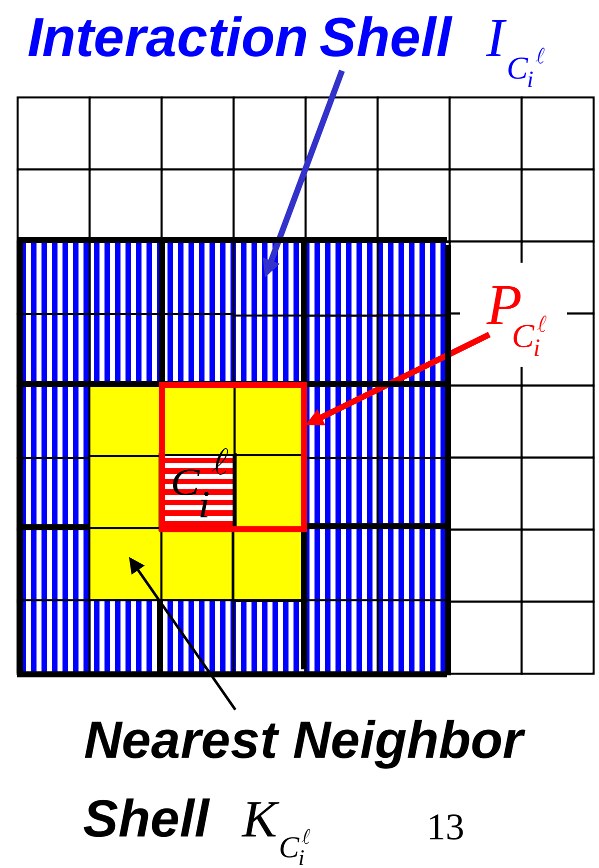
$K_{C_i^\ell}$ ***- Nearest Neighbor Shell :***

$$K_{C_i^\ell} = \left\{ \begin{array}{l} C_j^\ell \mid C_j^\ell \text{ is in the same level as } C_i^\ell \text{ and has} \\ \text{at least one point of contact with } C_i^\ell \end{array} \right\}$$

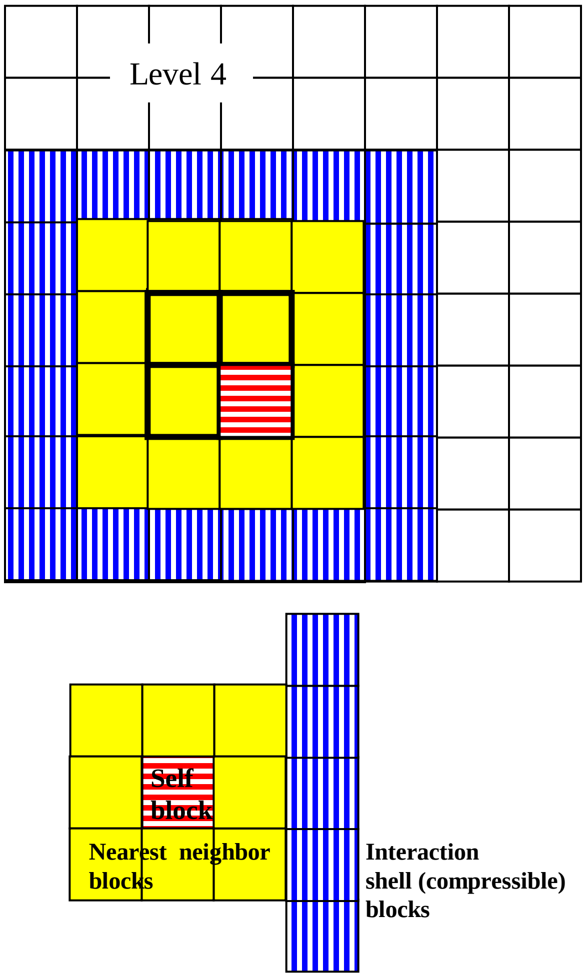
$I_{C_i^\ell}$ ***- Interaction Shell :***

$$I_{C_i^\ell} = \left\{ C_j^\ell \mid P_{C_j^\ell} \in K_{P_{C_i^\ell}} ; C_j^\ell \notin K_{C_i^\ell} \right\}$$

