www.ozeninc.com
info@ozeninc.com
(408) 732 4665
1210 E Arques Ave St 207
Sunnyvale, CA 94085

Reliable World Class Insights

Your Silicon Valley Partner in Simulation

ANSYS Sales, Consulting, Training & Support

ANSYS

Technical Paper

Finite Antenna Array Analysis with a Unit Cell Domain Decomposition Method

ANSYS has developed an efficient iterative domain decomposition-based finite element technique for modeling repetitive antenna arrays. The scheme leverages the repeating nature of an array to minimize the setup time for the domain matrices and the amount of computational resources required for a solution. Comparison results between this technique and an explicit FEM solution are presented along with a solution performance for a 1,024- element dual-slant polarized Vivaldi array.

Phased array antennas offer significant advantages for radar design, including the ability to steer beams or nulls of the pattern with progressive phase shifts along neighboring elements. This ability to steer the beam eliminates the need for mechanical steering; it also provides the ability to rapidly steer one or more beams or nulls for scanning along multiple planes. The number of elements in an array can range from tens to thousands, depending on the application and frequency. As a result, these geometries are inherently complex and large, making them difficult to analyze with traditional electromagnetic simulation methods, such as finite elements or method of moments [1].

The traditional approach for simulating large phased arrays approximates that behavior by assuming an infinitely large array. In such an approach, only the geometric description of a single unit cell is required. Then a periodic boundary approximation solution for this single unit cell can be developed assuming it is placed in an infinitely large array. Such infinite array analysis has been the staple of antenna array design in which the solution for this single unit cell is multiplied by an array factor to determine an approximate behavior of the finite sized array. The approximate nature of this infinite array solution is a result of the fact that the environment, fields and coupling experienced by individual elements of the array vary according to their locations in the array (interior, edge, corner, etc.). Lacking this element-level knowledge introduces challenges in finite-sized array design. The design of a corporate feed cannot assume that the active s-parameters of all elements are the same, and it has to allow for sometimes significant differences, especially at the edges and corners of the array. This effect can be mitigated in the design by implementing a band of passive or dummy elements around the perimeter of the array, which would allow for the corporate feed of the active elements to assume a more equal input — but it would require a larger footprint for the array.

To solve larger geometries with a rigorous finite element approach, domain decomposition techniques offer the ability to use a distributed network of

compute nodes and leverage larger blocks of distributed memory. It decomposes a mesh representation of a model into a series of non-overlapping mesh domains that, when each matrix is individually solved with a traditional direct matrix solver, can collectively be used as a preconditioner for an iterative matrix solution to the full model. Previous work has developed a generalized scheme in which a given geometry for simulation is meshed in whole, and this resulting mesh is automatically subdivided into equal-sized mesh domains for balanced parallel computing. For antenna array solved with this general approach, the meshing processes can be quite expensive if the entire array needs to be meshed in general. However, in the approach discussed here, the repetitive nature of the geometry for an array is leveraged in that only a single unit cell of the array is meshed, with this mesh repeated along the array lattice to develop a set of mesh domains for the entire finite sized array. Each cell of this array has a unique solution depending on where it is located in the array, and the resulting full solution will take into account the effects along the edge of the array. Finally, the approach is efficient as each individual cell need not be solved at once in parallel, and further efficiencies are realized by leveraging the repeating nature of resulting matrices for certain cells residing in identical environments.

Domain Decomposition for Finite Arrays

This section describes an iterative non-overlapping domain decomposition method (DDM). To apply DDM, the original problem domain is first partitioned into N non-overlapping sub-domains:

$$\Omega = \Omega_1 \bigcup \Omega_2 \cdots \bigcup \Omega_N \tag{1}$$

Note that in antenna array problems, each antenna element will naturally be a domain. Subsequently, in ith domain , the boundary value problem (BVP) is written as

$$\nabla \times \frac{1}{\mu_{ri}} \nabla \times \mathbf{E}_i^{(n)} - k^2 \varepsilon_{ri} \mathbf{E}_i^{(n)} = -jk\eta \mathbf{J}_i, \quad \text{in } \Omega_i \tag{2}$$

$$\mathbf{j}_i^{(n)} - jk\mathbf{e}_i^{(n)} = -\mathbf{b}_i^{(n-1)}, \quad \text{On the interface } \Gamma_i \tag{3}$$

