Highly Parallel SCF Calculation: the SYSMO Program.

Authors:
M. L. Mantovani and M. Malagoli
CICAIA (Computing Centre) and Department of Chemistry, Modena University, Modena (ITALY)

Abstract

The self consistent field (SCF) algorithm is one of the most performed tasks of computational chemistry. Determination of equilibrium geometry, potential energy surfaces, reaction paths, is usually achieved at the SCF level; SCF is also the starting point for more refined treatments including electron correlation, like Moeller Plesset perturbation theory (MP), coupled cluster (CC) and configuration interaction (CI), or for the calculation of molecular properties. The SCF approach based on the algebraic approximation, also known as linear combination of atomic orbitals (LCAO) SCF, has a computational complexity leaded by the number of two-electron integrals, which is proportional to the fourth power of the number of basis functions involved in the calculation. Here we present a parallel implementation of the LCAO-SCF algorithm, based on the Parallel Virtual Machine (PVM) environment and the System Modena (SYSMO) suite of computer programs, explicitly oriented for higly parallel computing. Parallelism is achieved at the single program multiple data (SPMD) level, dividing the calculation of two electron integrals between processes; attention is payed in load balancing and minimizing communications between processes. The algorithm is suitable for both conventional and direct SCF approaches. Tests give us results that show the high scalability of the algorithm having efficiency 0.97 on the average.

I. Introduction

The self consistent field (SCF) algorithm is one of the most used tasks of computational chemistry. Determination of equilibrium geometries, potential energy surfaces, reaction paths are usually achieved at SCF level; SCF is also the starting point for more refined treatments including electron correlation, like Moeller Plesset perturbation theory (MP), coupled cluster (CC) and configuration interaction (CI), or for the calculation of molecular properties. Owing to its popularity, SCF has been constantly studied and optimised, in order to speed up its performance and to make possible the study of larger systems. Improvements range from increasing the

efficiency of the two-electron integrals evaluation exploiting molecular symmetry [1] and using powerful recursive methods [2], to convergence accelerators [3], besides vectorization and, more recently, parallelization. The speed up of integral evaluation, together with the more and more increasing computational power of computers, had made convenient the direct SCF approach [4], which overcomes the bottleneck of disk storage of two-electron integrals.

Several computer programs implementing up to date SCF algorithms are nowadays available. Some of them, for instance GAMESS [5] and HONDO8 [6], provide parallel implementations based on message passing packages like PVM, Express and TCGMSG. Nevertheless more work is needed to obtain fully optimised parallel SCF programs, especially for massively parallel computing systems.

We present a parallel implementation of the SCF algorithm, based on the PVM environment and the System Modena (SYSMO) set of computer programs [7], explicitly oriented for highly parallel computing. The classical SCF algorithm is outlined in section II, in section III the parallel implementation of SCF in the SYSMO program is described and section IV presents some preliminary results. Further developments are outlined in section V.

II. The SCF Algorithm.

The SCF approach based on the algebraic approximation, also known as linear combination of atomic orbitals (LCAO) SCF, dates back to the work of Roothaan an Hall [8].
Within this method, the wave function of a closed shell molecular system having *2n* electrons is constituted by a Slater determinant of *n* doubly occupied molecular orbitals (MO) $\varphi_1, \varphi_2, \ldots \varphi_n$. Every MO is expanded over a basis set of *m* atomic orbitals (AO) $\chi_1, \chi_2, \ldots \chi_m$ (here the atomic basis is assumed orthonormal for sake of simplicity):

$$\varphi_i = \sum_j c_{ij} \chi_j . \qquad (1)$$

The expansion coefficients c_{ij} are variational quantities computed iteratively solving the eigenvalue matrix equation

(2)

$$\mathbf{FC} = \mathbf{C}\boldsymbol{\varepsilon},$$

where $\mathbf{F}$ is the Fock matrix, and ε is a matrix of lagrangian multipliers assumed to be diagonal.

The iterative process evolves as follow:

1) the density matrix $\mathbf{R}$ is constructed from the expansion coefficients:

(3)

$$\mathbf{R} = \sum_{i=1}^{n} \mathbf{c}_i \mathbf{c}_i^{+}.$$

2) The Fock matrix is constructed from its one-electron and two-electron components:

(4)

$$\mathbf{F} = \mathbf{H} + \mathbf{G}(\mathbf{R}).$$

3) A new estimate of the expansion coefficients is obtained by diagonalization of the Fock matrix:

(5)

$$\mathbf{C}^{+}\mathbf{FC} = \boldsymbol{\varepsilon}.$$

The process is started with an initial guess of the $\mathbf{C}$ matrix, and is continued until self consistence is achieved.

