CROATICA CHEMICA ACTA CCACAA **67** (1) 85–94 (1994)

ISSN 0011-1643
CCA–2168 *Original Scientific Paper*

Parallelization and Vectorization of Quantum Mechanical Programs in Solid State Physics

P. Otto

Chair for Theoretical Chemistry
Friedrich-Alexander University Erlangen-Nürnberg
Egerlandstr. 3, D – 91058 Erlangen, Germany

Received July 27, 1993

Knowledge of the structure and functionality of chemical systems does not only allow verification of physical and chemical mechanisms of model hypothesis but it also offers the possibility of designing new materials with improved properties. Quantum mechanical methods are the appropriate tool for performing theoretical investigations on the molecular level. Reliable calculations on complex chemical molecules and macromolecules require the usage of high-performance computers. The most promising development in the modern computer technology – the massive parallel multiprocessor systems – can be efficiently used to investigate the structure of large and complex chemical systems.

INTRODUCTION

Analysis, modeling, prediction and optimization of molecular and macromolecular structures using supercomputers is one of the »challenging classes« of scientific high-performance computing. Therefore, new approaches to scaling, that is adjusting problems to the architecture of computers, as well as further development of extreme capacity computer systems are necessary.

Application of the quantum mechanical methods to investigate physical and chemical problems at the molecular level mainly serves two tasks:

a) mechanisms of physical and chemical experimental findings can be enlightened and hypothetical interpretations can be verified. At present, it seems most probable that several important questions, *e.g.* high–T_c superconductivity of ceramic materials or the primary steps of chemical carcinogenesis, can be only solved with the help of theoretical investigations at the microscopic level.

b) Application of the theoretical quantum mechanical methods can be of great value in material sciences to predict new chemical systems with a selected combination of improved properties. In this way, expensive and time-consuming experiments can be avoided. Furthermore, theoretical structure-activity relations can be used to design *e.g.* enzymes and pharmaceuticals of very specific action.

To obtain reliable results from theoretical investigations of complex chemical systems, one has to perform highly accurate calculations using *ab initio* methods. This means that the requirements are very high with respect to the CPU-time, core memory and disk space.

In the last few years, the development of computer technology has followed mainly two ways: vector computers and multiprocessor systems and, their combinations. Cost/efficiency considerations turned out to be in favour of parallel computers, which can be realized in the simplest way by a loosely coupled array of work stations. As it can be expected at this stage of development and experiments, the commercially available systems do not only have different processor architectures, interprocessor communication systems and disk storage managements, but there is no common software supporting parallelization of programs, either.

Therefore, today the users have to adjust and implement their programs on various types of multiprocessor systems and discuss their experience with the teams involved in the design of hardware, software and operating systems. Most of the quantum chemical algorithms are optimally appropriate for parallelization and, in addition, for vectorization, and these properties can be used for the development of highly efficient program codes to perform large numerical applications.

In the next paragraph, we will briefly review the basic equations of the *ab initio* Hartree-Fock crystal orbital method.[1-3] Then, a brief summary follows on how vectorization has been accounted for. The strategy of integral programs parallelization is described and the results are given of the speed-up and efficiency measurements on different multiprocessor systems.

Finally, we present the more complex parallelization scheme of the SCF-program, including also a proposal on how to distribute the diagonalization of hermitian matrices on a multiprocessor system.

BASIC EQUATIONS OF THE *AB INITIO* HARTREE-FOCK CRYSTAL ORBITAL (HF-CO) METHOD

In the HF-CO theory,[1-3] the generalized hermitian eigenvalue problem

$$\boldsymbol{F}(k_i)\ c_n(k_i) = \varepsilon_n(k_i)\ \boldsymbol{S}(k_i)\ c_n(k_i) \tag{1}$$

has to be solved for $i = 1, ..., \mathrm{NKP}$ points in the Brillouin zone and $n = 1, ..., \mathrm{NBF}$ bands. The Fock and overlap matrices, $\boldsymbol{F}(k)$ and $\boldsymbol{S}(k)$, respectively, are obtained as the Fourier transforms of the corresponding matrices in direct space

$$\boldsymbol{M}(k) = \sum_{J=-\mathrm{NEIG}}^{\mathrm{NEIG}} \exp(iR_J k)\ \boldsymbol{M}^{0J} \tag{2}$$

