INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2008; **56**:979–989
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.1737

Development and application of a parallel multigrid solver for the simulation of spreading droplets

P. H. Gaskell[1], P. K. Jimack[2,*,†], Y.-Y. Koh[1] and H. M. Thompson[1]

[1]*School of Mechanical Engineering, University of Leeds, Leeds LS2 9JT, U.K.*
[2]*School of Computing, University of Leeds, Leeds LS2 9JT, U.K.*

SUMMARY

We consider the parallel application of an efficient solver developed for the accurate solution of a range of droplet spreading flows modelled as a coupled set of nonlinear lubrication equations. The underlying numerical scheme is based upon a second-order finite difference discretization in space and a second-order, fully implicit, adaptive scheme in time. At each time step, this leads to the need to solve a large system of nonlinear algebraic equations, for which the full approximation storage multigrid algorithm is employed. The motion of the contact line between the three phases (liquid, air and the solid substrate) is based upon the assumption of a thin precursor film, with a corresponding disjoining pressure term in the governing equations. It is the inclusion of this precursor film in the model that motivates the need for a parallel solution method. This is because the thickness of such a film must be very small in order to yield realistic predictions, while the finite difference grid must be correspondingly fine in order to obtain accurate numerical solutions. Results are presented which demonstrate that the parallel implementation is sufficiently efficient and robust to allow reliable numerical solutions to be obtained for a level of mesh resolution that is an order of magnitude finer than is possible using a single processor. Copyright © 2008 John Wiley & Sons, Ltd.

Received 30 April 2007; Revised 23 November 2007; Accepted 4 December 2007

KEY WORDS: thin film approximation; multigrid; parallel algorithms

1. INTRODUCTION

This paper is concerned with the efficient and reliable parallel numerical simulation of a range of droplet spreading problems that may be accurately modelled using a long wave, or lubrication, approximation. Such problems are of great importance in many industrial applications, ranging

*Correspondence to: P. K. Jimack, School of Computing, University of Leeds, Leeds LS2 9JT, U.K.
†E-mail: pkj@comp.leeds.ac.uk

from the deposition of coatings and inks to direct patterning of functional layers during microchip production [1, 2]. The lubrication approximation has been widely used to simplify the Navier–Stokes equations in the situation where inertia effects and the effects of flow across a film may be reasonably neglected [3–6]. Furthermore, to alleviate the resulting singularity at wetting lines and to facilitate the spreading motion of the droplets, a popular approach has been to specify a thin energetically stable wetting layer, or precursor film, over the whole of the substrate surface for all time [5, 7].

In previous work by the authors [3], we have used this approach to model the spreading of droplets over a substrate with topographic and/or wetting heterogeneities. By applying a finite difference discretization in space and an adaptive, implicit scheme in time, combined with a nonlinear multigrid solver [8–11], it is possible to obtain highly efficient numerical solutions to a wide range of spreading problems. The primary limitation of this approach, however, stems from the need to define the thickness of the precursor film in advance. As this thickness is reduced, the accuracy of the simulations increases but only at the expense of requiring a finer spatial mesh resolution. In [3] and similar work by other authors [4, 5, 7], calculations have been undertaken using a larger precursor film thickness than is desired due to the prohibitive computational cost of working with thicknesses that are physically realistic. In this paper, we demonstrate that it is possible to reach such physically realistic limits, provided a sufficiently fine finite difference mesh is used, requiring the exploitation of parallel computer architectures.

The layout of the paper is as follows. In the following section, we briefly introduce the governing equations and re-iterate the underlying assumptions that these are based upon. Section 3 then outlines the numerical algorithms that are used and quantifies the argument for finer grids and more computational resources. The parallel solution procedure is then described and some sample parallel numerical results are provided in Section 4. Finally, the paper concludes with a short discussion.

