Something different - Caching Applied to Calculation of Impedance Matrix Elements

A. A. Lysko[1]

***Abstract* – This paper introduces a new method generally termed memoization, to accelerate filling in the impedance matrix, e.g. in the method of moments (MoM). The memoization stores records for recently computed matrix elements in a cache, and, when an impedance matrix element with the same properties is requested, it fetches the answer quickly, from the cache instead of computing it anew. Under ideal circumstances assuming uniformity, the method can reduce the computational burden from O(N^2) to O(N). The method requires only minor modifications of an existing code, and has been realized in Matlab. The tests confirmed validity of the method and showed doubling the speed of filling the impedance matrix in.**

1 INTRODUCTION

The method of moments has been in active use in electromagnetics since it has been popularized by Harrington [1], and is still one of the predominant methods in much of the commercial and amateur software [2]-[7]. The method involves several steps:

i. The geometry of the problem is assigned a mesh with associated approximating basis functions to interpolate the unknown (which is often current), using some algorithm; the basis functions have unknown constant multipliers;
ii. An impedance matrix describing the coupling product between the mesh elements / basis functions (and possibly weighted with certain test functions), is computed, element by element; here, the symmetries in the geometry can be used to accelerate this process;
iii. Together with an excitation vector computed, the impedance matrix form a system of linear equations, which is solved to find the unknowns (multipliers);
iv. Using the determined values of the multipliers, the approximating functions are used any required parameters, such as input impedance or gain pattern etc.

The method is relatively straightforward but, especially for small to medium matrices, requires spending time on filling in the impedance matrix in (ii). Moreover, this process is often challenging to program, as it requires special techniques [8]-[13] with a very careful treatment of the integrands, which are near singular, especially for the main diagonal elements. The symmetries can be used to reduce the number of computations necessary, but may require one to perform nontrivial manipulations with indexing.

This paper proposes to take advantage of memoization, a technique based on using a cache memory, which is a method well known and widely applied everyday in various hardware and software, from computer processors and hard drives to network routers and web browsers [14]-[20]. The term *memoization* was introduced by the computer scientist Donald Michie [21] in 1968. This term refers to storing the results of computed functions in a fast storage/memory, and being able to retrieve the result fast from this quick memory, rather than re-computing it again.

The method can be explained, in terms of the computing the impedance matrix for the method of moments, or a similar method, such as boundary element method (BEM) [22], with the help of the flowchart shown in Figure 1.

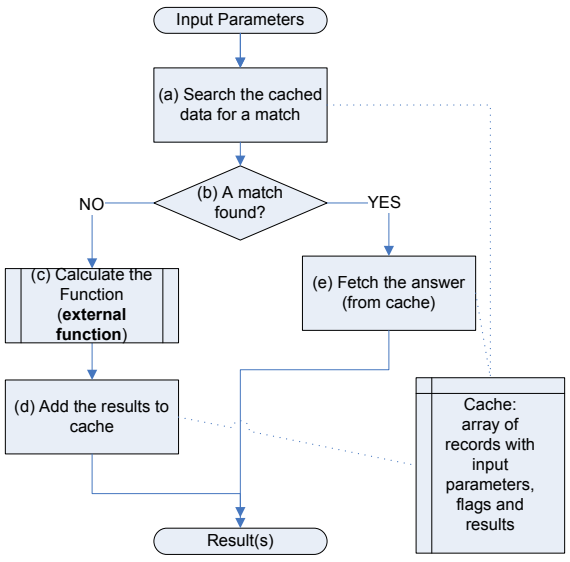

Figure 1: Operation of a cache.