Matrices $\boldsymbol{F}^{OJ}$ describe the interaction of electrons in the reference cell (denoted by 0) and the neighboring cells J, and NEIG is the number of repeating units that is taken explicitly into account in the calculation.[4]

The most time-consuming step consists of the calculation of the large number of two-electron integrals, which describe the Coulomb repulsion between two electrons. The general form is

$$\int X_r(r_1)\, X_u(r_2)\, \frac{1}{r_{12}}\, X_s(r_1)\, X_v(r_2)\, \mathrm{d}r_1\, \mathrm{d}r_2 \qquad (3)$$

where the subscripts r, s, u and v run over all basis functions and the superscripts can take the values –NEIG,...,+NEIG. Ngroup cell combinations are possible with respect to the location of the four Gaussian type atomic orbitals which, in turn, are linear combinations of the so-called primitive Gaussian function ($m = 2 - 10$). The value of Ngroup increases rapidly with the increasing number of interacting cells NEIG (*e.g.* NEIG = 1, 2, 3; Ngroup = 5, 15, 35). The total number of two-electron integrals to be computed (without taking into account symmetry restriction with respect to basis function indices) is given by

$$\mathrm{Ngroup} \times \mathrm{NBF}^4 \times m^4$$

Since Gaussian lobe functions[5,6] have been used as basis functions, each integral requires the same number of arithmetic operations:

– 22 additions
– 0 subtractions
– 20 multiplications
– 2 divisions
– 2 square roots

Since it is mandatory to introduce an integral threshold value, two words are needed to store the integral (one word for the indices and the other for the integral value; of course, packing of both informations into a single word is also possible).

It is obvious that the calculation of integrals involves a high degree of parallelization, because they can be calculated independently from each other. The requirements on the communication system are low and the principal difficulty consists of the load balancing of all processors. But, before going into more detail on how we have solved this problem, we would like to make some remarks on the vectorization of the code.

VECTORIZATION OF THE TWO-ELECTRON INTEGRAL PROGRAM

It has already been mentioned above that we use Gaussian lobe functions[5,6] in our new crystal orbital program,[4] which means that the Cartesian Gaussian p- and d-functions are approximated (up to any accuracy) by linear combinations of s-type orbitals. This has the advantage that integrals between all different types of atomic orbitals are calculated with the same algorithm. So the innermost loop in the program can be formulated (choosing high values for the loop variables) in an efficient vectorizable form.

Furthermore, to avoid redundant calculations of intermediate quantities, we precalculate one-electron terms (between two atomic orbitals) and even two-electron terms, including all four basis functions, and store them in vectors.

Calculation of the final integral value in the innermost loop in a vectorized form is prevented by the necessary evaluation of the Incomplete Gamma Function. Usually, one applies a different algorithm, depending on the value of the argument (*e.g.* a Taylor series expansion for medium size and an asymptotic expansion for large values.[7,8] We have developed two new algorithms, one is based on a polynomial division[9] and the other uses a sophisticated storage mode of a Table of values for an extrapolation technique:[10] both can be evaluated in an unconditional vectorized form.

PARALLELIZATION STRATEGY FOR THE TWO-ELECTRON INTEGRAL PROGRAM

Figure 1 shows the loop structure of the two-electron integral program.[4,11] Task distribution to individual processor nodes has not been done like in former investigations in the loop running over the i group[12,13] (a sufficiently uniform load balancing could only be achieved with large values of Ngroup). A finer granularity is obtained by partitioning the two loops over the basis functions i and j (NBF × NBF pairs). Since the tasks are distributed using the farming-concept, as depicted in Figure 2, load balancing is almost unaffected by the different sizes of the total task. One processor, the »master«, first precalculates the data, which are necessary for all integrals, and then divides the i,j-pairs for each group into as many parts as »slave«-processors available. As soon as one of the »slaves« sends the message to the »master«, that he has finished his task, the »master« will send a new task to him, which contains all the necessary information on the i,j-range and the precalculated data for the new group.

