PHILOSOPHICAL TRANSACTIONS OF THE ROYAL SOCIETY A

FirstCite® e-publishing

Phil. Trans. R. Soc. A
doi:10.1098/rsta.2005.1623
Published online

Hybrid molecular-continuum fluid models: implementation within a general coupling framework

By Rafael Delgado-Buscalioni[1], Peter V. Coveney[1], Graham D. Riley[2] and Rupert W. Ford[2]

[1] *Centre for Computational Science, Department of Chemistry, University College London, 20 Gordon Street, London WC1 OAJ, UK*
(p.v.coveney@ucl.ac.uk)
[2] *Centre for Novel Computing, School of Computer Science, The University of Manchester, Manchester M13 9PL, UK*

Over the past three years we have been developing a new approach for the modelling and simulation of complex fluids. This approach is based on a multiscale hybrid scheme, in which two or more contiguous subdomains are dynamically coupled together. One subdomain is described by molecular dynamics while the other is described by continuum fluid dynamics; such coupled models are of considerable importance for the study of fluid dynamics problems in which only a restricted aspect requires a fully molecular representation. Our model is representative of the generic set of *coupled models* whose algorithmic structure presents interesting opportunities for deployment on a range of architectures including computational grids. Here we describe the implementation of our HybridMD code within a coupling framework that facilitates flexible deployment on such architectures.

Keywords: molecular dynamics; continuum fluid dynamics; coupled models

1. Introduction

Many scientific problems have an intrinsically multiscale nature; that is, they involve processes occurring on more than one length and/or timescale. Examples occur in almost every branch of science, engineering and beyond. In computational chemistry and biology, one has to address the competition between the very fast time scale of quantum processes and the much slower ones associated for example with protein folding and relaxation in glassy materials. Systems biology (Ideker *et al.* 2001) is concerned with the vertical integration of data and models spanning a huge range of length and time scales, from the electronic/chemical through to the whole organ, organism and societies themselves. In fracture mechanics, one has to deal with everything from the breaking of chemical bonds at a fracture tip to the propagation of

One contribution of 27 to a Theme 'Scientific grid computing'.

fractures described by finite element methods, through classical molecular dynamics (MD).

The widespread existence of systems of this kind, in which processes occurring on several different length and timescales are involved, demands common computational approaches. From the computational point of view, one can conceive of two broadly different methodologies to address these problems. In the first, or hierarchical approach, ‘effective theories’ can be used to bridge the gap between different scales: one can arrange for suitable matching of parameters that enter the description of phenomena at different levels. In the second methodology, which is the one adopted in this work, hybrid or coupled schemes must be constructed when the physics, chemistry or biology is dynamically coupled across the length and time scales involved. Examples of this occur in situations where one needs to pay attention to molecular phenomena in a limited domain of interest, while treating the system in a more coarse-grained manner at further distances. Hydrodynamic effects can influence events on the molecular scale, so both representations must be coupled within the same simulation environment.

Just as computational science research is characterized by common underlying problems, so computational methods and algorithms transcend disciplinary divisions. In this paper, we show how a specific coupled fluid dynamics model can be implemented within a flexible coupling framework that is designed to permit its deployment on a range of architectures, including computational grids, while keeping the generic features of the coupling scheme in mind at all times.

Over the past 3 years, within the Centre for Computational Science at UCL we have been developing a new approach for the modelling and simulation of complex fluid flow for application to a wide range of problems in soft condensed matter science and engineering. This approach is based on a multiscale hybrid scheme, in which two or more contiguous subdomains are dynamically coupled together; the simplest case corresponding to one subdomain being described by MD while the other by continuum fluid dynamics. Such coupled models are of considerable importance for the study of fluid dynamics problems in which only a restricted region needs to be treated by MD, which is computationally extremely expensive relative to the continuum description. Here we describe the implementation of our HybridMD code within a coupling framework that facilitates flexible deployment on such disparate architectures. This framework has been developed at the Centre for Novel Computing at the University of Manchester and is called the general coupling framework (GCF). Our collaboration has been facilitated through the EPSRC RealityGrid e-Science Pilot Project (www.realitygrid.org).

