University of Leeds
SCHOOL OF COMPUTING
RESEARCH REPORT SERIES
Report 2005.05

Parallelisation and scalability issues in a multilevel EHL solver

by

C E Goodyer & M Berzins

December 2005

ABSTRACT

The computation of numerical solutions to elastohydrodynamic lubrication problems is only possible on fine meshes by using a combination of multigrid and multilevel techniques. In this paper we show how the parallelisation of both multigrid and multilevel multi-integration for these problems may be accomplished and discuss the scalability of the resulting code. A performance model of the solver is constructed and used to perform an analysis of the results obtained. Results are shown with good speed-ups and excellent scalability for distributed memory architectures and in agreement with the model.

Keywords: Elastohydrodynamic lubrication distributed memory parallelism scalability

INTRODUCTION

Parallelisation of scientific engineering codes has proved to be particularly useful whenever either results are needed quickly or the memory requirements are too large to be handled in serial. In the case of solvers for the important engineering problem of elastohydrodynamic lubrication (EHL) both these situtations can arise. The EHL regime occurs in journal bearings and gears, where, under severe loads in the presence of a lubricant, there may be 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, for example it may take on glass-like properties [1].

The computational challenge in solving such problems is considerable. Although the time dependent partial differential and integral equations apply only in one or two space dimensions, they have a dense sparsity pattern and are highly nonlinear. One of the problems of current interest is to calculate the frictional characteristics of measured surface roughness profiles. This has been successfully undertaken for one dimensional line contact cases, e.g. [2, 3]. Tackling the more realistic 2d case has been recognised as one of the immediate challenges in tribology [4]. In order to do this spatial meshes of $10^6 \times 10^6$ points may be needed. This means that 10^{12} dense nonlinear equations may need to be solved. This challenge is beyond a single workstation at present and requires the use of parallel computers. Given that at present calculations using more than 1000×1000 mesh points are rare to the best of the authors' knowledge, the need for a better understanding of how parallelism may be applied is obvious.

This paper will address the parallel solution of EHL problems by first describing the numerical problem to be solved with both the governing equations and a brief introduction to the solution methods used being covered. The multilevel techniques used will be highlighted, along with the reasons why they make effective parallelisation such a communication intensive process. The parallel approaches we have taken are then explained and analysed by means of a performance model. The results section demonstrates how effective these approaches have been in obtaining remarkably good speed-ups and scalabilities, given the amount of global communication present. The good agreement between the calculated efficiencies and those predicted by the performance model provides a way of predicting the scalability of larger problems on architectures with more processors. The paper is concluded with some suggestions for further work in this field.

SERIAL PROCESSOR SOLUTION METHODS

Full details of both the EHL problem and the serial solution methods used are described in the book by Venner and Lubrecht [5] and with details specific to the discussion here given by Goodyer in [6].

Governing equations

The EHL case is governed by two main sets of equations, namely those concerning physical behaviour of the contact, and those governing the changes in the lubricant. The solution variables which must be calculated are the pressure profile P, across the domain, the surface geometry H, the viscosity $\overline{\eta}$ and the density $\overline{\rho}$. The pressure distribution is described by the Reynolds Equation see [5], given in non-dimensional form by:

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

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 contiguous film of lubricant. The non-dimensional film thickness, H, is given by:

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

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 $\mathcal{R}$ is the roughness profile. The double integral defines the deformation of the surface due to the pressure distribution across the entire domain.

The conservation law for the applied force (the Force Balance Equation) is given by:

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}P(X,Y)\,\mathrm{d}X\mathrm{d}Y=\frac{2\pi}{3}\,. \tag{3}$$

Since an isothermal, generalised Newtonian lubricant model is being used in this work, only expressions for the density and viscosity will be required. The density

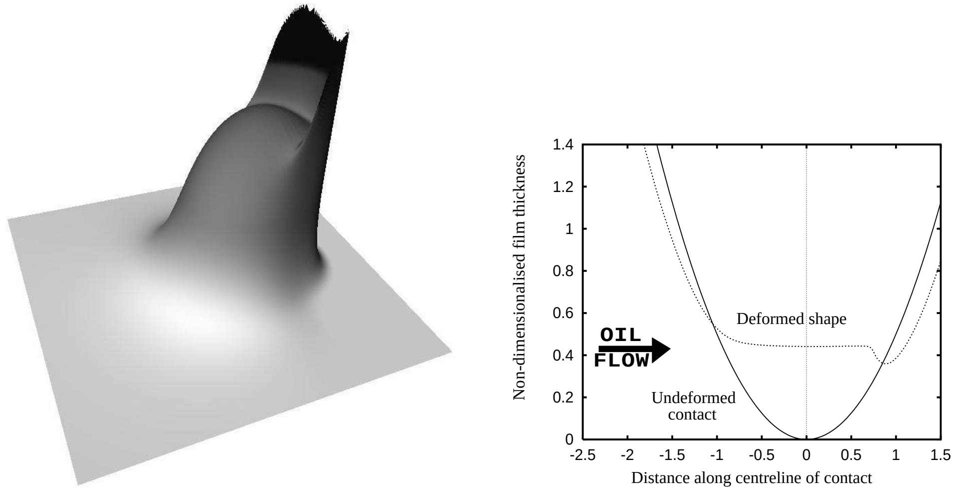

Figure 1: (a) Typical pressure values across an EHL point contact (b) Typical film thickness

model chosen is that of Dowson and Higginson, see [5], which takes into account the compressibility of the lubricant:

$$\overline{\rho}(P) = \frac{0.59 \times 10^9 + 1.34 p_h P}{0.59 \times 10^9 + p_h P}, \tag{4}$$

where p_h is the maximum Hertzian pressure.

The viscosity model used is the Roelands pressure-viscosity relation, see [5]:

$$\overline{\eta}(P) = \mathrm{e}^{\left\{\frac{\alpha p_0}{z_i}\left[-1+\left(1+\frac{p_h P}{p_0}\right)^{z_i}\right]\right\}}, \tag{5}$$

where η_0 is the viscosity at ambient pressure, p_0 is a constant (typically 1.98×10^8), z_i is the pressure viscosity index, taken as $z_i = 0.68$ and α is the pressure viscosity coefficient.

Numerical methods

The nature of the EHL problem means that there are three very different areas of the domain when calculating pressure. Firstly, the *cavitation region* is the area of the solution beyond the free boundary where the Reynolds Equation is not valid. Secondly, in the centre of the domain is the *contact area*, where the pressure rises sharply to reach its maximum peak in a near Hertzian shape. EHL pressure profiles do differ from purely hydrodynamic ones in that there is also the presence of a large ridge on the pressure peak, towards the outflow boundary, as can be seen in Figure 1 where the 3D non-dimensional pressure profile is shown as well as the film thickness along the centre line. which shows the shape of the contact along the centreline. The deformation away from the original surface geometry is clearly visible, and it can be seen that there is a constriction in the contact towards the outflow, which coincides with a position between the pressure spike and the cavitation boundary. Finally, in the *non-contact region* the pressure is very small compared to the contact region.

The methods used for solving for the pressure, the film thickness and the lubricant properties on a mesh of N_X by N_Y points with a mesh spacing of ΔX and ΔY in X and Y respectively, are now described in turn.

Pressure

The Reynolds Equation discretisation used in our software is a first order scheme, [7]:

$$\begin{aligned} &\frac{\epsilon_{i-\frac{1}{2},\,j}\left(P^n_{i-1,\,j}-P^n_{i,\,j}\right)+\epsilon_{i+\frac{1}{2},\,j}\left(P^n_{i+1,\,j}-P^n_{i,\,j}\right)}{(\Delta X)^2} \\ &+\frac{\epsilon_{i,\,j-\frac{1}{2}}\left(P^n_{i,\,j-1}-P^n_{i,\,j}\right)+\epsilon_{i,\,j+\frac{1}{2}}\left(P^n_{i,\,j+1}-P^n_{i,\,j}\right)}{(\Delta Y)^2} \\ -\frac{u_s(T)}{u_{ref}}&\frac{\rho^n_{i,\,j}H^n_{i,\,j}-\rho^n_{i-1,\,j}H^n_{i-1,\,j}}{\Delta X}-\frac{\rho^n_{i,\,j}H^n_{i,\,j}-\rho^{n-1}_{i,\,j}H^{n-1}_{i,\,j}}{\Delta T} = 0 \end{aligned} \tag{6}$$

where n is the current timestep, and

$$\epsilon_{i\pm\frac{1}{2},\,j}=\frac{\epsilon^n_{i\pm1,\,j}+\epsilon^n_{i,\,j}}{2},\,\epsilon_{i,\,j\pm\frac{1}{2}}=\frac{\epsilon^n_{i,\,j\pm1}+\epsilon^n_{i,\,j}}{2}, \tag{7}$$

where $\epsilon_{i,\,j}=\frac{a^3p_h}{6\eta_0R_x^2u_s(0)}\frac{\overline{\rho}_{i,\,j}H^3_{i,\,j}}{\overline{\eta}_{i,\,j}}$

The discretisations above are all very much aligned along the flow direction, i.e. parallel to the X-axis. The contributions from terms perpendicular to this axis are small. All the fast EHL solution techniques take advantage of this polarisation and tend to solve along mesh lines in the flow direction.

The three distinct regions described above require different numerical schemes to be employed when solving the Reynolds Equation. In the non-contact region a Gauss-Seidel line relaxation scheme is used; in the contact region a Jacobi line scheme is employed and in the cavitation region the Christopherson approach is used [8], where all calculated negative pressures are set to be zero.

The scope of the relaxation scheme used involves employing both the Gauss-Seidel and the Jacobi line relaxation schemes on the same grid, but without any overlap, depending on the position of the grid point (i, j) on the computational domain. The two relaxation schemes are employed as follows:

Given an approximation $\tilde{P}_{i,j}$ and the associated approximation $\tilde{H}_{i,j}$ to the pressure $P_{i,j}$ and the film thickness $H_{i,j}$ respectively, a new approximation $\overline{P}_{i,j}$ is computed using

$$\overline{P}_{i,j}=\tilde{P}_{i,j}+w\Delta P_{i,j} \tag{8}$$

where w is a damping factor, which is critical to ensure convergence of the method.

