Progress Report on Parallel PHOENICS

TABLE OF CONTENTS

1 Introduction

The purpose of this report is to provide information on the design, testing, use of Parallel PHOENICS 3.5.1 and further work on Parallel Phoenics 3.6. The report is structured as follows:

Main ideas, used during Parallel PHOENICS implementation, are described in Chapter 2. These are:

- domain decomposition method for calculations using multi-processor system,
- data exchange between processes,
- features of Linear Solver using for parallel calculations.

Chapter 3 contains brief information about setting and configuration of Parallel PHOENICS.

Chapter 4 describes methods of Parallel PHOENICS testing, results of testing and provides information about features of calculations by comparison with sequential version. In addition, Chapter 4 contains information for PHOENICS users about what tasks parallel PHOENICS can now solve and what it cannot yet.

Chapter 5 describes briefly main ideas using for further work on PHOENICS 3.6.

2 Design of Parallel PHOENICS 3.5 and Parallel EARTH implementation

2.1 Parallelisation Strategy

The EARTH module uses most of the computing time and therefore it is this part of PHOENICS, which is ported to the parallel computer. The most suitable strategy for parallelising EARTH is grid partitioning (or domain decomposition), where the computational domain is divided into sub-domains. The computational work related to each sub-domain is then assigned to a process. A modified version of EARTH is replicated over all available processors (or processes) and runs in parallel, exchanging boundary data at the appropriate times.

Fig. 1 illustrates, in a simplified form, the steps taken to run parallel PHOENICS 3.5.1 and how the processes interact with each other during a parallel run. The user starts with a **Q1** file, and runs SATELLITE to generate the data required by Parallel EARTH (**`eardat`** and/or **`facetdat`** and/or **`xyz`**). Using **`mpirun`** a parallel run begins, where a number of processes are started and **`parexe`** (parallel EARTH executable) runs on each of these processes. In the illustration of Fig. 1, four processes are running (PROC-0 to PROC-3), and we assume 1-D (z-direction) domain decomposition. Command of **`mpirun`** depends on version of MPI and users should learn documentation of MPI to configure it on the target system.

In the multi-EARTH implementation, a single process controls the input/output, acting as a server to the other processes; this is usually the ‘Master’ process (PROC-0). The main input operation is to read the required data files from disk, and broadcast them to the other processes. Fig. 1 shows only the **`eardat,`** produced by SATELLITE, being read by PROC-0 and broadcast to the other processes. In parallel, each process extracts information specific to the sub-domain it belongs to and continues with the solution procedure.

The ‘eardat splitting’ generates data in memory (represented in the diagram by E-010101 to E-010104) which are read at the appropriate times by **`parexe`**. After the usual initialisation and memory allocation etc., done by each **`parexe`**, the solution begins and data are exchanged between processes at appropriate times. At the end of the solution run, each process produces field data, which is a sub-set of the data from the whole domain.

The data from each process are sent to the Master process which are assembled by the Master, and written on the disk as one **`PHI`** file, exactly as in a sequential run (an option for writing individual PHI files also exists). A single **`result`** file is also written by the Master process (note that an option for writing individual result files also exists).

The objective is to develop a generic parallel version of EARTH, which can be ported easily to any message passing Distributed (or Shared Memory) System, and obtain satisfactory parallel efficiency without compromising on the numerical efficiency (i.e. needing more sweeps for convergence compared to a sequential run).

To preserve maintainability as much as possible, a single version of PHOENICS has been developed which will be able to run on both serial and parallel platforms. To achieve portability of the code, we have separated the computation from the communication. The communication functions use communication primitives specific to the target system, in this case MPI calls.

The main design issues considered for the parallel version are:

1. Input/Output operations
2. Domain decomposition and sub-domain links
3. Patch and VR-Object decomposition
4. The Linear Equation Solver and its parallelisation

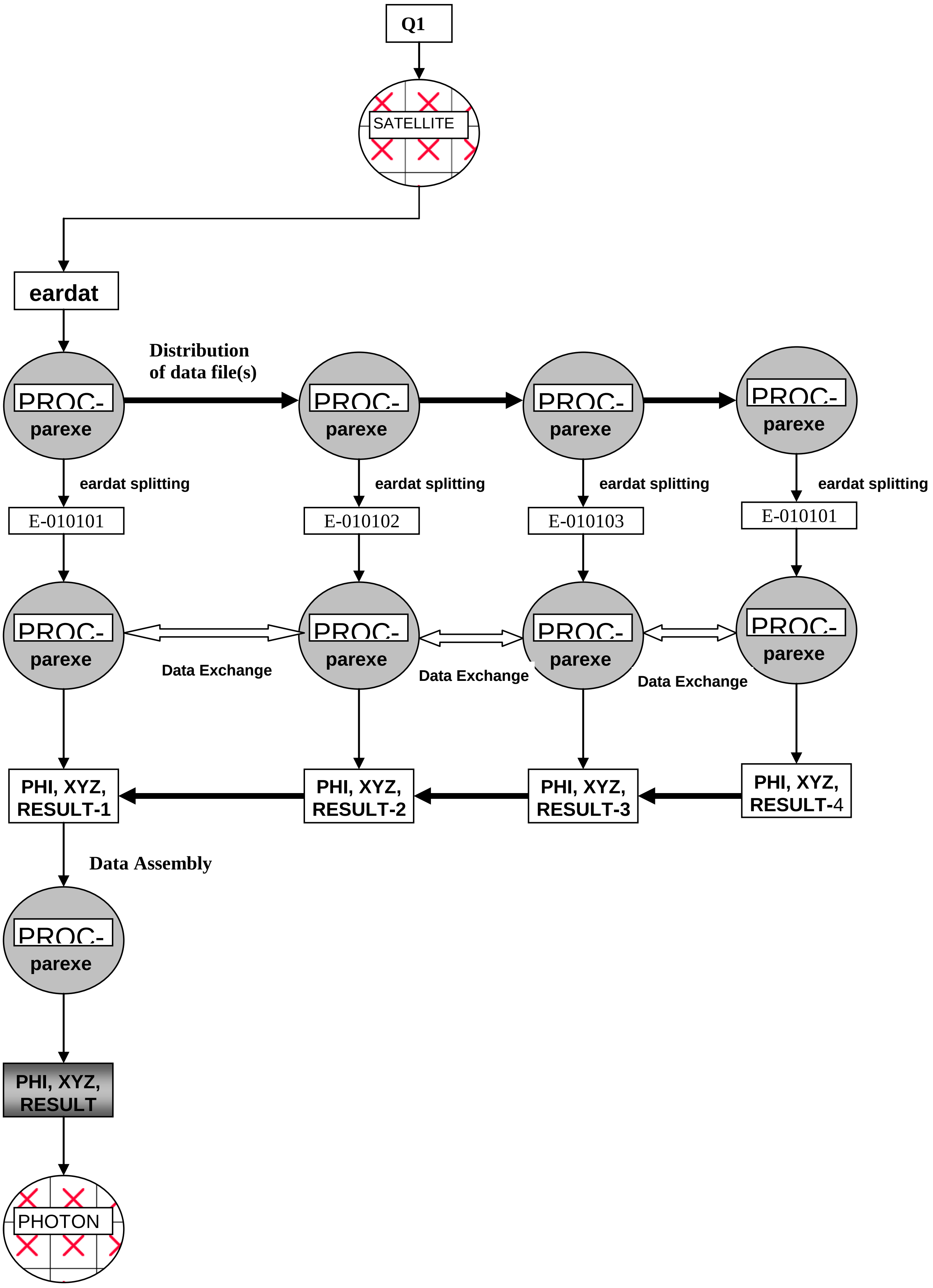


Figure 1: A simplified structure of Parallel PHOENICS

2.2 How Parallel EARTH differs from the Sequential EARTH

The block diagram below (Fig. 2) depicts the architectures of the parallel and sequential EARTH. There is common code in both versions but calls to the 'Parallel interface SUBROUTINES' will be made only if certain conditions are satisfied. Besides it is possible a sequential run is made but the solver used is the parallel solver executing on one processor.