The scientific motivation for this hybrid approach to fluids is that it enables us to tackle problems that are not addressable by other techniques; such problems include situations in which fluid flow and hydrodynamic effects influence molecular interactions, dynamics and structure at interfaces, lipid bilayer membranes, and within individual macromolecules or assemblies of them. Since the computational cost of coupling the molecular and continuum regions is very low, we are able to perform such coupled simulations using small MD domains which are free of finite size effects, concomitantly reducing the CPU time compared with fully atomistic simulations.

2. The Hybrid particle-continuum scheme

(*a*) *Overview and theoretical formalism*

A typical spatial domain decomposition structure for our hybrid scheme is depicted in figure 1. The particle region (P) contains $N(t)$ particles at time t and it is described using standard MD. Each particle i, has a mass m_i, a velocity $\boldsymbol{v}_i$ and a potential energy due to interparticle interactions $\psi_i(\{\boldsymbol{r}\})$, where $\{\boldsymbol{r}\}$ denotes the ensemble of particle positions. The force acting on each particle is given by $\boldsymbol{f}_i^{\mathrm{T}} = \boldsymbol{f}_i^{\mathrm{ext}} + \boldsymbol{f}_i$, where $\boldsymbol{f}_i = -\nabla\psi_i$ is due to inter-particle interactions and $\boldsymbol{f}_i^{\mathrm{ext}}$ is the external force released by the continuum (see below). The equations of motion for the particles, $\dot{\boldsymbol{r}}_i = \boldsymbol{v}_i$ and $\dot{\boldsymbol{v}}_i = \boldsymbol{f}_i^{\mathrm{T}}$, are solved by standard MD using a time step $\Delta t_{\mathrm{P}} \simeq 1$ fs, or $\Delta t_{\mathrm{P}} \simeq 0.005\tau = (m\sigma^2/\epsilon)^{1/2}$ in characteristic particle units, where σ is the particle radius, ϵ the typical energy of van der Waals interactions, and m the mass of the lighter particles. As an example, for N2 (nitrogen) $\sigma \simeq 0.35$ nm and $\epsilon/k_{\mathrm{B}} \simeq 100$ K), where k_{B} is the Boltzmann constant.

The rest of the computational domain (C) is described by the Navier–Stokes equations. The fluid variables at C are the densities of the conserved quantities for which the equations of motion in conservative form are $\partial\boldsymbol{\Phi}/\partial t = -\nabla\cdot\boldsymbol{J}_{\Phi}$ with $\boldsymbol{\Phi} = \{\rho, \rho\boldsymbol{u}, \rho e\}$ and $\boldsymbol{J}_{\Phi} = \{\rho\boldsymbol{u}, \rho\boldsymbol{u} + \Pi, \rho\boldsymbol{u}e + \Pi\cdot\boldsymbol{u} + \boldsymbol{Q}\}$ standing for the mass, momentum and energy fluxes, respectively. Here ρ is the density, $\boldsymbol{u}$ the local velocity, e the specific energy, $\Pi = P\mathbf{1} + \mathsf{T}$ the stress tensor which contains the pressure P and the viscous tensor T (for a Newtonian fluid) and $\boldsymbol{Q} = -k\nabla\cdot T$, the heat flux due to conduction, expressed via Fourier's law. These continuum equations are solved via standard CFD methods. Algorithms based on the finite volume method are the most suitable for the present flux-exchange hybrid scheme because this method is based on flux conservation (Patankar 1980). We note that the time step of the continuum solver Δt_{C} is about 100 times larger than that of the particle system, Δt_{P}. We set $\Delta t_{\mathrm{C}} = n\Delta t_{\mathrm{P}}$, where $n \sim 100$ has to be a natural number to ensure the synchronization of both solvers (P and C). This means that the C code is called each n P-time steps (see figure 2*a*).

(i) C$\rightarrow$P

The coupling scheme is a two-fold communication between C and P, as can be understood from figure 1. Within the C$\rightarrow$P domain, the fluxes of conserved quantities evaluated from C are imposed on the particle dynamics. Each C$\rightarrow$P buffer has a volume ΔV_{CP} around the interface position $x = x_{\mathrm{CP}}$ (see figure 1*a*). We refer to Delgado-Buscalioni & Coveney 2003*b* for a detailed discussion of the C$\rightarrow$P coupling, in the general case of mass, momentum and energy transfer in unsteady flows. An improved version of the heat transfer scheme has been recently proposed by Flekkøy *et al.* (in press). In what follows we briefly sketch the C$\rightarrow$P protocol in order to clarify what are the coupling variables required in the exchange.

