



Physica Ltd, 3 Rowan Drive, Witney, Oxon, OX28 1BH, United Kingdom
+44(0)1993 700051, info@physica.co.uk, www.physica.co.uk

PHYSICA 3.00 - WHITE PAPER

August 2005

1 Introduction

PHYSICA 3.00 is a new release of the well established PHYSICA multi-physics simulation software technology. PHYSICA is characterised by:

- Its software design which has a coherent structure for the solution of the full range of continuum physics and their interactions
- A three dimensional solver strategy using finite volume discretisation on an unstructured mesh of any arbitrary mix of element shapes from tetrahedral to hexahedral.
- Its ability to solve closely coupled thermo-fluid-structure problems
- Its capability for operating scalably on high performance parallel clusters

The existing version of PHYSICA is already rich in physics and numerical solution technology. Users are referred to the PHYSICA website, <http://www.physica.co.uk> for a detailed description of the software technology and a series of publications where the techniques embedded within PHYSICA and their applications are described.

However, in version 3.00, the most recent release of PHYSICA, there is a number of additions to the functionality which merit a basic technical description together with pointers to publications where more detail might be found.

2 Technology improvements to PHYSICA 3.00

2.1 Vertex-based CFD solver technology

In common with most other commercial flow software tools, PHYSICA's default solver technology uses a cell centred approach for mesh discretisation. This is efficient in memory and in compute time – however, it does not do well when parts of the mesh have distorted elements, as may well occur in complex geometries. As such, in this release we are making available a flow solver using vertex based discretisation. Although rather more expensive than cell centred approach in both memory and run time, it is nevertheless, very robust and will converge on just about any kind of mesh quality. This has proved to be especially useful in solving free surface flow problems in mould filling, for example, and in problems where the mesh becomes distorted during the calculation, such as, in fluid structure interaction. In this section we provide an overview description of vertex based discretisation technology embedded within PHYSICA 3.00.

In the vertex based method each element is divided into a number of sub-control volumes by connecting the element centroid to the element face centres. The control volume

consists of a number of sub-control volumes constructed around the mesh vertices. The internal surfaces of the sub-control volumes within the mesh element define the control volume, as illustrated in **Figure 1** for a two-dimensional mesh. The general variable ϕ is solved and stored at the vertices of the mesh elements.

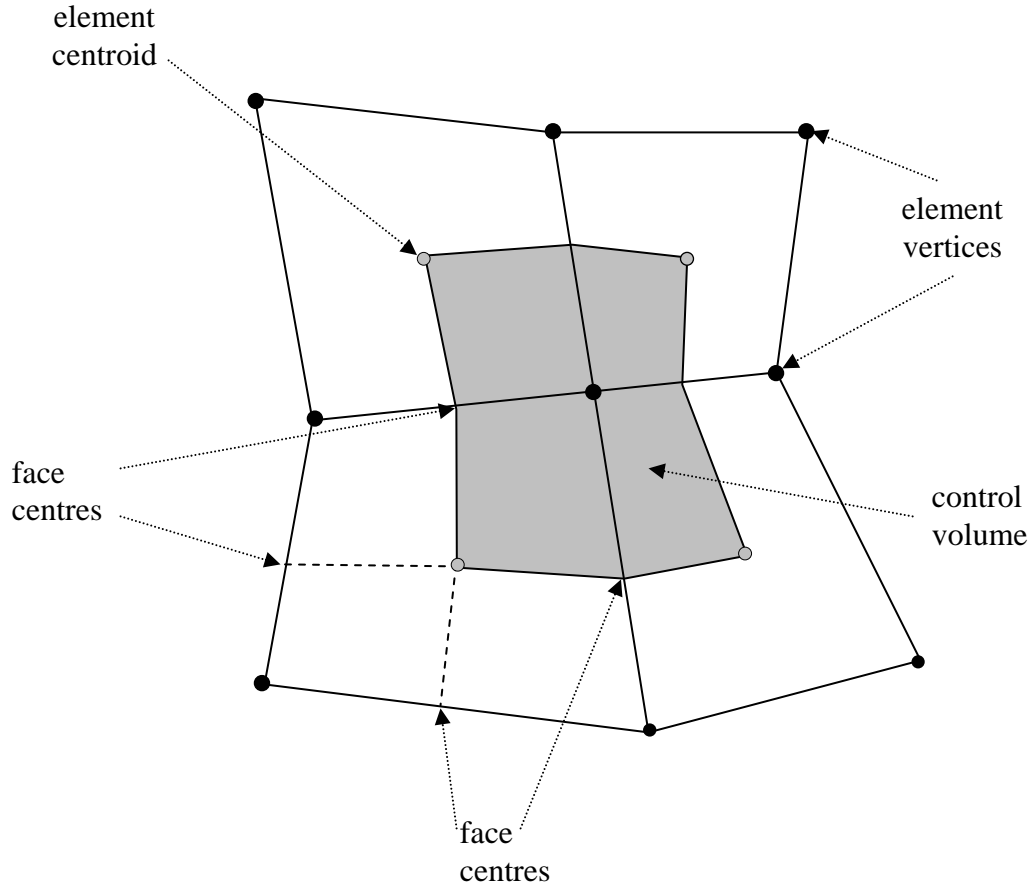


Figure 1 : Vertex-based control volume

The terms in the general transport equation are discretised using simple linear shape function approximations. The shape functions used on each element type and their associated derivatives are defined in local co-ordinates and given in the documentation of the **vtxb** module.