Note that an additional unknown carrying the physical meaning of electric current on the boundary, namely

is introduced on the domain interface, Γi. Also note that $\mathbf{E}_i$ is the electric

$$\mathbf{j}_i = \hat{\mathbf{n}}_i \times \frac{1}{\mu_{ri}} \nabla \times \mathbf{E}_i \tag{4}$$

field inside, Ω_i, $\mathbf{e}_i = \mathbf{n}_i \times \mathbf{E}_i \times \hat{\mathbf{n}}_i$ is the tangential electric field on boundary surface of Ω_i, and $\hat{\mathbf{n}}$ is unit outward normal of Ω_i. Superscript n stands for n^{th} iteration of alternating Schwarz algorithm. k, n, ε_{ri} and μ_{ri} are the free-space wave number, impedance, and relative permittivity and permeability of the medium in Ω_i, respectively. Equation (3) is generally known as the first-order Robin transmission condition. Its right-hand side is defined

$$\mathbf{b}_i^{(n-1)} = \mathbf{j}_{neig(i)}^{(n-1)} + jk\mathbf{e}_{neigh(i)}^{(n-1)} \tag{5}$$

in which *neigh(i)* indicates the neighboring domains of domain i. The above system corresponds to a Schwarz-type iteration scheme.

After the system of equations is appropriately tested through a Galerkin testing procedure, a matrix equation results

$$\mathbf{K}_i \mathbf{u}_i^{(n)} = \mathbf{y}_i + \mathbf{g}_i^{(n-1)}, \quad \forall i = 1, \cdots N, \tag{6}$$

in which $\mathbf{K}_i$, $\mathbf{y}_i$ are the system matrix and excitation vectors for Ω_i, respectively. $\mathbf{K}_i$ resembles a system matrix derived from traditional FEM using the first-order absorbing boundary condition (ABC) [2]. Solution vector and right-hand-side updating vector are given respectively as

$$\mathbf{u}_i^{(n)} = \left(\mathbf{E}_i^{(n)} \quad \mathbf{e}_i^{(n)} \quad \mathbf{j}_i^{(n)} \right)^T \tag{7}$$

$$\mathbf{g}_i^{(n-1)} = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{ij}^{je} & \mathbf{C}_{ij}^{jj} \end{pmatrix} \begin{pmatrix} \mathbf{0} \\ \mathbf{e}_j^{(n-1)} \\ \mathbf{j}_j^{(n-1)} \end{pmatrix} \tag{8}$$

Notice at each iteration that the right-hand side of (6) (c.f. (8)) is updated using only the information of surface unknowns. Thus, by introducing additional surface unknowns, the information in the entire volume of a domain is translated into information on the boundary surface, resulting in a tremendous reduction in memory requirements. Furthermore, for an array problem in which each array element is identical in terms of geometry and mesh, system matrix $\mathbf{K}_i$ and excitation vector $\mathbf{y}_i$ also remain the same for all domains, which further reduces memory requirements. The iterative process in (6) is trivially parallelizable as new right-hand-side (8) requires only information from the previous iteration. We remark that the formulation presented here allows both conformal and non-matching meshes across interfaces between adjacent subdomains. In our implementation, we have used matching surface meshes on master and slave boundaries (similar to those employed in the infinite array solver).

