Limits for Computational Electromagnetics Codes Imposed by Computer Architecture

Jürgen v. Hagen, Werner Wiesbeck
Institut für Höchstfrequenztechnik und Elektronik
Universität Karlsruhe, Kaiserstr. 12, D - 76128 Karlsruhe, Germany,
Phone: +49 721 608 76 76, Fax: +49 721 69 18 65
email: vonhagen@ihe.etec.uni-karlsruhe.de

Abstract— **The algorithmic complexity of the innermost loops that determine the complexity of algorithms in computational electromagnetics (CEM) codes are analyzed according to their operation count and the impact of underlying computer hardware. As memory chips are much slower than arithmetic processors, codes that involve a high data movement compared to the number of arithmetic operations are executed comparatively slower. Hence, matrix-matrix multiplications are much faster than matrix-vector multiplications. It is seen that it is not sufficient to compare only the complexity, but also the actual performance of algorithms to judge on faster execution. Implications involve FDTD loops, *LU* factorizations, and iterative solvers for dense matrices. Run times on two reference platforms, namely an Athlon 900** MHz **and an HP PA 8600 processor, verify the findings.**

I. Introduction

Most codes for computational electromagnetics, especially frequency domain methods, involve the solution of a linear system of equations. The efficient solution of linear systems of equations is, hence, one important part in improving the efficiency of computational electromagnetics. Depending on the method, the linear systems involve dense or sparse, real-valued or complex-valued, symmetric and non-symmetric matrices.

One of the most important methods, namely the Method of Moments, involves the solution of a dense linear system of equations. It is generally complex-valued, non-symmetric and non-hermitian. The solution is commonly obtained by a direct method, *i.e.*, an *LU* factorization [1]. A certain interest has appeared to solve systems with dense matrices by iterative methods [2] that are usually applied to sparse matrices. The solution time of either method depends on the algorithm, the arithmetic operations, the data of the matrices and vectors, and also the computer architecture which the code runs on. Iterative methods are also affected by the number of iterations given by their convergence properties.

In this paper, the influence of the computer architecture on the execution speed of innermost loops is reviewed. For this, today's computer architectures are briefly reviewed. Then, the complexities of iterative methods, direct methods, and FDTD loops are reviewed with taking into account a hypothetical reference computer architecture. Two specimens of present day computers, namely a Linux computer equipped with a 900 MHz Athlon CPU and a HP Risc PA 8600 processor at 552 MHz, provide measured data on run-times. Implications on CEM codes and their performance are finally drawn.

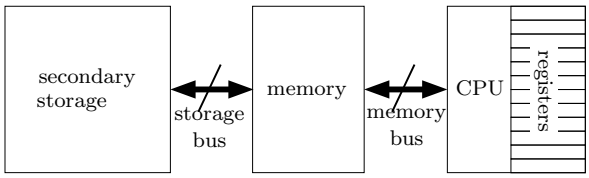

Fig. 1. Schematic computer system with CPU and memory

II. Computer Architecture

Figure 1 sketches the most important functional units of a computer with only those parts shown that are important for CEM codes. The computations are carried out in the registers within the central processing unit (CPU). Before usage, the data is stored in the secondary, usually disk, storage, and must then be transferred for execution into the RAM. This is done only once if the code and the data fit entirely into the memory, this part then influences run-times only secondarily. Later, the data is loaded into the registers on the CPU for arithmetic operations. The number and complexity of registers determines the number of operations that can take place simultaneously. This determines the theoretical performance expressed in FLOP(floating point operations per second). For the data transfer to take place, the band widths of the disk bus and mainly the memory bus determine the rate of data transfer. The performance of the busses is given as their band width in amount of data per second (*e.g.*, 1 Gbyte/s).