$=$ **set of cells C_j^ℓ at the same level as C_i^ℓ whose parents are in the nearest neighbor shell of C_i^ℓ's parent, but are not a nearest neighbor cell of C_i^ℓ**

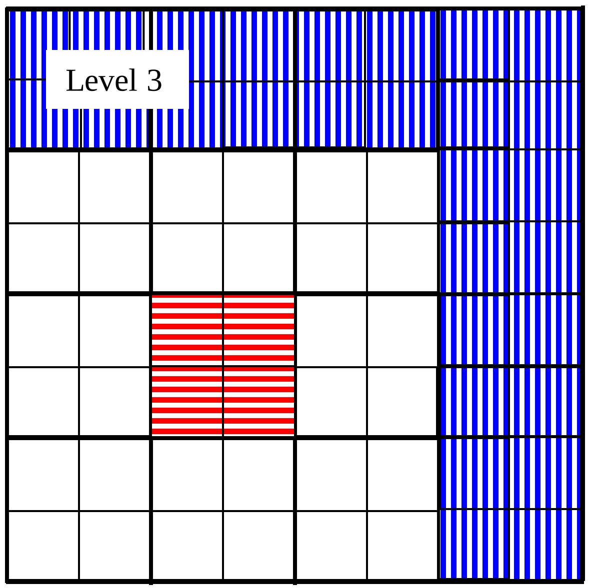


Begin at the Deepest Level and Fill System Matrix with Interactions between Elements in each Cell and Those of its Near Neighbors



- Find interactions between elements in each cell and elements in its near neighbor cells:
  - Self-blocks and nearest neighbor shell blocks are filled by usual MoM procedure
  - Interaction shell blocks are compressible, so fill using ACA, QR, SVD, FMM, etc.
- Treat all siblings as a group

Successively Move to Higher Levels (Larger Cell Sizes) and Fill (Compressed) Blocks Representing Coupling Between Elements in a Cell and Those of Same Level in its Interaction Shell



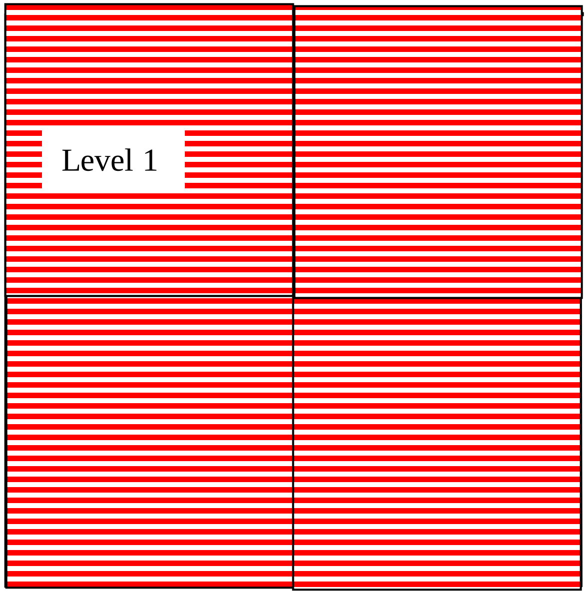
- Moving up a level, we next consider cells that are parents of the cells at the previous level

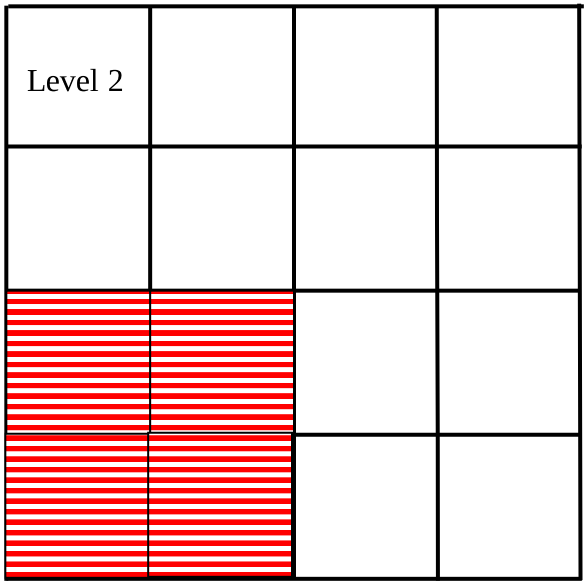
- Note that the nearest neighbor interactions at this level were treated at the previous level