On the line $Y = j$, the correction terms $\Delta P_{i,j}$ $(i = 1, \ldots, N_X)$ are solved simultaneously using a system of equations created at each grid point (i, j). Depending on the solution at the grid point (i, j), either the Gauss-Seidel or the Jacobi schemes are employed. If the grid point (i, j) lies in the non-contact region of the computational domain, then the Gauss-Seidel scheme is employed and the equation at this grid is given by

$$\frac{\partial \tilde{L}_{i,j}}{\partial \tilde{P}_{i-2,j}}\Delta P_{i-2,j} + \frac{\partial \tilde{L}_{i,j}}{\partial \tilde{P}_{i-1,j}}\Delta P_{i-1,j} + \frac{\partial \tilde{L}_{i,j}}{\partial \tilde{P}_{i,j}}\Delta P_{i,j} + \frac{\partial \tilde{L}_{i,j}}{\partial \tilde{P}_{i+1,j}}\Delta P_{i+1,j}\frac{\partial \tilde{L}_{i,j}}{\partial \tilde{P}_{i+2,j}}\Delta P_{i+2,j} = r_{i,j} \quad (9)$$

where, $\tilde{L}_{i,j} = L(\tilde{P}_{i,j}) = r_{i,j}$.

This system is solved using a pentadiagonal approximation to the Jacobian matrix along the line. Since the matrix entries in the full Jacobian are small away from (i, j) then making this penta-diagonal approximation [6], does not hamper convergence and allows much faster solution times.

The residual at the point (i, j), $r_{i,j}$, is given by

$$\begin{aligned} r_{i,j} = {} & \epsilon_{i-\frac{1}{2},j}(\tilde{P}_{i-1,j} - \tilde{P}_{i,j}) + \epsilon_{i+\frac{1}{2},j}(\tilde{P}_{i+1,j} - \tilde{P}_{i,j}) + h_x^2\, h_y^{-2}(\epsilon_{i,j-\frac{1}{2}}(\overline{P}_{i,j-1} - \tilde{P}_{i,j}) \\ & + \epsilon_{i,j+\frac{1}{2}}(\tilde{P}_{i,j+1} - \tilde{P}_{i,j})) - h_x(\overline{\rho}_{i,j}\tilde{H}_{i,j} - \overline{\rho}_{i-1,j}\tilde{H}_{i-1,j})\,. \end{aligned} \quad (10)$$

However, if the grid point (i, j) lies in the contact region of the computational domain, then the Jacobi scheme is employed and the equation at this grid point is as given by equation (9) except for the residual in which $\overline{P}_{i,j-1}$ is replaced by $\tilde{P}_{i,j-1}$. As the Jacobi and Gauss-Seidel schemes used do not converge quickly on fine grids, multigrid is often used to accelerate convergence and is summarised in the next section.

Film thickness calculation

The film thickness calculation, once discretised, has the form:

$$H_{i,\,j} = H_{00} + \frac{X_i^2}{2} + \frac{Y_j^2}{2} + \mathcal{R}_{i,j} + \delta_{i,j}^{k^{eval}} \quad (11)$$

where

$$\delta_{i,j}^{k^{eval}} = \Delta X \Delta Y \sum_{k=1}^{N_X} \sum_{l=1}^{N_Y} K_{i,\,j,\,k,\,l} P_{k,\,l}^{k^{eval}}\,, \quad (12)$$

where K is the film thickness kernel matrix, approximating the double integral of equation (2), where the superscript k^{eval} corresponds to the grid of N_X by N_Y points and where the factor $\Delta X \Delta Y$ is a scaling factor to give mesh independence to the $K_{....}$ coefficients. Hence for every mesh point, (i, j), the deformation term is a multi-summation of the pressures at all the other points in the computational domain. As this calculation is $O(N^4)$ where $N = N_Y = N_X$, the cost is reduced to $O(N^2 \ln N^2)$ by using the multilevel multi-integration technique of Brandt and Lubrecht [9] as described in the next section.

The calculation of H_{00} in equation (11) is accomplished by relaxation of the Force Balance Equation (3), according to:

$$H_{00} \leftarrow H_{00} - c\left(\frac{2\pi}{3} - \Delta X \Delta Y \sum_{i=1}^{N_X} \sum_{j=1}^{N_Y} P_{i,j}\right), \quad (13)$$

for all mesh points (i, j), where c is a small relaxation parameter. The mathematical basis justifying this update is described in [10]

In the context of EHL calculations one smoothing cycle is said to be the sequence of updating all the pressures, $P_{i,j}$, and film thickness values $H_{i,j}$, along with the corresponding density and viscosity values.

Multigrid and Multilevel Multi-integration techniques

The multilevel methods of Lubrecht, Brandt and Venner [5, 9, 11] have proved very successful in computing solutions to EHL problems quickly. There are two main multilevel components; the Full Approximation Scheme (FAS) multigrid is used to solve the nonlinear equations, whilst for the fast solution of equation (11) multilevel multi-integration (MLMI) is used.

Multigrid

The point contact EHL solver described here uses a hierarchy of regular multigrid meshes of size $N_x^k \times N_x^k$ elements, where $N_x^k = 2^k + 1$. The level of refinement of the mesh can then be referred to as being grid level k. Due to symmetry about the line $Y = 0$ it is only be necessary to solve on half of the computational domain. Since different grid resolutions have different smoothing properties this means we can eliminate errors quicker than by just working on the finest throughout. The multigrid Full Approximation Scheme (FAS), [12], aims to solve the nonlinear system

$$\mathcal{L}^k(\underline{P}^k) = f^k, \tag{14}$$

where $\mathcal{L}^k$ is a discrete approximation such as that of equation (6) to the differential operator $\mathcal{L}$ defined by equation (1). The solution to equation (14) obtained by an iterative method is denoted by $\underline{\tilde{u}}^k$ approximates the exact solution $\underline{u}$ with a residual defined by

$$r^k = f^k - \mathcal{L}^k \underline{\tilde{P}}^k. \tag{15}$$

After relaxing the system of equation on grid k to get an approximation $\underline{\tilde{u}}^k$, a representation of this on a coarser grid, j, can be formed using a suitable coarsening operator, I_k^j. On this coarser grid a system of equations in the same form as (14) can be formed as

$$\mathcal{L}^j \underline{\hat{P}}^j = \hat{f}^j, \tag{16}$$

where $\underline{\hat{P}}^j = I_k^j \underline{\tilde{P}}^k + \underline{e}^j$ and $\hat{f}^j = \mathcal{L}^j(I_k^j \underline{\tilde{P}}^k) + I_k^j r^k$. By solving equation (16) we can obtain a coarse grid correction to the solution on grid k which, using a suitable operator I_j^k

$$\underline{\tilde{P}}^k \leftarrow \underline{\tilde{P}}^k + I_j^k(\underline{\overline{P}}^k - I_k^j \underline{\tilde{P}}^k). \tag{17}$$

where $\underline{\overline{P}}^j$ is the calculated approximation to $\underline{\hat{P}}^j$ as in equation (16).

For cases with regular meshes of 2^k+1 points in each direction then all the mesh points on grid level k-1 are coincident with points on level k. This means simple intergrid operators I_k^j and I_j^k can be defined, either by injection or weighted interpolation of neighbouring points, [12].

By repeated application of the coarse grid correction process described by equation (17) the solution scheme can be built up to be solved on the hierarchy of grids. Assuming that the same iterative process can be used to solve the coarse grid system as the fine grid system, then the finest grid will be used to smooth the highest frequency errors, and progressively coarser grids used to smooth errors of progressively lower frequencies (coarsening), before returning to get an updated solution on the finest mesh (prolongation). The smoothing cycles done before coarsening are called *pre-smooths* and those done after prolongation and correction of the solution are referred to as *post-smooths*.

The simplest multigrid cycle is the V-cycle. An initial approximation on the finest grid has ν_1 pre-smooths before being coarsened. This is then repeated until the coarsest mesh is reached where ν_0 smoothing cycles are done. The solution on the next finer mesh is then corrected according to equation (17) before having ν_2 post-smooths. Again this process is repeated until a corrected, smoothed solution is reached on the finest mesh. This V-cycle is known as a V(ν_1,ν_2)-cycle. Typical values for ν_1 and ν_2 are three or less, although ν_0 may be much larger in order to obtain a much better coarse grid solution.

In EHL calculations the number of Newton iterations per smoothing step in the code described here is typically $\sigma_{newt} = 2$. In a multigrid V(σ_{pre},σ_{post}) cycle the Reynolds Equation is solved σ_{Re} times per level, where

$$\sigma_{Re} = \sigma_{pre} + \sigma_{post} + 1 \tag{18}$$

Denoting this number of solves per level to be

$$\sigma_{tot} = (\sigma_{Re}) \times \sigma_{newt} \tag{19}$$

and noting that on the coarsest grid typically many more smooths will be done, say $\sigma_{coarse} = 30$.

The process of *Full Multigrid* (FMG) is designed to eliminate the large errors which initially exist on the fine grid, by starting on the coarsest grid. FMG uses the same multigrid techniques and V-cycles as described above on each of the coarse grids. At the end of each set of V-cycles the computed solution is then prolonged up to the next finest grid level and the process is repeated until the finest grid is reached.

In-depth descriptions of how multigrid is applied to EHL problems can be found in [6] and [5]. For example the multigrid method has to be modified to deal with the free boundary at the edge of the cavitation region. If information is allowed to propagate from the cavitation region into the pressure positive region in the coarsening or prolonging stages, or if the solution on a coarser grid moves the cavitation boundary one coarse mesh point into the cavitation region, then stalling may occur [13]. This problem is eliminated by not applying multigrid near the boundary at the risk of slower convergence of solution boundary values.

