MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE-8752-TDI/795

SIMULAÇÃO NUMÉRICA BIDIMENSIONAL DE CRESCIMENTO DE LIGAS BINÁRIAS UTILIZANDO PROCESSAMENTO PARALELO

Nanci Naomi Arai

Dissertação de Mestrado em Computação Aplicada, orientada pelos Drs. Maurício Fabbri e Stephan Stephany, aprovada em 28 de fevereiro de 2001.

INPE
São José dos Campos
2002

681.3 : 539.2

ARAI, N. N.
Simulação numérica bidimensional de crescimento de ligas binárias utilizando processamento paralelo / N. N. Arai – São José dos Campos: INPE, 2001.
116p. – (INPE-8752-TDI/795).

1.Solidificação direcional. 2.Método de Bridgman. 3.Ligas binárias. 4.Modelos matemáticos. 5.Teste de desempenho. 6.Processamento paralelo. I.Título.

Aprovado pela Banca Examinadora em cumprimento a requisito exigido para a obtenção do Título de **Mestre** em **Computação Aplicada.**

Dr. Airam Jônatas Preto

Presidente/INPE/SJCampos-SP

Dr. Stephan Stephany

Orientador/INPE/SJCampos-SP

Dr. Maurício Fabbri

Orientador/INPE/SJCampos-SP

Dr. Antonio Carlos Hernandes

Membro da Banca
Convidado/USP/São Carlos-SP

Candidata: Nanci Naomi Arai

São José dos Campos, 28 de fevereiro de 2001.

A meus pais Tomico Arai e Seiichi Arai.

AGRADECIMENTOS

Aos Doutores Maurício Fabbri e Stephan Stephany pela orientação recebida ao longo desses anos, desde a Iniciação Científica; aos Doutores Airam Jonatas Preto e Antônio Carlos Hernandes, membros da banca, pela participação; ao Instituto Nacional de Pesquisas Espaciais (INPE) pela disponibilização de materiais e recursos; à Fundação Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo apoio financeiro; aos docentes do curso de Computação Aplicada que auxiliaram na formação de fundamentos para o desenvolvimento desta dissertação; aos colegas de curso pelo auxílio e amizade; a Marcelo Banik de Padua que elaborou o estilo de tese e dissertação, em Latex, segundo às normas de publicação do INPE; a Ricardo Varela que cedeu máquinas e auxiliou na realização de testes computacionais do algoritmo implementado, a Daniel Massaru pelo apoio, presença e auxílio em todos os momentos; e a todos que contribuíram, direta ou indiretamente, para a elaboração deste trabalho.

RESUMO

A solidificação direcionada de ligas semicondutoras é uma das técnicas mais utilizadas para a obtenção de substratos cristalinos de alta qualidade, constituindo um protótipo de estudo de fenômenos de transporte macroscópicos envolvendo o acoplamento das equações de conservação de massa, momento e energia. As propriedades cristalinas do material obtido dependem fundamentalmente da morfologia e estabilidade da interface sólido-líquido durante o crescimento. O método numérico proposto resolve essas equações de transporte através da discretização por volumes de controle com acompanhamento da interface, e permite operar com propriedades do material e condições de contorno variáveis, incorporando detalhes do diagrama de fases da liga. Foram realizadas simulações bidimensionais de crescimento de ligas binárias utilizando malha fixa, as quais demandam processamento de alto desempenho, devido às altas resoluções temporal e espacial envolvidas. Isso levou a utilizar uma máquina multiprocessada e um multicomputador composto por 2 microcomputadores ligados em rede executando programas compilados em *High Performance Fortran* (HPF).

BIDIMENSIONAL NUMERICAL SIMULATION OF GROWTH OF BINARY ALLOYS USING PARALLEL PROCESSING

ABSTRACT

The directional solidification of semiconductor alloys is an usual technique for the attainment of a high quality crystalline substratum. This technique is a prototype for the study of macrocospic transport phenomena and involves the coupling of the conservation equations for mass, moment and energy. The crystalline properties of the alloy depend basically on the morphology and stability of the solid-liquid interface during growth. The proposed numerical method solves these transport equations by discretization in control volumes with tracking of the interface. It allows for materials with variable properties and boundary conditions, and to include details of the alloy phase diagram. Two-dimensional, fixed-mesh simulations of binary alloy growth have been carried out. Due to the high spatial and temporal resolutions, these simulations were run on a multiprocessed machine and on a multicomputer composed by a cluster of two microcomputers. The programs were compiled in High Performance Fortran (HPF).

SUMÁRIO

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE SÍMBOLOS

Latinos

c	-	calor específico
C	-	concentração
C_0	-	concentração inicial do líquido homogêneo
D	-	coeficiente de difusão
e	-	energia interna
E	-	eficiência
g	-	concentração na região *mushy*
G	-	propriedade ou variável genérica
h	-	entalpia
$\vec{J}$	-	fluxo de massa
k	-	condutividade térmica
$\tilde{K}$	-	tensor de condutividade térmica
L	-	calor latente
p	-	número de processadores
$\vec{Q}$	-	fluxo de energia
r	-	fração paralelizável do código
S	-	*speedup*
t	-	tempo
t_{par}	-	tempo paralelo
t_{seq}	-	tempo sequencial
T	-	temperatura
T_0	-	temperatura inicial
T_m	-	temperatura de fusão
x	-	comprimento na direção axial
X	-	posição da interface

Gregos

Γ	-	coeficiente de segregação interfacial
Γ_e	-	coeficiente de distribuição efetivo
ϵ	-	precisão
λ	-	fração líquida
ρ	-	densidade
ω	-	fator de relaxação

Índices Superiores

^{A}	-	indica substância A
^{B}	-	indica substância B
$^{\text{approx}}$	-	indica valor aproximado
$^{\text{exact}}$	-	indica valor exato
$^{\text{new}}$	-	indica iteração atual
$^{\text{old}}$	-	indica iteração anterior

Índices Inferiores

$_{\text{interf}}$	-	indica interface
$_{l}$	-	indica estado líquido
$_{s}$	-	indica estado sólido
$_{P}$	-	indica ponto P da malha de discretização
$_{\text{NG}}$	-	indica pontos na vizinhança do ponto P

CAPÍTULO 1

Introdução

Cristais apresentam características especiais, as quais lhe conferem uma série de propriedades, tornando-os empregáveis, por exemplo no caso de semicondutores, na fabricação de diversos dispositivos eletrônicos. Estas propriedades, que decorrem da estrutura e da composição dos cristais, podem ser alteradas com a presença de defeitos. A formação de defeitos pode ser controlada pelas condições nas quais a cristalização ocorreu [8].

Cristais podem ser obtidos através de vários processos de solidificação como, por exemplo, a técnica de Bridgman-Stockbarger [25], que consiste em uma solidificação direcional e é comumente utilizada para obtenção de ligas binárias semicondutoras em cilindros com cerca de dois centímetros de diâmetro.

A simplicidade dessa técnica de crescimento torna-a atrativa para o estudo da complexa inter-relação entre convecção, difusão e transferência de calor que ocorre durante a solidificação. Os cristais, obtidos em laboratórios, são analisados geralmente em relação à distribuição de concentração e aos defeitos estruturais apresentados.

Inhomogeneidades na distribuição da concentração, resultantes da macrosegregação axial e radial estão relacionadas à morfologia e estabilidade da interface sólido-líquido durante o crescimento, que são dependentes dos gradientes térmicos perto da interface de crescimento e das taxas de transferência de massa por difusão e convecção no líquido.

A perfeição química e cristalina de cristais, produzidos pela técnica de Bridgman, poderia ser manipulada através do controle da morfologia da interface [25], no entanto a capacidade desse controle pode ser reduzida devido às condições térmicas, inerentes a essa técnica.

Medidas dos perfis de concentração de ligas binárias (Hg,Cd)Te [11] solidificadas direcionalmente, indicam que a interface sólido-líquido não é caracterizada, tipicamente, nem por uma isoterma nem por uma isoconcentração, o que é confirmado também nos experimentos realizados com ligas (Pb,Sn)Te [24]. Esses experimentos sugerem que a relação entre o perfil de composição radial e o formato

da interface de crescimento pode ser sensível aos detalhes do diagrama de fases da liga. O formato da interface é fortemente dependente das diferenças entre as condutividades do sólido e do líquido, bem como do material da ampola ou cadinho de crescimento [26]. A segregação radial e a forma da interface estão ligadas através do acoplamento entre a transferência de calor, que ocorre entre as regiões quente e fria do forno, o calor carregado pelo ampola [32] e a convecção do líquido [31].

Apesar de todos os detalhes complexos que envolvem o crescimento Bridgman, essa técnica é, essencialmente, um processo de quasi-equilíbrio, envolvendo ligas simples com diagramas de fases bem conhecidos. Uma vez que o super-resfriamento constitucional é evitado tanto quanto possível, a interface sólido-líquido apresenta-se suave e bem definida e é determinada, unicamente, pelos valores de concentração e temperaturas na frente da fase.

Durante o processo de mudança de fase de uma liga binária ocorrem diversos fenômenos como a transferência de calor e de massa que em conjunto com a convecção ocasionam o problema da segregação ou redistribuição de soluto.

A segregação deve-se ao fato de que a composição química do cristal difere, em geral mesmo no equilíbrio, da composição química do nutriente, que é a fase da qual o cristal se forma [12]. As composições do sólido e líquido, de uma liga binária, em função da temperatura são dadas pelo diagrama de fase do material. As propriedades da liga, tais como difusidade, condutividade térmica, calor específico, devem ser consideradas em função de sua composição e estado.

O problema de crescimento de cristais é bastante complexo e envolve diversas variáveis. Para a compreensão desses processos inicia-se com modelos mais simples, desprezando algumas variáveis, no caso de certos tipos especiais de crescimento e com justificativas físicas.

Na literatura sobre transferência de massa e de calor, há muitos artigos que propõem simplificações dos modelos de solidificações de ligas [2]. As duas aproximações mais comuns são ignorar ou a difusão ou a condução. Neste contexto, a solidificação de uma liga é vista ou como uma condução limitada ou como uma difusão limitada.

As soluções analíticas só existem para certos casos específicos, de forma que os problemas, de modo geral, são resolvidos numericamente [5].

Houve, recentemente, um grande desenvolvimento na área de crescimento de cristais: na tecnologia desenvolvida para projeto e controle de fornos, no uso de campos eletromagnéticos externos e na possibilidade de realizar crescimento em um ambiente de microgravidade. Estes avanços possibilitaram um alto grau de controle sobre os experimentos realizados, de forma que houve uma maior demanda por simulações numéricas mais precisas e confiáveis [17].

Há diversos métodos numéricos disponíveis para a simulação de crescimento e solidificação de cristais, tais como o Método de Front-Tracking [13], o Método da Entalpia [2] e o Método de Campo de Fase [18].

Uma das dificuldades encontradas nos problemas de solidificação é que a posição da interface não é conhecida a priori, enquadrando-se estes na classe de problemas de condição de contorno móvel (*moving-boundary problems*), de modo que a localização da interface, no decorrer do tempo, deve ser determinada como uma parte da solução do problema.

A formulação aqui apresentada do processo de solidificação, é baseada diretamente nas leis básicas de conservação de massa e energia interna, e na adoção de equilíbrio termodinâmico local, de forma que o diagrama de fases determina a fase, localmente, a cada instante de tempo [17]. A modelagem ocorre a nível macroscópico, não abordando o estudo da microestrutura da interface, nem tampouco os efeitos convectivos e as reações químicas [2]. O foco de estudo concentra-se no acoplamento entre a condução de calor e a difusão de massa no material.

O método numérico proposto implementa uma simulação bidimensional de crescimento de ligas binárias com malha fixa, baseado numa aproximação em volumes de controle, que é, em princípio, capaz de manipular propriedades do material e condições de contorno complicadas, incorporando detalhes do diagrama de fases da liga.

Os volumes de controle do método de malha fixa não precisam se adaptar à posição da interface, portanto esse método não apresenta as dificuldades de se lidar com malhas irregulares e com complicações na região da interface, como ocorre, geralmente, no método de Front Tracking de malha adaptativa.

A integração no tempo é feita de modo implícito. Os detalhes do diagrama de fases da liga são incluídos através de uma equação de estado apropriada, considerando-se

o método de Solomon, Alexiades and Wilson [39]. No método original a integração no tempo era feita explicitamente. Foi adotado um método implícito para que os campos de temperatura e concentração fossem obtidos de maneira autoconsistente e iterativa para cada passo no tempo, e também para aumentar a região de estabilidade do problema.

A integração no tempo é realizada de forma que qualquer célula retenha seu estado de fase durante o passo de tempo, e uma mudança somente ocorra no final do intervalo de tempo adotado. Esta metodologia é necessária por causa de grandes descontinuidades nas entalpias específicas e no calor específico de um volume de controle quando ele entra ou deixa uma região de mudança de fase.

Esse método numérico, para a resolução do processo de solidificação de ligas binárias, foi desenvolvido e implementado anteriormente, em linguagem C, por Fabbri [17].

Esta dissertação teve como objetivo avaliar a implementação do método numérico supracitado, em um ambiente de processamento paralelo, utilizando a linguagem High Performance Fortran (HPF).

Para realizar esse trabalho foi estudada a física do problema de forma a obter um modelo matemático adequado e, para esse modelo, foram escolhidos os métodos numéricos que levaram ao desenvolvimento de um algoritmo e sua decorrente implementação em Fortran 90. Foram também estudadas técnicas de processamento paralelo e de alto desempenho de forma a paralelizar esse código utilizando HPF.

As técnicas de paralelização de código consistem, basicamente, em dividir o domínio do problema ou o conjunto de tarefas entre os processadores, num multicomputador ou numa máquina multiprocessada, de modo a obter resultados corretos, mas com uma redução de tempo expressiva em comparação com uma máquina monoprocessada [16].

Para a obtenção de uma paralelização de código eficiente deve-se buscar atingir duas metas: balanceamento de carga entre processadores e minimização da troca de dados entre processadores, no caso de uma máquina multiprocessada, de memória compartilhada ou minimização da troca de mensagens entre processadores, no caso de um multicomputador, de memória distribuída.

Essas metas podem ser comprometidas pela dependências de dados, pelo fluxo de

controle do problema, pela impossibilidade de dividir o domínio ou as tarefas de maneira eficiente, etc.

Para a resolução do problema de solidificação proposto foi desenvolvido um novo algoritmo para a implementação do modelo utilizado. Inicialmente, esse algoritmo foi implementado em linguagem Fortran 90 [14] e executado em uma máquina monoprocessada. Posteriormente, este algoritmo foi otimizado de modo a obter uma redução no tempo de processamento, ainda em um máquina monoprocessada.

No tocante a paralelização, foram analisados os laços que poderiam ser paralelizados por não apresentarem dependências de dados entre iterações, bem como formas de distribuição dos dados entre os processadores disponíveis, referentes a essas iterações. A partir dessa análise foram inseridas diretivas de *High Performance Fortran* (HPF) [29], no código, para a realização de testes. Progressivamente, tentou-se paralelizar mais laços, tendo-se monitorado o tempo computacional gasto.

O Capítulo 2 inicia-se com uma breve descrição de cristais e suas características, propriedades, aplicações e defeitos. No Capítulo 3, apresentam-se algumas das técnicas utilizadas para a obtenção de cristais. No Capítulo 4, abordam-se fenômenos, como a difusão de massa e de calor, que ocorrem durante o processo de crescimento de cristais. No Capítulo 5, desenvolve-se a formulação do problema e apresentam-se os métodos numéricos utilizados para a sua resolução. No Capítulo 6, descrevem-se técnicas de processamento paralelo e de alto desempenho, as arquiteturas paralelas, os paradigmas de programação e as linguagens Fortran 90 e HPF. No Capítulo 7, apresenta-se o ambiente utilizado e a abordagem adotada para a realização das simulações, e os resultados obtidos. No Capítulo 8, são feitas as principais considerações do problema resolvido, apresentam-se conclusões bem como sugestões de trabalhos futuros.

CAPÍTULO 2

Cristais

O cristal é um arranjo tridimensional periódico de átomos. Um cristal ideal é constituído pela repetição infinita, no espaço, de unidades estruturais idênticas. Em cristais mais simples como cobre, prata, ferro e em metais álcalis, a unidade estrutural é um átomo simples. Geralmente a unidade estrutural são vários átomos ou moléculas, acima de cem em cristais inorgânicos e dez mil em cristais de proteínas [23].

A estrutura de todos os cristais é descrita em termos de uma rede com um grupo de átomos unidos a cada ponto da rede. O grupo é chamado de base, que é repetido no espaço para formar a estrutura do cristal. A rede é um arranjo periódico regular de pontos no espaço com uma simetria bem determinada, é uma abstração matemática, conhecida como rede de Bravais [41] [15].

A estrutura do cristal é formado somente quando a base de átomos é unida identicamente a cada ponto da rede. Cada base é idêntica em composição, arranjo e orientação. O monocristal é representado por uma única rede de Bravais juntamente com uma mesma base atômica que se repete nas posições da rede.

Alguns cristais são admirados pela sua beleza, que se deve à simetria, à estrutura e à pureza. Essas características conferem uma série de propriedades físicas e químicas ao cristal que tem sido utilizadas em várias aplicações.

Por um longo tempo a única fonte de cristais estava na natureza. A regularidade da forma externa exibida por cristais naturais reflete um ordenamento interno, ainda que, em geral, cristais naturais se apresentem na forma policristalina, ou seja,

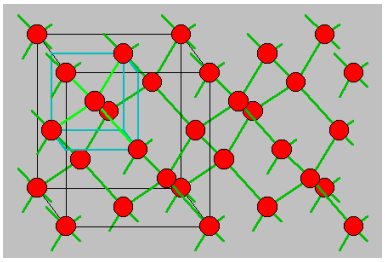

Fig. 2.1 – Rede de Diamante.
FONTE: adaptada de [41].

composta por grãos monocristalinos.