The *momentum flux* across $x = x_{\mathrm{CP}}$ is given by $(P_{\mathrm{CP}} + \mathsf{T}_{\mathrm{CP}})\cdot\boldsymbol{n}_{\mathrm{CP}}$, where the surface vector $\boldsymbol{n}_{\mathrm{CP}} = -\boldsymbol{n}$ points towards P. This flux is introduced within the particle dynamics by adding an external force $\boldsymbol{f}_i^{\mathrm{ext}} = A(P_{\mathrm{CP}} + \mathsf{T}_{\mathrm{CP}})\cdot\boldsymbol{n}_{\mathrm{CP}}/N_{\mathrm{CP}}$ to the $N_{\mathrm{CP}}(t)$ particles within ΔV_{CP} (if $\boldsymbol{r}_i \notin \Delta V_{\mathrm{CP}}$ then $\boldsymbol{f}_i^{\mathrm{ext}} = 0$). In order to maintain a desired density ρ_{c} within the particle system, the hydrostatic pressure

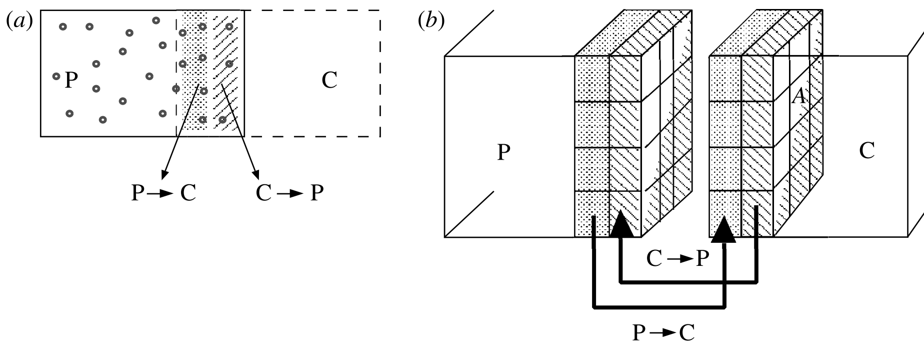

Figure 1. (*a*) A schematic domain decomposition of the hybrid scheme showing the particle domain (P) (particles included), the continuum domain (C) and the overlapping region (shaded) comprised of C→P and P→C. In (*b*) both domains are separated to show the two-dimensional array of C→P and P→C cells where the exchange of microscopic and macroscopic information is carried out. The surface area of each cell is *A*.

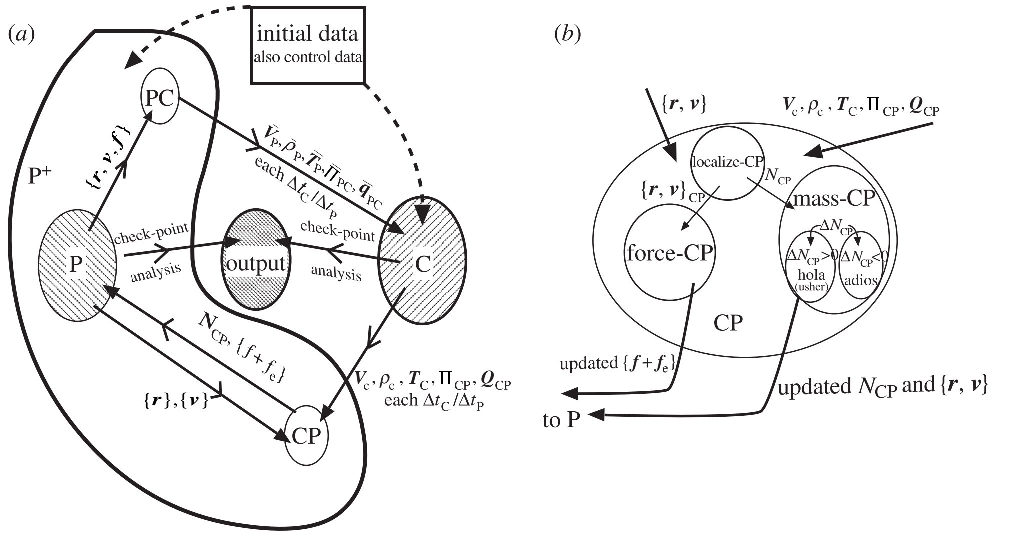

Figure 2. (*a*) Overview of the models and major coupling (input/output) data exchanges of the hybrid scheme and (*b*) a detail of the inner parts of the CP model. The square representing the initial data input consists of two data files (one for each of P and C). These files are generated by an initialization routine which provides to each model all the common variables (system geometry, initial state, check-point time, etc.) in the physical units of that model.

