A Parallel Grid Based PSE for EHL Problems

Christopher Goodyer[1], Jason Wood[2], and Martin Berzins[1]

[1] Computational PDEs Unit, School of Computing,
University of Leeds, Leeds, UK,
{ceg,martin}@comp.leeds.ac.uk,
http://www.comp.leeds.ac.uk/cpde

[2] IRIS Explorer Centre of Excellence, School of Computing,
University of Leeds, Leeds, UK,
jason@comp.leeds.ac.uk,
http://www.comp.leeds.ac.uk/iecoe

Abstract. In this paper we explain how a Problem Solving Environment (PSE) can be constructed such that the computationally intensive sections are run using resources on the Computational Grid, operating remotely from the machine running the visualisation. The PSE in question is one solving demanding elastohydrodynamic lubrication problems from mechanical engineering. The numerical calculation is done here using both multilevel techniques and using parallelism. The global nature of the equation system behind the problem means that, given that this is a communications intensive calculation, the code scales remarkably well.

1 Introduction

A Problem Solving Environment (PSE) is a tool for allowing scientists and engineers to manipulate a numerical simulation whilst interactively visualising the results. There are three ways in which even basic PSEs are advantageous. These are that the input parameters can all be set, or adjusted at run time; the numerical solver is only one part of the PSE and hence it can be possible to change solution methods, if appropriate; and finally the visualisation is an innate component of the package, and results can be visualised and studied as the calculation proceeds. Computational steering gives the PSE another advantage over traditional solution methods because this allows the test problem and/or the solution methods, to be updated during the calculation. The user, thus, “closes the loop” of the interactive visually-driven solution procedure [1].

For a PSE to be at its most useful there must be sufficient opportunity for the user to influence the calculation in real time, rather than having to wait minutes or hours for the next result. One of the most common ways to accomplish this, without changing the numerical algorithms used, is by using parallelism on faster machines. These machines are unlikely to be located on the researcher’s desktop, but rather as a shared resource for many users.

The Computational Grid is a global distributed parallel programming resource. With many different large scale resources located across the world, the idea is that researchers can submit computational jobs without caring where the actual clock cycles are spent. By being able to get quick turnaround from calculations on larger machines than would normally be available it is envisaged that more demanding and more realistic calculations will be possible.

With a PSE it is necessary to perform the visualisation locally and hence running the PSE on a grid resource would not be computationally efficient, either in terms of visualisation speed or for computational time spent using the grid. Being able to run the calculation on the grid, though, would bring great benefits.

In this work a PSE for a real world engineering problem is explained. The *Carmehl* solver, explained in Section 2, solves elastohydrodynamic lubrication (EHL) problems. This solver has been included in PSEs as detailed in [2] where the relative benefits of building the environment in both IRIS Explorer and SCIRun are examined. *Eclipse*, the version running in IRIS Explorer, is briefly explained in Section 3. The methods used to Grid-enable it are described in detail in Section 4, in which it is assumed that Globus[1] is being used as the Grid management software.

The ability to use Grid resources for parallel computation jobs is very important in reducing run times. EHL solvers, to the best of our knowledge, have not made significant use of parallel computers. The multilevel schemes now employed in EHL solvers have complex global operations requiring significant communication between processors. The first significant results, beyond simple work sharing in less efficient algorithms as shown by Goodyer [3] and Arenaz *at al.* [4], are given in Section 5.

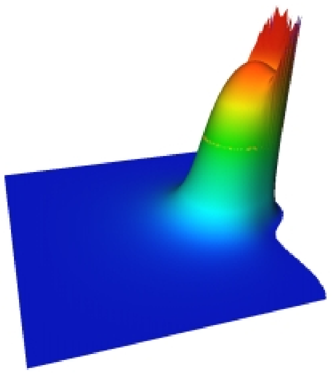
The paper is concluded in Section 6 where the applicability of these approaches are considered and future work directions suggested.

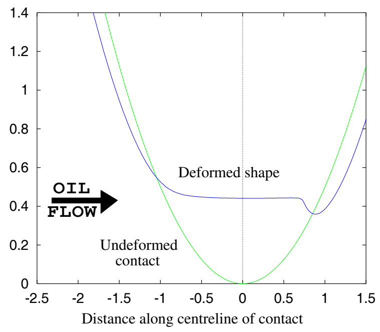
2 Elastohydrodynamic Lubrication

Elastohydrodynamic lubrication occurs in journal bearings and gears, where, in the presence of a lubricant, at the point of contact there is a very large pressure exerted on a very small area, often up to 3 G Pa. This causes the shape of the contacting surfaces to deform and flatten out at the centre of the contact. There are also significant changes in the behaviour of the lubricant in this area.