For a given hardware and a given computation, the FLOP number is not always defining the computational speed. Processors in workstations and PCs allow more floating-point operations per cycle than memory operations as different registers can hold and act on different data. This is due to the fact that the arithmetic reg-

isters in modern processors (up to 1.5 GHz for Athlon processors) are much faster than memory chips (about 120 MHz). If the data cannot be retrieved from memory sufficiently quickly, the CPU waits for the memory to deliver the data. Hence, the memory band-width is more limiting than the floating-point capabilities. The maximum performance Perf that is attainable on memory-bound computations (i.e., the CPU awaits memory operations) is approximated by

$$\text{Perf} = \frac{\begin{pmatrix}\text{floating}\\ \text{point}\\ \text{operations}\end{pmatrix} \cdot \begin{pmatrix}\text{memory}\\ \text{band-}\\ \text{width}\end{pmatrix}}{\begin{pmatrix}\text{data retrieved}\\ \text{from memory}\end{pmatrix}} \tag{1}$$

Here, we define a floating-point operation as an add or a multiply as both operations are carried out in about the same time. This definition is in accordance with the one in [1]. Finally, as arithmetic registers usually hold double precision floating point numbers, the execution speed of floating-point operations for single and double precision real numbers is about the same.

Modern processors contain between the processor and the main memory smaller and faster so-called cache memories. The access time for cache memories is between the access time of the main memory and the processor clock. However, the size of cache memories is in the range of only a few Mbyte, for larger data sizes not all the data can be held in the cache memory. Due to this fact, the band width of the main memory is the most important limiting factor.

Due to the hierarchical memory layout and a possible re-use of data already loaded in the faster cache memories, the actual performance obtained with optimized subroutines can be better than the above theoretical estimate. Optimized libraries are available for a variety of platforms and processors. An auto-optimizing library ATLAS [3] attains performance values that surpass usual implementations. For actual run-times, two reference platforms are chosen that should be representative for a wide range of platforms currently used. The first reference platform is a Linux based AMD Athlon processor clocked at 900 MHz. It includes a small cache of 125 kbyte that is clocked at the processor speed. A secondary level cache memory amounts to 512 kbyte and is clocked at half the CPU clock speed. The computations carried out are obtained when the above cited Atlas library is used. A second platform, an HP PA 8600 processor with HP-UX 11.00, has a primary cache of 1.0 Mbyte accessible at processor speed. The bus band width is 2 Gbyte/s. The run-times are reported when using the vendor supplied library mlib.

TABLE I
RUN-TIMES FOR 100 DOUBLE PRECISION MATRIX-VECTOR MULTIPLICATIONS ON TWO REFERENCE PLATFORMS.

Matrix Size N	Mbyte	Athlon 900 MHz	HP PA 8600
1000	15.4	1.47 s	0.61 s
1500	34.3	3.30 s	1.40 s
2000	61.1	5.94 s	2.42 s
2500	95.4	9.30 s	3.78 s
3000	137.3	13.49 s	5.67 s
Resulting Perf_2		133 MFLOP	317 MFLOP

III. Maximum Performance for Matrix-Vector Multiply

The matrix-vector multiply $y = Ax$ for a square $N \times N$ matrix A spells out

$$\begin{aligned} y(1{:}N) &= 0 \\ y(1{:}N) &= y(1{:}N) + \sum A(1{:}N, i)\, x(i) \quad \text{for } i = 1{:}N \end{aligned} \tag{2}$$

The total number of operations is $2N^2$. The inner loop $A(1{:}N, i)\, x(i)$ requires per element three memory operations (load $y(\cdot)$ and $A(\cdot, i)$, store result $y(\cdot)$) and two floating-point operations (multiply $x(i)$ and $A(\cdot, i)$, add to $y(\cdot)$).

