AN OBJECT ORIENTED IMPLEMENTATION OF
A FRONT TRACKING FINITE ELEMENT METHOD FOR
DIRECTIONAL SOLIDIFICATION PROCESSES

RAJIV SAMPATH AND NICHOLAS ZABARAS *

Cornell University, Sibley School of Mechanical and Aerospace Engineering, 188 Frank H. T. Rhodes Hall,
Ithaca, NY 14853-3801, U.S.A.

SUMMARY

This paper focuses on the numerical simulation of phase-change processes using a moving finite element
technique. In particular, directional solidification and melting processes for pure materials and binary alloys
are studied. The melt is modelled as a Boussinesq fluid and the transient Navier–Stokes equations are solved
simultaneously with the transient heat and mass transport equations as well as the Stefan condition. The
various streamline-upwind/Petrov–Galerkin-based FEM simulators developed for the heat, flow and mass
transport subproblems are reviewed. The use of classes, virtual functions and smart pointers to represent and
link the particular simulators in order to model a phase change process is discussed. The freezing front is
modelled using a spline interpolation, while the mesh motion is defined from the freezing front motion using a
transfinite mapping technique. Various two- and three-dimensional numerical tests are analysed and discussed
to demonstrate the efficiency of the proposed techniques. Copyright © 1999 John Wiley & Sons, Ltd.

KEY WORDS: object oriented programming; solidification; phase change; finite element method; front tracking technique

1. INTRODUCTION

Phase transformation from liquid to solid is a phenomenon central to a wide range of material
processes. The presence of phase transformations can induce melt convection through the liberation
of latent heat, the rejection of solute, the change of density upon freezing and by various capillary
forces on the solid–liquid interfaces. The phase transformation itself can be strongly altered by
convective transport in the liquid melt that affects the thermal and solutal environment in the
neighborhood of the solid–liquid freezing interface.1, 2 These local fields control the morphological
characteristics of the freezing interface.3

Phase change in binary alloy systems differs in many respects from solidification or melting of
pure substances. The phase transformation for alloys takes place over a temperature range rather
than at a discrete temperature. In addition, the solute has different solubilities in the liquid and
solid phases. Hence, the solute may be preferentially incorporated or rejected at the solid–liquid

* Correspondence to: Nicholas Zabaras, Sibley School of Mechanical and Aerospace Engineering, 188 Frank H. T. Rhodes
Hall, Cornell University, Ithaca, NY 14853-3801, U.S.A. E-mail: njz1@cornell.edu

Contract/grant sponsor: NSF
Contract/grant sponsor: IBM Corporation
Contract/grant sponsor: New York State
Contract/grant sponsor: CNRS

CCC 0029–5981/99/091227–39$17.50
Copyright © 1999 John Wiley & Sons, Ltd. Received 19 April 1997
Revised 9 March 1998
interface leading to competing mechanisms of heat and mass transfer in the melt. This combined process can lead to complex fluid patterns known as double-diffusive convection. Furthermore, the solid–liquid interface is usually not smooth but a variety of microscopically complex growth structures can develop. The region characterized by the presence of such irregular interfaces is often called a ‘mushy zone’.

It is clear that understanding of the various transport processes is absolutely essential for a successful design and control of solidification processes. Due to the complex nature of these processes, experimentation is limited to specific systems (for a recent experimental work see Reference 5). Moreover, analytical solutions for phase change processes are quite limited. In order to address multidimensional effects and non-discrete phase change, attention has been focused on the development of suitable numerical procedures.

Direct numerical simulations of phase change processes can currently be classified into two groups based on the techniques employed. The first group utilizes independent conservation equations for each phase and couples them with appropriate boundary conditions at the freezing interface. Such methods are often referred to as ‘multiple domain methods’. The second group consists of a single region (continuum) formulation which eliminates the need for separate conservation equations at the freezing front. The latter ‘enthalpy’ based methods are quite popular for problems that do not require precise tracking of the freezing front. Such formulations are generally developed from volume averaging techniques based on classical mixture theory and have been shown to provide realistic predictions of transport behavior for various problems.

Front tracking and front fixing techniques belong to the category of multiple domain methods and are used to explicitly and accurately track the moving freezing interface. The complexity of moving and deforming finite element grids and of co-ordinate mapping procedures has up to now limited the application of front tracking techniques to two-dimensional conduction-based solidification problems.

In the present work, a general FEM simulator is presented for the front tracking analysis of phase change processes for both pure metals and alloys. For alloy solidification, we a priori assume the existence of a stable sharp solid–liquid freezing front. The applicability of such a model is limited to systems with a thin mushy layer and to crystal growth processes with low growth rate. To the best of our knowledge, this is the first time that such a dynamic multidimensional front tracking deforming FEM simulation of binary alloy solidification processes is presented. In the present analysis, B-splines and tensor product surfaces (for three dimensional problems) are used for the representation of the freezing front.

A dynamically coupled solver using an object oriented approach has been developed for the implementation of the above-mentioned front tracking techniques. The fundamentals of object oriented C++ programming for scientific computations are reviewed in Reference 18. Object oriented calculations have received extensive attention in computational mathematics and, in particular, in fluid dynamics.

This paper provides a number of specific ideas for the development of classes that are appropriate for the analysis of solidification processes. The development and dynamic coupling of the thermal, flow, mass diffusion and phase change simulators is easily performed using classes, virtual functions and smart pointers. It is shown that such an environment is appropriate for the analysis of processes with many transport mechanisms for each of which individual simulators exist in the form of classes. To allow ourselves to concentrate on the development of solidification processing related classes, we have selected to adapt our computations to the widely used diffpack library of C++ classes. The diffpack libraries include a variety of classes for linear algebra operations, a number
of FEM-based classes for element definition, stiffness development, assembly and many other.\textsuperscript{29--31} In addition, \textit{diffpack} provides a secure way for pointer declarations as well as a framework for the development of \textit{fields} to represent the various continuous variables.\textsuperscript{31} Familiarity with the structure of the \textit{diffpack} libraries is useful but not essential in understanding the present developments. Most of the discussion here is general and the present algorithms and techniques can be implemented using other object oriented C++ libraries.

The structure of this paper is as follows. At first, the governing equations of alloy solidification are presented followed by the corresponding weak problems and their discrete counterparts. The paper proceeds with the time-integration algorithm and the coupling of the various subproblems corresponding to each physical mechanism. Details related to the object oriented implementation are then presented together with the geometric modelling of the freezing front and the calculation of the mesh motion. Various solidification problems are considered to demonstrate the versatility of the proposed simulator. These example problems include a three-dimensional simulation of solidification of a pure material and the development of double diffusive convection patterns during directional solidification of a binary alloy as studied numerically and experimentally in References 32--34. It is shown that front-tracking methods are accurate and efficient techniques for the analysis of directional solidification processes of pure metals and of certain binary alloy systems.

2. DEFINITION OF THE DIRECT SOLIDIFICATION PROBLEM

The physical problem to be analysed can be stated as follows: a binary alloy melt at initial temperature \( T_i(x,0) \) and initial uniform concentration \( c_i \) is assumed to occupy a region \( \Omega_0 \) with mass impermeable mold boundary \( \Gamma \). At time \( t = 0^+ \), a part of the boundary \( \Gamma \) is cooled down to a temperature lower than the melting temperature corresponding to \( c_i \). Only directional solidification is considered in this work. Let us denote the solid region as \( \Omega_s \) and the liquid region as \( \Omega_l \). These regions share the common interface boundary \( \Gamma_I(t) \) whose normal vector \( n \) is defined pointing outside the solid region. The region \( \Omega_l \) has a boundary \( \Gamma_l \) which includes \( \Gamma_l \) and similarly the region \( \Omega_s \) has a boundary \( \Gamma_s \) which includes \( \Gamma_s \). The regions \( \Omega_l \) and \( \Omega_s \) are time dependent and the interface \( \Gamma_I \) is moving with velocity \( v_I \). The subscripts \( s \) and \( l \) are used to denote quantities for the solid and liquid phases, respectively, while the subscripts \( I \) or \( f \) are used to denote the freezing interface (moving front). Figure 1 provides a schematic of the solidification problem definition.

We make the following assumptions about the transport of heat, solute and momentum in the solidification system:

1. Constant thermo-physical and transport properties, including thermal and solute diffusivities \( z \) and \( D \), respectively, viscosity \( \nu \), density \( \rho \), thermal conductivity \( k \) and phase change latent heat \( L \).
2. The melt flow is assumed to be a laminar flow caused by temperature-induced density variations.
3. The solute diffusion in the solid is negligible compared to that of the liquid, i.e. \( D_s / D_l \to 0 \).
4. A macroscopically stable sharp interface \( \Gamma_1 \) exists between the solid and the liquid regions.

For simplicity, we assume that heat flux and/or mixed thermal conditions are applied on the outside mold walls \( \Gamma_0(t) \) and \( \Gamma_0(t) \), corresponding to the solid \( \Omega_s(t) \) and liquid \( \Omega_l(t) \) domains, respectively. In particular, we define \( \Gamma_0(t) = \Gamma_0^s(t) \cup \Gamma_0^l(t) \), where the superscript \( m \) denotes the part
of the boundary with mixed thermal boundary condition, whereas the superscript \( h \) denotes the part of the boundary with a thermal flux condition. Similarly, we can write that \( \Gamma_{0l}(t) = \Gamma_{0l}^h(t) \cup \Gamma_{0l}^m(t) \) (see Figure 1).

Following standard practice, we introduce the following dimensionless variables:

\[
\hat{x} = \frac{x}{L}, \quad \hat{t} = \frac{\alpha t}{L^2}, \quad \hat{T} = \frac{T - T_0}{\Delta T_0},
\]

\[
\hat{c} = \frac{c - c_0}{\Delta c_0}, \quad \hat{v} = \frac{v_L}{\alpha t}, \quad \hat{p} = \frac{pL^2}{\rho \alpha x_t^2}
\]

The reference temperature \( T_0 \) and reference temperature difference \( \Delta T_0 \) will be defined later for each example problem examined. The governing non-dimensional parameters are the thermal Rayleigh number \( Ra_T \), the solutal Rayleigh number \( Ra_c \), the Lewis number \( Le \), the Stefan number \( Ste \), the Biot number \( Bi \) and the Prandtl number \( Pr \) (see Table I for their definition). Other non-dimensional variables include \( R_k \), the ratio of thermal diffusivities of the solid and liquid, \( R_k \), the ratio of thermal conductivities of the solid and liquid and \( \delta \) the ratio of the reference concentration \( c_0 \) (\( \equiv c_i \)) and the reference concentration difference \( \Delta c_0 \). Also, the dimensionless form of \( m \), the slope of the liquidus line becomes, \( \hat{m} = m\Delta c_0/\Delta T_0 \) and the dimensionless form of the melting point of the pure material \( \hat{T}_m \) is defined as \( \hat{T}_m = (T_m - T_0 - mc_0)/\Delta T_0 \).