needs to be given by the correct equation of state, $P_{\mathrm{CP}} = P(\rho_{\mathrm{c}}, T_{\mathrm{c}})$. In summary, the momentum flux requires specification of the pressure tensor Π from C.

The *heat flux* through the C→P interface can be implemented in the particle dynamics using a number of strategies. A crude way is to use a series of thermostats across C→P (Delgado-Buscalioni & Coveney 2003*b*). A better alternative is to insert the heat flux via the dissipation associated with the imposed external force (Flekkøy *et al.* in press). In both cases the C domain need only specify the heat flux $\boldsymbol{Q}$.

The *mass flux* can be also specified in several ways. The relevant equation has to provide the rate of change of the number of particles at the CP interface: $\Delta N_{\mathrm{CP}}/\Delta t$. We assume that

$$\frac{\Delta N_{\mathrm{CP}}}{\Delta t} = -\frac{1}{\tau_{\mathrm{r}}}(N_{\mathrm{CP}}(t) - M), \tag{2.1}$$

where M is a fixed number and τ_{r} is a relaxation time, at least of order of the sound time across the C→P buffer. In equation (2.1) one can fix the volume of the C→P region, V_{CP}, and set $M/V_{\mathrm{CP}} \leq \rho_{\mathrm{c}}$ as in (Barsky *et al.* 2003), or fix $N_{\mathrm{CP}} = M$ to any (large enough) desired value and leave V_{CP} undetermined (fluctuate) (Flekkøy *et al.* in press). This second choice corresponds to a vanishingly small relaxing time $\tau_{\mathrm{r}} = \infty$. In whatever choice we should provide the value of M to fix the mass flux.

(ii) P→C

At the P→C interface, the flux measured in the particle system must be imposed onto C as a flux boundary condition. The particle flux transferred to C has to be averaged over the local time and length scales of the continuum level. The mean value of any particle quantity ϕ_i within a volume ΔV of a cell centred at position $\boldsymbol{R}$ is $\phi_{\mathrm{R}}(t) = \sum_i^{N_{\mathrm{R}}} \phi_i(t)/N_{\mathrm{R}}$, where the sum is made over the N_{R} particles within ΔV. The coarse-grained quantity is defined by time-averaging over Δt_{av}, i.e. $\bar{\phi}_{\mathrm{R}}(t) \equiv \int_0^{\Delta t_{\mathrm{av}}} \phi_{\mathrm{R}}(t-\xi)\mathrm{d}\xi/\Delta t_{\mathrm{av}}$.

In a general case of mass, momentum and heat exchange within a single component fluid, the boundary condition for the C domain requires the following averaged fluxes across the P→C interface: momentum flux: $\bar{\boldsymbol{j}}_{\mathrm{PC}} \cdot \boldsymbol{n}$ and heat flux $\bar{\boldsymbol{q}}_{\mathrm{PC}} \cdot \boldsymbol{n}$. Expressions for these fluxes in terms of particle variables can be found in Delgado-Buscalioni & Coveney (2003*b*); for example the momentum flux is the average of $\phi_i = \Delta V_{\mathrm{PC}}^{-1}[m\boldsymbol{v}_i\boldsymbol{v}_i - (1/2)\sum_j^N \boldsymbol{r}_{ij}\boldsymbol{f}_{ij}] \cdot \boldsymbol{n}_{\mathrm{PC}}$. On the other hand, as explained by Delgado-Buscalioni *et al.* (2005) and Delgado-Buscalioni & Coveney (2003*a*), in order to ensure variable continuity across the P→C interface, one need to specify the average particle velocity at the overlapping cell $\bar{\boldsymbol{v}}_{\mathrm{P}}$, the average density $\bar{\rho}_{\mathrm{P}}$ and temperature $\bar{T}_{\mathrm{P}}$. The mass flux across the P→C interface can be obtained from $\rho_{\mathrm{P}}\bar{\boldsymbol{v}}_{\mathrm{P}}$.