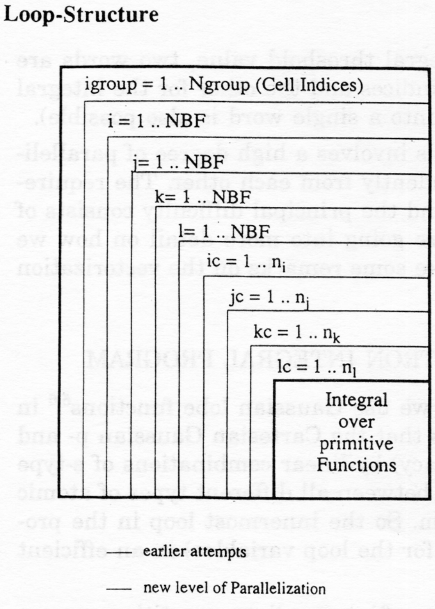

Figure 1. Loop-structure of the crystal orbital two electron integral program.

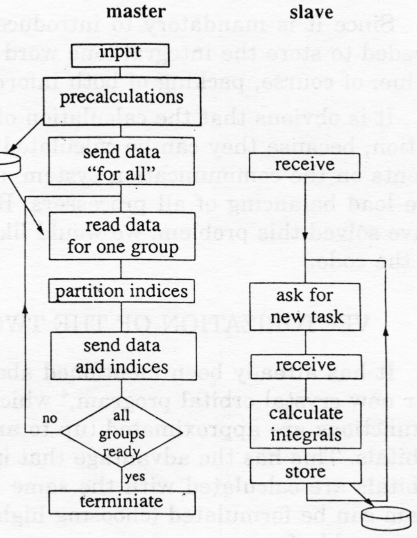

Figure 2. Flow chart of the master-slave concept of the integral program.

PARALLELIZATION RESULTS AND ANALYSES

The degree of parallelization can be best measured by the speed-up s and efficiency e, which are defined by the following relations with respect to t_s the time needed by the sequential program, and t_p, the time of parallel performance:

$$s = t_s / t_p$$
$$e = s / \text{nproc} = t_s / t_p \times \text{nproc}$$

Speed-up and efficiency measurements, as well as an analysis of communication, were carried out for different multiprocessor systems with distributed memory:

- SUPRENUM, with up to 256 processors and vector co-processors (maximum number was 32): 20 Mflops peak performance per node.
- Intel iPSC/860 with 32 processors (hypercube architecture); 16 MBytes memory; peak performance of 60 Mflops per node.

The results are given for the model system (NBF=14, Ngroup=5) in Figures 3a and 3b for both multiprocessor systems. Though the actual SUPRENUM CPU-time used is larger by a factor of 3.5, as compared to Intel, nearly identical speed-up and efficiency curves are obtained. Maximum efficiency for this polymer is reached with about ten processors. Increasing the number of processors, one observes a slow decrease of the efficiency below 0.8, which means that about 20 per cent of the CPU-time is used up for additional efforts related to parallelization.

Increasing the number of integral groups (Ngroup=15) the efficiency is more than 80 per cent, also by using more than ten processors. A corresponding improvement is observed when the number of basis functions (NBF) is increased, as it can be seen from Figures 4a and 4b. We can conclude that a sufficiently high speed-up,