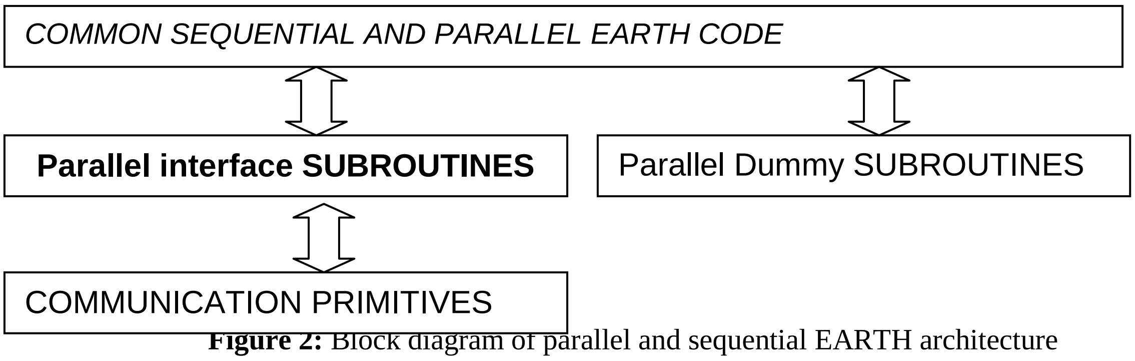

COMMON SEQUENTIAL AND PARALLEL EARTH CODE
Parallel interface SUBROUTINES
Parallel Dummy SUBROUTINES
COMMUNICATION PRIMITIVES

Figure 2: Block diagram of parallel and sequential EARTH architecture

2.3 Input/Output

As it was briefly described in 2.1, the multi-EARTH approach was adopted for this port; this implies that the tasks involved in the sequential EARTH are replicated over the available processes, with one exception, the I/O process. For default run, only one of the processes is responsible for the Input/Output operations and the file management. There is the option however (MULRES=ON) in the **`cham.ini`** file, when all the processes can open and write to their own RESULT files; or, for parallel RESTART, each process can dump its own PHI files. We usually set the MULRES flag to ON, when we require to print debugging information on the individual RESULT files.

For the default set-up, the main input operation is for the MASTER process, to read the various configuration files (**`prefix`**, **`config`**, **`earcon`**) and data file(s) produced by SATELLITE (**`eardat`**, **`facetdat`** and/or **`xyz`**). The MASTER passes full copies of these files to the other processes. In parallel, each process extracts an EARDAT (for BFCs the XYZ also), specific to its own sub-domain and begins the solution procedure. The **`facetdat`** is not modified by the processes, and it represents data for the whole domain.

At the end of the solution run, each process produces field data, which are assembled by the MASTER, reconstructed into one set of data for the entire problem and written on the disk as a single PHI file. For parallel RESTARTs, instead of the MASTER writing a single PHI file for the whole domain, each process writes its own PHI file which it can read back when a restart takes place.

Other input/output functions of the serial PHOENICS, involving files written on the disk, when running a problem out-of-core, have been disabled for the simple reason that these operations will slow down the parallel program.

2.4 Multi-domain link and communication

In order to solve a problem using domain decomposition, it is necessary to provide appropriate links between the sub-domains being solved in parallel by different processes. For a regularly structured grid, several alternative partitioning strategies are available (seven in total):

1-D decomposition in x, y or z direction;
2-D decomposition in xy, yz, or xz plane;
3-D decomposition.

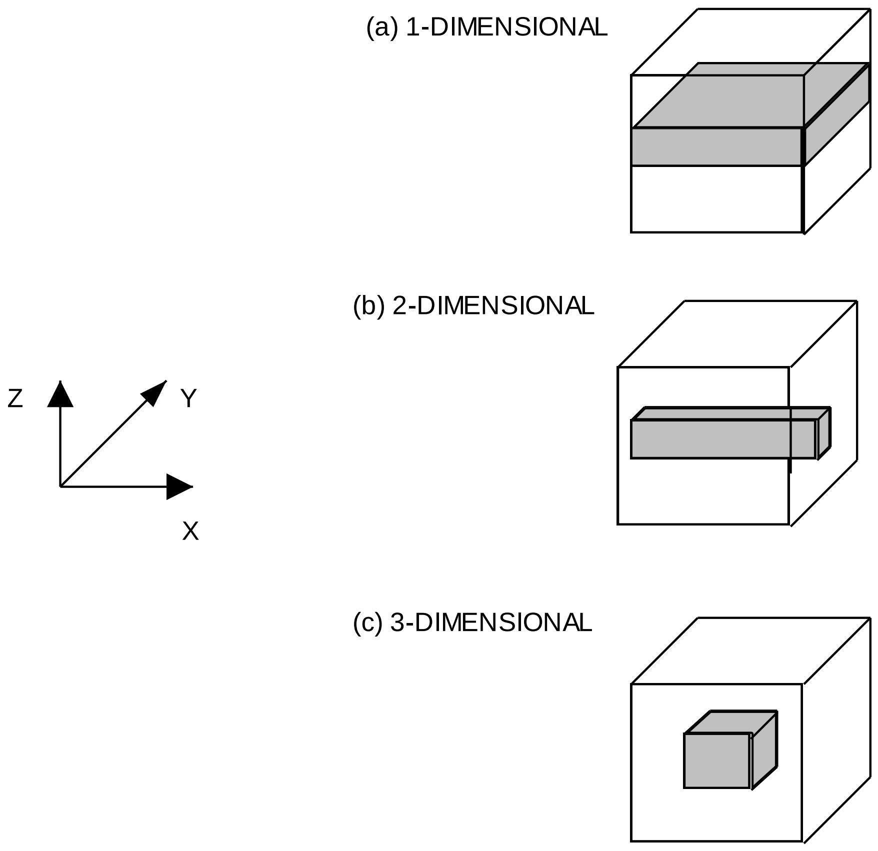


Figure 3: 1D, 2D and 3D domain decomposition

To link the sub-domains together, each process needs data from the adjacent sub-domains. When a decomposition (splitting) takes place the resulting grid is extended by two cells (or lines or planes) in each direction, for storing boundary data and also provide the correct interface data which are calculated by EARTH.

Figures 4 and 5 illustrate how the sub-domains are linked together (for 1D and 2D domain decompositions respectively), via data exchange of the boundary data in each split direction. If the grid were not extended, it would be necessary to calculate all the information on the interfaces separately, which would make the task very difficult if it was to provide all the EARTH physical and computational models.