(*b*) *Computational implementation: HybridMD*

The coupling (input/output) connections of the hybrid model are depicted in figure 2. 'Blobs' with thinner solid lines indicate the different models involved in the coupling endeavour: PC is in charge of the P→C coupling and the CP model implements the C→P counterpart. The hybrid model requires two traditionally independent models to solve standard MD and CFD problems. Also 'models' to deal with the analysis, visualization, etc. may be included. These independent models are the shaded blobs in figure 2 and are denoted by P, C and output, respectively. The hybrid scheme is the set of algorithms that enables the communication between C and P, and it is comprised of the PC and CP blobs in figure 2.

In the composition proposed in figure 2, P^+ denotes the *composite* model P+PC+CP. This sort of composite is naturally suited to the transformations carried out by the hybrid scheme within the particle domain, which requires information about the whole particle system in order to count the number of particles at the buffer domains, insert or extract particles, add external forces to particles at the C$\rightarrow$P buffer and evaluate average fluxes at the P$\rightarrow$C buffer. In numerical implementations it would quite inefficient to transfer the whole set of particle positions $\{\boldsymbol{r}\}$, velocities $\{\boldsymbol{v}\}$, and forces $\{\boldsymbol{f}\}$, each time P communicates with PC or CP. In figure 2, the 'blobs' with thick solid lines indicate the implemented model structure.

In order to incorporate PC and CP into the P^+ composite some minor changes and additions need to be made in the original MD code, which must therefore to be open source. In particular, a hybrid-enabled MD code should dynamically vary the number of particles N and add external forces to particles within the C$\rightarrow$P buffers. While the second condition is usually straightforward to implement, the first one is more involved, particularly for parallel codes.

While the transformations performed within the PC model are standard spatio-temporal averages, those carried out in the CP model are far from trivial. Indeed, the C$\rightarrow$P coupling is the most complex part of the hybrid scheme and this is reflected in the implementation. The CP model requires the whole set of particle positions and velocities $\{\boldsymbol{r}, \boldsymbol{f}\}$ from P, and each $n = \Delta t_\mathrm{C}/\Delta t_\mathrm{P}$ P-time steps, it updates the local continuum variables and fluxes within the C$\rightarrow$P buffer. The following operations are then performed in order:

(i) *Localize*-CP. Localize the particles within the C$\rightarrow$P buffers[1]:
(ii) *Count* these N_CP particles and evaluate the mass change ΔN_CP in the present P-time step, according to equation (2.1). Here one can use the strategy explained by Delgado-Buscalioni & Coveney (2003*b*) to ensure that particle insertions are equidistributed in time.
(iii) *Mass*-CP. Insertion or extraction steps: if $\Delta N_\mathrm{CP} > 0$ the insertion subroutine Hola is called (figure 2). In our model this subroutine contains the Usher algorithm for particle insertion: see Delgado-Buscalioni & Coveney (2003*c*). If $\Delta N_\mathrm{CP} < 0$ particles are extracted by the Adios subroutine, which starts to delete those particles nearest to the end of the P region. After insertion or extraction ($N_\mathrm{CP} \rightarrow N_\mathrm{CP} + \Delta N_\mathrm{CP}$) the neighbour list of the whole MD domain is updated.
(iv) *Force*-CP. Insertion of external forces from C to the particles within the C$\rightarrow$P buffers. This requires the C$\rightarrow$P neighbour list, N_PC and the momentum and the local heat fluxes: Π_CP and $\boldsymbol{Q}_\mathrm{CP}$.

These transformations are independent and, as illustrated in figure 2*b*, can be implemented in different submodels denoted mass-CP and force-CP, respectively.