With the above assumptions and notation, the governing equations for the temperature \( T \), concentration \( c \), melt velocity field \( \mathbf{v} \) and the interface velocity \( \mathbf{v}_f \) in non-dimensional form, are summarized in Box 1. In the equations of Box 1, \( \kappa \) denotes the partition coefficient and \( \mathbf{e}_y \) is the unit vector in the direction of gravity. For simplicity, the hats indicating dimensionless quantities are not shown in Box 1, and from now on, unless it is stated otherwise, all quantities will be assumed to be in a non-dimensional form. In summary, our task is to solve the initial-boundary-value problem of Box 1 and determine the solute distribution, flow and temperature histories on \( \Omega_t \cup \Omega_l = \Omega_0 \) as well as the solid–liquid interface velocity and location histories.

**Remark 1.** The model of Box 1 is the simplest possible model that maintains most of the essential features of binary alloy solidification. Due to the assumption of a stable planar interface,
Table I. Definition of non-dimensional parameters in the binary alloy solidification problem. Here, $\beta_T$ and $\beta_c$ are the thermal and solutal expansion coefficients, respectively, and $|g|$ is the gravity constant.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thermal Rayleigh number</td>
<td>$Ra_T = (\beta_T</td>
</tr>
<tr>
<td>Solutal Rayleigh number</td>
<td>$Ra_c = (\beta_c</td>
</tr>
<tr>
<td>Lewis number</td>
<td>$Le = \alpha / D_L$</td>
</tr>
<tr>
<td>Stefan number</td>
<td>$Ste = (c_0 \Delta T_0) / L_H$</td>
</tr>
<tr>
<td>Prandtl number</td>
<td>$Pr = v / \alpha_0$</td>
</tr>
<tr>
<td>Biot number</td>
<td>$Bi = hL / k$</td>
</tr>
</tbody>
</table>

it is restricted to alloys with a thin mushy layer and to crystal growth processes with low growth rate. For arbitrary boundary heating/cooling conditions it can be shown numerically and experimentally that the above model may lead to unstable freezing interface conditions. Such stability issues were addressed in Reference 35 where the model of Box 1 was used to design the mold cooling/heating boundary conditions such that a stable growth results with a desired growth velocity.

3. MOVING FINITE ELEMENT FORMULATION

Let us divide the region $\Omega_s \cup \Omega_l = \Omega_0$ into $E_s + E_l = E$ finite elements. Denote $M_s$, $M_l$ and $M_I$ the number of nodes in the solid phase, liquid phase and on the moving interface, respectively. The $E_s$ elements of the solid phase are assumed to share the same nodes on the solid–liquid interface boundary $\Gamma_3(t)$ with the $E_l$ elements of the liquid phase.

Since the finite element nodes are moving with time, the shape functions implicitly depend on time through the nodal co-ordinates. Thus, the temperature field (and similarly any other field) can be approximated within each finite element as

$$T(x, y, t) = T_j(t) N_j(x, y, t)$$

where from now on summation is implied on repeated indices. The basis functions, $N_j$, are functions of both space and time, because they are attached to the mesh which moves in time. It can be shown that,

$$\frac{\partial N_j}{\partial t} = -\mathbf{v} \cdot \nabla N_j$$

where the mesh velocity $\mathbf{V}$ is calculated by interpolating the nodal velocities $d\mathbf{x}_j/dt$, where $\mathbf{x}_j$ represents the co-ordinates of a node with respect to a fixed frame of reference. We will later discuss how these nodal velocities can be evaluated using the freezing interface velocity $v_f$.

Note that as a result of the deforming FEM formulation (equations (1) and (2)), an extra convection term appears here in the discrete forms of all the governing partial differential equations. This apparent convection is due to the movement of the mesh and highlights the fact that we are analysing the problem through a co-ordinate system implicitly attached to the mesh.
Box 1. Non-dimensional form of the governing equations for the solidification of a binary alloy

**Liquid region:** \((x, t) \in \Omega_0(t) \times [0, t_{\text{max}}]\)

\[
\begin{align*}
\frac{\partial T}{\partial t}(x, t) + \mathbf{v}(x, t) \cdot \nabla T(x, t) &= \nabla^2 T(x, t), \quad x \in \Omega_0(t) \\
\frac{\partial c}{\partial t}(x, t) + \mathbf{v}(x, t) \cdot \nabla c(x, t) &= L e^{-1} \nabla^2 c(x, t), \quad x \in \Omega_0(t) \\
\frac{\partial \mathbf{v}}{\partial t}(x, t) + \mathbf{v}(x, t) \cdot \nabla \mathbf{v}(x, t) &= Pr \nabla^2 \mathbf{v}(x, t) - Pr Ra_T T(x, t) e_g \\
&\quad + Pr Ra_e c(x, t) e_g - \nabla p(x, t), \quad x \in \Omega_0(t) \\
\nabla \cdot \mathbf{v}(x, t) &= 0, \quad x \in \Omega_0(t) \\
T(x, 0) &= T_i, \quad c(x, 0) = c_i, \quad \mathbf{v}(x, 0) = 0, \quad x \in \Omega_0(0) \\
\mathbf{v}(x, t) &= 0, \quad x \in \Gamma_0(t), \\
\frac{\partial T}{\partial n}(x, t) &= q_{00}(x, t), \quad x \in \Gamma_{00}^l(t) \\
\frac{\partial T}{\partial n}(x, t) &= Bi(T_{\infty} - T(x, t)), \quad x \in \Gamma_{00}^m(t)
\end{align*}
\]

**Solid region:** \((x, t) \in \Omega_s(t) \times [0, t_{\text{max}}]\)

\[
\begin{align*}
\frac{\partial T}{\partial t}(x, t) &= R_s \nabla^2 T(x, t), \quad x \in \Omega_s(t) \\
\frac{\partial T}{\partial n}(x, t) &= q_{0s}^h(x, t), \quad x \in \Gamma_{0s}^l(t) \\
\frac{\partial T}{\partial n}(x, t) &= Bi(T_{\infty} - T(x, t)), \quad x \in \Gamma_{0s}^m(t)
\end{align*}
\]

**Freezing interface:** \((x, t) \in \Gamma_1(t) \times [0, t_{\text{max}}]\)

\[
\begin{align*}
R_k \frac{\partial T_0}{\partial n}(x, t) - \frac{\partial T_1}{\partial n}(x, t) &= Ste^{-1} \mathbf{v}_f(x, t) \cdot \mathbf{n} \\
\frac{\partial c}{\partial n}(x, t) &= Le(\kappa - 1) \mathbf{v}_f(x, t) \cdot \mathbf{n} \left(c(x, t) + \delta\right) \\
T(x, t) &= T_m - mc(x, t)
\end{align*}
\]

3.1. Discrete finite element equations

The derivation and notation for the discrete streamline upwind-Petrov–Galerkin based finite element equations are similar in nature to those given in Reference 36 for the incompressible Navier–Stokes equations and they will not be repeated here. The incompressibility condition is implemented using the penalty approach (here \(\lambda\) denotes the penalty number). Further details and examples demonstrating the accuracy of such formulations can also be found in References 25–27, 37 and 38.
We finally arrive at the following discrete equations:

**Momentum equation:**

\[ M \ddot{v} + C'\dot{v} + C''v + N(v)v + E(T) = 0 \] (18)

where \( v \) and \( a \) are the vectors containing the velocity and acceleration degrees of freedom, respectively, and \( T \) is the vector containing the temperature degrees of freedom. The matrices \( M, C', C'', N(v) \) and the vector \( E(T) \) are given as follows:

\[
M_{pq} = \sum_{e=1}^{ne} \left\{ \int_{\Omega_e} \left( N_e^c + p_e^c \right) N_{p,j}^e \delta_{ij} \, d\Omega \right\}
\] (19)

\[
C'_{pq} = \lambda \left\{ \int_{\Omega_e} N_e^c N_{p,j}^e \, d\Omega \right\}
\] (20)

\[
C''_{pq} = Pr \left\{ \int_{\Omega_e} N_{x,k}^c N_{p,k}^e \delta_{ij} \, d\Omega \right\}
\] (21)

\[
N(v)_{pq} = \sum_{e=1}^{ne} \left\{ \int_{\Omega_e} \left( N_e^c + p_e^c \right) (v_e^k - V_e^k) N_{p,k}^e \delta_{ij} \, d\Omega \right\}
\] (22)

\[
E(T)_{p} = Pr RaT \sum_{e=1}^{ne} \left\{ \int_{\Omega_e} \left( N_e^c + p_e^c \right) \left( T - \frac{Ra_e}{RaT} \right) \cos \theta_j \, d\Omega \right\}
\] (23)

where \( \dot{p} \) refers to the global velocity degree of freedom in the final system of equations that corresponds to the \( i \)th velocity degree of freedom of the elemental node \( x (i = 1, \ldots, n_{ad}) \) in the \( e \)th element. Similarly, one can define \( \dot{q} \). Here, \( p_e^c \) denotes the discontinuous contributions to the Petrov–Galerkin weights. Also, note that \( \theta_j \) is the angle between the \( j \)th co-ordinate axis and the gravity vector.

