ACAsolver

Adaptive Cross Approximation (ACA) integration into existing MoM codes

Juan M. Rius

J. M. Tamayo, A. Heldring, J. Parrón and E. Ubeda

rius@tsc.upc.edu
AntennaLab,
Dept. Signal Theory and Communications
Universitat Politecnica de Catalunya (UPC),
Barcelona, Spain

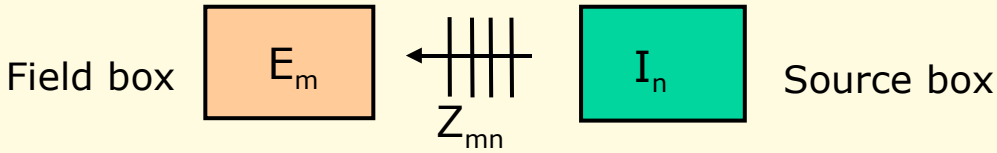
ACAsolver

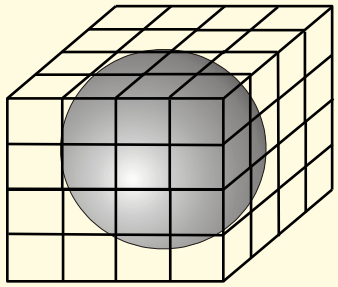
- ACAsolver is a fast iterative solver for compressible linear systems, with multilevel Adaptive Cross Approximation (ACA) matrix compression.
- ACA is based on the fact that interaction between far-field spatial subdomains has a reduced number of degrees of freedom.
- Although the solver has been developed for Electromagnetic Integral Equation problems discretized by Method of Moments (MoM), it can be applied to any physics or engineering linear system of equations with a compressible matrix that contains rank-deficient pseudo-submatrices.

ACAsolver is free-sorce software licensed under the GNU General public License

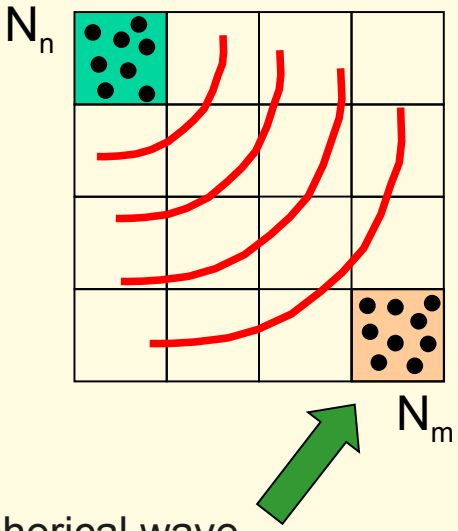
The degrees of freedom of EM field

- Subdivision of object in non-overlapping subdomains
- Impedance submatrices $[E_m] = [Z_{mn}][I_n]$





- If source and field boxes are **far enough** to each other: The field at the observation box is ≈ a **spherical wave**.

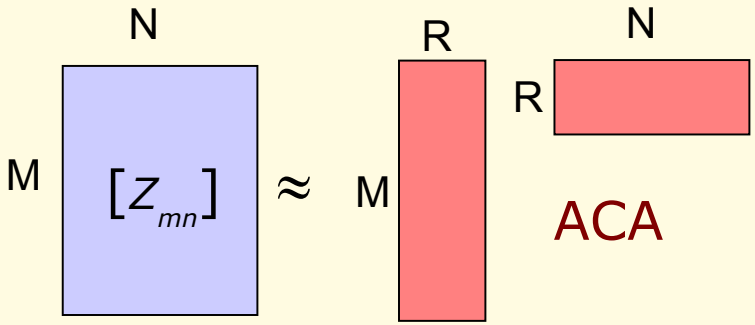


$[E_m]$ has a reduced number of Degrees of Freedom (DoF)

Adaptive Cross Approximation (ACA)

- ACA is -to our knowledge- the simplest Matrix Compression algorithm for rank-deficient far-field boxes:

$$[E_m] = [Z_{mn}][I_n] \approx [U_{mr}][V_{rn}][I_n]$$



The compressed matrices rank R is **much smaller** than the number of basis and testing functions in the source or field boxes, R << N and *R* << *M*