A typical solution profile is shown in Figure 1a for the pressure profile across a point contact, such as will be considered in this paper. This is the equivalent situation of a ball bearing travelling along a lubricated plane. With oil flow from left to right it can be seen that in the inlet region there is a very low pressure, in the centre of the contact there is a very high pressure, with an even higher pressure ridge around the back of the contact. Finally, in the outflow a cavitation region is formed where there is a void in the lubricant, and the pressure is assumed to be ambient there. The corresponding surface geometry profile for the bearing has the undeformed parabolic shape of the contact flattened in the high pressure area. The centreline solution is shown in Figure 1b.

[1] http://www.globus.org/


(a) Pressure solution



(b) Film thickness along the centreline

Fig. 1. Typical solutions across an EHL point contact

The history of the field is detailed out in papers such as [5]; much information about the numerical techniques currently used to obtain fast, stable solutions is given in both [6] and [3], the latter of which describes in great detail the precise methods used in the code employed in this work.

The EHL system being solved depends on many physical parameters concerning both the physical nature of the contacts, the properties of the lubricant used, the loading and the rotation speeds of the surfaces. The solution variables which must be solved for are the pressure profile P, across the domain, the surface geometry H, the viscosity $\overline{\eta}$ and the density $\overline{\rho}$. The full equation system is described in [6] and the appropriate references for the derivations are given therein. The pressure distribution is described by the Reynolds Equation:

$$\frac{\partial}{\partial X}\left(\frac{\overline{\rho}H^3}{\overline{\eta}\lambda}\frac{\partial P}{\partial X}\right)+\frac{\partial}{\partial Y}\left(\frac{\overline{\rho}H^3}{\overline{\eta}\lambda}\frac{\partial P}{\partial Y}\right)-\frac{u_s(T)}{u_s(0)}\frac{\partial(\overline{\rho}H)}{\partial X}-\frac{\partial(\overline{\rho}H)}{\partial T}=0\,, \qquad (1)$$

where u_s is the sum of the surface speeds in the X-direction at non-dimensional time T, λ is a non-dimensional constant, and X and Y are the non-dimensional coordinate directions, The standard non-dimensionalisation means that the contact has unit Hertzian radius, and that the maximum Hertzian pressure is represented by $P = 1$. The boundary conditions for pressure are such that $P = 0$. For the outflow boundary, once the lubricant has passed through the centre of the contact it will form a free boundary, the *cavitation boundary*, beyond which there is no lubricant. The non-dimensional film thickness, H, is given by:

$$H(X,Y) = H_{00} + \frac{X^2}{2} + \frac{Y^2}{2} + \frac{2}{\pi^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\frac{P(X',Y')\,\mathrm{d}X'\mathrm{d}Y'}{\sqrt{(X-X')^2+(Y-Y')^2}}\,, \qquad (2)$$

where H_{00} is the central offset film thickness, which defines the relative positions of the surfaces if no deformation was to occur. The two parabolic terms represent the undeformed shape of the surface, and the integral defines the deformation of the surface due to the pressure distribution across the entire domain. A conservation law for the applied force is also applied.

Since an isothermal, generalised Newtonian lubricant model is being used, only expressions for the density and viscosity will be required. The density model chosen is that of Dowson and Higginson, whilst the viscosity model used is the Roelands pressure-viscosity relation, again see [6].

The solution variables are discretised on a regular mesh. Once discretised, the equations are solved in turn to iteratively produce more accurate solutions. This process is described in detail in [6] where detailed descriptions of the need for, and benefits of multigrid and multilevel multi-integration are given.

The Fortran 77 software used here, *Carmehl*, is described by Goodyer in [3] and includes both variable timestepping for transient calculations and also has the option to use adaptive meshing. Conventionally, once execution is complete then output files of data for the key variables are produced which may then be post processed for visualisation purposes. The user may request that the output solution is saved for continuation purposes on a future run.