**Energy equation:**

\[
MT' + KT = -F(v)T
\] (24)

where the dot denotes time derivatives and \( T \) are the temperature degrees of freedom. The same notation is used as in the discrete momentum equation in order to emphasize the similar structure of the two subproblems. The matrices \( M, K, F(v) \) are here defined as follows:

\[
M_{pq} = \sum_{e=1}^{ne} \left\{ \int_{\Omega_e} \left( N_e^c + p_e^c \right) N_{p,j}^e \, d\Omega \right\}
\] (25)

\[
K_{pq} = \sum_{e=1}^{ne} \left\{ \int_{\Omega_e} N_{x,k}^c N_{p,k}^e \, d\Omega \right\}
\] (26)

\[
F(v)_{pq} = \sum_{e=1}^{ne} \left\{ \int_{\Omega_e} \left( N_e^c + p_e^c \right) (v_e^k - V_e^k) N_{p,k}^e \, d\Omega \right\}
\] (27)

where \( p \) and \( q \) refer to the global degrees of freedom in equation (24).

**Solute conservation equation:**

\[
Mc + Ke = -F(v)c
\] (28)
where \( c \) are the concentration degrees of freedom. The matrices \( M, K, F(v) \) have a form similar to that given by equations (25)–(27).

**Remark 2.** For the case of double diffusive convection, the body force term \( E(T) \hat{p} \) (equation (23)) was implemented using \( N^e_\text{pe} \) as weight (i.e. the discontinuous functions \( p^e_\text{pe} \) were not used here to produce a consistent Petrov–Galerkin formulation). This change was first discussed by Heinrich.\(^{38}\) Even though no theoretical justification for the need of such an inconsistent Petrov–Galerkin formulation is as yet known, with an extensive number of tests, we found that such a formulation produces accurate results, whereas the consistent Petrov–Galerkin formulation (i.e. using all terms in equation (23)) was shown to exhibit unwanted oscillations and for many cases not to converge.

### 3.2. Calculation of the freezing front motion

Using energy conservation arguments, a weak form of the Stefan condition is introduced to calculate the freezing front motion.\(^{9,15,16}\) Let us consider that the boundary \( \Gamma_1 \) consists of \( E \) boundary segments. Let \( N^\text{int} \) be the total number of nodes along \( \Gamma_1(t) \) with \( N^e_\text{int} \) the number of nodes on the interface segment \( e \). The interface velocity is now interpolated over each segment with shape functions \( \Psi^\beta \) (e.g. one-dimensional shape functions in 2-D problems) as follows:

\[
V_i = V_i^\beta \Psi^\beta, \quad i = 1, \ldots, N^\text{int}, \quad \beta = 1, \ldots, N^e_\text{int}
\]  

(29)

The following weak form of the Stefan condition is introduced:

\[
\sum_{e=1}^{E_1} \int_{\Gamma_1^e} W^\gamma \left( R_k \frac{\partial T_s}{\partial n} - \frac{\partial T_1}{\partial n} \right) \, d\Gamma_e = \sum_{e=1}^{E_1} \int_{\Gamma_1^e} W^\gamma \left( R_k \frac{\partial T_s}{\partial n} - \frac{\partial T_L}{\partial n} \right) \, d\Gamma_e
\]  

(30)

where \( W^\gamma \) is the Petrov–Galerkin shape function, \((N^e_\text{pe} + p^e_\text{pe})\), on the boundary \( \Gamma_1 \). By substituting the discretized form of the interface velocity, we have the following:

\[
\sum_{e=1}^{E_1} \int_{\Gamma_1^e} W^\gamma \Psi^\beta \left( R_k \frac{\partial T_s}{\partial n} - \frac{\partial T_L}{\partial n} \right) \, d\Gamma_e = \sum_{e=1}^{E_1} \int_{\Gamma_1^e} W^\gamma \left( R_k \frac{\partial T_s}{\partial n} - \frac{\partial T_L}{\partial n} \right) \, d\Gamma_e
\]  

(31)

where \( \hat{n}_i^e \) is the \( i \)th component of the unit normal vector to the \( e \)th boundary segment along \( \Gamma_1(t) \).

The above equation provides a linear system of equations for the calculation of the nodal velocities after its right-hand side has been evaluated. The calculation of the weighted fluxes on the right-hand side of equation (31) can be performed using the discretized energy equations for each phase once the temperature and temperature-rate fields have been calculated. Let us, for example, return to the energy equation (24) in the solid phase and consider its form for the freezing interface nodes where an essential boundary condition was applied in the solution process (see Section 4). Then, the first term on the right-hand side of equation (31) can be evaluated from the equation:

\[
Q_s = M \dot{T} + KT + F(v)\Gamma
\]  

(32)

The matrices \( M, K \) and \( F(v) \) have already been defined in equations (25)–(27) and the above equation is here restricted to the freezing interface nodes. \( Q_s \) is the flux contribution given by

\[
Q_s = \sum_{e=1}^{E_1} \int_{\Gamma_1^e} R_k W^\gamma \frac{\partial T_s}{\partial n} \, d\Gamma_e
\]  

(33)
Similarly, one can calculate the second term on the right-hand side of equation (31). Finally, the right-hand side of equation (31) can be written as

$$\sum_{e=1}^{E} W_e \left( R_s \frac{\partial T_s}{\partial n} - R_L \frac{\partial T_L}{\partial n} \right) d\Gamma_e = \frac{R_s}{R_L} Q_s + Q_l$$

(34)

Remark 3. The Petrov–Galerkin weighting function $W$ depends on the diffusive constant of the equation to which it is applied. Since the diffusive constant is not the same for the solid and liquid domains ($R_s$ and unity, respectively), the corresponding weighting functions in the energy weak forms for each phase are different. This will generally result in fluxes $Q_s$ and $Q_l$ that cannot simultaneously be consistent with the right-hand side of equation (31). This problem is avoided for physical situations for which $R_s = 1$. For the case of $R_s \neq 1$, a selection of $W$ in equation (31) is recommended in which the discontinuous contribution $p_e^c$ (see Reference 36) is calculated using an average diffusive constant.

Remark 4. The energy balance on the interface defines only the normal component of the interface velocity. In this paper, we consider directional solidification processes. For such processes, we assume that the freezing front velocity is parallel to an a priori specified direction of solidification. This implies that at each interface node, in addition to the correct normal to the interface velocity component, we also introduce a tangential nodal velocity component such that the total nodal interface velocity is in the direction of solidification.

4. THE TIME-STEPPING PROCEDURE

The semi-discrete system of equations for the temperature field in the solid domain and the temperature, concentration and flow fields in the liquid domain, are summarized below. Note that we intentionally keep the notation similar for each of these subproblems in order to emphasize their common mathematical structure. No notation inconsistencies appear as the time integration is performed separately for each problem.

Liquid region

Momentum equation:

$$Ma + C'v + C''v + N(v)v + E(T) = 0$$

(35)

Energy equation:

$$M\dot{T} + KT = -F(v)T$$

(36)

Solute conservation equation:

$$M\dot{c} + Kc = -F(v)c$$

(37)

Solid region

Energy equation:

$$M\dot{T} + K'T = -F'T$$

(38)
The algorithm used to solve this coupled system of equations along with the Stefan condition and solute conservation condition at the interface is similar to the predictor–multicorrector algorithm proposed in Reference 36 for the incompressible Navier–Stokes equations. The incremental solution procedure at time $t_{n+1}$ is as follows:

**Front prediction at time $t_{n+1}$:**

\[
\begin{align*}
V^{(0)}_{n+1} &= V_n \\
\mathbf{p}^{(0)}_{n+1} &= \mathbf{p}_n + \Delta t[(1 - \gamma)V_n + \gamma V^{(0)}_{n+1}]
\end{align*}
\]

**Main solution step:**

**Mesh motion:**

Using the obtained interface velocity and a transfinite mapping technique, update the solid and liquid domain grids and calculate the corresponding mesh motion. More on the interface modelling will be provided in the next section.

**Melt concentration (solve for $c_{n+1}$ with given front velocity, melt flow and mesh velocity):**

1. **Predictor step:**

\[
\begin{align*}
\mathbf{c}^{(0)}_{n+1} &= \mathbf{c}_n + \Delta t(1 - \gamma)\dot{\mathbf{c}}_n \\
\dot{\mathbf{c}}^{(0)}_{n+1} &= 0
\end{align*}
\]

2. Solve $\mathbf{M}^* \Delta \mathbf{c}^{(i)}_{n+1} = \mathbf{R}^{(i)}_{n+1}$ where $\mathbf{M}^* = \mathbf{M} + \gamma \Delta t \mathbf{K}$ and

\[
\mathbf{R}^{(i)}_{n+1} = \sum_{e=1}^{n_e} \left( -f(v)_{n+1} \mathbf{c}^{(i)}_{n+1} - m^e \mathbf{c}^{(i)}_{n+1} - k^e \mathbf{c}^{(i)}_{n+1} \right)
\]

Note that the interface solute conservation condition serves as a mixed boundary condition for the melt concentration solver (with the velocity of the front known).

3. **Corrector step:**

\[
\begin{align*}
\mathbf{c}^{(i+1)}_{n+1} &= \mathbf{c}^{(i)}_{n+1} + \Delta t \gamma \Delta \mathbf{c}^{(i)}_{n+1} \\
\dot{\mathbf{c}}^{(i+1)}_{n+1} &= \dot{\mathbf{c}}^{(i)}_{n+1} + \Delta \mathbf{c}^{(i)}_{n+1}
\end{align*}
\]

4. Repeat steps 2 and 3 until convergence in $\mathbf{c}_{n+1}$ is achieved.

**Melt temperature (solve for $T_{n+1}$ with given front velocity, melt flow and mesh velocity):**

1. **Predictor step:**

\[
\begin{align*}
T^{(0)}_{n+1} &= T_n + \Delta t(1 - \gamma)\dot{T}_{n} \\
\dot{T}^{(0)}_{n+1} &= 0
\end{align*}
\]

2. Solve $\mathbf{M}^* \Delta T^{(i)}_{n+1} = \mathbf{R}^{(i)}_{n+1}$ where $\mathbf{M}^* = \mathbf{M} + \gamma \Delta t \mathbf{K}$ and

\[
\begin{align*}
\mathbf{R}^{(i)}_{n+1} &= \sum_{e=1}^{n_e} \left( -f(v)_{n+1} T^{(i)}_{e,n+1} - m^e T^{(i)}_{e,n+1} - k^e T^{(i)}_{e,n+1} \right)
\end{align*}
\]

Note that the interface equilibrium condition $T_i = T_m - \mathbf{m}e$ is prescribed as an essential boundary condition using the concentration field obtained from the concentration solver. Also,
since the interface boundary condition is a time-dependent boundary condition, it is required
to impose another essential boundary condition on the time derivatives, i.e. $T_s = -\dot{m}c$.