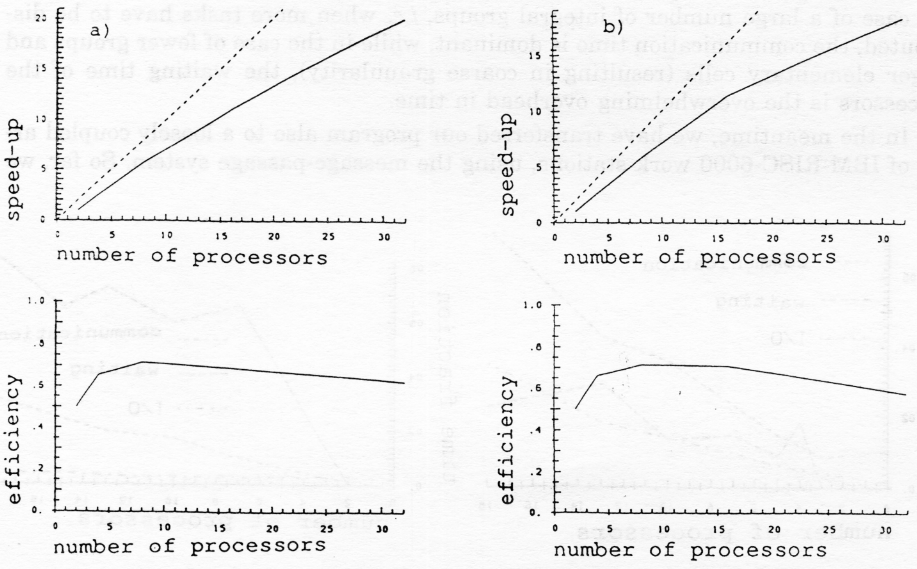

Figure 3. Speed-up and efficiency for the model system with NBF = 14, Ngroup = 5, for (a, left) SUPRENUM and (b, right) Intel iPSC/860.

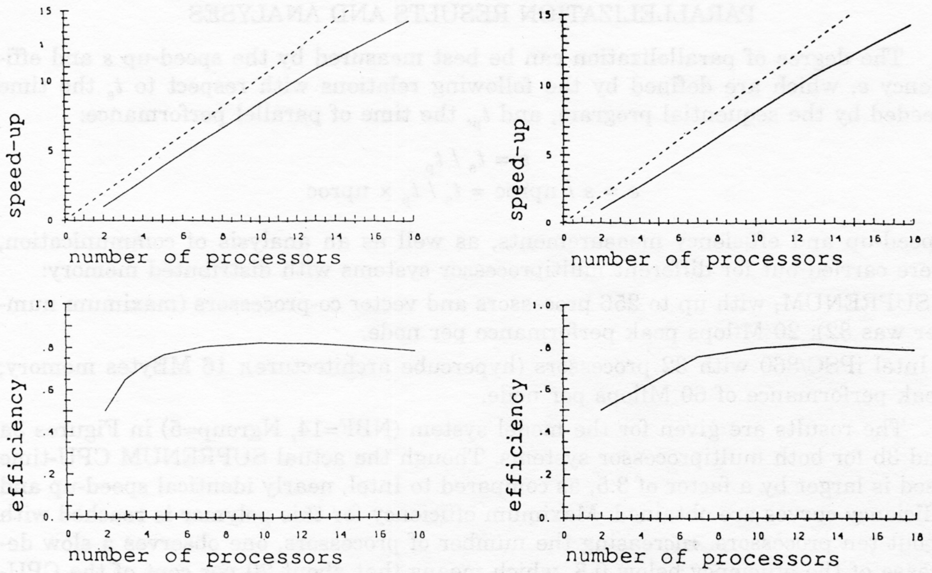

Figure 4. Speed-up and efficiency of SUPRENUM with the model system (a, left) NBF = 14, Ngroup = 15, and (b, right) NBF = 28, Ngroup = 5.

using a number of processors, can be achieved by performing the calculation of polymers with complex chemical structures.

The analysis of computation time loss (see Figures 5a and 5b) reveals that, in the case of a large number of integral groups, *i.e.* when more tasks have to be distributed, the communication time is dominant, while in the case of fewer groups and larger elementary cells (resulting in coarse granularity), the waiting time of the processors is the overwhelming overhead in time.