Para que o cristal pudesse ser utilizado, o primeiro passo foi produzir cristais, artificialmente, em larga escala, fornecendo amostras para um intensivo estudo científico que causou um desenvolvimento na química da matéria inorgânica (metalurgia, cerâmica), bem como uma rápida expansão na física do estado sólido.

Finalmente a fabricação de cristais tornou-se uma atividade comercial quando o transistor e outros dispositivos eletrônicos semelhantes foram inventados. Um breve histórico sobre a evolução na área de cristais pode ser encontrada em [8].

2.1 Aplicações

Cristais podem ser empregados na fabricação de diversos dispositivos, devido às propriedades que apresentam. Estas propriedades decorrem das características do cristal, que podem ser alteradas com a presença de defeitos. A formação de defeitos pode ser controlada, e utilizada de maneira vantajosa, conforme as condições nas quais a cristalização ocorreu.

Os cristais semicondutores constituem a maior classe com aplicações e o principal representante dessa classe é o silício [8].

Cristais semicondutores puros e perfeitos, à temperatura de zero absoluto seriam isolantes. As propriedades semicondutoras são causadas, usualmente, por agitação térmica, defeitos na rede cristalina e impurezas, conhecidas como dopantes, distribuídos de maneiras particulares. São conhecidos, na prática, por serem condutores eletrônicos cujas resistividades elétricas, a temperatura ambiente, variam em torno de 10^{-2} a 10^{9} ohm-cm, encontrando-se entre bons condutores (10^{-6}ohm-cm) e isolantes (10^{14} a 10^{22} ohm-cm) e são, usualmente, dependentes da temperatura.

Dentre várias utilizações citaremos:

- Dispositivos semicondutores

  - diodos elétricos: Si, Ge;

  - fotodiodos: Si, GaAs, $Cd_xHg_{1-x}Te$;

  - transistores: Si, GaAs;

- dispositivos fotocondutores: Si, $Cd_xHg_{1-x}Te$;

- circuitos integrados: Si, GaAs;

- dispositivos de emissão de luz: GaAs, $Sn_xPb_{1-x}Te$;

- detectores de radiação: Si, Ge, CdTe.

- Componentes mecânicos

  - ferramentas de corte e abrasivas: C (diamante), Al_2O_3.

- Dispositivos magnéticos

  - circuladores de microondas: silicatos;

  - cabeças de fita: ferrita.

- Dispositivos piezelétricos

  - transdutores (convertem vibrações para sinais elétricos e vice e versa).

- Dispositivos ópticos

  - prismas, lentes;

  - polarizadores: $CaCO_3$;

  - dispositivos magneto-ópticos: $Y_3Fe_5O_{12}$;

  - dispositivos eletro-ópticos: $LiNbO_3$, $NH_4H_2PO_4$, KH_2PO_4.

2.2 Presença de Defeitos

A perfeição desejada no cristal depende de uma série de fatores interdependentes, entre eles a dinâmica de transporte na fase de nutriente, o stress térmico, a cinética de agregação e a morfologia e estabilidade da interface sólido-líquido.

Quando os primeiros cristais artificiais foram produzidos, a qualidade do produto era medida comparando sua simetria com a do cristal natural. Atualmente preocupa-se também com a pureza e a ausência de defeitos sutis como linhas de deslocação [20].

Devemos ter um cristal sem falhas para que não haja descontinuidade das propriedades, e comprometimento no funcionamento do dispositivo final. Porém os defeitos também podem ser favoráveis quando produzem características desejáveis nos dispositivos.

Muitas propriedades importantes são controladas tanto pelas imperfeições quanto pela natureza do cristal, que pode atuar somente como um veículo, solvente ou matriz para as imperfeições. A condutividade de alguns semicondutores deve-se inteiramente à quantidade de vestígios de impurezas químicas. A luminescência dos cristais, bem como suas propriedades mecânicas e plásticas são quase sempre conectadas com a presença de impurezas ou imperfeições[28].

Desse modo, os defeitos podem ser estudados a fim de deduzir como as técnicas de crescimento poderiam ser modificadas para aumentar o grau de eficiência do material, quer seja pela ausência de defeitos ou pela distribuição adequada destes.

Qualquer desvio da estrutura ou rede periódica no cristal é considerado como uma imperfeição. Os defeitos podem ocorrer, portanto, na estrutura cristalina, na orientação ou composição. Impurezas químicas e vacâncias caracterizam imperfeições pontuais, as quais ocorrem em pontos ou átomos na estrutura, diferentemente do que ocorre quando se tem linhas ou planos de imperfeições, referentes a deslocações e contornos de grão.

O tipo de imperfeição mais simples é uma vacância, que é a ausência de um átomo ou íon na rede cristalina. É conhecido como defeito de Schottky, quando o átomo que falta no interior da rede cristalina encontra-se em um ponto da rede na superfície do cristal. Outro tipo de vacância é o defeito de Frenkel, no qual o átomo é tansferido de um ponto da rede para uma posição entre esses pontos, ou seja, uma posição que normalmente não seria ocupada por um átomo. São ilustrados na Figura 2.2.

A inhomogeneidade, que é a diferença de concentração das substâncias, que compõem uma liga, em cada ponto do cristal, não é considerada um defeito, embora altere o

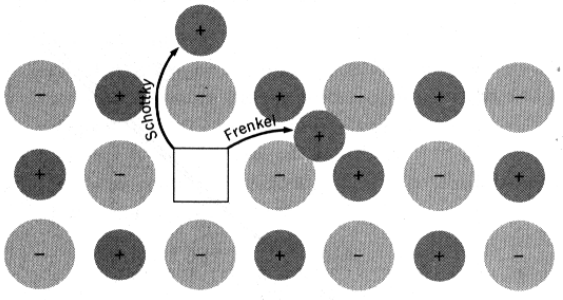

Fig. 2.2 – Defeitos de Schottky e Frenkel num cristal iônico.
FONTE: adaptada de [28].

desempenho do dispositivo.

As deslocações são um tipo de defeito causado por condições de não equilíbrio durante o crescimento do cristal. É conhecida como deslocação o limite entre duas regiões, o qual pode ser exemplificado através da Figura 2.3, onde o limite se encontra ao término da fileira vertical extra de átomos na parte superior. Podem ser encontrados vários tipos de deslocações. Experimentalmente comprova-se que cristais de metais e de semicondutores crescidos do líquido possuem, em geral, deslocações. O conteúdo das deslocações depende das condições sob as quais a solidificação ocorreu. É possível crescer cristais sem deslocações perceptíveis pelos métodos disponíveis correntemente, sob condições especiais.

Os contornos de grãos são defeitos tridimensionais e consistem de inclusões de blocos de fase sólida, líquida ou gasosa diferente do cristal. Podem ser causados, durante o crescimento, se for energeticamente favorável aos átomos adotar uma posição alternativa, mudando a ordem da rede cristalina, nesse caso pode até mesmo alterar o formato externo do cristal.

Um cristal apresenta, geralmente, imperfeições de vários tipos. Estudar qual a origem desses defeitos torna possível a pesquisa por novas técnicas que permitam o crescimento de cristais com alta perfeição na estrutura e composição[12].

Apresenta-se, no próximo capítulo, algumas das técnicas utilizadas para o crescimento de cristais.

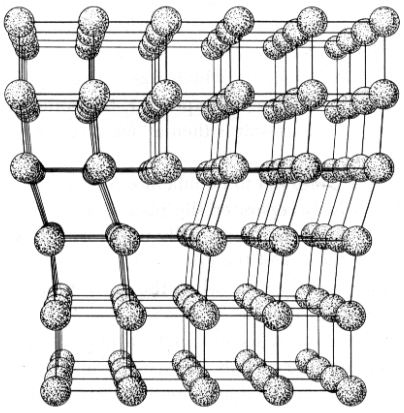

Fig. 2.3 – Deslocação.
FONTE: adaptada de [28].

CAPÍTULO 3

Crescimento de Cristais

A otimização das técnicas de crescimento mostra-se interessante na pesquisa básica dos processos clássicos de transporte, bem como de modelos de agregação/nucleação. Um grande número de fatores influenciam decisivamente a qualidade do cristal obtido em laboratório: os perfis de temperatura e velocidade do forno, a geometria e as propriedades físicas do material das ampolas, as características próprias do material crescido (diagrama de fase, constantes de difusão, condutividade térmica, etc.).

3.1 Processos de Crescimento

A produção sistemática de cristais artificiais consiste em crescer cristais sólidos a partir de uma fase nutriente (gás, líquido ou sólido). Para iniciar o processo de crescimento utiliza-se frequentemente uma semente. Diversos métodos podem ser encontrados em [8].

3.1.1 Crescimento a Partir da Fase Líquida

É conhecido como solidificação, onde a transformação de fase é causada pela extração de calor do líquido, e o progresso dessa transformação é separado em duas partes: a nucleação inicial do cristal e o crescimento desse núcleo inicial pela adição de átomos do líquido [10].

3.1.1.1 Método de Bridgman-Stockbarger

Uma técnica bastante utilizada para a obtenção de macrocristais semicondutores binários é o método de Bridgman-Stockbarger. É uma solidificação direcional sob um gradiente térmico móvel e consiste em, dentro de um forno, fundir o material contido numa ampola e provocar sua solidificação por meio da retirada lenta da zona quente para a zona fria. Pode ser realizada com deslocamento da ampola ou do forno.

3.1.1.2 Método de Czochralsky

O líquido é mantido em um cadinho de material refratário adequado (grafite, quartzo, alumina) ou de metal nobre (platina, ouro), que não contamine o material cristalino. Dentro do líquido é mergulhado uma semente que é retirada a uma velocidade

baixa e constante para formar o lingote. A semente é rotacionada lentamente para manter o cristal uniforme e com formato cilíndrico. O diâmetro do cristal depende da temperatura do líquido e da taxa de puxamento.

Os principais requisitos para se obter cristais de boa qualidade, por este método, são que as velocidades de rotação e de puxamento devem ser uniformes e a temperatura no líquido deve ser precisamente controlada.

Pode produzir cristais grandes e livres de deslocações em um espaço de tempo relativamente curto. A principal vantagem desse método é ser uma técnica em tempo real onde o tamanho e o diâmetro podem ser controlados enquanto o crescimento está se realizando.

3.1.1.3 Técnica de Refino por Zona

O princípio do refino por zona é a passagem de uma zona líquida através de um lingote. Muitas impurezas (aquelas com coeficientes de segregação < 1) tendem a ficar no líquido e, dessa forma, o sólido é purificado por passagens sucessivas da zona líquida. A descrição detalhada do processo de refino de zona e suas extensões e modificações pode ser encontrada em [35].

A função principal das técnicas de refino por zona é manipular ou redistribuir impurezas solúveis presentes. As impurezas possuem influência nas propriedades físicas, químicas e mecânicas das substâncias.

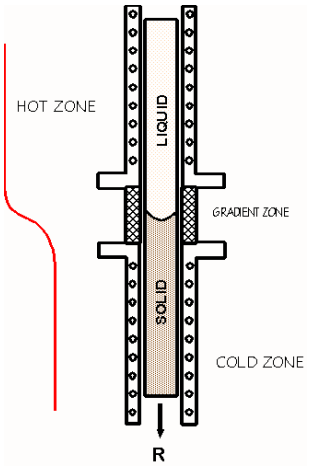

Fig. 3.1 – Crescimento Bridgman.

Portanto é de importância vital obter cristais com alto grau de pureza para estudar suas propriedades em geral ou para utilizar propriedades específicas de materiais puros na fabricação de dispositivos. As impurezas indesejadas são deslocadas para o final do lingote por passagens repetidas de uma zona derretida ou de várias zonas em uma direção.

Esse método também é utilizado para a realização de dopagem, que significa acrescentar uma impureza desejada em um material, com distribuição uniforme. Nesse processo, uma zona é passada repetidamente em ambas as direções eliminando, virtualmente, os efeitos de segregação [33].

3.1.1.4 Outros Métodos

A maior parte dos cristais, obtidos por solidificação, é produzido pelos métodos de Bridgman e de Czochralsky, porém há outros métodos também utilizados [8], tais como processo de Verneuil, processo de Zona Flutuante e crescimento de Fusão em arco.

3.1.2 Crescimento a Partir da Fase de Vapor

Nas técnicas de crescimento a partir da fase de vapor, o material a ser crescido é transportado para a zona de crescimento na forma de componentes voláteis.

O crescimento da fase de vapor tem aplicação comercial na produção de cristais

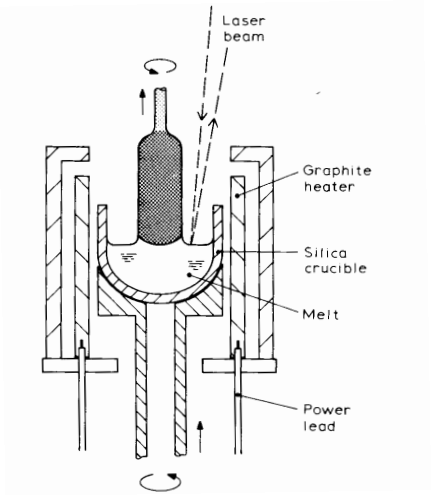

Fig. 3.2 – Crescimento Czochralsky.
FONTE: adaptada de [8].

luminescentes (sulfeto de zinco, sulfeto de cádmio) e cristais semicondutores (silício, germânio, compostos III - VI).

No caso de cristais de metais, o crescimento a partir da fase de vapor, é tecnologicamente de pouca importância, pois podem ser obtidos de maneira mais fácil através da solidificação.

Entretanto, há um aumento na demanda de cristais obtidos do vapor, devido ao fato desses cristais possuírem propriedades incomuns. Geralmente apresentam superfícies externas perfeitamente lisas que não podem ser reproduzidas por outras técnicas e possuem menos subestruturas e imperfeições que os obtidos por solidificação.

3.1.2.1 Processo de Deposição Química de Vapor (Chemical Vapour Deposition - CVD)

Nos processos químicos, os átomos ou moléculas de vapor são quimicamente diferentes daqueles no cristal crescido. As moléculas de vapor são absorvidos na superfície do substratos onde são decompostos termicamente, ou são reduzidos, de forma química, a liberar os átomos de metais.

São usados amplamente para depositar materiais policristalinos. Nesse processo, a fonte do material a ser crescido é um sólido que sublima ou um líquido que evapora.

Os átomos ou moléculas de vapor chocam-se com a superfície do substrato e são

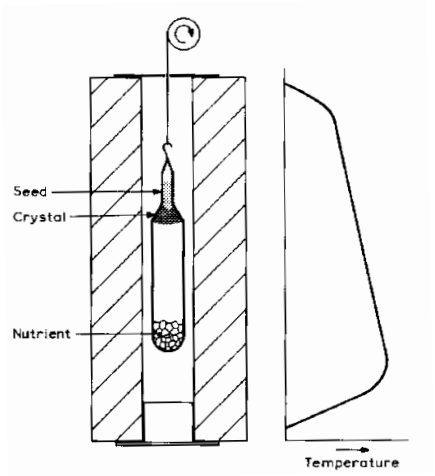

Fig. 3.3 – Processo de Deposição Química de Vapor (CVD).
FONTE: adaptada de [8].

absorvidos, liberando parte do calor latente de condensação e migram através da superfície do cristal. Se eles encontram os pontos da rede cristalina eles se incorporam, liberando o restante do calor latente, caso contrário eles reevaporam.

O procedimento geral é selar o material dentro de um tubo que é evacuado ou preenchido com um gás inerte. O tubo é aquecido num gradiente de temperatura e os vapores de metais efundem através do tubo e condensam nas paredes frias.

No Laboratório Associado de Sensores e Materiais (LAS/INPE) são realizadas pesquisas com crescimento de diamantes, utilizando o processo descrito acima [40], dentre outros processos de crescimento de cristais.

Processo de Epitaxia por Feixe Molecular (Molecular Beam Epitaxy - MBE)

É um método sofisticado de evaporação que utiliza técnicas de vácuo com pressões da ordem de $10^{-11} torr$ e fontes de evaporação especiais. Foi usado inicialmente para o estudo de processos de crescimento.

É possível crescer camadas de GaAs com menos de 0.03 ppm de impurezas. As velocidades de crescimento variam, geralmente, de 0.1 a 5 μ m/h.

O MBE oferece duas possibilidades difícieis de se obter em outros métodos, citados por [8]. Dohler & Ploog exemplificam a exatidão de controle sobre o processo, pelo crescimento de 100 Å de GaAs seguido por 100 Å de AlAs. A segunda possibilidade

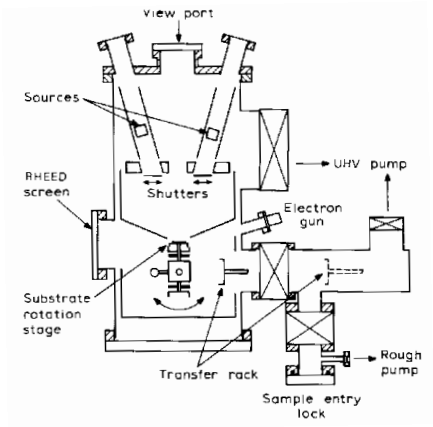

Fig. 3.4 – Processo de Epitaxia por Feixe Molecular (MBE).
FONTE: adaptada de [8].

é uma extensão desse processo na escala atômica. Por exemplo, Chang e Esakio trabalham com o crescimento de camadas alternadas de GaAs e AlAs, formando superestruturas.

3.1.3 Crescimento a Partir da Fase Sólida

Os crescimentos a partir da fase sólida são utilizados com menos frequência do que os métodos citados anteriormente. Foram amplamente realizados antes do desenvolvimento das técnicas de refino por zona.

O crescimento de cristais simples a partir de sólidos policristalinos ou amorfos apresenta as vantagens de ser simples (em princípio seria necessário apenas recozer a amostra), e de evitar contaminação (não são usados solventes e a amostra pode ter áreas limitadas de contatos com materiais estranhos). As taxas de crescimento são da ordem de 5 a 50 mm/h.