3. Corrector step:

$$T_{ln+1}^{(i+1)} = T_{ln+1}^i + \Delta t' \Delta T_{ln+1}^i$$
$$T_{ln+1}^{(i+1)} = T_{ln+1}^{(i)} + \Delta T_{ln+1}^{(i)}$$

4. Repeat steps 2 and 3 until convergence in $T_{ln+1}$ is achieved.

Melt velocity (solve for $v_{n+1}$ and $a_{n+1}$ with given front velocity, melt temperature, concentration
field and mesh velocity):

1. Predictor step:

$$v^{(0)}_{n+1} = v_n + \Delta t(1 - \gamma)a_n$$
$$a^{(0)}_{n+1} = 0$$

2. Solve $M^* \Delta a_{n+1} = R^{(i)}_{n+1}$ where $M^* = M + \gamma \Delta t(C' + C'')$ and

$$R^{(i)}_{n+1} = \frac{n_d}{e=1} (-e(T)_{h+1} - m^* a_{n+1}^{(e)} - (c' + c'' + n(v)_{e})v_{n+1}^{(e)})$$

The no-slip condition is used as a boundary condition for the solution process.

3. Corrector step:

$$v^{(i+1)}_{n+1} = v_{n+1}^{(i)} + \Delta t' \Delta a_{n+1}^{(i)}$$
$$a^{(i+1)}_{n+1} = a_{n+1}^{(i)} + \Delta a_{n+1}^{(i)}$$

4. Repeat steps 2 and 3 until convergence in $v_{n+1}$ is achieved.

Solid temperature (solve for $T_{sn+1}$ with given front and mesh velocities):

1. Predictor step:

$$T^{(0)}_{sn+1} = T_{sn} + \Delta t(1 - \gamma)T_{sn}$$
$$T^{(0)}_{sn+1} = 0$$

2. Solve $M^* \Delta T_{sn+1}^{(i)} = R^{(i)}_{n+1}$ where $M^* = M + \gamma \Delta tK'$ and

$$R^{(i)}_{n+1} = \frac{n_d}{e=1} (-f'(v)_{h+1} T_{sn}^{(i)} - m^* T_{sn+1}^{(i)} - k^* T_{sn+1}^{(i)})$$

The interface equilibrium condition $T_s = T_m - mc$ is prescribed as an essential boundary
condition using the concentration field obtained from the solution of the concentration solver.

Since the interface boundary condition is a time dependent boundary condition, we also
impose another essential boundary condition on the time derivatives, i.e. $T_s = -\dot{m}c$.

3. Corrector step:

$$T^{(i+1)}_{sn+1} = T_{sn+1}^{(i)} + \Delta t' \Delta T_{sn+1}^{(i)}$$
$$T^{(i+1)}_{sn+1} = T_{sn+1}^{(i)} + \Delta T_{sn+1}^{(i)}$$

4. Repeat steps 2 and 3 until convergence in $T_{sn+1}$ is achieved.
Figure 2. Flow chart describing the time-stepping procedure

Front update

1. Solve the Stefan condition using $T_{s,i,n+1}$ and $T_{l,i,n+1}$ to obtain $V_{n+1}^{i+1}$.

2. Update the front location as follows: $P_{n+1}^{i+1} = P_{n+1}^{i} + \Delta t [V_{n+1}^{i+1} - V_{n+1}^{i}]$.

3. If the error in the calculated front velocity $V$ is less than a specified tolerance, then update the various fields and move to the next time step. Else, repeat the main solution step.

The time-stepping algorithm is sketched in a flow-chart form in Figure 2.

Remark 5. The front velocity calculation via the Stefan condition is only performed in the solution phase of the algorithm, i.e. using the corrected temperature fields in the solid and liquid phases. Update of the front using the predicted temperature fields led to an oscillatory freezing front and to the non-convergence of the algorithm. It is finally noted that the proposed sequence of solution of the various subproblems was essential for the convergence of the overall algorithm.

5. NUMERICAL IMPLEMENTATION

Solidification and melting are typical examples of processes in which several transport mechanisms are simultaneously present. Such coupled mechanisms include heat conduction and convection, mass diffusion and melt flow. In addition, the solid/liquid phase transformation must also be considered.

Each of the above-mentioned physical mechanisms can be thought of as a separate subproblem that can be analysed and numerically formulated independently of the other subproblems. For example, the melt convection problem can be numerically implemented and solved to calculate the flow field assuming that the temperature and solute fields as well as the freezing front and mesh motions are known.

Object oriented programming provides an ideal framework for the finite element implementation of solidification processes. Each transport mechanism (e.g. the melt flow subproblem) can be represented with an appropriate class that contains all data and attributes of the particular mechanism including the algorithms necessary for its numerical solution. The simulators (solvers) for such subproblems can be developed and tested independently of each other. Software engineering provides us with techniques to combine the individual simulators of the various subproblems in order to analyse and test a number of phase change solidification algorithms. For example, the temporal integration scheme presented in the previous section is one example of a solidification algorithm that combines the time integration algorithms corresponding to the discrete temporal equations of each subproblem.

To allow ourselves to concentrate only on the implementation of phase change related classes, we have selected to base our computations on the widely used Diffpack library of classes for analysis of systems governed by partial differential equations. Most of the present developments are general and other object oriented frameworks can also be considered. A brief review of Diffpack is given next followed by a presentation of the classes developed to model melting and solidification processes.

5.1. Diffpack: an object oriented framework for the analysis of continuum systems

Only the very basic structure of Diffpack will be reviewed here and the interested reader is advised to consult References 29–31.

Pointers are treated in Diffpack using the class ‘HandleID’. This class allows the automatic deletion of a pointer only when all objects of various classes that are binded to this pointer have been deleted. This is important when more than one objects have been binded to a given pointer.

The class ‘FiniteElement’ provides the framework for all basic finite element calculations like evaluation of shape functions and their derivatives at Gauss points, and other. The stiffness and load calculations and the assembly of the linear system of equations (e.g. in a Newton analysis of a non-linear system of equations) is performed using the class ‘FEM’. The particular problem-dependent calculations are defined in a simulator class derived from ‘FEM’. For example, the main stiffness and load calculations at a particular Gauss point are performed using the pure virtual function ‘FEM:integrands()’. As such, Diffpack provides the general structure for the development of solvers for continuum systems. The use of class hierarchies and virtual functions is essential in such an implementation.

The class ‘ElmItgRules’ has also been implemented to allow a selection by the user of full, uniformly reduced, or selective reduced integration rules. For problems with selective reduced integration, one may define derived classes of ‘ElmMatVecCalcStd’ and ‘IntegrandCalc’ and implement their virtual functions ‘calcElmMatVec’ and ‘integrands’. The pure virtual function ‘calcElmMatVec’ accepts as argument an ‘IntegrandCalc’ object. This allows the definition of ‘calcElmMatVec’ in a derived class such that the contributions to the element stiffness from both reduced and fully integrated terms are evaluated. Such techniques have been found to be useful for the modelling of incompressible flows using the consistent penalty method.
Continuous fields are treated in \texttt{diffpack} using the classes ‘FieldFE’ (for scalar fields) and ‘FieldsFE’ (for vector fields). An object of these classes is binded with a particular finite element grid (object of class ‘GridFE’). The functionality of the class ‘FieldFE’ allows the calculation of the nodal values of the field, the calculation of the field at any point within the domain, re-binding of the field with another grid, etc. Also, note that each ‘GridFE’ object contains geometric information as well as information specific to a given boundary-value problem (e.g. problem specific boundary indicators).

The classes ‘LinEqAdm’ and ‘NonLinEqSolver’ provide the necessary structure for the selection of various linear and non-linear algebraic equation solvers. The user via a menu (class ‘MenuUDC’) is allowed to select from a list of available methods or to implement his own solvers. The \texttt{diffpack} class ‘Ptv(real)’ provides an efficient way to define ‘point’ objects. Finally, basic time management as required by time-integration algorithms is performed using the class ‘TimePrm’.

5.2. Class development for each transport process

In this section, a brief review of the various classes developed to represent the melt flow, solute diffusion and heat transfer processes is given. The flow solver is implemented as a class ‘Navier–Stokes’ (see Boxes 2a and b). This class provides the functionality necessary for the calculation of all terms in the discrete equation (18) and the temporal integration algorithm for the Navier–Stokes equations as given in Section 4.

As discussed in Section 5.1, the selective reduced integration scheme is implemented by introducing the classes ‘ReducedIntg:IntegrandCalc’, ‘NonReducedIntg:IntegrandCalc’, and ‘ElmMatVecPenalty: ElmMatVecCalcStd’. The virtual function ‘solveAtThisTimeLevel()’ performs the calculation of the melt flow ‘v’ and acceleration ‘a’ at the end of a time step, given the melt flow ‘v\_prev’ and acceleration ‘a\_prev’ at the beginning of the step, and assuming that the mesh motion ‘V’, temperature field ‘T’ and concentration field ‘c’ are known. This function invokes the virtual function ‘makeSystem’ (with an ‘ElmMatVecCalcStd’ object as one of its arguments) in order to produce the linear system of equations at each iteration within a time step. More details on the class ‘NavierStokes’ can be found in References 25 and 26.

The heat transfer in the melt has been modelled with the class ‘NIHeat1’ (see Boxes 3a and b). This simulator contains as members the melt flow field ‘v’ as well as the mesh velocity ‘V’. The virtual function ‘solveAtThisTimeLevel()’ performs the calculation of the melt temperature ‘T’ and temperature rate ‘T\_dot’ at the end of a time step, given the melt temperature ‘T\_prev’ and temperature rate ‘T\_dot\_prev’ at the beginning of the step, and assuming that the mesh motion ‘V’ and melt flow field ‘v’ are known.

In addition to the temperature field, the class ‘NIHeat1’ provides the necessary functionality for the calculation of the heat fluxes at the interface nodes as well as for the calculation of the projection points (‘VecSimple(Ptv(real)) X’) of the interface nodes on a plane normal to the solidification direction. These projection points are useful for the calculation of a spline fit to the freezing interface (see Section 5.3).

