k-space parallelization of WIEN97

Joachim Luitz*

Theoretical Chemistry
Vienna University of Technology
Getreidemarkt 9/158
A-1060 Vienna
jluitz@zeus.theochem.tuwien.ac.at

November 1997

Abstract. **WIEN97** is a program package to calculate the electronic structure of solids. The calculations are based on Density Functional Theory. A coarse grain parallelization of the code based on distributing k-points onto different processors is not only relatively simple to implement but also very efficient, especially in metals, where a large number of k-points must be calculated. After the development of a running parallel version, care was taken to implement features to ensure workload-balancing for a cluster of heterogeneous workstations.

1 Introduction to WIEN97

WIEN97 is a program package for calculating crystal properties and is based on the Full-Potential Linearized Augmented Plane Wave (FP-LAPW) Method. The LAPW method is one of the most accurate methods to compute the electronic structure within Density Functional Theory (DFT). The current version is based on the original WIEN code ([1]), which has been successively improved in WIEN95 ([2]) and the current version **WIEN97** ([3]) version. It is used worldwide by more than 150 different academic and commercial groups. It is written almost entirely in standard FORTRAN 77 and the links between the programs are realized using C-Shell and UNIX commands.

* The work described in this paper was supported by the Special Research Program SFB F011 "AURORA" of the Austrian Science Fund.

2 Calculations with WIEN97

To initialize a calculation a set of programs is run. This will generate input files and a starting potential. This potential is then used to solve the Kohn-Sham equations which yield the wave functions and the electron density from which a new potential is generated, which in turn will be used as a new starting potential. This cycle is repeated until some convergence criterion is met (See figure 1) and is called **Self-Consistent Field Cycle** (SCF-cycle).

The tasks performed by the programs in the SCF-cycle are:

LAPW0 generate a potential from the density
LAPW1 calculate valence bands (eigenvalues and eigenvectors)
LAPW2 compute valence densities from eigenvectors
LCORE compute core states and densities
MIXER mix input and output densities

The major part of a calculation takes place in the SCF-cycle. Within the SCF-cycle the programs LAPW1 and LAPW2 consume most of the CPU time.

The Kohn-Sham equations in a solid must be solved for a number of different k-vectors. They are independent by definition and thus the corresponding tasks can easily be solved in parallel. Therefore the programs `lapw1` and `lapw2` were parallelized. Parallelization of the other programs is not so crucial at first because their CPU consumption is much less than for the two programs mentioned above.

3 Shell-Script Parallelization

As a first step a very coarse grain parallelization was designed that utilizes nothing more than a few (C-Shell) scripts, a common NFS-system on all workstations used and a small additional FORTRAN program.

The following steps were necessary to achieve parallelization:

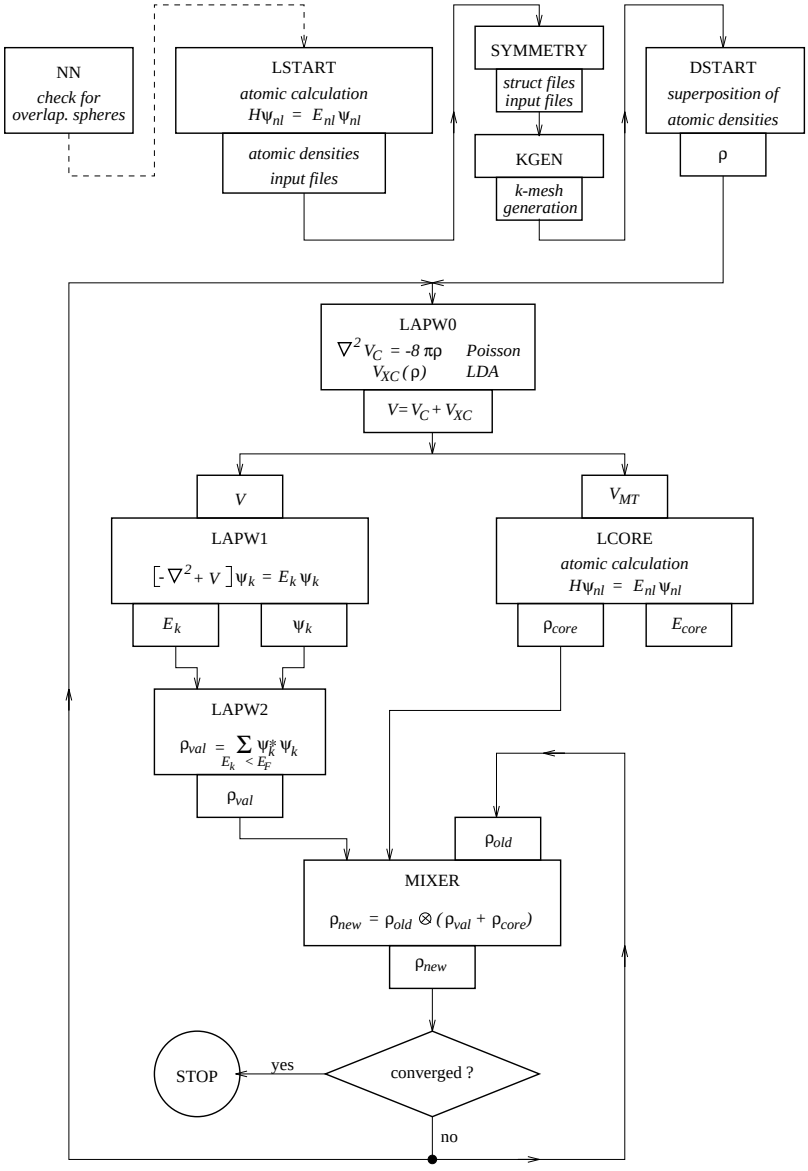

Fig. 1. Flow of programs in **WIEN97**

- generating independent input files for `lapw1`
- running `lapw1` in parallel with these input files
- calculating the Fermi energy using `lapw2` in a special mode (on a single processor)
- calculating (partial) valence charge densities with `lapw2` in parallel
- summing up all partial valence charge densities (generated on different nodes).

In the following the scripts and utility programs that were developed for running this parallel version are briefly described.

3.1 Parallel lapw1 (lapw1para)

In the `lapw1para` script the list of k-vectors (stored in file `case.klist` is split into smaller sets of k-points that can be solved independently. A control file `.machines` is used to specify the machines to be used and their "relative speed" (weight). The list of k-points is split into subsets according to the weights specified in the control file.

$$K_i = INT\left(\frac{weight_i * klist}{\sum_j weight_i}\right)$$

$$K_{plus} = klist - \sum_j K_i$$

whereas K_i is the number of k-points to be calculated on processor i

The residual K_{plus} k-points can be handles in two possible ways:

- distribution over processors: to each processor one k-point is added until no k-points are left
- calculation of the K_{plus} k-points as a separate process (on a 'residue' machine).

Having prepared the independent input files the parallel calculation of `lapw1` can be performed. See figure 2 for a schematic illustration.

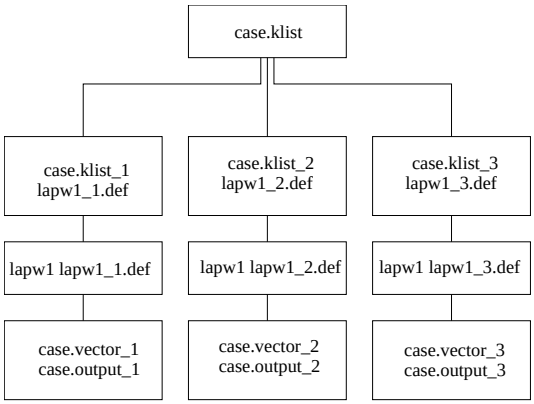

Fig. 2. Parallel lapw1

3.2 Parallel lapw2 (lapw2para)

Modifications of the program `lapw2` were more sophisticated. In `lapw1` eigenvectors and eigenenergies are stored on a file. In the parallel version several of such `case.vector` files are generated. In order to calculate the Fermi energy all of these files are needed.

Once the Fermi energy is determined, the calculation can proceed again in parallel to calculate the "partial" valence charge densities corresponding to the subset of k-vectors. When all these calculations are finished, a final summation over all these "partial" densities must be done to yield the total valence charge density. This is done by a separate FORTRAN program called `sumpara`. See figure 3 for a schematic illustration of the program flow.

4 Development of the parallel versions

4.1 A first test version

In a first test version a fixed weight corresponding to the speed was assigned (in the control file) to each processor. The surplus k-vectors