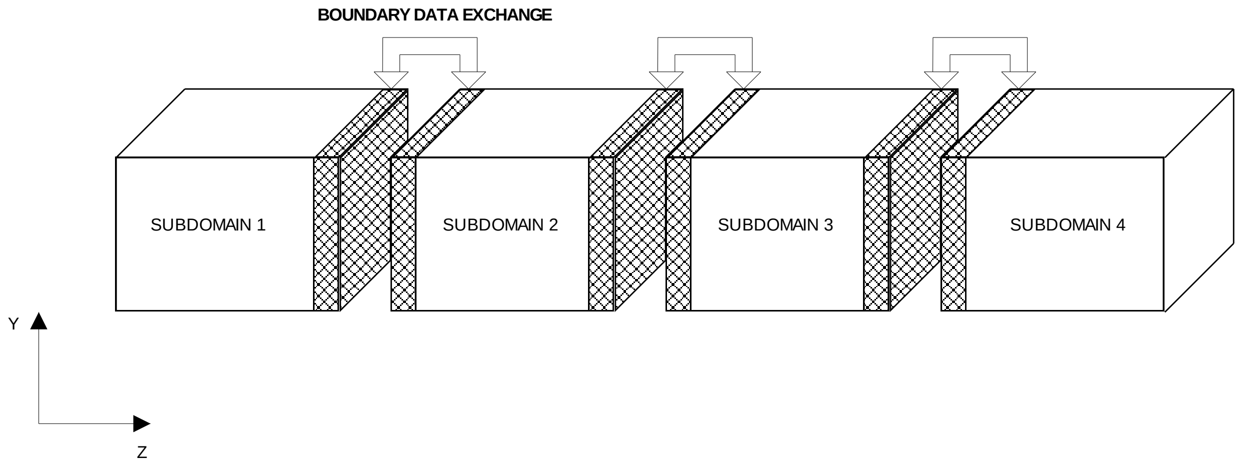


Figure 4: One-dimensional partition and data exchange

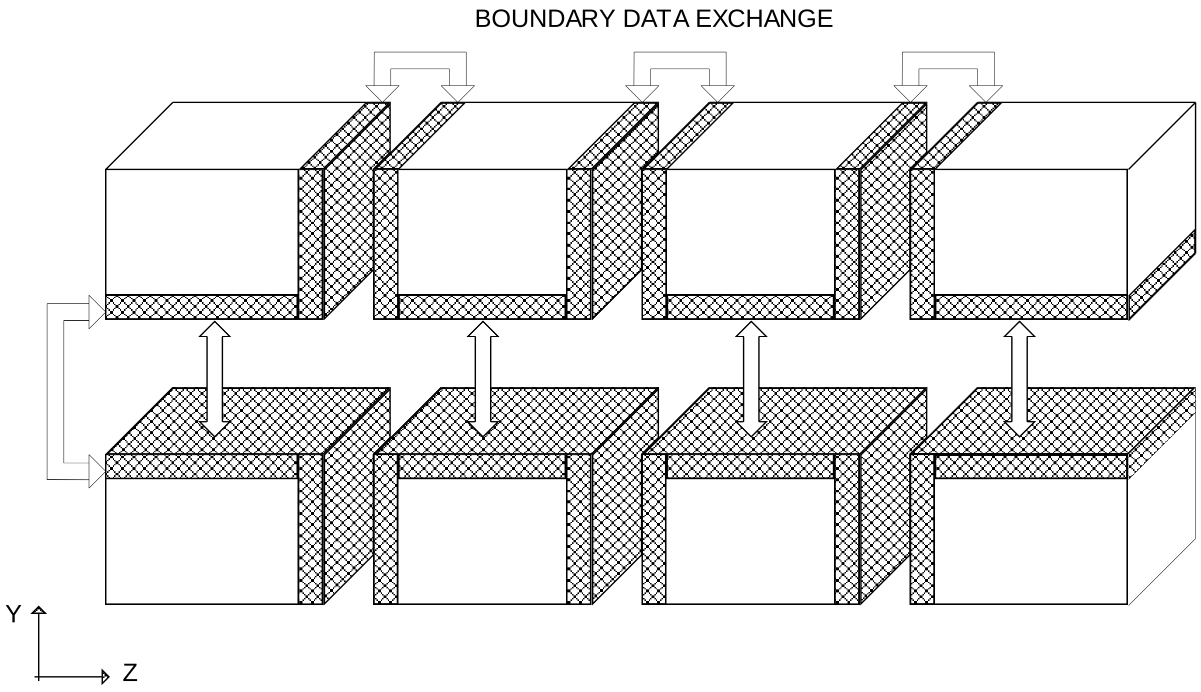


Figure 5: Two-dimensional partition and data exchange

2.5 Domain Decomposition Algorithm

We are interested in the partitioning of the global domain of size (`NX,NY,NZ`) into NPROC sub-domains, with the constraints that each sub-domain will have the same load (no. of cells), and minimise communication between sub-domains by minimising the surface/volume ratio between sub-domains.

The strategy is quite simple:

1. Factorise NPROC, so that you have a product of prime numbers; the product may contain the same prime number more than once. (NPROC=np1*np1*np2*np3)
2. Split the longest direction first by dividing with the largest factor;
3. Establish which direction is now the longest, and split the new longest direction by dividing with the next largest prime number;
4. Repeat 3 until all factors of NPROC have been used.

To bypass the automatic domain decomposition, the user can define his/her own sub-domains by making the following settings in the Q1 file:
LG(2)=T

IG(1)=4	(No. of sub-domains in x-direction)
IG(2)=3	(No. of sub-domains in y-direction)
IG(3)=5	(No. of sub-domains in z-direction)

2.6 Patch and VR-Object decomposition

Each process make decomposition of all patches after decomposition of whole domain. Correction of BoundBox (ixF,ixL,iyF,iyL,izF,izL) is provided for each patches. For this purpose section of patch BoundBox and current Sub-Domain is founded. And the section of patch is bounded in Sub-Domain and is used as new patch only for current Sub-Domain of current process. If section is empty then current Sub-Domain has not this patch. It is demonstrated on Fig. 6.

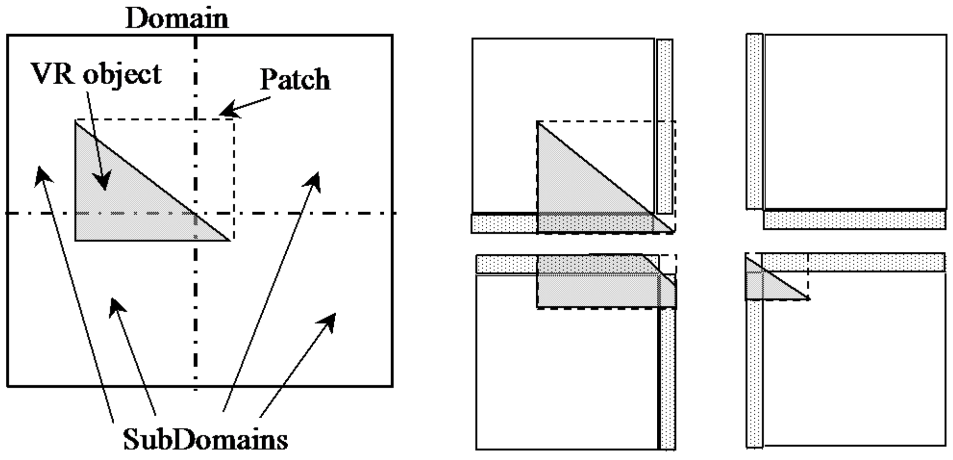


Figure 6. Patch decomposition

We should note that described patch decomposition method is not implemented in PHOENICS 3.5.1 it will be done in PHOENICS 3.6.

2.7 The Linear Equation Solver

Grid-partitioning schemes are very efficient only when the sub-domains are tightly coupled, ensuring that the number of iterations for a converged solution does not increase when using more than one processor. To achieve this, it is necessary to couple the sub-domains at the lowest possible level, i.e. within the Linear Equation Solver (LES). The standard PHOENICS LES is very efficient for sequential runs, but its recursive structure poses great difficulty for implementation on parallel computers.

The need for an easily parallelisable LES led to the investigation of a number of acceleration techniques based on conjugate direction methods, including the Conjugate Residual method. This acceleration procedure is used with the classical 2-step Jacobi iterative relaxation procedure. On its own, the 2-step Jacobi method has a poor rate of convergence, but when combined with the Conjugate Residual acceleration, convergence rate is enhanced significantly by changing the strategy of the solution search.

The Conjugate Residual acceleration technique is based on the algorithm used by Amsden and O'Rourke and we called it Conjugate Residual (CR) solver. The Conjugate Residual method is used as an acceleration technique for the classical point Jacobi and the m-step Jacobi iterative procedures.

2.8 Parallelisation of the Linear Equation Solver

The parallelisation of the Conjugate Residual method (using the m-step Jacobi preconditioning) is straightforward. The domain-decomposition approach caters for the necessary storage required for the linking of neighbouring domains. On the parallel systems the values of variables can be determined by transferring the correct values in the boundary cells. Then all the operations can be done independently on each processor.

2.9 Sub-domain Links

PHOENICS uses the SIMPLEST algorithm to link the hydrodynamic equations. SIMPLEST computes velocity components in a slabwise fashion and this is a disadvantage when one tries to parallelise the algorithm in the z-direction. However PHOENICS can also solve velocities WHOLEFIELD and this is the preferred setting when we solve problems in parallel. Both options for velocities are supported in parallel.

The linking of sub-domains has been designed bearing in mind that the parallel calculations should not differ from the serial calculations. Therefore in the parallel version, no changes will be made to relaxation parameters or any other settings that influence the solution procedure already set for a serial computation.

Additionally a converged solution should be obtained with approximately the same number of SWEEPS as obtained with the serial algorithm. Note that this can only be true when velocities are solved wholefield, and although convergence can occur when solving velocities slabwise (when z-wise decomposition is involved), more sweeps are usually needed.