The other major difference in the multigrid EHL solver is concerned with the iteration for H_{00}, and hence updating the Force Balance Equation (3). This value of H_{00} is only ever corrected once per multigrid cycle, using equation (13) and this is done on the coarsest grid. Appropriate corrections from finer levels are necessary to ensure that it is the applied force, as defined by equation (3) on the finest grid which is being conserved, rather than that on coarser grids, e.g. [5]. The inclusion of the force balance equation is done through a relaxation of the H_{00} parameter, as given in equation (13). This is clearly a global operation. This relaxation only ever takes place on the coarsest grid and so the actual update is given by

$$H_{00} \leftarrow H_{00} - c\left(\frac{2\pi}{3} - (\Delta X)^k (\Delta Y)^k \sum_{i=1}^{N_X^k} \sum_{j=1}^{N_Y^k} P_{i,j}^k + \tau^k\right), \tag{20}$$

for where grid corrections τ are defined by

$$\tau^{k-1} = \tau^k + (\Delta X)^k (\Delta Y)^k \sum_{i=1}^{N_X^k} \sum_{j=1}^{N_Y^k} P_{i,j}^k - (\Delta X)^{k-1} (\Delta Y)^{k-1} \sum_{i=1}^{N_X^{k-1}} \sum_{j=1}^{N_Y^{k-1}} \overline{P}_{i,j}^{k-1}, \tag{21}$$

with $P_{i,j}^k$ and $\overline{P}_{i,j}^{k-1}$ defined as the fine and coarse grid approximations to the pressure solution on grids k and $k-1$ respectively, similar to equation (17).

Multilevel Multi-integration

The most computationally expensive part of any EHL calculation is the potentially N^2 evaluation of the double summation in equation (11) for each of N^2 mesh points. Brandt and Lubrecht [9] developed multilevel multi-integration (MLMI) in order to reduce such a calculation from $O(N^4)$ to $O(N^2 \ln N^2)$.

MLMI assumes that the kernel matrix, as defined in equation (12), K represents the discretisation of a smooth kernel function, at least greater than a small distance away from the point (i, j). This means that provided suitably accurate restriction operators are used then the multi-summation can be performed on a coarser grid than k^{eval} as defined by equation (11), say k^{sum}. A hierarchy of grids is therefore again used to calculate the deformation on a grid k^{eval}. If the $k^{eval} \leqslant k^{sum}$ then the multi-summation given by equation (11) will be performed. However $k^{eval} > k^{sum}$ then MLMI will be used.

The relationship between a multigrid V-cycle and an MLMI cycle is shown diagrammatically Figure 2. In contrast to the multigrid method the most striking change is that there is no calculation other than the multi-summation on the coarsest grid, and the correction stages, meaning almost all the work is in grid transfer operations.

The method can be thus be reduced to four main operations:

(i) Coarsening the pressure solution and the kernel matrix to grid k^{sum}. These transfer operators are denoted here by J_j^k for transfer from grid j to grid k, with $J_j^k = \left(J_k^j\right)^T$. The grid transfer operations for both the coarsening and refinement stages are done with 6^{th} order interpolation operators are used which cover most cases feasible for point contact EHL cases, with a slightly larger correction patch for very fine grids. The stencil used in this work is given by [5] as

$$P_I = \frac{-25P_{i-2} + 150P_{i-1} + 256P_i + 150P_{i+1} - 25P_{i+2} + 3P_{i+3}}{512}, \tag{22}$$

where P_I on grid $k-1$ is coincident with P_i on the grid k.

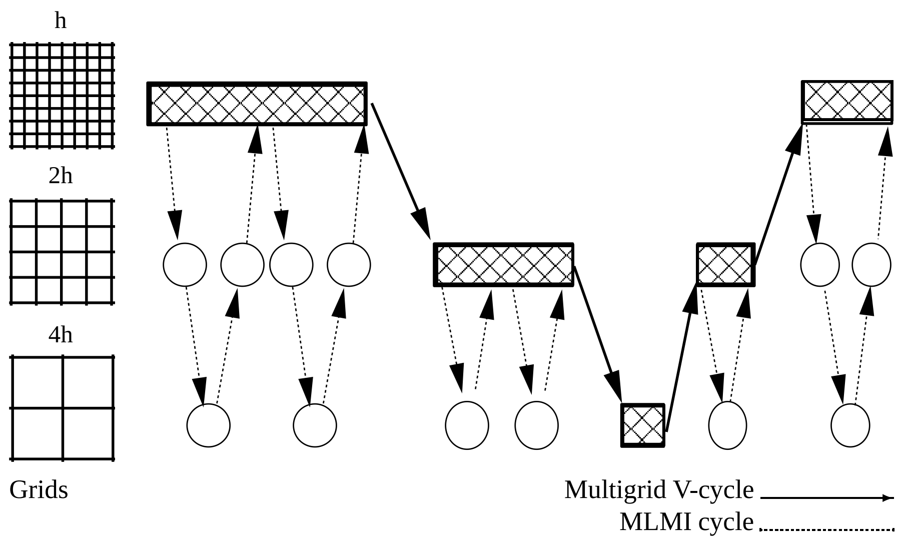

Figure 2: Example of a V-cycle with MLMI at each stage

(ii) Performing the multi-summation on grid k^{sum} to calculate an approximate deformation At all the points (I, J) of grid k^{sum} there are coincident points (i, j) on grid k^{eval} and hence the coarse grid multi-summation is given by

$$\delta_{I,J}^{k^{sum}} = (\Delta X \Delta Y) \sum_{k=1}^{n_x^{k^{sum}}} \sum_{l=1}^{n_y^{k^{sum}}} K_{I,J,k,l}^{k^{sum}} J_{k^{eval}}^{k^{sum}} P_{i,j}^{k^{eval}} \tag{23}$$

$$= (\Delta X \Delta Y)^{(k^{eval} - k^{sum})} \sum_{k=1}^{n_x^{k^{sum}}} \sum_{l=1}^{n_y^{k^{sum}}} K_{i,j,k,l}^{k^{eval}} J_{k^{eval}}^{k^{sum}} P_{i,j}^{k^{eval}} \tag{24}$$

(iii) Interpolation of the calculated deformation back to the finer grids . This is simply the reverse of the process in (i), [5].

and

(iv) Correction of the deformation around the kernel's singularity . The singular kernel at the point (i, j) requires a local correction to the deformation calculated around this point on the coarser grid k^{sum}. This correction needs to be done over as small a correction patch as possible and hence the multi-summation must be performed on each grid in the refining process back to grid k^{eval}. The number of points in this region for correction in each dimension is given by [5] as

$$m = 3 + \ln(n), \tag{25}$$

which thus defines Ω_{sing}. The correction comes in two parts, namely subtraction of the contributions already included in the deformation from the coincident points on the coarse grid, and then the inclusion of the contributions from the finer grid. Mathematically these are given by:

$$\delta_{I,J}^{k^{sum}} \leftarrow \delta_{I,J}^{k^{sum}} + (\Delta X \Delta Y)^{(k^{eval}-k^{sum})} \sum_{(i,j)\in\Omega_{sing}} \left(K_{2I,2J,k,l}^{k^{eval}} - \tilde{K}_{2I,2J,k,l}^{k^{eval}}\right) P_{k,l}^{k^{eval}}, \tag{26}$$

for the correction at coincident points, and

$$\delta_{i,j}^{k^{eval}} = \left[J_{k^{sum}}^{k^{eval}} \delta_{\cdot}^{k^{sum}}\right]_i + (\Delta X \Delta Y)^{(k^{eval}-k^{sum})} \sum_{(i,j)\in\Omega_{sing}} \left(K_{i,j,k,l}^{k^{eval}} - \tilde{K}_{i,j,k,l}^{k^{eval}}\right) P_{k,l}^{k^{eval}}, \tag{27}$$

for the non-coincident points, where $\tilde{K}_{i,j,k,l}^{k^{eval}}$ is the kernel function on the coarse mesh k^{sum} interpolated back onto the fine mesh k^{eval} see Section 5.7.3 of [5].

In the code implementation both coarsening and refining methods are done via "half-grids" where only one dimension is coarsened at a time. This means that the algorithm above can be iteratively applied in alternating directions for two dimensional cases. A full description of this method is given in [5].

SERIAL COMPUTATIONAL COMPLEXITY

In this section the computational costs of the multigrid algorithm and of the multilevel multi-integration are estimated.

Let us define the total number of grids used in the solution scheme to be

$$k^{tot} = k^{fine} - k^{coarse} + 1, \quad k^{dif} = k^{fine} - k^{coarse} \tag{28}$$

and that the number of grid points in the x and y directions is defined by

$$N_X^k = N_Y^k = 2^k + 1. \tag{29}$$

Also for notational convenience let

$$k^c = k^{coarse} \tag{30}$$

and

$$k^f = k^{fine}\ . \tag{31}$$

V-cycle Computation Costs

We are using a V(3,1) cycle with 4 smooths on each non coarse grid and 30 smooths on the coarsest grid. For each processor the following costs are incurred on grid level k:

- Banded line solve for Reynolds equation: $N_Y^k \times \mathcal{O}\left(5^2 N_X^k\right)$;
- Viscosity and density calculations, each $\mathcal{O}(N_X^k N_Y^k)$ but involve expensive power and exponential calculations;

- Grid transfer operations, $O(N_X^k N_Y^k)$;
- Calculation of the deformation using MLMI.

There are also additional calculations of the viscosity, density and deformation during the grid transfer operations which are almost the total cost of an extra smooth.

The V-cycle computational cost is thus given by:

$$VC_{cost} \approx \sum_{k=k_c}^{k_f} \gamma_k^{MG} \left[\kappa_{vc} N_X^k N_Y^k + VC_{cost}^{mlmi} \right] \tag{32}$$

where $\gamma_k^{MG} = 6$ except for the coarsest mesh where $\gamma_{k_{coarse}}^{MG} = 30$, and κ_{vc} is a constant denoting the number of operations done to compute a single point.