2. THIN FILM FLOW AND DROPLET SPREADING

Figure 1 shows a schematic of the flow of a droplet on a flat substrate inclined at an angle α to the horizontal. It is assumed that the ratio, ε, of typical length scales, H_0 and L_0, in the perpendicular

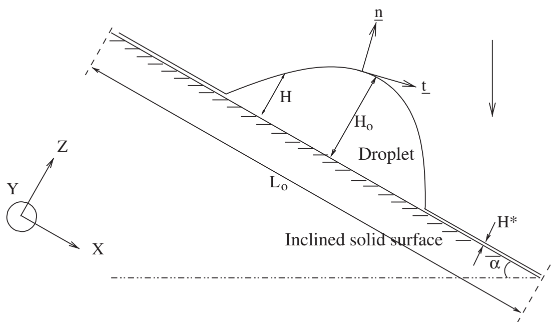

Figure 1. Schematic of the flow of a droplet on a flat inclined substrate.

and parallel directions, respectively, is small. It is also assumed that the ratio of the precursor film thickness, H^*, to the typical droplet height, H_0, is small. It is then possible to non-dimensionalize the Navier–Stokes equations and collect together the leading order terms in the small parameter ε in order to yield a simplified system of equations for this thin film flow. Further details are provided in [3, 6], for example, but the resulting partial differential equations take the following form:

$$\frac{\partial h}{\partial t}=\frac{\partial}{\partial x}\left[\frac{h^3}{3}\left(\frac{\partial p}{\partial x}-\frac{\mathrm{Bo}}{\varepsilon}\sin\alpha\right)\right]+\frac{\partial}{\partial y}\left[\frac{h^3}{3}\left(\frac{\partial p}{\partial y}\right)\right] \tag{1}$$

$$p=-\nabla^2(h+s)-\Pi(h)+\mathrm{Bo}\cos\alpha(h+s) \tag{2}$$

Here, h and p are the dependent variables (non-dimensional film height and non-dimensional pressure, respectively), $\mathrm{Bo}=\rho g L_0^2/\sigma$ is the dimensionless Bond number (which represents the ratio of gravitational to surface tension forces) and s is a known function describing surface roughness of the inclined plane. As has already been indicated, in this model we also include a so-called *disjoining pressure* term $\Pi(h)$ in Equation (2), which is used to model slip of the moving contact lines. This term is based upon the assumption of the presence a thin precursor film, of non-dimensional thickness $h^*=H^*/H_0$, which relates the observed contact angle for partially wetting systems to intermolecular forces that become important for liquids at sub-microscopic dimensions [5]. Following [3], here we express this disjoining pressure term in the following form:

$$\Pi(h)=\frac{(n-1)(m-1)}{h^*(n-m)}\sigma(1-\cos\theta_{\mathrm{c}})\left[\left(\frac{h^*}{h}\right)^n-\left(\frac{h^*}{h}\right)^m\right] \tag{3}$$

where θ_{c} is the equilibrium contact angle and n and m are the exponents of the interaction potential, chosen to be 3 and 2, respectively, for all calculations described in this paper.

Note that experimental evidence, such as that presented in [12], suggests that H^* lies in the broad range 1–100 nm. Moreover, when the non-dimensionalized film thickness corresponds to a larger value of H^*, the spreading rate is not generally predicted to a high accuracy [5]. Furthermore, if the mesh resolution is not of the same order of magnitude as h^*, then there is a danger of the numerical scheme yielding oscillatory results that may even predict negative film thickness near to the wetting line [13]. With care, it is possible to develop schemes that are guaranteed to be positivity preserving regardless of the mesh resolution [14]; however, they are still not accurate when the mesh spacing is not of the same order as h^* [6].

