Chapter 1

An object-oriented approach to the finite element modeling and design of material processes

Nicholas Zabaras[1]

[1]Sibley School of Mechanical and Aerospace Engineering, 188 Frank H. T. Rhodes Hall, Cornell University, Ithaca, NY 14853–3801, Email: zabaras@cornell.edu.

Abstract

An object-oriented approach to the analysis and design of metal forming and directional solidification processes is presented. A number of specific ideas are introduced on how a deformation or a solidification process can be partitioned into subproblems that can be independently developed and tested in the form of class hierarchies. Class development based on mathematical and/or physical arguments is emphasized. General ideas are provided to demonstrate that an OOP approach to the FEM modeling and design of material processes can lead to efficient simulators that are easily maintainable and expandable. A metal forming example is presented to demonstrate the ability of the deformation simulator to handle the analysis of industrial metal forming processes. Also, a design directional solidification example is presented to demonstrate the substantial benefits of using OOP techniques for the design of processes with a desired solidification morphology.

1.1 Introduction

Various commercial FEM software codes have been developed for the implementation of a variety of material processes and in particular forming and solidification processes (e.g. [1], [2]). These codes have been developed with the non-expert user in mind and generally provide only limited flexibility for the implementation of particular desired algorithms or processes. The user, however, is allowed to implement a particular material model, boundary conditions, geometry (including the finite element grid), etc. as required by the problem at hand.

In this communication, a brief presentation is given of an object-oriented programming (OOP) approach to modeling forming and solidification processes. The discussion is restricted to the development of OOP formulations that allow the user, using a well defined programming structure, to implement his/her own algorithms and/or a particular process.

The fundamentals of object-oriented programming for scientific applications are reviewed in [3]. Various applications of OOP to engineering problems can be found in [4]–[9]. Diffpack is an example of a C++ based object-oriented environment developed to address the modeling of continuum systems governed by partial differential equations (PDEs) using both finite differences and finite elements [10]. Basic finite element operations (e.g. finite element definition, integration rules, assembly of the stiffness matrix and load vectors, application of essential and natural boundary conditions, definition of continuum fields over finite element grids, etc.) have been introduced in Diffpack using class hierarchies. This provides the flexibility of solving a boundary value problem by only having to declare the particulars of the problem that may include the coefficients of the PDE, and the essential and/or natural boundary conditions. Such information is introduced by using virtual functions in hierarchies derived from base classes currently developed in Diffpack.

The major advantage of using an OOP environment such as Diffpack is that the user can utilize as much or as little of the provided class structure as it is appropriate in a particular application. Such flexibility may look against the objective of general purpose simulators developed with procedural languages, however, it provides an enormous flexibility to address various algorithms and processes that do not necessarily fit within the program structure already developed. Such flexibility does not exist with most of the current simulators based on procedural languages.

As it is impossible in one paper to present even the basic aspects of forming and solidification processes, this paper is restricted to a few mathematical and physical aspects of forming and solidification processes for which the essentials of an OOP structure are presented. However, references to the particular models implemented in the presented simulators will be given for further consultation and details.

Finite element simulation of a material process follows different ap-

proaches depending on the mathematical and physical structure of the process. For example, solidification processes are usually modeled as a coupled system of PDEs each of them linked to a particular physical mechanism (heat or solute diffusion, fluid (melt) flow, etc.) [11]. On the other hand, forming processes are considered as a set of coupled problems each of them represented with PDEs but also with variational inequalities (as for example is the case of modeling the contact/frictional constraints) [12]–[14].

The features of OOP that are explored here are inheritance and polymorphism. They lead to code expandability, maintainability and reusability. In the present work, we have used the widely used Diffpack library of classes for analysis of systems governed by partial differential equations, so as to concentrate only on the implementation of the particular physical mechanisms in the form of classes [10]. Depending on the type of process examined, a common approach to OOP implementation is by partitioning the original problem in various subproblems based on physical and/or mathematical arguments. Examples of both of these cases will be shown later in this paper. For each subproblem defined, appropriate class hierarchies are introduced to allow the user to implement process/problem specific functionality via virtual functions (`functors`). The base classes define the general structure of the problem, whereas the derived classes are implementing functionality specific to each process. This, for example, is important while modeling forming processes in order to account for the significant differences in the nature of boundary conditions between various forming processes (e.g. rolling, forging, extrusion, etc.). In addition, this structure can for example account for the similar nature of the deformation at a particular material point in the Lagrangian analysis of all forming processes.

Each subproblem can be modelled and tested independently of the other and many algorithms can be used for the coupling of all subproblems. It is here demonstrated via specific examples that an OOP approach to material processing is ideal for (1) development and testing of the various subproblems (2) coupling of the various subproblems (3) generating a generic structure that can be used as prototype for implementation of other processes, material models, and/or algorithms and (4) for a reliable memory management.

The basic structure of this paper is as follows. Forming processes are introduced in Section 1.2. A brief descriptive definition of a typical forming problem is presented in Section 1.2.1 using a Lagrangian framework. The main algorithm is summarized in Section 1.2.2. The developed object-oriented class structure based on the physics of each subproblem is reviewed in Section 1.2.3. An example of a plane strain closed-die forging process is summarized in Section 1.2.4.

A directional solidification simulator is presented in Section 1.3. A brief problem definition is given in Section 1.3.1 and the main algorithm is discussed in Section 1.3.2. The OOP implementation of FEM algorithms is

presented in Section 1.3.3. As a demonstration of a design simulator, Section 1.4 presents the definition, analysis and OOP implementation of a design binary alloy solidification problem for a desired stable growth. Such problems are here formulated as functional optimization problems. The design algorithm considered is an implementation of the adjoint method in the L_2 space. Appropriate continuum sensitivity and adjoint problems are defined as required for the implementation of functional optimization techniques (e.g. the conjugate gradient method)) [15]–[18]. It is shown that an OOP approach to such design problems provides an ideal framework in which the similarities in the mathematical structure of the various subproblems (direct, sensitivity and adjoint fluid flow, heat and mass transfer, etc., problems) are taken into account. As a particular case of the design methodology, an example problem is presented in Section 1.4.4 for the design of the boundary heat fluxes in the directional solidification of a pure material that lead to desired interface heat fluxes and growth velocity. Finally, the paper concludes with a discussion on the efficiency of OOP techniques for the implementation of FEM formulations of material processes.

1.2 Lagrangian analysis of forming processes

A research object-oriented simulator has been developed for the analysis of large deformations of isotropic, hyperelastic, viscoplastic microvoided materials including damage as well as thermal coupling. A generic OOP structure has been introduced to account for various constitutive models, various integration algorithms of constitutive models, various contact algorithms, various thermo-mechanical coupling algorithms, process-dependent conditions (including die shape, boundary conditions), etc. No attempt is made here to review all details, but the generic mathematical structure of some of the above mentioned subproblems is identified in order to motivate the selected object-oriented environment for their implementation. In Section 1.2.1 a brief definition of the deformation problem is given followed in Section 1.2.2 by a presentation of the main subproblems to be examined. Section 1.2.3 presents some of the key elements of the OOP approach to the modeling of these type of processes. More details on the precise problem definition, FEM algorithm and OOP structure can be found in [19]–[21].

1.2.1 Problem description

In a typical deformation problem, one seeks the history-dependent solution (in terms of incremental displacements, stresses, strains, state variables, etc.) that satisfies the equilibrium equations, the kinematic equations (strain/displacement relations), the constitutive equations (elastic and inelastic including the evolution of the material state) and the contact con-

straint equations. A Lagrangian framework of analysis is considered here in order to allow ourselves to evaluate residual stresses. Consider the smooth deformation mapping defined as $\boldsymbol{x} = \phi(\boldsymbol{X}, t) : \boldsymbol{B}_o \rightarrow \mathcal{R}^3$ which maps particles $\boldsymbol{X}$ in the reference configuration $\boldsymbol{B}_o$ to their positions $\boldsymbol{x}$ in the deformed configuration $\boldsymbol{B} \subset \mathcal{R}^3$ at time t. The deformation gradient $\boldsymbol{F}$ with respect to the reference configuration $\boldsymbol{B}_o$ is given by,

$$\boldsymbol{F}(\boldsymbol{X}, t) = \frac{\partial \phi(\boldsymbol{X}, t)}{\partial \boldsymbol{X}}, \qquad \det \boldsymbol{F} > 0 \tag{1.1}$$

The deformation gradient is decomposed into thermal, plastic, and elastic parts as follows:

$$\boldsymbol{F} = \boldsymbol{F}^e \ \boldsymbol{F}^p \ \boldsymbol{F}^\theta, \qquad \det \boldsymbol{F}^p > 0, \ \det \boldsymbol{F}^e > 0, \ \det \boldsymbol{F}^\theta > 0 \tag{1.2}$$

where $\boldsymbol{F}^e$ is the elastic deformation gradient, $\boldsymbol{F}^p$, the plastic deformation gradient and $\boldsymbol{F}^\theta$ is the thermal part of the deformation gradient. A graphical representation of equation (1.2) is given in Figure 1.1. Assuming isotropic thermal expansion, the evolution of the intermediate thermally expanded unstressed configuration is given as follows:

$$\dot{\boldsymbol{F}^\theta} \ \boldsymbol{F}^{\theta\,-1} = \beta \ \dot{\theta} \boldsymbol{I} \tag{1.3}$$

where β is the thermal expansion coefficient, $\dot{\theta}$ the temperature rate and $\boldsymbol{I}$ the second-order identity tensor.

From the polar decomposition of the elastic deformation gradient, we can write the following:

$$\boldsymbol{F}^e = \boldsymbol{R}^e \ \boldsymbol{U}^e \tag{1.4}$$

In the constitutive equations to be defined here, logarithmic stretches are used as strain measures. The elastic strain denoted by $\bar{\boldsymbol{E}}^e$ is defined in the intermediate hot unstressed configuration as

$$\bar{\boldsymbol{E}}^e = \ln \boldsymbol{U}^e \tag{1.5}$$

The corresponding conjugate stress measure $\bar{\boldsymbol{T}}$ used in this model is the pullback of the Kirchhoff stress with respect to $\boldsymbol{R}^e$,

$$\bar{\boldsymbol{T}} = \det(\boldsymbol{U}^e) \ \boldsymbol{R}^{e\,T} \ \boldsymbol{T} \ \boldsymbol{R}^e \tag{1.6}$$

where $\boldsymbol{T}$ represents the Cauchy stress [13].

Damage in the form of void growth is considered in the present analysis. The body is assumed to be a continuum despite the presence of microvoids. The internal state variables are taken to be $(s, \ f)$, where s represents the scalar resistance to plastic flow offered by the matrix material and f is the void volume fraction in the deformed configuration.

It is well known that the equivalent tensile stress σ_m of the matrix material should be defined implicitly in terms of the Cauchy stress and f.

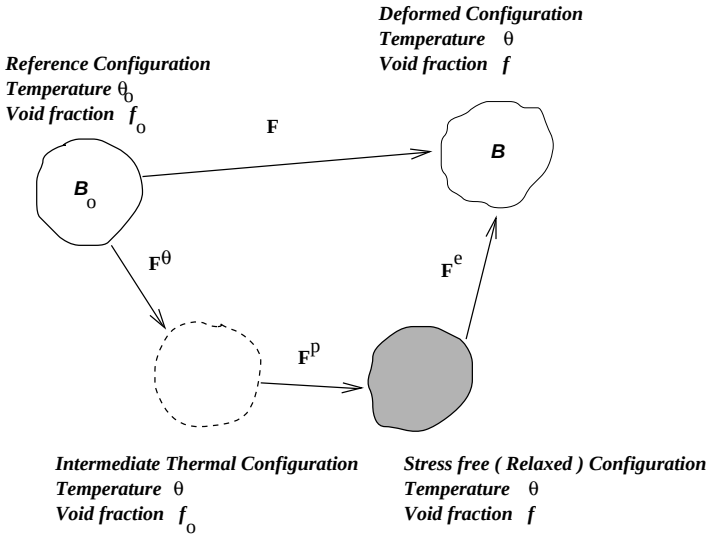

Figure 1.1: A kinematic framework for the constitutive modeling of thermo-inelastic deformations coupled with damage. The intermediate thermally expanded hot unstressed configuration and the intermediate hot plastically deformed relaxed (unstressed) configuration are shown.

With the assumption of isotropy, a particular form of this dependency is the following:

$$\Phi = \Phi(\sigma_m, f, p, \mathcal{S}) = 0 \tag{1.7}$$

where the dependence of the potential Φ on the stress $\bar{\boldsymbol{T}}$ is restricted to its first and second invariants [22], [23]. Here, the norm $\mathcal{S}$ of the stress deviator $\bar{\boldsymbol{T}}' = \bar{\boldsymbol{T}} + p\ \boldsymbol{I}$, is given by $\mathcal{S} = \sqrt{\bar{\boldsymbol{T}}' \cdot \bar{\boldsymbol{T}}'}$, where p is the mean normal pressure, $p = -\text{tr}\ \bar{\boldsymbol{T}}/3$.

The evolution of plastic flow is defined with the following normality rule:

$$\bar{\boldsymbol{D}}^p = \text{sym}\left(\bar{\boldsymbol{L}}^p\right) = \dot{\gamma}\ \partial_{\bar{\boldsymbol{T}}}\Phi = \dot{\gamma}\ \left[(\partial_{\mathcal{S}}\Phi)\boldsymbol{N} - \frac{1}{3}(\partial_p\Phi)\boldsymbol{I}\right] \tag{1.8}$$

where $\bar{\boldsymbol{L}}^p = \dot{\boldsymbol{F}}^p\ \boldsymbol{F}^{p-1}$ and $\boldsymbol{N} = \frac{\bar{\boldsymbol{T}}'}{\mathcal{S}}$ is the direction of the stress deviator. The spin in the intermediate hot unstressed plastically deformed configuration is immaterial for isotropic materials and is assumed to vanish resulting in the following equations [13]:

$$\begin{aligned} \bar{\boldsymbol{D}}^p &= \bar{\boldsymbol{L}}^p \\ \bar{\boldsymbol{W}}^p &= 0 \end{aligned} \tag{1.9}$$

The parameter $\dot{\gamma}$ in equation (1.8) is determined using the work equivalence relation $\bar{\boldsymbol{T}}' \cdot \bar{\boldsymbol{D}}^p = (1-f)\sigma_m \dot{\bar{\epsilon}}_m^p$.