As desvantagens são a ausência de reprodutibilidade e a dificuldade de atingir alto grau de perfeição, as quais são fortemente dependentes da amostra utilizada. Mesmo com processamentos ideais, alguns defeitos cristalográficos são inevitáveis.

A recristalização engloba todos os fenômenos os quais provocam algum tipo de alteração no número, tamanho, forma ou orientação dos constituintes da amostra. Consiste, na prática, na transformação de uma amostra policristalina com grãos em um grande cristal, através de tratamento térmico ou de deformação plástica seguida de tratamento térmico. Do ponto de vista energético, esse processo deve ser favorável, uma vez que a estrutura policristalina não representa o estado termodinâmico mais estável do metal.

Alguns processos de crescimento a partir da fase sólida estão ilustrados em [8].

Durante a realização desses processos de crescimento de cristais ocorrem diversos fenômenos, os quais podem ser avaliados a nível macroscópico e microscópico e são descritos no próximo capítulo.

CAPÍTULO 4

Mudança de Fase

Segundo Alexiades e Solomon, no livro *Mathematical Modeling of Melting and Freezing Processes* ([2]), as fases sólida e líquida são caracterizadas pela presença de forças coesivas que mantém a proximidade dos átomos. No sólido, os átomos vibram em torno de posições de equilíbrio fixas, enquanto no líquido eles podem saltar entre essas posições. A manifestação macroscópica dessa energia de vibração é chamada de calor ou energia térmica, que é medida pela temperatura.

Se a solidificação for definida como o processo pelo qual um sólido cresce a partir do líquido no qual ele está em contato, tem-se como ponto de partida a análise das condições nas quais o sólido e o líquido podem coexistir. O estado de equilíbrio é definido pela coexistência do sólido e do líquido sem mudança em suas quantidades relativas. Essa é a condição na qual a solidificação não ocorre, porém tanto experimental quanto teoricamente verifica-se que a solidificação pode ocorrer quando as condições diferem levemente das de equilíbrio.

Os átomos na fase líquida são mais energéticos que os na fase sólida. Desse modo para um sólido passar para líquido, ele precisa adquirir uma certa quantidade de energia para superar as forças de ligação que mantém a estrutura sólida. Essa energia é conhecida como calor latente L do material e representa a diferença nos níveis de energia térmica (entalpia) entre os estados líquido e sólido.

A solidificação do líquido requer a remoção desse calor latente e a estruturação dos átomos em posições dos pontos da rede mais estáveis. Há três tipos de transferência de calor em um material: condução, convecção e radiação. Condução é a transferência de energia cinética entre átomos por diversas maneiras, por exemplo, a colisão de átomos vizinhos e o movimento de elétrons; nao há fluxo ou transferência de massa do material. Isto acontece em sólidos opacos. Por sua vez em um líquido aquecido pode ocorrer transferência através do fluxo de partículas, ou seja, por convecção. E a radiação é a único tipo de transferência de energia que pode ocorrer no vácuo (não requer meio participante).

A transição de uma fase para outra, ou seja, a absorção ou liberação de calor latente, ocorre em uma temperatura na qual a estabilidade de uma fase sucumbe em favor da outra de acordo com a energia disponível. Essa temperatura de mudança de fase

depende da pressão.

A temperatura na qual um elemento puro ou componente pode coexistir em estado sólido e líquido, em equilíbrio, é conhecida como Ponto de Fusão, acima do qual a substância permanece na forma estável de líquido, e abaixo da qual o sólido é estável. O ponto de fusão de elemento puro ou componente é geralmente constante, porém pode apresentar variações de acordo com a pressão. A aplicação de pressão tende a privilegiar a formação da fase que tem o menor volume específico. Na maioria dos casos as substâncias se expandem quando se liquefazem de forma que, nesses casos, a temperatura do ponto de solidificação é diretamente proporcional à pressão. Essas considerações implicam que a temperatura do líquido não pode estar abaixo do ponto de fusão e, inversamente, que a temperatura do sólido não pode excedê-lo, exceto pela quantidade requerida para dirigir o processo de solidificação ou de fusão. No caso de metais e de muitos outros materiais, esta diferença é bastante pequena.

A maioria dos sólidos são cristalinos, significando que suas partículas (átomos, moléculas, ou íons) estão arranjados em uma estrutura repetitiva de pontos de rede, que se estende por distâncias significativas em termos atômicos. Assim, uma vez que a formação de um cristal pode requerer o movimento de átomos na estrutura da rede sólida, é possível que a temperatura do material seja reduzida abaixo da temperatura de fusão, sem a formação de sólido. Portanto, líquido super-resfriado, ou seja, líquido na temperatura abaixo da temperatura de fusão pode aparecer, como um estado metaestável.

Durante a solidificação de ligas, em circunstâncias normais e com um gradiente de temperatura positivo no líquido, este estará a uma temperatura acima da linha *liquidus* e não ocorrerá o super-resfriamento. Variações locais na composição, associadas com a redistribuição de soluto podem ocasionar um variação na temperatura *liquidus* efetiva. Portanto, nessas condições, parte do líquido pode ser super-resfriada constitucionalmente, implicando que o super-resfriamento é resultante de variações constitucionais, ou seja, composicionais no líquido. Algumas condições podem favorecer o super-resfriamento, tais como:

- baixo gradiente de temperatura no líquido;
- altas velocidades de crescimento;
- linhas *liquidus* abruptas;

- altas porcentagens de liga;
- baixa difusividade no líquido.

A região da liga onde ocorre a transição de fase, ou seja, sólido e líquido coexistem, é chamada de interface. Sua espessura pode variar de alguns angstroms a alguns centímetros, e a microestrutura pode ser bastante complexa, dependendo de vários fatores (do próprio material, da taxa de resfriamento, do gradiente de temperatura no líquido, da tensão de superfície, etc).

Segundo [12], evidências teóricas e experimentais sugerem que a interface sólido-líquido pode ser suave ou rugosa de acordo com as características do material e a natureza cristalográfica da superfície. Pode ser controlada, em parte, por condições térmicas locais.

O efeito de interface se deve à diferença entre as condutividades térmicas do líquido e do sólido e à liberação do calor latente na interface de crescimento, na presença da ampola. A diferença de temperatura entre o material e a ampola faz com que a interface seja não planar. Para semicondutores, cujas condutividades térmicas do líquido são maiores do que no sólido, os materiais são resfriados pelas ampolas e portanto os cristais obtidos apresentam interfaces côncavas, em relação ao sólido.

Além do efeito de interface, a localização da ampola dentro do forno também infuencia na morfologia da interface que pode variar de côncava à convexa. Segundo [11] na maioria dos casos deseja-se uma interface convexa, porém em certos crescimentos uma interface côncava se faz necessária, por exemplo, no crescimento da liga (Cd Hg)Te uma interface côncava limita variações na composição radial.

A morfologia da interface pode ser controlada, de forma a obter interfaces planas ou reverter a sua direção de curvatura [25].

Segundo [11], experimentos realizados comprovam que baixas taxas de crescimento conduzem a uniformizar o material na direção radial, porém a interface não é uma isoterma nem uma isoconcentração, de forma que os perfis de composição radiais não podem ser utilizados para determinar a morfologia da interface.

A morfologia da interface é um fator importante na obtenção de perfeição estrutural e uniformidade de composição. Geralmente, deseja-se uma interface planar ou

ligeiramente convexa para limitar nucleações espúrias nas paredes da ampola de crescimento [24].

Se a interface for suave e houver alguma dificuldade em iniciar novas camadas, o cristal poderá ter faces planas e extensas. Porém se a interface for rugosa, o crescimento será muito mais sensível a pequenas diferenças de temperatura.

A existência de uma zona de super-resfriamento fará com que a interface fique morfologicamente instável, com perturbações na sua forma. Na ausência de super-resfriamento tem-se interface localmente planar, conhecida como interface abrupta *sharp front*). Para baixos graus de super-resfriamento desenvolve-se uma interface celular, a partir de uma interface planar, inicialmente caracterizada por pequenas cavidades as quais passam para células alongadas. Como o aumento do super-resfriamento, as superfícies das células estendem-se e, eventualmente, ramificam-se para formar dendritas celulares. Não há um critério bem definido para a transição de células a dentritas. Nesse caso a interface apresenta uma largura aparente, denominada de região *mushy* que, em uma solidificação dendrítica, pode apresentar uma morfologia bastante intricada.

O processo de crescimento ocorre preferencialmente nas áreas das ramificações da estrutura dendrítica, devido a maior área de contato com o meio apresentada. Porém os efeitos de tensão superficial tendem a limitar esse crescimento, pois a temperatura da interface depende da curvatura local, caracterizando o fenômeno conhecido como Efeito de Gibbs-Thompson.

A variação de densidade com a temperatura induz ao fluxo por convecção natural na presença de gravidade, rapidamente igualando a temperatura no líquido e aumentando a transferência de calor. Em microgravidade não há convecção natural (por densidade), mas ocorre a convecção de Marangoni, devido aos efeitos da tensão superficial, que pode dominar a transferência de calor. Todos esses efeitos podem complicar um processo de mudança de fase. Durante o crescimento do cristal, a convecção ajuda a homogeneizar o líquido, na medida em que espalha o soluto pelo líquido, porém trata-se de um fenômeno sobre o qual não se tem controle e que gera áreas de estagnação, onde ocorre acúmulo do soluto e diferenças de concentração.

4.1 Condução de Calor

De acordo com Alexiades e Solomon, no livro *Mathematical Modeling of Melting and Freezing Processes* ([2]), as quantidades fundamentais envolvidas na condução de calor através do material são: temperatura, calor (energia térmica) e fluxo de calor.

A temperatura é a medida macroscópica do movimento de moléculas. É dada em Kelvin (K), graus Celsius (°C) ou graus Fahrenheit (°F).

O calor absorvido por um material sob pressão constante é uma quantidade termodinâmica chamada de entalpia. O calor é expresso em joules (J). Denota-se a entalpia por unidade de massa e por unidade de volume, dados repectivamente por KJ/Kg e KJ/m^3.

O calor específico dado por KJ/Kg°C, sob pressão constante, representa o calor necessário para aumentar a temperatura de 1 Kg de material de 1 °C. É uma propriedade do material, sempre positiva, que varia levemente com a temperatura e é usualmente mais alta para a fase líquida que para a fase sólida.

O calor latente de mudança de fase é absorvido ou liberado à temperatura constante (temperatura de fusão).

A quantidade de energia atravessando a unidade de área por unidade de tempo é chamado de fluxo de energia, denotado por $\vec{Q}$, e medido em KJ/s m^2. O fluxo de calor é dado pela Lei de Fourier:

$$\vec{Q} \;=\; -\tilde{K}\nabla T\,, \tag{4.1}$$

na qual $\tilde{K}$ é o tensor de condutividade térmica do material, medido em KJ/m s°C. Em geral é um tensor com compontentes positivos variando com a temperatura. Na maioria dos casos práticos, assume-se uma condução isotrópica, isto é, k é um escalar. Tipicamente esse valor é mais alto para o sólido que para o líquido, exceto no caso de semicondutores.

4.2 Mudança de Fase em Ligas Binárias

Uma liga binária é um sistema de dois componentes, A-B no qual as moléculas de A e B formam limites físicos criando um novo componente, mas não uma nova

substância química, isto é, eles não reagem quimicamente. Exemplos: Al-Si, Cu-Ni, Ge-Si, (HgTe)-(CdTe). O componente A é conhecido como solvente e B como soluto.

O fenômeno envolvido na solidificação de ligas ocorre em duas escalas distintas, macroscópica ou microscópica. Na escala macroscópica os efeitos da transferência de massa e de calor são globais, e a evolução das fases e dos campos de concentração e de temperatura, durante o curso da solidificação, são dirigidos pelas condições iniciais e de contorno. Na escala microscópica a estrutura cristalina e a morfologia da interface estão em foco. Para o controle efetivo da microestrutura há necessidade de analisar como as estruturas macroscópica e microscópica interagem em um modelo unificado.

Muitas aplicações do processo de solidificação ocorrem com ligas, portanto é necessário considerar o ponto de fusão, ou o seu equivalente, de uma liga. Nas ligas de fase simples, o líquido e o sólido contendo os componentes da liga podem existir em equilíbrio. O sólido e o líquido apresentam, quase sempre, composições diferentes e a temperatura de equilíbrio depende da composição. Para cada composição do líquido há uma temperatura, temperatura *liquidus*, na qual ela está em equilíbrio com o sólido apropriado e, inversamente, cada composição do sólido tem uma temperatura *solidus* na qual ele está em equilíbrio com o líquido apropriado. O sólido e o líquido podem estar em equilíbrio somente a uma mesma temperatura, que é a temperatura *liquidus* do líquido e a temperatura *solidus* do sólido. Essas duas curvas definem o diagrama de fase da liga.

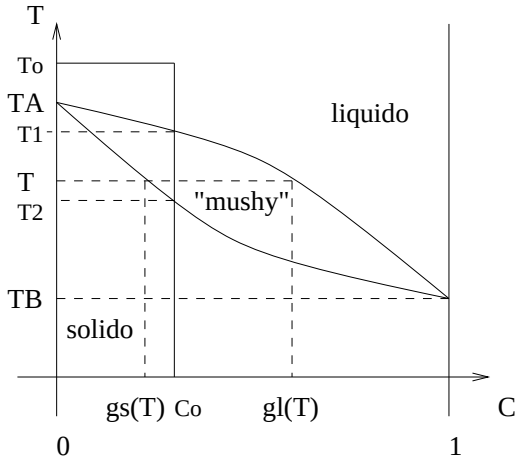

Fig. 4.1 – Diagrama de fase de uma liga binária A-B. Tem-se temperatura em função da concentração, no qual T^A e T^B referem-se, respectivamente, à temperatura de fusão da substância pura A e B. $g_l(T)$ e $g_s(T)$ correspondem à concentração do líquido e do sólido, na região *mushy*.

Há diversos tipos de diagrama de fase, dependendo dos materiais da liga. Vários exemplos podem ser encontrados em [37] e [4].

As mudanças de fase são governadas pelo diagrama de fase. O diagrama descreve os estados termodinâmicos nos quais várias fases podem coexistir em equilíbrio termodinâmico. O estado termodinâmico da liga é determinado por duas variáveis, temperatura e concentração. As curvas *solidus* e *liquidus* demarcam as fases possíveis: a liga está líquida acima da curva *liquidus* e está sólida abaixo da curva *solidus*. Há combinações de composição e temperatura que se encontram entre as curvas *liquidus* e *solidus*, refletindo o fato de que o material não está em equilíbrio estável ou ele consiste de partes sólidas e líquidas, cada qual com a composição apropriada para a temperatura do ponto.

Segundo [12], muitos diagramas de fases binários de metais foram determinados e estão corretos, entretanto alguns apresentam erros, devido ao uso de materiais de baixa pureza e também devido à dificuldade inerente de determinar a curva *solidus*.

Durante o processo de solidificação de uma liga, temperatura e concentração estão relacionadas pelo diagrama de fases mostrado na Figura 4.1. Inicialmente, todo o líquido está a uma temperatura T_0 e concentração inicial C_0, $T_0 > T_l(C_0)$. Diminuindo um pouco a temperatura tem-se $T1 = T_l(C_0)$, que é a temperatura a qual o líquido atinge mantendo a concentração de C_0 devido à ausência de gradientes de concentração. Na temperatura $T_l(C_0)$ a estabilidade do líquido, sob condições de equilíbrio, é destruída e uma fina camada de sólido começa a se formar, na ausência de dificuldades de nucleação. Entretanto, nesta temperatura, sólido e líquido podem coexistir somente nas concentrações indicadas no diagrama de fase. Portanto, o primeiro sólido precisa ter concentração $g_s(T1)$, que é mais baixa que $g_l(T1) \equiv C_0$, dessa maneira o sólido conterá menos soluto que o líquido com o qual ele pode coexistir. Isto requer redistribuição (segregação) do componente soluto, para empobrecer o sólido. Com a queda de temperatura, o líquido remanescente vai enriquecendo e o sólido pregressivamente contém mais soluto. Quando a temperatura chega a $T2 = T_s(C_0)$, todo o líquido se solidificou e o processo foi completado. Observe que enquanto $T_s(C_0) < T < T_l(C_0)$, o líquido está abaixo da temperatura $T_l(C_0)$ e aparece super-resfriado e o sólido aparece superaquecido. Segundo [12], pode-se dizer que a liga está super-resfriada constitucionalmente.

O coeficiente de segregação Γ é dado por C_s/C_l, em que C_s é a concentração do sólido

que é formado em algum instante pela solidificação do líquido de concentração C_l. O coeficiente de distribuição efetivo Γ_e é dado por C_s/C_0, em que C_s é a concentração do sólido formado a partir do líquido de concentração média C_0. O valor de Γ_e depende das condições sob as quais a solidificação ocorre, enquanto que Γ é uma característica do sistema. Γ não é necessariamente uma constante para um dado sistema, porque as linhas *solidus* e *liquidus* podem não manter uma relação constante.

O soluto pode ser distribuído por difusão e por convecção. A difusão está sempre presente, e os efeitos da convecção dependem das condições de crescimento.

4.3 Difusão

Alexiades e Solomon, no livro *Mathematical Modeling of Melting and Freezing Processes* ([2]), definem a difusão como o movimento relativo de moléculas de uma substância em outra.