The local co-ordinate system is transformed into global co-ordinates, the co-ordinate transformation is performed using the same shape functions. The elements described here are iso-parametric, allowing for both the co-ordinate transformations and the variable approximation within the element to be described by the same shape functions. The local to global transformations, the variation of a variable within the element and the partial derivatives of the variable with respect to local co-ordinates are described in the documentation of the **vtxb** module.

2.1.1 Transient term

In the vertex based method the control volume consists of a number of sub-control volumes. Hence the transient term can be written as

$$\int_{t-\Delta t}^t \int_{CV} \frac{\partial(\rho\phi)}{\partial t} dV dt = \int_{t-\Delta t}^t \sum_{scv} \int_{scv} \frac{\partial(\rho\phi)}{\partial t} dV dt \quad (1)$$

Discretisation of this term follows the same methods as the cell centred to give a contribution of the form

$$\frac{1}{\Delta t} \sum_{scv} \left(\rho_{scv} V_{scv} \phi_{scv} - \rho_{scv}^0 V_{scv}^0 \phi_{scv}^0 \right) \quad (2)$$

The chosen integration point could be any position within the sub-control volume. The position is assumed to be representative of the average value in the sub-control volume and presumed to prevail over the whole sub-control volume. The value of ϕ and position in local co-ordinates can be described by local shape functions giving the discretised term as,

$$\frac{1}{\Delta t} \sum_{scv} \sum_{i=1}^n \left(\rho_{scv} V_{scv} N_i \phi_i - \rho_{scv}^0 V_{scv}^0 N_i \phi_i^0 \right) \quad (3)$$

where n is the number of element nodes. N_i is the shape function associated with node i and ϕ_i is the value of the variable at node i .

The logical integration point is the centre point of a sub-control volume. Care must be taken when using this form of discretisation since adding to neighbouring coefficients can lead to $\sum a_{nb}$ becoming negative resulting in 'unbounded' or diverging solutions.

2.1.2 Source term

As with the cell centred approach the source is ideally expressed in a linearised form

$$S_\phi = S_c - S_p \phi \quad (4)$$

Integration of this term over the control volume leads to a sum over contributions over sub-control volumes

$$\int_{cv} (S_c - S_p \phi) dV = \sum_{scv} \int_{scv} (S_c - S_p \phi) dV \quad (5)$$



Approximating the integrand in each sub-control volume and using shape functions to express the value of ϕ at the chosen integration point gives

$$\sum_{scv} \sum_{i=1}^n V_{scv} \left((S_c)_{scv} - (S_p)_{scv} N_i \phi_i \right) \quad (6)$$

where all terms are evaluated at an elemental level and subscript i refers to element nodal values. If the source term is large this form of discretisation can result in diverging solutions that do not satisfy boundedness, since it contains the potential for $\sum a_{nb}$ to become negative.

2.1.3 Diffusion term

As with the cell centred approach integration of the diffusion term results in the need to approximate

$$\int \Gamma \text{grad} \phi dS \quad (7)$$

where the integration is over all faces of the control volume. Approximation of this integral leads to

$$\sum_f \sum_{i=xyz} \Gamma_f A_f \left(\frac{\partial \phi}{\partial x_i} \cdot n_i \right)_f \quad (8)$$

where A_f is the area of face f .

The partial derivatives of ϕ are calculated at integration points situated at the face centres and can be described by partial derivatives of the shape functions. Hence the discretised form of the equation is,

$$\sum_f \Gamma_f A_f \sum_{i=1}^3 \sum_{j=1}^n \phi_j \left(\frac{\partial N_j}{\partial x_i} \cdot n_i \right)_f \quad (9)$$

where N_j is the local shape function associated with node j on the element that contains the face f and ϕ_j is the value of the variable at node j .

The discretised diffusion term for internal solution points leads to a positive indefinite solution matrix, giving a solution matrix which is near singular and extremely susceptible to rounding error. The addition of boundary conditions stabilises the matrix and fulfils positive definite requirements.

2.1.4 Convection term

The vertex based discretisation of the convection follows the same route as the cell centred method up to the calculation of the face value of the dependent variable. Assuming that the velocity field has been resolved all that remains is to estimate the value of the solved variable on each face of the control volume. A number of methods, based on those used for the cell centred approach, are available.