The most time consuming step of the process is the construction of the electron repulsion matrix $\mathbf{G}$, which is obtained contracting the density matrix

(6)

$$\mathbf{G}_{pq} = \sum_{rs} R_{sr}(\langle pr|qs\rangle - \langle pr|sq\rangle),$$

with the two-electron integrals over the atomic basis set:

(7)

$$\langle pr|qs\rangle = \int d\tau_1 \int d\tau_2 \chi^{*}_{p(1)} \chi^{*}_{r(2)} \frac{1}{r_{12}} \chi_{q(1)} \chi_{s(2)}.$$

The complexity of such process is leaded by the number of two-electron integrals, which is proportional to m^4. Usually the AOs are chosen as Gaussian functions, to make easier the evaluation of integrals.
In conventional SCF approaches a non-redundant list of non-zero two-electron integrals is computed once at the beginning of the process and stored in a file to be processed at every iteration (3-5); in direct SCF calculations two-electron integrals are recomputed every time they are needed [4]. In the latter case screening techniques and difference densities [9] are used to reduce the number of integrals needed at every iteration.

Molecular symmetry, if present, can be exploited to reduce the number of two electron integrals to be computed: permutation symmetry reduce the number of unique two-electron integrals to approximately $\frac{m^4}{8}$, in the presence of a point group of order $|g|$, this number is reduced to approximately $\frac{m^4}{8|g|}$ [4].

III. Parallel implementation of SCF in the SYSMO program

The original "SYSMO" program implements the conventional SCF approach and it has been designed to allow vectorization. As a result we get an optimised and fast construction of the Electron Repulsion Matrix **G** when the program is executed on a vector computer. The scalar part of the algorithm concerns the calculations of the two electron integrals.
The present work focalizes its attention on the distribution of the integrals calculation to the various parallel processes.

Each two electron integral is determined by a permutation of four (*IJKL*) integer labels so, if *M* is the number of the basis functions, approximating the atomic orbitals, we have:

$$0 \le I \le M \qquad 0 \le J \le I \qquad 0 \le K \le I \qquad \begin{cases} 0 \le L \le J & \text{if } K = I \\ 0 \le L \le K & \text{if } K < I \end{cases} \tag{8}$$

The permutations *IJKL* are generated in standard order and successively discarded if they determine duplicated integrals. This happens each time the symmetry operation transforms the integral under exam in an equivalent one.

The classical algorithm used to calculate the four labels (*IJKL*) is:

(9)

```
for (I=1; I<=M;  I++)
        for (J=1; J<=I;  J++)
                for (K=1; K<=I;  K++)
                        {
                         (K==I) ? LMX=J : LMX=K;
                         for (L=1; L<=LMX; L++)
                                {
                                 .
                                 .
                                 .
                                }
                        }
```

At this point the uniqueness of the permutation (*IJKL*) is verified for the symmetry and if this is the case the adequate integral is calculated.

The parallel implementation could be realised by two techniques:

1) Master - Slave technique:

The master process distributes to the slaves processes a contiguous part of calculations and, by polling various processes, verifies their termination status. When the status indicates the completion of a process the master assigns a new portion of calculations to the currently free processor. The disadvantages of this method are:

- it requires two versions of the same program, one master and one slave
- the master does not perform calculations but only coordinates and communicates the data to the slave processes each time a slave process restarts.

2) Single Program Multiple Data (SPMD) technique:

We adopted this method (based on data parallelization) because it is more efficient and does not present the disadvantages of the master-salve method. A unique program runs as concurrent tasks on different processors. The task itself decides, by looking at the current process number, which integrals will be calculated. There is no need to communicate with other processes.

However a problem arises during the load balancing phase. As a matter of fact the previous method does not have this kind of problem as the master is in charge of the balancing. The SPMD method, instead, requires parallelized input data to produce balanced tasks. We solved this problem at the generation of permutations (*IJKL*) stage.

It is evident that dividing the loop on I (9) in N parts does not produce balanced processes. The same holds for other loops. Consequently a new piece of code to calculate the combinations of labels is necessary.

As we have:

(10)

$$IJ = \frac{I(I-1)}{2} + J$$
$$KL = \frac{K(K-1)}{2} + L$$

Therefore the loop, generating the permutation of the four labels, changes and becomes the following:

(11)

```
I=1; NMAX=M*(M+1)/2;
for (IJ=1; IJ<=NMAX; IJ++) {
        while ( (J=IJ-I*(I-1)/2) > I)          I++;
        K=1;
        for (KL=1; KL<=IJ; KL++)



                while ( (L=KL-K*(K-1)/2 > K) {
                K++;
                .
                .
                .
                }
        }
```

The parallelization is achieved at the inner loop level by substituting it with:

(12)

for (KL=NCP; KL<=IJ; KL+=NP)

where **NPC** is the number of current process and **NP** is the total number of processes. Each process then verifies if the permutation (*IJKL*) is unique respect to the symmetry operation and if true the correspondent integral is calculated, otherwise it is eliminated.