Supondo uma liga binária A-B, seja ρ a densidade local (massa por unidade de volume) de uma liga e $\rho^A(x,t)$ e $\rho^B(x,t)$ as densidades de A e B respectivamente, de modo que

$$\rho \;=\; \rho^A + \rho^B. \tag{4.2}$$

Sejam J^A e J^B os fluxos de massa de A e B (massa atravessando uma unidade de área por unidade de tempo). O fluxo de massa total é nulo, porque a liga não se move, somente há troca de lugares entre moléculas de A e B:

$$J^A + J^B \;=\; 0. \tag{4.3}$$

Considere um volume elementar com área de secção unitária e comprimento Δx centrada em x. A massa do componente A neste volume (por unidade de área) no tempo t é $\rho^A(x,t)\Delta x$. Durante o intervalo Δt, o ganho de massa

$$[\rho^A(x,t+\Delta t) - \rho^A(x,t)]\Delta x\,, \tag{4.4}$$

é igual a quantidade de massa que atravessa o volume,

$$\Delta t \left[J^A \left(x - \frac{\Delta x}{2}, t \right) - J^A \left(x + \frac{\Delta x}{2}, t \right) \right]. \tag{4.5}$$

Dividindo esse balanço de massa por Δx e Δt e no limite, quando esses valores tendem a zero, obtém-se a lei de conservação de massa:

$$\rho_t^A + J_x^A \;=\; 0\,, \tag{4.6}$$

para a substância A.

Similarmente para B temos

$$\rho_t^B + J_x^B \;=\; 0. \tag{4.7}$$

A lei mais simples relacionando fluxo e densidade é a Lei de Fick:

$$J^A \;=\; -D^{AB}\rho_x^A\,, \qquad J^B \;=\; -D^{BA}\rho_x^B\,, \tag{4.8}$$

onde D^{AB}, D^{BA} (m^2/s) são as difusividades de A em B e de B em A, respectivamente. A densidade da liga permanece constante:

$$\rho_x^A + \rho_x^B \;=\; \rho_x \equiv 0.$$

Das equações 4.3 e 4.8:

$$0 \;=\; J^A + J^B \;=\; -(D^{AB}\rho_x^A + D^{BA}\rho_x^B) \;=\; -(D^{AB} - D^{BA})\rho_x^A. \tag{4.9}$$

Portanto $D^{AB} = D^{BA}$, que é chamado de coeficiente de difusão.

As leis de conservação de massa podem ser expressas pelas equações de difusão:

$$\rho_t^A \;=\; (D\rho_x^A)_x\,, \qquad \rho_t^B \;=\; (D\rho_x^B)_x\,, \tag{4.10}$$

genericamente como

$$\rho_t^i \;=\; div(D\nabla\rho^i)\,, \qquad \text{i = A, B.} \tag{4.11}$$

A equação acima pode ser escrita em termos de concentração:

$$C_i \;=\; \frac{\rho^i}{\rho} \;=\; \frac{\rho^i}{\rho^A + \rho^B}\,, \qquad \text{i = A, B,}$$

$$\frac{\partial C_i}{\partial t} \;=\; div(D\nabla C_i)\,, \qquad \text{i = A, B.} \tag{4.12}$$

Em geral, o coeficiente de difusão é uma função do estado termodinâmico, $D \;=\; D(C,T)$, e de difícil medida. Considera-se como sendo da ordem de 10^{-5} cm^2/s para líquidos e 10^{-10} cm^2/s para sólidos, sendo que uma aproximação usual, para estes, é zero.

Em [6] podem ser encontrados tabelas com valores de difusividades experimentais.

Há uma similaridade entre a Lei de Difusão de Fick e a Lei de Condução de Fourier; ambas as equações são do mesmo tipo.

4.4 Redistribuição de Soluto

As variações das condições de crescimento acarretam variações na forma pela qual o soluto é distribuído entre as fases líquida e sólida. Essa redistribuição de soluto, combinada com as correntes de convecção do líquido durante a solidificação, geram o problema da segregação.

Por todos esses fatores, a modelagem e simulação de processos de crescimento consiste num problema que tem atraído muita atenção durante as últimas décadas.

Ao se solidificar, o soluto rejeitado se acumula na fase líquida, para casos nos quais $\Gamma < 1$. Este aumento da concentração no líquido produzirá um aumento da concentração no sólido, levando a segregação ao longo do cilindro sólido (segregação normal).

São possíveis os seguintes casos de segregação:

- a solidificação é lenta o suficiente, para que a difusão no líquido e no sólido eliminem todos os gradientes de concentração. Este caso, conhecido como solidificação em equilíbrio, conduz à não segregação.

  A hipótese de que o sólido todo está a todo tempo, em equilíbrio com o líquido implica que não há gradiente de concentração no sólido, embora o processo de solidificação tenha a tendência de gerá-lo pela deposição de sólido formado do líquido com uma contínua mudança de composição. Essa hipótese demanda, portanto, que o processo de difusão no sólido é rápido o suficiente para reduzir o gradiente de concentração para um valor pequeno negligenciável. Isto somente ocorre se a taxa de avanço da interface é pequena comparada com a taxa de difusão do soluto no sólido, e se o comprimento de difusão (distância total através do qual a interface se move) requerido é pequeno.

  Estas condições não ocorrem na prática porque as taxas de difusão no sólido são bastante baixas. De acordo com [12] a melhor aproximação ocorreria provavelmente em processos geológicos, nos quais a escala de tempo é extremamente longa para os humanos ou para os padrões industriais. Entre os processos metalúrgicos, o maior efeito da difusão durante solidificação é provavelmente encontrado no caso do carbono e nitrogênio no aço, desde que esses solutos intersticiais tenham coeficientes de difusão muito mais altos que o soluto substitucional. Uma quantidade significante de difusão pode ocorrer após a solidificação enquanto o material está esfriando.

- a solidificação é lenta o suficiente para que a mistura no líquido não permita gradientes de concentração no líquido, porém é rápida o suficiente para desprezar a difusão no sólido. Este caso de mistura completa, leva a segregação máxima possível e pode ser aproximada na prática.

- o caso oposto ao primeiro apresentado seria aquele no qual o soluto não se move e, consequentemente, não há mistura no sólido ou no líquido, implicando que o líquido se solidifica sem mudança de composição. Entretanto a melhor aproximação possível para este caso, conhecido como agitação nula é o caso no qual o transporte ocorre apenas por difusão. Como a difusão no sólido é geralmente muito menor que a do líquido,

ela pode ser desprezada. A solidificação é rápida o suficiente para que somente a difusão no líquido afete a distribuição de soluto na interface. Este caso leva a uma leve segregação e é frequentemente realizado na prática.

- a distribuição de soluto é afetada por difusão e convecção. Este caso conhecido como agitação parcial ([21]) leva a uma segregação intermediária e é também comum na prática.

O processo de mudança de fase de uma liga binária, o qual envolve diversos fenômenos como o transporte de calor e de massa e a convecção, descritos anteriormente, pode ser modelado e resolvido numericamente. A formulação do problema e os métodos numéricos utilizados são apresentados no próximo capítulo.

CAPÍTULO 5

Modelagem Numérica

Um dado problema físico pode ser modelado matematicamente, permitindo a realização de simulações em computador, as quais resultam em um conjunto de equações a serem resolvidas. Soluções analíticas podem ser obtidas apenas para problemas mais simples. Em casos mais complicados, como na maioria dos processos de mudança de fase, fazem-se necessárias aproximações convenientes.

Em qualquer caso, simulações em computador demandam o uso de técnicas numéricas. A modelagem matemática em si geralmente leva a uma discretização do tempo e do espaço, sendo por exemplo a região física representada por um número de volumes de controle, o tempo variando em passos discretos, e as derivadas, integrais e limites representados por diferenças finitas.

Obtém-se simulações mais precisas à medida que se utiliza malhas de discretização mais finas, o que demanda mais tempo de processamento.

Normalmente uma simulação numérica envolve os seguintes passos:

- análise física do problema: determinar os fenômenos físicos que serão considerados e quais poderão ser desprezados, definindo as variáveis do sistema;
- modelagem do problema: encontrar equações que descrevam (matematicamente) os fenômenos físicos abordados;
- discretização: aproximar os domínios contínuos do tempo e do espaço por malhas discretas ou, equivalentemente, reduzir a dimensão do espaço onde residem as possíveis soluções;
- desenvolvimento e implementação dos algoritmos em linguagem de alto nível.

As seguintes características devem ser atendidas para que a representação aproximada do sistema possa ser aceitável:

- consistência: significa que as equações discretas utilizadas tendem às leis

físicas quando as malhas de discretização são refinadas, isto é, tendem a domínios contínuos;

- convergência: significa que a solução calculada numericamente aproxima-se da solução exata da equação diferencial à medida que as malhas de discretização são refinadas;

- estabilidade: significa que erros, tais como erros de arredondamento e truncamento do computador, e aproximações do algoritmo, não irão se propagar de forma exponencial.

A precisão de um método é medido em termos do erro nos resultados obtidos, que pode ser expresso de forma absoluta ou relativa:

$$\text{erro absoluto} = \text{valor exato} - \text{valor aproximado}, \tag{5.1}$$

e

$$\text{erro relativo} = \frac{\text{erro absoluto}}{\text{valor exato}}. \tag{5.2}$$

A convergência de um processo iterativo é alcançado quando o critério de precisão adequado é satisfeito. Uma vez que a solução exata é desconhecida, o erro em qualquer iteração pode ser baseado na mudança dos valores calculados de uma iteração para outra. Portanto, para uma dada variável G, em um ponto P qualquer, o erro $\Delta G_P = G_P^{\text{exact}} - G_P^{\text{approx}}$ pode ser aproximado por $G_P^{\text{new}} - G_P^{\text{old}}$, a cada iteração, sendo que G_P^{exact} refere-se ao valor exato, G_P^{approx} ao valor aproximado, G_P^{new} ao valor da iteração mais recente e G_P^{old} ao valor da iteração anterior, da variável G no ponto P.

Há vários critérios de erro que podem ser adotados, em função da precisão ϵ requerida.

No caso de erros absolutos,

$$|(\Delta G_P)_{\max}| \leq \epsilon, \quad \sum_{P=1}^{n} |\Delta G_P| \leq \epsilon, \quad ou \quad \left(\sum_{P=1}^{n} (\Delta G_P)^2 \right)^{1/2} \leq \epsilon, \tag{5.3}$$

e no caso de erros relativos,

$$\left|\frac{(\Delta G_P)_{\max}}{G_P}\right| \leq \epsilon, \quad \sum_{P=1}^{n}\left|\frac{\Delta G_P}{G_P}\right| \leq \epsilon, \quad ou \quad \left(\sum_{P=1}^{n}\left(\frac{(\Delta G_P)}{G_P}\right)^2\right)^{1/2} \leq \epsilon. \tag{5.4}$$

5.1 Métodos Numéricos em Problemas de Contorno Móvel

Problemas de condições de contorno são aqueles que apresentam equações diferenciais as quais devem satisfazer condições impostas nos contornos do domínio. Porém há casos em que a localização exata do contorno não é conhecida e deve ser determinada como uma parte da solução do problema. Segundo [13], esses problemas podem ser divididos em dois tipos: problemas de contorno-livre (*free boundary problems*), em que o contorno é estacionário, e problemas de contorno móvel (*moving-boundary problems*), cuja localização do contorno é dependente do tempo. Essa classificação varia de autor para autor, alguns incluem os dois tipos de problema na classe dos problemas de contorno-livre.

Frequentemente os problemas de contorno móvel estão associados aos problemas de Stefan, definidos a seguir.

5.2 Formulação do Problema de Stefan

O problema de Stefan originalmente foi utilizado para designar o problema de calcular a distribuição de temperatura e a localização da interface de separação entre sólido e líquido de um volume de água. Porém com o passar do tempo esse nome foi utilizado para representar uma classe de problemas que inclui solidificações de ligas.

Por exemplo, uma barra de comprimento lt, $0 \leq x \leq$ lt, inicialmente sólida à temperatura de solidificação, é aquecida a uma temperatura $T > T_m$, na face $x = 0$ e mantida isolada na outra face $x =$ lt.

A cada instante de tempo t, o líquido ocupa $[0, X(t)]$ e o sólido $[X(t), \text{lt}]$, sendo que $X(t)$ representa a localização da interface, e demarca as regiões sólida e líquida. A equação de transporte de calor deve satisfazer cada região de fase e as condições de contorno estabelecidas.

O problema consiste em determinar a localização da interface $X(t)$ e as temperaturas

da fase líquida e sólida, de forma que a equação

$$\rho_f c_f \frac{\partial T_f}{\partial t} = k_f \frac{\partial^2 T_f}{\partial x^2}, \tag{5.5}$$

deve ser satisfeita, na qual f denota a fase sólida ou líquida.

A localização da interface é determinada pela condição de Stefan

$$L\rho \frac{\partial X}{\partial t} = k_s \frac{\partial T_s}{\partial x} - k_l \frac{\partial T_l}{\partial l}, \tag{5.6}$$

que expressa o balanço de calor na interface.

O problema de Stefan pode ser classificado em problemas de uma fase e de duas fases. No problema de Stefan de uma fase, os cálculos da temperatura são efetuados na parte líquida apenas, considerando que o sólido esteja à temperatura T_m. Pode-se considerar um sólido semi-infinito, ocupando a posição $(X(t), \infty)$. No outro caso, a distribuição de temperatura é calculada em ambas as fases.

5.3 Método de Front Tracking

O método de Front Tracking consiste em rastrear explicitamente a localização da interface sólido-líquido, a cada iteração no tempo, através da condição da Stefan. Localizada a interface, é possível calcular a distribuição de temperatura pela equação 5.5 referente a cada fase.

Podem-se desenvolver diversos algoritmos baseados em Front Tracking que, a cada passo no tempo, computam a posição do contorno móvel. Quando a solução é calculada em pontos de uma malha fixa no domínio de espaço e tempo, seguindo métodos usuais, a interface geralmente fica entre dois pontos da malha para um tempo dado qualquer. Portanto faz-se necessário uma formulação especial para os pontos na vizinhança da interface móvel.

Foram desenvolvidas diversas maneiras de modificar a malha, com o propósito de evitar complicações e perda de precisão associada com intervalos de espaço desiguais perto da interface [13].

Pode-se manter um número fixo i de intervalos entre $x = 0$ e $x = X(t)$, ou seja, entre os limites fixo e móvel, de modo que a qualquer tempo, Δx seja diferente e que a interface móvel esteja sempre na i-ésima linha da malha.

Para evitar complicações devido à malha de espaçamento variável perto da interface, foi desenvolvido um método que movimenta a malha uniforme inteira com a velocidade de propagação da interface móvel. A cada passo de tempo Δt toda a malha move-se a uma distância ϵ para a esquerda. Portanto o intervalo variável é transferido da região da interface para o início da malha, que é normalmente uma região mais fácil de ser tratada.

Outro método possível incorpora a idéia de ajustar o intervalo temporal de modo que a interface coincida sempre sobre um ponto da malha. Dentre esses métodos surgem pequenas variantes de acordo com a maneira de atualizar Δt.

Segundo [2], o método de Front Tracking apresenta bons resultados para problemas de Stefan simples, em que são encontradas interfaces abruptas que separam as duas fases. Por exemplo, o super-resfriamento constitucional em ligas binárias, ou a transferência simultânea de massa devido à difusão e convecção, complicam o processo de mudança de fase, de modo que o problema não pode ser formulado como um problema de Stefan com interface abrupta. Essas dificuldades já surgem em simulações unidimensionais, e se amplificam à medida que cresce o número de dimensões do problema, tornando esses métodos complicados para modelar processos de mudança de fase, de modo preciso.

5.4 Método de Entalpia

No método de Entalpia, a localização da interface pode ser determinada *a posteriori* a partir dos valores da entalpia, podendo ser caracterizado como um esquema de Front-Capturing, similar aos esquemas de Shock-Capturing em dinâmica de gases [7].

A função de entalpia determina que para $h \leq 0$, a substância encontra-se no estado sólido, para $h \geq \rho L$, no estado líquido e para valores $0 < h < \rho L$, a substância está parcialmente líquida e sólida, conhecida como região *mushy*.

Pode-se estabelecer um algoritmo que calcule a entalpia a partir da lei de Fourier

$$\frac{\partial(\rho h)}{\partial t} = \nabla \cdot (k \nabla T), \tag{5.7}$$

e com base nesse valor determinar o campo de temperatura através das relações:

$$T = \begin{cases} T_m + \dfrac{h - \rho L}{\rho c_l} & h \geq \rho L \quad \text{no líquido,} \\ T_m & 0 < h < \rho L \quad \text{na interface,} \\ T_m + \dfrac{h}{\rho c_s} & h \leq 0 \quad \text{no sólido.} \end{cases}$$

O método de entalpia também apresenta variações e adaptações que podem ser encontradas na literatura.

5.5 Formulação do Problema de Mudança de Fase

Estuda-se a solidificação de uma liga binária formada por dois componentes A e B. Assume-se que o processo de solidificação seja governado localmente por um diagrama de fases.

Em especial, será considerada a liga binária semicondutora PbSnTe, que é um material utilizado para detectores de infravermelho e lasers de diodo. Uma das dificuldades encontradas no crescimento desse material é obter cristais de composição homogênea. Através do diagrama de fases da liga PbTe-SnTe, nota-se que o componente SnTe é rejeitado da interface sólido-líquido quando ocorre a solidificação, formando uma camada enriquecida de SnTe no lado líquido da interface.

No ambiente gravitacional da Terra, essa camada tende a desaparecer rapidamente através de fluxos convectivos no líquido, e o cristal cresce contendo um enriquecimento gradual do componente SnTe. Uma maneira possível de melhorar a homogeneidade do cristal é realizar o crescimento em ambiente de microgravidade, onde a convecção pode ser desconsiderada e um crescimento puramente difusivo é realizado. Esses cristais apresentam uma região de concentração constante exceto pelas regiões transientes inicial e final. Para obter esse tipo de crescimento, em gravidade normal, é necessário uma ampola de diâmetro reduzido o sufiente para evitar fluxos convectivos [27].

Na simulação de crescimento de cristais, uma prática comum é considerar casos nos quais somente um processo seja o dominante, geralmente a difusão, e desprezando os demais. Essa simplificação visa evitar o acoplamento dos processos de transferência de massa e de calor. Mesmo assim, o problema apresenta complexidade, uma vez que a localização da interface é desconhecida, enquadrando-se nos problemas de

fronteira móvel. Nessa classe de problemas, um dos mais estudados é o problema de Stefan, que modela a solidificação de um material puro, onde as variáveis a serem determinadas são a distribuição de temperatura e a localização da interface. Esse problema considera a condução de calor como dominante e a convecção no líquido como desprezível [2].