Central difference discretisation

In the central difference scheme the face value is calculated simply using interpolation

$$\phi_f = \sum_{i=1}^n N_i \phi_i \quad (10)$$

where ϕ_i is the variable described at node i of the element which contains face (f), giving the discretised convection term as:

$$\sum_f \rho_f A_f (\underline{u} \cdot \underline{n})_f \sum_{i=1}^n N_i \phi_i \quad (11)$$

In discretising the convection term over a control volume, only surrounding nodal values of ϕ appear in the equation, terms connecting ϕ at the control volume node cancel out. This produces a positive-definite system of equations with the leading diagonal containing zero coefficients. The solution of the system matrix requires division by the leading coefficients, division by zero is encountered. Since diffusion always occurs along-side convection in nature, the problem of zero leading coefficients is overcome by coupling the convection and diffusion terms. This method produces stable and accurate solutions when the strength of convection relative to diffusion is low. When convection dominates the sum of the neighbouring coefficients can become negative. The boundedness criterion is violated and the solution fails to converge or produces physically unrealistic oscillating results.

Upwind discretisation

Using an upwind formulation the direction of flow is ascertained on each cell face. The value of ϕ on the cell face is taken as the nodal value in the upstream control volume. This method produces a system of equations which is well conditioned, contains non-zero leading coefficients, and gives stable and converging solutions for convection and convection-diffusion problems. The upwind discretisation satisfies transportiveness requirements but is only first order accurate which makes it prone to numerical diffusion. The vertex-based approach allows more flexibility in selecting upwind values. A vertex-based control volume comprises a far greater number of faces than a cell-centred control volume. As each face is upwinded the combination of different upwind values for a vertex-based control volume is greater than for a cell-centred control volume. The



upwind nodal value is also influenced by a greater number of neighbouring values, shown in **Figure 2** for hexahedral mesh. The indirect influence of neighbouring values of ϕ on ϕ_f is much increased using vertex-based techniques.

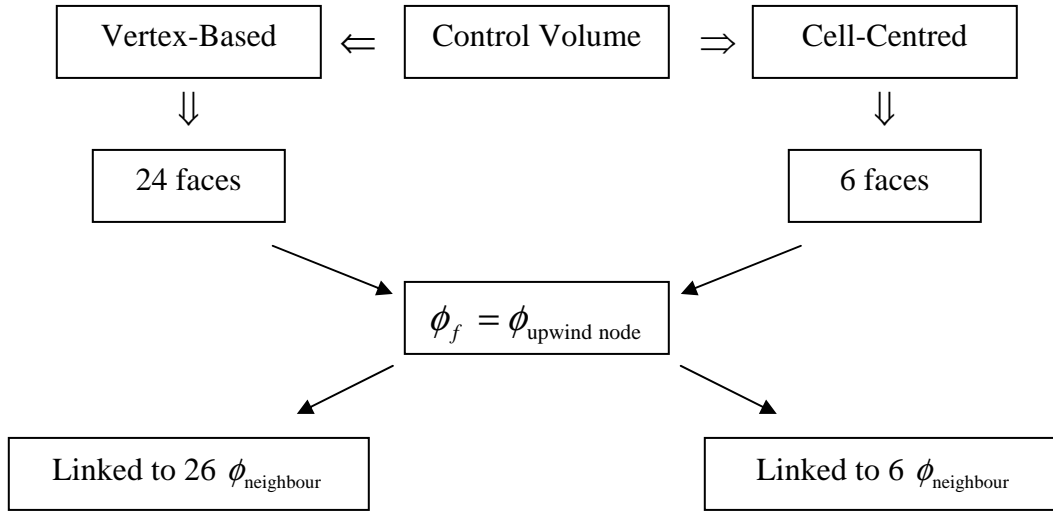


Figure 2 : Influence of nodal values on face values for hexahedral elements

Hybrid discretisation

A measure of the relative strengths of convection and diffusion can be defined by a non-dimensional cell Peclet number (Pe), shown below,

$$Pe = \frac{\rho u \delta x}{\Gamma} \quad (12)$$

where δx is the element width across the centre point of the face in the direction of the face normal vector, and u is the resultant velocity at the face centre.

The local Pe number is computed on each cell face, employing linear shape function interpolation for small Peclet numbers ($|Pe| < 2$) and the upwind formulation for large Peclet number ($|Pe| > 2$).

$$\begin{aligned} |Pe| < 2 & \quad \phi_f = \sum_{i=1}^n N_i \phi_i \\ |Pe| > 2 & \quad \phi_f = \phi_{\text{adjacent upwind cv}} \end{aligned} \quad (13)$$

For strongly convective flows ($|Pe| > 2$), the face value of ϕ is more strongly influenced by the relevant upstream value. As $|Pe|$ increases, $\partial\phi/\partial x$ tends to zero, indicating that the diffusive effect becomes negligible. Hence, for large $|Pe|$ values inclusion of the diffusion term in the upwind formulation leads to diffusion becoming overestimated.