It is easy to see (12) that distribution of permutations calculus over the processes is very well balanced, but it is necessary to consider that the effective number of integrals to be calculated by each process is much lower than the number of permutation (*IJKL*) - as we said at the end of paragraph II. So, the amount of integrals discarded by the parallel task has to be uniform. As the number of discarded integrals depends on the symmetry of the molecule in exam, it is difficult to predict the balancement of the tasks. However we expect to obtain a good balancement because the algorithm (11-12) distributes permutations of similar functions over different processes.

IV. Some preliminary results

The program was tested using PVM v. 3.2 installed on the following platforms:

1 CONVEX C220 (vector parallel computer with 2 processors);
2 HP 9000/700
2 SUN SparcStation 2
2 Silicon Graphics IRIS

Molecule	Number of basis functions	Number of symmetry operation	Total numeber of permutation of IJKL labels	Total number of calculated integrals	Number of processes	Number of calculated integrals per process	Average	Standard Deviation	Percentage of the average
C6H6	120	24	26357430	1230809	2	612207 618603	615405	3198	0,52
					4	304725 311263 307481 307339	307702	2330	0,76
					5	242806 249628 247567 248295 242508	246160,8	2938	1,19
					6	198845 216867 206209 196025 195521 217334	205133,5	9152	4,46
					8	151294 155462 151726 155801 155755 150963 153439 156384	153853	2121	1,38
C60	300	120	1019283825	8541739	10	804564 800270 902283 785990 864359 916266 965619 831346 799611 871719	854202,7	57022	6,68
SF6	190	48	164629585	3640150	10	363668 359882 356770 361491 371481 366140 366861 364359 365010 364488	364015	3818	1,05

tab. 1

We made integrals calculations for very symmetric molecules as C_6H_6 having 24 symmetry operations ($|g|=24$), SF_6 ($|g|=48$), C_{60} ($|g|=120$). The behaviour of the high symmetry molecules is the most difficult to predict because of big amount of integrals to discard.

Results we achieved show:

1) the balanced distribution of integrals (tab. 1);

2) the program is scalable and efficient (tab. 2).

If I_j is the number of integrals calculated by process j, $j = 1,\ldots, NP$, where NP is the number of parallel processes, we define the average $A = \frac{\sum_{j=1}^{NP} I_j}{NP}$, the standard deviation

$SD = \frac{\sum_{j=1}^{NP} |A - I_j|}{NP}$ and the error percentage $E = \frac{SD * 100}{A}$. Results show that the error is in the range of 0.52% and 6.68% (tab. 1).

Table 2 and fig. 1 show execution times of integrals calculations for the C_6H_6 molecule on the HP9000/700 platform. The speedup is calculated dividing scalar execution time by the average execution time of parallel processes. Efficiency is calculated dividing the speedup by the number of processes. In this example one can see the efficiency of about 0.97.

Number of processes	1	2	3	4	5	6	7	8	9	10	11
Average seconds	450	229	156	113	91	79	66	61	52	46	42
Minimum seconds	450	227	155	111	89	78	65	57	45	44	39
Maximum secondes	450	231	156	114	92	80	66	65	54	48	45
Speedup		1,965066	2,884615	3,982301	4,945055	5,696203	6,818182	7,377049	8,653846	9,782609	10,71429
Efficiency		0,982533	0,961538	0,995575	0,989011	0,949367	0,974026	0,922131	0,961538	0,978261	0,974026
Efficiency average		0,968801									

tab. 2

V. Further Developments

The SYSMO package is a set of computer programs devoted to the ab-initio study of molecular electronic response properties, and the SCF algorithm represents the starting point of such calculations. Within the SYSMO program two techniques are available for property determination, namely coupled Hartree-Fock (CHF) method [10] and random phase approximation (RPA) [11]. The CHF approach is based on a series of iterative calculations which share the most time consuming step with SCF. In this case the extension of the parallelization is straightforward. In the case of RPA calculations, the most complicated task is the parallelization of two-electron integrals transformation from AO to MO basis, which is a m^5 process. After this step the RPA calculation consists on a series of matrix diagonalizations, for which parallelization could be easily achieved.