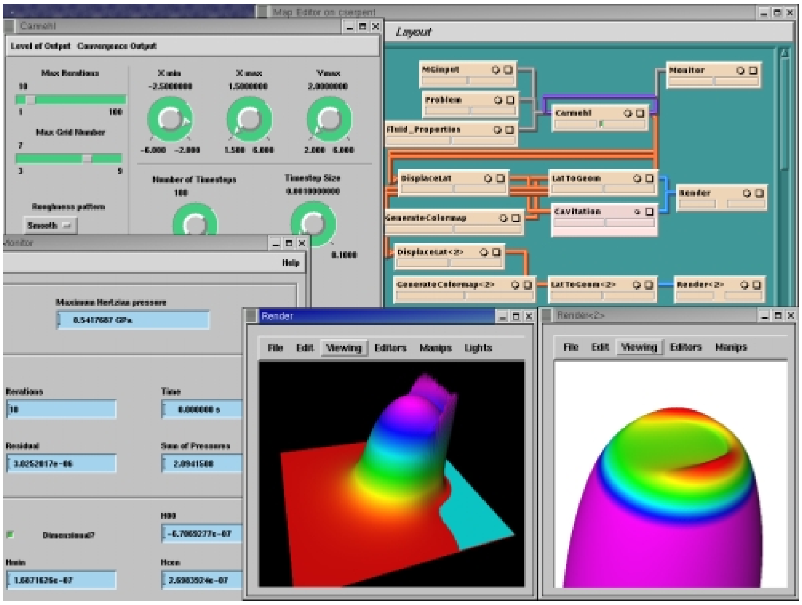
3 Eclipse in IRIS Explorer

IRIS Explorer [7] is a Modular Visualisation Environment that implements the data flow concept of Haber and McNabb [8]. This describes the visualisation process as a sequence of steps that incrementaly change the data from a set of numbers to a visualised image by applying a series of visualisation algorithms. IRIS Explorer implements this concept by providing a library of visualisation techniques represented as a set modules, each module implementing a single visualisation algorithm. These modules are graphically represented as blocks which the user can place on the provided workspace (map editor) and visually connect together by means of wires to describe the flow of the data. This connected set of modules is referred to as a map. Modules in the map execute (fire) only when they have all their required input data. Once they have fired, they pass newly generated data to connected downstream modules which in turn may fire.

IRIS Explorer implements the map as a set of intercommunicating processes attached to a shared memory arena. The flow of data between modules is managed by simply passing a pointer reference to a data object held in the arena. When run on a multiprocessor machine, modules are free to execute in parallel if they have all required input data.

While IRIS Explorer provides many modules for visualisation, it also offers an Application Programming Interface (API) to allow users to incorporate their own codes as modules. A tool, Module Builder, is provided to simplify this process. It allows the user to graphically generate a user interface for their modules by selecting and placing widgets on a control panel. It also generates code to give the user's module access to data in shared memory and to control the execution of the module.

These features of IRIS Explorer make it an ideal framework in which to construct a PSE, as has been shown in previous work such as Wright *et al.* [9].



Level of Output Convergence Output

Max Iterations 10 (1 – 100)

Max Grid Number 7 (3 – 9)

X min -2.5000000 (-6.000 – -2.000)

X max 1.5000000 (1.500 – 6.000)

Ymax 2.0000000 (2.000 – 6.000)

Number of Timesteps 100

Timestep Size 0.0010000000 (0.1000)

Roughness pattern Smooth

Maximum Hertzian pressure 0.5417687 GPa

Iterations 10

Time 0.000000 s

Residual 3.0252017e-06

Sum of Pressures 2.0941500

Dimensional?

H00 -6.7068277e-07

Hmin 1.6871626e-07

Hcen 2.6983574e-07

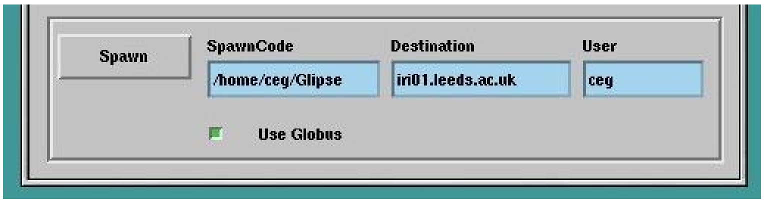
Fig. 2. Eclipse running in IRIS Explorer

For our work in EHL we have implemented the Carmehl code as one module containing the entirety of the numerical solver. One advantage of this closer integration with the visualisation is that as the simulation progresses, at regular intervals output datasets of the pressure and film thickness are produced and immediately sent down the map for visualisation.