- Hence, find interactions between each cell at this level and the cells of its interaction shell; the resulting interaction blocks are all compressible

- Repeat this procedure at each level until we reach level 2

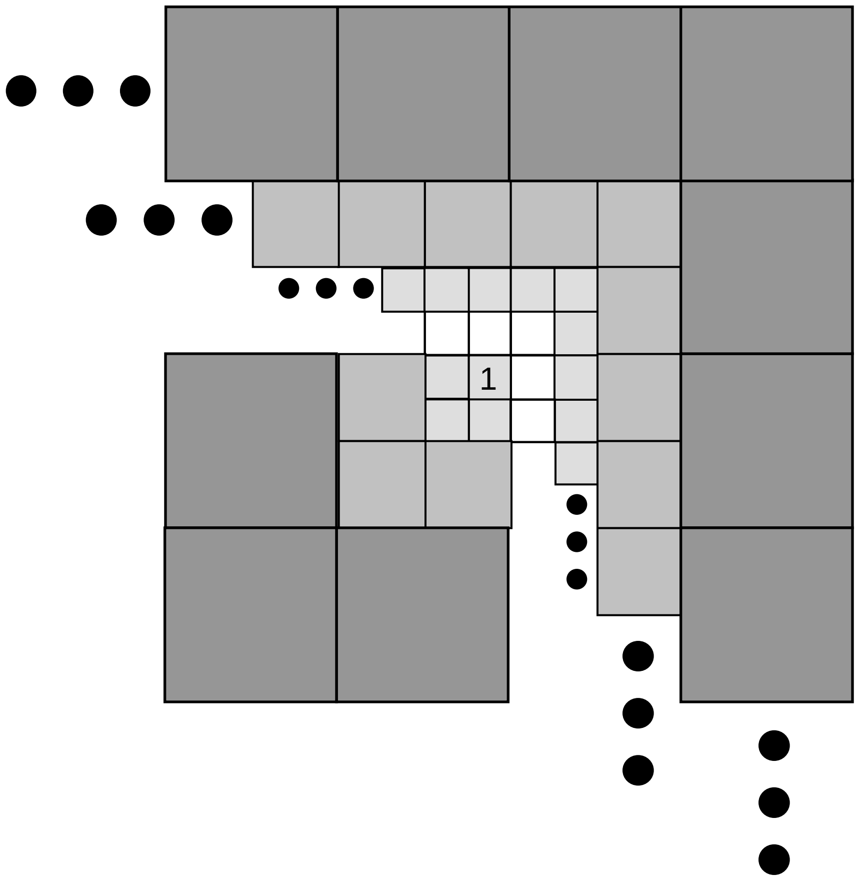
The Filling Procedure is Finished When Level 2 Is Reached





- **Level 1 has no nearest neighbor or interaction shells**
- **Level 2 has only previously-filled nearest neighbors**