In its most simplified form (1-D decomposition) the linking of sub-domains works as follows:

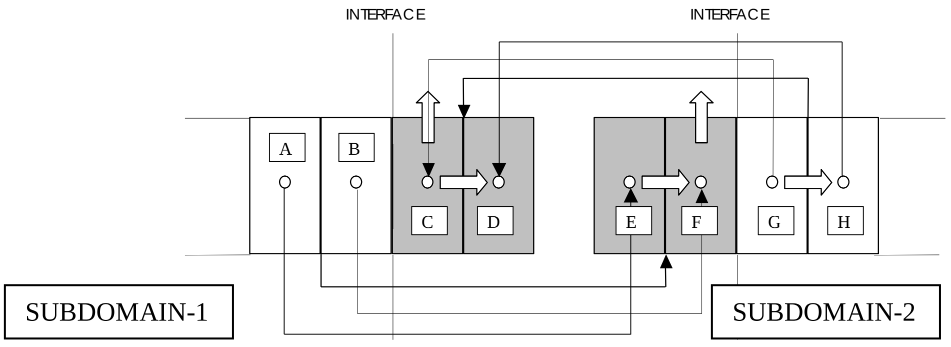


Figure 7: Domain links

1. Extend the grid by two slabs, for the purpose of storing information from the adjacent sub-domains. Cell C contains values taken from G, cell D contains the scalar value taken from H, cells E and F contain values taken from A and B (illustration in Fig.7).
2. Information is exchanged as described above, by placing the values of G and H, taken from sub-domain 2, in the correct position in the F-array of subdomain-1 which corresponds to the C and D cell store. Similarly values from cells A and B (from subdomain-1) are placed in the correct position in the F-array of subdomain-2 occupied by the elements E and F respectively.
3. Each processor will calculate the coefficients for each domain for all slabs, including the dummy slabs.
4. All the variables are solved from IZ=1 to NZ-2, for subdomain-1, from IZ=3 to NZ-2, for subdomain-2, and from IZ=3 to NZ-2 for subdomain-3. Similarly in the other directions (x,y).
5. The Conjugate Residual Solver is used, and for these variables solved 'wholefield', values are exchanged at each iteration.

6. For velocities, if solved 'slabwise', values are exchanged at each SWEEP; however velocities are not solved at the dummy slabs. Note that velocities can be solved either ‘slabwise’ or ‘wholefield’, although the preferred setting for parallel is wholefield.

2.10 Data exchange between processes/sub-domains

The data exchange between processes has been designed such as each process will exchange data in two steps in each direction. This is quite efficient and it is independent on the number of processors used. Depending on the topology used (see below), data will be exchanged in 2, 4 or 6 directions. For illustration purposes here we describe the data exchange process when a 2DGRID topology is used. We use the checkerboard approach.

In this approach we can imagine that the processes of a two-dimensional grid topology are arranged so that a black square is adjacent to white squares only, and vice-versa (see Fig.8). Having this kind of arrangement the processes that correspond to the black boxes say, can send simultaneously data and those that correspond to white boxes they receive. The opposite then happens afterwards, so data exchange takes place very fast. For this implementation synchronous communication was chosen. The above method can easily be extended to a three-dimensional grid topology when a three-dimensional domain decomposition is considered.

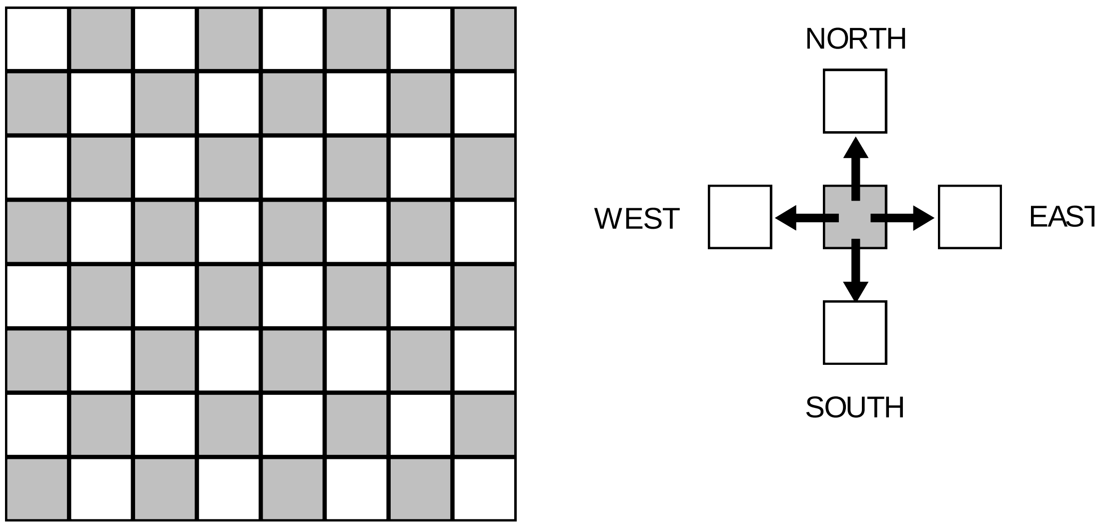


Figure 8: Two-dimensional data exchange

The algorithm to perform the data exchange is illustrated below.

1. Determine the location of the current processor on the grid, i.e. whether it belongs to a ‘black’ or ‘white’ box.

2. Determine whether there is a neighbour in any of those directions.

3. Start sending/receiving data if the processor belongs to a black/white box on the checkerboard.

3 About setting-up and running Parallel PHOENICS

Before you attempt to run Parallel PHOENICS, you must ensure that MPI (LAM for LINUX or MPICH.NT for Windows) has been installed and configured. If you are using PC-clusters you must also ensure that permissions have been set as required.

3.1 Automatic domain decomposition

There is no difference in running sequential or parallel PHOENICS. You can take a standard **Q1** file and run SATELLITE as usual. Then using one of the provided run-scripts, (run1, run2, run4, run6,…run16) you can run on the number of processes specified by the run-scripts. For example:

1. Go to **`d_priv1`** directory, run SATELLITE and load case-100;
2. Sure that domain can be spitted on selected number of parts;
3. Run, for example, on 4 processors, by invoking run4;
4. After the Parallel-EARTH run, you will have a single PHI file and a single RESULT file, as in sequential EARTH.

The output from the Parallel-EARTH is as follows:

```
There are no S-DOMAIN settings in X-direction.


There are  2 sub-domains in Y- direction.


There are  2 sub-domains in Z- direction.


There are  4 SUB-DOMAINs defined.

 The ID of this node is...
 46376908
[More output from EARTH…………………..]
```

First, the domain decomposition software, divides the domain into 4 sub-domains, and then based on this decomposition, it continues the run.

3.2 Manually specified sub-domains in Q1

It is also possible to by-pass the automatic domain decomposition algorithm, and specify how you want to decompose the main domain into sub-domains, by setting the following variables in the Q1 file. For example, to split the domain into 8 sub-domains (2 in each direction), the following settings must be included in the Q1 file:

LG(2)=T
IG(1)=2
IG(2)=2
IG(3)=2

The logical LG(2), will instruct the splitter to by-pass the automatic domain decomposition, and split the domain according to the settings defined in the IG array.
IG(1) specifies the number of sub-domains in the x-direction;
IG(2) specifies the number of sub-domains in the y-direction;
IG(3) specifies the number of sub-domains in the z-direction;

In this case, the domain has been divided into sub-domains according to the settings made in the Q1 file.

3.3 Settings that can influence Parallel PHOENICS

Parallel RESTART

In order to enable a RESTART in parallel PHOENICS, you must save the PHI files from each sub-domain separately. This can be effected by setting LG(1)=T in the Q1 file.

This flag forces each processor to create its own PHI file, which corresponds to its own sub-domain with a distinct name specifying the processor that created it. After saving the PHI files, we can use the RESTRT command as usual.

For example, if there are 6-processors available for a run, 6 PHI files will be created with the names phi001, phi002, phi003, phi004, phi005, phi006. The numbers are used so each processor will know which file to read when entering a RESTART run. Hence, phi001 was created by processor 0, phi002 by processor 1, and phi006 by processor 5. The numbers do not necessarily correspond to the respective sub-domain number.

The CHAM.INI file

The CHAM.INI file provides an entry for Parallel PHOENICS, which can effect the solution and output. The following 'flags' can be set to ON or OFF under the `[PSolve]` section, namely,

```
[PSolve]
SEQSOLVE = OFF
MULRES = OFF
```

If **`SEQSOLVE=ON`** then a sequential run is made.
If **`MULRES=ON`** then multiple **`RESULT`** files are created, i.e. each process creates its own **`RESULT`** file.
By default all the flags are set to **`OFF`**.