The hybrid scheme is the default differencing scheme in many CFD codes due to its stability and robustness. It gives good approximations to exact solution when flow aligns closely with grid lines and sources are small. Numerical accuracy is degraded due to the

introduction of artificial numerical diffusion whenever flow streamlines are at an angle to grid lines. Solutions can suffer from a jump at points where the local $|Pe| = 2$ due to the suppression of physical diffusion.

Sources of more in-depth information underlying the vertex based solver strategy is provided in the bibliography at the end of this paper, see refs [1-3]

2.2 Viscoelastic flow

In the viscoelastic flow models implemented in PHYSICA, the continuity and momentum equations are given by

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \underline{u}) = 0 \quad (14)$$

$$\frac{\partial \rho u_i}{\partial t} + \nabla \cdot (\rho \underline{u} u_i) = -\nabla p + \nabla \cdot \underline{T} \quad (15)$$

where \underline{T} is the extra-stress tensor given by

$$\underline{T} = \begin{pmatrix} T^{xx} & T^{xy} & T^{xz} \\ T^{yx} & T^{yy} & T^{yz} \\ T^{zx} & T^{zy} & T^{zz} \end{pmatrix} \quad (16)$$

Note that the stress tensor is symmetrical, i.e., $T^{xy} = T^{yx}$, $T^{xz} = T^{zx}$ and $T^{yz} = T^{zy}$. The components T^{xx} , T^{yy} and T^{zz} are called the normal stresses, and the components T^{xy} , T^{yz} and T^{zx} are called the shear stresses.

The value of \underline{T} is determined through the solution of a constitutive equation. Two viscoelastic constitutive equations have been implemented in PHYSICA - the Upper Convected Maxwell model and the Oldroyd-B model.

2.2.1 Upper Convected Maxwell (UCM) model

In the UCM model the constitutive equation is defined by

$$\underline{T} + \lambda_1 \overset{\nabla}{\underline{T}} = 2\mu_0 \underline{D} \quad (17)$$

where the upper convected derivative $\overset{\nabla}{\underline{T}}$ is given by

$$\overset{\nabla}{\underline{T}} = \frac{\partial \underline{T}}{\partial t} + \underline{u} \nabla \underline{T} - \underline{T} \nabla \underline{u} - \nabla \underline{u}^T \underline{T} \quad (18)$$



The rate of deformation tensor \underline{D} is given by

$$\underline{D} = \frac{1}{2} (\nabla \underline{u} + \nabla \underline{u}^T) \quad (19)$$

Here λ_1 is the relaxation time and μ_0 is the dynamic total viscosity.

The full set of equations for the UCM model is given by

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \underline{u}) &= 0 \\ \frac{\partial (\rho u_i)}{\partial t} + \nabla \cdot (\rho u u_i) &= -\nabla p + \nabla \cdot \underline{T} \\ \underline{T} + \lambda_1 \overset{\nabla}{\underline{T}} &= 2\mu_0 \underline{D} \end{aligned} \quad (20)$$

2.2.2 Oldroyd-B model

The Oldroyd-B constitutive equation is defined by

$$\underline{T} + \lambda_1 \overset{\nabla}{\underline{T}} = 2\mu_0 \left(\underline{D} + \lambda_2 \overset{\nabla}{\underline{D}} \right) \quad (21)$$

where the total viscosity μ_0 is defined by

$$\mu_0 = \mu_s + \mu_p$$

with μ_s as the solvent viscosity and μ_0 as the polymeric viscosity. The retardation time of the fluid λ_2 is given by

$$\lambda_2 = \frac{\mu_s}{\mu_0} \lambda_1$$

Separating the solvent and elastic contributions to the stress gives

$$\underline{T} = 2\mu_s \underline{D} + \underline{\tau} \quad (22)$$

where $\underline{\tau}$ represents the elastic contribution. Substituting (22) into (21) gives

$$\underline{\tau} + \lambda_1 \overset{\nabla}{\underline{\tau}} = 2\mu_p \underline{D} \quad (23)$$

Using (22) causes the momentum equation (15) to be written as

$$\frac{\partial(\rho u_i)}{\partial t} + \nabla \cdot (\rho \underline{u} u_i) = -\nabla p + \nabla \cdot (\mu_s \nabla u_i) + \nabla \cdot \underline{\tau}$$

and within PHYSICA the Oldroyd-B model has been implemented in the following form:

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \underline{u}) &= 0 \\ \frac{\partial(\rho u_i)}{\partial t} + \nabla \cdot (\rho \underline{u} u_i) &= -\nabla p + \nabla \cdot (\mu_s \nabla u_i) + \nabla \cdot \underline{\tau} \\ \underline{\tau} + \lambda_1 \overset{\nabla}{\underline{\tau}} &= 2\mu_p \underline{D} \end{aligned} \quad (24)$$

NB If $\mu_s = 0$ then the UCM model is recovered i.e., $\underline{T} = \underline{\tau}$, $\mu_0 = \mu_p$ and

$$\underline{T} + \lambda_1 \overset{\nabla}{\underline{T}} = 2\mu_0 \underline{D}$$

2.3 Non-Newtonian models for viscosity

The viscosity models which have been implemented are the Power Law model, the Cross model, the Carreau-Yasuda model and the Bingham model. In general these are entered through the **inform** as **dynamic viscosity** models.

