DIRECT VS. ITERATIVE METHODS FOR NUMERICAL EM

<table>
<thead>
<tr>
<th>CRITERIA</th>
<th>DIRECT</th>
<th>ITERATIVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONVERGENCE</td>
<td>BEST CHOICE</td>
<td>-</td>
</tr>
<tr>
<td>MEMORY (DENSE/SPARSE)</td>
<td>-</td>
<td>BEST CHOICE</td>
</tr>
<tr>
<td>PARALLEL IMPLEMENTATION</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ERROR</td>
<td>BEST CHOICE</td>
<td>-</td>
</tr>
<tr>
<td>PROCESSING TIME</td>
<td>-</td>
<td>BEST CHOICE</td>
</tr>
</tbody>
</table>

SPARSE MATRICES

Matrices derived from PDE's have a number of non-zero elements proportional to the matrix size (M), and a number of total elements of M x M.

Example: Model of human at 1mm resolution, 1800 x 600 x 300 mm = $3.24 \times 10^6$ elements (equations/unknowns). Assuming the coefficient matrix is type float, and each equation relates to 7 elements,

$$3.24 \times 10^6 \times 7 \times 4 = \boxed{9.11 \text{ GB}}$$

of RAM to store non-zero elements.

But if all of the matrix elements need to be stored,

$$M^2 = 1.05 \times 10^{14}$$

And if each coefficient is a float,

$$1.05 \times 10^{17} \times 4 = \boxed{4.2 \times 10^8 \text{ GB}}$$

of storage are needed.
For real EM numerical problems, there are free quality libraries available:

**BLAS**: Optimized vector/matrix library for dense matrices.

**LAPACK**: Dense system linear algebra library

**IML++**: Iterating solver library (Krylov sub-space), works with SparseLib++

**SparseLib++**: Sparse matrix storage library.

For smaller problems, MATLAB has support for iterating and direct solvers, as well as sparse matrix storage.

Refer to MATLAB literature and example code provided in this course.