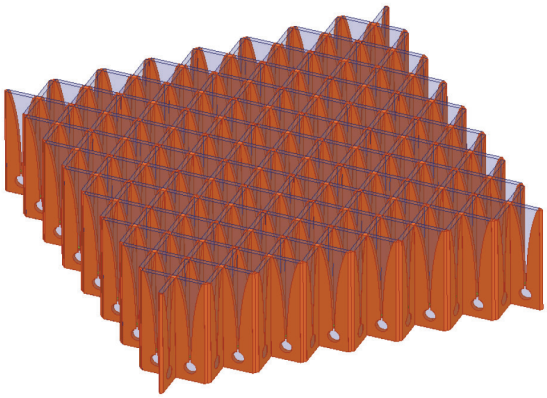

Figure 1. 256-element dual polarized slant Vivaldi array

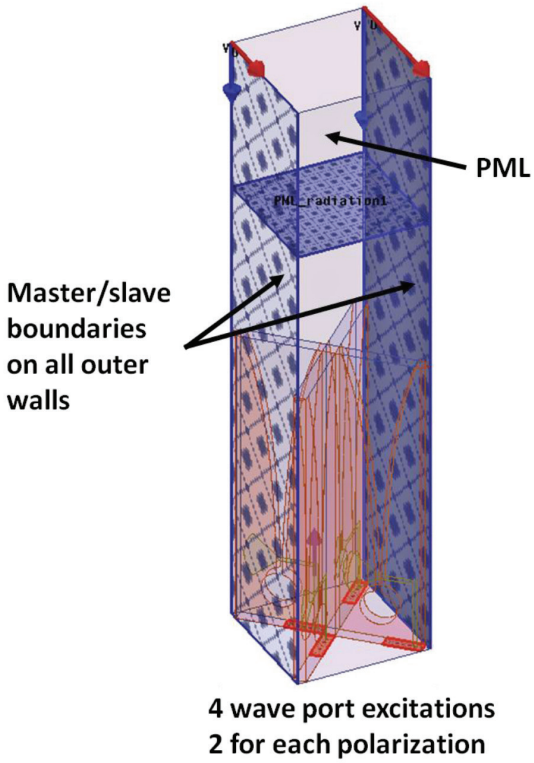

Figure 2. Unit cell for Vivaldi array

Dual Slant Polarized Vivaldi Array

Ultra-wideband radiators are ideal for modern radar systems, which present significant challenges in size, weight and cost for multifunctional systems. One such element is a dual polarized Vivaldi antenna with its large bandwidth and potential scan volume in an array configuration. Figure 1 depicts a 256-element array of dual-polarized Vivaldi antenna elements to be analyzed with this new technique and compared to an existing direct solver technique of an explicit representation of the full array.

Figure 2 shows the single unit cell used to simulate the array. The unit cell of the antenna array, including its automatically adapted mesh developed in a periodic boundary condition analysis, is virtually duplicated into the 256-element array geometry. Each unit cell is composed of four individual Vivaldi antenna elements arranged in 2X2 cross-polarized set. These elements are arranged in the unit cell at a 45 degree slant to provide a slant polarization to the overall 8X8 rectangular array lattice. The unit cell and its duplicates are each treated as individual domain solutions for a DDM solution to the entire finite antenna array. The electromagnetic interface between the individual cells is captured by a Robin transmission condition applied on the transverse faces of the cells (4) that, for the unit cell shown, correspond to the master/slave boundary pairs. By employing master slave pairs, a continuous conformal tetrahedral mesh is effectively maintained across this interface as an identical triangular mesh is enforced on parallel faces of the unit cell.

PMLs are used as absorbers for solution truncation to free space in the +z direction. Not depicted in Figures 1 or 2 but present in the solved model is a perimeter of bounding air padding cells around the array. These are implemented to space the radiating elements a sufficient distance from the virtual first-order radiation boundary condition sidewalls of the finite array.

The calculation of the element domains can be simplified by exploiting the repetitive nature of the elements matrices A, in the Ax=b calculation for each individual cell. However, not all cells of the array have the same matrix, as edge and corner elements reside in a different environment depending on how the elements of the perimeter are terminated, and, thus, corner elements and elements along an edge each have distinct A matrices. Ultimately, to describe a rectangular array, nine unique parent elements are required, one interior plus four edge and four corner. With this technique, the finite nature of the array, including edge effects, are captured since a unique set of fields are computed for all elements. Figure 3 shows results from the analysis in which the radiation from three distinct cells is overlayed on the same plot. The element pattern is highly dependent upon the element location in the finite-sized array; it is this type of effect that is missed with a traditional unit cell modeled with only periodic boundary conditions.