The **Bingham** model is defined by

$$\begin{aligned} \dot{\gamma} &= 0 & \text{for } \tau < \tau_y \\ \tau &= \mu_p \dot{\gamma} + \tau_y & \text{for } \tau \geq \tau_y \end{aligned} \quad (25)$$

where τ is the shear stress, τ_y is the yield stress and μ_p is the plastic viscosity. The shear rate $\dot{\gamma}$ is defined by

$$\dot{\gamma} = \nabla \underline{u} + \nabla \underline{u}^T \quad (26)$$

The implementation of the model in PHYSICA is based on a model proposed by Papanastasiou as follows

$$\mu = \mu_p + \frac{\tau_y}{|\dot{\gamma}|} (1 - \exp(-m|\dot{\gamma}|)) \quad (27)$$



where m is the stress growth exponent. The model tends to true Bingham behaviour as $m \rightarrow \infty$. The effective shear rate $|\dot{\gamma}|$ is given by

$$|\dot{\gamma}| = \sqrt{\frac{1}{2} \dot{\gamma}_{ij} \dot{\gamma}_{ij}} \quad (28)$$

In the **Carreau-Yasuda** model the dynamic viscosity μ is defined by

$$\mu = \mu_{\infty} + (\mu_0 - \mu_{\infty}) \left(1 + (K|\dot{\gamma}|)^a \right)^{(n-1)/a} \quad (29)$$

where μ_0 is the viscosity at zero shear rate, μ_{∞} is the viscosity at infinite shear rate, K is a constant parameter and n is a dimensionless constant. The effective shear rate is defined by (28). **N.B.** The **Bird-Carreau** model is given by $a = 2$. For many shear thinning fluids $a \approx 2$.

The equation for the **Cross** model is given by

$$\mu = \frac{\mu_0 - \mu_{\infty}}{1 + (K|\dot{\gamma}|)^m} + \mu_{\infty} \quad (30)$$

where μ_0 is the viscosity at zero shear rate, μ_{∞} is the viscosity at infinite shear rate, K is a constant associated with the breaking of structural linkages and m is a dimensionless constant. The effective shear rate is defined by (28).

The equation for the **Power Law** model is given by

$$\mu = K|\dot{\gamma}|^{n-1} \quad (31)$$

where K is the consistency of the material, and n is the power law index. The effective shear rate is defined by (28).

Sources for more information on non-Newtonian and viscoelastic fluid modelling used in PHYSICA and their applications is provided in refs [4-6].

2.4 Group solver technology

One of the main aspects of multi-physics simulation is the spatial and temporal variation in the physics that require solution. For example, consider the casting process which involves the filling of a mould and subsequent solidification of the cast piece. In the mould, assuming it to be non-porous, the thermal field and mesh deformation require resolution but there is no need to solve for fluid flow. In the cast piece there will always



be the requirement to solve the thermal field but as it solidifies there will be a change from solving the hydrodynamics to solving structural mechanics on an element by element basis. One mechanism for the solution of such a system is to employ suitable sources, constraint equations or material properties to ensure that variables retain sensible values at locations they need not be calculated. The advantage of such an approach is that the conventional whole field solution strategy can be employed. The cost is that computational effort is wasted on calculations associated with values that do not require solution.

An alternative approach is to define groups of elements and solve only over the elements on the 'ON' group. The elements in the 'OFF' group retain their initial value, this is not inconsequential as fluxes between 'ON' and 'OFF' elements will still be calculated. The benefit of the use of groups is that fluxes between and sources in unsolved elements need not be evaluated. The cost is that logical tests need to be added to test whether calculations are necessary, this slightly complicates the coding of algorithms

A more detailed description of this technology and its assessment is in Williams et al [7]. Results obtained for metal extrusion examples, which involve closely coupled free-surface flow, heat transfer and structural mechanics, have shown that using the group solver can reduce run-times by approximately 30%.

2.5 Finite element discretisation for Structural Mechanics

The deformation of a structure can now be solved using the finite element method in addition to the previously available vertex based – finite volume method. All the functionality which was present in the finite volume method is available for use with the finite element method.