The evolution of the equivalent tensile plastic strain in the matrix $\tilde{\epsilon}_m^p$ is specified via uniaxial experiments as:

$$\dot{\bar{\epsilon}}_m^p = f(\sigma_m, s, \theta) \tag{1.10}$$

and the evolution of the isotropic scalar resistance s is also obtained from experiments and has the form,

$$\dot{s} = g(\sigma_m, s, \theta) = h(\sigma_m, s, \theta)\dot{\bar{\epsilon}}_m^p - \dot{r}(s, \theta) \tag{1.11}$$

where $\dot{r}(s, \theta)$ is the static recovery function. For example of such a model see [24]. The evolution equation for the void fraction is essentially a statement concerning conservation of mass and takes the form:

$$\dot{f} = (1-f)\mathrm{tr}(\bar{\boldsymbol{D}}^p) \tag{1.12}$$

The hyperelastic constitutive model used here is as follows [13]:

$$\bar{\boldsymbol{T}} = \boldsymbol{\mathcal{L}}^e \left[\bar{\boldsymbol{E}}^e\right] \tag{1.13}$$

where the isotropic elastic moduli $\boldsymbol{\mathcal{L}}^e$ are generally functions of f and θ [25].

Finally, in the absence of external heat sources, the balance of energy in the current configuration takes the following form:

$$\rho c \dot{\theta} = \mathcal{W}_{\mathrm{mech}} + \nabla \cdot k \, \nabla\theta \tag{1.14}$$

where c is the specific heat capacity per unit mass (generally a function of f and θ), ρ the density in the current configuration and k the conductivity. In general the thermophysical properties are considered to be functions of f and θ. The mechanical dissipation $\mathcal{W}_{\mathrm{mech}}$ is usually specified in terms of the plastic power by the following empirical law,

$$\mathcal{W}_{\mathrm{mech}} = \omega \, \bar{\boldsymbol{T}} \cdot \bar{\boldsymbol{D}}^p \tag{1.15}$$

where the constant dissipation factor ω represents the fraction of the plastic work that is dissipated as heat.

1.2.2 FEM algorithm

The overall algorithm is presented in Figure 1.2 (assuming for simplicity isothermal conditions). The incremental deformation problem consists of a Newton-Raphson iteration for the calculation of the incremental displacements. Such a step requires full linearization of the principle of virtual work (including contributions from contact and from the assumed strain

model used to model the near-incompressibility conditions, [19]) as well as a calculation of the consistent linearized material moduli.

In the *constitutive integration incremental subproblem*, the variables $(\boldsymbol{T}, s, f, \boldsymbol{F}^e)$ are determined at the end of the time step (time t_{n+1}), given the body configuration $\boldsymbol{B}_{n+1}$ and the temperature field at time t_{n+1}. The body configuration $\boldsymbol{B}_n$, the temperature field at time t_n and the variables $(\boldsymbol{T}, s, f, \boldsymbol{F}^e)$ at the beginning of the time step (time t_n) are known. Thus, the solution is advanced within the incremental solution scheme by integrating the flow rule (equation (1.8)), the evolution equation for the state variable s (equation (1.11)) and the evolution equation for the void volume fraction f (equation (1.12)) over a finite time step $\Delta t = t_{n+1} - t_n$ [12], [13], [19]. In the present simulator [19]–[21], a full Newton-Raphson scheme has been developed for the calculation of the incremental displacements. Radial-return algorithms have been implemented for both rate-dependent and rate-independent models of dense [13], [19] and micro-voided materials [20], [26]–[30].

In the *contact/friction subproblem*, given the configuration $\boldsymbol{B}_{n+1}$, the die location and shape at time t_{n+1}, as well as estimates of the contact tractions (e.g. from the previous time step or a previous Newton-Raphson iteration), one updates the regions of contact as well as the applied contact tractions at t_{n+1}. The contact subproblem is coupled with the *kinematic incremental problem* where given $(\boldsymbol{T}_{n+1}, s_{n+1}, f_{n+1}, \bar{\boldsymbol{F}}^p_{n+1})$, the configuration $\boldsymbol{B}_n$ and the applied boundary conditions at t_{n+1} (including the contact tractions), one calculates (updates) the configuration $\boldsymbol{B}_{n+1}$. Various algorithms have been developed to implement contact and friction. The augmented Lagrangian formulation of Simo and Laursen was found appropriate for fully implicit Lagrangian formulations [19].

For thermomechanical processes including hot forming, a coupling of a thermal analysis with the deformation algorithm of Figure 1.2 must be introduced. Various implicit, explicit and semi-implicit algorithms can be found in [20] and [31]–[33].

1.2.3 An OOP approach to the FEM analysis of forming processes

Figure 1.3 presents the main object-oriented simulator `LargeDef` introduced to model large deformations of inelastic bodies. Most of the members of this class are developed based on physical and algorithmic arguments. For example, the material constitutive model, the integration of the constitutive model, the die geometry, the contact/friction model and its integration, the description of the local kinematics at a material point, etc. are implemented in the form of classes. Such a class structure allows us to easily implement a number of coupling algorithms for the various subproblems without paying special attention to the formulation and algorithmic details of each of them.

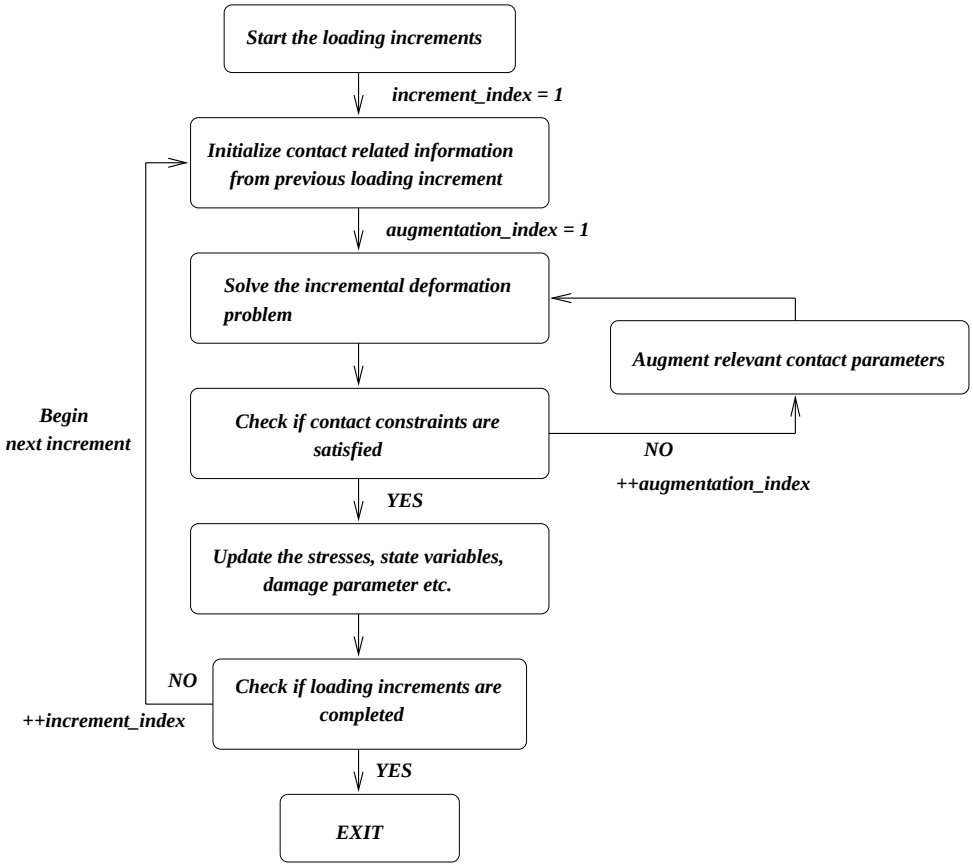

Figure 1.2: The overall algorithm for the solution of an isothermal deformation problem. The contact algorithm is based on an augmented Lagrangian implicit formulation [14].

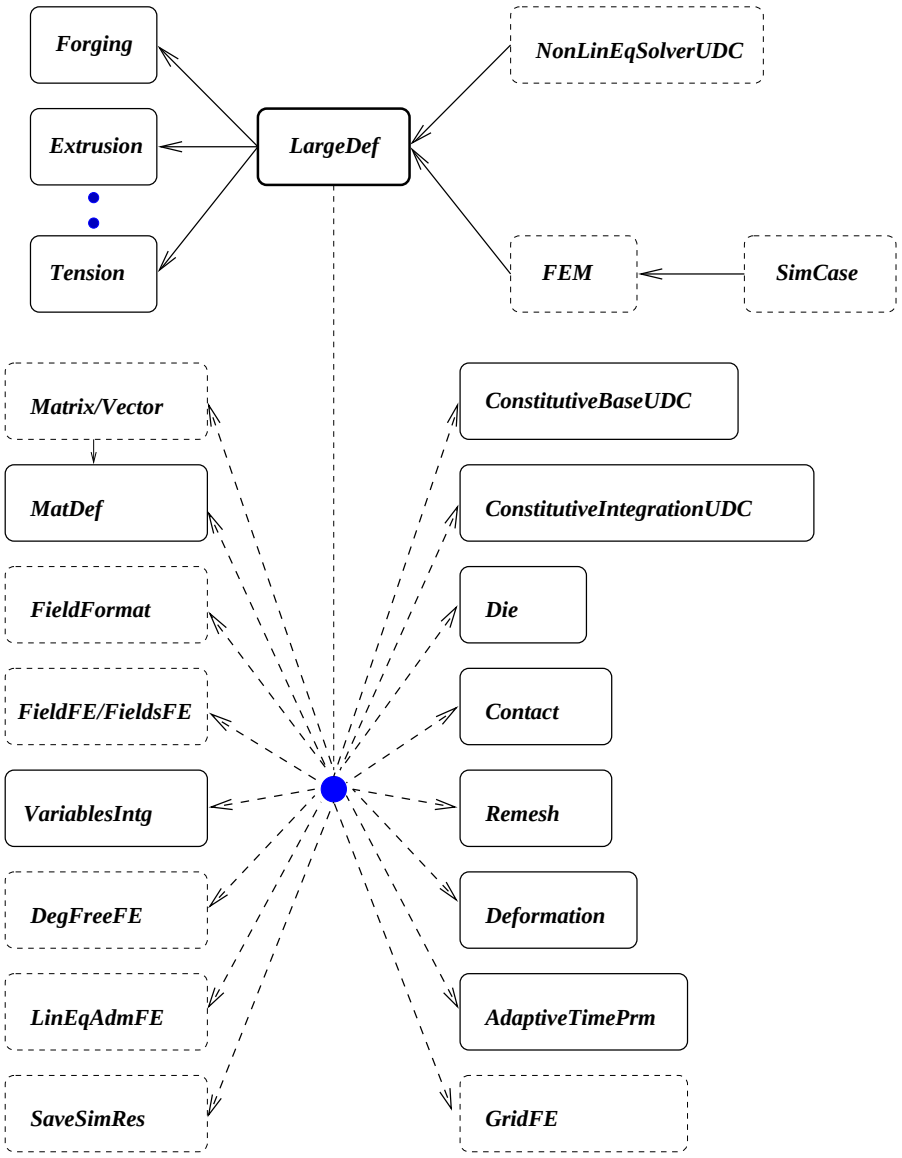

Figure 1.3: A schematic of the finite deformation class `LargeDef`. In this and the following class schematics, the solid arrow lines indicate an `is-a` relation, whereas the dashed arrow lines represent a `has-a` relation. Classes within solid line boxes are particular to this simulator, whereas classes within dotted line boxes are part of the Diffpack libraries.

The classes `ConstitutiveBaseUDC` and `ConstitutiveIntegrationUDC` were introduced to describe the general functionality of inelastic constitutive models and of integration schemes for such models, respectively. The generic structure of the class `ConstitutiveBaseUDC` is shown in Figure 1.4. Its member classes `ElasticUDC`, `PStrainUDC`, `StateEvolveUDC` and `DamageUDC` are introduced to define the elastic properties (equation (1.13)), the evolution of the tensile plastic strain (equation (1.10)), the evolution of state (equations (1.11) and (1.12)) and the functional form of the damage potential (equation (1.7)), respectively. As shown in Figure 1.4, the actual material properties are defined in classes derived from the above base classes.

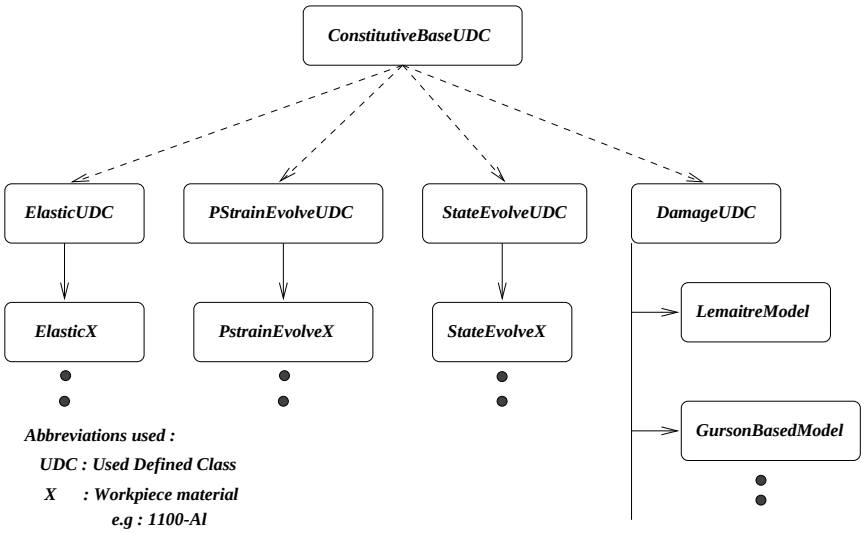

Figure 1.4: A schematic of the base class `ConstitutiveBaseUDC` and its members `ElasticUDC`, `PStrainUDC`, `StateEvolveUDC` and `DamageUDC`. A `String X` is introduced to identify each constitutive model, with `X` usually taken to be the name of the particular model. For example, the classes `ElasticX`, `PStrainX`, `StateEvolveX` define the constitutive model `X`. The `String X` is used to bind pointers to the base classes with objects of the derived classes where the specific functionality of each constitutive model is defined. Note that a particular constitutive model with given thermoelastic properties and evolution laws for the tensile plastic strain and state variables can be combined with various type of damage models (e.g. a Gurson based model or the Lemaitre model).

Radial-return like algorithms have currently been implemented based on a generic structure provided in the class `ConstitutiveIntegrationUDC`. In derived classes, the constitutive integration algorithms are provided for

isotropic rate-dependent and rate-independent models with and without temperature/damage effects [19]–[21].