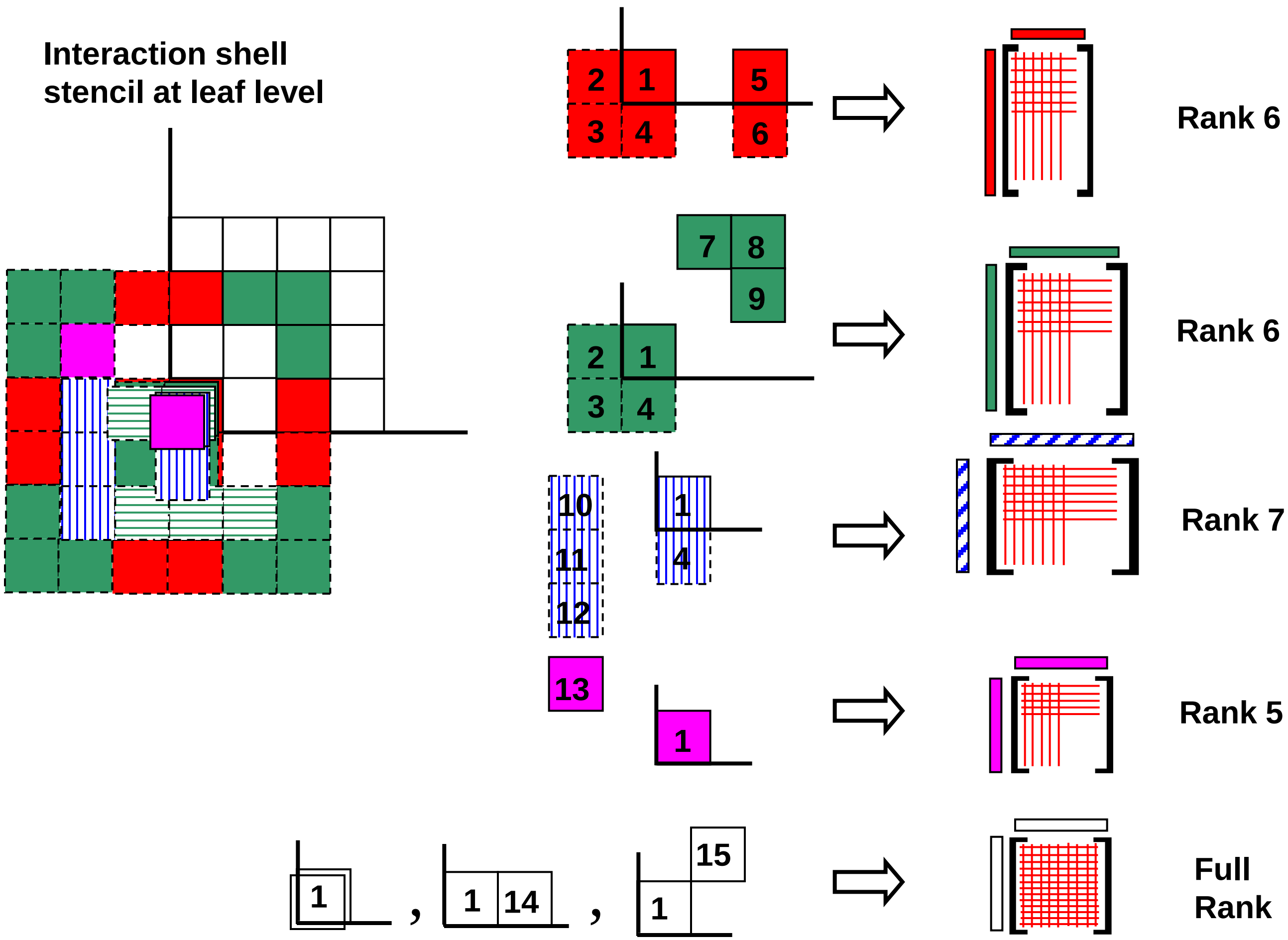
Note That We Tile All Interaction Domains Using Blocks of Ever-Increasing Size



- As levels are added, all interaction groups are “tiled” by the increasingly larger groups
- Maximum rank pattern remains same at each level up to scales of almost a wavelength
- FMM or similar algorithms can be used beginning at scale levels on the order of a wavelength or larger

The PILOT algorithm attempts to further compress the system matrix by combining neighboring groups of cells at each stage

Predetermined Interaction List Oct-Tree (PILOT) Algorithm for Domain Decomposition



- Attempt to group cell 1 with its siblings before interacting it with non-adjacent cell groups