The control panel for Carmehl is used to set the dimensions of the computational domain, the mesh refinement level, along with the total number of iterations required on each execution of the module. Other problem specific properties can also be defined on this control panel including information concerning transient problems, and parameters governing surface features. The actual non-dimensional parameters governing the case in question may also be set here, or, through the addition of further modules, as shown in Figure 2, the actual operating conditions for the case defined upstream in the map, will be displayed.

It is through the addition of these extra input modules that steering may be abstracted. Shown in Figure 2 are three different input modules which define the physical conditions of the contact, the parameters defining the lubricant, and parameters used to set the number of iterations in the multilevel schemes.

To implement computational steering, the simulation must be capable of stopping before the computation is complete, reading new control parameters, and then continuing from where it stopped. The cycle is repeated at regular intervals, the frequency of which can be set by the user. For the Carmehl code, this requires the internal work arrays to be statically defined so that the state of the computation is preserved while the module pauses to check its control panel.



SpawnCode: /home/ceg/Glipse
Destination: iri01.leeds.ac.uk
User: ceg

Fig. 3. Grid enabling options for an IRIS Explorer PSE

Once the parameters have been read, the module refires itself and the simulation continues.

4 Grid Enabling

IRIS Explorer modules run, as explained above, by launching a new process on the host machine. With the intention of performing the intensive computations on a Grid resource this conventional approach would require running all of the PSE on the Grid machine. This method is not practical for efficient use of Grid resources. A method of running the computation with IRIS Explorer running on the local machine is therefore required.

The method used here uses Globus tools for Grid job management, including file input and output. We shall assume that the necessary Globus certification process has already been undertaken before launching the PSE.

The PSE in IRIS Explorer will still look identical to that shown in Figure 2. The change comes in the *Carmehl* module, which will now simply be the user interface to the simulation running on the Grid. Outwardly it will appear the same to any user, except there are now extra boxes at the bottom (shown in Figure 3). These options are the only interaction that the user needs to have with the Grid, over Globus: once the destination host has been chosen, the relevant username supplied and the location of the source code to be run specified, then simply clicking on the 'Spawn' button launches the job using `globus-job-run`. It is possible to run jobs through other launching mechanisms similarly. For example by not checking the 'Use Globus' button then `rsh` is used instead.

When the job on the remote machine is started, communication between the launching IRIS Explorer module and the launched Grid process is done through sockets. The launched process knows where to connected to, by means of extra flags passed to it when it is started. Once contact has been established, the launched process then is the dominant communicator, with the launcher as the listener. When the launched process needs data from the PSE, e.g. control parameters for the simulation, it sends a request to the listener who packs the values up into a data array of predefined size and structure, and sends it to the Grid process.

Similarly, output data is packaged by the Grid process and sent to the listener. The received data is then formatted into IRIS Explorer output datatypes which are flushed to downstream modules for visualisation.

The addition of extra input modules before the Grid module poses few problems. The incoming data is packed into arrays which are sent to the Grid module, as with the control parameters. Since these input modules need not always be present, then there must be a default set of parameters for cases where they are not connected so the application can operate accordingly.

Having completed the requested number of iterations, the Grid process does not terminate but regularly polls the associated PSE module until it is requested to perform the next cycle. This eliminates the cost of starting up a job on the Grid, and also means that results from the previous iteration can still be stored statically in memory.

During computationally intensive simulations it used to be the case that the simulation module spent considerable amounts of time 'firing'. Since the work is being done outside the PSE this is no longer the case and so firings only occur when new data is received, be it from changes to input parameters or through the receipt of output data. Whereas input data was only ever available to the simulation at the start of execution it may now be requested at any time, and hence the opportunity for steering the calculation is increased. This may not always be sensible so care must be taken when constructing the communication over which parameters can be allowed to change during the solution process.