The base class `Die` provides the parametrization of the die surface, whereas the class `Contact` defines the algorithm for the integration of the contact problem [19]. Dimension-specific and process-specific dies are defined in hierarchies derived from the base classes `Die` and `Contact`.

Various aspects of the non-linear kinematics are accounted for in class `Deformation`. This class defines the relation between the deformation gradient and the incremental displacements given the reference grid and it also provides the functionality for the calculation of the deformation gradient given the reference and current (deformed) grids.

A matrix class `MatDef(real)` has been introduced to represent tensors (e.g. the deformation gradient at a Gauss point). This matrix class allows various common tensor operations (e.g. the `RU` decomposition for the deformation gradient, tensor algebra operations, etc.) [19]. The representation of the various tensor variables (stress tensor, rotation and stretch tensors, plastic deformation gradient, etc.) that appear in the linearized form of the principle of virtual work as `MatDef` objects facilitates the calculation of the tangent stiffness and the formation of a linear system of equations for the incremental displacements.

The calculation of the tangent stiffness is performed with the virtual function `integrands()` inherited from the basic Diffpack class `FEM`. Diffpack provides the functionality for essential boundary conditions and here the problem-dependent information is provided in derived classes (e.g. `Forging`, etc.).

To model coupled thermomechanical processes, a simulator `ThermalLargeDef` has been developed with `LargeDef` and `NonLinearHeat` as members. The present heat transfer analysis implemented in `NonLinearHeat` provides the integration of equation (1.14). The thermal/deformation coupling arises mainly from the heat source given by equation (1.15), the temperature dependence of the mechanical response, and the change in geometry due to deformation (which may lead to a change in the nature of the applied thermal boundary conditions). As we discussed earlier, process-dependent forming conditions are implemented in classes derived from `LargeDef` (e.g. `Tension`, `Forging`, `Extrusion`, `Rolling`, etc.). Similarly, thermal process-dependent conditions are implemented in classes derived from `NonLinearHeat`. These hierarchies are shown in Figure 1.5.

The representation of continuous variables as `FieldsFE` objects is essential in the development of the present deformation simulator [10]. Such fields provide the nodal field values in a given finite element mesh but also allow the use of various postprocessing operations available in Diffpack as well as easy data transfers after remeshing or when restarting a previously interrupted run.

The use of `smart pointers` is an essential tool for memory binding. The class `Handle` is used widely in Diffpack to provide intelligent pointers

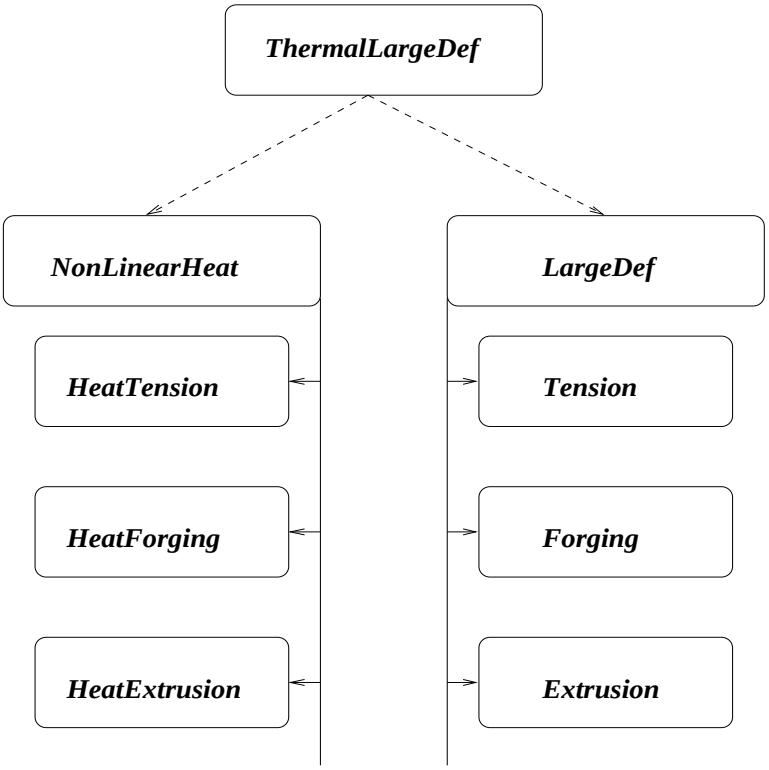

Figure 1.5: The thermomechanical simulator.

with reference counting [10]. This class allows the automatic deletion of an object only when all pointers that are bound to this particular object have been deleted. This is important since in large scale simulators, it is customary to have more than one pointer bound to a given object. As an example, the pointers to the objects representing the plastic work rate at each Gauss point in the deformation and heat transfer simulators point to the same point in memory. Thus duplication of memory is avoided and changing the plastic work in the deformation simulator implies that the same changes will be in effect in the thermal analysis. This pointer binding process is applied extensively in the developed simulator. For more details, [19] provides a number of specific examples.

Finally, as in most Lagrangian large-deformation simulators, `LargeDef` provides functionality for remeshing (class `Remesh`), for transferring data (in a `FieldFE` format) between different finite element grids (class `VariablesIntg`), for automatic time stepping (class `AdaptiveTimeStep`) and other (see Figure 1.3). The interested reader should consult [19]–[21] for further details.

1.2.4 An analysis of a plane strain closed-die forging process

An example problem is presented here to briefly demonstrate the ability of the developed OOP simulator is modeling forming problems of industrial importance. For the detailed implementation of various forming processes including CPU requirements of the present simulator see [19]–[21].

The influence of damage on macroscopic deformation characteristics is shown in this axisymmetric isothermal closed die forging example of an 1100-Al cylindrical workpiece. The workpiece is assumed to be initially voided with $f_o = 0.05$. An effective forging process requires that these existing voids be eliminated if the forged product is to achieve reasonable mechanical property levels.

Unlike conventional fully-dense materials, the plastic deformation of a voided material undergoes volume change. As such, the macroscopic deformation response of voided materials poses practical challenges in the design of *flashless* closed die forging processes.

The current process is analyzed for 2 preforms. The cylindrical preforms are chosen such that the initial volume equals that of the die cavity. Preform A has an initial radius of 6.31 mm and an initial height of 10.0 mm. Preform B has an initial radius of 7.0 mm and an initial height of 8.12 mm. A realistic friction coefficient of 0.1 is assumed at the die-workpiece interface and a nominal strain rate of 0.01 is applied during the forging process. Due to the symmetry of the problem, only one-fourth of the geometry is modelled.

The deformed mesh at the final stage is shown together with the initial mesh for preform A and preform B in Figures 1.6 and 1.7, respectively. It is evident from Figures 1.6 and 1.7 that there is a significant overall decrease in the volume of the workpiece as a result of void closure and the workpiece does not fill up the die cavity exactly. An accurate estimate of this volume change, which is process-dependent, is necessary for the design of *flashless* closed die forging processes.

The force required to forge the product for both the preforms is shown in Figure 1.8. The void fraction distribution in the final product for preform A and preform B are shown in Figures 1.9 and 1.10, respectively. It is evident from Figure 1.8 that the required work in obtaining the final product is significantly smaller for preform B than for preform A. However, the void fraction distribution in the forged product indicates that for preform A, the final void fraction has decreased significantly in all the regions of the workpiece. This was not the case with preform B with some regions in the outer radii of the forged product showing an increase in void fraction of about 6%.

The pressure contours at the final stage of deformation for preform A and preform B are shown in Figures 1.11 and 1.12, respectively and are quite different from each other. The results clearly indicate the important role

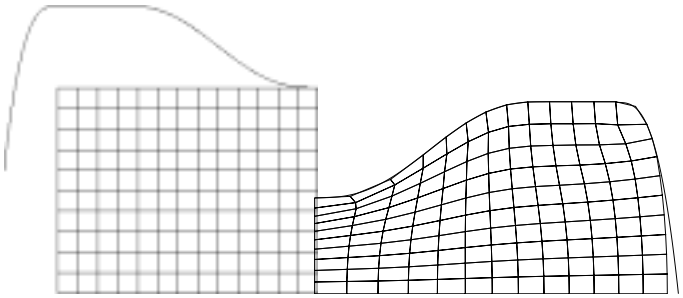

Figure 1.6: Initial mesh and final mesh for the axisymmetric closed die forging of a cylindrical billet for preform A.

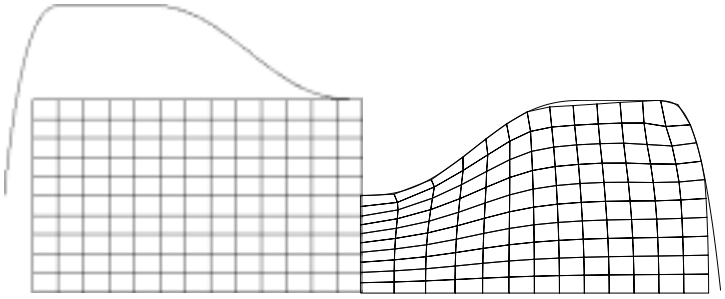

Figure 1.7: Initial mesh and final mesh for the axisymmetric closed die forging of a cylindrical billet for preform B.

that the preform geometry plays in determining the quality of the forged product. As a result of the change in the initial geometry, the workpiece was subjected to a different contact history and the damage distribution in the final product was significantly different.

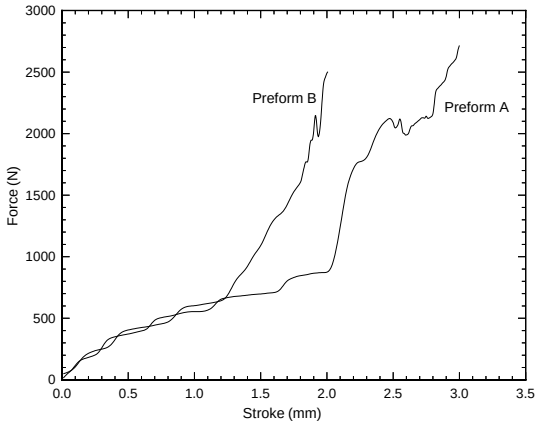

Figure 1.8: Force required to forge preforms A and B.

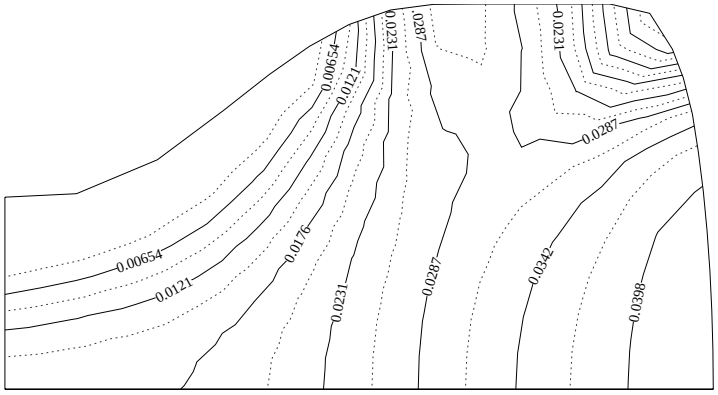

Figure 1.9: Void fraction distribution in the forged product for preform A.

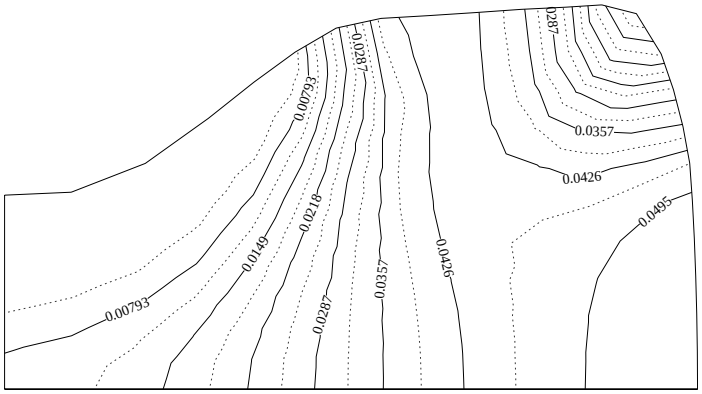

Figure 1.10: Void fraction distribution in the forged product for preform B.

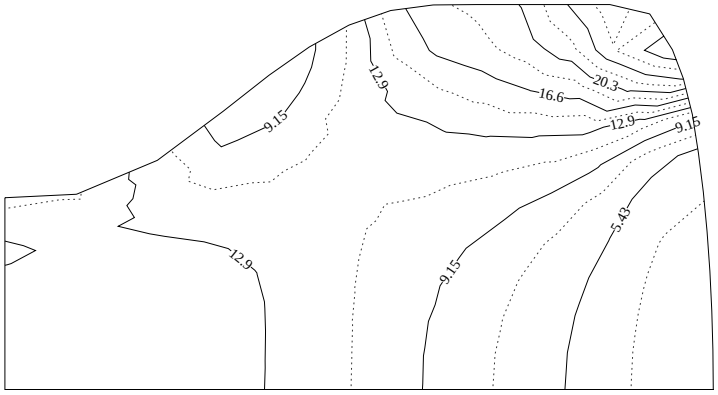

Figure 1.11: Pressure contours at the final stage of deformation for preform A.

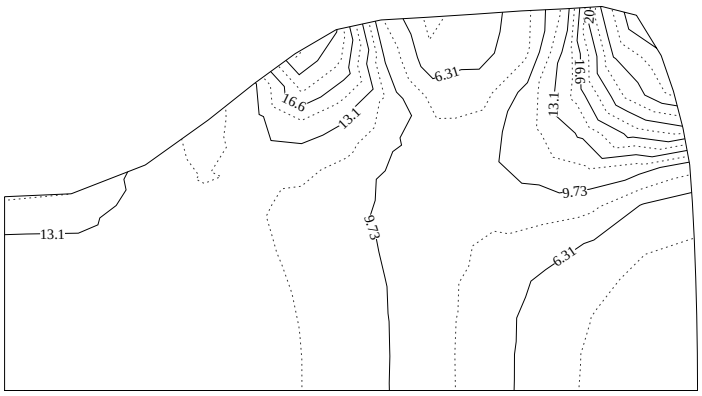

Figure 1.12: Pressure contours at the final stage of deformation for preform B.

1.3 Analysis and design of directional solidification processes

Some *direct* binary alloy directional solidification problems are introduced here as well as a *design* solidification problem. In particular, Section 1.3.1 presents the direct problem definition. Section 1.3.2 briefly discusses a front tracking FEM modeling of direct solidification problems. In Section 1.3.3, the common structure of the various subproblems of the direct problem (e.g. of the heat transfer and melt flow subproblems) is identified. An appropriate class hierarchy is then introduced for the representation of the subproblems.