The input parameters for the memoization will normally be the same inputs as for a routine

[1] Meraka Institute, Council for Scientific and Indistrial Research (CSIR), Pretoria, 0001, South Africa,
e-mail: alysko@csir.co.za, tel.: +27 12 841 4609, fax: +27 12 841 4720.

computing the impedance matrix elements. The cache routine will then try to find whether the cache has an entry matching these parameters: (a)→(b). If the computations just started, the entry will not be found (branch 'NO' in the figure), and the function for calculating the corresponding impedance matrix element must be called (c). Once the result is calculated, this result, together with the input parameters used, can then be stored in the cache, (d). The next time the cache routine is called with input parameters matching one of the records stored in the cache, the search (a) will find the match (b) and take the branch 'YES' of the flowchart, take the result straight from the cache instead of re-computing it.

2 ADVANTAGES AND CHALLENGES

Applying memoization routine normally means working with memory and indexing. Only little actual computations may be required and only in special cases, e.g. if one wants to match the records using certain advanced criteria, instead of a simple one to one comparison of binary numbers.

However, searching through the cache is an overhead, and usually is the main challenge in the method. A cache too small, will not store enough values and will force computations too frequently. If the cache stores too many entries, then the 'misses', when a matching record is not in the cache, will lead to excessive delays before needing to compute or even before finding the value. In both cases, it might be faster to compute the function directly rather than use the cache (unless there are special circumstances such as having a fast memory and a slow floating point arithmetic unit simultaneously). Normally, there is however a size of the cache optimum to the problem being solved, subject to the constraint discussed next.

In addition, as any other cache based method [14]-[20], the memoization needs a sufficient degree of repetition in the data it stores – here, in the impedance matrix. There are solutions that may be offered to improve the repetition rate:

a) Usage of dissimilar basis functions can be accommodated by either having several independent caches, e.g. one for each basis function, or adding some indexes associated with these basis functions to each record stored in the cache (to be able to identify the basis function).
b) Mesh can be made more uniform, or minimizing the number of unique-sized or shaped elements.
c) The order the matrix is computed can be set to maximize the probability of finding the same elements. For example, the most indexing schemes generate impedance matrices having elements which are the same or similar along diagonals. Thus, filling the impedance matrix in in this particular order may substantially improve the repeatability.
d) Alternatively, in order to reduce the required size of the cache, the memoization may be applied selectively, e.g. only to the main diagonal, or clearing the cache before starting each new diagonal.

The maximum efficiency can be achieved for instance for a uniformly meshed dipole or monopole. This has a direct connection to the traditional MoM implementations, where such scenarios lead to the impedance matrix being a Toeplitz matrix, i.e. being determined by only one column with N elements. Similarly, under such circumstances, the cache should need to compute only N elements and can fill the rest of the matrix in using these stored values. Based on this, the best possible acceleration expected from using memoization is in reducing the computational complexity from $O(N^2)$ to $O(N)$.

The other strength of the method comes when one needs to compute the impedance matrix elements very accurately. As the method is blind to the function(s) used to compute the impedance matrix elements, the computational effort used by the memoization (mostly, for the search) will remain the same regardless of the accuracy of the function to be memoized. Thus, the effect of memoization will be more pronounced for higher accuracy requirements.

2.1 Optimizing the order of elements in the cache

An important aspect of the method is the way the elements are arranged and/or prioritized. This is a field of research on its own, e.g. [23]-[25].

3 IMPLEMENTATION AND TESTING

The cache has been implemented in a sub-domain version of the method of moments, based on [23]-[28]. The practical realization of the memoization done by the author is shown in Figure 2. The method uses an opportunistic approach to finding the elements - it is assumed that the most recently found elements are the most likely to be found soon again.

This algorithm has been applied to an example of a dipole composed of 4 wire segments of equal length. This dipole required 3 unknowns, so the size of the impedance matrix was 3 x 3. Each of the 9 elements of this matrix included 4 partial impedances [23], leading to 36 double integrals that needs to be computed to fill in the matrix.

Table 1 summarizes the results of the tests:

i) When there is no cache, everything is done according to a standard method of moments.
ii) As the cache is enabled, but has no actual storage (size=0), as expected, it adds some

20% of an overhead to the computations. From this, one can also conclude that the cache routine tool, on average, 1/5-th of the time needed to compute a matrix element. However, one may need to be careful with such a generalization, as the main diagonal elements must have taken significantly longer than the off-diagonal elements. At this point, it is already possible to forecast that the selective memoization may improve the speed up.

iii) When the cache is fully enabled and has plenty of storage, it permitted to accelerate the computations by the factors of 2.