Operation count proportional to MR + NR = (M+N)R << MN

ACA algorithm

- ACA is adaptive:
  - Columns of U and rows of V are iteratively added until an error criterion is reached

$$\left[Z\right]^{MxN} \approx \left[\tilde{Z}\right]^{MxN} = \left[U\right]^{MxR}\left[V\right]^{RxN} = \sum_{r=1}^{R} u_r^{Mx1}\, v_r^{1xN}$$

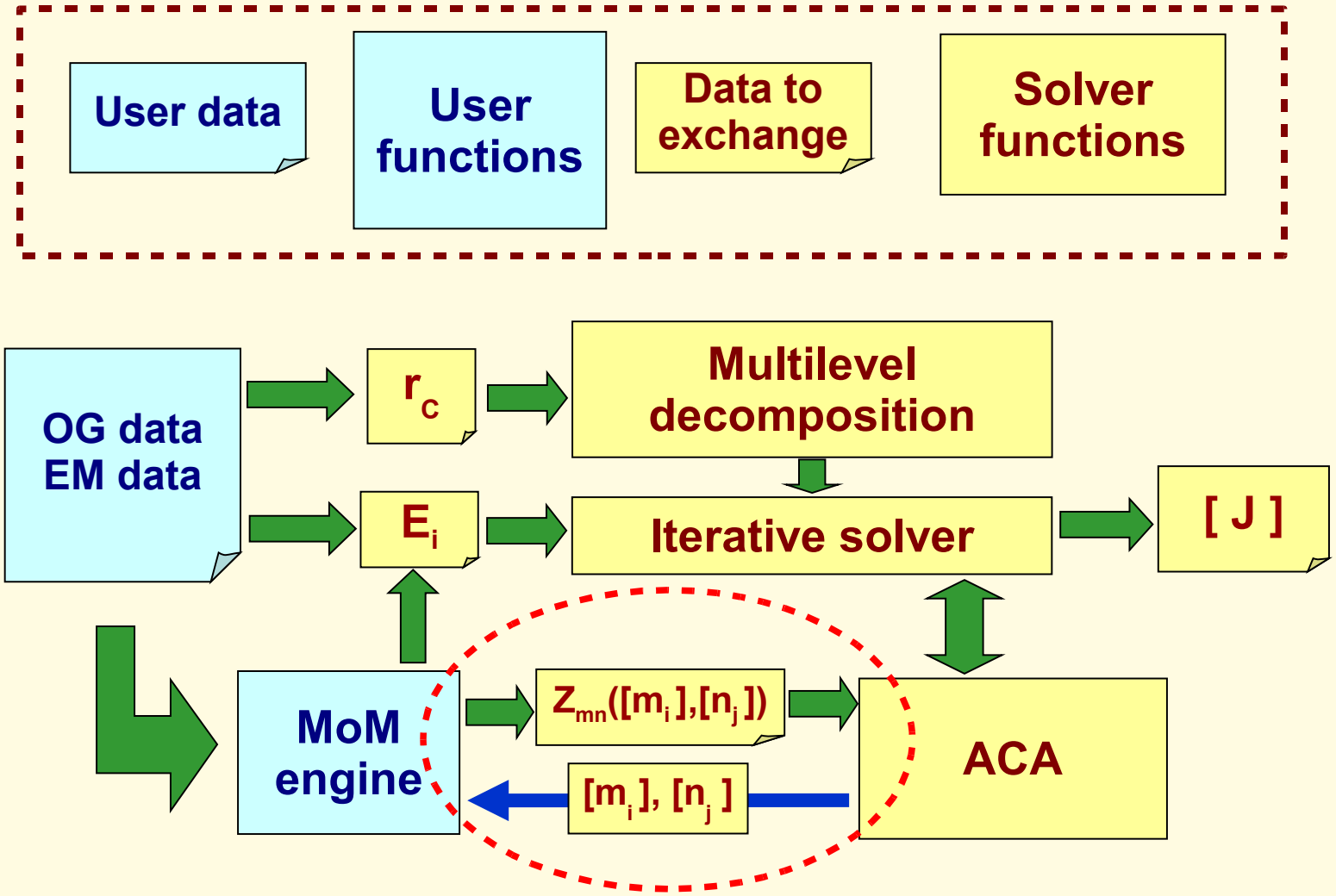
  - The r=1 to R iteration is stopped when