Such hierarchies are shown in Section 1.4 to be very efficient tools for the analysis of design solidification problems in which the boundary cooling/heating is calculated such that desired growth conditions at the freezing front are achieved. In particular, Section 1.4.1 introduces the definition of a design solidification problem, Section 1.4.2 presents an adjoint method formulation using an infinite dimensional optimization scheme and Section 1.4.3 presents the OOP structure of the developed design simulator. Section 1.4.4 presents the solution of a design problem in which the boundary heat fluxes are calculated such that a desired freezing front velocity and interface heat fluxes are obtained in the solidification of a pure material in the presence of a magnetic field.

1.3.1 Problem description

A binary alloy melt is considered at a uniform concentration $\hat{c}_i$ and a uniform temperature $\hat{T}_i$. At time $t = 0^+$, a cooling heat flux is applied at the side Γ_{os} of the mold wall to bring the boundary temperature to the freezing temperature corresponding to concentration $\hat{c}_i$ (see Figure 1.13). Solidification starts and proceeds from Γ_{os} to Γ_{ol}. It takes place in the presence of a strong externally applied magnetic field $\mathbf{B_o}$ which is at an angle γ to the direction of solidification.

The melt flow is initially driven by thermal and solutal gradients which in turn induce density gradients. The flow in the presence of a magnetic field also gives rise to a Laplace force term in the Navier-Stokes equation. It is here assumed that the induced magnetic field is negligible with respect to the imposed constant magnetic field $\mathbf{B_o}$ and that the electric charge density, ρ_e, is small.

To simplify the presentation, only the governing equations in the melt are presented here. Let L be a characteristic length of the domain, ρ the density, k the conductivity, α the thermal diffusivity, D the solute diffusivity, σ the electrical conductivity and ν the kinematic viscosity. All the thermo-physical properties are assumed to be constant. The characteristic scale for time is taken as L^2/α and for velocity as α/L. The dimension-

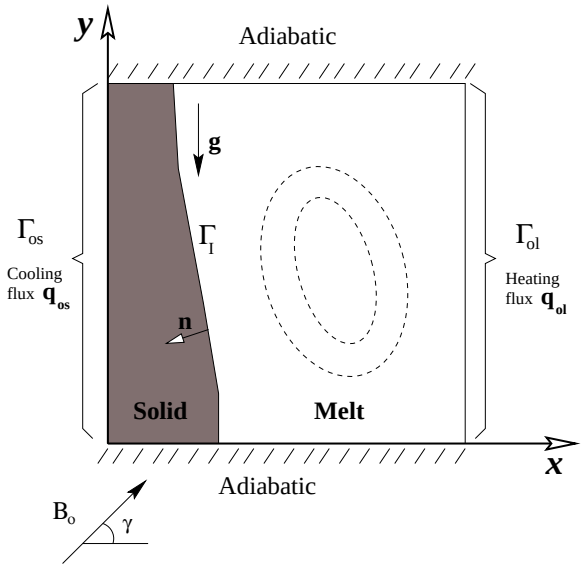

Figure 1.13: Schematic of the direct and design alloy solidification problem. The solid region is denoted as Ω_s and the liquid region as Ω_l. These regions share the interface Γ_I. The region Ω_l has a boundary Γ_l which consists of Γ_I (the solid-liquid interface), Γ_{ol} and the remaining boundary Γ_{hl} (the top and bottom mold walls). Similarly Ω_s has boundary Γ_s, which includes Γ_I, Γ_{os} and the top and bottom mold boundary Γ_{hs}.

less temperature θ is taken as $\theta = (\hat{T} - T_o)/\Delta T$ where $\hat{T}$, T_o and ΔT are the temperature, reference temperature and reference temperature drop, respectively. Likewise, the dimensionless concentration field c is defined as $(\hat{c} - c_o)/\Delta c$ where $\hat{c}$, c_o and Δc are the concentration, reference concentration and reference concentration drop, respectively. The dimensionless form m of the slop of the liquidus line, is taken as $\hat{m}\Delta c/\Delta T$, whereas the dimensionless form of the melting point of the pure material T_m is defined as $\theta_m = (\hat{T}_m - T_o + \hat{m}c_o)/\Delta T$. The characteristic scale for the electric potential is taken as αB_o. The symbol ϕ is here used to denote the nondimensional electric potential. Other dimensionless parameters used later in this paper include the partition coefficient κ and the parameter $\delta = c_o/\Delta c$ [34].

The key dimensionless quantities are the Prandtl number ($\mathrm{Pr} = \nu/\alpha$), the Lewis number ($\mathrm{Le} = \alpha/D$), the Stefan number ($\mathrm{Ste} = (c\Delta T_o)/L_H$, with L_H the latent heat), the thermal Rayleigh number ($\mathrm{Ra_T} = g\beta_T \Delta T L^3/\nu\alpha$, with β_T the thermal expansion coefficient), the solutal Rayleigh number ($\mathrm{Ra_c} = g\beta_c \Delta c L^3/\nu\alpha$, with β_c the solutal expansion coefficient), and the

Hartmann number ($\mathrm{Ha} = \left[\left(\frac{\sigma}{\rho\nu}\right)^{1/2} B_o L\right]$) (see [34] for further review of these definitions). The governing PDEs are summarized in a dimensionless form in Box 1 together with typical initial conditions for the field variables. In addition to these equations, one must consider the solute balance at the freezing front, the Stefan condition, the equilibrium condition at the front relating the temperature and concentration (via the liquidus line of the phase diagram) and appropriate boundary conditions for all field variables.

The system of PDEs in the melt has to be solved simultaneously with the heat conduction problem in the solid phase to calculate the main field variables $(\theta, c, \boldsymbol{v}, \phi)$ as well as the location of the freezing front. More details on the definition of directional alloy solidification problems can be found in [11] and [34]. Issues related to the physical feasibility of such processes with an a-priori assumed stable growth (i.e. with a sharp interface) are discussed in [17] and [34].

For reasons to be explained later, the heat fluxes q_{os} and q_{ol} are considered as parameters to this direct problem. In particular, Box 1 shows the governing PDEs in the melt considering the heat flux $q_{ol}(\boldsymbol{x}, t)$, $(\boldsymbol{x}, t) \in \Gamma_{ol} \times [0, t_{max}]$ as a (functional) parameter.

$$\frac{\partial \theta}{\partial t} + \boldsymbol{v} \cdot \nabla \theta = \nabla^2 \theta \tag{1.16}$$

$$\begin{aligned} \frac{\partial \boldsymbol{v}}{\partial t} + (\nabla \boldsymbol{v})\ \boldsymbol{v} &= -\nabla p + \mathrm{Pr}\nabla^2 \boldsymbol{v} - \mathrm{Ra_T Pr}\theta \boldsymbol{e_g} + \mathrm{Ra_c Pr} c \boldsymbol{e_g} \\ &+ \mathrm{Ha}^2 \mathrm{Pr}\left[-\nabla\phi + \boldsymbol{v} \times \mathbf{e_B}\right] \times \mathbf{e_B} \end{aligned} \tag{1.17}$$

$$\nabla \cdot \boldsymbol{v} = 0 \tag{1.18}$$

$$\frac{\partial c}{\partial t} + \boldsymbol{v} \cdot \nabla c = Le^{-1} \nabla^2 c \tag{1.19}$$

$$\nabla^2 \phi = \nabla \cdot (\boldsymbol{v} \times \mathbf{e_B}) \tag{1.20}$$

$$\theta(\boldsymbol{x}, 0) = \theta_i, \quad c(\boldsymbol{x}, 0) = c_i, \quad \boldsymbol{v}(\boldsymbol{x}, 0) = \mathbf{0}, \quad \boldsymbol{x} \in \Omega_l(0) \tag{1.21}$$

Box 1: The governing PDEs and initial conditions for the direct problem in the melt in terms of $\theta(\boldsymbol{x}, t; q_o)$, $\boldsymbol{v}(\boldsymbol{x}, t; q_o)$, $c(\boldsymbol{x}, t; q_o)$ and $\phi(\boldsymbol{x}, t; q_o)$, with $(\boldsymbol{x}, t) \in \Omega_l(t) \times [0, t_{max}]$. Also, $\mathbf{e_g}$ and $\mathbf{e_B}$ denote the unit vectors in the direction of gravity and magnetic field, respectively.

1.3.2 FEM algorithm

From the problem statement given in Box 1, it becomes clear that the overall direct solidification problem can be considered as the coupling of four main PDE based subproblems (the flow, heat diffusion/convection, mass diffusion/convection and the magnetic field potential problem). In addition, the motion of the freezing interface must be considered in the analysis.

Let us first recognize that the three transport problems have the same convection/diffusion form. Any algorithm addressing such convection/diffusion problems can be applied for the solution of these three problems. A Petrov-Galerkin analysis with the consistent penalty method was applied in [11] to produce a spatially discretized weak form for such problems. Other techniques can be applied as well [35]–[37].

Most FEM techniques for advection/diffusion problems finally lead to a system of discretized equations of the following form:

$$[\mathbf{M}]\{\dot{\alpha}\} + [\mathbf{C}]\{\alpha\} = \{F\} \tag{1.22}$$

where $\{\alpha\}$ is the vector of the unknown degrees of freedom (nodal velocity components). Various time integration schemes can be applied for the solution of the above ODE. A predictor-corrector scheme was used in [11] following the work of [38]. Other time stepping techniques can for example be found in [35].

The solution of the electric potential subproblem is not discussed in any detail here as a simple Galerkin formulation can be applied once the velocity field $\boldsymbol{v}$ is calculated.

The various time sequential algorithms for each subproblem (the flow, heat and solute diffusion/convection and electromagnetics subproblems) must be coupled within each time step. The calculation of the freezing front motion is an integrable part of the overall algorithm [11]. In this work, a deforming FEM technique is used to track the freezing interface. In such formulations, one must account for the apparent convection resulting from the mesh motion. More on the selected mesh motion for directional solidification problems can be found in [11].

1.3.3 An OOP approach to the FEM analysis of solidification processes

In the previous section, three main convection/diffusion subproblems were identified that in a spatially discretized form fit in the category of the algebraic system given in the model equation (1.22). A general abstract base class `ConvectionDiffusion` is defined to manage the solution process for the model scalar-convection diffusion equation. The general structure of the class `ConvectionDiffusion` is shown in Box 2. Further information

for object-oriented programming of convection/diffusion problems can be found in [39]–[43].

The above development of an abstract base class that solves a general model equation helps in *avoiding repetitive codes*. As such all three of the above diffusion/convection subproblems can be described by the `ConvectionDiffusion` class introduced here. The main differences between the heat transfer, solute transport and melt flow mechanisms are in the coefficients of the corresponding PDEs as well as in the initial and boundary conditions. This information particular to each PDE can be introduced using the virtual functions `integrands()`, `setIC()` and `essBC`, respectively [10]. For example, the virtual function `FEM:integrands()` is used to define the weak Petrov-Galerkin form of each PDE and perform the stiffness and load calculations at a particular Gauss point. These details are obvious and will not be further discussed here.

Experimentation with a different integration scheme for equation (1.22) will only require the development of a new derived class from `ConvectionDiffusion` where the particulars of the new algorithm are implemented. The binding of a pointer to an object of the base class `ConvectionDiffusion` to the particular algorithm is performed at run time. The particular predictor/corrector scheme used in [11] is here implemented with a derived class `PredictorCorrector`.

To further emphasize the importance of recognizing the common structure of the three convection/diffusion subproblems of Box 1, we will next present the definition, solution algorithm and OOP implementation of an inverse design directional solidification problem.

```
class ConvectionDiffusion: public ... {
  protected:
    Handle(GridFE)         grid;              // finite element grid
    Handle(DegFreeFE)      dof;               // mapping of nodal values <->linear system
    Handle(TimePrm)        tip;               // time loop parameters
    Handle(FieldsFE)       alpha;             // FE field, the primary unknown
    Handle(FieldsFE)       alpha_prev;        // solution in the previous time step
    Handle(FieldsFE)       alpha_dot;         // rate field
    Handle(FieldsFE)       alpha_dot_prev;    // rate field in the previous time step
    real                   gamma;             // time integration parameter
    real                   err_tolerance;     // error tolerance for the nonlinear solver
    ...
    Vec(real)              linear_solution;   // solution of the linear subsystem
    Handle(LinEqAdmFE)     lineq;             // linear system and solution
    ...
    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);
    virtual void setIC() = 0;                      // initial conditions
    virtual real essBC(int nno,
                       const Ptv(real)& x) = 0;    // essential boundary conditions
  public:
    ...
    virtual void define(MenuSystem& menu, int level = MAIN);
    virtual void scan(MenuSystem& menu);
    virtual void adm(MenuSystem& menu);
    virtual void solveProblem();
    virtual void storeResults() = 0;
    virtual void updateDataStructures();
};
```

Box 2: The main public and protected members of the class `ConvectionDiffusion` introduced for carrying out program administration and the various numerical steps in the algorithms.

1.4 On the optimum design of directional solidification processes

A typical inverse design problem is one governed by a system of PDEs with insufficient or no boundary conditions in part of the boundary but overspecified conditions in another part of the boundary or within the domain. These problems are generally ill-posed and a quasi-solution is being sought in some optimization sense [16]. The type of approach employed here for the solution of such inverse problems is the so called *iterative regularization technique* [44], [45]. In simple terms, the problem is formulated as a functional optimization problem over the whole time and space domain [16]. For example, in an inverse heat transfer problem, the overspecified data in the part Γ_I of the boundary Γ can be the temperature θ_I and heat flux q_I, whereas the heat flux q_o (and temperature) are unknown in another part Γ_o of the boundary. The cost function in this example is defined as the error $||\theta(\boldsymbol{x}, t; q_o) - \theta_I||^2_{L_2(\Gamma_I \times [0, t_{max}])}$ between the computed (via a well-posed direct analysis) solution $\theta(\boldsymbol{x}, t; q_o)$ using q_o and q_I as boundary conditions on Γ_o and Γ_I, respectively, and the given temperature θ_I on Γ_I. The gradient of such cost functionals is calculated with a *continuum adjoint problem*, whereas the step size in each optimization step requires the solution of a *continuum sensitivity problem*. In addition to these problems, the solution of the well-posed direct problem is required as well.