The cost of the MLMI calculation, VC_{cost}^{mlmi}, can be broken into three parts. First the multi-summation has a computational cost of $O\left(\left[M_X^{k_c} M_Y^{k_c}\right]^2\right)$. The corrections to each point during the refinement sequence on each grid are almost independent of grid level, given from equation (25), $O\left[(3 + \ln M_X)(3 + \ln M_Y) M_X^k M_Y^k\right]$ Grid transfer operators are of similar cost to the transfer operators in the V-cycle however with higher multipliers since there are now the extra ghost points, half grids are used as well and also the transfer operators are of a higher order: $O\left(M_X^k M_Y^k\right)$. The MLMI cycle computational cost may therefore be approximated by:

$$\begin{aligned} VC_{cost}^{mlmi} \quad \approx \quad & \kappa_{sum} \left[\left(M_X^{k_c} M_Y^{k_c} \right)^2 \right] \\ & + \sum_{k=k_{c+1}}^{K} M_X^k M_Y^k \left[\kappa_{trans} + \kappa_{corr} \left(3 + \log\left(M_X^k \right) \right) \left(3 + \log\left(M_Y^k \right) \right) \right], \end{aligned} \tag{33}$$

where κ_{trans}, κ_{corr} and κ_{sum} are measures of the number of operations needed for each pointwise calculation. In the model results presented later we evaluate this sum explicitly but for reasons of brevity of the algebra such an expansion is not presented here.

Our serial experiments have shown that it is possible to make estimates for each of the $\kappa_{.}$ values. The value of κ_{vc} is estimated at 1456. κ_{sum} is countable from the code, and the value agrees with our experimental value of 7. The values of κ_{trans} and κ_{corr} are similarly taken to be 48 and 5 respectively. These values will be used in the later comparisons between the parallel model efficiency and the observed parallel efficiency of the code.

PARALLELISATION OF MULTILEVEL EHL SOLVER

The starting point for the parallelisation of the method described above is the large amount of work done on parallel multigrid methods and work by the authors on shared memory machines, [14]. Discussions as to why the parallelisation of the already computationally optimal multigrid algorithm does not produce high efficiencies are given by McBryan *et al.* [15], Llorente *et al.* [16, 17] and Tuminaro and Womble [18]. The main problems are the frequency with which coarse grids are encountered meaning that there are very high communications costs relative to the computation. This is especially true once the *critical level* has been reached, namely the coarse grid where each processor has the smallest non-trivial amount of computation. The choice left is whether to use the critical level as the coarsest in the multilevel scheme; to agglomerate, by moving all the work to a single processor as in Linden *et al.* [19, 20]; or to have idle processors, such as used by Brown *et al.* [21]. Even for the simple application considered in [21] the algorithm scaled better for 1-d lines solved in serial. Prieto *et al.* [22] noted that they used the critical level as their coarsest grid due to load balancing issues and that agglomeration "is more suitable to pointwise relaxation". However some codes, such as the NAS benchmark, do scale relatively well [23].

In the case of EHL problems the addition of MLMI causes extra difficulties as even more work is done at coarse mesh levels. It is already known that existing MLMI type operations may not scale well, [24]. In particular, since no significant computation is done during the MLMI coarsening, the communication costs of this process are a significant factor in terms of parallel efficiency. The key issue is thus that as we go to coarser grids the communications costs do not decrease as quickly as the computational costs, due to this extra overhead. In the explanation that follows it will be seen how the high order coarsening strategy required means that the communication halos are large, typically on the coarsest meshes even larger than a processor's own work array. Also there are global operations that require global knowledge, and local operations that require broadcasts from a small number of processors.

Besides the work described above, the only known previous parallel EHL solver was presented by Arenaz *et al.* [25] although, like the early work shown in [6] the time savings came from the parallelisation of the multi-summation, since neither used MLMI.

Stripwise Domain Decomposition

Assuming we are on multigrid level k then the half domain on which the solution is calculated is $2^{k+1}+1 \times 2^k+1$ points, i.e. $N_X^k \times N_Y^k$. The solution methods described above rely on a line solve in the direction of the fluid flow. This makes it natural to consider a stripwise decomposition, parallel to the direction of fluid flow. The decomposition explained below may not be ideal for parallel efficiency but is that used in the serial codes and is probably necessary for realistically fast EHL solutions when using the present relaxation schemes. The partitioning is such that $\frac{N_Y^k}{n_p}$ rows are thus allocated to each processor. Since the top row, $j = N_Y^k$ is a boundary line then not all solution variables are calculated here, meaning that if n_p, the total number of processors used, is of the form 2^n then the effective load balancing is heuristically equal between processors. Therefore the number points allocated to processor $p = 0, \ldots, n_p-1$ for computation are:

$$\mathcal{S}_p^k = N_X^k \times \frac{N_Y^k - 1}{n_p}. \tag{34}$$

The assignment of the set $\mathcal{S}_p^k$ is not the only memory requirement per processor. Many of the calculations need more information than is contained in $\mathcal{S}_p^k$. For instance, the solution of the discrete Reynolds Equation (6) requires density, viscosity, film thickness and pressure values at adjacent rows which may be located in $\mathcal{S}^k_{p\pm1}$. The requirements of the deformation calculation are discussed in more detail below.

The MLMI solve also uses a hierarchy of grids used to accelerate the calculation of the deformation. It was explained in the Multilevel Multi-integration Schemes Section above how it is necessary to use sixth order interpolation operators in multi-integration. These operators act on the coarsening of the pressure, the coarsening of the kernel matrix used in the correction area, and also the refinement of the deformations calculated on the coarser grids. These sixth order schemes therefore require up to three rows of ghost cells.

The use of the multigrid method means that each processor will need to calculate the solution of $\frac{1}{n_p}$th of each grid used. This means the inter-grid transfer operators must also scale easily. This has been accomplished by ensuring that inter-processor boundaries occur on mesh lines on the coarsest multigrid used, say k=C. This is again easily accomplished by choosing n_p to be of the form 2^n. Using parallelism with MLMI does, however place additional constraints on the parallelism strategies used. The halos required on a grid mean that, for an efficient algorithm in terms of memory usage, it has been necessary for each processor to have a more complex message passing structure to receive these dummy points from multiple processors. We have not implemented agglomeration-style techniques and so all processors are never idle. This, in turn, means that the level of coarsest grid used for both multigrid and MLMI is restricted by the need for each processor to have a non-trivial amount of work.

Other memory costs in the MLMI solve are incurred by the multi-summation having to be performed on grids $C \leqslant k \leqslant k^{sum}$. This means that on each of these grids there must be enough computational memory allocated for the complete pressure and kernel solutions to be stored. Also, in the entire multi-integration solve extra cells are used to extend the domain on each level for use in the sixth-order coarsening routines. Given this higher order method uses four extra points over every edge the domain is thus extended to be $M^k_X \times M^k_Y = (N^k_X + 8) \times (N^k_Y + 8)$ points. Therefore the number of points chosen for the multi-integration calculations are not as given in equation (34), but actually

$$\mathcal{T}^k_p = M^k_X \times \frac{M^k_Y}{n_p} \tag{35}$$

and hence there must be a small amount of realignment of data done at both the start and end of each deformation calculation.

PARALLEL COMPUTATIONAL COMPLEXITY

In this section the computational and communication costs of the parallel multigrid algorithm and of the multilevel integration are calculated. By combining these it will be possible to form a theoretical model of performance and scalability which will be able to be compared against the actual scalability of the software.

Assuming that we have n_p processors then, since all the computation has been parallelised, we can simply take the relevant fraction of the serial cost given by equation (32), as follows:

$$VC_{cost}^{parallel} = \frac{1}{n_p} \sum_{k=k_c}^{k_f} \gamma_k^{MG} \Big\{ \kappa_{vc} N_X^k N_Y^k + \kappa_{sum} \left[\left(M_X^{k_c} M_Y^{k_c} \right)^2 \right]$$
$$+ \sum_{K=k_{c+1}}^{k} M_X^K M_Y^K \left[\kappa_{trans} + \kappa_{corr} \left(3 + \log\left(M_X^K\right)\right) \left(3 + \log\left(M_Y^K\right)\right) \right] \Big\}. \tag{36}$$

Communication costs

Some communications requirements, such as the size of halos, have already been covered when discussing the partitioning of the domain. Here we cover the specific costs associated with the parallel implementation in detail. In describing these costs it is important to note that the communications costs from the top and bottom processors are approximately one half of the costs of the interior processors, although this has been neglected in the analysis to follow. The communications model used is the standard approximation in which the cost of sending N_x data items from one processor to another as denoted by C_{N_X} is defined by

$$C_{N_X}^{send} = \alpha_0 + \beta N_X \tag{37}$$

where $\alpha_0 \approx 10^{-5}, \beta \approx 2.510^{-8}$ and the cost of a floating point operation $\gamma \approx 10^{-10}$ on the machine for which we compare the model against the experimental results below. Clearly the number of communications and their associated costs are governed by the number of grid levels used in the multilevel scheme.

The communications model used for a broadcast of N data items from one processor to all the others is denoted by C_N^{BCast} is defined by

$$C_N^{Bcast} = \alpha_0 + 3log(p)(\beta N + \alpha_1) \tag{38}$$

where $\alpha_1 = 10^{-6}$. Let us define the total number of grids used in the solution scheme to be $k^{\mathrm{tot}} = k^{\mathrm{fi\,ne}} - k^{\mathrm{coarse}} + 1$.

In the parallel EHL code there are three parts to the communication pattern:

- Multigrid for the pressure and the fluid model;
- Multi-integration for film thickness evaluations;
- Force balance calculation to compute H_{00};

which are addressed in turn in the next three sections.

Pressure and fluid calculations

In the Numerical Methods Section it was explained how two different numerical schemes are used for the update for pressure, in and out of the contact region. For the Gauss-Seidel region it is necessary to have the boundary value updates for adjoining processors. Each processor will do a send and a receive of N_X^k pressure points to and from adjoining processors. Similarly, the communication requirements for viscosity and density along with the film thickness are limited to filling the ghost points over processor boundaries, hence the cost of doing each of these is the same as for the pressure given in equation (38).

MLMI communications requirements

The multi-integration solve to calculate the deformation requires communications down to the coarsest grid and back up. The level on which the deformation is to be calculated is denoted by k^{eval} and that the coarsest level used in the multi-integration solve, (hence the level on which a multi-summation is performed), is denoted by k^{sum}, where $k^{\text{eval}} < k^{\text{sum}}$.

The sixth order smoothing operations defined by equation (22) used mean that the overlap between partitions consists of at least four rows of ghost cells above and four below. These are needed for both the coarsening of the pressure and kernel and also of the restriction of the deformations back to the fine grid.