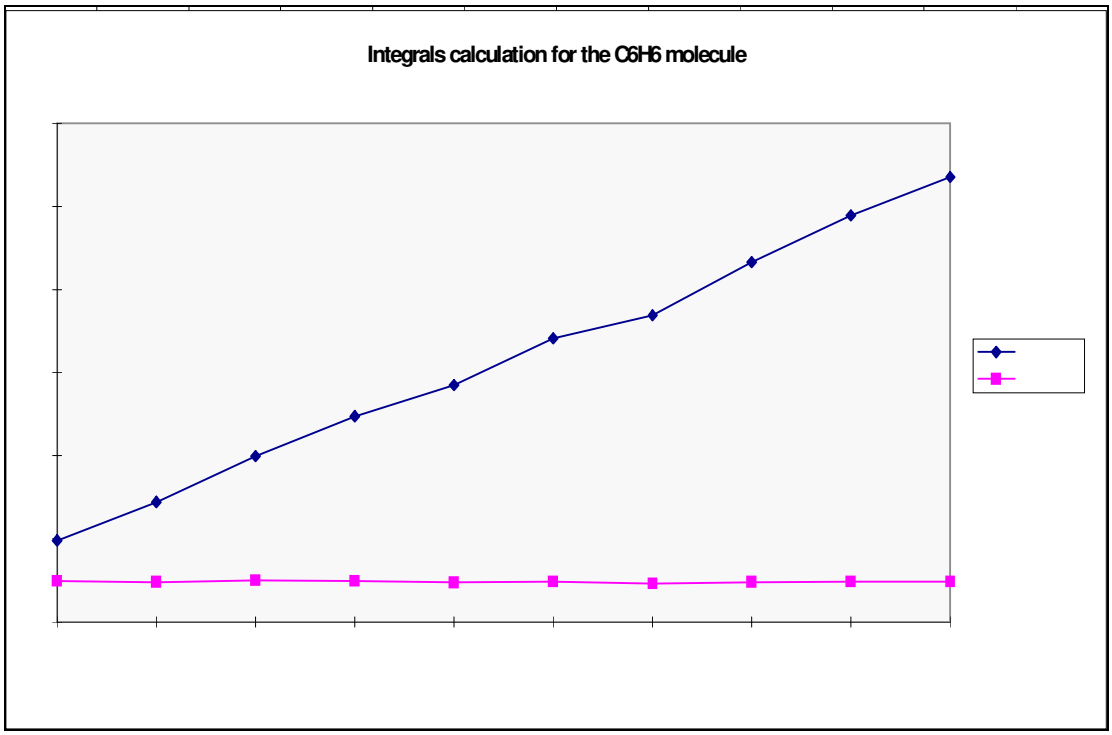

fig. 1

Bibliography.

[1] P. D. Dacre, Chem. Phys. Letters 7, 47 (1970);
M. Elder, Int. J. Quantum Chem. VII, 75 (1973).

[2] S. Obara, A. Saika, J. Chem. Phys. 89(3), 1540, (1988);
P. M. W. Gill, J. A. Pople, Int. J. Quantum Chem. 40, 753 (1981)

[3] P. Pulay, Chem. Phys. Letters 73, 383 (1980);
P. Pulay, J. Comp. Chem. 3, 556 (1982);
T. P. Harnilton and P. Pulay, J. Chem. Phys. 84, 5728 (1986).

[4] J. Almlof, K. Faegri and K. Korsell, J. Comp. Chem. 2, 285 (1982).

[5] M. W. Schmidt, K. K. Baldridge, J. A. Boatz, S. T. Elbert, M. S. Gordon, J. H. Jensen, S. Koseki, N. Matsonaga, K. A. Nguyen, S. Su, T.L. Windus, M. Dupuis and J. A. Montgomery Jr, J. Comp. Chem. 14, 1347 (1993).

[6] M. Dupuis, A. Farazdel, S. P. Karna and S. A. Maluendes, in Modern Techniques in Computational Chemistry,
E. Clementi, ed ESCOM, Leiden, The Netherlands, 1990, p. 277.

[7] P. Lazzeretti, M. Malagoli and R. Zanasi, CNR, Tech. Rep. on project "Sistemi Informatici e Calcolo Parallelo", n. 1/67, Rome (1991).

[8] C. C. J. Roothaan, Rev. Mod. Phys. 23, 69 (1951);
G. G. Hall, Proc. R. Soc. London Ser. A, 208, 328 (1951).

[9] M. Haser and J. Ahlrichs, J. Comp. Chem. 10, 104 (1989).

[10] R. McWeeny, Phys. Rev. 126, 1028 (1962);
G. F. Diercksen and R. McWeeny, J. Chem. Phys. 44, 3554 (1966);
R. McWeeny, Chem. Phys. Letters 1, 567 (1968); J. L. Dodds, R. McWeeny, W. T. Raynes and J. P. Riley, Molec. Phys. 33, 611 (1977);
J. L. Dodds, R. McWeeny and A. J. Sadlej, Molec. Phys. 34, 1779 (1977).

[11] D. J. Rowe, Rev. Mod. Phys. 40, 153 (1968);
P. Jorgensen and J. Linderberg, Int. J. Quantum Chem IV, 587 (1970).