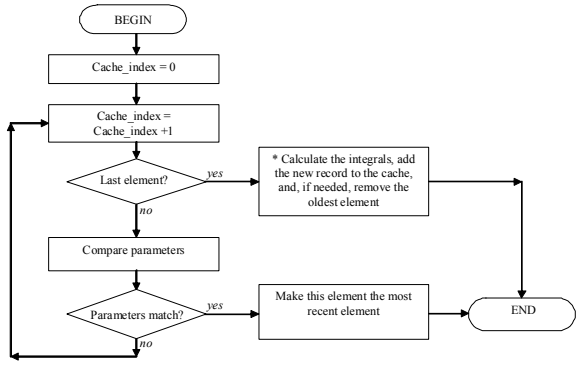

Figure 2: Diagram showing a practical realization for the cache used in this paper.

Item	No cache used	Cache *ON*, Size=0	Cache *ON*, Size=64
Total number of times function *Cache* called	0	74	58
… including *Init* and *SetSize*		2	2
… including *FindElement*		none[2]	35+1 (first time no search was done)
… including *AddNewElement*		36	20
Total relative time to fill in impedance matrix,	8.25 ±0.02	10.66 ±0.02	4.71 ±0.04

Table 1: Profiling the execution of the impedance matrix filling in process.

4 KNOWN FURTHER POSSIBILITIES

The patent [29] takes the use of cache to yet another dimension. It discusses the possibility to interpolate between the elements which are in the cache and close enough to the point of interest, which is otherwise not present in the cache directly.

5 CONCLUSION

A new method for accelerating filling in the impedance matrix, memoization, has been introduced. The various advantages and disadvantages have been discussed, with examples of possible solutions. A practical realization of the memoization has confirmed the practical value of the method, and possibility to accelerate the process of filling in impedance matrix manifold. The test demonstrated reduction of the fill-in time by a factor of two. All of this is achieved with just few lines of code, and very minor modifications to an existing method of moments program. It is expected that the method can also be applied to other numerical techniques relying on dense matrices.

References

[1] R.F. Harrington, "Field Computation by Moment Methods", Macmillan, 1968. Reprinted by IEEE Press, 1993.

[2] G.J. Burke, and A.J. Poggio, "Numerical electromagnetics Code (NEC) – Method of Moments. Part II: Program Description – Code." Report UCID – 18834, Lawrence Livermore Laboratory, January 1981.

[3] C.A. Balanis, "Antenna Theory: Analysis and Design", 2nd Ed., 1997, 960 pages.

[4] Windows: Analysis of Wire Antennas and Scatterers", Artech House, 1995.

[5] A.R. Djordjevic, M.B. Bazdar, T.K. Sarkar, and R.F. Harrington, "AWAS for Windows Version 2.0: Analysis of Wire Antennas and Scatterers, Software and User's Manual", Artech House, 2002.

[6] S.N. Makarov, "Antenna an EM Modeling with Matlab", John Wiley & Sons, 2002, 288 pages.

[7] B.M. Kolundzija et al., "WIPL-D Pro v6.1: 3D Electromagnetic Solver, Professional Edition. User's Manual", WIPL-D d.o.o., 2006.

[8] M.G. Duffy, "Quadrature over a pyramid or cube of integrands with a singularity at a vertex." SIAM Journal on Numerical Analysis, 19(6), 1982, pp. 1260-1262.

[9] B.M. Kolundzija, and B.D. Popović, "A New, Rapid and Accurate Method for Evaluation of Potential Integrals in Thin-Wire Antenna Problems," Proc. 5th ICAP, York, Pt. I, 1987, pp. 35-38.