Foi implementado, por Fabbri [17], um método numérico de malha fixa, baseado numa aproximação em volumes de controle, capaz de operar com condições de contorno e propriedades do material variáveis. Os detalhes do diagrama de fases da liga são incluídos através de uma equação de estado apropriada, considerando-se o método de Solomon, Alexiades and Wilson [39].

A integração no tempo é realizada de forma que qualquer célula retenha seu estado de fase durante o passo de tempo, e uma mudança somente ocorra no final do intervalo de tempo adotado. Esta metodologia é necessária por causa de grandes descontinuidades nas entalpias específicas e no calor específico de um volume de controle quando ele entra ou deixa uma região de mudança de fase.

O uso de um método de malha fixa implica que a frente de mudança de fase não pode ser localizada precisamente em um ponto dentro de um volume de controle, e que são necessárias algumas hipóteses adicionais para modelar as propriedades físicas de uma célula sob mudança de fase. Investigações prévias, com solidificação de substâncias puras, indicam que essas aproximações não impossibilitam a obtenção de bons resultados em relação à velocidade e ao formato das interfaces sólido-líquido.

De acordo com o método de Solomon, Alexiades e Wilson [39], [3], [1], é utilizada uma formulação global de forma que as leis de conservação sejam válidas para todo o domínio, com a adoção da fração líquida definida por

$$\lambda = \begin{cases} 1 & \text{no líquido,} \\ \dfrac{C - g_s(T)}{g_l(T) - g_s(T)} & \text{na região } \textit{mushy}, \\ 0 & \text{no sólido,} \end{cases}$$

de modo que uma propriedade G, num volume de controle j seja dada por

$$G^j = \lambda G_l + (1 - \lambda) G_s \ . \tag{5.8}$$

A energia interna de uma liga pode ser escrita como

$$e = e(T, C) \ , \tag{5.9}$$

que é válida para qualquer região do diagrama de fases: sólida, líquida ou *mushy*, podendo ser representada por

$$e = \lambda e_l + (1 - \lambda) e_s \ , \tag{5.10}$$

e

$$e_l = \begin{cases} e_l(T, C) & \text{se } \lambda = 1, \\ e_l(T, g_l) & \text{se } 0 < \lambda < 1, \\ 0 & \text{se } \lambda = 0, \end{cases}$$

$$e_s = \begin{cases} e_s(T, C) & \text{se } \lambda = 0, \\ e_s(T, g_s) & \text{se } 0 < \lambda < 1, \\ 0 & \text{se } \lambda = 1. \end{cases}$$

A forma explícita da equação de estado é obtida pela integração da relação de Gibbs:

$$de = cdT + hdC. \tag{5.11}$$

A energia interna de qualquer ponto na região líquida pode ser obtida através de

$$e_l(T, C) = L^A + \int_0^C h_l(T^A, \epsilon) d\epsilon + \int_{T^A}^T c_l(\tau, C) d\tau \ , \tag{5.12}$$

e na região sólida por

$$e_s(T, C) = \int_0^C h_s(T, \epsilon) d\epsilon + \int_{T^A}^T c_s(\tau, 0) d\tau \ , \tag{5.13}$$

em que L^A é o calor latente de fusão da substância pura A.

Assume-se o comportamento mais simples possível das propriedades termofísicas h (entalpia parcial específica) e c (calor específico). Supõe-se que c_l e c_s são fornecidos como dados experimentais para as substâncias puras A e B, e que a dependência

destes com a temperatura possa ser negligenciada. Para uma primeira aproximação, assume-se que c seja função somente de C e que h seja função de T somente, de forma que a equação 5.12 seja reescrita como

$$e_l(T, C) = L^A + h_l(T^A)C + c_l(C)(T - T^A). \tag{5.14}$$

Adotando o mesmo procedimento para a região sólida:

$$e_s(T, C) = h_s(T)C + c_s(0)(T - T^A). \tag{5.15}$$

A compatibilidade termodinâmica deve ser imposta entre as equações 5.14 e 5.15, e as condições suficientes são que h_l e h_s sejam funções lineares de T e que c_l e c_s sejam funções lineares de C.

Portanto:

$$\begin{aligned}
c_l(C) &= [c_l(1) - c_l(0)]C + c_l(0), && (5.16)\\
c_s(C) &= [c_s(1) - c_s(0)]C + c_s(0), && (5.17)\\
h_l(T) &= [c_l(1) - c_l(0)](T - T^A) + h_l(T^A). && (5.18)\\
h_s(T) &= [c_s(1) - c_s(0)](T - T^B) + h_s(T^B). && (5.19)
\end{aligned}$$

Uma relação adicional vem da condição

$$e_l(T^A, 0) - e_s(T^A, 0) = L^A, \;\; e_l(T^B, 0) - e_s(T^B, 0) = L^B, \tag{5.20}$$

a qual impõe que

$$h_l(T^A) - h_s(T^B) = L^B - L^A + (T^B - T^A)[c_s(0) - c_l(1)]. \tag{5.21}$$

As equações 5.17, 5.19, 5.20 e 5.21 definem a equação de estado, para uma primeira aproximação, em termos de valores, experimentais conhecidos, c_l, c_s e L para os componentes puros A e B.

5.5.1 Equações Básicas

São basicamente as mesmas equações do método de Solomon, Alexiades e Wilson [1], com algumas alterações no cálculo dos valores dos coeficientes de transporte da interface. Uma aproximação por volumes de controle é considerada em todas as relações básicas.

São desprezadas as diferenças entre as densidades do sólido e líquido, que são assumidas como constantes e também é desprezada a convecção no líquido. Todos esses efeitos devem ser importantes durante o crescimento de cristais, e poderão ser incluídos no modelo sem modificações essenciais.

5.5.1.1 Transporte de Massa

A continuidade de massa é estabelecida por

$$C_t = -\nabla \cdot \vec{J}, \tag{5.22}$$

em que

$$C = \lambda C_l + (1 - \lambda) C_s, \tag{5.23}$$

e

$$C_l = \begin{cases} 0 & \text{se } \lambda = 0, \\ g_l(T) & \text{se } 0 < \lambda < 1, \\ C & \text{se } \lambda = 1, \end{cases}$$

$$C_s = \begin{cases} C & \text{se } \lambda = 0, \\ g_s(T) & \text{se } 0 < \lambda < 1, \\ 0 & \text{se } \lambda = 1. \end{cases}$$

Os fluxos de massa entre células são dados usando-se a lei de Fick com pequenas modificações. O fluxo em uma dada interface entre uma célula P e uma das suas vizinhas NG é expresso por

$$\vec{J}_{\text{interf}} = \min(\lambda_P, \lambda_{\text{NG}})[-D_{\text{interf}}(\nabla C_l)] \ , \tag{5.24}$$

em que $\min(\lambda_P, \lambda_{\mathrm{NG}})$ é a medida do contato líquido-líquido e

$$D_{\mathrm{interf}} = \frac{2 D_P D_{\mathrm{NG}}}{D_P + D_{\mathrm{NG}}}. \tag{5.25}$$

$D_P = D_l$ se $\lambda \neq 0$, o coeficiente de difusão mútua usual no líquido e $D_P = D_s$ se $\lambda = 0$.

5.5.1.2 Transporte de Energia

Aplicando a relação de Gibbs 5.11 na equação 5.10 tem-se

$$de = \lambda de_l + (1-\lambda) de_s + (e_l - e_s) d\lambda \tag{5.26}$$

$$= \lambda \left[\frac{\partial e_l}{\partial T} dT + \frac{\partial e_l}{\partial C} dC \right] + (1-\lambda) \left[\frac{\partial e_s}{\partial T} dT + \frac{\partial e_s}{\partial C} dC \right] + (e_l - e_s) \left[\frac{\partial \lambda}{\partial T} dT + \frac{\partial \lambda}{\partial C} dC \right] \tag{5.27}$$

$$= \left[\lambda \frac{\partial e_l}{\partial T} + (1-\lambda) \frac{\partial e_s}{\partial T} + (e_l - e_s) \frac{\partial \lambda}{\partial T} \right] dT + \left[\lambda \frac{\partial e_l}{\partial C} + (1-\lambda) \frac{\partial e_s}{\partial C} + (e_l - e_s) \frac{\partial \lambda}{\partial C} \right] dC \tag{5.28}$$

$$= \left[\lambda c_l + (1-\lambda) c_s + L \frac{\partial \lambda}{\partial T} \right] dT + \left[\lambda h_l + (1-\lambda) h_s + L \frac{\partial \lambda}{\partial C} \right] dC \tag{5.29}$$

Aplicando a equação obtida acima na equação de energia

$$\rho e_t = \nabla \cdot (k \nabla T) + \rho \nabla \cdot (h D \nabla C) \tag{5.30}$$

obtém-se

$$\rho c T_t = -\nabla \cdot \vec{Q} - \rho \nabla \cdot (h \vec{J}) + \rho h \nabla \cdot \vec{J}, \tag{5.31}$$

com

$$c = \lambda c_l + (1-\lambda)c_s + L\frac{\partial \lambda}{\partial T}, \tag{5.32}$$

$$h = \lambda h_l + (1-\lambda)h_s + L\frac{\partial \lambda}{\partial C}, \tag{5.33}$$

$$L = L(T) = e_l - e_s. \tag{5.34}$$

O fluxo de calor entre células obedece a lei de Fourier usual considerando as condutividades da interface:

$$\vec{Q}_{\text{interf}} = -k_{\text{interf}} \nabla T, \tag{5.35}$$

$$k_{\text{interf}} = \frac{2k_P k_{\text{NG}}}{k_P + k_{\text{NG}}}. \tag{5.36}$$

Adota-se para $k_P(C,T)$ um modelo linear simples que interpola entre os valores experimentais k_l e k_s para substâncias puras A e B, e estende-se esse resultado para a região *mushy*:

$$k_P(C,T) = \begin{cases} [k_l(1) - k_l(0)]C + k_l(0) & \text{no líquido,} \\ \dfrac{k(g_l) - k(g_s)}{g_l - g_s}(C - g_s) + k(g_s) & \text{na região } mushy, \\ [k_s(1) - k_s(0)]C + k_s(0) & \text{no sólido.} \end{cases} \tag{5.37}$$

O valor de entalpia parcial específica na interface

$$h_{\text{interf}} = \frac{h_P + h_{\text{NG}}}{2}. \tag{5.38}$$

A difusividade e a condutividade térmica dependem, em geral, da temperatura e da concentração e podem apresentar descontinuidades do líquido para o sólido.

5.5.2 Discretização Adotada

As equações de concentração e energia são discretizadas por volumes de controle, para uma malha cartesiana regular cobrindo o plano X-Y, definido em função da

geometria cilíndrica da ampola, sendo que X refere-se à direção axial e Y à radial. Emprega-se a notação usual de Patankar [34].

Adota-se que o ponto a ser calculado é representado por **P** e seus vizinhos por **W**, **E**, **S** e **N** que tratam-se, respectivamente, dos pontos à esquerda, à direita, abaixo e acima do ponto **P**. Essas mesmas letras em minúsculo, representam as paredes das células, uma vez que os pontos estão localizados no centro dos volumes de controle.

Assim, a equação de discretização pode ser expressa por

$$a_P X_P = a_W X_W + a_E X_E + a_S X_S + a_N X_N + b, \tag{5.39}$$

no qual X representa a incógnita a ser calculada e b é uma constante conhecida para cada ponto da malha de discretização.

Dessa forma, as equações discretizadas finais para concentração e energia são dadas por

$$\frac{C_P - C_P^0}{\Delta t}\Delta x \Delta y = \sum_{J=E,W,N,S} a_J \min(\lambda_P, \lambda_J) D_J (C_{lJ} - C_{lP}), \tag{5.40}$$

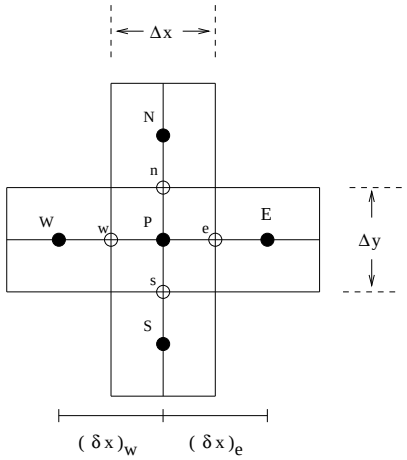

Fig. 5.1 – Volume de controle para o caso bidimensional, ilustrando a célula **P** e as células vizinhas.

$$c_P \frac{T_P - T_P^0}{\Delta t} \Delta x \Delta y = \sum_{J=E,W,N,S} a_J \left\{ \frac{k_J}{\rho}(T_J - T_P) + \frac{h_J - h_P}{2} \min(\lambda_P, \lambda_J) D_J (C_{lJ} - C_{lP}) \right\} + \frac{\Delta x \Delta y}{\rho}(S_P^C + S_P^P T_P), \quad (5.41)$$

na qual

$$a_E = a_W = \frac{\Delta y}{\Delta x}, \quad a_N = a_S = \frac{\Delta x}{\Delta y} \quad \text{e} \quad C_{li} = C_{\text{liq}} \quad \text{na i-ésima célula.}$$

Um termo fonte linearizado foi introduzido na equação de energia, na forma $S^C + S^P T$, para se poder manipular facilmente as condições de contorno dependentes do tempo. A condição de contorno para a equação da concentração é um fluxo de massa nulo no contorno externo e é automaticamente incluído na equação 5.40. Nos experimentos a serem realizados, o domínio será considerado como cercado por paredes rígidas de uma dada espessura, a qual não participa da transferência de massa. As propriedades da parede serão introduzidas por valores adequados de k e c.

5.5.3 Algoritmo Proposto

A equação de estado $e(T, C)$ não é diferenciável sobre as curvas *liquidus* e *solidus*. O calor específico e as entalpias parciais apresentam grandes descontinuidades, causadas pelos termos $L\partial\lambda/\partial T$ e $L\partial\lambda/\partial C$, quando uma célula entra ou deixa a região de mudança de fase. Um esquema especial de integração foi adotado, no qual cada célula retém seu estado de fase durante o passo de tempo; qualquer célula pode entrar ou sair da região de mudança de fase somente no final de um intervalo de tempo, e quando um erro envolvido é estimado como dentro de uma certa tolerância aceitável. Dessa maneira é utilizado um valor de Δt conveniente, que é reduzido caso a atualização do estado de fase seja requerido em alguma célula. O esquema resultante é apresentado no algoritmo abaixo. A redução de Δt, quando necessária, será realizada através de uma interpolação linear simples entre os valores novos e velhos da energia interna $e(T, C)$.

Erros admissíveis são todos baseados no parâmetro de entrada, o erro relativo

permitido para a temperatura, ϵ_T. Estimativas para os erros aceitáveis no campo de concentração , ϵ_C, e an energia interna, ϵ_e, são obtidas através da análise de pior caso para os dados de entrada das propriedades físicas e do diagrama de fases. ϵ_e é utilizado como o erro máximo permitido quando uma célula entra ou deixa uma região de mudança de fase.

A solução para os campos T-C são obtidos através de varreduras das linha através de um procedimento Gauss-Seidel simples. O acomplamento entre as equações de T e C são tratadas considerando que, em experimentos de solidificação direcional, a mudança de fase é essencialmente dirigida por mudanças térmicas.

Conhecendo-se os valores de C e T, é possível atualizar C e e no próximo passo de tempo, e usar a definição de e para encontrar T, análogo ao método de entalpia para o problema de Stefan. Isto requer a inversão da energia, ou seja, resolver para T, a equação $e(C,T) = e$.

O algoritmo do problema é apresentado de uma maneira simplificada, nos seguintes passos:

a) inicializar tempo: $t = 0$;

b) inicializar campos T, C, e, $phase$, c, h, k, λ;

c) inicializar passo de tempo: $\Delta t = (\Delta t)_0$;

d) atualizar tempo: $t = t + \Delta t$;

e) testar condição de parada referente ao tempo máximo de execução do programa: *if* $t > t_{final}$, $STOP$;

f) testar condição de parada referente ao estado das células: se todas forem sólidas encerrar o programa, pois solidificação foi concluída;

g) calcular e atribuir condições de contorno;

h) calcular novos campos de C e T:

- calcular C^{new}, até atingir convergência;
- atualizar λ, g_l, g_s, c, k, h e $phase$;
- calcular T^{new}, até atingir convergência;

- atualizar $e(C^{new}, T^{new})$;

- atualizar C;

- calcular $T(e, C)$;

i) verificar consistência dos campos T, C e e:
se forem consistentes, dentro da precisão requerida, voltar ao passo d;
se não forem consistentes:

- voltar ao tempo anterior: $t = t - \Delta t$;

- reduzir Δt;

- voltar, ao valor anterior, de T, C e e;

- retornar ao passo d;

5.5.4 Considerações sobre Técnicas Paralelas para a Resolução das Equações Diferenciais Parciais

Equações diferenciais parciais podem ser aproximadas por diferenças finitas e ser resolvidas por métodos iterativos, tais como Jacobi e Gauss-Seidel. Esses esquemas são inerentemente paralelos, de forma que basta apenas organizá-los para poder distribuir os dados e/ou tarefas entre os processadores.

Por exemplo, a equação 5.22 é discretizada na seção 5.5.2, resultando na equação 5.40. A derivada temporal

$$\frac{\partial C}{\partial t} \tag{5.42}$$

é aproximada pela diferença

$$\frac{C_P^{\text{new}} - C_P^{\text{old}}}{\Delta t}, \tag{5.43}$$

e a derivada espacial

$$-\nabla \cdot \vec{J} \tag{5.44}$$

por

$$\sum_{J=E,W,N,S} a_J \min(\lambda_P, \lambda_J) D_J (C_{lJ} - C_{lP}). \tag{5.45}$$

Essas diferenças espaciais podem ser expressas no tempo *new* ou *old*, caracterizando o método como ímplicito ou explícito.