Finally, to accompany Equations (1) and (2), boundary conditions are required. So long as the droplet does not reach the edge of the domain it is acceptable, and simplest, to use symmetry boundary conditions for both h and p:

$$\frac{\partial h}{\partial n}=\frac{\partial p}{\partial n}=0 \tag{4}$$

3. NUMERICAL METHODS AND PARALLEL SOLUTION PROCEDURE

Application of central finite differences on a mesh with equal spacing, Δ, in both the x and y directions yields the following system of differential-algebraic equations for $h_{i,j}$ and $p_{i,j}$:

$$\begin{aligned}\frac{\partial h_{i,j}}{\partial t} = \frac{1}{\Delta^2}\Bigg[& \frac{h^3}{3}\bigg|_{i+1/2,j}(p_{i+1,j}-p_{i,j}) - \frac{h^3}{3}\bigg|_{i-1/2,j}(p_{i,j}-p_{i-1,j}) \\ & + \frac{h^3}{3}\bigg|_{i,j+1/2}(p_{i,j+1}-p_{i,j}) - \frac{h^3}{3}\bigg|_{i,j-1/2}(p_{i,j}-p_{i,j-1})\Bigg] \\ & - \frac{\mathrm{Bo}}{\varepsilon}\sin\alpha h_{i,j}^2\left(\frac{h_{i+1,j}-h_{i-1,j}}{2\Delta}\right)\end{aligned} \tag{5}$$

$$0 = p_{i,j} + \frac{1}{\Delta^2}[h_{i+1,j}+h_{i-1,j}+h_{i,j+1}+h_{i,j-1}-4h_{i,j}] + \Pi(h_{i,j}) - h_{i,j}\mathrm{Bo}\cos\alpha \tag{6}$$

As described in [3], these equations may be integrated in time using an implicit scheme. Here, we follow [3] and apply a second-order scheme with adaptive step-size selection based upon a local error estimate. At each time step, this therefore results in the need to solve a large system of nonlinear algebraic equations for $h_{i,j}^{n+1}$ and $p_{i,j}^{n+1}$, the approximations to $h_{i,j}$ and $p_{i,j}$, respectively, at the new time level t^{n+1}.

In this work, we make use of a full approximation storage (FAS) multigrid scheme [8, 9] to solve the nonlinear systems that arise. This implementation is described in general in [10], for example, and for these specific equations in [3] (with full details available in [6]). The same solution algorithm is used at all grid levels, apart from at the coarsest mesh, where an exact solve is undertaken. To demonstrate the need for a parallel implementation of this solver, consider Figure 2 that shows mesh convergence results for two different choices of h^* on the square domain $(0,1)\times(0,1)$ with $\alpha=0$. It is clear from these figures that the spreading rate is still dependent upon h^* for values of this magnitude and that the mesh required to obtain a converged solution gets finer as h^* is reduced. Specifically, when $h^*=0.05$ the solution is fully converged on a 513×513 mesh, whereas for $h^*=0.02$ the solution is not fully converged on such a mesh and a 1025×1025 mesh is required.

The parallel implementation undertaken here follows [15] in its basic philosophy. In particular, the parallelism is achieved via a geometric decomposition of the domain, which is based upon a partition of the coarsest grid. This ensures that each of the grids in the multigrid hierarchy is partitioned in the same geometric manner and, by assigning each subdomain to a single parallel processor, interpolation and restriction between grids may be undertaken with minimal inter-processor communication. Such a decomposition is illustrated for 15 subdomains in Figure 3, where film thickness contours are given for each subdomain. Note that there is a slight mismatch in contour levels between neighbouring subdomains since, for simplicity, in each subdomain an equal number of contour levels are generated automatically.