1.4.1 Thermal design of binary alloy solidification to achieve a stable desired growth

The design solidification problem of interest here can be stated as follows:

> *Find the boundary heating/cooling fluxes on the left and right vertical walls of Figure* 1.13 *such that a stable desired interface growth is achieved.*

Similar design problems can be stated in which the path in the state space $G - \boldsymbol{v}_f$ is defined [46], [47] (here G denotes the interface flux and $\boldsymbol{v}_f$ the growth velocity). Since the growth velocity and the interface heat fluxes control the obtained solidification microstructures, such problems can be used to design solidification processes that lead to desired properties in the cast product [46], [47], [17], [34].

The overspecified conditions in the present inverse design problems are defined at the freezing front, whereas no thermal conditions are available on part of the outside boundary (boundaries Γ_{ol} and Γ_{os} (see Figure 1.13)). The two overspecified conditions on the interface include the growth velocity $\boldsymbol{v}_f$ and the stability condition. The last condition has been interpreted in [34] as a relation between the thermal gradient and the solute gradient at the interface. In particular, the following form of the stability condition

is considered [34]:

$$\frac{\partial \theta}{\partial n}(\boldsymbol{x}, t; q_{ol}) = m \frac{\partial c}{\partial n}(\boldsymbol{x}, t; q_{ol}) + \epsilon(\boldsymbol{x}, t), \qquad (\boldsymbol{x}, t) \in \Gamma_I(t) \times [0, t_{max}] \quad (1.23)$$

Note that the vector $\boldsymbol{n}$ used above to define the normal derivatives at the interface points towards the solid region (see Figure 1.13).

There are various possibilities for the selection of the parameter $\epsilon \leq 0$ [17] and [34]. Equation (1.23) is the dimensionless form of the well known constitutional undercooling stability condition [46] and the parameter ϵ can here be interpreted as the level of 'over-stability' desired in the solidification system.

In addition to the stability condition and $\boldsymbol{v}_f$, the solute balance and the temperature/concentration equilibrium relation are also provided at the freezing front. The overall design problem can now be considered with the solution of two inverse problems, one in the solid phase and another in the liquid melt.

The optimization scheme for the inverse melt flow problem is briefly presented next. The unknown heat flux q_{ol} on the boundary Γ_{ol} is assumed to be an $L_2(\Gamma_{0l} \times [0, t_{max}])$ function. Using part of the overspecified data on the freezing front and for a guess value of the unknown heat flux q_{ol}, one can solve a direct problem with the PDEs and initial conditions of Box 1 and the boundary/interface conditions given in Box 3.

In particular, the conditions considered at the freezing front as part of this direct problem are the desired front velocity $\boldsymbol{v}_f$, the stability condition given by equation (1.23) or its equivalent form (1.27) and the solute balance condition (equation (1.29)). The solution of this direct problem provides the values of the temperature and concentration fields at the freezing front. The discrepancy of these values from the ones corresponding to thermodynamic equilibrium is the driving force for the optimization algorithm.

$$\boldsymbol{v}(\boldsymbol{x},t;q_{ol}) = \boldsymbol{0}, \quad \boldsymbol{x} \in \Gamma_l(t) \tag{1.24}$$

$$\frac{\partial\theta}{\partial n}(\boldsymbol{x},t;q_{ol}) = 0, \quad \boldsymbol{x} \in \Gamma_{hl}(t) \tag{1.25}$$

$$\frac{\partial\theta}{\partial n}(\boldsymbol{x},t;q_{ol}) = q_{ol}(\boldsymbol{x},t), \quad \boldsymbol{x} \in \Gamma_{ol}(t) \tag{1.26}$$

$$\frac{\partial\theta}{\partial n}(\boldsymbol{x},t;q_{ol}) = \epsilon(\boldsymbol{x},t) + mLe(\kappa-1)\boldsymbol{v}_f \cdot \boldsymbol{n}(c(\boldsymbol{x},t;q_{ol}) + \delta), \ \boldsymbol{x} \in \Gamma_I(t) \tag{1.27}$$

$$\frac{\partial c}{\partial n}(\boldsymbol{x},t;q_{ol}) = 0, \quad \boldsymbol{x} \in \Gamma_{hl}(t) \cup \Gamma_{ol}(t) \tag{1.28}$$

$$\frac{\partial c}{\partial n}(\boldsymbol{x},t;q_{ol}) = Le(\kappa-1)\boldsymbol{v}_f \cdot \boldsymbol{n}(c(\boldsymbol{x},t;q_{ol}) + \delta), \quad \boldsymbol{x} \in \Gamma_I(t) \tag{1.29}$$

$$\frac{\partial\phi}{\partial n}(\boldsymbol{x},t;q_{ol}) = 0, \quad \boldsymbol{x} \in \Gamma_l(t) \tag{1.30}$$

Box 3: The boundary conditions given here together with the equations of Box 1 define $\theta(\boldsymbol{x},t;q_{ol})$, $\boldsymbol{v}(\boldsymbol{x},t;q_{ol})$, $c(\boldsymbol{x},t;q_{ol})$ and $\phi(\boldsymbol{x},t;q_{ol})$ for $t \in [0,t_{max}]$.

For an arbitrary q_{ol}, the following cost functional is defined:

$$J(q_{ol}) = \frac{1}{2}||\theta(\boldsymbol{x},t;q_{ol}) - (\theta_m + mc(\boldsymbol{x},t;q_{ol}))||^2_{L_2(\Gamma_I\times[0,t_{max}])}, \tag{1.31}$$

to indicate the discrepancy of the calculated temperature (using the direct problem defined in Boxes 1 and 3) from the concentration-dependent liquidus temperature at the interface. Finally, the *control problem* of interest is defined as follows:

Find $\bar{q}_{ol} \in L_2(\Gamma_{0l} \times [0,t_{max}])$ such that

$$J(\bar{q}_{ol}) \leq J(q_{ol}), \forall q_{ol} \in L_2(\Gamma_{0l} \times [0,t_{max}]) \tag{1.32}$$

In this paper, our objective is to construct a minimizing sequence $q^k_{ol}(\boldsymbol{x},t) \in L_2(\Gamma_{0l} \times [0,t_{max}])$, $k = 1,2,\ldots$ that converges to at least a local minimum of $J(q_{ol})$.

1.4.2 The adjoint method

The main difficulty with the above optimization problem is the calculation of the gradient $J'(q_{ol})(\boldsymbol{x},t)$ in the space $L_2(\Gamma_{0l}\times[0,t_{max}])$. Taking the directional derivative of the cost functional with respect to q_{ol} in the direction of the variation Δq_{ol} yields the following:

$$D_{\Delta q_{ol}}J(q_{ol}) = (J'(q_{ol}),\Delta q_{ol})_{\mathcal{U}} = (\Theta - mC, R)_{L_2(\Gamma_I\times[0,t_{max}])} \tag{1.33}$$

where the residual temperature field R is defined as $R(\boldsymbol{x},t;q_{ol}) \equiv \theta(\boldsymbol{x},t;q_{ol}) - (\theta_m + mc(\boldsymbol{x},t;q_{ol}))$. The sensitivity fields corresponding to the magnetic potential, temperature, melt velocity and concentration fields are here denoted as $\Phi(\boldsymbol{x},t;q_{ol},\Delta q_{ol})$, $\Theta(\boldsymbol{x},t;q_{ol},\Delta q_{ol})$, $\boldsymbol{V}(\boldsymbol{x},t;q_{ol},\Delta q_{ol})$ and $C(\boldsymbol{x},t;q_{ol},\Delta q_{ol})$, respectively. These sensitivity fields are defined as directional derivatives of the direct fields calculated at q_{ol} in the direction Δq_{ol}. Due to the form of equation (1.33), it is easier from now on to work with the variable $R(\boldsymbol{x},t;q_{ol})$ instead of the temperature field θ. The Boxes 1 and 3 can be modified appropriately by expressing all equations in terms of $R, \boldsymbol{v}, c$ and ϕ. The sensitivity field corresponding to R is here denoted as $\Re(\boldsymbol{x},t;q_{ol},\Delta q_{ol})$. The governing PDEs for the sensitivity problem are given in Box 4.

$$\frac{\partial \Re}{\partial t} + \boldsymbol{v}\cdot\nabla\Re + \boldsymbol{V}\cdot\nabla R = \nabla^2\Re + \mathrm{m}(1-\mathrm{Le}^{-1})\nabla^2 C \tag{1.34}$$

$$\frac{\partial \boldsymbol{V}}{\partial t} + (\nabla\boldsymbol{v})\,\boldsymbol{V} + (\nabla\boldsymbol{V})\,\boldsymbol{v} = -\nabla\Pi + \mathrm{Pr}\nabla^2\boldsymbol{V} - \mathrm{Ra_T Pr}\Re\boldsymbol{e_g}$$
$$-\mathrm{Ra_T Pr}\left(\mathrm{m} - \frac{\mathrm{Ra_c}}{\mathrm{Ra_T}}\right)C\boldsymbol{e_g} + \mathrm{Ha}^2\mathrm{Pr}[-\nabla\Phi + \boldsymbol{V}\times\mathbf{e_B}]\times\mathbf{e_B} \tag{1.35}$$

$$\nabla\cdot\boldsymbol{V} = 0 \tag{1.36}$$

$$\frac{\partial C}{\partial t} + \boldsymbol{v}\cdot\nabla C + \boldsymbol{V}\cdot\nabla c = \mathrm{Le}^{-1}\nabla^2 C \tag{1.37}$$

$$\nabla^2\Phi = \nabla\cdot(\boldsymbol{V}\times\mathbf{e_B}) \tag{1.38}$$

Box 4: PDEs governing the sensitivity problem that defines $\Re(\boldsymbol{x},t;q_{ol},\Delta q_{ol})$, $\boldsymbol{V}(\boldsymbol{x},t;q_{ol},\Delta q_{ol})$, $C(\boldsymbol{x},t;q_{ol},\Delta q_{ol})$ and $\Phi(\boldsymbol{x},t;q_{ol},\Delta q_{ol})$, with $(\boldsymbol{x},t)\in\Omega_l(t)\times[0,t_{max}]$.

It is clear from equation (1.33) that the calculation of the gradient of the cost functional requires the evaluation of the adjoint field $\psi(\boldsymbol{x},t;q_{ol})$ corresponding to the sensitivity field $\Re(\boldsymbol{x},t;q_{ol},\Delta q_{ol})$. In the process of evaluating ψ, one must also obtain the corresponding adjoint fields for melt velocity, electric field and concentration here denoted as $\boldsymbol{\phi}(\boldsymbol{x},t;q_{ol})$, $\eta(\boldsymbol{x},t;q_{ol})$ and $\rho(\boldsymbol{x},t;q_{ol})$, respectively. The governing PDEs for the adjoint

problem are shown in Box 5 (for the derivation of these equations see [34]). In addition to the dimensionless parameters introduced earlier, $\bar{\mathrm{Ra}}_\mathrm{c}$ and χ are here defined as $\bar{\mathrm{Ra}}_\mathrm{c} = \mathrm{Ra}_\mathrm{T}\left[\mathrm{m} - \frac{\mathrm{Ra}_\mathrm{c}}{\mathrm{Ra}_\mathrm{T}}\right]$ and $\chi = \mathrm{m}\frac{\mathrm{Ra}_\mathrm{T}}{\bar{\mathrm{Ra}}_\mathrm{c}}(\mathrm{Le}^{-1} - 1)$, respectively [34].

$$\frac{\partial \psi}{\partial t} + \boldsymbol{v} \cdot \nabla \psi = -\nabla^2 \psi + \boldsymbol{\phi} \cdot \boldsymbol{e_g} \tag{1.39}$$

$$\begin{aligned} \frac{\partial \boldsymbol{\phi}}{\partial t} \quad &+ \quad (\nabla \boldsymbol{\phi})\, \boldsymbol{v} - (\nabla \boldsymbol{v})^T \boldsymbol{\phi} = \nabla \pi - Pr \nabla^2 \boldsymbol{\phi} - Ha^2 Pr[\mathbf{e_B}(\boldsymbol{\phi} \cdot \mathbf{e_B}) - \boldsymbol{\phi}] \\ &+ \quad Pr Ra_T \psi \nabla R + Pr \bar{Ra}_c \rho \nabla c + Ha^2 Pr[\nabla \eta \times \mathbf{e_B}] \end{aligned} \tag{1.40}$$

$$\nabla \cdot \boldsymbol{\phi} = 0 \tag{1.41}$$

$$\nabla^2 \eta = \nabla \cdot (\boldsymbol{\phi} \times \mathbf{e_B}) \tag{1.42}$$

$$\frac{\partial \rho}{\partial t} + \boldsymbol{v} \cdot \nabla \rho(\boldsymbol{x}, t; q_{ol}) = -Le^{-1} \nabla^2 \rho + \boldsymbol{\phi} \cdot \boldsymbol{e_g} + \chi \nabla^2 \psi \tag{1.43}$$

Box 5: PDEs governing the adjoint problem that defines $\psi(\boldsymbol{x}, t; q_{ol})$, $\boldsymbol{\phi}(\boldsymbol{x}, t; q_{ol})$, $\eta(\boldsymbol{x}, t; q_{ol})$ and $\rho(\boldsymbol{x}, t; q_{ol})$, with $(\boldsymbol{x}, t) \in \Omega_l(t) \times [0, t_{max}]$.

It can be shown that the desired gradient $J'(q_{ol})$ of the cost functional $J(q_{ol})$ is given as follows [34]:

$$J'(q_{ol}) = \psi(\boldsymbol{x}, t; q_{ol}), \quad (\boldsymbol{x}, t) \in \Gamma_{ol} \times [0, t_{max}] \tag{1.44}$$

A typical adjoint formulation for an inverse design problem is effectively an infinite dimensional optimization problem that can be approached with a variety of techniques such as the conjugate gradient method [48]. The solution procedure for the particular inverse solidification design problem is shown in Box 6.

In the following section, an elegant technique is presented for the OOP implementation of the algorithm of Box 6 as applied to the above design problem. The ideas here are general and applicable to a variety of optimum continuum design problems.

1.4.3 An OOP approach to the implementation of the adjoint method

The first task in the OOP design of the simulator is to recognize the similarities of the various subproblems. The governing PDEs for the direct, adjoint and sensitivity flow, heat and solute transport problems are of similar nature and fall under the general category of scalar or vector transient convection-diffusion equations. One possible design for the considered

direct, adjoint and sensitivity problems (defined by the collection of 9 diffusion/convection like PDEs plus three simpler steady equations for the electromagnetics) would have been to declare 9 different classes, each one managing its own fields, grids, linear system solvers and nonlinear solution process. Such a strategy would not have taken into account the similarity in the problems involved and would thus involve repeated code. In addition, changes in the solution scheme for the integration of the PDEs as well as changes in the optimization algorithm could not be applied without a significant programming effort. Also, testing of the various algorithms for the adjoint and sensitivity subproblems would not have been easy following such an approach.