In the meantime, we have transferred our program also to a loosely coupled array of IBM-RISC-6000 work stations, using the message-passage system. So far, we

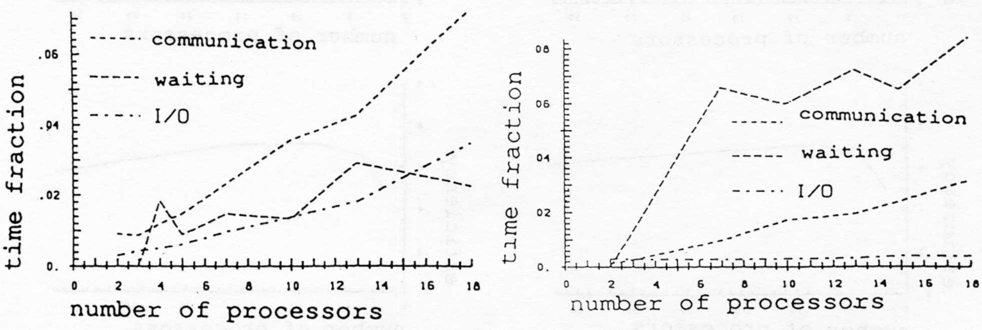

Figure 5. Analysis of parallelization overhead for (a, left) NBF = 14, Ngroup = 15, and (b, right) NBF = 28, Ngroup = 5 for the multiprocessor system SUPRENUM.

could not obtain results for the speed-up and efficiency comparable with those of SUPRENUM or Intel.

Furthermore, completely new version of the integral program has recently been developed[14,15] for the multiprocessor system MEMSY (modular expandable multiprocessor system), which is under development in the Computer Science Department of our University. The architecture of this parallel system has a pyramidal structure and consists of three levels of processors having, besides a global disk (which can serve as a device to exchange data), local communication storage devices that can be used for nearest neighbor data transfer. This multiprocessor system differs from other systems in having highly complex and complicated communication structure signals, short messages semaphores *etc.* First applications, however, show that we can obtain results as good as those for the other parallel computers investigated. Details of this work can be found elsewhere.[14,15]

AN OUTLINE OF THE SCF-PROGRAM PARALLEL STRUCTURE

The generalized hermitian eigenvalue problem defined by Eq. (1) has to be solved iteratively because the matrix elements are a function of the solution. NKP matrix diagonalizations have to be performed in each iteration step.

Simultaneously with the development of the SCF-program, we developed parallelization of the QR-diagonalization algorithm.[16,17] We succeeded in working out an optimal distribution of tasks starting with the matrix in its tridiagonal form[18,19] (the Householder transformation has not yet been worked out in parallel). We first developed a concept without performing the diagonalization in parallel. In the next section, we will point out how the program structure has to be organized, taking this additional parallelization into account.[20]

Figure 6 gives the flow chart for the first approach. Again, the »master-slave« strategy has been adopted. In the first step, the »master« sends the charge-bond order matrices to all »slaves«, building up matrices $\boldsymbol{F}^{0J}$ for all values of J. Since each processor can only access a part of integrals (those which have been calculated using the »farming« concept), incomplete matrices are stored in the core memory of each processor. These partial matrices are sent to the »master« where they are added up. Having obtained a complete set of matrices, $\boldsymbol{F}^{0J}$ each »slave« can build up matrix $\boldsymbol{F}(k)$ for a given value of k, which will be separately diagonalized on each node. From the eigenvectors of a certain k-point, a partial charge-bond order matrix can be calculated. To be able to calculate the Fock matrices in direct space, first the »slaves« have to send the partial density matrices back to the »master«, where they are added up, and then they receive them again. This mutual intercommunication has to be repeated in each iteration cycle.

In the case of diagonalization performed also in parallel, the structure is somewhat more complicated, as seen from Figure 7. The first steps remain the same as in the previous case but the NKP diagonalizations per iteration cycle are now distributed to the nodes as follows: first, one groups the nslav »slave« processors into NKP clusters, each having nslav/NKP nodes, and defines one processor as the »clustermaster«, responsible for coordination tasks. Within each cluster, complete $\boldsymbol{F}^{0J}$ matrices are distributed columnwise to the individual nodes, where the Fourier transform $\boldsymbol{F}(k_i)$ for the respective k-value is constructed. The summation is then done by the »clustermaster«. The number of data to be sent is $\text{NBF}^2 \times (\text{NEIG}+1)$. The »clus-