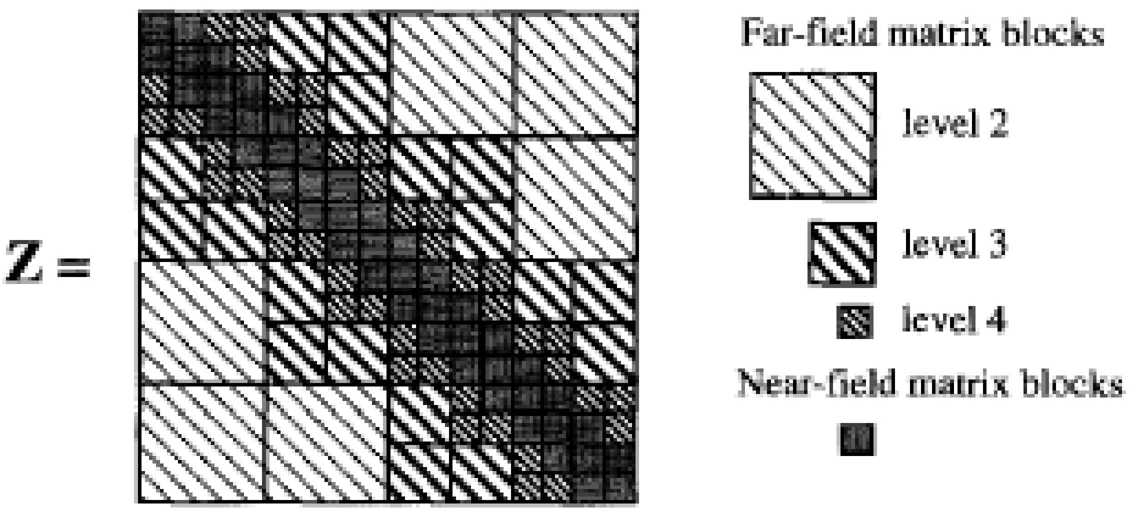
- Complete grouping pattern by rotational symmetry

- Cells 2,3 are adjacent to 10,11,12, so select a smaller sibling group
- Group so rotations produce only a single interaction between group pairs

- Only a single cell-to-cell interaction remains between non-adjacent cells in the interaction shell

- Near field interactions

Typical Matrix Block Decomposition



PILOT Performance: Cone Problem

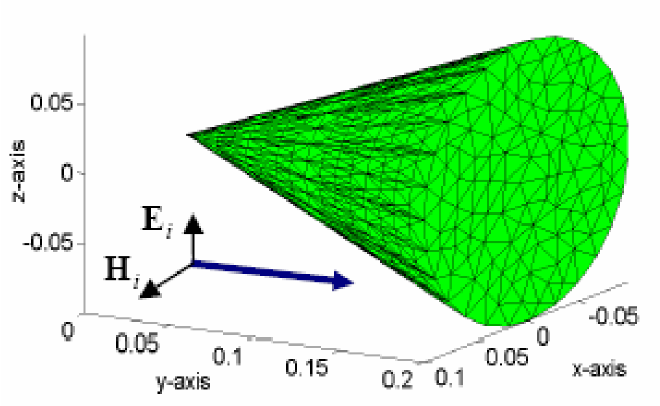


Figure 6a: Conducting cone and incident plane wave.