The variables of environment

User can use environment variables to control calculations:

Variable **`PHOE_SOLVE_PAR`** controls type of used solver. If `PHOE_SOLVE_PAR` > 0 and user run one process, then PHOENICS will use parallel solver (CR LES). In the other cases it will be used sequential one.

Variable **`PHMULRES`** handles creation of multiple **`RESULT`** files as variable **`MULRES`** in CHAM.INI file.

4 Testing and validation of Parallel PHOENICS

This chapter describes the testing procedure of Parallel PHOENICS and some validation tests with comparison of results.

4.1 Method of testing

Preliminary notes

The most number of tests were made for stable cases therefore below in list of tasks the transient cases have special mark. We used cases with VR-objects during testing but its were a simple VR-objects which don't use information from file `facetdat` and we excluded cases when handling of facets was required. Most number of tests was made for Cartesian grid (2D or 3D) other cases are marked specially in list below.

All calculations were made on one computer with operational system Windows 2000 with MPICH version 1.2.5. The purpose of calculations is debugging and validation. Benchmark tests were not made.

List of tests

1. Linear 2D and 3D heat conducting without VR-objects. Several cases having various number of control volumes on each direction. Boundary conditions was set by usual patches.
2. Non-Linear 2D and 3D heat conducting without VR-objects. Several cases having various number of control volumes on each direction. Boundary conditions was set by usual patches.
3. Linear 2D and 3D heat conducting with simple VR-objects.
4. Straightforward laminar flow without VR-objects.
5. Straightforward laminar flow with VR-objects, including divided objects by bound of Sub-Domain.
6. Laminar flow in complicated configurations of objects with vortex.
7. Turbulent flows (LVEL and K-E models) with VR-objects, including divided objects by bound of Sub-Domain.
8. Laminar and turbulent flows (LVEL and K-E models) with VR-objects and heat exchanges.
9. Transient laminar flow with heat exchanges.
10. Stable and transient laminar flow in polar grid.
11. Laminar flow in domain with BFC grid.

Method of outcomes comparison.

All calculations was provided for each cases using several methods:

- Using sequential default PHOENICS solver (triaxial sweep solver);
- Using sequential CR solver which used in Parallel PHOENICS;
- Parallel calculations with several variants of domain decomposition.

Decompositions of domain was made in each direction (X,Y and Z) separately and in various combinations. Used number of processes was from 2 to 8.

Calculation parameters (parameters of relaxation, number of sweeps) was selected such that all tests have the exact solution. Comparison of results was made using special code for selected variable (temperature, pressure, component of velocity and so on). The basic solution was solution provided by sequential default PHOENICS solver.

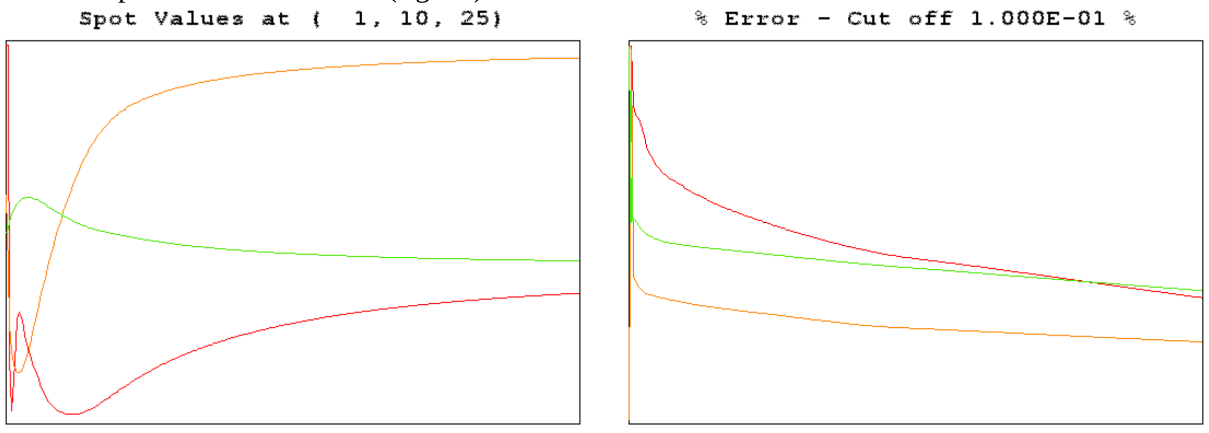
4.2 Outcomes of testing

The most cases have relative error less than 0,01 % between variants of calculation in the same case.

First tests for simple tasks (heat conducting single equation) show that CR solver gives not bad convergence than user can wait. But its capability to provide converged solution is poor than sequential default PHOENICS solver has. Therefore, for using the same number of sweeps users should to increase number of internal solver iterations (parameter `LITER`) and reduce condition of termination of internal iteration (parameter `ENDIT`).

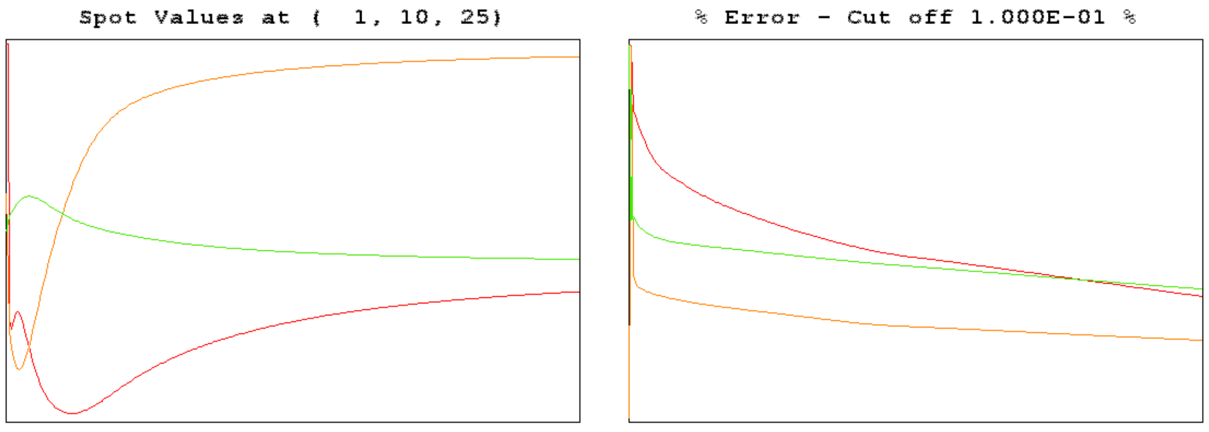
Calculations of hydrodynamics tasks are shown that very often CR solver gives the same convergence rate as default solver and sometime convergence rate was better. In any case, almost always we can get the exact solution using the same number sweeps as default solver by using exact setting of parameters `LITER` and `ENDIT`.

The Figure 9-11 demonstrates the process of convergence for real PHOENICS Library case 250 for sequential EARTH (fig. 9), parallel solver on one process (fig. 10) and parallel solver on 5 processes with decomposition in Z direction (fig. 11). The case calculates laminar flow in 2D area.



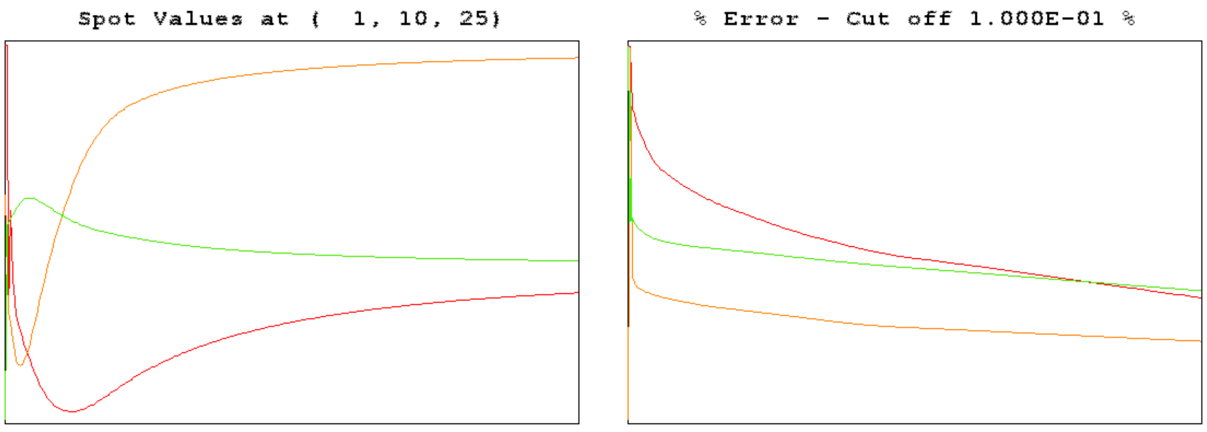
Min	Max	Spot Value	Change	Variable	Max	% Error	Change
-4.00E-01	0.00E+00	-2.65E-01	1.04E-04	P1	1.00E+03	2.89E+00	0.00E+00
-3.00E-01	2.00E-01	1.83E-01	8.91E-06	V1	1.00E+14	1.25E+02	0.00E+00
0.00E+00	2.00E+00	8.47E-01	5.36E-07	W1	1.00E+07	7.74E+01	0.00E+00