The communications are broken down into three main parts: the coarsening, the refinement and the grid alignment. This last part comes from the attempt to equidistribute work between processors, given by the difference between Equations (34) and (35). The overall cost of this is small as the difference between $\mathcal{S}_p^k$ and $\mathcal{T}_p^k$ will rarely be more than a couple of rows for fine meshes or large numbers of processors. These transfers are also done by non-blocking local communications.

The coarsening work is divided between the coarsening of the kernel and the coarsening of the pressures. The kernel actually requires coarsening by two different procedures, namely injection and high order coarsening.

Straight injection is used in the multi-summation of equation (23). The injected kernel, therefore, requires global broadcasting on the coarsest grid.

Sixth-order coarsened kernels are required for the correction of the calculated deformations computed using equations (26) and (27). The sixth-order versions are therefore only required to be valid up to the width of the correction patch. This means that only the first two processor's partitions are will be required, however these must be replicated to all the other processors on all grid levels in order to compute the corrections.

MLMI coarse mesh halos

For the correction part of the multilevel multi-integration solve it is necessary to use a multi-summation of all points within a much larger radius than are used in the sixth-order coarsening of the MLMI solve. The difficulty is that halos of size $4 + \log N$ on the coarsest grid correspond to halos of between 12 and 20 points on the finest grids used in the line contact solution domains. These larger requirements are needed for both the coarsened pressure and the coarsened kernel functions. An inexpensive MPI method for dealing with these halos will now be shown.

The information per processor is as shown in Figure 3 for four processors. In this diagram each processor owns a subset of the information it requires. For example, processor 3 knows all the values for rows *b3* to *c3* but actually needs all the information

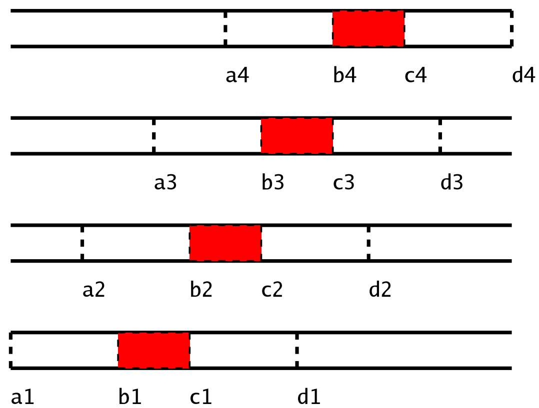

Figure 3: Information owned and required, per processor

for *a3* to *d3* e.g. the 20 row halos on either side, which are owned, by processors 1 to 4.

The proposed solution for this to share the information on the required rows with all the other processors, and then gather back the information, a row at a time. Since each row is needed by multiple processors, it is necessary to know where we are intending to send that row if global broadcasts are to be avoided.

The algorithm for processor i is

- Distribute arrays $\underline{a}$ and $\underline{d}$ –
  $2 \times$ MPI_AllGather

- `for` j `= 0,` n_p `(i` $\neq$ `j)`
  `if (`$b_i > a_j$ `&&` $c_i < d_j$`)`
  `for k = MAX(`b_i`,` a_j`) to MIN (`c_i`,` d_j`)`
  `MPI_ISend row` k `to proc` j`, tag` k

- `for` $k = a_i$ `to` d_i `(i` $\neq$ `j)`
  `MPI_IRecv row` k `from ANY, tag` k

This changes the communication costs from n_p processors doing a Bcast (i.e. $2n_p$ messages per processor sent and received of length N^2/n_p) to 2 AllGathers (i.e. $2n_p$ messages, length 2 ints) and twice the correction box length Isends of length N.

The benefit of this communication method is that when it is used for the sixth-order coarsening/refinement halos it enables the efficient gathering information from multiple processors, and distribution to a local neighbourhood of processors, rather than the exchange of information only between adjacent processors.

Refinement of the calculated deformations from k^{coarse} back to k^{eval} are again done via half grids. These transfers require only the ghost cell rows needed for the sixth order refinement. Hence the bottom three rows need to be sent to the processor calculating the partition below and the top two rows to the partition calculating above.

Calculation	Operation	When used	Length	Messages in/out
V-cycle				
Force Balance	All_Reduce	each coarsening	1	1
Viscosity	Isend/Irecv	each smooth	N_X^k	2
Density	Isend/Irecv	each smooth	N_X^k	2
Pressure				
- calculation	Isend/Irecv	each smooth	N_X^k	2
	local_gather	each smooth	N_X^k	24
MLMI cycle				
Kernel coarsening	Bcast (from P0)	each grid and half-grid	$8\times M_X^k$	1
Pressure coarsening	local_gather	per grid	M_X^k	24
		per half-grid	M_X^k	12
Coarse grid multisum.	Bcast	each MLMI	$\mathcal{R}^{coarse}$	$2n_p$
Deformation prolong.	local_gather	each grid and half-grid	M_X^k	10

Table I Communications Cost on Grid k

MLMI coarse mesh broadcasts

On the coarsest mesh of the MLMI iteration it is necessary to communicate the coarsened pressures and kernel functions to all the processors. This is because on the coarsest grid a multi-summation of the product of these two arrays is needed. Whilst the relevant arrays of coarsened kernels may be stored on each processor, the extra memory requirements of saving coarsened kernels solutions would also become highly prohibitive as both coarse and fine grids become increasingly refined.The need to broadcast the coarse grid pressures on every solve makes it just as easy to broadcast the kernel too.

Force balance calculation

In solution of the Force Balance equation (13), parallel communication is restricted to global broadcasts of each processor's contribution to the H_{00} correction, as defined by equations (20) and (21), on both fine and coarse grids every time the pressure is coarsened in the multigrid cycle. There is also a global broadcast of each processors sum of pressures on the coarsest grids. These broadcast values are then combined on each processor to update H_{00} identically.

Each processor sends one double precision number out and receives $n_p - 1$ back for every grid giving combined communication cost of

$$VC_{comm}^{H_{00}} = \left(k_{dif} + 1\right)\left(3\alpha_0 \log(n_p)\right) \tag{39}$$

Combining the communications costs

A summary of the communications costs are given in Table I. Gathering together these costs for the combined V-cycle and associated MLMI calculations gives the communications costs of the parallel multigrid algorithm and of the multilevel integration. Excluding the deformation calculation, for each processor the following costs are incurred on grid level k in a multigrid V-cycle are:

$$
\begin{aligned}
VC_{comm}^{non-def} &= \sum_{k=k^c}^{k^f} \left[12\gamma_k^{MG} \left(\alpha_0 + \beta 2 N_X^k \right) + 3\alpha_0 \log(p) \right] && (40) \\
&= \sum_{k=k^c}^{k^f} \left[12\gamma_k^{MG}\alpha_0 + 12\beta\gamma_k^{MG} N_X^k + 3\alpha_0 \log(p) \right] && (41) \\
&= 12\alpha_0 \sum_{k=k^c}^{k^f} \gamma_k^{MG} + 12\beta \sum_{k=k^c}^{k^f} \gamma_k^{MG} N_X^k + 3\alpha_0 \log(p) k_{dif}. && (42) \\
& && (43)
\end{aligned}
$$

Now,

$$
\gamma_k^{MG} = \begin{cases} n_{pre} + n_{post} + 2, k \neq k^c \\ n_{coarse}, k = k^c \end{cases} \tag{44}
$$

which in a V(3,1,30) cycle gives:

$$
\gamma_k^{MG} = \begin{cases} 6, k \neq k^c \\ 30, k = k^c. \end{cases} \tag{45}
$$

Substituting (42) into (45) gives:

$$
\begin{aligned}
VC_{comm}^{non-def} &= 12\alpha_0 \left(k_{dif} 6 + 30 \right) + 12\beta \left(12 N_X^{k^f} + 24 N_X^{k^c} \right) + 3\alpha_0 \log(p) k_{dif} && (46) \\
&= 72\alpha_0 \left(k_{dif} + 5 \right) + 144\beta \left(2^{k^f} + 2^{k^c+1} \right) + 3\alpha_0 \log(p) k_{dif}. && (47)
\end{aligned}
$$

The MLMI costs for the calculation on grid K are given by:

$$
\begin{aligned}
VMLMI_{comm}^K &= \text{local gather of rows} \\
&\quad + \text{local gather all reduces} \\
&\quad + \text{kernel broadcasts from P0 for correction patches} \\
&\quad + \text{coarsest grid kernel and pressure} \\
&= \sum_{k=k^c+1}^{K} \gamma_{MI}^k \left(\alpha_0 + \beta M_X^k \right) + \sum_{k=k^c+1}^{K} 4 \left(\alpha_0 + 3\beta \log(p) M_X^k \right) \\
&\quad + \sum_{k=k^c}^{K} 2 \left[\alpha_0 + 3 \log(p) \left(\alpha_0 + 8 M_X^k \right) \right] \\
&\quad + 2p \left\{ \alpha_0 + 3 \log(p) \left[\alpha_0 + \frac{\left(M_X^k \right)^2}{p} \right] \right\} && (48) \\
&= [A] + [B] + [C] + [D] && (49)
\end{aligned}
$$

Taking these terms in sequence, and noting that γ_{MI}^{k} is the number of rows sent plus the number of rows received on grid k, i.e. $24+12+10+10=56$, and $\overline{k}_{dif}=K-k^{c}$ gives:

$$\begin{aligned}
[A] &= \sum_{k=k^c+1}^{K} \gamma_{MI}^{k}\left(\alpha_0+\beta M_X^k\right) \\
&= 56\alpha_0\overline{k}_{dif}+56\beta\sum_{k=k^c+1}^{K} M_X^k \\
&= 56\alpha_0\overline{k}_{dif}+56\beta 2M_X^K-56\beta 2M_X^{k^c}
\end{aligned} \tag{50}$$

$$\begin{aligned}
[B] &= \sum_{k=k^c+1}^{K} 4\left(\alpha_0+3\beta\log(p)M_X^k\right) \\
&= 4\alpha_0\overline{k}_{dif}+12\beta\log(p)\left(2M_X^K-M_X^{k^c}\right) \\
&= 4\alpha_0\overline{k}_{dif}+24\beta\log(p)M_X^K-12\beta\log(p)M_X^{k^c}
\end{aligned} \tag{51}$$

$$\begin{aligned}
[C] &= \sum_{k=k^c}^{K} 2\left[\alpha_0+3\log(p)\left(\alpha_0+8M_X^k\right)\right] \\
&= 2\alpha_0\overline{k}_{dif}+6\log(p)\alpha\overline{k}_{dif}+6\log(p)\beta\left(16M_X^K-8M_X^{k^c}\right) \\
&= 2\alpha_0\overline{k}_{dif}\left(1+6\log(p)\right)+48\beta\log(p)\left(2M_X^K-M_X^{k^c}\right)
\end{aligned} \tag{52}$$

$$\begin{aligned}
[D] &= 2p\left\{\alpha_0+3\log(p)\left[\alpha_0+\beta\frac{\left(M_X^k\right)^2}{p}\right]\right\} \\
&= 2p\alpha_0\left(1+3\log(p)\right)+6\beta\log(p)\left(M_X^{k^c}\right)^2
\end{aligned} \tag{53}$$