On a hardware that is memory-bound by a bus band width of 2 Gbyte/s, the maximum attainable performance is

$$\text{Perf}_2 = \frac{2 \text{ FLOP} \cdot 2 \text{ Gbyte/s}}{3 \cdot 8 \text{ byte}} \approx 170 \text{ MFLOP} \tag{3}$$

if each datum occupies 8 byte. Standardized subroutines available as the BLAS subroutines [4] that compute matrix-vector products are called Level 2 subroutines. We hence define the above performance as level-2 performance Perf_2. The run-times for matrix-vector multiplications of square matrices with double precision real elements and size $N \times N$ are reported in Table I. The last line in the table concludes the run-times in a performance number given as floating-point operations per second FLOP.

IV. Maximum Performance for Matrix-Matrix Multiply

A matrix-matrix multiply of $N \times N$ matrices $C = AB$ is computed, *e.g.*, by

$$C(i, j) = \sum A(i, 1{:}N)\, B(1{:}N, j) \quad \text{for } i, j = 1{:}N \tag{4}$$

TABLE II
RUN-TIMES FOR A DOUBLE PRECISION MATRIX-MATRIX MULTIPLICATIONS ON TWO REFERENCE PLATFORMS.

Matrix Size N	Mbyte	Athlon 900 MHz	HP PA 8600
1000	15.4	2.30 s	1.27 s
1500	34.3	7.77 s	4.39 s
2000	61.1	19.05 s	10.05 s
2500	95.4	38.18 s	19.93 s
3000	137.3	74.11 s	34.41 s
Resulting Perf_3		364 MFLOP	784 MFLOP

The total number of floating-point operations is now $2N^3$. The inner loop $A(i,1:N)B(1:N,j)$ requires per element only two memory operations (load $A(i,\cdot)$ and $B(\cdot,j)$) and again two floating-point operations (multiply $A(i,\cdot)$ with $B(\cdot,j)$, add to a running sum). The result of the loop is the running sum that is stored at the end of the loop. On the same hardware and the same data size as above, the maximum attainable performance is

$$\frac{2 \text{ FLOP} \cdot 2 \text{ Gbyte/s}}{2 \cdot 8 \text{ byte}} \approx 256 \text{ MFLOP} \tag{5}$$

According to the name of BLAS Level 3 [5] of matrix-matrix subroutines, we identify the above performance as level 3 performance Perf_3. For memory-bound hardware, the matrix-matrix multiply always tops the matrix-vector multiply performance as more floating-point operations are needed for the same data. The ratio of Perf_2 to Perf_3 indicates the higher efficiency of level-3 operations. This ratio is 0.66 for the above estimates. Actual run-times are reported in table II. The performance measured for the different platforms is higher than the above estimate due to the influence of cache memories. For both platforms, the level-3 performance is more than double the level-2 performance.

V. Maximum Performance for FDTD Loops

An update per field component (e.g., E_x) in an FDTD computation on a three-dimensional grid of size $N_x \times N_y \times N_z$ conforms to the following equation

$$\begin{aligned} E_x^{t+1}(x,y,z) &= C_1(z) \cdot E_x^t(x,y,z) \\ &+ \left(\frac{H_z^{t+1/2}(x,y,z-1/2) - H_z^{t+1/2}(x,y,z)}{\Delta y \, \mu(z)} \right. \\ &\left. + \frac{(H_y^{t+1/2}(x,y,z) - H_y^{t+1/2}(x,y-1,z)}{\Delta z \, \mu(y)} \right) \cdot C_4(z) \end{aligned} \tag{6}$$

For the update in each cell, eight floating-point operations take place, whereas ten memory operations are carried out. With the same restrictions as above, the performance of an FDTD loop is hence

$$\frac{8 \text{ FLOP} \cdot 2 \text{ Gbyte/s}}{10 \cdot 8 \text{ byte}} \approx 205 \text{ MFLOP} \tag{7}$$

which is slightly lower than the one for the matrix-matrix multiply.

VI. Implications

For high-power computers the computational performance is often memory-bound, and not floating-point bound. We have shown that for memory-bound architectures, *i.e.* all present day platforms, the performance of matrix-vector multiplications is always lower than the one of matrix-matrix multiplications.