Part of the rationale for use of the Grid is to gain access to remote parallel machines. Information about the parallel requirements can be incorporated into the launching mechanism. Two more widgets must be added to the control panel: one detailing the number of processors and one confirming that a parallel job is to be started. It is imperative to specify that the spawned job will be parallel since different launch mechanisms must be used for serial and parallel execution. These mechanisms are summarised in Table 1. To run a job on the Grid using Globus, MPICH-G2 is used to allow communication between globally distributed processors, rather than just those in a fixed machine. MPICH-G2's version of `mpirun` is also required as this does the `globus-job-run` command. If the command "`globus-job-run mpirun ...`" was attempted then most Globus servers would refuse the action as this could be used to get around security restrictions since it launches a Grid process (`mpirun`) which then launches other jobs on the machine that Globus does not manage. The use of a file containing the machines on which the processes will be run (the `-machinefile <machinelist>` flag in Table 1) can be removed if Globus jobs are wanted to run on single systems without MPI

5 Parallel Results

In order to solve equation (2) in a discretised form, the double integral is written as a multisummation over all the points to calculate the deformation at each point. Therefore, for a domain of $N \times N$ points, this calculation would be $\mathcal{O}(N^4)$. Using the multilevel multi-integration technique, as described in [6], this reduces to $\mathcal{O}(N^2 \ln N^2)$. This is still an expensive calculation to do, but can now be

Table 1. Launching mechanisms both on and off the Grid for serial and parallel jobs, executed by IRIS Explorer module

Where run	Parallel?	Launch command
Locally	No	`<path_of_executable> -port %d -machine %s`
Locally	Yes	`NATIVE-mpirun -np %d <path_of_executable> -port %d -machine %s`
Network	No	`rsh <target> <path_of_executable> -port %d -machine %s`
Network	Yes	`rsh <target> NATIVE-mpirun -np %d <path_of_executable> -port %d -machine %s`
Grid	No	`globus-job-run <path_of_executable> -port %d -machine %s`
Grid	Yes	`MPICH-G2-mpirun -machinefile <machinelist> -np %d <path_of_executable> -port %d -machine %s`

tackled in realistic amounts of time for fine grids. Being able to accelerate this process still further, through parallelisation is the aim.

In order to maintain the strong convergence properties of the numerical solver, the domain has been partitioned into strips in the X direction. Communication within solves is restricted to the Jacobi and Gauss-Seidel iterations required to solve Equation (1) for the pressure, P, and to the deformation part of the film thickness solve. For this multilevel scheme there needs to be many communications between processors because the algorithm's global nature, which requires sixth order coarsening operators to be used.

The parallelisation of the code has been done using MPI. Currently a fully distributed memory model is not used. This is because of the complex nature of the calculations undertaken and the complete restructuring that will be required. For this work the memory is still such that all processors have arrays large enough to hold all the data. Communication is only done, however, for required data.

For the deformation calculation, the domain has been partitioned such that the points requiring calculation on each processor are aligned to reduce communication whilst maximising computation. This is only an issue in the deformation calculation where the domain sizes 'changes' as the coarsening operators introduce ghost cells for some of the calculations. The memory model used for the parallel EHL calculation is the same as that used for the serial version. That is with each processor allocating storage space for all the points of all the arrays, and with global synchronisation of data only when necessary.

The test cases shown below are for a grid resolution of 1025×1025 points. Solution times are given for 10 multigrid V-cycles after a full multigrid start, see e.g. [10] for details. The case chosen is a standard test case with non-dimensional Moes parameters of M=35, L=10. The test case was run on three very different hardware architectures. Each process takes 1Gb of memory and five coarser levels are used in the multi-integration algorithm for cases (a) to (c) in Table 2. The results clearly show how much better the code scales on the shared memory machines of the Onyx and the SUN. This is due to the large amounts of communication involved in the deformation calculation. For the Intel Pentium 4 machine, the processors are physically paired with shared memory only inside the node. This means that beyond two processors communication is a lot slower.

Table 2. Parallel timings for the EHL code

Np	Time (s)	Speed-up
1	385.21	1.00
2	256.55	1.50
4	128.85	2.99
8	-	-

(a) SGI Onyx R12000

Np	Time (s)	Speed-up
1	481.58	1.00
2	254.64	1.88
4	140.65	3.42
8	91.81	5.24

(b) SUN V880 750MHz

Np	Time (s)	Speed-up
1	394.92	1.00
2	216.94	1.82
4	148.88	2.65
8	119.93	3.29

(c) Intel Pentium 4 2.0GHz

Np	Time (s)	Speed-up
1	581.18	1.00
2	370.40	1.57
4	214.07	2.71
8	144.74	4.01

(d) Intel Pentium 4 2.0GHz, only 3 levels of coarser grids

If less coarse grid levels are used in the multi-integration then there is more computation per communication and hence the code scales better, as shown in Table 2d.