Therefore combining (50), (51), (52) and (53) and substituting into (48) gives

$$\begin{aligned}
VMLMI_{comm}^{K} &= \begin{array}{lll}
56\alpha_0\overline{k}_{dif} & +112\beta M_X^K & -56\beta M_X^{k^c} \\
+4\alpha_0\overline{k}_{dif} & +24\beta\log(p)M_X^K & -12\beta\log(p)M_X^{k^c} \\
+2\alpha_0\overline{k}_{dif}\left(1+6\log(p)\right) & +96\beta\log(p)M_X^K & -48\beta\log(p)M_X^{k^c} \\
+2p\alpha_0\left(1+3\log(p)\right) & & +6\beta\log(p)\left(M_X^{k^c}\right)^2
\end{array} \\
&= 2\alpha_0\left(62\overline{k}_{dif}+6\overline{k}_{dif}\log(p)+p+3p\log(p)\right) \\
&\quad +8\beta M_X^K\left(14+15\log(p)\right) \\
&\quad -2\beta M_X^{k^c}\left(28+30\log(p)-3\log(p)M_X^{k^c}\right)
\end{aligned} \tag{54}$$

The MLMI costs over a complete V-cycle are then given by:

$$
\begin{aligned}
VC_{MLMI} &= \sum_{K=k^c}^{k^f} \gamma_k^{MG} VMLMI_{comm}^K \\
&= \sum_{K=k^c+1}^{k^f} 6VMLMI_{comm}^K + 30\left[2p\alpha\left(1+3\log(p)\right) + 6\beta\log(p)\left(M_X^{k^c}\right)^2\right] \\
&= 12\alpha_0 p\left(1+3\log(p)\right)k_{dif} \\
&\quad +2\alpha_0 \sum_{K=k^c+1}^{k^f}\left(62K - 62k^c + 6K\log(p) - 6k^c\log(p)\right) \\
&\quad +48\beta\left(14+15\log(p)\right)\left(2M_X^{k^f} - M_X^{k^c}\right) \\
&\quad -12\beta M_X^{k^c}\left(28+30\log(p) - 3\log(p)M_X^{k^c}\right) \\
&\quad +30\left(2p\alpha\left(1+3\log(p)\right) + 6\beta\log(p)\left(M_X^{k^c}\right)^2\right) \\
&= 12\alpha_0 p\left(1+3\log(p)\right)k_{dif} \\
&\quad +2\alpha_0 31\left(1+\log(p)\right)\left(k^f k^{f+1} - k^c k^{c+1}\right) - 62k^c k_{dif}\left(1+\log(p)\right) \\
&\quad +48\beta\left(14+15\log(p)\right)\left(2M_X^{k^f} - M_X^{k^c}\right) \\
&\quad -12\beta M_X^{k^c}\left(28+30\log(p) - 3\log(p)M_X^{k^c}\right) \\
&\quad +30\left(2p\alpha\left(1+3\log(p)\right) + 6\beta\log(p)\left(M_X^{k^c}\right)^2\right) \\
&= 12\alpha_0 p\left(1+3\log(p)\right)\left(k_{dif}+5\right) \\
&\quad +62\alpha_0\left(1+\log(p)\right)\left(k^f k^{f+1} - k^c k^{c+1} - 2k^c k_{dif}\right) \\
&\quad +12\beta\left[2M_X^{k^f}\left(14+15\log(p)\right) + \right. \\
&\quad \left. 6M_X^{k^c}\left(3\log(p)M_X^{k^c} - 14 - 15\log(p)\right)\right] \qquad (55)
\end{aligned}
$$

The total communications cost of a V-cycle is given by adding equations (47) and (55) to give:

$$
\begin{aligned}
VC_{comm} &= 72\alpha_0\left(k_{dif}+5\right) \\
&\quad +144\beta\left(2^{k^f} + 2^{k^c+1}\right) \\
&\quad +3\alpha_0\log(p)k_{dif} \\
&\quad +12\alpha_0 p\left(1+3\log(p)\right)\left(k_{dif}+5\right) \\
&\quad +62\alpha_0\left(1+\log(p)\right)\left(k^f k^{f+1} - k^c k^{c+1} - 2k^c k_{dif}\right) \\
&\quad +12\beta\left[2M_X^{k^f}\left(14+15\log(p)\right) + \right. \\
&\quad \left. 6M_X^{k^c}\left(3\log(p)M_X^{k^c} - 14 - 15\log(p)\right)\right] \\
&= \alpha_0\left[72\left(k_{dif}+5\right) + 3\log(p)k_{dif} + 12p\left(1+3\log(p)\right)\left(k_{dif}+5\right)\right.
\end{aligned}
$$

$$+\,62\,(1+\log(p))\left(k^f k^{f+1} - k^c k^{c+1} - 2k^c k_{dif}\right)\Big]$$
$$+12\beta\Big[12\left(2^{k^f} + 2^{k^c+1}\right) + 2M_X^{k^f}\,(14 + 15\log(p))$$
$$+\,6M_X^{k^c}\left(3\log(p)M_X^{k^c} - 14 - 15\log(p)\right)\Big]. \qquad (56)$$

Memory Costs

The efficient distribution of the memory requirements for the EHL code was challenging in that the trade-offs between memory and global communication (e.g. due to the coarse grid kernel). nature of some of the operations described above. Only once the communication algorithm had been constructed could the parallel memory issues be tackled. The need to reach as fine a grid as possible meant that the memory allocation model needed to be efficient since the presence of the coarser meshes will cause the memory per processor to grow by more than the extra resolution needed for the finest grid alone. In fact, being able to efficiently use the memory for large numbers of processors on fine grid cases is perhaps equally important as the parallel algorithm scaling. These factors will be discussed in the next section.

Defining the standard processor share on a grid j to be:

$$\mathcal{R}^j = \frac{N_Y}{n_p} \times N_X$$

then it is possible to define the size of almost all the storage to arrays of size

$$\mathcal{D}^j = \mathcal{R}^j + 2N_X$$

The factor of 2 represents one row above and below to be passed to neighbouring processors.
The only important exceptions that may be greater than this are as follows:

Pressure	$\mathcal{D}_P^j = \mathcal{R}^j + 17 \times 2(N_X + 16)$
Deformation	$\mathcal{D}_\delta^j = \mathcal{R}^j + 4 \times 2(N_X + 16)$
MLMI Kernel	$\mathcal{D}_k^j = \mathcal{R}^j + 4 \times 2(N_X + 16)$
MLMI corrections	$\mathcal{D}_{corr}^j = 9 \times (N_X + 16)$

Some other work arrays are larger than $\mathcal{R}^j$ but are only needed for the grid being used, hence their total size is constrained to $\mathcal{R}^{finest}$ rather than $\sum_{j=coarse}^{finest} \mathcal{R}^j$.

RESULTS

In this section computational results are presented for the example EHL test problem, which corresponds to the calculation of the initial steady state conditions of the ex-

Table I: $K_{tot} = 257 \times 257$

Number of	Snowdon		NGS		Memory		K_{sum}
n_p	Time	Effic.	Time	Effic.	Mb	Iso-memory	Grid Used
1	12.87	1.00	9.09	1.00	12	1.00	33x33
2	6.45	1.00	4.93	0.92	7	0.86	33x33
4	4.34	0.74	3.60	0.63	5	0.60	33x33
8	3.79	0.42	3.27	0.35	3	0.50	33x33
16	5.01	0.16	4.61	0.12	3	0.25	33x33
32	9.70	0.04	6.09	0.05	3	0.13	65x65
64	16.19	0.01	9.19	0.02	4	0.05	128x128
128	29.89	0.00	17.49	0.00	4	0.02	256x256

Table II: $K_{tot} = 513 \times 513$

Number of	Snowdon		NGS		Memory		K_{sum}
n_p	Time	Effic.	Time	Effic.	Mb	Iso-memory	Grid Used
1	45.71	1.00	32.10	1.00	46	1.00	33x33
2	22.32	1.02	17.21	0.93	24	0.96	33x33
4	13.53	0.85	10.52	0.76	14	0.82	33x33
8	9.42	0.61	7.82	0.51	8	0.72	33x33
16	9.86	0.29	9.46	0.21	6	0.48	33x33
32	16.02	0.09	9.96	0.10	5	0.29	65x65
64	25.98	0.03	15.24	0.03	5	0.14	128x128
128	135.11	0.00	70.23	0.00	9	0.04	256x256

ample of reversal solved by Scales *et al.* [26]. These are compared to the theoretical predictions of our model in the following section.

In all the solutions to follow the numerical solver successfully converged on all grids with all numbers of processors tested without producing 'incorrect' solutions. The parallel code has been tested on a variety of machines. The two architectures reproduced here are both distributed memory Linux clusters. The first, `snowdon`, has up to 128 dual processor nodes containing two Intel P4 2.2GHz Xeon processors with 0.5Mb of secondary cache and 2GB of physical memory, with all nodes connected via Myrinet 2000. The second machine, `NGS`, is the Leeds node of the UK's National Grid Service, which is similar to `snowdon` but with processor speeds of 3.06GHz. The Intel compiler has been used on both machines, with identical optimisation levels.