Note that the strip-wise partition illustrated in Figure 3 ensures that each subdomain (and therefore each processor) has at most two immediate neighbours. Each processor is responsible for implementing the FAS algorithm on its own subdomain, making use of additional columns of

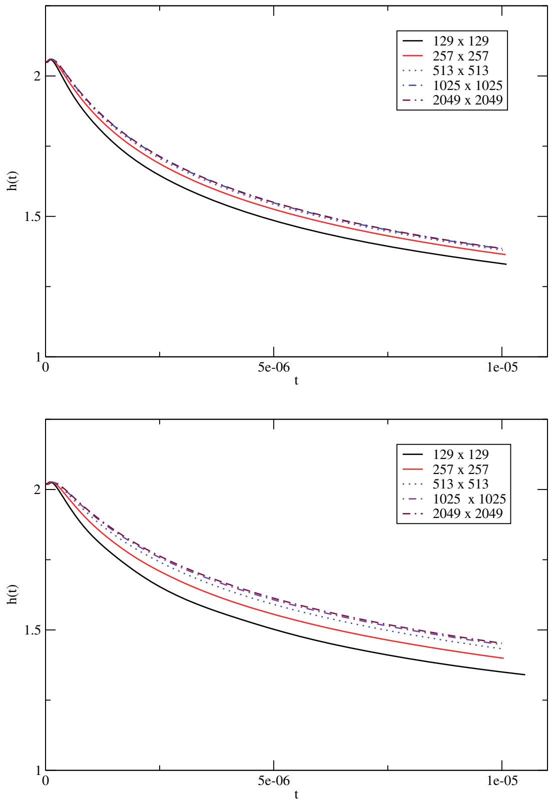

Figure 2. Plot of the maximum film height against time computed on a sequence of finer meshes for two different choices of h^*: 0.05 (upper) and 0.02 (lower).

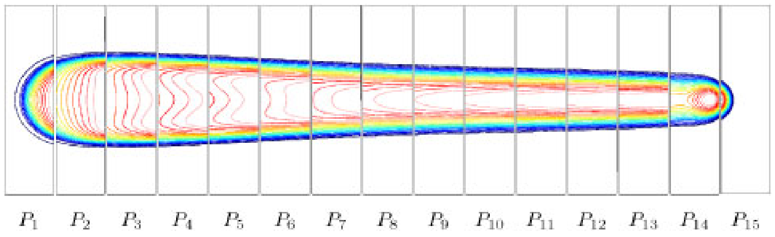

Figure 3. Film thickness contour plots on 15 processors for a typical droplet spreading problem [7].

'ghost nodes' immediately to either side of this subdomain at each mesh level. These ghost nodes are used to store a copy of the last column of values computed on each neighbouring processor on each mesh. An illustration of this for a single mesh level is provided in Figure 4. This shows a partition into four subdomains and the dotted line indicates the part of the grid that is stored on a typical processor: note that the first and last columns only store a copy of the values actually computed on the left and right neighbouring processors, respectively.

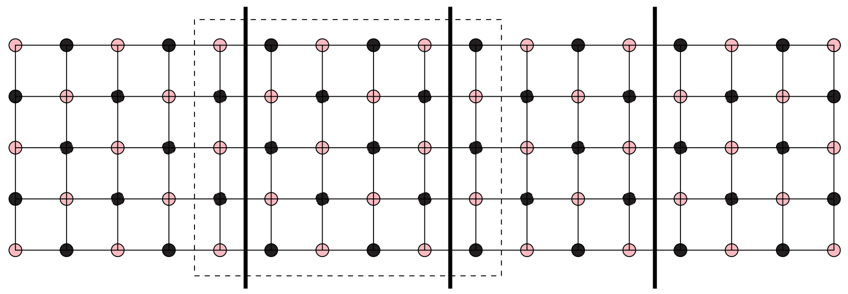

Figure 4. Partition of a typical coarse grid across four processors: the dashed line indicates the grid stored on a typical processor and the shading illustrates the red–black colouring that is used.