Solving a linear system of equations of size N by iterative solvers requires subsequent matrix-vector multiplications. Each matrix-vector multiplication requires $2N^2$ operations executed with the above level-2 performance Perf_2 resulting in a run-time of

$$T_{\text{MV}} = \frac{2N^2}{\text{Perf}_2} \tag{8}$$

for one matrix-vector multiplication, and

$$T_{\text{iter}} = 2n_{\text{iter}} \frac{2N^2}{\text{Perf}_2} \tag{9}$$

for a solution of a linear system of equations after n_{iter} iterations with two matrix-vector multiplications each.

In contrast, a LU factorization requires $2N^3/3$ operations [1], however executed with a level-3 performance Perf_3 resulting in a total run-time of

$$T_{LU} = \frac{2N^3}{3\text{Perf}_3} \tag{10}$$

Comparing the execution time of the LU factorization (10) with the one of an iterative solution (9), and deriving the number of iterations n_{iter} for which the iterative algorithm is faster, one obtains

$$\frac{2N^3/3}{\text{Perf}_3} > n_{\text{iter}} \frac{4N^2}{\text{Perf}_2} \tag{11}$$

which yields

$$n_{\text{iter}} < \frac{N}{6} \frac{\text{Perf}_2}{\text{Perf}_3} \tag{12}$$

For the hypothetical level-3 to level-2 ratio of about 0.66, the number of iterations is $n_{\text{iter}} < 0.11N$, and not only $n_{\text{iter}} < N/3$ as often cited in literature. Even worse,

TABLE III
RUN-TIMES FOR DOUBLE PRECISION *LU* FACTORIZATIONS ON TWO REFERENCE PLATFORMS.

Matrix Size N	Mbyte	Athlon 900 MHz	HP PA 8600
1000	15.4	0.89 s	0.59 s
1500	34.3	2.88 s	1.9 s
2000	61.1	6.74 s	4.36 s
2500	95.4	12.90 s	8.38 s
3000	137.3	22.11 s	14.98 s
resulting Perf_{LU}		815 MFLOP	1.2 GFLOP

the above reference platform HP PA 8600 has a ratio of $\text{Perf}_2/\text{Perf}_3 \approx 0.2$. Hence $n_{\text{iter}} < 0.034N << N/3$.

On the two test platforms, the LAPACK *LU* factorization routine dgetrf is executed according to the run-times in Table III. Due to an efficient reuse of cache memory [6], [7], [8], the performance Perf_{LU} is even higher than Perf_{MM}. The number of iterations for an iterative method to be faster than the direct method is now $0.022N$.

VII. CONCLUSIONS

The actual execution speed of computational electromagnetics codes is not only affected by the complexity of algorithms, but also by the computer hardware. It is shown that on memory-bound platforms, *i.e.*, any modern computer platform, the performance of matrix-matrix multiplications is always higher than for matrix-vector multiplications. This implies that for dense matrices, iterative solvers must converge with very few iterations for an iterative algorithm to be faster than direct methods. Furthermore, FDTD loops do not attain the raw performance of *LU* factorizations on mermory-bound platforms.

REFERENCES

[1] G. H. Golub and C. F. van Loan, *Matrix Computations*. Baltimore and London: The Johns Hopkins University Press, 1996.

[2] Y. Saad, *Iterative Methods for Sparse Linear Systems*. Boston: PWS Publishing Company, 1996.

[3] R. C. Whaley, A. Petitet, and J. J. Dongarra, "Automated empirical optimization of software and the atlas project," *To appear in Parallel Computing*, 2001. Also avalable as University of Tennessee LAPACK Working Note #147, UT-CS-00-448, 2000 (www.netlib.org/lapack/lawns/lawn147.ps).