In summary, the hybrid protocol for coupling MD and CFD codes involves a relatively small amount of data exchanged between models. It is important to stress that the computational cost of the coupling protocol itself is negligible. This fact has been shown in very different scenarios involving modelling of water shells (De Fabritiis *et al.* 2004), polymer in an explicit Lennard–Jones (LJ)

[1] This operation can be optimized by maintaining a neighbour list (such as Verlet) for particles within the overlapping region.

solvent (Barsky *et al.* 2003), and unsteady oscillatory flow of pure LJ fluid (Delgado-Buscalioni & Coveney 2003*a*). In all these cases, the computational cost of coupling was smaller than 5% of the simulation total. From a computational point of view, these two properties (small size data-exchange and low coupling cost) mean that the hybrid MD/CFD model can be efficiently implemented on a Grid.

(*c*) *Composition and deployment requirements of HybridMD*

The HybridMD code was initially developed as a *bespoke* software engineering solution in which each model runs in turn within a single executable on a single processor workstation. In the original implementation, communication of coupling data between the two models is implemented by passing data through argument lists, and within P^+ use is made of shared data (e.g. common blocks) to minimize memory requirements.

An important user requirement is that the *composition* of the hybrid scheme has to be *flexible* enough to couple different versions of MD and CFD models. The optimal choice of the MD and CFD models depends on the balance between *simulation fidelity* and *performance* required. Furthermore, the resulting HybridMD model should be executable on a variety of computational resources, reflecting the requirement for flexibility in the *deployment*.

For instance, for small problems (or for test purposes) one may run the coupled model as a single executable on a desktop workstation or laptop. However, larger problems require different strategies depending on the physical phenomena under consideration. We illustrate this point with two research examples: (i) the effect of complex boundary conditions on fluid flow and (ii) the study of one (or a few) macromolecules in solution. The first case encompasses many different surface phenomena (such as surface growth by adsorption, the effect of tethered polymers on drag reduction near walls, boundary conditions for microfluidics and so on) on which the bulk fluid flow is an important part of the problem, requiring significant computational effort. By contrast, for the second case (an archetypal problem being protein folding) our interest and computational effort resides almost exclusively within the MD region, while the continuum is solved by a simple CFD approach providing a 'hydrodynamic bath' around the MD domain. While case (i) requires the exploitation of concurrent execution of individual models (i.e. P^+ and C), in case (ii) the majority of the computational resources should be applied in the MD domain as the C domain demands are relatively small.

3. Coupling HybridMD using the Bespoke Framework Generator

The GCF approach as well as its prototype implementation the Bespoke Framework Generator (BFG v1.0) aims to provide flexibility in both composition and deployment to coupled model developers. In order to use the BFG system, models must conform to the BFG single model rules (Ford & Riley 2003) and metadata must be written to *Describe* the interface and implementation details of each individual model (the blobs with thick lines in figure 2), capture their *Composition* into a coupled system and specify their *Deployment* onto (a set of) resources. This metadata, referred to as DCD metadata, is captured in a small number of XML documents. Using the metadata the BFG software, which is currently written in XSLT, generates the appropriate control (i.e. main) and communication code to enable the models to couple in the specified manner, targeted at deployment on the specified resources. The three steps in the DCD process are now briefly explained. For more details the reader is referred to Ford *et al.* (in press) and Ford & Riley (2003).

(*a*) *Description stage*

The details of each model are captured by the user in three XML documents. The interface.xml document specifies the list of fields required and potentially provided by each model. A field consists of a (scientific) name describing the field (e.g. TEMPP), an (integer) tag which is used to link this metadata name to a data structure in a source code implementation of the model, and a direction, 'IN' or 'OUT', depending on whether the field is required or provided, respectively[2]. We illustrate this with the variable $\bar{T}_{\mathrm{P}}$ whose logic flow is illustrated in figure 2*a*. For the C model interface this field is specified as: `name=TEMPP`, `tag=1`, direction=IN. On the other hand, in the specification of the P^+ model interface we could write `name=TEMPP`, `tag=6`, direction= OUT[3]. The document source.xml specifies other general information about the model such as the type of variable associated with each field (e.g. integer, scalar, array, string), the programming language of an implementation of the model and on what platform(s) the implementation may be run. Finally the document execution.xml provides the model simulation time step (Δt_{P} or Δt_{C}), the size in bytes of each datatype and the location of the source code related to implementations of the model.