Results are shown in Tables I to VII for grids All timings are for FMG followed by 10 multigrid V-cycles using the coarsest possible grids allowed, from a coarsest level as given in the final column of each table. The efficiencies are compared against the case using the lowest number of processors that would fit into the 2Gb memory of each node. This means that the efficiencies for 128 processors have a finer coarsest mesh used than for lower numbers of processors. This only harms the performance since the computational cost on the coarsest mesh is $O(N^4)$ as discussed earlier.

The above results are representative of the true speed-ups possible due to parallelism when using many processors. The scalability of the code can also be assessed by considering how the solution time is affected when the problem size is increased

Table III: $K_{tot} = 1025 \times 1025$

Number of	Snowdon		NGS		Memory		K_{sum}
n_p	Time	Effic.	Time	Effic.	Mb	Iso-memory	Grid Used
1	174.23	1.00	121.82	1.00	178	1.00	33x33
2	84.15	1.04	64.28	0.95	92	0.97	33x33
4	48.66	0.90	36.04	0.85	48	0.93	33x33
8	28.54	0.76	21.48	0.71	26	0.86	33x33
16	22.24	0.49	18.69	0.41	15	0.74	65x65
32	28.24	0.19	17.09	0.22	10	0.56	65x65
64	42.12	0.06	23.36	0.08	9	0.31	128x128
128	226.76	0.01	108.26	0.01	12	0.12	256x256

Table IV: $K_{tot} = 2049 \times 2049$

Number of	Snowdon		NGS		Memory		K_{sum}
n_p	Time	Effic.	Time	Effic.	Mb	Iso-memory	Grid Used
1	672.86	1.00	514.10	1.00	705	1.00	65x65
2	367.77	0.91	253.87	1.01	357	0.99	65x65
4	174.39	0.96	133.57	0.96	182	0.97	65x65
8	102.61	0.82	73.97	0.87	95	0.93	65x65
16	66.27	0.63	42.36	0.76	51	0.86	65x65
32	78.76	0.27	35.64	0.45	29	0.76	65x65
64	125.86	0.08	40.87	0.20	20	0.55	128x128
128	314.63	0.02	155.51	0.03	19	0.29	256x256

Table V: $K_{tot} = 4097 \times 4094$

Number of	Snowdon		NGS		Memory		K_{sum}
n_p	Time	Effic.	Time	Effic.	Mb	Iso-memory	Grid Used
1	-		-		*2807*		65x65
2	1520.87	1.00	1073.58	1.00	1413	0.99	65x65
4	701.71	1.08	554.05	0.97	713	0.98	65x65
8	402.81	0.94	278.07	0.97	363	0.97	65x65
16	228.59	0.83	146.91	0.91	188	0.93	65x65
32	163.25	0.58	98.67	0.68	101	0.87	65x65
64	142.93	0.33	79.96	0.42	59	0.74	128x128
128	410.41	0.06	237.40	0.07	41	0.53	256x256

Table VI: $K_{tot} = 8193 \times 8193$

Number of	Snowdon		NGS		Memory		K_{sum}
n_p	Time	Effic.	Time	Effic.	Mb	Iso-memory	Grid Used
1	-		-		*11202*		
8	1798.74	1.00	1147.41	1.00	1424	0.98	65x65
16	952.35	0.94	586.60	0.98	725	0.97	65x65
32	551.11	0.82	357.94	0.80	375	0.93	65x65
64	338.32	0.66	233.18	0.62	202	0.87	128x128
128	640.33	0.18	338.47	0.21	119	0.74	256x256

Table VII: $K_{tot} = 16385 \times 16385$

Number of n_p	Snowdon Time	Snowdon Effic.	NGS Time	NGS Effic.	Memory Mb	Memory Iso-memory	K_{sum} Grid Used
1	-		-		*44761*		
32	2092.52	1.00	1310.16	1.00	1449	0.97	65x65
64	1365.05	0.77	774.59	0.85	751	0.93	128x128
128	1297.46	0.40	672.61	0.49	406	0.86	256x256

Table VIII: Comparison of timings on increasing fine grid level with coarse grid always 257x257. For each case three timings are shown: the broadcasts for the multi-summation (top), the local gather timings (middle) and the other computation and local communication (bottom)

N_p	513×513	1025×1025	2049×2049	4097×4097	8193×8193	16385×16385
32	4.04	5.60	7.48	9.49	11.77	13.82
	2.55	6.16	14.84	42.84	119.56	386.28
	226.30	317.20	440.14	683.35	1265.46	3409.13
64	7.09	10.02	13.48	17.01	19.94	24.99
	3.12	7.43	15.42	40.17	98.40	334.82
	185.07	244.40	339.51	492.40	881.34	1922.91
128	-	17.14	22.99	29.29	35.54	42.37
	-	9.19	19.13	39.57	92.78	210.37
	-	183.61	250.07	319.19	529.75	1030.94

at the same rate as the number of processors used to solve the problem. For the EHL problem as solved here, doubling the number of points in the domain does not double the work required as the multigrid method itself is assumed to be $O(N^2)$ whilst the MLMI solve is $O(N^2 \ln N^2)$. However, bearing these facts in mind it is not unreasonable to still compare grid levels k on p processors against grid $k+1$ on $p\times4$, since each grid has four times the number of mesh points.

Another set of results worth comparing are those for a fixed coarsest mesh. Here we have used an alternatively compiled version to break down the timings into three parts, namely that for the broadcasts before the multi-summation, the local gather operations during the MLMI deformation calculation and thirdly the rest of the computation and local communication. These results are shown in Table VIII for the coarsest grid used of 257×257 points. It can be seen how increasing the fine grid for fixed number of processors leads to good scaling of the broadcasts and the computation, however the local gather cost is growing fastest. Meanwhile with increasing numbers of processors it is only the broadcast time which grows as expected with the main performance cost showing a good scaling.

Looking again at Tables I to VII it is possible to assess how successful the use of the distributed memory on the system has been. It is seen that in all cases the iso-memory figure, given by

$$\text{Isomemory} = \frac{\text{memory with one processor}}{\text{memory with p processors}}, \tag{57}$$

is significantly better than the computational efficiency. More importantly on the finest meshes for which the memory model was devised, i.e. those grids which couldn't have been stored completely on a single node, then the memory efficiency is good even when high numbers of processors are being used. These figures lead us to believe that the code would be extensible to successfully run on much larger systems for much finer problems.

PERFORMANCE MODEL COMPARISON

In this section we draw together the communications and computations costs of earlier and compare them against the experimental results achieved. We then use these results to make further predictions about how scalable the code may be on finer grid with more processors.

The efficiency of the model we have defined using

$$E_{ff} = \frac{1}{1 + \frac{VC_{cost} \times n_p}{VC_{comm}}} \tag{58}$$

for n_p processors, and VC_{cost} and VC_{comm} defined using equations (32) and (56) respectively.

The results of the computational experiments on `snowdon`, shown in Tables I to VII, are compared against the performance model, in Figure 4 where the appropriate α_0, β and γ chosen. For all the cases we have used the same coarsest grid as was usable in the minimum processor case for comparison purposes, and it is against these results that the model has been compared.

It can be clearly seen what a close correlation there is between the two sets of results. This is especially pleasing since only the main computational elements have been included into the model, and estimates of the operations counts were made independently of the parallel timings.

With such close agreement, even on high number of processors with the finest meshes being used, gives us good confidence in being able to make predictions of how the solution algorithm may behave on finer grids with more processors being used. What we are able to learn from the model and the other results concerning memory and the breakdown of timings is that the efficiency of the parallel code at larger numbers of processors will be good for even finer grids than have been tackled thus far. The isomemory results indicate that we ought to be able to fit the memory requirements into that available for a single node, assuming that the memory per node is not significantly less than is currently available.

In Figure 5 we see how the performance model expects the software to behave on grids finer than those for which we have experimental results, and also on more processors. The assumption has been made that these problems will fit into the locally available memory irregardless of the number of processors used (since we are using a hypothetical extension to a real computer system supplying the α_0, β and γ values) hence the coarsest grid used has been kept fixed at 257×257 points. The coarsest grids used have all been increased following the same rules as in our current experimental