The universal law governing any continuum undergoing motion is given by Cauchy's equation:

$$\nabla \cdot \boldsymbol{\sigma} + \mathbf{b} = \rho \mathbf{a} \quad (32)$$

where $\boldsymbol{\sigma}$ is the stress tensor, \mathbf{b} is a vector of body forces and \mathbf{a} is the acceleration of the structure, which for a static implementation is zero. The generalised form of Hooke's law gives the following constitutive relationship between stress and strain for an isotropic homogeneous material undergoing small strains in three dimensions

$$\boldsymbol{\sigma} = \mathbf{D} \boldsymbol{\varepsilon}_{el} \quad (33)$$

where \mathbf{D} is defined in terms of Young's modulus and Poisson's ratio in the conventional manner and $\boldsymbol{\sigma} = (\sigma_x, \sigma_y, \sigma_z, \sigma_{xy}, \sigma_{yz}, \sigma_{xz})^T$ is the stress vector and $\boldsymbol{\varepsilon}_{el} = (\varepsilon_x, \varepsilon_y, \varepsilon_z, \varepsilon_{xy}, \varepsilon_{yz}, \varepsilon_{xz})^T$ is the elastic strain vector. The structural mechanics approach is based on a linear strain–displacement formulation using the small strain assumption, which is valid for strains of the order of a few percent. Thus the strains may be decomposed into the



product of the matrix of linear operators and the displacement vector which, for a 3D approximation, enables the strains to be defined in the general displacement form:

$$\boldsymbol{\varepsilon}_{el} = \mathbf{L}\mathbf{d} \quad (34)$$

where \mathbf{L} is the differential operator and \mathbf{d} is the displacement vector. Thus Cauchy's equation (32) for static analysis, where $\mathbf{a} = 0$, may be expressed as:

$$\mathbf{L}^T(\mathbf{D}\mathbf{L}\mathbf{d}) + \mathbf{b} = \mathbf{0} \quad (35)$$

Equation (35) is subject on the surface $\Gamma = \Gamma_t \cup \Gamma_d$ to a traction and displacement boundary conditions, such that:

$$\begin{aligned} \mathbf{R}^T \boldsymbol{\sigma} &= \mathbf{t}_p & \text{on } \Gamma_t & \text{ and} \\ \mathbf{d} &= \mathbf{d}_p & \text{on } \Gamma_d \end{aligned} \quad (36)$$

where \mathbf{R} is the outward normal operator, \mathbf{t}_p are the prescribed tractions on the boundary Γ_t and \mathbf{d}_p are the prescribed displacements on the boundary Γ_d . Applying the method of weighted residuals to the equilibrium equation (35) subject to the boundary conditions, as expressed by equation (36), gives the familiar weak form of the equilibrium equations [9],[10]

$$-\int_{\Omega} [\mathbf{L}\mathbf{W}]^T (\mathbf{D}\mathbf{L}\mathbf{d}) d\Omega + \int_{\Omega} [\mathbf{W}]^T \mathbf{b} d\Omega + \int_{\Gamma_d} [\mathbf{R}\mathbf{W}]^T (\mathbf{D}\mathbf{L}\mathbf{d}) d\Gamma + \int_{\Gamma_t} [\mathbf{W}]^T \mathbf{t}_p d\Gamma = \mathbf{0} \quad (37)$$

In both FE and FV methods the independent unknown displacements are approximated using:

$$\mathbf{d} \approx \mathbf{N}_j \hat{\mathbf{d}}_j \quad \text{for } j=1, \dots, n \quad (38)$$

where $\hat{\mathbf{d}}_j$ are the unknown displacement approximation and \mathbf{N}_j are the basis functions

Equation (37) may be written as a system of linear equations:

$$\mathbf{K}\hat{\mathbf{d}} = \mathbf{f} \quad (39)$$

where

$$\mathbf{K} = \int_{\Omega} [\mathbf{L}\mathbf{W}]^T (\mathbf{D}\mathbf{L}\mathbf{N}) d\Omega - \int_{\Gamma_d} [\mathbf{R}\mathbf{W}]^T (\mathbf{D}\mathbf{L}\mathbf{N}) d\Gamma \quad (40)$$

and

$$\mathbf{f} = \int_{\Omega} [\mathbf{W}]^T \mathbf{b} d\Omega + \int_{\Gamma_f} [\mathbf{W}]^T \mathbf{t}_p d\Gamma \quad (41)$$

The essential difference between the FE and the FV methods is in the definition of the weighting functions. In the Bubnov-Galerkin FE discretisation the weighting functions are the shape functions, \mathbf{N} . Whereas for FV discretisation the weighting functions are taken to be the identity matrix within the control volume and zero elsewhere [10], i.e.

$$\mathbf{W} = \mathbf{I} \quad \text{in} \quad \Omega \quad (42)$$

and zero elsewhere. Thus for the FE method

$$\mathbf{K} = \int_{\Omega} [\mathbf{LW}]^T (\mathbf{DN}) d\Omega \quad (43)$$

and for the FV method.

$$\mathbf{K} = - \int_{\Gamma_d} [\mathbf{RW}]^T (\mathbf{DL}) d\Gamma \quad (44)$$

The load vector \mathbf{f} is the same for both methods. The numerical issues underlying the FE implementation are covered in [8].