[10] P. Yla-Oijala, and M. Taskinen, "Calculation of CFIE impedance matrix elements with RWG and n times RWG

[2] The subroutine checks whether the cache has at least 1 element before trying to search.

functions." -IEEE Transactions on Antennas and Propagation, Vol. 51, No 8, Aug. 2003, pp. 1837 – 1846.

[11] S. Caorsi, D. Moreno, and F. Sidoti, "Theoretical and numerical treatment of surface integrals involving the free-space Green's function." IEEE Trans. on Antennas and Propagation, Vol. 41, no. 9, 1993, pp. 1296-1301.

[12] S. Järvenpää, M. Taskinen, and P. Ylä-Oijala, "Singularity extraction technique for integral equation methods with higher order basis functions on plane triangles and tetrahedral", Intl. J. Numer. Meth. Eng., Vol. 58, 2003, pp. 1149-1165

[13] W.C. Chew, J. Jin, E. Michielssen, and J. Song, "Fast and Efficient Algorithms in Computational Electromagnetics", Artech House, 2001, 931 pages.

[14] J. Shim, P. Scheuermann, and R. Vingralek, "Proxy Cache Algorithms: Design, Implementation, and Performance", IEEE Transactions on Knowledge and Data Engineering, Vol. 11, No. 4, July/August 1999.

[15] S.E. Richardson, "Caching Function Results: Faster Arithmetic by Avoiding Unnecessary Computation", SMLI TR-92-1, Sun Microsystems, September 1992.

[16] M. Abadi, B. Lampson, and J.J. Levy, "Analysis and Caching of Dependencies", ACM SIGPLAN Notices archive, Vol. 31, June 1996.

[17] A. Heydon, R. Levin, and Y. Yu, "Caching Function Calls Using Precise Dependencies", in Proc. of ACM SIGPLAN 2000.

[18] U.A. Acar, G.E. Blelloch, and R. Harper, "Selective Memoization", ACM 2003

[19] S.-E. Yoon, P. Lindstrom, V. Pascucci, D. Manocha, "Cache-Oblivious Mesh Layouts". ACM Transactions on Graphics, SIGGRAPH 2005, April 28, 2005.

[20] J. Křivánek, J. Žára, "Radiance Caching for Fast Global Illumination with Arbitrary BRDFs", In Workshop, CTU Prague, 2004.

[21] D. Michie, "Memo Functions and Machine Learning," Nature, No. 218, 1968, pp. 19–22.

[22] D. Poljak, "Advanced Modeling in Computational Electromagnetic Compatibility", Wiley-Interscience, 2007 496 pages.

[23] R.P. Cook, C.J. Linn, J.L. Linn, T.M. Walker, "Cache memories: A tutorial and survey of current research directions", Proceedings of the ACM '82 conference, 1982, pages 99 – 110.

[24] M. Frigo, C.E. Leiserson, H. Prokop, S. Ramachandran, "Cache-oblivious algorithms", 40th Annual Symposium on Foundations of Computer Science, 1999.

[25] M. Bader, C. Zenger, "Cache oblivious matrix multiplication using an elementordering based on a Peano curve", Linear Algebra and its Applications, Vol. 417, Nrs 2–3, Sep 2006, pages 301–313.

[26] A.A. Lysko, "On Multiple Domain Basis Functions and Their Application to Wire Radiators", Doctoral thesis, NTNU-trykk, Norwegian University of Science and Technology, 2010, 291 pages. Electronic file available at http://urn.kb.se/resolve?urn=urn:nbn:no:ntnu:diva-7872

[27] B.M. Kolundzija, and B.D. Popovic, "Entire Domain Galerkin Method for Analysis of Generalised Wire Antennas and Scatterers," IEE Proc.-H, Vol. 139, No 1, Feb. 1992, pp. 13-18.

[28] A.A. Lysko, "New MoM code incorporating multiple domain basis functions", General Assembly and Scientific Symposium of the International Union of Radio Science (URSI GASS 2011), Istanbul, Turkey, 13-20 August 2011.

[29] U.S. patent No 6553394, "Continuous memoization".