For the parallel FAS implementation, the smoother that is used is the same nonlinear red–black Gauss–Seidel scheme that is shown to be effective in [3]. This scheme is ideal for parallel implementation since it labels each of the grid points on each processor as either *red* or *black* in such a way that each red point has only black neighbours and *vice versa*. This is illustrated in Figure 4 and ensures that all of the red degrees of freedom may be updated concurrently, followed by all of the black degrees of freedom. Some neighbour-to-neighbour inter-processor communication is required after each red and each black sweep through the grid in order to update the corresponding ghost node values. In fact, such inter-processor communication is required at a number of points in the parallel FAS implementation:

- after each red and each black sweep of the nonlinear red–black Gauss–Seidel smoother at each grid level;
- after restriction of the residual and solution to each coarser level;
- to obtain the exact solution at the coarsest level;
- after interpolation of the error to each finer level.

In addition to the above neighbour-to-neighbour communication, a short global communication is also required after each multigrid V-cycle in order to decide whether convergence has been achieved. Depending upon the choice of solver on the coarsest grid, a similar global communication may be required for each solve on this grid too.

Note that, for the implementation used in this work, the size of the coarsest possible grid is determined by the number of processors being used. This is because it is a helpful simplification to always assume that at least one column of the coarsest grid is owned by each processor. Furthermore, in situations where the number of columns of the coarsest grid is not an exact multiple of the processors (such as in Figure 4, where there are 17 columns and 4 processors), then the additional columns are shared equally between the lowest numbered processors. Hence, for the example in Figure 4, the first processor owns one more column than each of the other processors.

4. SAMPLE RESULTS

In this section, we present some typical results that have been obtained using the parallel lubrication solver that has been described. These results fall into two classes: initially we assess the parallel performance of the solver and then we assess how well we have obtained our goal of reducing the precursor film thickness to a size that is consistent with experimental evidence, such as provided by Starov *et al.* [12].

Figure 5 shows graphs of the speedup of the parallel solver when run on (i) between 2 and 32 processors for a grid of size 1025×1025 and (ii) between 4 and 32 processors for a grid of size 2049×2049. In each case, results are plotted for three different parallel computers (denoted *ABAX*, *SNOWDON* and *EVEREST*) and are presented relative to the smallest number of processors (i.e. time on p processors divided by time on 2 processors in the first case and divided by time on 4 processors in the second case). All three computers are distributed memory architectures with fast switching. As can be seen, the parallel performance is relatively good in each case with, as one would expect, better speedups obtained for the finer mesh problem.

Given the similarities in the scalability of the solver on all three computers, further results will only be presented for one of the architectures. This machine (EVEREST) has multiple nodes, each with 8 Gb of memory and 2 dual core processors, although the maximum queue size is restricted to 128 cores. Table I shows typical execution times for five time steps taken on this computer using a 4097×4097 fine grid, and Table II shows equivalent results using a 8193×8193 fine grid.

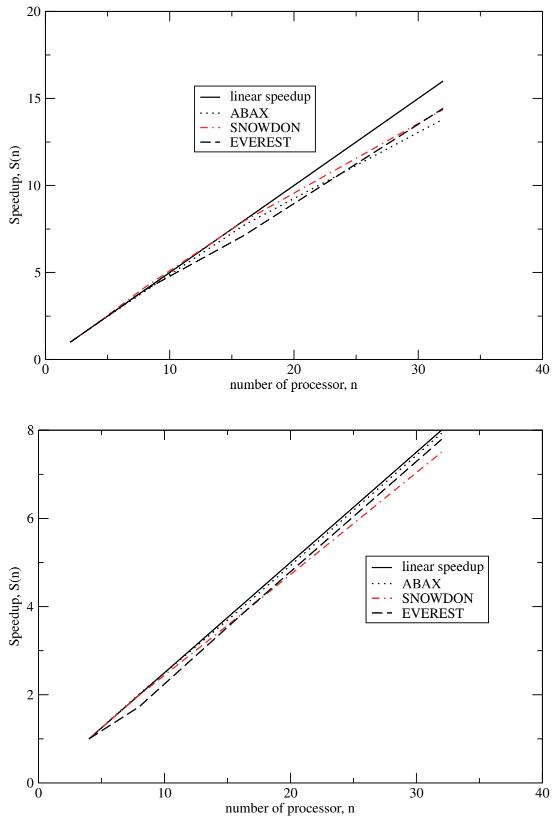

Figure 5. Relative speedup for the parallel multigrid solution on three different parallel architectures with mesh sizes 1025×1025 (upper) and 2049×2049 (lower): in the first case a minimum of 2 processors is used and in the second case a minimum of 4.