Figure 9



Min	Max	Spot Value	Change	Variable	Max	% Error	Change
-4.00E-01	0.00E+00	-2.65E-01	1.07E-04	P1	1.00E+03	2.87E+00	0.00E+00
-3.00E-01	2.00E-01	1.84E-01	9.12E-06	V1	1.00E+14	1.26E+02	0.00E+00
0.00E+00	2.00E+00	8.47E-01	5.96E-07	W1	1.00E+07	7.75E+01	0.00E+00

Figure 10

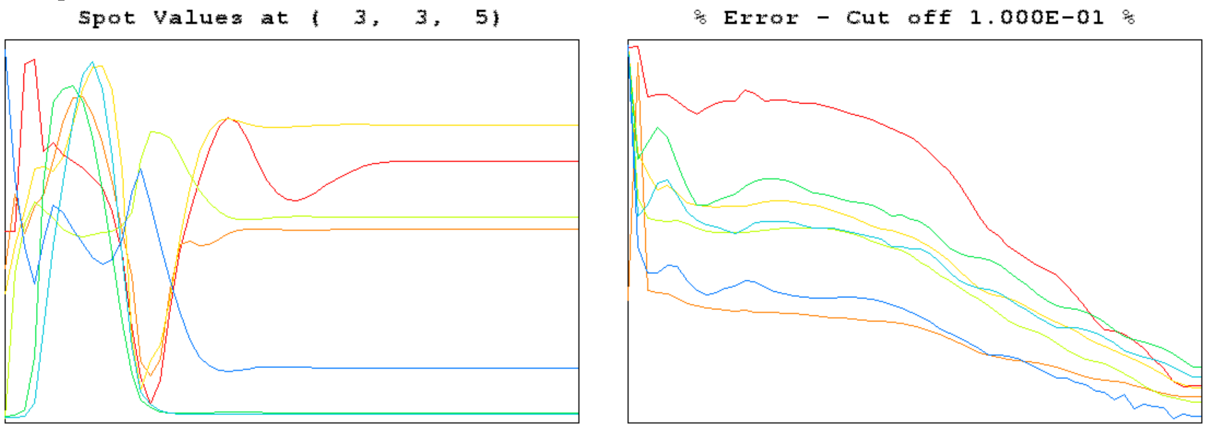


Min	Max	Spot Value	Change	Variable	Max	% Error	Change
-4.00E-01	0.00E+00	-2.65E-01	1.02E-04	P1	1.00E+03	2.88E+00	0.00E+00
-3.00E-01	2.00E-01	1.84E-01	5.03E-05	V1	1.00E+14	1.32E+02	0.00E+00
0.00E+00	2.00E+00	8.46E-01	-1.20E-04	W1	1.00E+07	7.67E+01	0.00E+00

Figure 11

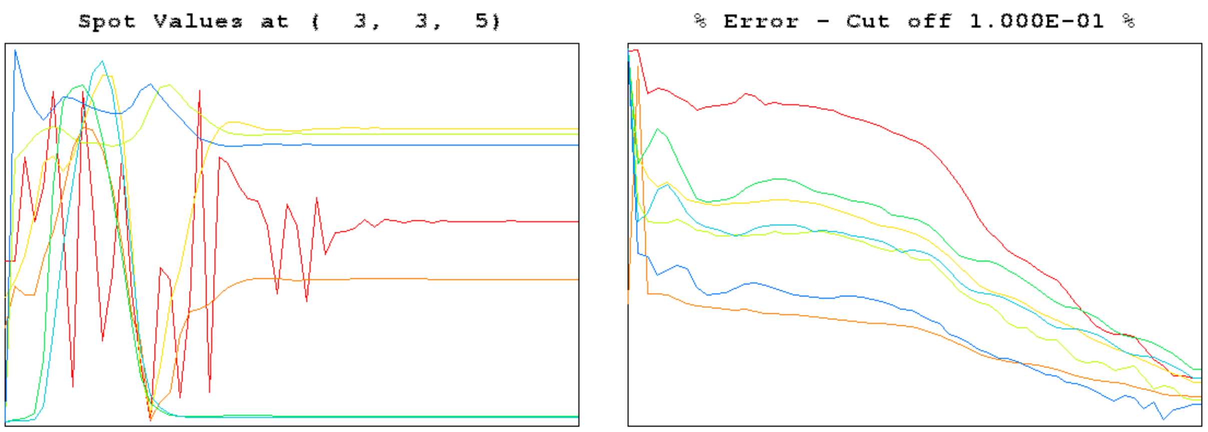
You can see that process of convergence is almost the same for all variants. Small differences were in the beginning. We have used the same Q1 file for calculations. Although the solution is not converged fully the fields of variables are differed very small (less 0,1 %). We can demonstrate the same pictures for other decompositions in other directions.

The Figures 12-13 demonstrates outcomes of computational test for more complicated case PHOENICS Library 273. It is turbulent flow with heat exchange. The figure 12 shows results for sequential EARTH and figure 13 shows outcomes for parallel solver for two processes with decomposition in Z direction.



Min	Max	Spot Value	Change	Variable	Max	% Error	Change
-2.00E+01	2.00E+01	7.49E+00	-1.41E-03	P1	1.00E+03	2.49E-01	0.00E+00
-2.00E+00	3.00E+00	5.23E-01	-3.52E-06	U1	1.00E+15	7.84E-01	0.00E+00
-1.00E+00	2.00E+00	1.35E+00	2.15E-06	V1	1.00E+06	4.16E-01	0.00E+00
1.00E+01	2.00E+01	1.54E+01	9.54E-07	W1	1.00E+06	2.05E-01	0.00E+00
0.00E+00	8.00E+01	1.13E+00	0.00E+00	KE	1.00E+06	1.07E+00	0.00E+00
0.00E+00	4.00E+03	4.18E+01	0.00E+00	EP	1.00E+08	9.75E-01	0.00E+00
7.00E+02	1.00E+03	7.40E+02	0.00E+00	H1	1.00E+09	1.12E-01	0.00E+00

Figure 12



Min	Max	Spot Value	Change	Variable	Max	% Error	Change
-3.00E+01	4.00E+01	7.49E+00	5.78E-04	P1	1.00E+03	3.44E-01	0.00E+00
-1.00E+00	3.00E+00	5.23E-01	-9.66E-06	U1	1.00E+15	1.11E+00	0.00E+00
-1.00E+00	2.00E+00	1.35E+00	-2.37E-05	V1	1.00E+06	6.55E-01	0.00E+00
0.00E+00	2.00E+01	1.54E+01	2.10E-05	W1	1.00E+06	2.70E-01	0.00E+00
0.00E+00	8.00E+01	1.13E+00	1.25E-05	KE	1.00E+06	1.09E+00	0.00E+00
0.00E+00	4.00E+03	4.18E+01	4.65E-04	EP	1.00E+08	1.12E+00	0.00E+00
0.00E+00	1.00E+03	7.40E+02	0.00E+00	H1	1.00E+08	2.73E-01	0.00E+00

Figure 13

You can see that process of residual changes is almost the same for both variants. But there are differences in variable behavior in the beginning of convergence process. We have used the same Q1 file for calculations. Although the solution is not converged fully the fields of variables are differed very small (less 0,01 %).

The shown samples demonstrate the fact that parallel PHOENICS can give correct solution for many cases.