O método de Jacobi é um esquema explícito, resolvido ponto-a-ponto. C_P^{new} pode ser determinado em função dos valores adjacentes já conhecidos, da iteração passada, de forma que os cálculos para todos os pontos são independentes e podem ser realizados em paralelo, sem nenhum empecilho.

O método de Gauss-Seidel também é um esquema explícito, no qual o ponto C_P^{new} é calculado utilizando os valores mais recentes à medida que eles são disponibilizados na iteração corrente. Para os pontos ainda não atualizados são utilizados os valores da iteração passada. Esse método apresenta convergência mais rápida que o Jacobi. Porém os cálculos dos pontos não são totalmente independentes de forma que eles não podem ser atualizados todos ao mesmo tempo, como no Jacobi.

Entretanto esse método pode ser ligeiramente modificado para que os cálculos possam ser realizados em dois níveis de iteração a cada passo de tempo, o qual é exemplificado na figura 5.2. Os pontos pretos são calculados no primeiro nível, utilizando os valores dos brancos da iteração passada, e os pontos brancos são determinados no segundo nível, a partir dos valores dos pretos já atualizados no nível anterior, porém no passo de tempo corrente. Desse modo os cálculos em cada nível de tempo podem ser realizados em paralelo.

Os métodos ímplicitos para problemas bidimensionais resultam em matrizes penta-diagonais a serem resolvidas, as quais tornam-se computacionalmente dispendiosas. Uma saída, para este caso, seria realizar os cálculos de forma implícita em uma dimensão e explícita na outra, sendo resolvido para linhas e/ou colunas, conforme a direção preferencial do problema.

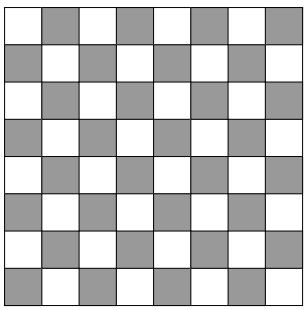

Fig. 5.2 – Tabuleiro de xadrez para o método de Jacobi.

Uma das dificuldades encontradas para a resolução desses problemas é determinar qual a direção que deve ser tratada de forma implícita e qual de forma explícita. Tem-se como possibilidade o método Implícito de Direções Alternadas, o qual realiza varredura das linhas e colunas de forma alternada, como diz o nome do método.

Adotando como exemplo que os cálculos na linha são implícitos tem-se que uma linha inteira de pontos é atualizada simultaneamente, através da resolução de um sistema tridiagonal. Esses cálculos utilizam os valores das colunas inferior e superior, e o tempo no qual esses valores são tomados definem o método como Jacobi de linhas ou Gauss-Seidel de linhas, que utilizam respectivamente os valores da iteração passada e os valores disponíveis da iteração atual.

O método Jacobi de linhas define linhas independentes, as quais podem ser calculadas ao mesmo tempo, portanto, em paralelo.

O método Gauss-Seidel de linhas define uma sequência de linhas a serem calculadas, de forma que as linhas não podem ser atualizadas simultaneamente. Porém, como no método de Gauss-Seidel, pode ser adotado um esquema de atualização em dois níveis de iteração, a cada intervalo de tempo, o qual define linhas pretas a serem atualizadas no primeiro nível e linhas brancas no segundo nível, como pode ser visto na figura 5.3.

Pode ser incorporado um fator de relaxação ω para auxiliar na convergência da solução. Os valores novos são calculados no tempo atual através do esquema adotado. Porém para atualizar os pontos são utilizados, além dos valores recém obtidos, os valores da iteração passada:

$$C_P^{\text{new}} = (1-\omega)C_P^{\text{old}} + \omega C_P^{\text{new}}, \tag{5.46}$$

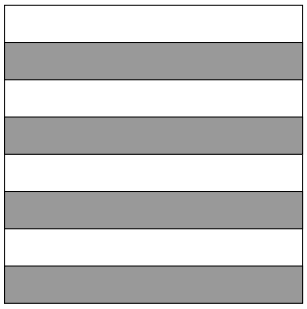

Fig. 5.3 – Tabuleiro de linhas para o método de Gauss-Seidel.

resultando no esquema de sobre-relaxação sucessiva de linhas.

Esses métodos numéricos podem ser vistos com mais detalhes em [22] e as técnicas paralelas em [30].

O processamento de alto desempenho, as arquiteturas paralelas, os paradigmas de programação e as linguagens Fortran 90 e HPF são descritos no próximo capítulo, juntamente com as definições de *speedup*, eficiência e Lei de Amdahl.

CAPÍTULO 6

Processamento de Alto Desempenho

A utilização crescente da computação em todos os ramos da ciência e, consequentemente, a execução de programas cada vez mais dispendiosos manipulando volumes maiores de dados implicam em um tempo de processamento maior também. Para que os resultados gerados por esses programas possam ser obtidos em tempo viável torna-se necessária a utilização de máquinas mais potentes, ou seja, mais rápidas.

Durante as últimas décadas, a evolução do computador digital, em termos de hardware e software, foi suprindo esta demanda. Entretanto, melhoras posteriores no desempenho de máquinas monoprocessadas dependem da adoção de novas tecnologias, que podem tardar a serem implementadas devido a limitações de custo e que continuam sujeitas às limitações físicas do hardware, tais como a velocidade de propagação máxima no meio condutor utilizado. Houve também o aperfeiçoamento de linguagens tais como o Fortran e a evolução dos compiladores otimizantes, mas há um limite em termos do desempenho que se consegue extrair de uma determinada máquina monoprocessada.

Assim, uma alternativa promissora é o processamento paralelo, no qual são utilizados dois ou mais processadores, partindo-se do pressuposto de que o conjunto de tarefas ou o domínio do problema possa ser dividido entre os processadores e o resultado seja correto. Para que essa divisão possa ser implementada, foram desenvolvidas linguagens paralelas e bibliotecas e extensões de linguagens existentes para paralelismo.

Define-se um computador paralelo como sendo um conjunto de processadores que são capazes de trabalhar cooperativamente para resolver um problema computacional [19]. Há outros níveis de paralelismo, mesmo nas máquinas monoprocessadas, iniciando-se pelo paralelismo de instruções implementado pelos "pipelines" de instruções, no qual estas são executadas concorrentemente, sendo possível obter a taxa de uma instrução executada a cada ciclo do relógio, no caso do processador escalar. Isso implica num paralelismo de hardware, para permitir que cada estágio do "pipeline" possa ser executado independentemente dos demais. A seguir, há o paralelismo das máquinas superescalares, em que a arquitetura do processador

implementa vários "pipelines" independentes, por exemplo, um para operações com inteiros, outro para pontos flutuantes, etc. Finalmente, há o paralelismo devido à existência de mais de um processador, numa máquina multiprocessada ou num multicomputador. Processadores vetoriais, que podem ou não serem multiprocessados, implementam ainda uma outra forma de paralelismo, através de instruções, acesso a memória, registradores e "pipelines" específicos para vetores.

No tocante à programação, o problema usual é portar um código escrito para uma máquina monoprocessada para uma máquina paralela, o que exige conhecimento de sua arquitetura, referente à quantidade e topologia de comunicação dos processadores e ao arranjo da memória física. Por exemplo, a partir da linguagem Fortran 77, foi desenvolvido o Fortran 90, com novos comandos e estruturas de dados, sendo este último compatível com o High Performance Fortran (HPF), uma extensão do Fortran 90 que inclui diretivas para paralelização, permitindo executar um programa utilizando vários processadores.

Ao se portar o código para uma máquina paralela, dois objetivos são fundamentais, sendo dependentes da arquitetura acima citada: tentar obter-se um balanceamento de carga entre os processadores e minimizar a troca de dados entre estes. O primeiro ponto está associado com a utilização eficiente dos processadores, enquanto que o segundo está relacionado às dependências dos dados utilizados pelos processadores, o que também compromete o desempenho global, uma vez que um processador pode ter que esperar por um resultado fornecido por outro ou, no caso de a memória não ser compartilhada entre os processadores, estes podem ser sobrecarregados por mensagens relativas à troca de dados.

6.1 Arquitetura Paralela

Basicamente, qualquer computador paralelo é caracterizado por três elementos principais: conjunto de processadores, memória e rede de interconexão para habilitar a comunicação entre esses elementos. Geralmente os processadores fazem acesso a memórias locais através de caches. O conjunto de todas essas memórias locais compõe o sistema de memória genérica. De acordo com a forma de organizção da memória, em máquinas paralelas, pode-se representar arquiteturas diferentes, as quais são usualmente classificadas, de acordo com a organização do espaço de endereçamento da memória, em dois grupos:

- máquinas de memória distribuída, conhecidas também como máquinas com espaços de endereçamento múltiplos;
- máquinas de memória compartilhada ou máquinas com um espaço de endereçamento único.

Em um sistema de memória distribuída, comumente denominado de multicomputador, cada processador possui sua própria memória, pode fazer acessos a ela através da memória cache e encontra-se conectado a uma rede de interconexão, ilustrado de maneira genérica na figura 6.1. Para um processador ter acesso a dados remotos, isto é, dados que estão na memória de outros processadores, torna-se necessária a comunicação por mensagem, por exemplo, através de bibliotecas específicas, programada pelo usuário. Eventualmente pode-se utilizar uma linguagem, como o High Performance Fortran (HPF), na qual o compilador implementa a comunicação automaticamente. O tempo de acesso depende da memória que está sendo requisitada: o acesso à memória local é mais rápido do que o acesso à memória remota.

Em um sistema de memória compartilhada, comumente denominada de multiprocessador, cada processador está conectado através da rede de interconexão aos elementos de memória, ilustrado de maneira genérica na figura 6.2. Uma maneira de implementar essa arquitetura é conectar processadores, memória e periféricos através de um barramento de transferência de dados. A programação, nesse tipo de sistema, é mais fácil do que em uma arquitetura de memória distribuída, não requerendo comunicação por mensagem, porém pode ocorrer um congestionamento quando vários processadores tentam utilizar o barramento simultaneamente. Os dados também podem ser comunicados, eventualmente, através de comunicação por mensagem, utilizando ou não uma rede de interconexão, fazendo com que o

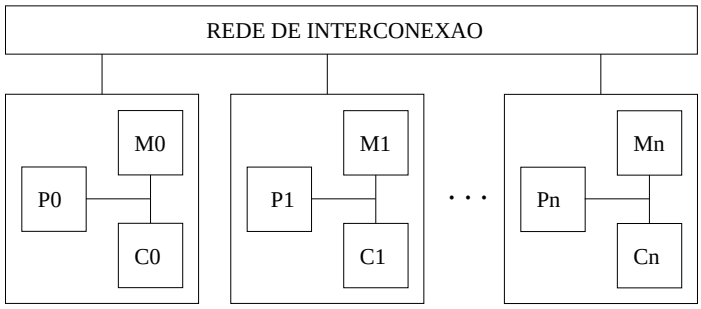

Fig. 6.1 – Arquitetura de Memória Distribuída.

código possa ser, nesse caso, compatível com a arquitetura de memória distribuída. No caso de acesso através de um barramento sem troca de mensagem, o tempo de acesso à memória costuma ser uniforme para os processadores (Uniform Memory Access - UMA), nas máquinas classificadas como SMP (symmetric multi-processors), geralmente restritas a 16 ou 32 processadores, devido à limitação de largura de banda do barramento. Por outro lado, pode-se ter a ligação entre os processadores e a memória através de uma rede de interconexão chaveada dinamicamente por um comutador (switch) ou não, com tempo de acesso à memória variável (Non-uniform Memory Access - NUMA).

6.2 Paradigmas de Programação

Há, basicamente, dois tipos de estilos ou paradigmas de programação: paralelismo de tarefas e paralelismo de dados. Tipicamente, em máquinas paralelas o problema a ser resolvido é dividido pelos processadores disponíveis, com o objetivo de produzir o mesmo resultado que uma máquina monoprocessada. Pode-se dividir as tarefas, ou seja, atribuir tarefas funcionalmente diferentes a processadores diferentes, caracterizando o paralelismo de tarefas. Por outro lado, pode-se dividir os dados de forma que cada processador execute a mesma função, porém em conjuntos de dados distintos, paradigma conhecido como SPMD (single program multiple data) que geralmente caracteriza o paralelismo de dados. Haveria ainda um esquema híbrido no qual tarefas e dados são divididos pelos processadores, caracterizando o paralelismo de tarefas e dados.

Na prática, para programação paralela, utilizam-se extensões de linguagens conhecidas ou bibliotecas para comunicação entre processadores, utilizadas

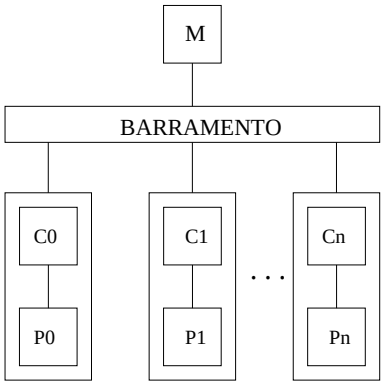

Fig. 6.2 – Arquitetura de Memória Compartilhada.

conjuntamente com essas linguagens.

Tipicamente, nas máquinas de memória distribuída, o trabalho de programação é maior pois, deve-se programar explicitamente utilizando-se rotinas de comunicação de baixo nível, visando obter balanceamento de carga entre processadores e minimização da comunicação entre processadores. Na comunicação por mensagem, as interações entre processadores são realizadas por chamadas a rotinas de comunicação. O programador pode ter controle completo sobre a distribuição de dados, de tarefas e a comunicação entre processadores.

Por exemplo, cada processador pode ficar a cargo de submatrizes de uma determinada matriz. No entanto, se houver dependência de dados entre submatrizes, como cada processador somente tem acesso à submatriz que está em sua memória local, somente poderá fazer acesso a elementos de outra submatriz através da comunicação por mensagem. Isto requer que o programador especifique explicitamente, a troca de mensagens necessárias a esse fim.

A execução das tarefas não precisa ser executada de maneira sincronizada em todos os processadores. Entretanto algumas estruturas de programação, tais como laços, forçam a sincronização, fazendo com que um processador tenha que esperar que outro conclua uma dada instrução.

A necessidade de comunicação entre processadores introduz claramente uma sobrecarga adicional na execução do programa, constituindo uma parcela significativa da programação de multicomputadores. A comunicação pode ser requerida para mover dados de um processador para outro, ou para que as tarefas possam ser divididas de forma eficiente entre os processadores, sendo feita, em ambos os casos, de forma explícita pelo programador. Claramente, há um compromisso entre o custo da comunicação e o aumento de desempenho devido à paralelização.

Nas linguagens em que o compilador determina, automaticamente, a troca de mensagens entre processadores ficam ocultos os aspectos relacionados à memória distribuída e à troca de mensagens. Assim, nessas linguagens pretende-se transferir para o compilador a programação de baixo nível, como por exemplo aquela relativa à troca de mensagens. Esse é o caso do High Performance Fortran (HPF), extensão do Fortran 90, descrito na próxima seção, que implementa diretivas de paralelização para, por exemplo, dividir os dados a serem processados entre os

diversos processadores de uma máquina multiprocessada ou de um multicomputador.

Por outro lado, bibliotecas de comunicação implementam explicitamente as trocas de mensagens entre processadores, sendo utilizadas na programação paralela em Fortran 77, Fortran 90, C e C++. É o caso das bibliotecas MPI (Message Passing Interface) e PVM (Parallel Virtual Machine), aplicáveis a conjuntos heterogêneos de processadores formados por máquinas multiprocessadas e/ou por multicomputadores. O MPI define o conjunto de processadores utilizados através de um arquivo de configuração, enquanto que o PVM pode utilizar uma interface gráfica com o usuário, por exemplo, o XPVM.

Aborda-se, a seguir, o HPF, que será utilizado para programação paralela neste trabalho.

6.3 Linguagem Fortran em Aplicações Paralelas

6.3.1 Fortran 90

De acordo com [29], o Fortran 77 tem sido usado amplamente por cientistas e engenheiros ao longo dos anos, porém ficou defasado em face a novas linguagens. O Fortran padrão foi atualizado resultando no Fortran 90. O Fortran, cujo nome derivou de *IBM Mathematical FORmula TRANslation System*, foi desenvolvido por um grupo na IBM na década de 1950, tornando popular em 1960, quando surgiram diversas versões, até que em 1963 encontravam-se disponíveis cerca de quarenta compiladores diferentes. Devido à dificuldade gerada por essa diversidade, em 1972 foi padronizado o Fortran 66. Com a facilidade de se portar programas de um sistema para outro, o Fortran tornou-se a linguagem mais utilizada para aplicações científicas e tecnológicas.

No final da década de 1970, houve uma nova atualização publicada pelo *American National Standards Institute*, que mais tarde foi adotada pelo *International Standards Organisation* (ISO), como um padrão internacional conhecido como Fortran 77.

Devido às limitações intrínsecas da linguagem iniciou-se, em 1980 uma reformulação completa da linguagem que levaria, no início da década de 90, ao Fortran 90, o qual provê compatibilidade com o Fortran 77.

O Fortran foi uma das linguagens mais difundidas no meio técnico-científico devido ao seu pioneirismo. Criou-se assim uma comunidade de usuários que resistiu ao surgimento de novas linguagens, seja pelo costume de utilizar FORTRAN ou seja para preservar milhões de linhas de código já escritas. Ainda hoje, em muitas áreas de pesquisa ou de engenharia, utiliza-se exclusivamente Fortran. Seria inviável reescrever todos esses softwares numa linguagem mais moderna, ou mesmo numa versão mais moderna de Fortran, pois isso demandaria equipes enormes de programadores e analistas trabalhando durante anos.

O Fortran 90 incorpora novas extensões e características ao Fortran 77, algumas semelhantes a C e C++, porém outras inéditas, tais como as operações com matrizes ou vetores.