Table I. Execution time for 5 time steps for different coarsest grids and different numbers of processors on a finest mesh size of 4097×4097.

Multigrid levels	Coarse grid	Number of processors			
		16	32	64	128
4	513	215.5	106.9	53.1	24.2
5	257	143.7	69.9	34.5	15.7
6	129	150.1	73.1	35.8	18.6

Table II. Execution time for 5 time steps for different coarsest grids and different numbers of processors on a finest mesh size of 8193×8193.

Multigrid levels	Coarse grid	Number of processors		
		32	64	128
5	513	250.7	154.9	73.6
6	257	247.5	151.0	74.2
7	129	256.0	156.4	78.0

Table III. Execution time for 5 time steps for different finest grids and different numbers of processors (the brackets indicate the number of multigrid levels used).

Fine grid	Number of processors						
	2	4	8	16	32	64	128
1025	56.9 (4)	$\underline{28.5}$ (4)	$\underline{\underline{13.9}}$ (5)				
2049			67.3 (5)	$\underline{30.8}$ (5)	$\underline{\underline{15.0}}$ (5)		
4097					69.9 (5)	$\underline{34.5}$ (5)	$\underline{\underline{15.7}}$ (5)
8193							73.6 (5)

In each case, different numbers of multigrid levels are considered, corresponding to different choices of the coarsest grid.

It is clear from Tables I and II that the parallel speedup can continue to increase with the number of processors, provided the size of the problem is sufficiently large. Indeed, super-linear speedups even appear to occur as we move from 64 for 128 processors; however, this is primarily due to the efficiency being worse on 64 than on 128 processors for some reason (note that the time on 128 processors is still more than a quarter of the time on 32 processors in almost every case). It is also apparent from these tables that the precise choice made for the coarsest grid also has a bearing on the overall multigrid solution time and, therefore, on the parallel efficiency. In Table III we, therefore, present results for a variety of fine mesh sizes and process numbers, using the best possible choice for the coarsest grid in each case.

Note that the results presented in Table III correspond to calculations with three different problem sizes per processor. The largest problems (for which each processor stores a total of $\sim 524\,000$ grid points) are denoted by timings which are not underlined. The next sequence of problems (for which each processor stores a total of ∼262 000 grid points) have timings with one underline, and the smallest problems (for which each processor stores a total of ∼131 000 grid points) are denoted by timings that are twice underlined. Given that the computational complexity of the underlying sequential algorithm is shown in [3] to be $O(N)$, where N is the total number of unknowns, it is clear that a perfectly scalable algorithm should yield the same time for all runs with a fixed number of points per processor. Furthermore, as the number of points per processor doubles, the solution time should double. What we actually see in Table III is remarkably close to this, suggesting that we have a very good parallel implementation. The small loss of efficiency that is observed may be attributed to the overhead of inter-processor communication. This overhead will be most apparent on the coarsest levels of the multigrid hierarchy, where the amount of computation per processor is relatively small.