Analysis of many tests is shown that *wholefield* method of solution for any variables is preferred than *slabwise*. The *slabwise* method can give wrong converged results at decomposition in Z direction (see 4.4). If user want to use *slabwise* method then he should use decomposition in X and/or Y directions only.

Results of tests are shown also that character of convergence and error in provided solution can depend on decomposition type (number of Sub-Domain and directions of decomposition) although differences between calculation results are enough small.

4.3 What tasks can Parallel PHOENICS solve now and what tasks can not yet

List of tasks, which can be solved by Parallel PHOENICS 3.5.1, is large. Item 4.1 contains not full list, therefore you can see below list of tasks which Parallel PHOENICS can not solve yet because not all features of sequential PHOENICS is implemented now. But we hope they will be realized in future.

- Using VR-objects with facets;
- Using EXPERT;
- Using PARSOL;
- Using any multi-block and multi-domain methods (Fine Grid Embedding, multi-domain BFC grid and etc);
- Using GENTRA;
- Using CCM/GCM;
- Using Surface to Surface Radiation;
- Using Advance Multi-Phase Models of Flows (IPSA,HOL,SEM);
- Using MOFOR.

4.4 Known bugs

List of known bugs founded during testing but not fixed yet is shown below.

- *slabwise* method gives sometimes wrong converged solution if user uses decomposition in Z direction (or in combination Z and other direction);
- tasks with cyclic boundary conditions can not be correct solved if user uses decomposition in X direction;
- tasks using Darcy hydrodynamics can give wrong solution;
- in some cases during calculation in polar grid the solution is wrong if flow has significant value of the X-component of velocity.

5 Further work on Parallel PHOENICS 3.6

The main purpose of development Parallel PHOENICS 3.6 is to provide that the all features of sequential PHOENICS work properly in parallel.

The main direction of activity at present connected with:

- Changing of internal structure of PAREAR;
- Creation of flexible decomposition and providing the possibility to work with Sub-Domain having various sizes;
- To provide properly work of PARSOL and any VR-objects;
- To add features, which implemented in sequential PHOENICS but don’t work in parallel.

5.1 Internal structure of PAREAR

Parallel PHOENICS 3.5.1 has implementation when main process read **eardat** file and broadcast it to other processes each of them selects needed information from **eardat** file. The realization of such approach contains the many repetitions in source code having similar parts for parallel and sequential PHOENICS. Therefore if developer changed any part of source code then he/she should change and parallel part of code. It is not suitable and lead to development slowdown for Parallel PHOENICS. Therefore the repetition part will be deleted in parallel and sequential code and both versions will integrated in one EAREXE.

At start of EAREXE in parallel:

1) Main process make decomposition and create for other processes needed parts of **EARDAT**, **XYZ** and **FACETDAT** files. Other processes will wait while main process prepare all data;
2) The main process broadcast only needed information for each processes;
3) All process run as sequential;
4) The single difference form sequential code is exchange of data at appropriate time between Sub-Domains on the bounds and colleting by main process the results of calculations.

5.2 Dividing on various Sub-Domains

User will can use Sub-Domains having various number of cells in Parallel PHOENICS 3.6. It is necessary to create file **PARDAT** in working folder of main process for the purpose. The file should have the next structure:

NSDX - number Sub-Domains in X direction
NSDY - number Sub-Domains in Y direction
NSDZ - number Sub-Domains in Z direction
NXS(1..NSDX) – array containing number of cells into Sub-Domains in X direction
NYS(1..NSDY) – array containing number of cells into Sub-Domains in Y direction
NZS(1..NSDZ) – array containing number of cells into Sub-Domains in Z direction

Sum of cells along each direction should be equal all number of cells in this direction.
Summa(NXS(I)) = NX, Summa(NYS(I)) = NY, Summa(NZS(I)) = NZ

5.3 Decomposition of VR-objects and PARSOL

Decomposition of Domain into Sub-Domain is accompanied by splitting of **`facetdat`** file. The leading idea of algorithm of patch splitting is next (see Fig. 6 in item 2.6):

1) From file **`facetdat`** we get information about BoundBox of whole patch, split it on truncated BoundBox according to placement of Sub-Domain bounds;
2) Facets of VR-object are corrected using truncated BoundBox and Sub-Domain bounds. This procedure should be analogous to utility FACETFIX.

Such procedure of patch decomposition allows applying standard algorithms of sequential EAREXE in other places of source code. More detailed discussion of this topic you can find below.

5.3.1 What are VR-objects?

VR-object is 3D (body) or 2D (plate) geometry object bounded by enclosed surface. VR-objects are used in PHOENICS for description of bounds (INLET, OUTLET and PLATE objects for example), solid areas (BLOCKAGE), areas having sources (PRESSURE_RELIEF, USER_DEFINED) and so on. For determination of VR-object PHOENICS uses the next manner: enclosed surface is represented by set of plane objects (triangle or quadrangle) – facets. We shall name object, having facets, as VR-Facetted object. Set of facets for all objects of task is saved in file **`facetdat`**.

Program EAREXE transforms VR-Facetted object to VR-patch as it is shown on Fig. 14.

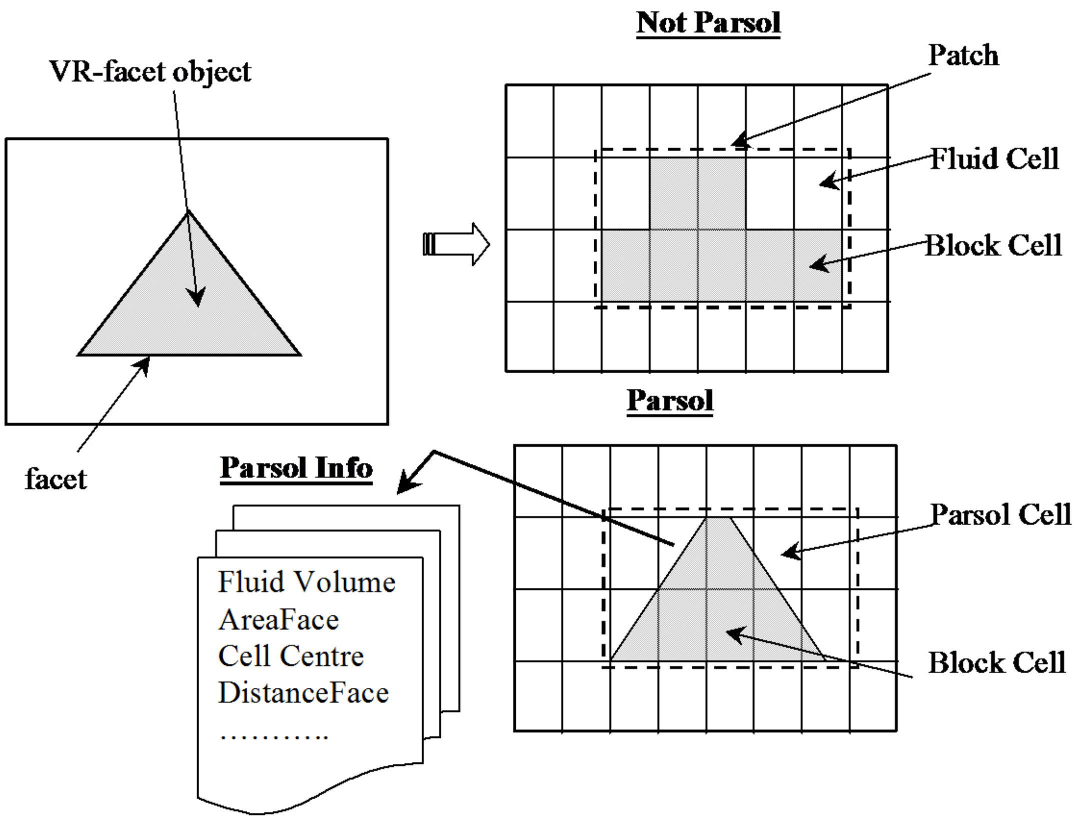


Figure 14

Internal structure of VR-patch depends on whether PHOENICS uses PARSOL or not. If it is non-PARSOL case then structure of VR-patch is next:

- Usual patch of PHOENICS, circumscribed around VR-Facetted object (BoundBox);
- Array of bit flags for each cells, indicated the FLUID cells and blocked cells.