Foram incluídas algumas características que a tornam mais adequada a um ambiente de programação moderna, como o formato livre do código, que permite o uso de um conjunto maior de caracteres e oferece mais liberdade na disposição de um comando em uma linha, possibilitando que sejam dispostos até 132 caracteres e não restringindo a utilização das primeiras colunas.

Há novas estruturas de controle, que oferecem maiores facilidades para uma execução condicional. Além do IF e do laço DO ... END DO, há o DO WHILE e o SELECT CASE, o qual permite múltiplos testes. E comandos que permitem interrupção de laços, como CYCLE e EXIT.

A alocação de memória dinâmica, diferentemente da alocação estática que reserva espaço no início da execução, permite alocação e liberação de memória durante a execução do programa. E os ponteiros possibilitam a implementação de estruturas de dados dinâmicas como listas encadeadas e árvores.

Os tipos de dados definidos pelo usuário podem ser criados a partir de tipos já existentes na linguagem.

Pode-se definir módulos (MODULE), os quais agrupam, em arquivos separados, conjuntos de declarações e/ou procedimentos. Isso possibilita que a linguagem controle o acesso a dados e procedimentos de uma forma semelhante ao que se tem nos métodos públicos e privados da linguagem C++. Podem ser utilizados como bibliotecas de dados globais e rotinas.

Na parte de matrizes é que notam-se grandes mudanças. Matrizes e vetores inteiros ou seções podem ser operados em um único comando, por exemplo $\mathbf{A} = \mathbf{A} + \mathbf{B}$, ao invés do *loop* tradicional

do $10\, i = 1, n$
    $\mathbf{A}(i) = \mathbf{A}(i) + \mathbf{B}(i)$
enddo

É possível determinar seções de matrizes de maneira bastante flexível e selecionar apenas os elementos a serem operados. E construtores permitem que sejam feitas atribuições a matrizes ou seções de maneira simplificada.

Matrizes podem ser redimensionadas, alocadas dinamicamente e passadas como argumento para funções, as quais, também podem retornar matrizes, seções ou escalares conforme a operação realizada.

Máscaras podem ser aplicadas, para que seções não regulares de matrizes possam ser operadas, com o uso de WHERE, que especifica os elementos desejados.

Um vetor unidimensional pode ser utilizado como índice de uma das dimensões da matriz, a qual pode ser utilizada em qualquer dos dois lados de uma atribuição. Nota-se porém que quando a matriz está recebendo valores, os elementos não devem ser repetidos, ou seja, o que implicaria numa atribuição múltipla, para preservar a integridade das operações paralelas.

Foram introduzidas novas funções intrínsecas, as quais podem operar com matrizes ou vetores, e retornar matrizes, seções de matrizes ou escalares conforme os argumentos passados a essas funções e às operação realizadas. Por exempo
$REAL\, \mathbf{A}(100, 10, 2)$
...
$\mathbf{A} = SIN(\mathbf{A})$

Entre as novas funções disponíveis pode-se citar funções que realizam reduções como MAXVAL, MINVAL e SUM. Para matrizes de ordem superior, ou seja, de ordem maior que um vetor, tais operações podem realizar redução ao longo de uma dimensão em particular. Há ainda funções que realizam manipulação de matrizes ou vetores como MATMUL (multiplicação de matrizes) e TRANSPOSE (transposta de matrizes) que trabalham com matrizes ou vetores inteiros. Informações a respeito de

matrizes ou vetores podem ser obtidas com o uso de funções como SHAPE, SIZE, LBOUND e UBOUND.

Pode-se também redimensionar matrizes ou vetores através de funções como RESHAPE que permite a criação de uma nova matriz a partir de elementos de uma matriz já existente, SPREAD, a qual replica uma matriz ao longo de uma nova dimensão, MERGE, que copia partes de uma matriz em outra obedecendo uma máscara.

Interfaces de procedimento mais simples e seguras podem ser declaradas, que fornecem ao compilador as informações necessárias para que ele verifique a consistência e assegure que argumentos suficientes estão sendo fornecidos em tempo de execução.

Uma característica nova é a recursão explícita, a qual permite que seja definido funções ou procedimentos que chamem a si próprios.

6.3.2 High Performance Fortran

Fortran 90 é uma linguagem vetorial explícita, que possui as características necessárias para programação em processadores vetoriais. Entretanto, exceto em aplicações específicas, esse tipo de máquina vem sendo substituído por máquinas paralelas nos últimos anos. Isto gerou a necessidade de estender a linguagem de forma a adequá-la ao uso de múltiplos processadores, o que levou ao *High Performance Fortran*(HPF).

As discussões a respeito do HPF começaram no *Supercomputing'91*, onde a *Digital Equipment Corporation* (*DEC*), atualmente pertencente a *Compaq*, organizou um grupo de discussão que levou a formação do *High Performance Fortran Forum* (HPFF), com a colaboração de universidades (particularmente *Rice University* e *Syracuse University*) e de indústrias (*Intel*, *Thinking Machines* e outras), procurando criar novas extensões para o Fortran 90, visando paralelização. A primeiro reunião do HPFF foi em Houston, Texas, em janeiro de 1992. A publicação do HPF versão 1.0 ocorreu em maio de 1993, depois de várias reuniões acontecidos em 1992.

O HPF é uma extensão da linguagem Fortran 90, que suporta programação paralela, seguindo o modelo SPMD, ou seja, tem-se uma programação voltada para o paralelismo de dados, embora permita também paralelismo de tarefas, de maneira

mais limitada.

O HPF inclui diretivas para definir uma grade abstrata de processadores e distribuir os dados a serem processados entre estes. Muitas dessas diretivas podem ser interpretadas como "sugestões" ao compilador, podendo ou não serem adotadas, enquanto que outras permitem indicar ao compilador que não há dependência entre certos conjuntos de dados, provendo maior liberdade ao compilador para sua paralelização. Assim, mesmo um programa em Fortran 90, sem diretivas HPF poderá ser compilado em HPF, sendo que o compilador fará automaticamente a alocação de processadores e a distribuição dos dados.

Em qualquer caso, quer tenham sido incluídas diretivas HPF ou não, a comunicação entre processadores é implementada automaticamente pelo compilador. O programador, através da inserção de diretivas, e o compilador buscam assegurar o balanceamento de carga entre processadores e a minimização de comunicação entre estes.

As diretivas podem ser classificadas em:

- diretivas de especificação;
- diretivas de execução.

São diretivas de especificação:

- PROCESSORS;
- DISTRIBUTE;
- ALIGN;
- TEMPLATE;
- INHERIT;

e diretivas de execução:

- REDISTRIBUTE e REALIGN;
- INDEPENDENT.

6.3.2.1 Diretiva PROCESSORS

Determina o arranjo abstrato dos processadores. Não é obrigatória, pois o compilador pode prover um arranjo padrão. A diretiva pode ser utilizada para nomear o conjunto de processadores e para especificar o número deles em cada dimensão. Algumas regras devem ser observadas quanto ao uso dessa diretiva:

- todas as dimensões da matriz abstrata de processadores devem ser especificadas;
- A matriz de processadores precisa apresentar dimensão igual ao dos dados a serem distribuídos, ou seja, para distribuir uma matriz de dados 2-D é necessário um arranjo de processadores 2-D também;
- o nome do arranjo de processadores não pode ser igual ao nome de variáveis, nem de rotinas;
- elementos correspondentes de arranjos de processadores com o mesmo formato, referem-se ao mesmo processador;
- se dois objetos são mapeados ao mesmo processador abstrato, eles são mapeados ao mesmo processador físico;
- se o formato do arranjo não for especificado significa que o arranjo é escalar.

É possível obter o número e a topologia dos processadores em tempo de execução através das funções, NUMBER_OF_PROCESSOR e PROCESSOR_SHAPE, intrínsecas da linguagem.

O uso da diretiva PROCESSORS define o arranjo abstrato de um conjunto de processadores, que não corresponde ao arranjo físico real decorrente da topologia da máquina.

Por exemplo, a diretiva

!HPF$ PROCESSORS pro(2,2)

especifica o arranjo abstrato bidimensional "pro" de 4 processadores.

6.3.2.2 Diretiva DISTRIBUTE

Especifica a distribuição de cada dimensão da matriz, entre os processadores, a qual pode ocorrer de duas maneiras:

BLOCK faz distribuição da matriz, em blocos, para cada processador. O tamanho do bloco atribuído a cada processador é igual ao número de elementos da matriz dividido pelo número de processadores. Se o número obtido não for exato, o último bloco terá menos elementos que os outros.

É adequada para aplicações nas quais as operações num elemento dependem dos vizinhos mais próximos. Esse tipo de distribuição assegura que os elementos vizinhos estejam, normalmente, no mesmo processador, minimizando a comunicação entre eles.

Por exemplo, a diretiva

!HPF$ DISTRIBUTE (BLOCK) ONTO pro :: A

especifica uma distribuição em blocos do vetor "A" no arranjo de processadores "pro".

CYCLIC distribui os elementos da matriz de forma alternada entre os processadores, o primeiro elemento no processador 1, o segundo no processador 2 e assim por diante, até alcançar o último processador, e a distribuição recomeça pelo primeiro.

Esse tipo de distribuição é comumente empregada quando os cálculos são efetuados com elementos aleatórios, ou seja, quando não se pode assumir uma regularidade do problema. É adequado para as aplicações nas quais o trabalho requerido por cada elemento varie significantemente através da matriz e a localidade não seja importante.

Por exemplo, a diretiva

!HPF$ DISTRIBUTE (CYCLIC) ONTO pro :: A

especifica uma distribuição cíclica do vetor "A" no arranjo de processadores "pro".

Podem aparecer também sob a forma de BLOCK(n) e CYCLIC(n). No primeiro caso os conjuntos de n elementos são distribuídos de forma circular entre os processadores até a distribuição de todos os conjuntos. No caso de BLOCK(n), cada bloco é

distribuído em cada processador, sendo que o número de blocos deve ser menor ou igual ao número de processadores. Um número maior de blocos não está em conformidade com o HPF, significando que o código quebra as restrições impostas pelo HPF e o resultado é indefinido, ou seja, não se pode garantir qual será o comportamento assumido.

Por exemplo, a diretiva

!HPF$ DISTRIBUTE (CYCLIC(2),BLOCK(2)) ONTO pro :: B

especifica uma distribuição cíclica de um conjunto de 2 elementos, das linhas da matriz, e uma distribuição de blocos de 2 elementos, das colunas da matriz "B" no arranjo de processadores "pro".

Pode-se optar por não distribuir uma das dimensões da matriz, utilizando "*". Nesse caso todos os elementos nessa dimensão estarão localizados no mesmo processador. Esse tipo de distribuição é útil quando não se deseja distribuir uma das dimensões da matriz, ou seja, deixar as linhas ou colunas no mesmo processador.

Por exemplo, a diretiva

!HPF$ DISTRIBUTE (CYCLIC,*) ONTO pro :: B

especifica uma distribuição cíclica das linhas e não distribui as colunas da matriz "B" no arranjo de processadores "pro".

O HPF suporta um modo de distribuição de dados bastante flexível, e os dados distribuídos podem ser operados de maneira normal. As opções de distribuições especificadas acima podem ser aplicadas para cada dimensão da matriz, de forma que uma matriz com várias dimensões pode ser distribuída de diversas formas conforme as combinações possíveis.

Para se obter desempenho é fundamental a escolha apropriada da forma de distribuição de dados, visando maximizar o trabalho local e minimizar a comunicação. Essa escolha depende do problema a ser resolvido e de como os dados serão utilizados nos cálculos.

6.3.2.3 Diretiva ALIGN

Determina a distribuição de uma matriz baseada na distribuição de outra matriz ou *template* (diretiva que especifica um arranjo abstrato de elementos sem ocupar espaço na memória), ou seja, fazendo um alinhamento entre essas matrizes. Apenas uma matriz deve ser distribuída e as outras alinhadas a essa.

O programador pode efetuar o mesmo tipo de distribuição para as matrizes, de forma que o alinhamento fique implícito. Entretanto essa diretiva garante que os elementos desejados continuem alinhados, mesmo que a forma de distribuição seja alterada posteriormente.

Por exemplo, a diretiva

!HPF$ ALIGN C WITH B

especifica o alinhamento das matrizes "C" e "B", de modo que a matriz "C" possua a mesma distribuição que foi determinada para a matriz "B".

O uso de ":" implica em conformidade, ou seja, que as matrizes alinhadas possuam o mesmo número de dimensões e elementos em cada uma dessas dimensões.

Por exemplo, a diretiva

!HPF$ ALIGN C(:,:) WITH B(:,:)

especifica o alinhamento das matrizes "C" e "B", sendo que ambas possuem o mesmo tamanho.

O tipo de alinhamento abaixo

!HPF$ ALIGN D(i,j) WITH B(i,j)

implica que os elementos D(i,j) e B(i,j) estejam no mesmo processador, porém não implica em tamanhos iguais. Entretanto B precisa ter tamanho igual ou superior a D.

É possível combinar essas duas formas de alinhamento e realizar operações com i e j obtendo diversas formas de alinhamento. Também pode-se utilizar "*", para replicar ou colapsar uma das dimensões da matriz.

Matrizes alocadas dinamicamente também podem ser alinhadas, sendo que o alinhamento ocorre após a alocação da matriz.

6.3.2.4 Diretiva INHERIT

Propriedades da distribuição de dados dos argumentos de procedimento são herdados da chamada do procedimento. A diretiva DISTRIBUTE especifica a distribuição dos dados no programa principal e quando esses dados são passados como argumentos a rotinas ou funções, a diretiva INHERIT faz com que os parâmetros recebidos mantenham a distribuição original.

6.3.2.5 Diretivas REDISTRIBUTE e REALIGN

São diretivas dinâmicas que possuem o atributo DYNAMIC. São as formas dinâmicas das diretivas ALIGN e DISTRIBUTE, que são estáticas e definidas em tempo de compilação. A redistribuição e o realinhamento pode ocorre a qualquer instante durante a execução do programa. Porém isso implica em custo no desempenho, pois pode ser necessária a comunicação, entre processadores, para o remapeamento de uma matriz.

6.3.2.6 Diretiva INDEPENDENT

Usada para informar ao compilador que o laço DO ou FORALL, realiza operações que são independentes e portanto podem ser realizadas em paralelo.

O laço tradicional DO executa comandos em uma ordem sequencial determinada. O ponto de sincronização ocorre no ponto onde o programa espera o término de um comando antes de passar ao próximo.

A diretiva INDEPENDENT, quando aplicada ao laço DO indica que a ordem da iterações não afeta o resultado, ou seja, que uma iteração não interfere em outra.

Admite a cláusula NEW que permite replicar variáveis utilizadas nas várias iterações. Restringe o escopo das variáveis utilizadas, ou seja, cria variáveis temporárias que são privadas a cada iteração do laço.

Permite que o compilador gere um laço paralelo em situações nas quais o paralelismo não é detectado pelo compilador, sendo que estas iterações são divididas entre os processadores, criando um código mais eficiente. O programador precisa assegurar

que as instruções podem ser realizadas em paralelo e que, portanto, os dados obtidos nesses loops sejam iguais aos obtidos por uma execução sequencial. Se essa restrição for quebrada e uma iteração influenciar em outra tem-se a não conformidade com o HPF.

O FORALL é um comando explicitamente paralelo para fazer uma atribuição não sequencial em uma matriz ou vetor.

O FORALL éxecutado em quatro estágios:

- definição do conjunto de índices válidos;
- definição dos índices ativos, ou seja, aqueles que atenderam a uma máscara (opcional) especificada;
- cálculo do lado direito da expressão de atribuição;
- atribuição ao lado esquerdo.

Cada um dos quatro estágios apresentados precisa ser completado antes de passar para o próximo, definindo uma sincronização implícita entre eles, que pode reduzir o desempenho do código. Porém dentro de cada estágio as operações podem ser executadas nos elementos em qualquer ordem.

A diretiva INDEPENDENT, quando aplicada ao FORALL, quebra a sincronização dos quatro estágios.

6.4 Speedup e Lei de Amdahl

Define-se *speedup*:

$$S(p) = \frac{t_{\text{seq}}(p)}{t_{\text{par}}}, \tag{6.1}$$

na qual p = número de processadores, t_{seq} = tempo sequencial, referente à execução em uma máquina monoprocessada e t_{par} = tempo paralelo, referente à execução em uma máquina paralela.

Se para um dado valor de p, $S(p) = p$, então diz-se que se obteve um *speedup* linear.

Definição de eficiência:

$$E(p) = \frac{S(p)}{p}. \tag{6.2}$$

Assim, se eficiência for igual a 1, o *speedup* é linear.

O *Speedup* está limitado pela parte do código que não pode ser paralelizado. Considerando um código cuja fração r seja perfeitamente paralelizável, ou seja, o tempo de processamento dos comandos executáveis corresponde exatamente ao tempo de execução sequencial dividido pelo número de processadores p, o que obviamente despreza eventuais tempos de comunicação, pode-se expressar o *speedup* por

$$S(p) = \frac{t_{\text{seq}}}{(1-r)t_{\text{seq}} + \frac{rt_{\text{seq}}}{p}} = \frac{1}{(1-r) + \frac{r}{p}}.$$

No limite, para um número de processadores tendendo a infinito

$$p \to \infty \qquad S(p) \to \frac{1}{1-r}, \tag{6.3}$$

ou seja, o *speedup* é limitado superiormente por essa razão, a qual depende da fração de código não paralelizável. Esta afirmativa constitui a Lei de Amdahl.

As técnicas de processamento de alto desempenho foram utilizadas para a implementação do modelo formulado para o problema de solificação de ligas binárias. Foram realizadas várias simulações, as quais são descritas no próximo capítulo, e também são apresentados o ambiente utilizado, a abordagem adotada e os resultados obtidos.

CAPÍTULO 7

Simulações Efetuadas

Para a simulação da solidificação de uma liga binária foi utilizado o método apresentado na seção 5.5, utilizando-se técnicas de Processamento de Alto Desempenho.