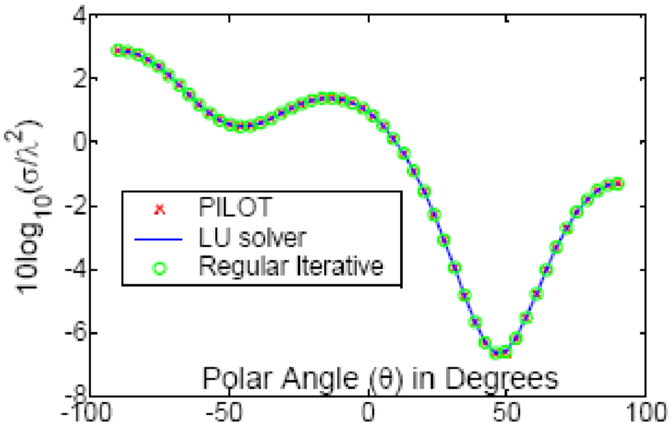


Figure 6b: The bi-static E-plane RCS

PILOT Performance : Cone Problem

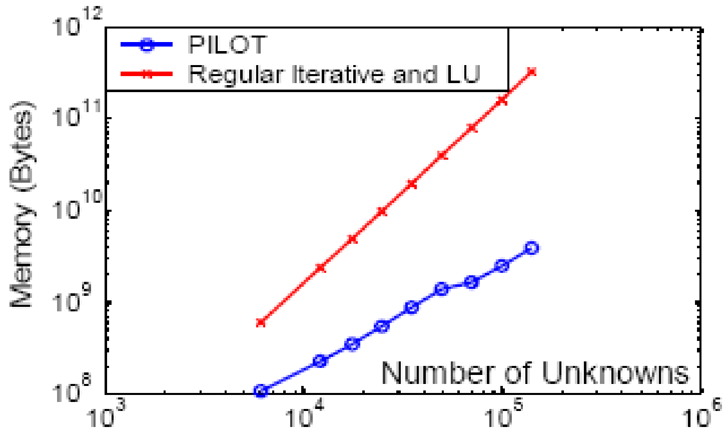


Figure 7a: Memory Requirement

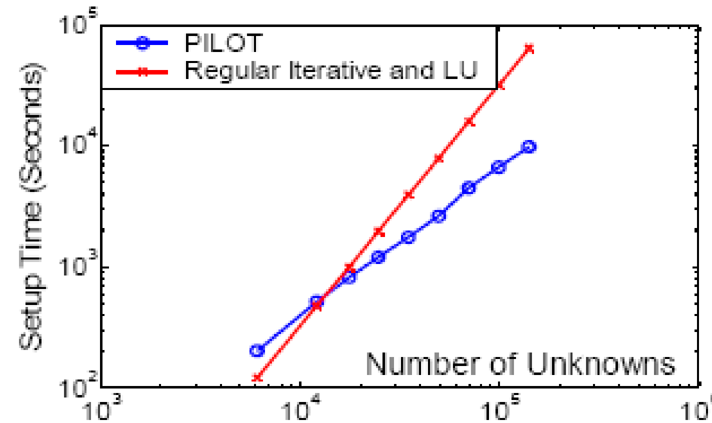


Figure 7b: Matrix Setup Time

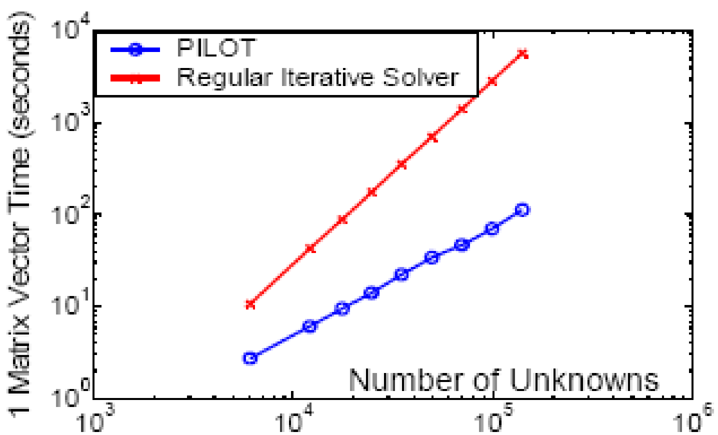


Figure 7c: Matrix Vector Product Time

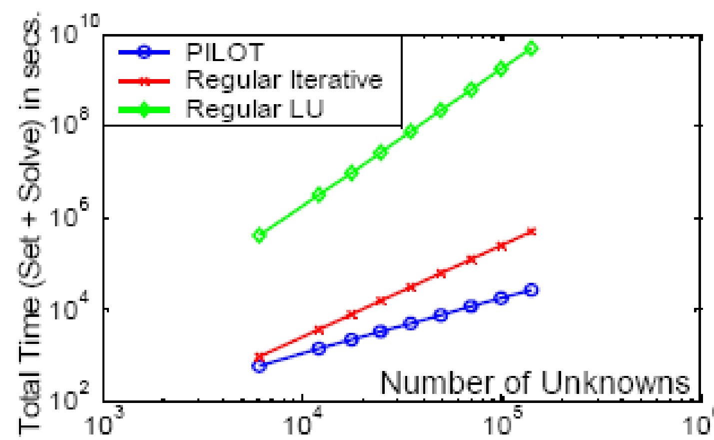


Figure 7d: Total Time Required