If PHOENICS uses PARSOL then VR-patch contains also information about placement and parameters cut-off cells (PARSOL cells). Part of this information is stored in **`pbcl.dat`** file to draw PARSOL cells in post-processor. It is necessary to mark that the algorithm for transformation from VR-Facetted objects to VR-patch is very complicated. Therefore, many efforts were made and will be made to create effective algorithms for work with VR-Facetted objects and VR-patchs (essentially in connection with MOFOR).

5.3.2 VR-objects in sequential and parallel code

EAREXE uses the next way to handle of VR-objects (see Fig. 15) before calculations

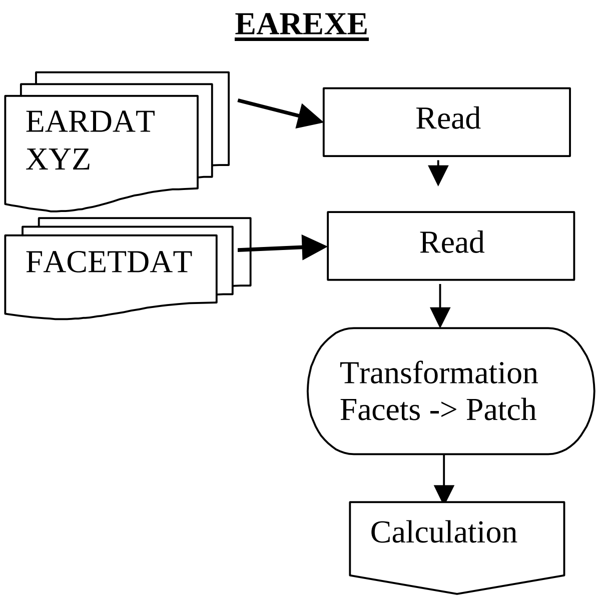


Figure 15

The main idea of parallel PHOENICS contains the next steps:

- To divide Global Domain on Sub-Domains;
- To produce parallel calculations in each Sub-Domain using copies of parallel PAREXE on each node, calculations should have synchronization;
- Synchronization is exchange information on bounds of Sub-Domains.

During dividing Global Domain on Sub-Domains it is necessary to make splitting all objects depending on coordinates, for example, usual patches and VR-objects. Splitting of standard patches we can make very simple. We can edit BoundBox for each patch in each Sub-Domain according to bounds of each Sub-Domain. To handle VR-object it is possible to use two approach:

1) to make transformation VR-Facetted object to VR-patch into Global Domain as sequential EAREXE works and after VR-patch can be spitted during Domain decomposition;
2) to split VR-Facetted objects for each Sub-Domain (to make decomposition of Global **`facetdat`** file to set of Local **`facetdat`** files). Each process will handle own **`facetdat`** file as sequential EAREXE. The splitting of VR-Facetted objects is dividing enclosed surface by planes on enclosed surfaces in each Sub-Domain as it utility FACETFIX makes. (see Fig. 6 in item 2.6)

Algorithm of first approach is demonstrated on Fig. 16.

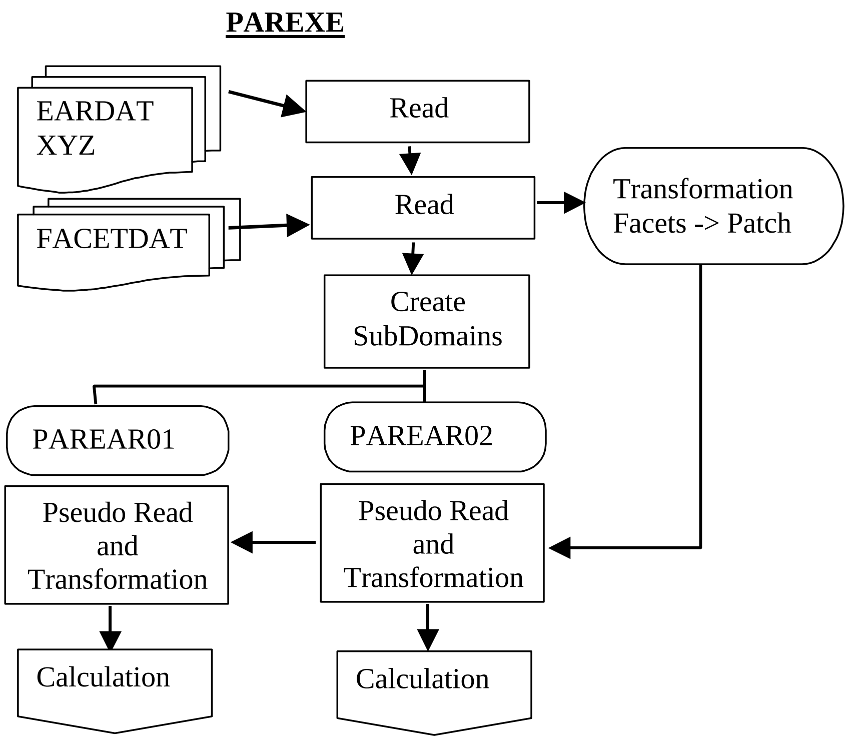


Figure 16

Algorithm of second approach is demonstrated on Fig. 17.

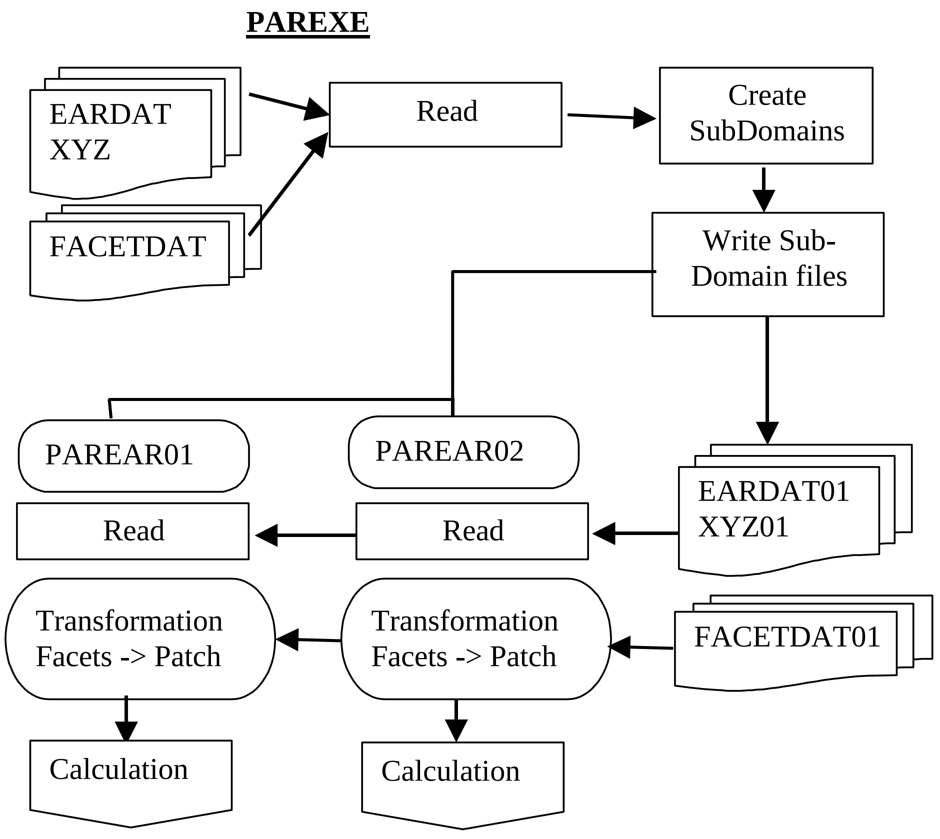


Figure 17

The first approach was realized in Parallel PHOENICS versions 3.4 and 3.5.1. The main disadvantage of the approach is duplication of transformation algorithm VR-Facetted object to VR-patch into sequential and parallel codes. And this duplication complexify the modification of this algorithm.

The second approach excludes the duplication and we can combine sequential and parallel code as following:

1) to create at start of program sub-block “starter” for splitting Global Domain on Sub-Domains and splitting **`eardat`**, **`xyz`** and **`facetdat`** files to local files for each Sub-Domain;
2) to insert into appropriated places the subroutines for exchanges information between Sub-Domains.

These sub-blocks of source code will not be used for sequential mode (Domain = Sub-Domain).

The last approach is realized now in PHOENICS v.3.6.