Having established the efficiency of the parallel implementation, we now present some results that make use of this parallel code in order to reduce the size of the precursor film thickness significantly from that which is possible in a sequential implementation. Figure 2 clearly shows that the converged results obtained using a precursor film thickness of 0.05 are significantly different from those obtained when $h^*=0.02$. In Figure 6, we now present results computed for

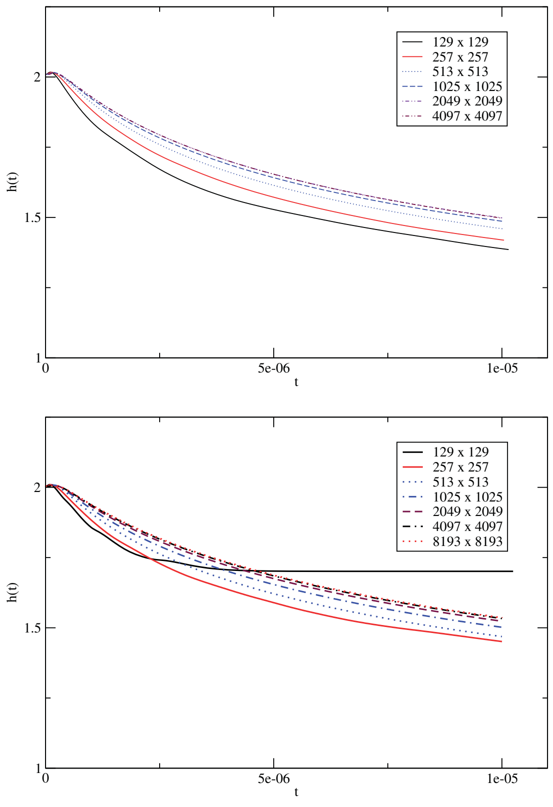

Figure 6. Plot of the maximum film height against time computed on a sequence of finer meshes for two different choices of h^*: 0.01 (upper) and 0.005 (lower).

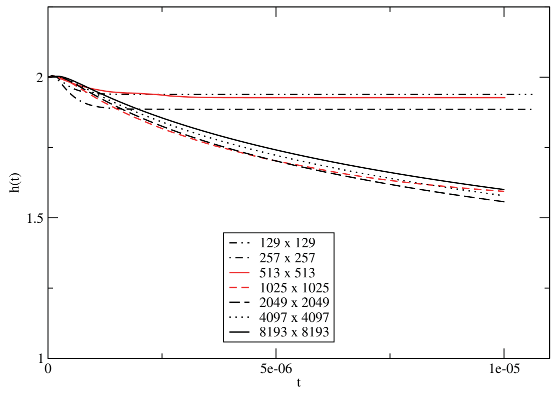

Figure 7. Plot of the maximum film height against time computed on a sequence of finer meshes for $h^* = 0.001$.

$h^* = 0.01$ and 0.005 on a sequence of grids of up to 8193×8193. It may be observed that in the former case ($h^* = 0.01$) a fully converged solution is obtained on the 2049×2049 grid, while the 4097×4097 grid is required for convergence when $h^* = 0.005$. By simple extrapolation, we may reasonably assume that the 8193×8193 grid will be sufficient to compute a converged solution when $h^* = 0.0025$; however, it would only be possible to confirm this using an even finer grid (and showing that the solution is unchanged) which would require more than 128 processors.

Given the non-dimensionalization that was used to derive Equations (1) and (2), it is possible to associate a given non-dimensional film thickness h^* with a dimensional value for any given problem. For the parameters used to obtain the results shown in Figure 6, $h^* = 0.005$ corresponds to a dimensional precursor film thickness of 65 nm (see [16]), which is just within the range of 1–100 nm suggested by Starov *et al.* [12]. We finish this section by showing results for an even smaller value of h^*, 0.001, corresponding to a dimensional precursor film thickness of 13 nm. The maximum film thickness against time is plotted for a variety of finest meshes in Figure 7. It is clear from this figure that, as expected, the 8193×8193 mesh is still not sufficiently fine to obtain a fully converged solution. A finer grid and more processors are therefore required. As a final observation, we note that when h^* is very small, if the mesh spacing is much greater than h^*, then the computed results are extremely poor (e.g. a mesh of dimension at least 1025×1025 is required to get even qualitatively realistic results when $h^* = 0.001$).