Finally, it is noted that the interface nodes are sorted once in the ‘scan’ function of ‘NIHeat1’, and their connectivities to the liquid grid are calculated (‘VecSort(int) node\_number’). Similarly, the interface nodal connectivity to the solid grid is calculated in class ‘NIHeat2’.

The calculation of the term $Q_s$ is performed using equation (32) for the interface nodes. The matrices $M$, $K$ and the vector $F(v)$ are assembled over the finite elements that have one of their
Box 2a. The main members of the class ‘NavierStokes’ that is used to model the melt flow

```cpp
class NavierStokes: public FEM, public MenuUDC, public Store4Plotting {
protected:
    Handle(DegFreeFE) dof; // Mapping: nodal values ← → linear system
    Handle(GridFE) grid; // Finite element mesh
    Handle(FieldFE) T; // Known temperature field
    Handle(FieldFE) c; // Known concentration field
    Handle(FieldFE) v; // Velocity field
    Handle(FieldFE) V; // Known mesh velocity field
    Handle(FieldFE) a; // Acceleration field
    Handle(FieldFE) v_prev; // Known velocity field at the previous time step
    Handle(FieldFE) a_prev; // Known acceleration at the previous time step
    UpwindFE PG; // Petrov–Galerkin weights
    Handle(TimePrm) tip; // Time discretization parameters
    Vec(real) nonlin_solution1; // Velocity at previous step (scratch vector)
    Vec(real) nonlin_solnext1; // Velocity at present step (scratch vector)
    Vec(real) nonlin_solution2; // Acceleration at previous step (scratch vector)
    Vec(real) nonlin_solnext2; // Acceleration at next step (scratch vector)
    Vec(real) linear_solution; // Solution of linear subsystem
    LinEqAdm lineq; // Interface to linear solvers
    real lambda; // Penalty parameter
    real mu; // Dynamic viscosity
    real density; // Density
    real buoyancy; // Constant for the buoyancy term
    Ptv(real) theta; // Directional cosines for gravity
    real gamma; // Time integration parameter
};
```

sides on the freezing front. These calculations are performed with member functions of the class ‘HeatFluxCalc’. Further details on the class ‘NlHeat1’ are given in Section 5.3 and in Reference 26.

The heat transfer problem in the solid and the solute diffusion problem have basically the same structure as the melt heat transfer problem (both fall under the category of scalar convection–diffusion problems). As such, the presentation of the classes ‘NlHeat2’ and ‘Concentration’ introduced to model the heat conduction process in the solid phase and the solute diffusion in the melt, respectively, will not be given here.

5.3. Modelling of the freezing interface

As mentioned earlier, a tangential front velocity is superimposed on the normal velocity predicted by equation (31) so that the interface nodes move in the direction of solidification. The right-hand side of equation (31) contains the terms $Q_s$ and $Q_l$ (see also equation (34)) that are calculated
Box 2b. The main member functions of the class ‘NavierStokes’

```c++
class NavierStokes: public FEM, public MenuUDC, public Store4Plotting {
    friend class Solidification;
    friend class NonReducedIntg;
    friend class ReducedIntg;
    friend class ElmMatVecPenalty;
    protected:
        ElmMatVecPenalty elmv; // Object for administering the penalty term
        void setIC();
        void timeLoop();
        void solveAtThisTimeLevel();
        void storeResults();
        void updateDataStructures();
    ..
    public:
        NavierStokes ();
        ~NavierStokes ();
        void define(MenuSystem& menu, int level = MAIN);
        void scan(MenuSystem& menu);
        void adm(MenuSystem& menu);
        void driver();
        virtual void fillEssBC();
    ..
};
```

from the discretized energy equations in the solid and liquid phases, respectively. Considering that two different (but compatible) grids are used to discretize the solid and liquid domains, an efficient scheme must be developed to relate the interface nodes to the corresponding nodes in the liquid and solid domains. Such a scheme must be geometry independent so that various directional solidification problems could be analysed with the same simulator.

In the present work, the interface grid is modelled as a grid of scattered points. The `diffpack` class ‘GridScatPt’ is used. In particular, the point projections of the interface nodes on a plane normal to the solidification direction are treated as a grid of scattered points. To each projection point corresponds a unique location of the interface in the direction of solidification.

The various variables on the interface are defined as fields over scattered points (class ‘FieldScatPt’). This allows us to bind such fields to the same scattered point grid in various simulators. For example, the flux term \( Q_l \) is evaluated and assembled over the scattered grid in the melt heat solver (see Box 5a). Similarly, \( Q_s \) is evaluated and assembled over the scattered grid in the solid heat simulator (class ‘NlHeat2’). Since there is only one grid of scattered points on the interface, there is a direct one to one correspondence between the two flux terms.

The representation of the front velocity as a field is particularly useful. It plays a dual role, one as the output of the front-velocity solver (performed in the main Simulator ‘Solidification’) and another as the input to the transfinite mapping solver in class ‘Move’ (see Section 5.4).
Box 3a. Some of the private and protected members of the class ‘NIHeat1’ that is used to model heat transfer in the melt

```cpp
class NIHeat1: public FEM, public MenuUDC, public Store4Plotting { 
    friend class Solidification;
    friend class HeatFluxCalc;
    protected:
        Handle(DegFreeFE) dof; // Mapping: nodal values → linear system
        Handle(GridFE) grid; // Finite element grid
        Handle(GridScatPt) grid_front; // Freezing front definition
        Handle(FieldScatPt) Q; // Heat flux on the front nodes
        VecSort(int) node_number; // Melt grid nodes on the interface
        VecSimple(Ptv(real)) X; // Projection points of the front nodes
        Handle(FieldFE) T; // The unknown temperature
        Handle(FieldFE) T_prev; // Known temperature at the previous time step
        Handle(FieldFE) T_dot; // The unknown temperature rate
        Handle(FieldFE) T_dot_prev; // Known temperature rate at the previous time step
        Handle(FieldsFE) v; // Known fluid velocity
        Handle(FieldsFE) V; // Known mesh velocity
        UpwindFE pt; // Petro-Galerkin weights
        // — Problem dependent data —
        real alpha; // Thermal diffusivity
        real gamma; // Time integration parameter
        real mixed_bdy_const; // Constant for mixed boundary condition
        real T_infinity; // T_infinity for mixed boundary condition
        //
        HeatFluxCalc hfc; // Administering the heat flux calculation
        Vec(real) nonlin_solution1; // Temperature at previous step (scratch vector)
        Vec(real) nonlin_solution2; // Temperature rate at previous step (scratch vector)
        Vec(real) linear_solution; // Solution of linear subsystem
        Handle(LinEqAdm) lineq; // Interface to linear solvers
        Handle(LinEqAdm) lineq_flux; // Linear system and solution for heat flux evaluation
        Handle(TimePrm) tip; // Time loop parameters
};
```

we have a separate data structure to represent fields over scattered data points, a pointer of this data structure can be declared as a member of both solvers with the two pointers binded together. Thus, a physical copy of the interface grid and interface velocity are avoided.

5.4. Modelling the mesh motion

The motion of the mesh and the adaptation of the grids of the solid and liquid phases to the freezing front motion is performed using member functions of the class ‘Move’ (Box 4). This class has
Box 3b. Basic functionality of the class ‘NlHeat1’

```cpp
class NlHeat1: public FEM, public MenuUDC, public Store4Plotting {
  .
  protected:
  virtual void timeLoop();
  virtual void solveAtThisTimeLevel();
  virtual void fillEssBC();
  virtual void integrands(ElmMatVec& elmat, FiniteElement& fe);
  virtual void calcElmMatVec(int elm_no, ElmMatVec& elmat, FiniteElement& fe);
  virtual void integrands4side(int side, int boind, ElmMatVec& elmat, FiniteElement& fe);

  // The following three functions are problem dependent and hence
  // must be defined in a derived class

  virtual void setIC(Bool lean restart, real t_start) = 0; // Initial temperature
  virtual real newmann(const Ptv(real)& x) = 0; // Boundary heat flux
  virtual real essBC(int nno, const Ptv(real)& x) = 0; // Prescribed bound. temp.
  void sort(Ptv(int) no_nodes_dir, VecSimple(Ptv(real))& x, // Sorting of the
            VecSort(int)& node_number); // front nodes

  public:
  NlHeat1();
  ~NlHeat1() {}
  virtual void define(MenuSystem& menu, int level = MAIN);
  virtual void scan(MenuSystem& menu);
  virtual void adm(MenuSystem& menu);
  virtual void driver();
  void storeResults();
  void updateDataStructures();
  void calcHeatFlux();
  .
};
```

been developed for purely geometric reasons. Given the solid and liquid phase grids at any iteration 
$i$ during the solution process and the interface update velocity (i.e. the front velocity $V_n$ at the first 
itration or at the following iterations the incremental velocity $\Delta V_{n+1} = \gamma(V_{n+1} - V_{n+1})$), it provides 
the means to update the two grids to the next iteration $(i+1)$ as well as to calculate the corresponding 
overall mesh motion from time $t_n$ to time $t_{n+1}$ using the updated grids and the grids at time $t_n$.