[4] J. J. Dongarra, J. du Croz, S. Hammarling, and R. J. Hanson, "An extended set of fortran basic linear algebra subprograms," *ACM Transactions on Mathematical Software*, vol. 14, pp. 1–17, Mar. 1988.

[5] J. J. Dongarra, J. du Croz, S. Hammarling, and I. Duff, "A set of level 3 basic linear algebra subprograms," *ACM Transactions on Mathematical Software*, vol. 16, pp. 1–17, Mar. 1990.

[6] B. Kågström, P. Ling, and C. van Loan, "GEMM-based level 3 BLAS: Portability and optimization issues," *ACM Transactions on Mathematical Software*, vol. 24, pp. 303–316, Sept. 1998.

[7] B. Kågström, P. Ling, and C. van Loan, "Algorithm 784: GEMM-based level 3 BLAS: High-performance model implementations and performance evaluation benchmark," *ACM Transactions on Mathematical Software*, vol. 24, pp. 268–302, Sept. 1998.

[8] J. W. Demmel, N. J. Higham, and R. S. Schreiber, "Stability of block *lu* factorizations," *Num. lin. Alg. with Appl.*, vol. 2, no. 2, pp. 173–190, 1995.

Jürgen v. Hagen (S'97, M'98) recevied the Dipl.-Ing. (M.S.E.E) and the Dr.-Ing. (Ph.D.E.E.) degrees from the University of Karlsruhe (TH) in 1994 and 1997, respectively. From 1994 to 1997 he was working with the CNRS (Comité National de la Recherche Scientifique), France on electromagnetic compatibility. In 1998 he held a postdoctoral position at the Electromagnetics Research Laboratory at the Pennsylvania State University, in State College, PA, USA. Since 1999 he has been with the Institut für Höchstfrequenztechnik und Elektronik (IHE) at the Universität Karlsruhe (TH), Germany where he is currently lecturer responsible for planar and conformal antennas as well as numerical techniques in electromagnetics and research assistant for industrial applications of microwaves, microwave heating processes, and EMC.

His research interests are electromagnetic theory, numerical techniques including frequency and time domain techniques, planar and conformal antennas, industrial applications of microwave power, and automotive sensors.

Dr. v. Hagen is member of IEEE, ACES, and VDE.

Werner Wiesbeck (SM'87, F'94) received the Dipl.-Ing. (M.S.E.E.) and the Dr.-Ing. (Ph.D.E.E.) degrees from the Technical University Munich, Germany in 1969 and 1972, respectively. From 1972 to 1983 he was with AEG-Telefunken in various positions including that of head of R&D of the Microwave Division in Flensburg and marketing director Receiver and Direction Finder Division, Ulm. During this period he had product responsibility for mm-wave radars, receivers, direction finders and electronic warfare systems. Since 1983 he has been director of the Institut für Höchstfrequenztechnik und Elektronik (IHE) at the Universität Karlsruhe (TH), Germany.

Research topics include radar, remote sensing, wave propagation and antennas. In 1989 and 1994, respectively, he spent a six month sabbatical at the Jet Propulsion Laboratory, Pasadena. He is a member of the IEEE GRS-S AdCom (1992-2000), Chairman of the GRS-S Awards Committee (1994-1998), Executive Vice President IEEE GRS-S (1998-1999), President IEEE GRS-S (2000 - 2001), Associate Editor IEEE-AP Transactions (1996-1999), past Treasurer of the IEEE German Section.

He has been General Chairman of the 1987 Heinrich Hertz Centenial Symposium, the '93 Conference on Microwaves and Optics (MIOP '93) and he has been a member of scientific committees of many conferences. For the Carl Cranz Series for Scientific Education he serves as a permanent lecturer for *Radar System Engineering* and for *Wave Propagation*. He is a member of an Advisory Committee of the EU - Joint Research Centre (Ispra/Italy), and he is an advisor to the German Research Council (DFG), to the Federal German Ministry for Research (BMBF) and to industry in Germany.