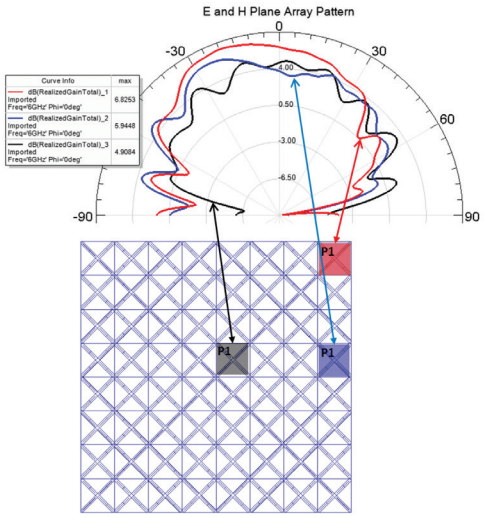

Figure 3. Far field patterns from three distinct cell types, P1, P2 and P3

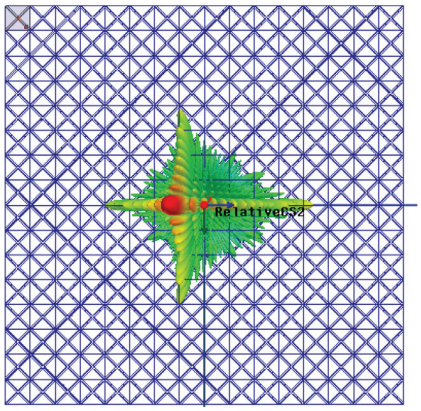

Figure 4. 1,024-element Vivaldi array with overlayed 3-D gain pattern

A direct simulation of the array required 211 GB RAM and over 122 hours to complete on a single 256 GB RAM machine. The DDM simulation required 69 GB total RAM and 12.5 hours computation time on a cluster of 24 machines, resulting in a simulation that overall required 67 percent less RAM and was 9.8 times faster compared to a full array direct solver solution. The equivalent machines or engines used in the DDM run need only enough shared RAM to accommodate the solution for a single unit cell, which in this example is approximately 3GB.

Very Large Example

The scalability of the technique on the preceding example was explored by simulating the same 2X2 slant polarized unit cell on a 16X16 array lattice. This provides for 1,024 total elements in the array. The simulation was run on 33 computes nodes. It is worthwhile to note here that these should not be physically separate but network compute nodes. A single physical machine can be defined as multiple compute nodes as long as the shared memory on this machine is equal to the number of nodes multiplied by the amount of RAM required for a single unit cell analysis. The resulting simulation was quite large, solving a matrix size of over 211 M matrix unknowns on 1,024 matrix right-hand sides or excitations. Most remarkable is that this simulation required only 111 GB of networked RAM in total to complete. Figure 4 shows a top-down image of the array with an overlayed 3-D far field gain patterned under a magnitude and phase tapering that provides a roughly 20 degree of azimuth main beam [3].

References

[1] L. Williams, M. Commens, S. Rousselle. Advances in Radar Simulation Design, *Microwave Journal*, Jan. 2012.

[2] J. Jin, *The Finite Element Method in Electromagnetics*, second ed. New York, New York: John Wiley & Sons, Inc., 2002.

[3] D. Thompson, "HFSS with HPC Options for Large Finite Antenna Array Design," ANSYS Usergroup Meeting, unpublished, Nov. 2011.

ANSYS, Inc.
Southpointe
275 Technology Drive
Canonsburg, PA 15317
U.S.A.
724.746.3304
ansysinfo@ansys.com

© 2013 ANSYS, Inc. All Rights Reserved.

ANSYS, Inc. is one of the world's leading engineering simulation software providers. Its technology has enabled customers to predict with accuracy that their product designs will thrive in the real world. The company offers a common platform of fully integrated multiphysics software tools designed to optimize product development processes for a wide range of industries, including aerospace, automotive, civil engineering, consumer products, chemical process, electronics, environmental, healthcare, marine, power, sports and others. Applied to design concept, final-stage testing, validation and trouble-shooting existing designs, software from ANSYS can significantly speed design and development times, reduce costs, and provide insight and understanding into product and process performance.
Visit www.ansys.com for more information.

Any and all ANSYS, Inc. brand, product, service and feature names, logos and slogans are registered trademarks or trademarks of ANSYS, Inc. or its subsidiaries in the United States or other countries. All other brand, product, service and feature names or trademarks are the property of their respective owners.