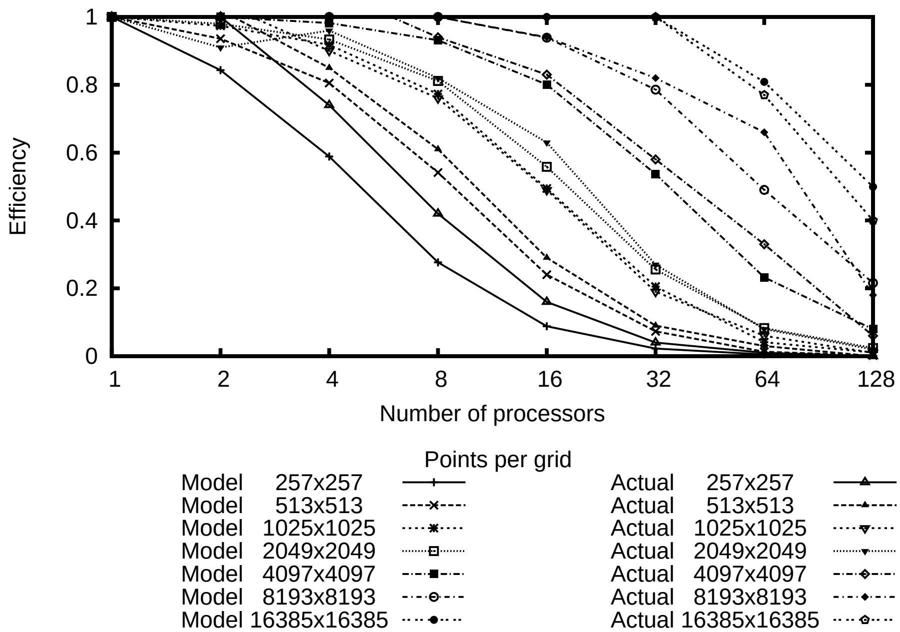

Figure 4: Comparison of the performance model to the actual parallel experiments

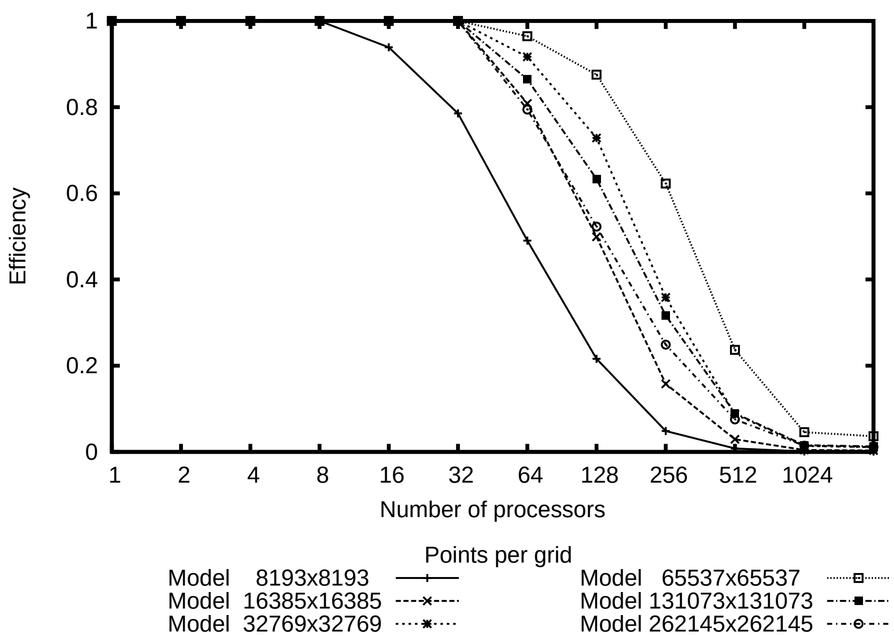

Figure 5: Performance model prediction for very fine grids and high numbers of processors

rules. What the results show is that the parallel performace of the code is expected to peak at 65537×65537 points, beyond which the global operations on these very long rows will have become too expensive for the best scalability. The model does show, however, that 50% efficiency may be possible for real surface roughness cases of over 10^{10} mesh points on 128 large memory nodes. These efficiencies would appear improved if the minimum number of processors necessary was larger, as would currently be the case.

CONCLUSIONS

In this paper we have shown that a demanding numerical problem, which is both highly intensive in terms of communication, and requires significant global communications, has been successfully parallelised. Communication costs have been limited through use of non-blocking local directives, and the memory requirements per process have been significantly reduced.

The overall speed-up of the code is excellent, especially on higher grid resolutions, such as will be required to tackle real surface roughness problems. The scalability has been shown to be similarly good with comparable results when increasing the problem size and number of processors whilst utilising the same coarsest MLMI level.

A parallel model has been presented that shows very similar behaviour to the computational results obtained. It has been seen how the change of coarsest grid used due to the multigrid critical level changing makes an impact on performance while still giving good scalability when the fine mesh is varied relative to the coarse grid used.

We have now been able to solve the largest EHL point contact cases that the authors know about. The future holds three main directions for this work. It is clear that to tackle very fine mesh levels, large amounts of physical memory are required on the individual computers. To progress further on the distributed architectures available then the computational model developed here could be analysed in great detail to develop more efficient communications models for large number of processors.

The second direction is to start using these very fine meshes to solve real surface roughness problems. To solve these accurately in a transient manner will probably require spatial adaptivity [27] and variable timestepping [10,28] to be introduced to the parallel solver. These problems may thus require even larger machines to be employed to handle such enormous meshes quickly, and hence moving to meta-computing on the Grid will seem an obvious next stage.

A final idea to be considered will be for improving the current solver by have varying numbers of processors per grid. This would eradicate the need to use finer coarsest grids in the calculation if some processors could be ‘switched out’ for grid levels where too many processors are present.

Acknowledgements

The authors wish to thank Roger Fairlie, Laurence Scales, Shell Global Solutions and EPSRC.

References

[1] Alsaad, M., Bair, S., Sanborn, D. M. and Winer, W. O. Glass transitions in lubricants: its relation to elastohydrodynamic lubrication (EHD). Trans. ASME, J. Lubric. Technol., 1978, 100, 404–417.

[2] Evans, H. P. and Snidle, R. W., Analysis of micro-elastohydrodynamic lubrication for engineering contacts, Tribology International, 29, 8, 659–667, 1996

[3] Fang, N. and Chang, L. and Johnston, G. J., Some insights into micro-EHL pressures, ASME Journal of Tribology, 121, 3, 473–480, 1999

[4] Spikes, H., Tribology Research in the twenty-first century, Tribology International, 34, 789–799, 2001

[5] Venner, C. H. and Lubrecht, A. A., Multilevel Methods in Lubrication, Elsevier, ISBN 0-444-50503-2, 2000,

[6] Goodyer, C. E., Adaptive Numerical Methods for Elastohydrodynamic Lubrication, University of Leeds, Leeds, England, 2001

[7] Nurgat, E. and Berzins, M. and Scales, L. E., Solving EHL Problems Using Iterative, Multigrid and Homotopy Methods, Trans. ASME, Journal of Tribology, 121, 1, 28–34, 1999

[8] Dowson, D. and Taylor, C. M., Cavitation in Bearings, Annual Review of Fluid Mechanics, 11, 35–66, 1979

[9] Brandt, A. and Lubrecht, A. A., Multilevel Matrix Multiplication and Fast Solution of Integral Equations, Journal of Computational Physics, 90, 348–370, 1990

[10] Goodyer, C. E. and Berzins, M. , Adaptive Timestepping for Elastohydrodynamic Lubrication Solvers SIAM Journal on Scientific Computing, (accepted)

[11] Venner, C. H., Multilevel Solution of the EHL Line and Point Contact Problems, University of Twente, Endschende, The Netherlands, ISBN 90-9003974-0, 1991

[12] Trottenberg, U. and Oosterlee, C. and Schuller, A., Multigrid, Academic Press, ISBN 0-12-701070-X, 2001,

[13] Goodyer, C. E. and Fairlie, R. and Berzins, M. and Scales, L. E., An In-depth investigation of the Multigrid approach for steady and transient EHL problems, Thinning films and Tribological Interfaces, Proceedings of the 26^{th} Leeds-Lyon Symposium on Tribology, D. Dowson *et al.*, Elsevier,95–102, 2000

[14] Goodyer, C. E. and Wood, J. and Berzins, M. , A Parallel Grid Based PSE for EHL Problems, Applied Parallel Computing, Proceedings of PARA ’02, Lecture Notes in Computer Science, 2367, Fagerholm, J. and Haataja, J. and Jarvinen, J. and Lyly, M. and Rback P. and Savolainen, V., Springer, 523–532, 2002

[15] McBryan, O. A. and Frederickson, P. O. and Linden, J. and Schuller, A. and Solchenbach, K. and Stuben, K. and Thole, C.-A. and Trottenberg, U., Multigrid Methods on Parallel Computers – A Survey of Recent Developments, Impact of Computing in Science and Engineering, 3, 1–75, 1991

[16] Llorente, I. M. and Prieto-Matías, M. and Diskin, B., An efficient parallel multigrid solver for 3-d convection-dominated problems, ICASE, TR-2000-29, 2000,

[17] Llorente, M. and Tirado, F. and Vázquez, L., Some Aspects about the Scalability of Scientific Applications on Parallel Computers, Parallel Computing, 22, 1169–1195, 1996

[18] Tuminaro, R. S. and Womble, D. E., Analysis of the multigrid FMV cycle on large-scale parallel machines, SIAM Journal of Scientific Computation, 14, 5, 1159–1173, 1993

[19] Linden, J. and Lonsdale, G. and Ritzdorf, H. and Schuller, A., Block-structured multigrid for the Navier-Stokes equations: experiences and scalability questions, Visualization Development Environments 2000 Proceedings, Proceedings of the Conference on Parallel Computational Fluid Dynamics 1992, Elsevier Science Publishers B.V., Amsterdam, 1992

[20] Linden, J. and Lonsdale, G. and Ritzdorf, H. and Schuller, A., Scalability Aspects of Parallel Multigrid, Future Generation Computer Systems, 10, 4, 429–449, 1994

[21] Brown, P. N. and Falgout, R. D. and Jones, J. E., Semicoarsening Multigrid on Distributed Memory Machines, SIAM Journal on Scientific Computing, 21, 5, 1823-1834, 2000

[22] Prieto, M. and Montero, R. S. and Llorente, I. G., A parallel multigrid solver for viscous flows on anisotropic structured grids, ICASE Report, 2001-34, 2001,

[23] Chamberlain B.L., Deitz S. and Snyder L. A Comparative Study of the NAS Benchmark across Parallel Languages and Architectures. In Proceedings of the ACM Conference on Supercomputing, 2000

[24] Jess A. Izaguirre and Scott S. Hampton and Thierry Matthey, Parallel Multigrid Summation for the N-body Problem, accepted, Journal of Parallel and Distributed Computing, 2005.

[25] Arenaz, M. and Doallo, R. and Touriño, J. and Vázquez, C., Efficient parallel numerical solver for the elastohydrodynamic Reynolds-Hertz problem, Parallel Computing, 27, 1743–1765, 2000

[26] Scales, L. E. and Rycroft, J. E. and Horswill, N. R. and Williamson, B. P., Simulation and observation of transient effects in elastohydrodynamic lubrication, SP-1182, SAE International Fuels and Lubricants Meeting, Dearborn, Michigan, 23–34, 1996

[27] Goodyer, C. E. and Fairlie, R. and Berzins, M. and Scales, L. E., Adaptive Techniques for Elastohydrodynamic Lubrication Solvers, Tribology Research: From Model Experiment to Industrial Problem, Proceedings of the 27^{th} Leeds-Lyon Symposium on Tribology, Eds G. Dalmaz *et al.*, Elsevier, 2001

[28] Goodyer, C. E. and Fairlie, R. and Berzins, M. and Scales, L. E., Adaptive Mesh Methods for Elastohydrodynamic Lubrication, ECCOMAS CFD 2001 : Computational Fluid Dynamics Conference Proceedings, Institute of Mathematics and its Applications, ISBN 0-905-091-12-4, 2001