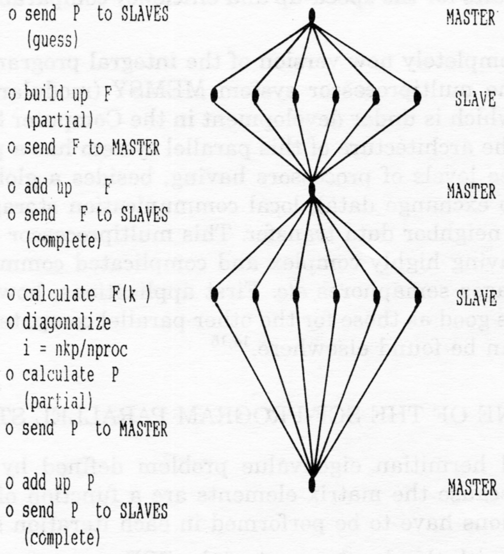

Figure 6. Parallel concept of the SCF program without performing diagonalization in parallel.

termaster« then performs a Löwdin orthogonalization to transform the task into a special eigenvalue problem. Again, the step can be distributed to the processors of the cluster as well as the following tridiagonalization with the help of the Householder- or Block-Householder-method.

The diagonalization of the tridiagonal matrix is done iteratively using the QR-algorithm. Its parallel structure in the »slave«-cluster is illustrated in Figure 8.[16-19] Two steps are necessary, calculation of the matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ (factorization $\boldsymbol{H} = \boldsymbol{QR}$) and multiplication $\boldsymbol{H}' = \boldsymbol{RQ}$. Calculation of the new columns of $\boldsymbol{Q}$ can be performed without information on the neighboring rows. Therefore, the rows of $\boldsymbol{Q}$ can be distributed without additional communication. A corresponding statement can be made with respect to the columns of $\boldsymbol{R}$. In addition, to calculate the elements of the product matrix $\boldsymbol{H}$; only the information on the involved columns and rows is necessary. Therefore, it can be performed also in parallel. To avoid load balancing problems, caused by the structure of the matrices ($\boldsymbol{R}$ and $\boldsymbol{Q}$ are of triangular and Hessenberg form, respectively), the distribution to the processors is not done blockwise but with the help of the »team mapping« concept, corresponding to an alteration ascending and descending distribution of rows (respectively columns). When the diagonalization is finished, each »clustermaster« calculates the incomplete charge-bond order matrices from its eigenvectors. These will be sent to the »master«, where they are added up.

In contrast to the calculation of integrals, additional communication and synchronization problems arise in the SCF-program. Communication is of the type »all-to-one« and »one-to-all«. Implementation of individual steps depends strongly on the architecture of the computer and communication system. Especially the architecture of the MEMSY-processors with local communication storage turns out to be very ap-

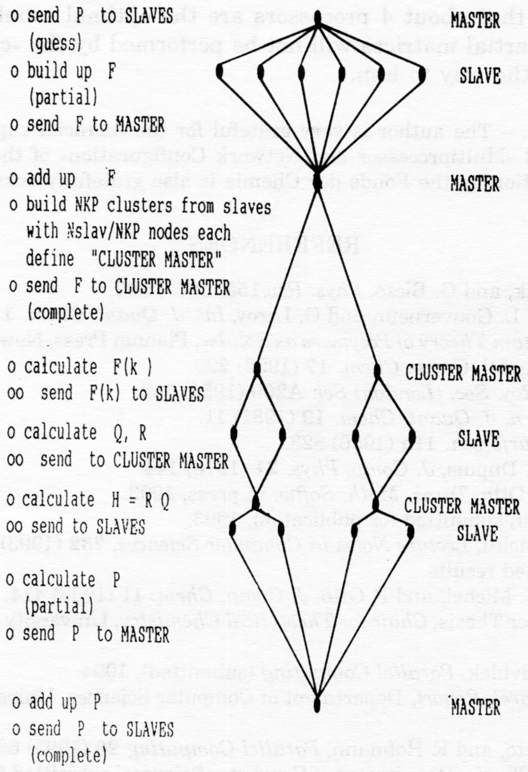

Figure 7. Communication structure of the SCF program, including parallelization of the matrix diagonalization.