2.5.1 Element types

The following element types are available for both the finite element and finite volume structural mechanics discretisations within PHYSICA 3.00:

- 2-noded trusses
- 3-noded constant strain triangles
- 4-noded bilinear quadrilaterals
- 8-noded trilinear hexahedra
- 6-noded wedges
- 4-noded tetrahedra
- 5-noded square-based pyramids

3 Code restructuring

The following sections describe the code restructuring that has taken place within PHYSICA 3.00. This restructuring, for both the structural mechanics and computational fluid dynamics sections of PHYSICA, has been carefully designed to ensure backward compatibility with **inform** files that were created for use with PHYSICA 2.12.



3.1 Restructuring of the Structural Mechanics

The **evp** module in version 2.12 has been split into two parts, namely the **vtxb** and **struct** modules. The vertex based module (**vtxb**) contains the utility routines that can be employed as part of the discretisation of any vertex based variable. The routines in this module are currently split into 4 distinct classes which handle the:

1. Geometric requirements of the vertex based method. Namely the calculation of geometric quantities associated with the sub-control volumes and the construction of index array to speed access to data.
2. Construction of the standard system matrix. This set of routines calculates the memory requirements of the arrays associated with the system matrix before constraints are considered.
3. Handling of the constraint equations
4. Issues associated with the shape functions and integration points.

The remainder of the routines previously in the **evp** module have been moved into the new structural mechanics (**struct**) module. Currently this module contains 3 files which handle

1. The general routines that link with the remainder of PHYSICA
2. The routines related to an elastic solution.
3. The routines related to a visco-plastic solution.

3.2 Restructuring of the Computational Fluid Dynamics

There has been a major restructuring of the physical modules related to the solution of the CFD variables. Now all CFD variables are created as variables in the **scalar** module. The advantage of this is that any functionality available to any CFD variable is now available to all. This results in consistency across all variables and reduces that effort required to add new functionality to PHYSICA. The main cosmetic difference caused by this change is that the amount of code in most physical modules has been significantly reduced at the cost of an enlarged **scalar** module. Similarly all the values related to the solution of a variable are now held by the **scalar** module and consequently there has been a large reduction in the number of common variables in each module.

Access routines have been provided to allow any of the values held by the scalar module to be either retrieved or set. Separate routine exist for each data type and have the following form

```
CALL scalar_pointer
@      ( Ia, Ra, Xa, Dra, La, Cha, Vrname, Var_id, Mode,
@      Ptr_id, Ptr, Errinf, Failed)
```

for pointers,

```
CALL scalar_real_values
@      ( Ia, Ra, Xa, Dra, La, Cha, Vrname, Var_id, Mode,
@      Val_id, Value, Errinf, Failed)
```

for real valued variables,

```
CALL scalar_integer_values
@      ( Ia, Ra, Xa, Dra, La, Cha, Vrname, Var_id, Mode,
@      Val_id, Value, Errinf, Failed)
```

for integer valued variables and



```
CALL scalar_logical_values
@      ( Ia, Ra, Xa, Dra, La, Cha, Vrname, Var_id, Mode,
@      Val_id, Value, Errinf, Failed)
```

for logical variables. In this routines the large arrays (Ia to Cha) and error flags (Errinf, Failed) are common to many routines and so will be ignored. A description of the important arguments follows

Vrname The memory manager name of the scalar variable. If the Var_id is not a positive number then this string will be used to identify which scalar is of interest.

Var_id The number of the scalar variable. If the number is not known then pass a variable set to zero into the routine. The variable name will then be used to calculate this value and on return the variable passed into the routine will be reset to the correct identifier.

Mode Can be set to either of the parameters SCALAR_SET or SCALAR_GET dependent on whether a value is to be set or retrieved. The parameters can be located in the header file `inc/scalar.fh`.

Val_id Identifier of which value associated with the scalar is to be set. A full list of the parameters that can be used for the argument can be found in `inc/scalar.fh`.

Value The value to which the indicated scalar value is to set or the variable into which the retrieved value will be placed.

4 Additional developments

4.1 Support for ANSYS format output

The ANSYS result format will produce a set of files that can be read into its own post processor. The files written by the ANSYS filter are:

1. A file containing the nodes that can be read using the ANSYS command NREAD. This file will have the extension `.axy`.
2. A file containing the topological information that can be read using the command EREAD. This file will have the extension `.ael`
3. A set of result files that can be read using the VREAD command. These files all have a `.avr` extension.
4. A file called `vars.avr` that contains details of the result files.