PILOT Performance : Drone Problem

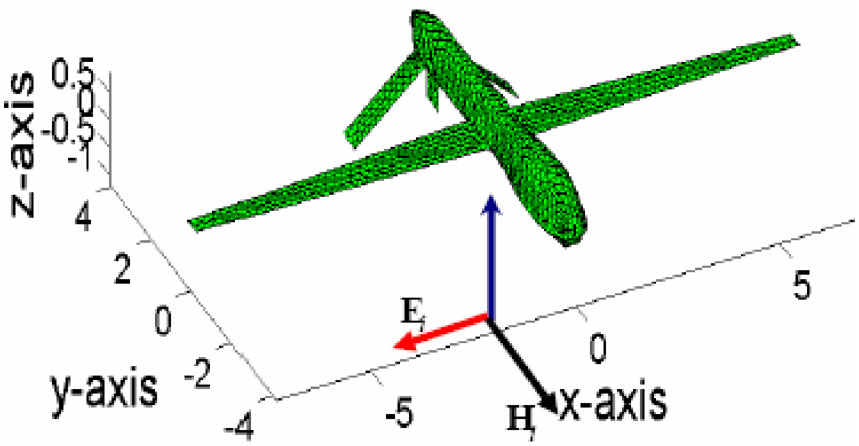


Figure 8a: Surface mesh for airborne drone

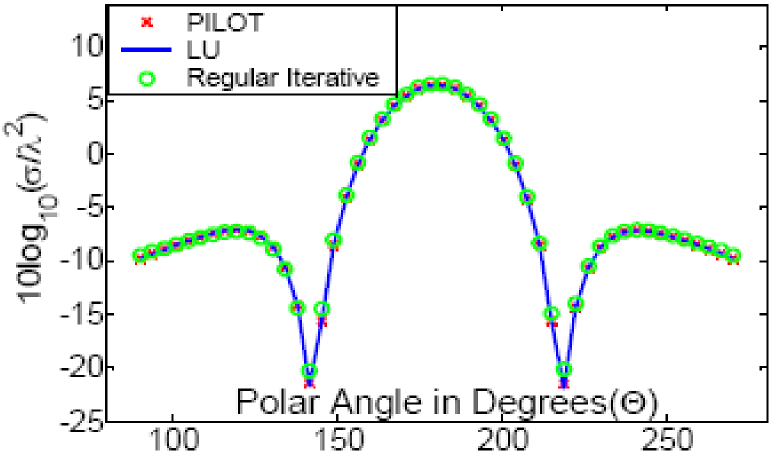


Figure 8b: The bi-static RCS of the drone

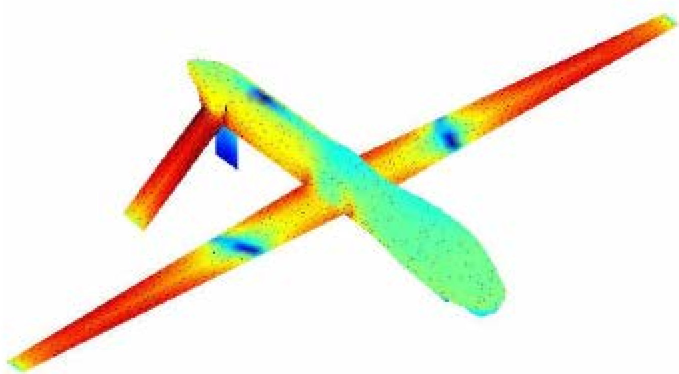


Figure 9a: The current density with LU

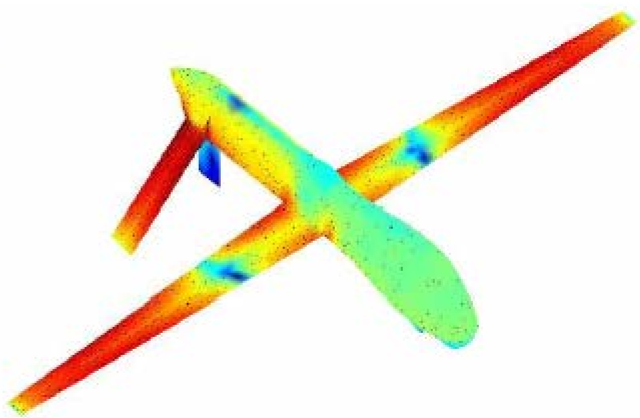


Figure 9b: The current density with PILOT

PILOT Performance vs. Frequency