5. DISCUSSION

This work has been motivated by the need to use very fine spatial resolution when solving the lubrication equations for a spreading droplet based upon the assumption of a precursor film. This model can provide accurate numerical predictions, provided the precursor film thickness is sufficiently small and the computational grid is correspondingly fine. It has been demonstrated that the use of parallel processing allows much finer meshes to be used than is otherwise feasible and at a parallel efficiency that is both scalable and cost effective. As a result, it has been possible to simulate spreading droplets using a precursor film thickness that is physically justified for the first time.

Current work now focuses on applying the numerical methods and software that we have developed here to a range of challenging flow problems. Future developments could include application of adaptive meshing techniques to ensure that the finest multigrid levels are only present in the regions in which they are required, e.g. [17], and the incorporation of this approach within the parallel computing framework, as in [18], for example.

REFERENCES

1. De Gennes PG. Wetting: statics and dynamics. *Reviews of Modern Physics* 1985; **57**:827.
2. Peurring LM, Graves DB. Spin coating over topography. *IEEE Transactions on Semiconductor Manufacturing* 1993; **6**:72–76.
3. Gaskell PH, Jimack PK, Sellier M, Thompson HM. Efficient and accurate time-adaptive multigrid simulations of droplet spreading. *International Journal for Numerical Methods in Fluids* 2004; **45**:1161–1186.
4. Oron A, Davis SH, Bankoff SG. Long-scale evolution of thin liquid films. *Reviews of Modern Physics* 1997; **69**:931–980.
5. Schwartz LW. Hysteretic effects in droplet motion on heterogeneous substrates: direct numerical simulation. *Langmuir* 1998; **14**:3440–3453.
6. Sellier M. The numerical simulation of thin film flow over heterogeneous substrates. *Ph.D. Thesis*, University of Leeds, U.K., 2006.
7. Schwartz LW, Eley RR. Simulation of droplet motion on low-energy and heterogeneous surfaces. *Journal of Colloid and Interface Science* 1998; **202**:173–188.
8. Brandt A. Multi-level adaptive solutions to boundary-value problems. *Mathematics of Computation* 1977; **31**: 333–390.
9. Brandt A. Guide to multigrid development. In *Multigrid Methods*, Hackbusch W, Trottenberg U (eds). Lecture Notes in Mathematics, vol. 960. Springer: Berlin, 1982; 220–312.
10. Trottenberg U, Oosterlee CW, Schüller A. *Multigrid*. Academic Press: New York, 2000.
11. Wesseling P. *Introduction to Multigrid Methods*. Wiley: New York, 1992.
12. Starov VM, Kalinin VV, Chen JD. Spreading of liquid drops over dry surfaces. *Advances in Colloid and Interface Science* 1994; **50**:187–221.
13. Bertozzi A. The mathematics of moving contact lines in this liquid films. *Notices of the AMS* 1998; **45**:689–697.
14. Zhornitskaya L, Bertozzi A. Positivity-preserving numerical schemes for lubrication-type equations. *SIAM Journal on Numerical Analysis* 2000; **37**:523–555.
15. Goodyer CE, Berzins M. Parallelisation and scalability issues of a multilevel elastohydrodynamic lubrication solver. *Concurrency and Computation*: *Practice & Experience* 2007; **19**:369–396.
16. Koh Y-Y. Efficient numerical solutions of droplet spreading flows. *Ph.D. Thesis*, University of Leeds, U.K., 2007.
17. Rosam J, Jimack PK, Mullis AM. A fully implicit, fully adaptive time and space discretisation method for phase-field simulation of binary alloy solidification. *Journal of Computational Physics* 2007; **225**:1271–1285.
18. Bank RE, Jimack PK. A new parallel domain decomposition method for the adaptive finite element solution of elliptic partial differential equations. *Concurrency and Computation*: *Practice & Experience* 2001; **13**:327–350.