(*b*) *Composition stage*

The composition metadata describes the coupling (input/output) connections established between the different models. This is specified in the XML document compose.xml. In our example, the composition would link the field with `tag=6` provided by the P^+ model to the field with `tag=1` required by the C model. In the case of figure 2, the OUTPUT model only requires fields from the scientific models, and it has no provides fields. In this document, the duration of the coupled model simulation is also be specified.

(*c*) *Deployment stage*

In an XML document called deploy.xml the user specifies the mapping of models to executables—whether, for example, a single executable is to be created for the whole coupled scheme which will run on a single machine, or whether different models should be in separate executables in order to run on different machines. This document also specifies the mechanism to be used to implement the communication of coupling fields between models (e.g. using shared memory,

[2] The *tag* has to be unique for each field of a model with direction 'IN' and each field with direction 'OUT'.

[3] The name could be different in each specification. An 'IN' field in one model is linked with an 'OUT' field in another model in the composition metadata.

MPI or a Grid technology such as MPICH-G2). The Grid is a deployment target of particular interest, and BFG can be used in conjunction with the PerCo Performance Control System (Mayes *et al.* 2004, Armstrong *et al.* 2005), also developed by the CNC in the RealityGrid project, to deploy and dynamically control the performance of coupled models on a Grid.

Finally, the coupled.xml document gathers together the names of all the other XML documents. This document, which is the complete specification of the desired coupled model, is passed as a parameter to the BFG processor. In practice, the three documents required to describe a model are developed once by the model developer. For a coupled problem, the coupled model developer has to create only the two documents describing the composition and deployment, and provide the document pointing to all the XML documents involved. In the composition of figure 2, this corresponds to 12 XML documents in all, since for N models there are $3N+3$ files in total, and $N=3$ in this case.

(*d*) *Requirements on model code structure*

In BFG v1.0, the rules require that the model code should be contained within a subprogram unit (i.e., a subroutine for Fortran or C) called with zero arguments. The subprogram unit should declare all the model data. These rules isolate the model code from the control code of the coupled model. In practice, achieving this isolation may be achieved by producing a small piece of 'wrapper' code which makes a call to an existing argument passing routine containing the model code.

In order to send and receive information from/to other models, the user has to add calls to put and get communication routines, the content of which is provided by the BFG for the specific composition and deployment requested. The syntax for these routines is put (*fieldname*, *tag*) and get (*fieldname*, *tag*). The *fieldname* argument provides a reference to the data to be communicated (it is the variable name in the source code) and the *tag* uniquely links the data field involved in a put or a get call to the model's metadata describing the associated field in the model's interface. The *tag* is used in creating compositions of models, thus isolating the composition from model code details such as variable names. To illustrate this point let us use our previous example: at the end of the P^+ model code one may write, for example, put (`TP`, `6`), while at the beginning of the continuum (C) code one could have get(`T_P`, `1`), where `TP` is the name used within the source code of the P^+ subroutine for the scientific variable 'TEMPP', appearing in the model's metadata, which is being exchanged between the models. This same variable is referred to as `T_P` in the source code of the C model.

(*e*) *Summary*

One benefit of the BFG approach is that individual models which comply with the BFG rules can be easily composed into different coupled models by specifying the appropriate composition metadata; *no change to the model code is required.* A second benefit of this approach is that code can be generated to run a particular coupled model on a laptop, on a parallel machine (using MPI) or on a Grid (using distributed MPI), simply by changing the deployment metadata. Again no change to the model code is required. This flexibility allows large simulations to be run on, for example, the potentially disparate resources of a

Grid. Accessing the increased computational power of, for example, the Grid necessarily involves additional memory overheads and some increase in execution time due to the costs of copying and transferring the coupling data. It is obvious that these (communication) overheads need not exist when the target is a single processor machine, such as a workstation. The value of the GCF approach is that the use of BFG provides the flexibility to deploy a coupled model on a range of resources while aiming to reproduce the performance of the original single-processor bespoke implementation.

4. Conclusions

The GCF approach provides the flexibility required by the HybridMD model developers in order to facilitate their continued research. It has been demonstrated that the division of the code into separate (BFG-compliant) models is a relatively simple task. This is, to a large degree, due to the modular nature of the hybrid MD/CFD scheme. The present paper is largely a progress report on our efforts to deploy the HybridMD code using the GCF approach. At the time of writing, an implementation of an early version of the hybrid code exists, but this does not provide the checkpoint-restart facility required for use of the coupled model with the PerCo performance control system (Mayes *et al.* 2005). A full implementation of the latest version of the code is anticipated within the next few months.