The front nodal velocities which are input to the solver are used to calculate the position of 
the interface at the next iteration (i.e. at the first iteration update as $P_n^{(0)} = P_n + V_n \Delta t$ and at 
the following iterations update as $P_n^{(i+1)} = P_n^{(i)} + \Delta t / \Delta V_{n+1}$). The calculated set of points on 
the interface is used to define a spline function passing through these points. A B-spline or a
Box 4. Description of the class ‘Move’ that defines the mess motion

class Move: public MenuUDC {
    friend class Solidification;

    protected:
    Handle(GridFE) grid_sol; // Grid in the solid at the current iteration
    Handle(GridFE) grid_liq; // Grid in the liquid at the current iteration
    Handle(TimePrm) tip; // Time discretization parameters
    VecSimple(Ptv(real)) X; // Coordinates of the projection of the front
    // nodes on a plane normal to the solidification direction
    VecSort(real) f; // Coordinates of front nodes in the solidification direction
    GridFE grid_sol_prev; // Grid in the solid region at the previous time step
    GridFE grid_liq_prev; // Grid in the liquid region at the previous time step
    Handle(FieldsFE) v_sol; // Solid mesh velocity (at time $t_{n+1}$)
    Handle(FieldsFE) v_liq; // Liquid mesh velocity (at time $t_{n+1}$)
    Handle(GridScatPt) grid_front;
    VecSimple(real) node_number; // Mapping: solid grid nodes on the front to
    // local front nodes
    Handle(FieldScatPt) V; // Front velocity ($V_n$ or $\gamma \Delta V'_{n+1}$)
    Handle(TransMap) mapping_liq; // Mapping of the melt region
    Handle(TransMap) mapping_sol; // Mapping of the solid region
    Ptv(int) no_nodes_dir; // Numbering of front nodes in each direction
    // on the projection plane

    protected:
    void newPosition(); // Calculates the new front position
    void velocityField(); // Calculates the solid and liquid mesh velocity fields
    void setIC(); // Defines the spline interpolation for the interface at $t_n$
    void timeLoop();
    void solveAtThisTimeLevel(); // Moves the grids and calculates the mesh velocities
    void updateDataStructures(); // Updates the splines at every iteration

    public:
    Move();
    ~Move();
    void define (MenuSystem& menu, int level = MAIN);
    void scan (MenuSystem& menu);
    void adm (MenuSystem& menu);
    void driver();
};

tensor-product surface is used in two-dimensional and three-dimensional problems, respectively.\textsuperscript{39−41}
This spline representation is defined using the sorted list (in an ascending order) of the projection points of the interface nodes as well as the co-ordinates of the interface nodes in the direction of solidification.\textsuperscript{41} The present implementation is limited to problems for which there is a one-to-one
mapping between each interface point and its projection on a plane normal to the direction of solidification.

There is a two-fold advantage of representing the interface using a spline. Firstly, a spline form of the interface allows the calculation of a smooth normal unit vector to the interface. Secondly, a spline representation of the interface allows a transfinite mapping to be defined pointwise for the transformation of the solid and liquid domains at iteration $i$ to the solid and liquid domains, respectively, at iteration $(i + 1)$. Various secondary advantages can also be mentioned including the ease of plotting fields defined over the interface.

The ‘Move’ class consists of two pointers (‘Handles’) of the base class ‘TransMap’ that is used to define the pointwise transfinite mapping function for a moving grid in a region using information on the motion of the region’s boundary. These pointers are binded to the objects of the derived classes ‘TransSol’ and ‘TransLiq’, the mappings for the solid and the liquid domains, respectively. The use of pointers of the base class ‘TransMap’ is necessary in order to allow us to also address melting problems in which the geometric roles of the solid and liquid mapping objects are interchanged. In this case, the solid mapping pointer is binded to an instance of the class ‘TransLiq’ and the liquid mapping pointer to an instance of the class ‘TransSol’. The classes ‘TransSol’ and ‘TransLiq’ provide the functions necessary for the spline interface fitting as well as for the definition of the mapping of a point from each grid from iteration $i$ to the corresponding grid at iteration $(i + 1)$. These operations require knowledge of the interface nodal co-ordinates in the direction of solidification as well as of the projection points. Such information is available in class ‘Move’ that is declared as a friend of the ‘TransMap’ class. Taking into account the directional nature of the problems examined, we here consider a linear stretching/or shrinking of each grid in the direction of solidification.

The mesh velocity at each node is estimated using a finite differences approximation in time based on the locations of the node in the two meshes at times $t_n$ and $t_{n+1}$.

5.5. The main simulator

The main simulator is implemented as a class ‘Solidification’ (Box 5). In this class, five main solvers are incorporated as pointers to objects of the simulators outlined earlier. These simulators include the melt flow solver ‘NavierStokes’, the heat ‘NIHeat1’ and solute ‘Concentration’ field solvers for the melt, the heat solver ‘NIHeat2’ for the solid and the transfinite mapping solver ‘Move’ for the mesh motion. The front velocity calculation (via the Stefan condition) is performed with a member function in the main simulator. The entire structure and interaction between the different solvers is graphically presented in Plate 1. In this plate, an arrow going into a solver implies input member data to the solver, whereas an arrow pointing out of a solver implies calculated data. Finally, note that the concentration field is needed in both the ‘NIHeat1’ and ‘NIHeat2’ classes due to the temperature/concentration coupling at the freezing interface introduced by the phase diagram.

Each of the various continuum fields, e.g. the temperature in the melt, is a member of various classes (e.g. the melt temperature is a member of both ‘NavierStokes’ and ‘NIHeat1’). To avoid creating multiple copies of the melt temperature field when objects of the various classes are defined, the melt temperature fields of the various classes must be binded to the same location in memory. The function ‘scan’ of the main simulator performs this task for the various fields (Box 6).

Remark 6. Note that multiple copies of the finite element grid in the melt and in the solid phase are introduced. For example, objects of the classes ‘NavierStokes’ and ‘NIHeat1’ are binded
Plate 1. Details of the structure and interaction between the various solvers (classes).
Plate 2. 3D solidification of aluminum—finite element mesh in the solid phase at an intermediate time step (τ = 0.35)
Plate 3. 3D solidification of aluminum—dimensionless velocity contours at: (a) $\tau = 0.35$, $u_{\text{max}} = 19.819$; (b) $\tau = 0.55$, $u_{\text{max}} = 4.136$
Plate 4. 3D solidification of aluminum—dimensionless temperature contours at: (a) $\tau = 0.325$; (b) $\tau = 0.525$
Box 5. The main solidification simulator

class Solidification: public MenuUDC, public Store4Plotting {
    protected:
        Handle(Move) front;
        Handle(NlHeat2) solid_heat;
        Handle(NlHeat1) melt_heat;
        Handle(NavierStokes) melt_flow;
        Handle(Concentration) melt_conc;
        Handle(TimePrm) tip;
        real tolerance;       // Relative error in the front velocity
        real gamma;
        real stefan_number;
        real cond_alpha_ratio; // Ratio $R_b/R_s$
        real initial_temp_grad; // Initial temp. gradient discontinuity at the front
        String problem;       // Solidification or melting
        Boolean restart;
        Handle(GridScatPt) grid_front;
        Handle(FieldScatPt) V.p;    // Front velocity at $t_n$
        Handle(FieldScatPt) V.i_next; // Front velocity at $t_{n+1}$ at the prev. iter.
        Handle(FieldScatPt) V.i_next_next; // Front velocity at $t_{n+1}$ at current iter.
        Handle(FieldScatPt) V;       // Front velocity to move (update) the grids
    ...
    public:
        Solidification();
        ~Solidification();
        void define(MenuSystem& menu, int level = MAIN);
        void scan(MenuSystem& menu);
        void adm(MenuSystem& menu);
        void driver();
        void timeLoop();
        void solveAtThisTimeLevel();
        void calcNewVelocity();
        void calcError(real& x);
        void initialVelocity(Boolean restart);
        void solveForVelocity(Vec(real)& solution);
    ...
};

to different ‘GridFE’ objects. These multiple copies of grids are essential, since a ‘GridFE’ object contains much more information than the partition of the geometry in finite elements. For example, a ‘GridFE’ object contains information on the boundary indicators that are different for the flow, heat transfer or concentration simulators. Box 7 gives the required steps for copying the
Box 6. The binding process of the pointers of various fields that are used as members in multiple classes

```c
void Solidification::scan(MenuSystem& menu) {
    // Definition of the various data
    ..
    // Binding of the various pointers
    melt_heat → v.rebind(melt_flow → v());
    melt_flow → T.rebind(melt_heat → T());
    front → tip.rebind(tip());
    solid_heat → tip.rebind(tip());
    melt_heat → tip.rebind(tip());
    melt_flow → tip.rebind(tip());
    front → grid_front.rebind(grid_front());
    front → node_number.redim(grid_front → getNoPoints());
    for(int i1=1; i1 < grid_front → getNoPoints(); i1++)
        front → node_number(i1) = solid_heat → node_number(i1);
    ..
}
```

Box 7. The grid copying process. Note that the appropriate boundary indicators have to be attached to the grid object of each simulator

```c
void Solidification::solveAtThisTimeLevel() {
    ..
    solid_heat → grid() = front → grid_sol();
    melt_heat → grid() = front → grid_liq();
    melt_flow → grid() = front → grid_liq();
    ..
    solid_heat → grid → redefineBoInds(solid_heat → redef);
    melt_heat → grid → redefineBoInds(melt_heat → redef);
    melt_flow → grid → redefineBoInds(melt_flow → redef);
    ..
}
```
calculated grids in class ‘Move’ to the various ‘GridFE’ members in classes ‘NlHeat1’, ‘NlHeat2’, ‘NavierStokes’ and ‘Concentration’.

6. NUMERICAL EXAMPLES

Four examples are examined in this section. The first and second examples examine a solidification and a melting process for a pure material. Both of these examples have been investigated using
Table II. Thermal properties of pure aluminium

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heat conductivity in solid</td>
<td>$k_s$</td>
<td>0.0548 kcal/m°C</td>
</tr>
<tr>
<td>Heat conductivity in melt</td>
<td>$k_l$</td>
<td>0.0548 kcal/m°C</td>
</tr>
<tr>
<td>Specific heat in solid</td>
<td>$C_s$</td>
<td>0.2526 kcal/kg°C</td>
</tr>
<tr>
<td>Specific heat in melt</td>
<td>$C_l$</td>
<td>0.2526 kcal/kg°C</td>
</tr>
<tr>
<td>Latent heat</td>
<td>$L_H$</td>
<td>95.0 kcal/kg</td>
</tr>
<tr>
<td>Density</td>
<td>$\rho$</td>
<td>2650 kg/m³</td>
</tr>
<tr>
<td>Kinematic viscosity</td>
<td>$\nu$</td>
<td>1.22 x 10⁻⁶ m²/s</td>
</tr>
<tr>
<td>Thermal expansion coefficient</td>
<td>$\beta$</td>
<td>3.84 x 10⁻⁵ °C⁻¹</td>
</tr>
<tr>
<td>Melting temperature</td>
<td>$T_m$</td>
<td>660 °C</td>
</tr>
<tr>
<td>Reference temperature</td>
<td>$T_0$</td>
<td>860 °C</td>
</tr>
</tbody>
</table>

many other algorithms and as such they are necessary to demonstrate the relative accuracy of the present simulator. The third example examines the stable directional growth of an alloy and to the best of our knowledge this is the first time that such a simulation is reported using a front tracking finite element method. The last example examines a three-dimensional solidification process for a pure material. In the following examples, the time step selected was constant for most of the calculation. In the beginning of solidification or melting, a smaller time step than the one reported here was used to account for initial transients as well as to reduce the error related to the assumption of a small solid region (or liquid in the case of melting) at the beginning of the simulation. The time required for the formation of the assumed initial solid or liquid region was estimated when possible using the solution of the corresponding one-dimensional Stefan problem. The dimensionless time in this section is denoted as $\tau$ and the actual time as $t$.