Step A: Start with an initial guess q_{0l}^0 for the unknown function. Set $k = 0$.
Step B: Solve the direct problem with the current estimate of
q_{0l} and using part of the overspecified data.
Step C: Solve the adjoint problem.
Step D: Calculate the search direction $\mathbf{p}_k$ in this optimization step.
Step E: Solve the sensitivity problem.
Step F: Calculate the step length α_k in the search direction.
Step G: Update the heat flux $q_{0l}^{k+1} = q_{0l}^k + \alpha_k \mathbf{p}_k$.
Step H: Check if the cost functional is reduced below the desired tolerance.
If not, proceed to another optimization step (i.e. return to step B).
If yes, STOP.

Box 6: A typical gradient algorithm for the solution of an inverse optimum design problem.

We here re-introduce the master class `ConvectionDiffusion` (see Box 2) from which the classes corresponding to each subproblem of the direct, sensitivity and adjoint problems are derived. Next stage in the design of the simulator is to recognize specific differences that distinguish one PDE from another. For example what distinguishes the flow equation from the heat equation? The immediate task is to incorporate those differences in some form into the simulator. The fundamental principle of object-oriented programming requires that these changes should be incorporated without affecting the already existing code and should be flexible enough to allow further changes. In the present simulator the heat, flow and solute solvers of the direct, adjoint and sensitivity problems have been built on these lines. Nine classes, one for each PDE, are declared but they are all derived (or inherited) from the base class `ConvectionDiffusion` and hence contain only the *differences* between the various solvers such as the weak form of each PDE and the corresponding initial/boundary conditions. This con-

struction helps in sharing common features such as the solution process, storing of fields etc. Moreover, this method provides more robustness as the base-class can be tested independently on simpler problems. This construction basically demonstrates the effectiveness of using the concept of *inheritance*.

With the above tasks completed, one needs to address the important process of combining the individual simulators to form the direct, adjoint and sensitivity subproblems. Three subproblem classes are declared (`Direct`, `Adjoint` and `Sensitivity`) to manage the time stepping and whole time domain solution of the three subproblems (see Figure 1.14). Each of these classes has pointers to the corresponding PDE classes and provides the required time stepping functionality. As it is customary with OOP implementation of complex systems, the development of the classes for the direct, sensitivity and adjoint problems was performed and tested in a number of steps. For example, a class hierarchy `Direct`, `DirectAlloy` and `DirectMagnetoAlloy` was implemented to model the direct analysis of natural convection of pure materials, double diffusive convection in alloys, and magneto-convection in alloys, respectively. Such class hierarchies are not shown in Figure 1.14.

Some details of the class `DirectAlloy` are given in Box 7. In addition to the functionality and data of the class `Direct`, the concentration field is added as a protected member of the class `DirectAlloy`. To allow access of an object of the class `DirectAlloy` to the protected members of the class `Direct`, pointers to the flow and heat solvers are also declared as members of the class `DirectAlloy`. These pointers are appropriately bound to the flow and heat objects of the class `Direct`.

An appropriate binding between the various smart pointers (handles) is an essential element of an efficient OOP implementation of such problems. For example, in the class `Direct` (that has two protected data members pointing to objects of the classes `NavierStokes` and `NlHeat`), the velocity field $\boldsymbol{v}$ which is a member of the `NavierStokes` class needs to be passed on as input to the heat transfer modeling class `NlHeat` as this is dictated by the physics involved. This is just an example and there are plenty of such couplings that need to be achieved in such problems. The set of program command lines shown below demonstrates how the above mentioned coupling is achieved:

heat→ v. rebind (flow→alpha());
flow→ T. rebind (heat→alpha());

where `heat` and `flow` are the `handles` (pointers) to `NlHeat` and `NavierStokes` classes, respectively (see Box 2 for the definition of `alpha`).

Also, since the time integration of the direct problem involves solving the heat, flow and solute diffusion subproblems simultaneously and the same being the case for the adjoint and sensitivity problems, the 'time integra-

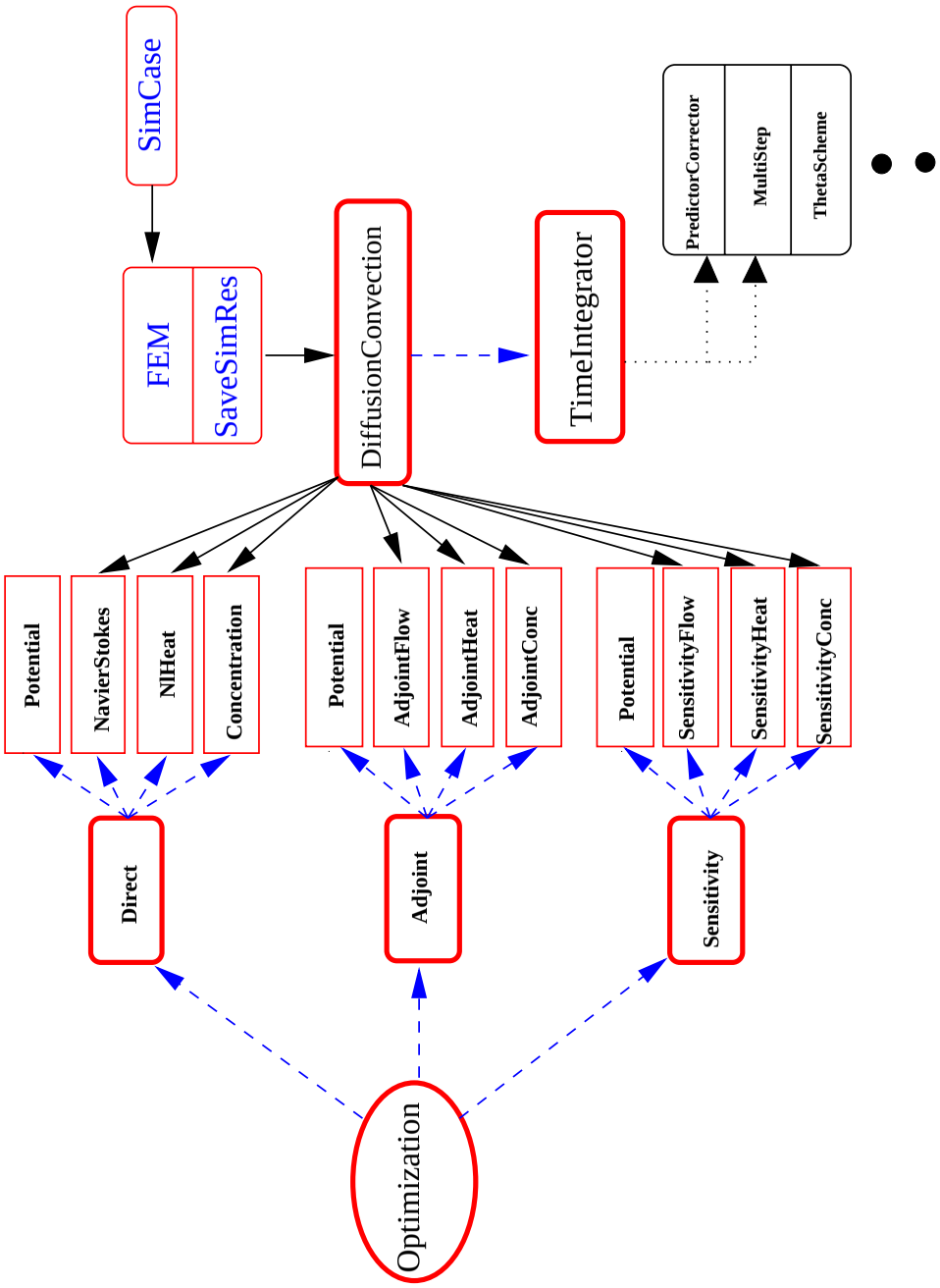

Figure 1.14: A sketch of the inverse design simulator (Optimization), the base classes and some layers of the internal objects. A solid line indicates class derivation (`is-a` relationship), whereas dashed lines represent a pointer/handle (`has-a` relationship).

tion objects' of the pointers to the subclasses `NlHeat`, `NavierStokes` and `Concentration` need to be bound so that there exist only one 'time integration object'. An example of this process (in the class `Direct`) is shown in the command lines below:

heat→ tip. rebind (tip());
flow→ tip. rebind (tip());

where here `tip` refers to the object that manages the time integration, e.g. the initialization of the time loop, the time stepping process, etc. (see Box 2).

The next important step after the declaration of the subproblem classes is the design of the main simulator. This task involves combining the individual solution routines to the overall optimization scheme. An abstract base class `Optimization` is defined for this purpose that has pointers to objects of the `Direct`, `Adjoint` and `Sensitivity` classes (Figure 1.14). The general structure of this class is shown in Box 8. The class `Optimization` and all its subproblem classes are defined as abstract base classes. This is quite an important and elegant feature of an object-oriented implementation.

```
class DirectAlloy: public Direct, ... {
  protected:
    Handle (NavierStokes1) flow1;   // pointer to the flow problem (double-diffusive)
    Handle (NlHeatAlloy) heat1;     // pointer to the heat transport problem
    Handle (Concentration) conc;    // pointer to the solute transport problem
    Handle(Mat(real)) conc_field;   // concentration field on the interface
                                    // (input to the inverse problem in the solid)
  public:
    DirectAlloy();
    ~DirectAlloy();
    ...
    virtual void define(MenuSystem& menu, int level = MAIN);
    virtual void scan(MenuSystem& menu);
    ...
    virtual void storeResults(BooLean Append, int slice);
    virtual void updateDataStructures();
    virtual void solveAtThisTimeLevel();
  protected:
    virtual void setIC(int restart_level, real t_start);
    virtual void calcDerivedQuantities(int time);
    virtual void loadData4Restart(real tstart);
    virtual void reinitializeGrids();
    virtual void moveGridsToThisTimeLevel();
    virtual void store4restart();
    virtual void storeResidual(int iter);
    ...
};
```

Box 7: The main members of the class `DirectAlloy`.

To allow the developed simulator to accommodate various thermophysical problems, initial conditions, boundary conditions, etc., the actual direct problem details are implemented as virtual functions (`functors` in the terminology of Diffpack) in derived classes from `NlHeat`, `NavierStokes` and `Concentration`. For example, if we declare a pure function `neumann` in the base class `NlHeat` and pass on the specific neumann boundary condition in a problem-dependent class, say `NlHeatDerv`, then the base class pointer could still access the problem-dependent function during runtime. This technique has been used extensively in the present simulator and has been shown to provide clarity and robustness to the developed code. More details on the OOP implementation of the adjoint method for complex transport systems are provided in [49].

```
class Optimization: public ... {
  protected:
    Handle(Direct)              direct;              // Pointer to direct problem
    Handle(Adjoint)             adjoint;             // Pointer to adjoint problem
    Handle(Sensitivity)         sensitivity;         // Pointer to sensitivity problem
    Handle(Mat(real))           heatFlux;            // Primary unknown: design flux
    Mat(real)                   heatFlux_prev;       // Known flux at previous iter
    Handle(Mat(real))           gradient_Qo;         // ∇J(q_o^k)
    Mat(real)                   gradient_Qo_prev;    // ∇J(q_o^{k-1})
    Handle(Mat(real))           directionVec;        // p^k
    Mat(real)                   directionVec_prev;   // p^{k-1}
    Handle(Mat(real))           residual;            // θ − θ_m
    Handle(Mat(real))           sens_T_gamma_I;      // Θ on Γ_I
    ...
    int                         method;              // CGM update formula
  public:
    ...
    virtual void define(MenuSystem& menu, int level = MAIN);
    virtual void scan(MenuSystem& menu);
    virtual void adm(MenuSystem& menu);
    virtual void driver();
    virtual void storeResults(int k);
    virtual void updateDataStructures();
  protected:
    ...
    virtual void conjugateGradient();
    virtual real costFunctional();
    virtual void calculateDirectionVec(int iteration);
    virtual real innerProduct(Mat(real)& f, Mat(real)& g, int n);
    ...
};
```

Box 8: The main members of the class `Optimization`.

1.4.4 An example of the design of directional solidification of pure materials with desired growth and freezing interface heat fluxes

The example to be addressed in this section consists of the inverse design of a directional solidification system through thermal boundary heat flux control in the presence of a vertical externally applied constant magnetic

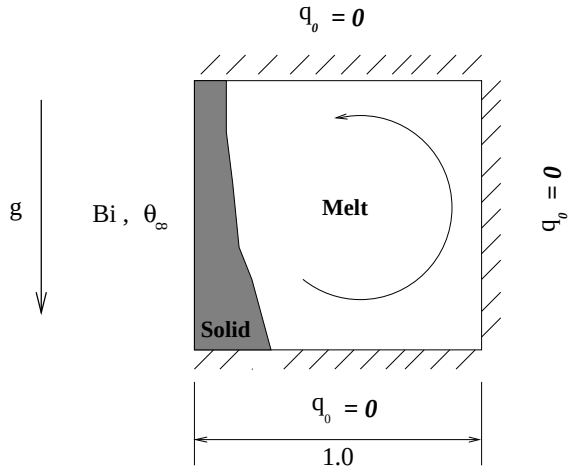

Figure 1.15: Unidirectional solidification of Aluminum in a square enclosure. A constant vertical magnetic field is applied.

	Symbol	Value
Prandtl number	Pr	0.0149
Rayleigh number	Ra	2×10^4
Stefan number	Ste	0.5317
Heat conductivity ratio	R_k	1.0
Specific heat ratio	R_C	1.0
Biot number	Bi	3.3

Table 1.1: Dimensionless variables governing the solidification of pure Aluminum.

field. A square mold with aspect ratio of 1.0 is considered. It is initially occupied by liquid Aluminum whose properties are given in [11]. The ambient temperature was taken to be 25^o C. The melt is initially superheated by an amount of $\Delta T_o = 200^o$ C. A mixed temperature/flux boundary condition was applied to one wall and all other walls were insulated as shown in Figure 1.15. The dimensionless variables governing the direct problem are listed in Table 1.1.