In the future, it is anticipated that the HybridMD approach will be extended to include multiple MD regions embedded in a three-dimensional (CFD-modelled) volume, in which the MD regions move and change shape as they track the dynamic development of ‘interesting’ phenomena which require modelling at the molecular level. Such requirements present research challenges for flexible modelling environments and Grid computing systems, such as the BFG and PerCo. From the user’s perspective, one of the biggest current requirements in the use of BFG is the need to expose separate components of the coupled models as subroutines. This is quite straightforward for serial codes such as HybridMD; work with the Met Office in the FLUME project (Ford & Riley 2003) suggests the problems of extracting BFG-compliant models from component codes of any serious degree of complexity, such as the Met Office’s Unified Model, are mainly due to the scale of the task, rather than any increase in the difficulty of dealing with individual models.

We thank EPSRC for funding this research through the RealityGrid grant GR/R67699. Much of the GCF design and the BFG prototyping work at Manchester has been sponsored by the UK Met Office under their FLUME project (Ford & Riley 2003).

References

Armstrong, C. W., Ford, R. W., Gurd, J. R., Lujan, M., Mayes, K. R. & Riley, G. D. 2005 Performance control of scientific coupled models in grid environments. *Concurr. Comput. Pract. Exp.* **17**, 259–295.

Barsky, S., Delgado-Buscalioni, R. & Coveney, P. V. 2003 Comparison of molecular dynamics with hybrid continuum-molecular dynamics for a single tethered polymer in a solvent. *J. Chem. Phys.* **121**, 2403–2411.

De Fabritiis, G., Delgado-Buscalioni, R. & Coveney, P. V. 2004 Energy controlled insertion of polar molecules in dense fluids. *J. Chem. Phys.* **121**, 12 139–12 142.

Delgado-Buscalioni, R. & Coveney, P. V. 2003*a* Hybrid molecular-continuum fluid dynamics. *Phil. Trans. R. Soc. A* **362**, 1639–1654. (doi:10.1098/rsta.2004.1401.)

Delgado-Buscalioni, R. & Coveney, P. V. 2003*b* Continuum-particle hybrid coupling for mass, momentum, and energy transfers in unsteady fluid flow. *Phys. Rev. E* **67**, 046704.

Delgado-Buscalioni, R. & Coveney, P. V. 2003*c* USHER: an algorithm for particle insertion in dense fluids. *J. Chem. Phys.* **119**, 978.

Delgado-Buscalioni, R., Flekkøy, E. & Coveney, P. V. 2005 Fluctuations and continuity in particle-continuum hybrid simulations of unsteady flows based on flux-exchange. *Eur. Phys. Lett.* **69**, 959–965.

Flekkøy, E., Delgado-Buscalioni, R. & Coveney, P. V. In press. Flux boundary conditions in particle simulations. *Phys. Rev. E*

Ford, R. & Riley, G. 2003 A flexible unified model environment, single model software architecture, v. 1.2; (The Met Office, FLUME project). See http://www.metoffice.com/research/interproj/flume.

Ford, R., Riley, G., Bane, M., Armstrong, C. & Freeman, T. L. In press. GCF: a general coupling framework. *Concurr. Comput. Pract. Exp.*

Ideker, T., Galitski, T. & Hood, L. 2001 *Annu. Rev. Genomics Hum. Genet.* **2**, 334–372.

Mayes, K., Riley, G., Ford, R., Lujan, M., Freeman, T. & Addison, C. 2004 The design of a performance steering system for component-based grid applications. In *Performance analysis and grid computing* (ed. V. Getov, M. Gerndt, A. Hoisie, A. Malony & B. Miller), pp. 111–127. Boston: Kluwer Academic Publishers.

Mayes, K. R., Luján, M., Riley, G. D., Chin, J., Coveney, P. V. & Gurd, J. R. 2005 Towards performance control on the Grid. *Phil. Trans. R. Soc. A* **363.** (doi:10.1098/rsta.2005.1607.)

Patankar, S. V. 1980 *Numerical heat transfer and fluid flow.* New York: Hemisphere.