6.1. Solidification of pure aluminum in a square region

This problem was examined earlier in Reference 16. The problem definition is repeated here for completeness. A square mold with aspect ratio of 1:0 is considered. It is initially occupied by liquid aluminium whose properties are given in Table II. The ambient temperature was taken to be $25^\circ$C. The melt is initially superheated by an amount of $\Delta T_0 = 200^\circ$C. A mixed temperature/flux boundary condition was applied to one wall and all other walls were insulated as shown in Figure 3. The dimensionless variables governing the problem are summarized in Table III.

Two cases were analysed, corresponding to Rayleigh numbers of $Ra = 10^5$ and $10^6$. A $15 \times 25$ finite element mesh was chosen for the solid domain and a $25 \times 25$ mesh for the liquid domain. A small initial solid region (1 per cent solid) was assumed at the start of the simulation. A uniform temperature gradient was assumed in the initial solid region. As the solidification progresses, an anticlockwise circulation is established in the melt with the fluid ascending along the heated wall and cold fluid descending along the interface as shown in Figure 3. As the cold fluid descends, the solidification progresses faster in the lower part of the cavity and hence the solid region is thicker at the bottom than at the top as seen in Figure 4. A nondimensional time step of $\Delta \tau = 0.001$ was used in the simulation process for the $Ra = 10^5$ case. For the $Ra = 10^6$ case, a smaller time step of $\Delta \tau = 0.0005$ was used to account for stronger convection and the fact that the interface curvature develops very early in the simulation. The interface positions obtained for this case are compared with the results of Reference 16 in Figure 5. It can be seen that the results agree very well. The same is true for the case of $Ra = 10^5$. The dimensionless velocity vectors and isotherms
Table III. Dimensionless variables governing the solidification of pure aluminium

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prandtl number</td>
<td>$Pr$</td>
</tr>
<tr>
<td>Rayleigh number</td>
<td>$Ra$</td>
</tr>
<tr>
<td>Stefan number</td>
<td>$Ste$</td>
</tr>
<tr>
<td>Heat conductivity ratio</td>
<td>$R_k$</td>
</tr>
<tr>
<td>Specific heat ratio</td>
<td>$R_c$</td>
</tr>
<tr>
<td>Biot number</td>
<td>$Bi$</td>
</tr>
</tbody>
</table>

Table III lists dimensionless variables governing the solidification of pure aluminium. The symbols and their values are as follows:

- **Prandtl number** ($Pr$): 0.0149
- **Rayleigh number** ($Ra$): $10^3 / 10^6$
- **Stefan number** ($Ste$): 0.5317
- **Heat conductivity ratio** ($R_k$): 1.0
- **Specific heat ratio** ($R_c$): 1.0
- **Biot number** ($Bi$): 3.3

These values are obtained at different nondimensional times and are shown in Figures 6(a)–(d) and 7(a)–(d) for the $Ra = 10^5$ case. For brevity, the temperature and flow results for the case $Ra = 10^6$ are omitted.

The results in both cases agree very well with those obtained in Reference 16. The discrepancy in the maximum velocity values at the initial stage of solidification is due to the uncertainties in the initial temperature distribution in the assumed solid region. The approximate time for complete solidification is around $\tau = 0.8$. It can be observed from the velocity fields that there is a tendency to form secondary cells near the upper end of the right wall. This is in accordance with the observations made in Reference 16. Also, it must be noted that the Rayleigh number plays an important role in the solidification process since different flow and temperature patterns resulted for the two $Ra$ number cases that were examined.

6.2. Melting of pure gallium

This benchmark problem has been analysed by various authors.43–45 Experimental results are reported in Reference 46. Pure solid gallium at its melting point is held in a two-dimensional cavity (height $H = 0.045$ m; width $L = 0.09$ m). At time $\tau = 0$, the temperature of the left-hand wall of the cavity is raised whereas the temperature of the right-hand-wall is maintained at the melting temperature and the upper and lower walls are insulated. The hot wall is kept at 38.3°C. As time increases, melting proceeds to the right from the left vertical wall. Due to natural convection in the melt, the isothermal solid–liquid interface does not remain parallel to the left vertical wall.
Figure 4. Finite element grid at an intermediate stage of the solidification of pure aluminium ($\tau = 0.35$)

Figure 5. Solidification of pure aluminium—the calculated interface positions for the $Ra = 10^6$ case compared with the results reported in Reference 16

An illustration of the process at some intermediate stage of melting is given in Figure 8. The material properties of gallium are listed in Table IV, and the values of the parameters governing the problem in Table V.

As can be noted from Figure 9, the buoyancy-induced natural convection plays a much stronger role in determining the shape of the solid–liquid interface in a melting problem than in a solidification problem. Thus, a fine grid is required near the interface to capture the features of the flow in that region. As a compromise between the computational cost and the obtained accuracy,
Figure 6. Solidification of pure aluminium—dimensionless velocity fields for the case $Ra = 10^5$: (a) $\tau = 0.05$, $|\mathbf{u}|_{\text{max}} = 34.56$; (b) $\tau = 0.15$, $|\mathbf{u}|_{\text{max}} = 30.07$; (c) $\tau = 0.35$, $|\mathbf{u}|_{\text{max}} = 13.01$; (d) $\tau = 0.55$, $|\mathbf{u}|_{\text{max}} = 2.722$

A finite element grid consisting of 806 nodes and 750 elements in the liquid domain was used to simulate the process. The initial solid region was selected as 0.5 per cent of the total region (corresponding to a non-dimensional length of 0.01).

Since the solid remains at the constant melting temperature, there is no need to solve the heat conduction problem in the solid phase. A non-dimensional time step of 0.00025 was used for the
Figure 7. Solidification of pure aluminium—dimensionless temperature fields for the case \( Ra = 10^5 \): (a) \( \tau = 0.05 \); (b) \( \tau = 0.15 \); (c) \( \tau = 0.35 \); (d) \( \tau = 0.55 \)

calculations which corresponds to 0.35 s of actual time. The entire simulation extended over a total
time of approximately 12 min.

The results obtained are compared with the experiments of Gau and Viskanta\(^{46}\) and also with the
numerical simulations given in Reference 44 (see Figure 10). As already explained in Reference 44,
the discrepancy between the numerical simulation is due to uncertainties in the experimental data.
The present results are also compared with the simulations reported in Reference 43. The maximum
stream function values reported in Reference 43 using a transformed grid technique are very close
Figure 8. Melting of gallium—schematic illustrating an intermediate stage of the melting process

Table IV. Material properties of gallium

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fusion temperature</td>
<td>$T_m$ 29.78°C</td>
</tr>
<tr>
<td>Conductivity in the solid</td>
<td>$k_s$ 33.5 W/mK</td>
</tr>
<tr>
<td>Conductivity in the melt</td>
<td>$k_l$ 33.5 W/mK</td>
</tr>
<tr>
<td>Specific heat in the solid</td>
<td>$C_s$ 395.15 J/kgK</td>
</tr>
<tr>
<td>Specific heat in the melt</td>
<td>$C_s$ 395.15 J/kgK</td>
</tr>
<tr>
<td>Latent heat</td>
<td>$L_H$ 80.16 kJ/kg</td>
</tr>
<tr>
<td>Density</td>
<td>$\rho$ 6095 kg/m³</td>
</tr>
<tr>
<td>Thermal diffusivity</td>
<td>$\alpha$ 1.44 x 10⁻⁵ cm²/s</td>
</tr>
<tr>
<td>Kinematic viscosity</td>
<td>$\nu$ 2.92 x 10⁻³ cm²/s</td>
</tr>
<tr>
<td>Thermal expansion coefficient</td>
<td>$\beta$ 1.73 x 10⁻⁴ 1/°C</td>
</tr>
</tbody>
</table>

Table V. Dimensionless variables for the melting of gallium

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prandtl number</td>
<td>$Pr$ 0.021</td>
</tr>
<tr>
<td>Rayleigh number</td>
<td>$Ra$ 2.2 x 10⁷</td>
</tr>
<tr>
<td>Stefan number</td>
<td>$Ste$ 0.042</td>
</tr>
<tr>
<td>Heat conductivity ratio</td>
<td>$R_k$ 1.0</td>
</tr>
<tr>
<td>Specific heat ratio</td>
<td>$R_C$ 1.0</td>
</tr>
</tbody>
</table>

to the ones calculated with the present front tracking method and are shown in Table VI. Also, very good agreement is seen with the flow patterns reported in Reference 43. The mid-plane vertical velocity profiles agree well with those reported in Reference 43 except very close to the interface (see Figure 11). This is due to the fine grid used there compared to the one in the present simulation (2000 elements compared to 750 elements).

The dimensionless streamlines and isotherms at various non-dimensional times are shown in Figs. 12(a)–(c) and 13(a)–(c). The flow patterns are similar to those reported in Reference 43 using a transformed grid method. At very early time secondary cells are formed that significantly
Figure 9. Melting of gallium—finite element grid at an intermediate stage of the melting process ($t = 7.66\text{min}$)

Figure 10. Melting of gallium—comparison of melting front locations at different times

distort the shape of the interface. However, as the solidification proceeds, the secondary cells merge quickly and a single circulation persists for the remaining of the melting process.

6.3. Solidification of an alloy crystallizing at a vertical wall

In the present section, we report results on a numerical simulation of an alloy solidification problem involving an aqueous solution. The problem was obtained from the work of Thompson and Szekely.32,33 No point-to-point comparisons with the results of Thompson and Szekely are attempted here due to a number of simplifications employed in their implementation as well as a
number of inconsistencies in the reported dimensionless quantities. Hence, the calculated results are compared only qualitatively with the results in References 32 and 33.