The penalty number to enforce incompressibility was chosen to be 10^8. A finite element mesh of 18×18 4-noded quadrilateral elements is used in the melt domain while a compatible grid of 15×18 is used in the solid domain. Note that as part of the moving FEM formulation, the FEM grids in the solid and liquid domains must share the same nodes at the solid/liquid interface. A small initial solid region (1 percent solid) was

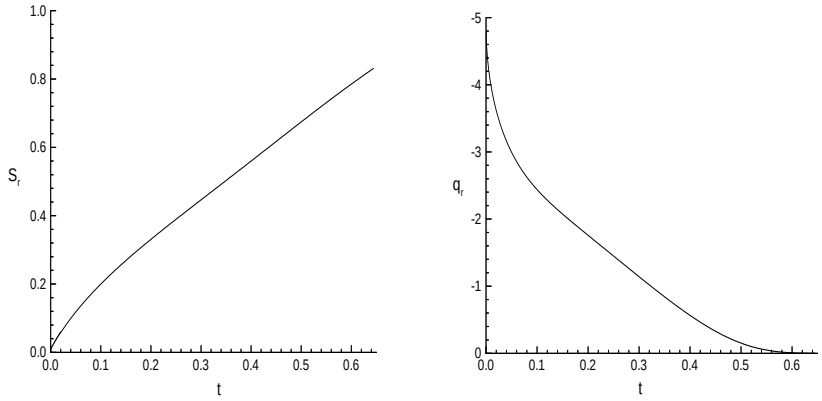

Figure 1.16: The desired interface location and heat flux calculated from the solution of a Stefan problem (Ra = 0).

assumed at the start of the direct simulation. A uniform temperature gradient was assumed in the initial solid region. A complete simulation of the direct problem for various strengths of the magnetic field has been performed. Such simulations are similar to the ones examined in [11] without the presence of magnetic fields. The above direct problem constitutes what is referred to here as the *initial design*. A *reference design problem* or a "target state" is defined next.

In the above described direct convection based solidification problem (see Figure 1.15), the onset of natural convection results in a curved solid-liquid interface ($\frac{\partial s}{\partial y} \neq 0$), as well as in a vertical non-uniformity of the interfacial heat flux ($\frac{\partial q_I}{\partial y} \neq 0$). The interface location is here denoted as $s(y,t)$ and q_I is the heat flux on the liquid side of the freezing front.

If we neglect natural convection ($Ra = 0$), the above problem is reduced to a one-dimensional two-phase Stefan problem in a finite domain [50]. Using the numerical scheme discussed in [11], the interface location $s_r(t)$ and the interface heat flux $q_r(t)$ for this Stefan problem have been calculated and are shown in Figure 1.16 (recall that with the selection of the vector $\boldsymbol{n}$ pointing towards the solid, the interface heat flux $q_r < 0$). This solution is used to define the *desired design objectives* of the present inverse analysis.

The design problem of interest can now be stated as follows:

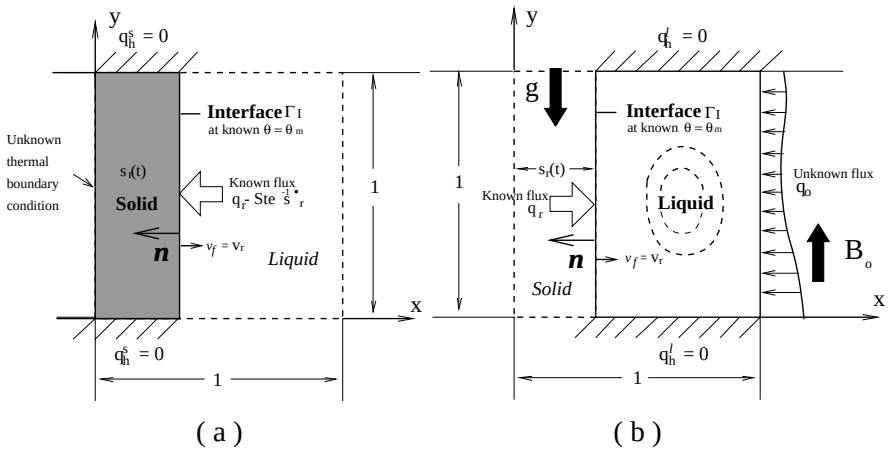

Figure 1.17: Inverse design problems in the solid and melt domains for an Aluminum solidification system. With the particular selection of the vector $\boldsymbol{n}$, the Stefan condition (for $R_k = R_C = 1$) connecting the two inverse problems takes the form $\frac{\partial \theta_s}{\partial n} = \frac{\partial \theta_l}{\partial n} - \mathrm{Ste}^{-1}\dot{s}_r \equiv q_r - \mathrm{Ste}^{-1}\dot{s}_r$.

Find the thermal condition on the left wall $x = 0$ and the right wall $x = 1$ such that with magneto-convection in the melt, a vertical interface $v_f = v_r(t)$ and a vertically uniform interfacial heat flux $q_I = q_r(t)$ are achieved.[a]

[a] This design problem is a particular case of the inverse design problem presented in Section 1.4.1 for stable desired growth of binary alloys. Here, $c = 0$ and ϵ in equation (1.23) is the desired interface heat flux. With $\epsilon \leq 0$, columnar growth is always stable in pure materials.

Zabaras and Yang [17] investigated earlier this problem without the presence of a magnetic field. They obtained an optimal heat flux with quite a complex spatial variation. Based on earlier work in the literature on electro-magnetic damping of melt flow, one using the above defined inverse design problem can investigate the effects that the magnetic field can have in the computed optimal heat flux solution. As such, various strengths of magnetic fields are considered in this analysis. Similar studies can also be conducted by varying the orientation of the magnetic field with respect to the direction of solidification.

The above design solidification problem is converted to two sub-design problems, one in the melt domain and another in the solid domain. Figures 1.17a, b give the complete definition of the two inverse design subproblems. Only details of the inverse design problem in the melt domain are considered here.

The inverse problem in the liquid domain solves for $q_o(y,t)$ at $x = 1$ with

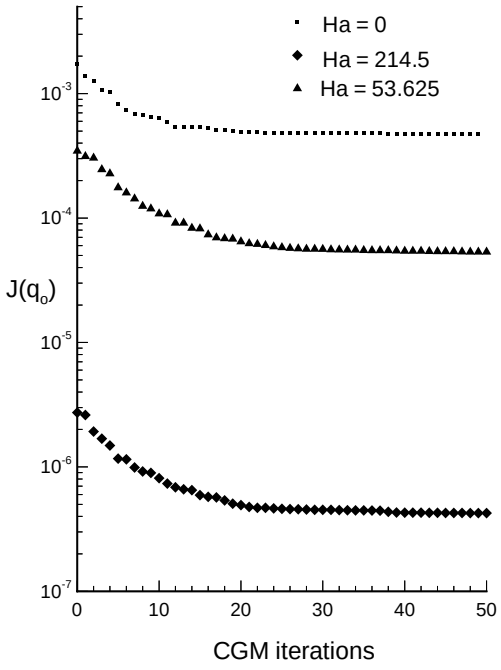

Figure 1.18: Values of the objective/cost function $J(q_{ol})$ versus number of CGM iterations for Hartmann numbers 0, 53.625 and 214.5.

given freezing interface velocity $v_r(t)$ and heat flux $q_r(t)$ and the other mold walls being adiabatic. An initial guess $q_o^0(y,t) = 0$ is taken (this corresponds to the initial design introduced earlier). The time domain $[0, t_{max}]$ is taken with $t_{max} = 0.645$ which corresponds to solidification of about 85% of the initial melt.

Within each CGM iteration, the adjoint and sensitivity problems are solved with the same finite element algorithm as that used in the direct solidification problem. The spatial and temporal discretizations remain the same for all three problems. A deforming/moving front tracking FEM analysis is used for these type of solidification problems with a sharp solid/liquid freezing front [11]. The total number of time steps involved in the solution of each of the direct, adjoint and sensitivity problems is 825. The total computational time for each CGM iteration including the solutions of the three subproblems is about three hours CPU time on an IBM RS-6000 workstation. The convergence of the CGM method for the three cases with different magnetic field strengths is shown in Figure 1.18. The computations are stopped after 50 iterations as the cost functional reaches an asymptotic value.

The spatial and temporal variations of q_o^k at various iterations for the

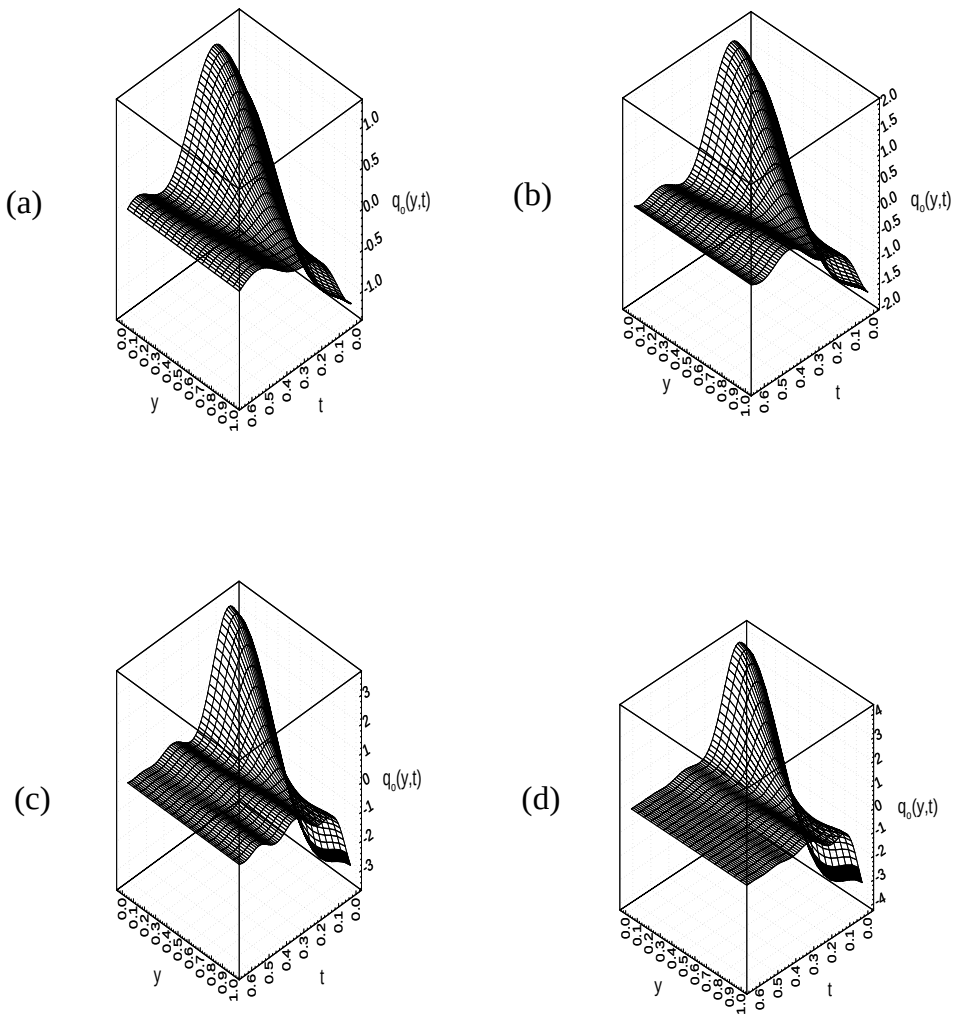

Figure 1.19: Flux solution $q_o^k(y,t)$ at iterations (a) 5, (b) 10, (c) 20 and (d) 50 for Ha = 53.625.

case of Hartmann number 53.625 are shown in Figure 1.19. The optimal solution ($k = 50$) exhibits the largest amount of heating and cooling at the corners of $x = 1$ and at the earlier stages of solidification. This is expected as the boundary heat flux q_{ol} should be adjusted early in time so that at later times it can influence the effects that convection has on the freezing interface.

From additional calculations with other strengths of the magnetic field, it is observed that the magnetic field strength decreases the order of magnitude of the cost function. This effect is due to damping of the flow caused by the applied magnetic field. The effect of magnetic damping reduces the role played by the thermal boundary fluxes in producing the desired interface data. Also, the optimal fluxes obtained with an externally applied magnetic field are shown to be smaller in magnitude and to vary in a more smoother manner compared to the optimal heat flux obtained in the case of no external magnetic field.

To evaluate how accurately the design objectives have been met, the direct magneto-solidification problem is considered (including the simultaneous analysis of the solid and liquid phases) with the optimal heat flux $\bar{q}_o(y,t)$ applied at $x = 1$, mixed boundary condition with $\theta_\infty = -3.175$ and $Bi = 3.3$ at $x = 0$ and the remaining walls insulated. This direct problem solution will be referred to as the *optimal design solution.*

In order to quantify the achievement of vertical uniformities of the interface heat flux and growth velocity, the following two standard deviations are defined for the inverse design objectives s and q_I:

$$\sigma_s(t) = \left\{ \sum_{i=0}^{N} [s(i,t) - s_r(t)]^2 \right\}^{\frac{1}{2}} \tag{1.45}$$

$$\sigma_q(t) = \left\{ \sum_{i=0}^{N} [q_I(i,t) - q_r(t)]^2 \right\}^{\frac{1}{2}} \tag{1.46}$$

where N refers to the number of nodes on the interface in the discretized problem.

The time histories of $\sigma_s(t)$ and $\sigma_q(t)$ are shown in Figure 1.20 for the optimal design solutions for various magnetic fields along with their counterparts for the initial design problem. It can be clearly seen that with both an inverse design solution and an applied magnetic field the design solution gets closer to the reference problem and furthermore the solution improves with increasing magnetic field.

The freezing interface shapes obtained in the validation problems with various magnetic field strengths are plotted at a time which corresponds to mid-solidification, when the interface curvature is maximum for the initial design problem (see Figure 1.21). It can be concluded that application of a moderate magnetic field along with inverse thermal design almost achieves

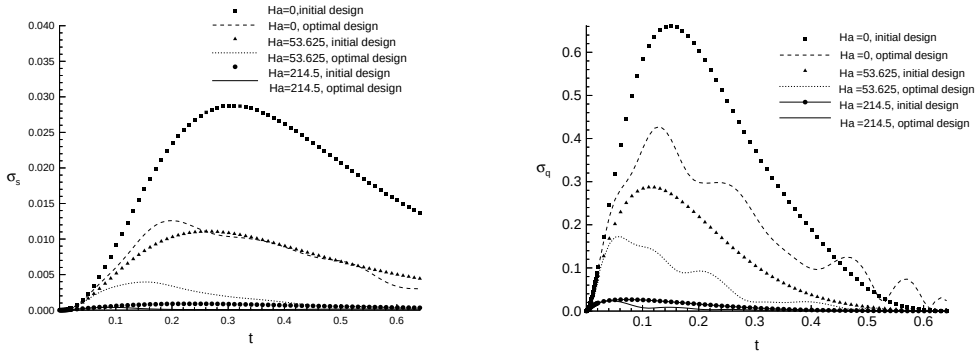

Figure 1.20: Standard deviations of s(y,t) and q_I(y,t) for the initial and optimal designs.

the required objective. The application of a high magnetic field, even though producing excellent agreement with the desired objective, may not be practically feasible.