As with most other formats output to the ANSYS result file(s) can be at the end of the simulation only, at a selected frequency of time steps or at a selected time frequency. The base name for the node and element topology files may be set, through the **inform**, and will default to the unextended name of the PHYSICA geometry file. For a steady state simulation the name of a result file will be the first 8 letters of the memory manager name of the variable followed by `'.avr'`. If results are being written at selected times or time steps then the name will have a `'_'` followed by the count of time steps written, in I4.4



format, added after the variable name. As an example the name of the file associated with pressure at the fourth write will be 'PN_0004.avr'

The topology file will only accept brick, wedge, tetrahedral and pyramid elements. All elements are written as (degenerate) bricks. If any other type of element exists in the mesh then an error will be raised when writing the ANSYS file and PHYSICA will terminate.

Only variables located at nodes and elements can be written to result files. Element based values will be converted into an equivalent node based variable before writing.

The format of the vars.avr file is

1. A header line that contains the string 'NUMB' followed by the number of written variables, the number of time steps at which results are written and the number of variables written at each time step. The first and third numbers may differ because some variables may only be written at the end of the simulation. The format used in A8, 3I8.
2. A line for each variable containing the name of the variable, its width, the base name of the file associated with the variable and the extension. In all cases the variable name and the base file name will be identical and the extension will always be avr. The format is designed to make the file easy to read using Ansys macros. The format of this line is A8, I8, A8, 'avr '

An example of the file is

```
NUMB          3          2          2
VELOCITY      3VELOCITYavr
PN            1PN        avr
TN            1TN        avr
```

4.2 Display of sources

It is now possible to print the sources due to all face patches, element patches and extra terms (both standard and user defined) at the end of the simulation for all CFD variables. There is no control over which sources or which variables should be displayed. The format of the output is as follows:

```

-----
Variable : MASS
  Face Patches
      1          = 1.00000E+01
      2          = -9.99996E+00
      3          = 0.00000E+00
      4          = 0.00000E+00
      5          = 0.00000E+00
=====
      Balance    = 4.00543E-05
-----

```

```

-----
Variable : UN
  Face Patches
      1          = 1.00000E+00
      2          = 9.20888E-07
      3          = -1.57542E+00
      4          = -1.71319E+00
      5          = 0.00000E+00
  Extra Terms
      FLOW_PGRADS = 2.28868E+00
=====
      Balance    = 6.74725E-05
-----

```

5 Conclusions

The objective of this White Paper is to provide a more in-depth description of the new features available in PHYSICA 3.00. Further details can be obtained by visiting www.physica.co.uk or by emailing info@physica.co.uk.

Bibliography

1. D. McBride, *Vertex based discretisation methods for thermo-fluid flow in a finite volume –unstructured mesh context*, PhD Thesis, University of Greenwich (2003)
2. M. Cross, K. Pericleous, T. N. Croft, D. McBride, J. A. Lawrence and A. J. Williams, *Computational modelling of mold-filling and related free surface flows in shape casting: An overview of the challenges involved*, in *Shape Casting: the John Campbell Symposium* (ed. M. Tiryakioglu and P. N. Crepeau), pp 305-316, Pub TMS (2005)
3. D. McBride, T. N. Croft and M. Cross, *Combined Vertex-Based – Cell-Centred Finite Volume Method for Flows in Complex Geometries*, CD Proceedings of the 3rd International Conference on CFD In Minerals and Process Industries, pub CSIRO, Melbourne, Australia (2003)



4. S. S. Edussuriya, *Computational Analysis of Viscoelastic Free Surface Flows*, PhD Thesis, University of Greenwich, London (2003)
5. S. S. Edussuriya, A. J. Williams, and C. Bailey, *A CFD Study of the Underfill Encapsulation Process in Flip-Chip Packaging Using the Oldroyd-B Model*. Proceedings of the 6th European Conference on Rheology, Ed(s): H. Munstedt, J. Kaschta and A. Merten, pp. 285-286 (2002)
6. S. S. Edussuriya, A. J. Williams and C. Bailey, *A Cell-Centred Finite Volume Method for Modelling Viscoelastic Flow*, Journal of Non-Newtonian Fluid Mechanics, 117, 47-61 (2004)
7. A. J. Williams, T. N. Croft and M. Cross, *A group based solution strategy for multi-physics simulations in parallel*, to appear in Applied Mathematical Modelling
8. A. K. Slone and M. Cross, *A comparison of finite element and finite volume methods for computational structural mechanics and their applications in multi-physics problems*, CD Proceedings of 4th Intl Conf on Engineering Technology, Lisbon, paper 71 (2004)
9. O. C. Zienkiewicz and R. L. Taylor, *The Finite Element Method: Volumes 1-3*, Butterworth-Heinemann, (2000)
10. E. Oñate, M. Cervera and O. C. Zienkiewicz, *A finite volume format for structural mechanics*, International Journal for Numerical Methods in Engineering 37, pp181-201, (1994)