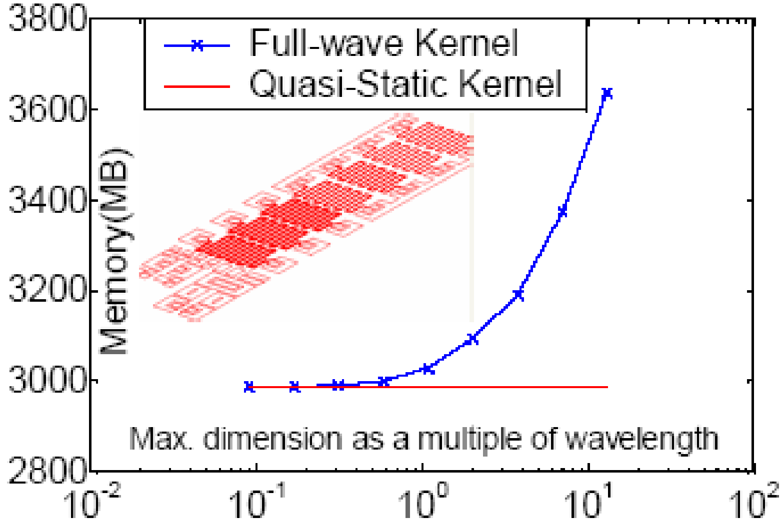


Figure 10a: Memory required by PILOT for a 2D structure. The corresponding MoM memory requirement is 540 GB.

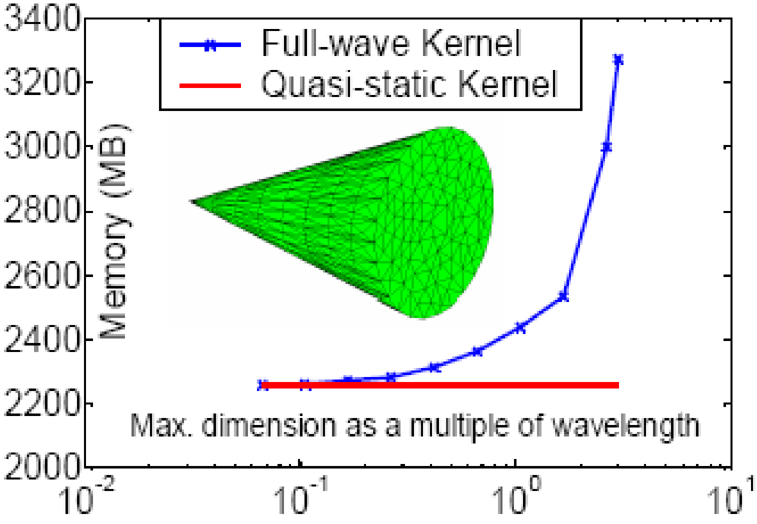


Figure 10b: Memory required by PILOT for a 3D structure. The corresponding MoM memory requirement is 150 GB.

Results can be improved by switching to an FMM or similar scheme when block sizes are on the order of a wavelength or larger.

The End