The solidification system consists of a square cavity approximately 1 cm x 1 cm filled with a 2 wt per cent aqueous sodium carbonate solution. The experimental results reported in References 32 and 33 show that this alloy exhibits a thin mushy zone and as such the distinct sharp interface approximation is a reasonable assumption for the solidification analysis. Further experiments by Leitch\(^47\) in the solidification of sodium carbonate solutions confirm the validity of the thin mushy layer approximation for such systems (see also Reference 48).

The right and left vertical walls are at constant hot and cold temperatures of \(+10^\circ\text{C}\) and \(−10^\circ\text{C}\), respectively. The exact numerical values of the physical constants used in the simulation are listed in Table VII. The corresponding dimensionless quantities used are listed in Table VIII. The problem geometry and non-dimensional boundary conditions used are summarized in Figure 14. The flow pattern and expected behaviour is also shown in the same figure.

Due to the high Lewis number and large front velocity in the present solidification system, a severe restriction is placed on the grid spacing which in turn affects the time step. A boundary-layer analysis shows that for the given problem data the value of the concentration boundary layer \(\delta_c\) is of the order of \(10^{-4}\). This clearly shows that the grid at the interface boundary must be at least as small as \(\delta_c\).

In the present simulation, finite element grids consisting of 25 x 50 divisions in the solid region and 40 x 50 divisions in the liquid region were chosen. The grids are highly graded to accommodate

| Time (in min) | \(|\Psi|_{\text{present}}\) | \(|\Psi|_{\text{reported}}\) |
|--------------|-----------------|-----------------|
| 1.02         | 2.51            | 2.50            |
| 2.04         | 3.86            | 3.50            |
| 2.74         | 4.51            | 4.3             |
| 4.85         | 6.09            | 5.85            |
| 6.25         | 6.23            | 6.7             |
| 9.54         | 6.69            | 7.2             |
Figure 12. Melting of gallium—dimensionless streamline contours: (a) $\tau = 2.07$; (b) $\tau = 3.27$; (c) $\tau = 5.07$
Figure 13. Melting of gallium—dimensionless temperature fields: (a) $\tau = 2.07$; (b) $\tau = 3.27$; (c) $\tau = 5.07$
Table VII. Alloy solidification—physical constants for the Na₂CO₃–H₂O system

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value for Na₂CO₃–H₂O System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Density</td>
<td>ρ 1.0 g/cm³ (15 wt per cent Na₂CO₃)</td>
</tr>
<tr>
<td></td>
<td>1.16 g/cm³ (15 wt per cent Na₂CO₃)</td>
</tr>
<tr>
<td>Thermal diffusivity</td>
<td>α 1.5 × 10⁻³ cm²/s (liquid)</td>
</tr>
<tr>
<td></td>
<td>1.5 × 10⁻³ cm²/s (solid)</td>
</tr>
<tr>
<td>Latent heat of fusion</td>
<td>Lₜ 76.0 cal/gm</td>
</tr>
<tr>
<td>Kinematic viscosity</td>
<td>v 0.93 × 10⁻² cm²/s</td>
</tr>
<tr>
<td>Solute diffusivity</td>
<td>D 1.5 × 10⁻³ cm²/s</td>
</tr>
<tr>
<td>Coefficient of thermal expansion</td>
<td>βₜ 3.0 × 10⁻⁴/°C</td>
</tr>
<tr>
<td>Coefficient of solutal expansion</td>
<td>βc 8.0 × 10⁻⁵ wt per cent</td>
</tr>
<tr>
<td>Slope of liquidus line</td>
<td>m₁ −0.33°C/wt per cent</td>
</tr>
<tr>
<td>Slope of solidus line</td>
<td>mₖ 1.7°C/wt per cent</td>
</tr>
<tr>
<td>Characteristic length</td>
<td>L 1 cm</td>
</tr>
<tr>
<td>Thermal conductivity</td>
<td>k 1 × 10⁻³ cal/cm s°C (liquid)</td>
</tr>
<tr>
<td></td>
<td>1 × 10⁻³ cal/cm s°C (solid)</td>
</tr>
</tbody>
</table>

Table VIII. Alloy solidification—dimensionless variables governing the alloy solidification problem

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prandtl number</td>
<td>Pr 6.0</td>
</tr>
<tr>
<td>Thermal Rayleigh number</td>
<td>Raₜ 5 × 10⁵</td>
</tr>
<tr>
<td>Solutal Rayleigh number</td>
<td>Raₖ 1 × 10⁶</td>
</tr>
<tr>
<td>Stefan number</td>
<td>Ste 0.2</td>
</tr>
<tr>
<td>Heat conductivity ratio</td>
<td>Rₘ 1.0</td>
</tr>
<tr>
<td>Heat diffusivity ratio</td>
<td>Rₜ 1.0</td>
</tr>
<tr>
<td>Lewis number</td>
<td>Le 10.7.0</td>
</tr>
<tr>
<td>Buoyancy ratio</td>
<td>Γ((βₜΔT₀)/(βcΔc₀)) 0.5</td>
</tr>
<tr>
<td>Distribution coefficient</td>
<td>κ ≈0</td>
</tr>
</tbody>
</table>

at least 2–3 nodes in the boundary layer as shown in Figure 15. Even with this fine mesh, the initial transient was oscillatory. A non-dimensional time step of 5 × 10⁻⁵ was used due to the small grid size.

The dimensionless streamline, temperature and concentration field contours obtained in the simulation are plotted in Figures 16 and 17, respectively. Confirming the observations reported in earlier analysis and experiments,³²,³³ the present simulation shows that during the initial stages of solidification the liquid circulates in a single convection cell and that the solid/liquid interface moves forward with the maximum width at the base. The solute rich liquid having a higher density is transported along the solid/liquid interface to the base of the enclosure where it accumulates to form a separate layer (see Figures 17(a)–(c)). Due to density differences, this lower layer remains distinct and resists mixing with the overlying, thermally convecting liquid as can be seen in Figure 16(a).

As solidification progresses, the combination of the vertical solute gradient and the lateral temperature gradient causes a low-aspect ratio horizontal convection cell to develop slowly as can be
Figure 14. Binary alloy solidification—schematic illustrating the geometry and expected flow pattern

Figure 15. Alloy solidification—finite element mesh at an intermediate stage of solidification ($\tau = 0.34$)

seen in Figure 16(b). The convective velocities in this cell are much smaller than those in the upper main circulating cell. As reported in Reference 34, the two cells are generated sequentially rather than simultaneously. Also, as was also observed in References 33 and 34, the concentration field does not vary much within these cells (see Figures 17(a) and 17(b)). Finally, we note that the agreement between the calculated flow patterns and solidification behaviour using this simulator and the methodology in References 32 and 33 is very good. As solidification progresses
further, more cells are generated. Due to the heavy computation, the simulation was stopped at a non-dimensional time of $\tau = 0.72$ which corresponds to approximately 8 min.

### 6.4. Solidification of a pure metal in a three-dimensional square cavity

Here, we consider the solidification of pure aluminium in a three-dimensional square cavity. This example will demonstrate that the present formulation and object oriented approach can easily tackle three-dimensional directional solidification problems.
The problem definition is as follows. A cubical mold with aspect ratio of 1:0 is considered. All the walls of the cavity are insulated except one, where a mixed/flux boundary condition is applied. Figure 18 shows the details. The material properties and non-dimensional parameters are exactly the same as those used in the first example. The Rayleigh number used was $10^5$.

A $6 \times 7 \times 7$ finite element uniform mesh was chosen for the solid domain and a $7 \times 7 \times 7$ mesh for the liquid domain. A small initial solid region (1 per cent solid) was assumed at the start of the simulation. A uniform temperature gradient was assumed in the initial solid region. The total simulated dimensionless time was $\tau = 0.75$, which is very close to complete solidification.
The results of the numerical simulation are shown in Plates 2–4. Note that in the following velocity vector plots the shaded region is the solid region and that in the temperature plots only the isosurfaces in the liquid are shown. As can be seen the results almost duplicate the results of Example 1 except for the expected ‘edge effects’.

As solidification progresses, the solid–liquid interface starts to curve, the solid being thicker at the base and thinner at the top (see Plate 2). The only difference between the present case and the first example is that since the problem is solved in three dimensions there is an edge-effect, which can be clearly seen in the temperature contours (Plate 4). Also, note that all three velocity components are non-zero (Plate 3) although one component is quite small. The flow behaviour is similar to the 2-D model, i.e. the flow initially develops and causes curving of the freezing interface and after reaching a peak it damps down and is almost negligible near the end of solidification. Finally, it is noted that due to the three-dimensional nature of the melt flow, the maximum velocity values listed here are not very close to the values listed in the first example for the \( Ra = 10^5 \) case.

7. CONCLUSIONS

A C++ object oriented finite element implementation for transport systems was presented with application to directional solidification processes. The capabilities of this implementation were demonstrated with the solidification and melting of pure materials as well as with the solidification of a binary alloy system under the assumption of a sharp freezing front. In addition to the \textit{diffpack} based finite element classes, the various simulators for each transport process were developed in the form of classes. The use of smart pointers, virtual functions and field objects to represent the various continuum variables allowed an easy and reliable coupling of the various simulators as well as the testing of various time integration algorithms. Classes were also developed to account for the freezing front motion and the moving grids in the solid and liquid phases.

The developed code can be used for both two- and three-dimensional problems and it can run without modifications in most computer platforms. In comparison to our earlier C based simulator,\textsuperscript{35} the present object oriented approach was only slightly more expensive. As with other object oriented applications, this extra cost is a small penalty that someone has to pay for receiving the rewards of a robust code that allows generality, expandability, maintainability and FEM code re-usability.

ACKNOWLEDGEMENTS

The computing for this project was supported by the Cornell Theory Center, which receives major funding by the NSF and IBM corporation, with additional support from the New York State. The academic license for using the various libraries of \textit{diffpack} is appreciated. The last version of this paper was prepared while the senior author was on sabbatical leave at École Polytechnique, Palaiseau, France and supported in part by the CNRS.

REFERENCES