6 Conclusion and Future Work

In this paper we have shown that it is possible to construct a PSE which is capable of both launching and interacting with computational jobs on the Grid. Using Globus tools computational jobs have been started on Grid machines away from the visualisation machine running the PSE. By having a 'listener' module in the PSE connected to the grid machine it is possible to handle requests from the Grid job both for input data from the module's control panel and from other input modules, and also to process output data from the simulation for visualisation.

The jobs started on the Grid include the possibility for launching parallel MPI processes, something which is not possible using the standard job control methods in IRIS Explorer. It was seen that different launch methods were required for the combination of both serial and parallel codes, and for the different types of MPI used.

The parallel EHL code has shown remarkably good scaling, especially on shared memory machines for which the huge communication overhead of the deformation calculation is less significant.

The intention for future work is to build on the parallelism started here. The industrial challenge facing EHL is now limited by the problem size that can be tackled. The resolutions of grids required in the next decade will grow significantly as realistic surface roughness patterns are attempted. The global nature of the film thickness calculation means that completely distributed memory storage has not previously been attempted, as explained in Section 5, but it will become a necessity.

One issue that did become clear during this work was the lack of an API for communication of data to and from the Grid module. Every data block received must be strictly programmed into the code. With a more flexible and generic communication style predefined datatypes could be passed between processes more easily.

Further development of the PSE is also proposed with the inclusion of collaborative techniques. This will use the COVISA toolkit [11] and build on previous collaborative work in IRIS Explorer, such as [12]. The advantage of such an approach, especially when combined with the Grid, are that two scientists in, perhaps, globally separated locations can both visualise and influence a simulation taking place elsewhere. This kind of technique is especially useful for multi-disciplinary collaborations between scientists at different sites.

Acknowledgments. This work has been supported by an EPSRC ROPA award for CEG and part funded from an EPSRC e-Science Demonstrator for JW.

References

1. Parker, S.G., Johnson, C.R.: SCIRun: A scientific programming environment for computational steering. In Meuer, H.W., ed.: Proceedings of Supercomputer '95, New York, Springer-Verlag (1995)
2. Goodyer, C.E., Berzins, M.: Eclipse and Ellipse: PSEs for EHL solutions using IRIS Explorer and SCIRun. In: Proceedings of ICCS '02, Amsterdam, The Netherlands, Lecture Notes in Computer Science. Volume 2329-31., Springer (2002)
3. Goodyer, C.E.: Adaptive Numerical Methods for Elastohydrodynamic Lubrication. PhD thesis, University of Leeds, Leeds, England (2001)
4. Arenaz, M., Doallo, R., Touriño, J., Vázquez, C.: Efficient parallel numerical solver for the elastohydrodynamic Reynolds-Hertz problem. Parallel Computing **27** (2000) 1743–1765
5. Dowson, D., Ehret, P.: Past, present and future studies in elastohydrodynamics. Proceedings of the Institution of Mechanical Engineers Part J, Journal of Engineering Tribology **213** (1999) 317–333
6. Venner, C.H., Lubrecht, A.A.: Multilevel Methods in Lubrication. Elsevier (2000)
7. Walton, J.P.R.B.: Now you see it – interactive visualisation of large datasets. In Brebbia, C.A., Power, H., eds.: Applications of Supercomputers in Engineering III. Computatational Mechanics Publications / Elsevier Applied Science (1993)
8. Haber, R.B., McNabb, D.A.: Visualization idioms : A conceptual model for scientific visualization systems. In G.M. Nielson, B.S., Rosenblum, L., eds.: Visualization in Scientific Computing. IEEE (1990) 74–93
9. Wright, H., Brodlie, K.W., David, T.: Navigating high-dimensional spaces to support design steering. In: VIS 2000, IEEE (2000) 291–296
10. Trottenberg, U., Oosterlee, C., Schüller, A.: Multigrid. Academic Press (2001)
11. Wood, J., Wright, H., Brodlie, K.W.: Collaborative visualization. In: Proceedings of IEEE Visualization 97. (1997) 253–259
12. Walkley, M.A., Wood, J., Brodlie, K.W.: A collaborative problem solving environment in IRIS Explorer. In: Proceedings of ICCS '02, Amsterdam, The Netherlands, Lecture Notes in Computer Science. Volume 2329-31., Springer (2002)