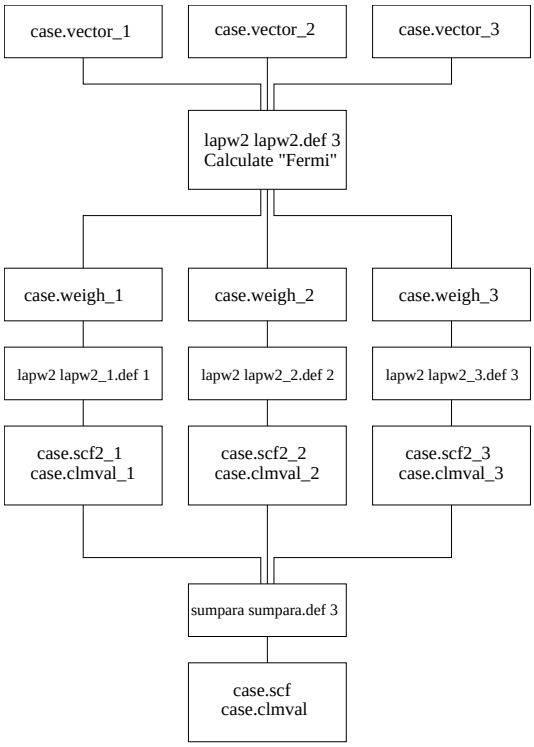

Fig. 3. Parallel lapw2

(at the end of the k-vector list) were distributed to all available processors or to a "residue" machine.

Using the scheme described above two versions were tested: one for a multi-processor architecture and one for a heterogeneous workstation cluster but they were only used temporarily to figure out performance data.

The speed-up was found to be linear with the number of processors provided the adjustments of the different processor weights (speed) in a heterogeneous workstation cluster was properly done. This ideal situation can only be achieved as long as the number of k-points is sufficiently large with respect to the number of nodes on which the load should be distributed. This is often the case in metallic systems.

4.2 First official version

A first parallel version was made available to the users of **WIEN97** with the distribution of release WIEN 97.1. It consisted of a unified version of the two test versions (multi-processor and heterogeneous version). It was a well developed running version. The speed-up was near linear.

4.3 Developing version for load balancing

The first parallel version had one major drawback: the speed of all nodes must be known before run time (it could also be re-adjusted during a run after completing a whole SCF cycle). Especially on a heterogeneous cluster of workstations this proved to be almost impossible, since each additional process that runs on certain machine affects its relative speed with respect to the other nodes.

In a developing version we try to overcome this by distributing each k-vector consecutively to the next free node. If a processor is already busy no new task sould be assigned to this processor. Thus the next k-point in the list will be sent to that processor which becomes available first.

In principle this worked quite well. However, there are reasons why one should not distribute every k-point independently:

- For small systems the calculation of one k-point takes only a very short time. Since there was a delay of up to two seconds to start up a job on the remote machines using the remote shell command, too much overhead is found for such systems.
- For each k-point a "partial" valence charge density file (named `case.clmval_*`) was generated. This file has the same size, regardless of the number of k-points calculated. Therefore the number of files equals the number of k-points and cause waste of disk space with respect to the serial version.

These reasons led to the development of the next version.

4.4 New version with load balancing

To reduce the disadvantages of distributing every k-point independently, we introduced a "**granularity factor**". This factor defines the approximate number of files (and therefore also the number of start-up sequences) for each processor.

Setting the granularity factor equal to one is (almost) equivalent to using the first parallel version. The difference can only occur due to round-off errors which can lead to 2 files for one processor.

The number of k-points to be distributed to a specific processor is now given by:

$$newweight_i = INT\left(\frac{weight_i * klist}{granularity * \sum_i weight_i}\right)$$

A value of $newweight_i = 1$ is used whenever the right-hand side equals zero.

This is the new version included in **WIEN97** with release WIEN97.5, the latest update.

5 Outlook

The program `LAPW2` contains a part where a large list of reciprocal vectors is calculated. Currently this part is calculated on every processor. For high symmetry systems this part is negligible, but for others it needs significant CPU time. Therefore we now calculate this K-list once (already in the serial part of the parallel execution of `LAPW2`, i.e. after calculating the Fermy energy), store it in the file `case.reprlist`, and reuse it in the parallel part. This modification will be made available to the **WIEN97** users with the next release (WIEN97.6, scheduled for April 98).

For (especially non-metallic) systems with many atoms per unit cell it is often sufficient to make calculations with only **one** k-point. In

such a case this coarse grain "k-parallelization" can no longer be used. For theses cases fine grain parallelization must be developed.

For this purpuse small but CPU intensive kernels of the **WIEN97** code have been extracted and serve as benchmark tests for efficient implements in HPF. Work along this line is in progress.

References

1. Blaha P., Schwarz K., Sorantin P. and Trickey S. B., *Comput. Phys. Commun.* 59(1990) 399
2. Blaha P., Schwarz K., Dufek P. and Augustyn R., *WIEN95* (Vienna University of Technology 1995)
3. Blaha P, Schwarz K., Luitz J., *WIEN97* (Vienna University of Technology 1997)
4. Gilly D., *UNIX in a Nutshell* (O'Reilly 1994)