1.5 Conclusions and future work

From the developments presented here, it is clear that significant advantages result from the object-oriented implementation of FEM algorithms for the analysis, design and control of material processes. The use of abstract classes and class hierarchies has been shown to reduce the required programming effort while it increases the readability, clarity and robustness of the developed codes. From the various preliminary simulations in both forming and solidification processing, it has been shown that the efficiency of the developed OOP simulators is comparable to that of codes developed using traditional programming approaches (including C or FORTRAN).

The presented solidification design simulator was recently used to design alloy solidification processes with desired stable growth and reduced segregation in the presence of magnetic fields [34]. The OOP structure of the present simulator can easily be extended to allow us to address other design solidification processes in which the physics is much more involved than the one considered here. For example, there is an interest in the control of solidification processes with proper *simultaneous* design of magnetic fields, rotation and thermal cooling/heating.

Finally, the deformation simulator presented in the first part of this paper was recently extended to include an implementation of the continuum sensitivity analysis given in [51]. Such sensitivity analysis for large inelastic deformations can be very useful for the design and control of metal forming processes. In particular, our interest is in the development of robust algo-

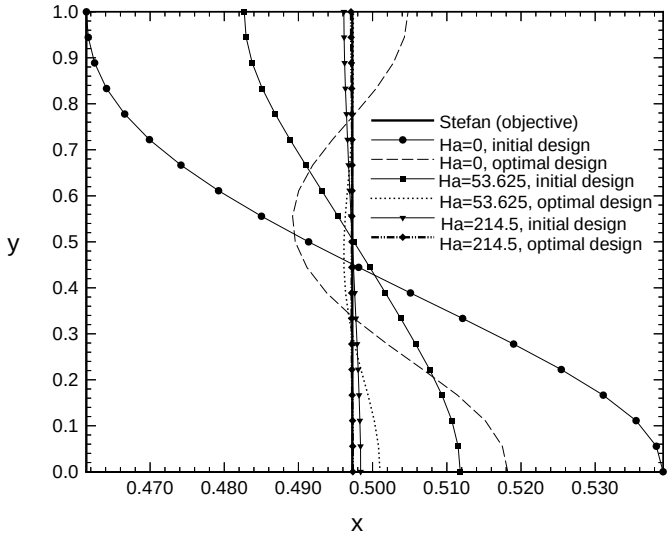

Figure 1.21: Comparison of the interface locations at $\tau = 0.35$ for the various validation problems. These results demonstrate the effectiveness of both inverse thermal design and magnetic field in controlling the solidification system.

rithms for die and preform (shape) design and process parameter design. The objective of these design problems is to obtain a product of a given shape with a desired material state and with a minimum cost (in terms of required forces or plastic work). This work is discussed in detail in two recent publications [52], [53].

Many of the OOP ideas presented here may play an important role in implementation and verification procedures for many other continuous processes governed by coupled systems of PDEs. In particular, it was demonstrated that a whole class of optimum control problems can be formulated with the adjoint method and efficiently implemented using an object-oriented environment.

1.6 Acknowledgments

The work presented here was funded by NSF grant DMII–9522613 to Cornell University. Additional support was provided by the Alcoa Laboratories (Dr. Paul Wang, program manager). The computational facilities provided by the Cornell Theory Center are acknowledged. The academic license for using the various libraries of Diffpack is appreciated. The contributions of R. Sampath in the development of the solidification simulator and of A. Srikanth and Y. Bao in the development of the deformation simulator are appreciated. Finally, the various constructive comments and suggestions of Prof. H. P. Langtangen of the University of Oslo and of three anonymous reviewers have contributed significantly to the improvement of the original manuscript.

Bibliography

[1] ABAQUS. *Reference Manual.* Hibbit, Karlsson and Sorensen Inc., 100 Medway Street, Providence, RI, 02906–4402, 1989.

[2] M. S. Engleman. *FIDAP*, v. 7.52. Fluid Dynamic International, Incorporated, 1996.

[3] J. J. Barton and L. R. Nackman. *Scientific and Engineering C++.* Addison-Wesley, New York, 1994.

[4] R. I. Mackie. Object oriented programming of the finite element method. *Int. j. numer. methods engr.* 35 (1992) 425-436.

[5] T. Zimmermann, Y. Dubois-Pèlerin and P. Bomme. Object-oriented finite element programming: I Governing principles. *Comp. Methods Appl. Mech. Engr.* 98 (1992) 291-303.

[6] R. I. Mackie. Using objects to handle complexity in FE software. *Engineering with Computers* 13 (1997) 99-111.

[7] E. Arge, A. M. Bruaset and H.-P. Langtangen (edts.). *Modern Software Tools for Scientific Computing.* Birkhauser, Boston, 1997.

[8] M. Daehlen and A. Tveito (edts.). *Numerical Methods and Software Tools in Industrial Mathematics.* Birkhauser, Boston, 1997.

[9] P. Bomme. *Intelligent Objects in Object-Oriented Engineering Environments.* Ph.D. Thesis, École Polytechnique Fédérale de Lausanne, Thèse No. 1763, 1998.

[10] H. P. Langtangen. *Computational Partial Differential Equations - Numerical Methods and Diffpack Programming.* Springer-Verlag, New York, 1999.

[11] R. Sampath and N. Zabaras. An object oriented implementation of a front tracking finite element method for directional solidification processes. *Int. j. numer. methods engr.* 44(9) (1999) 1227–1265.

[12] B. Moran, M. Ortiz and C. F. Shih. Formulation of implicit finite element methods for multiplicative finite deformation plasticity. *Int. j. numer. methods engr.* 29 (1990) 483–514.

[13] G. Weber and L. Anand. Finite deformation constitutive equations and a time integration procedure for isotropic, hyperelastic-viscoplastic solids. *Comp. Methods Appl. Mech. Engr.*, 79 (1990) 173–202.

[14] T. A. Laursen and J. C. Simo. On the formulation and numerical treatment of finite deformation frictional contact problems. *Nonlinear Computational Mechanics - State of the Art*, (edts.) P. Wriggers and W. Wager. Springer Verlag, Berlin, (1991) 716–736.

[15] N. Zabaras and T. Hung Nguyen. Control of the freezing interface morphology in solidification processes in the presence of natural convection. *Int. J. Num. Meth. Eng.* 38 (1995) 1555-1578.

[16] N. Zabaras and G. Yang. A functional optimization formulation and FEM implementation of an inverse natural convection problem. *Comp. Meth. Appl. Mech. Eng.* 144(3-4) (1997) 245-274.

[17] G. Yang and N. Zabaras. The adjoint method for an inverse design problem in the directional solidification of binary alloys. *J. Comp. Phys.* 140(2) (1988) 432-452.

[18] G. Yang and N. Zabaras. An adjoint method for the inverse design of solidification processes with natural convection. *Int. J. Num. Meth. Eng.* 42 (1998) 1121–1144.

[19] N. Zabaras and A. Srikanth. An object oriented programming approach to the Lagrangian FEM analysis of large inelastic deformations and metal forming processes. *Int. J. Num. Meth. Eng.*, in press.

[20] A. Srikanth and N. Zabaras. A computational model for the finite element analysis of thermoplasticity coupled with ductile damage at finite strains. *Int. J. Num. Meth. Eng.*, in press.

[21] N. Zabaras and A. Srikanth. Using objects to model finite deformation plasticity. *Engineering with Computers*, in press.

[22] A. L. Gurson. Continuum theory of ductile rupture by void nucleation and growth: Part 1 - Yield criteria and flow rules for a porous ductile media. *J. Engr. Mat. Techn.* 99 (1977) 2–15.

[23] V. Tvergaard and A. Needleman. Elastic-Viscoplastic analysis of ductile fracture. *Finite Inelastic Deformations - Theory and Applications*, (edts. D. Bedso and E. Stein). IUTAM symposium Hannover, Germany (1991) 3–14.

[24] S. B. Brown, K. H. Kim and L. Anand. An internal variable constitutive model for hot working of metals. *Int. J. Plasticity* 5 (1989) 95–130.

[25] J. Budiansky. Thermal and thermoelastic properties of isotropic composites. *J. Composite Materials* 4 (1970) 286–295.

[26] N. Aravas. On the numerical integration of a class of pressure-dependent plasticity models. *Int. j. numer. methods engr.* 24 (1987) 1395–1416.

[27] A. Zavaliangos, L. Anand, B. F. von Turkovich. Deformation processing. *Annals of the CIRP* 40 (1991) 267–271.

[28] A. Zavaliangos and L. Anand. Thermal aspects of shear localization in microporous viscoplastic solids. *Int. j. numer. methods engr.* 33 (1992) 595–634.

[29] Z. L. Zhang. On the accuracy of numerical integration algorithms for Gurson-based pressure dependent elastoplastic constitutive models. *Comp. Methods Appl. Mech. Engr.* 121 (1995) 15-28.

[30] Z. L. Zhang. Explicit consistent tangent moduli with a return mapping algorithm for pressure-dependent elastoplasticity models. *Comp. Methods Appl. Mech. Engr.* 121 (1995) 29-44.

[31] A.M. Lush. *Computational procedures for finite element analysis of hot working.* Ph.D. Thesis, MIT, 1990.

[32] A. M. Lush. Thermo-mechanically-coupled finite element analysis of hot-working using an implicit constitutive time integration scheme. *Numerical Methods in Industrial Forming Processes* (edts. J.-L. Chenot, R.D. Wood and O.C. Zienkiewicz) (1992) 281–286.

[33] J. C. Simo and C. Miehe. Associative coupled thermoplasticity at finite strains: Formulation, numerical analysis and implementation. *Comp. Methods Appl. Mech. Engr.* 98 (1992) 41–104.

[34] R. Sampath and N. Zabaras. Inverse thermal design and control of solidification processes in the presence of a strong magnetic field. *J. Comp. Physics*, submitted for publication.

[35] T.E. Tezduyar. Stabilized finite element formulations for incompressible flow computations. *Advances in Applied Mechanics* 28 (1991) 1-44.

[36] T. E. Tezduyar, M. Behr and J. Liou. A new strategy for finite element computations involving moving boundaries and interfaces

- the deforming-spatial-domain/space-time procedure: I. The concept and the preliminary tests. *Comp. Meth. Appl. Mech. Engr.* 94 (1992) 339-351.

[37] T. E. Tezduyar, M. Behr, S. Mittal and A. A. Johnson. Computation of unsteady incompressible flows with the finite element method - space-time formulations, iterative strategies and massively parallel implementations. *New Methods in Transient Analysis* (edts. P. Smolinski, W. K. Liu, G. Hulbert and K. Tamma, ASME, New York) AMD 143 (1992) 7-24.

[38] A. N. Brooks and T. J. R. Hughes. Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. *Comp. Methods Appl. Mech. Engr.* 32 (1982) 199-259.

[39] R. Sampath and N. Zabaras. A diffpack implementation of the Brooks/Hughes Streamline-upwind/Petrov-Galerkin FEM formulation for the incompressible Navier-Stokes equations. Research Report MM-97-01, Sibley School of Mechanical and Aerospace Engineering, Cornell University, (URL: http://www.mae.cornell.edu/research/zabaras), 1997

[40] R. Sampath and N. Zabaras. Finite element simulation of buoyancy driven flows using an object-oriented approach. Research Report MM-97-02, Sibley School of Mechanical and Aerospace Engineering, Cornell University (URL: http://www.mae.cornell.edu/research/zabaras), 1997.

[41] R. Sampath and N. Zabaras. FEM simulation of double diffusive convection using an object-oriented approach. Research Report MM-97-03, Sibley School of Mechanical and Aerospace Engineering, Cornell University (URL: http://www.mae.cornell.edu/research/zabaras), 1997.

[42] R. Sampath and N. Zabaras. FEM simulation of alloy solidification in the presence of magnetic fields. Research Report MM-97-04, Sibley School of Mechanical and Aerospace Engineering, Cornell University (URL: http://www.mae.cornell.edu/research/zabaras), 1997.

[43] R. Sampath and N. Zabaras. Design and object-oriented implementation of a preconditioned-stabilized incompressible Navier-Stokes solver using equal-order interpolation velocity-pressure elements. Research Report MM-99-01, Sibley School of Mechanical and Aerospace Engineering, Cornell University (URL: http://www.mae.cornell.edu/research/zabaras), 1999.

[44] O. M.Alifanov. *Inverse Heat Transfer Problems.* Springer-Verlag, Berlin, 1994.

[45] G. I. Marchuck. *Adjoint Equations and Analysis of Complex Systems.* Kluwer Academic Publishers, Boston, 1995.

[46] W. Kurz and D. J. Fisher. *Fundamentals of Solidification.* Trans Tech Publications Ltd, Switzerland, 1989.

[47] N. Zabaras. Adjoint methods for inverse free convection problems with application to solidification processes. *Computational Methods for Optimal Design and Control* (edts. J. Borggaard, E. Cliff, S. Schreck and J. Burns). Birkhauser Series in Progress in Systems and Control Theory, Birkhauser (1998) 391–426.

[48] D. G. Luenberger. *Optimization by vector space methods (Series in Decision and Control).* J. Wiley and Sons, New York, 1997.

[49] R. Sampath and N. Zabaras. An object-oriented implementation of adjoint techniques for the design of complex continuum systems. *Int. j. numer. methods engr.*, submitted for publication.

[50] H. S. Carslaw and J. C. Jaeger. *Conduction of Heat in Solids.* Oxford University Press, 2nd Ed., New York, 1959.

[51] S. Badrinarayanan and N. Zabaras. A sensitivity analysis for the optimal design of metal forming processes. *Comp. Methods Appl. Mech. Engr.* 129 (1996) 319–348.

[52] N. Zabaras, Y. Bao, A. Srikanth and W. G. Frazier. A continuum sensitivity analysis for metal forming processes with application to die design problems. *Int. j. numer. methods engr.*, submitted for publication.

[53] A. Srikanth and N. Zabaras. Preform design and shape optimization in metal forming. *Comp. Methods Appl. Mech. Engr.*, submitted for publication.