$$\left\|u_r\right\| \left\|v_r\right\| \leq \varepsilon \left\|\tilde{Z}\right\|^{(r)}$$

  - The approximation error is

$$\left[R\right] = \left[Z\right] - \left[\tilde{Z}\right] \qquad \left\|R\right\| \leq \varepsilon \left\|\tilde{Z}\right\|$$

ACA solver integration with MoM codes



User data

- Data needed by user functions (Z and Ei computation, etc.)
  - User data is encapsulated in two variables:

EM_data: Electromagnetic data
OG_data: Object geometry data

```
EM_data = user_set_EM_data()
OG_data = user_obj_geom_data(EM_data)
```

  - The solver passes user data as argument to user functions.
  - The solver does not know the contents and structure of user data.
  - If the solver needs some user data, it calls an interface function.

User functions

Must be created by the user to solve his particular linear system:

- Functions that encapsulate user data:

```
EM_data = user_set_EM_data()
OG_data = user_obj_geom_data(EM_data)
```

- Functions that compute the linear system:

```
Zmn = user_impedance(m, n, OG_data, EM_data)
Ei = user_compute_Ei(OG_data, EM_data)
```

- Functions that allow the solver to access user data:

```
[rcx,rcy,rcz,N] = user_get_OG_basis_center(OG_data)
```

- Functions to post-process data

```
user_plot_obj_current(J, OG_data, EM_data)
```

ACA multilevel iterative solver

USER FUNCTIONS USER DATA SOLVER

```
% Set electromagnetic data and object geometry
EM_data = user_set_EM_data();
OG_data = user_obj_geom_data(EM_data);
[rcx,rcy,rcz,N] = user_get_OG_basis_center(OG_data);

% Compute excitation vector (incident field)
Ei = user_compute_Ei(OG_data, EM_data);

% Compute compressed Z
[basis_func_boxes, L] = prepare_multilevel(rcx, ...);
Z_comp = multilevel_compress(...,OG_data, EM_data);

% Solve
J = gmres_Zcomp(Z_comp, Ei, tol, ...);

% Post-processing
user_plot_obj_current(J, OG_data, EM_data);
```

user_impedance()

- Submatrices of the impedance matrix:
  - The solver must call a user function that returns a submatrix of the impedance matrix, with row and column indices given by function arguments array m[] and array n[].

	n_1	n_2		n_j	
m_1		$Z(m_1,n_2)$			
m_2					
m_i					

```
Zmn = user_impedance(m,   n,   OG_data, EM_data);
```

Array of row indices

Array of col indices

User data

The future

- **Software integration** between University research groups is the key for successful analysis of very complex antenna configurations, that cannot be handled yet by commercial CEM packages.
- The integration effort started in Antennas Center of Excellence (ACE) network can be the seed of a future Linux-like **open library** of CEM engines and solvers.
- The programming language proposed is python-scipy:
  - Python: Very high level language (even higher than MATLAB)
  - Scipy: High level matrix operations (like MATLAB)
  - Free (unlike MATLAB)
  - Very easy call to FORTRAN and C/C++ functions (easier than MATLAB)

ACA References

- M. Bebendorf, “Approximation of boundary element matrices”, *Numer. Math.*, vol. 86, no. 4, pp. 565–589, 2000.
- K. Zhao, M. Vouvakis, and J.-F. Lee, “The adaptive cross approximation algorithm for accelerated method of moment computations of emc problems”, *IEEE Trans. on Electromagnetic Compatibility*, vol. 47, no. 4, pp. 763–773, Nov. 2005.