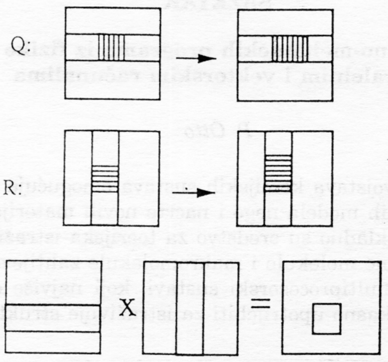

Figure 8. Parallelization of the QR-algorithm. Calculation of matrices *Q* (distributed by rows) and *R* (distributed by columns) and matrix multiplication *RQ*.

propriate for this problem (the first results of parallelization of matrices in the order 400 × 400 indicate that about 4 processors are the optimal number). In this case, the summation of partial matrices will not be performed by the »clustermaster« but is done already on the way to him.

Acknowledgement. – The author is very grateful for the financial support of the Sonderforschungsbereich 182 »Multiprocessor and Network Configuration« of the German Research Council. The contribution by the Fonds der Chemie is also gratefully acknowledged.

REFERENCES

1. G. Del Re, J. Ladik, and G. Biczo, *Phys. Rev.* **155** (1967) 997.
2. J. J.-M. André, G. L. Gouverneur, and G. Leroy, *Int. J. Quant. Chem.* **1** (1967) 427, 451.
3. J. Ladik in »*Quantum Theory of Polymers as Solids*«, Plenum Press, New York, London, 1988.
4. P. Otto and H. Früchtl, *Comp. Chem.* **17** (1993) 229.
5. S. F. Boys, *Proc. Roy. Soc. (London) Ser.* **A200** (1950) 542.
6. H. Le Rouzo, *Intern. J. Quant. Chem.* **19** (1981) 11.
7. H. Preuss, *Z. Naturforsch.* **11a** (1956) 823.
8. H. F. King and M. Dupuis, *J. Comp. Phys.* **21** (1976) 144.
9. H. Früchtl and P. Otto, *Trans. Math. Softw.* in press, 1993.
10. P. Otto and H. Reif, submitted for publication, 1993.
11. P. Otto and H. Früchtl, *Lecture Notes in Computer Sciences*, **732** (1993) 265.
12. P. Otto, unpublished results.
13. S. Kindermann, E. Michel, and P. Otto, *J. Comp. Chem.* **11** (1992) 414.
14. J. Nedvidek, Master Thesis, *Chair for Theoretical Chemistry*, University Erlangen-Nürnberg, July, 1993.
15. P. Otto and J. Nedvidek, *Parallel Computing* (submitted), 1994.
16. T. Schreiber, *Research Report,* Department of Computer Sciences, University Erlangen-Nürnberg, 1992.
17. T. Schreiber, P. Otto, and F. Hofmann, *Parallel Computing* **20** (1993) 63.
18. O. Kaiser, *Master Thesis*, Department of Computer Sciences, submitted for publication, 1993.
19. O. Kaiser, F. Hofmann, and P. Otto, to be published.
20. P. Otto and H. Früchtl, to be published.

SAŽETAK

Implementacija kvantno-mehaničkih programa iz fizike čvrstog stanja na paralelnim i vektorskim računalima

P. Otto

Poznavanje strukture i svojstava kemijskih sustava omogućuje ne samo provjeru pretpostavljenih fizikalnih i kemijskih modela nego i nacrte novih materijala poboljšanih svojstava. Kvatno-mehaničke metode prikladno su sredstvo za toerijska istraživanja na molekulskoj razini. Pouzdani računi za složene molekule i makromolekule zahtijevaju primjenu vrlo moćnih računala. Masivni paralelni multiprocesorski sustavi, koji najviše obećavaju u suvremenom razvoju računala, mogu se efikasno upotrijebiti za istraživnje strukture velikih i složenih kemijskih sustava.