O problema numérico foi resolvido através do método de Gauss-Seidel de linhas, devido à direção de propagação do problema, no qual sistemas tridiagonais são resolvidos em paralelo, ou seja, linhas inteiras de pontos são atualizadas simultaneamente.

Para uma simulação mais precisa é necessário uma malha de discretização bastante fina, que resulta em grandes matrizes e uma série de cálculos a serem efetuados nos elementos dessas matrizes. De forma que a utilização de mais de um processador possibilitaria a divisão de tarefas ou de dados, diminuindo o tempo de processamento necessário.

A programação do método proposto adota o paralelismo de dados e foi implementado em HPF, portanto os dados foram distribuídos através da utilização de diretivas apropriadas, sem que a comunicação por passagem de mensagem seja programada explicitamente pelo usuário.

7.1 Considerações sobre o ambiente utilizado

Utilizou-se o compilador Fortran 90 e HPF da Portland Group Inc. (PGI) para arquitetura IA-32 ("microcomputador") com sistema operacional Linux.

Segundo [16] a maioria dos compiladores HPF são pré-compiladores paralelizantes, os quais basicamente traduzem o código em Fortran 90 ou HPF para um equivalente na forma Fortran 77, com chamadas à biblioteca de comunicação.

O compilador utilizado é executado através de um *script*, o *pghpf*, o qual realiza dois estágios no processo de compilação: traduz do HPF para o Fortran 77 incorporando rotinas para passagem de mensagem, e também compila esse código intermediário, gerando o executável.

HPF funciona para máquinas de memória compartilhada e também para máquinas

de memória distribuída. Nesse último caso, o HPF utiliza-se de uma biblioteca de troca de mensagens, cujas rotinas tem funcionalidade similar àquelas do MPI (Message Passing Interface). A troca de mensagens é implementada utilizando-se os protocolos de uma rede local qualquer e a pilha de protocolos TCP/IP.

O compilador HPF da PGI permite gerar executáveis que incluem as rotinas de troca de mensagem necessárias, isto é, o executável pode ser portado para máquinas Linux que não tenham instalado o compilador HPF da PGI (nem tampouco a biblioteca de troca de mensagens associada).

Alguns testes foram realizados em uma máquina que dispõe de 4 processadores Pentium Pro 200 Mhz e 128 MB de memória principal. Seu sistema operacional é o Conectiva Linux versão 4.0, Kernel 2.2.5-23clsmp. Trata-se de uma máquina multiprocessada, nela utiliza-se o mesmo compilador HPF para gerar um executável a ser executado na própria máquina ou a ser portado para um conjunto de máquinas ligadas em rede.

No caso, as duas máquinas utilizadas do Setor de Lançamento de Balões, Divisão de Astrofísica, INPE (SLB/DAS) estão ligadas por uma rede local padrão Ethernet (10 Mbps). Trata-se de um Pentium II Celeron 300Mhz (CPU Intel), com sistema operacional Conectiva Linux 6.0, Kernel 2.2.17-14cl, e de um Pentium I MMX 200MHz com Conectiva Linux 5.0, Kernel 2.2.14-14cl, ambas com 64Mb de memória principal e 128Kb de memória cache. Essas duas máquinas compõem um multicomputador.

Além da máquina multiprocessada e do multicomputador do SLB/DAS alguns testes também foram realizados em um microcomputador Pentium III 450 MHz com 128 Mb de memoria principal, com o sistema operacional Conectiva Linux versao 6.0, kernel 2.2.17-14cl.

Os testes foram analisados com o uso de algumas ferramentas como o comando *time*, presente nos sistemas operacionais UNIX e Linux, e que fornece informações sobre a execução do programa e apresenta variações de acordo com o *shell* utilizado.

Algumas informações dadas, ao término de execução do programa a ser analisado, são o *user time*, *system time* e *elapsed time*. Basicamente, o *user time* refere-se ao tempo gasto com a execução de instruções e chamadas a rotinas de bibliotecas. O *system time* refere-se a todos os serviços que são dependentes do sistema operacional,

como entrada e saída de dados. Esses dois tempos resultam no *CPU time*. O *elapsed time* ou *real time* é a medida do tempo real que se passou desde o início até o término de execução. O *elapsed time* e o *CPU time* apresentam valores próximos, porém caso o programa analisado compartilhe a máquina com outros processos, ou realize uma grande quantidade de tarefas de leitura e gravação de dados, o *elapsed time* poderá ser bem maior.

Esses tempos são fornecidos pelo comando *time* no *Korn Shell* e no *Bourne Shell*. No *C Shell* são dadas mais algumas informações sobre a porcentagem de *CPU* utilizada, *page faults*, *swaps*, operações de entrada e saída e algumas medidas sobre a memória física que o programa ocupa durante a execução.

Outra ferramenta muito útil é o *pgprof profiler*, que analisa dados gerados durante a execução de programas compilados em *C*, *C++*, *F77*, *F90* e *HPF*, com opções especiais. Permite localizar quais as funções e/ou linhas que consomem mais tempo de processamento. Também apresenta informações dos processadores em programas *HPF*.

Ao se compilar o programa, com o compilador *PGI*, pode-se utilizar uma das duas opções fornecidas: uma fornece informações a nível de função e outra a nível de função e linhas de código. Durante a execução do programa é criado um arquivo *pgprof.out*, que poderá ser analisado como o uso do *pgprof profiler*, o qual tem saída gráfica e apresenta, em uma janela, as informações selecionadas dos menus disponíveis, e esses dados podem ser gravados em arquivos.

Detalhes sobre os testes e as condições nas quais eles foram realizados são dados a seguir.

7.2 Abordagem Utilizada

Para a simulação do sistema são fornecidas certos parâmetros referentes às propriedades da liga binária, ao seu diagrama de fases, à ampola e ao forno. E são dadas, também, as condições iniciais e condições de contorno do sistema.

As propriedades da liga como coeficiente de difusão, condutividade térmica, calor latente, calor específico e o digrama de fases são dadas por valores experimentais [27; 9].

A ampola possui cinco centímetros de comprimento e dois de diâmetro e considera-se a condutividade do seu material.

O forno apresenta um determinado gradiente térmico, dado pela diferença de temperatura em seus extremos e pelo comprimento de zona adiabática. Apresenta velocidade de puxamento definida, a qual controla a velocidade de crescimento da liga.

Como condições iniciais e de contorno, considera-se que a liga se encontra a uma concentração C_0, no início do processo de solidificação, e que nas paredes, em todo o tempo, a difusão de massa é nula, ou seja, não há perda nem fontes de massa e a temperatura depende da posição da ampola dentro do forno.

As condições iniciais de concentração e de temperatura podem ser vistas nas figuras 7.1 e 7.2.

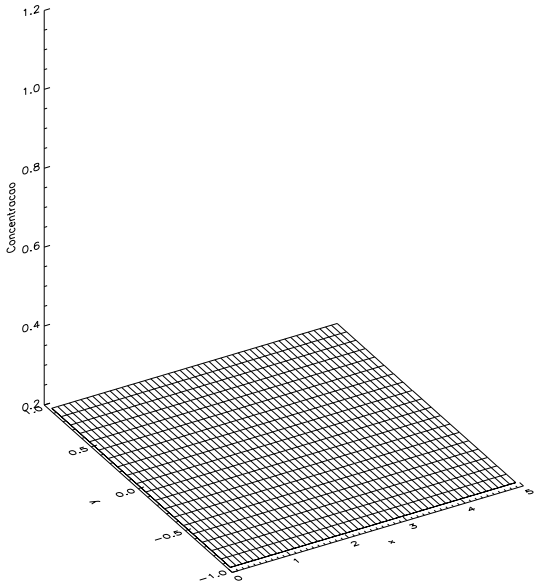

Fig. 7.1 – Campo de concentração inicial.

7.3 Resultados

Ao final da execução do programa, os resultados são gravados em 4 arquivos. Um deles contém um relatório de atividades do programa, tais como número de iterações realizadas, quantas vezes foi necessário reduzir o intervalo de tempo (Δt) utilizado e, em quais tempos de execução isso foi feito, e aparecem também mensagens de erro, caso o programa seja interrompido. Outro contém a fase (sólida, líquida ou *mushy*), que deverá ser sólida, para todos os pontos, ao final de uma solidificação bem sucedida. Outro arquivo contém o campo de concentração final do sólido obtido e outro contém todas as alterações, de fase, ocorridas para cada célula da malha de discretização, a cada intervalo de tempo adotado. Esse dados permitem que seja possível acompanhar a propagação da interface durante o processo de solidificação.

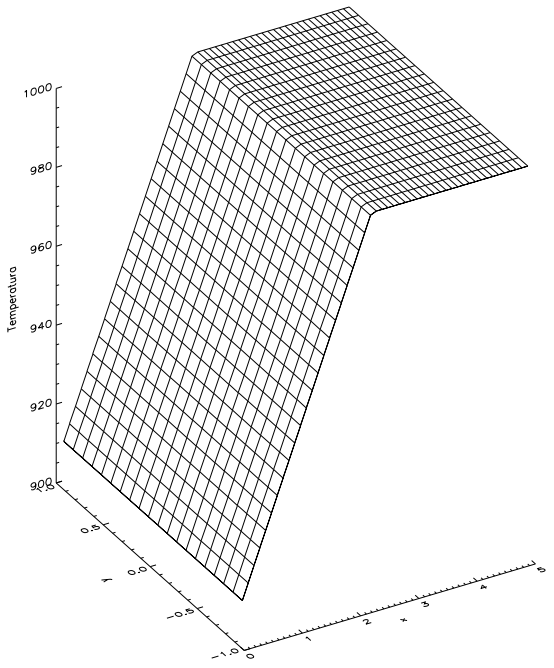

Fig. 7.2 – Campo de temperatura inicial.

Utilizando-se um algoritmo já implementado em linguagem C para a plataforma *Windows* [17], o qual permite realizar uma animação gráfica do avanço da interface, baseado nos dados obtidos pela simulação numérica, foi desenvolvido um programa escrito em *Interactive Data Language* (IDL) [36] para a plataforma *UNIX*.

A morfologia da interface sólido-líquido pode ser vista na figura 7.3, a qual ilustra a posição da interface num determinado instante de solidificação. Foram utilizados dados de uma malha de 60 por 300 elementos; malhas mais refinadas apresentam mais pontos e, consequentemente, uma resolução maior. Os campos de concentração e temperatura, referentes a essa posição da interface, são ilustrados nas figuras 7.4 e 7.5.

Também podem ser vistos os campos de concentração e de temperatura, nas figuras 7.6, 7.7, 7.8, 7.9, 7.10, 7.11 e 7.12, e os gráficos da concentração média radial, nas figuras 7.13 e 7.14, as quais ilustram um instante intermediário do processo de crescimento considerando-se duas velocidades de crescimento (10 mm/h e 1 mm/h).

Ao final da solidificação, o campo de concentração resultante pode ser visualizado nas figuras 7.15, 7.16, 7.17, 7.18 e a concentração média radial e o desvio médio quadrático na concentração média radial, nas figuras 7.19, 7.20, 7.21 e 7.22, respectivamente para as velocidades de crescimento de 10 mm/h e de 1 mm/h.

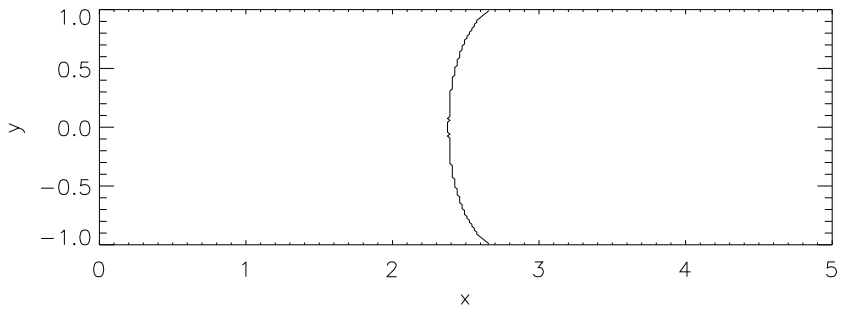

Fig. 7.3 – Posição da interface em um instante intermediário, $t = 10359.8s$, de uma solidificação realizada à velocidade de 10 mm/h.

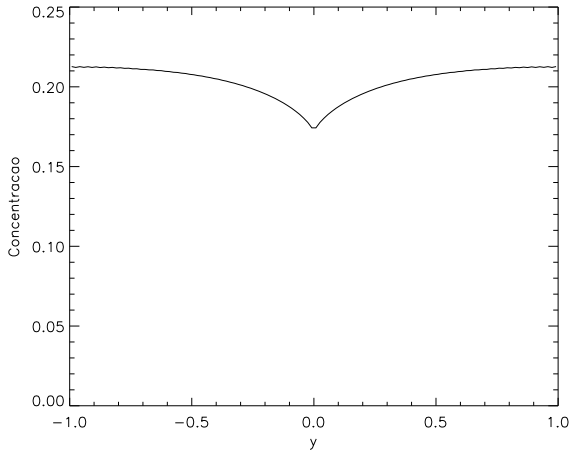

Fig. 7.4 – Concentração da interface em um instante intermediário, $t = 10359.8s$, da solidificação realizada à velocidade de 10 mm/h.

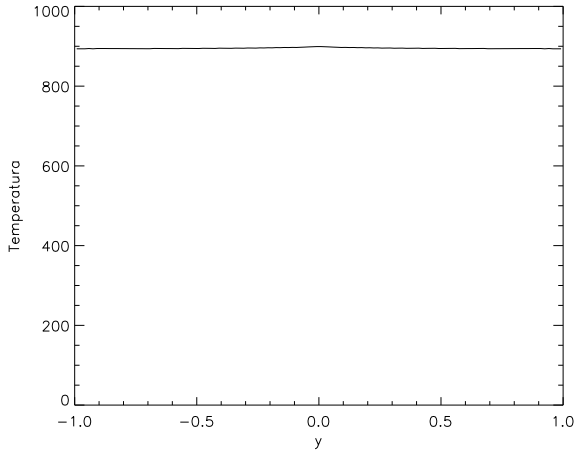

Fig. 7.5 – Temperatura da interface em um instante intermediário, $t = 10359.8s$, da solidificação realizada à velocidade de 10 mm/h.

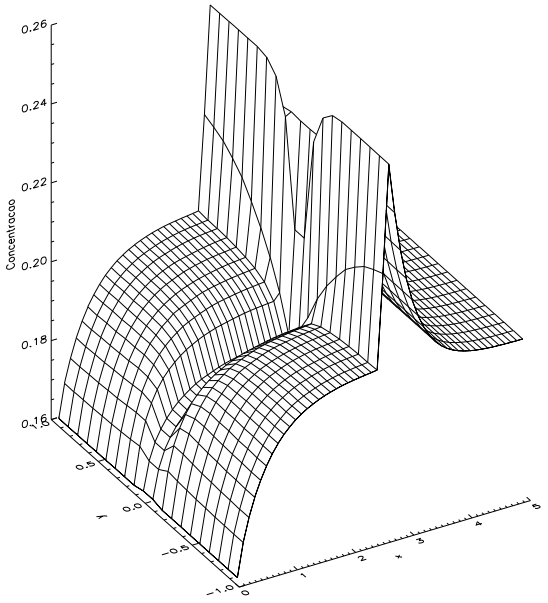

Fig. 7.6 – Campo de concentração em um instante intermediário da solidificação realizada à velocidade de 10 mm/h.

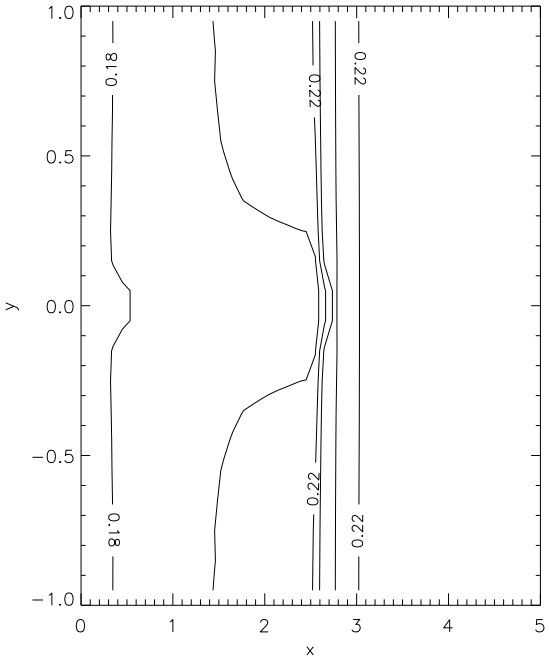

Fig. 7.7 – Linhas de isoconcentração em um instante intermediário da solidificação realizada à velocidade de 10 mm/h.

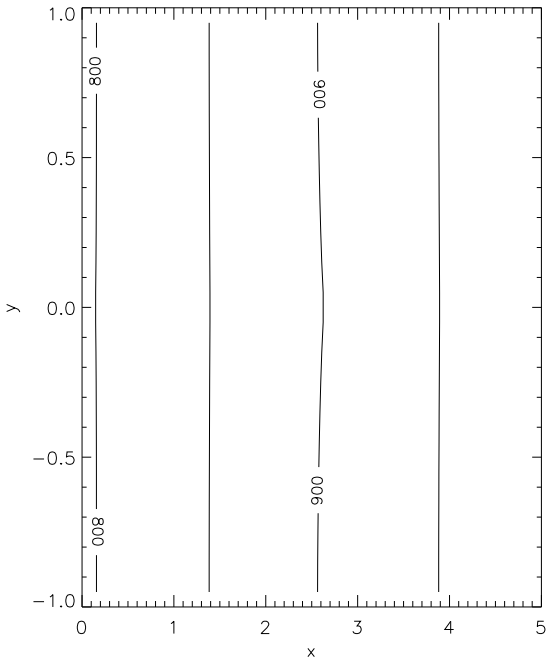

Fig. 7.8 – Isotermas em um instante intermediário da solidificação realizada à velocidade de 10 mm/h.

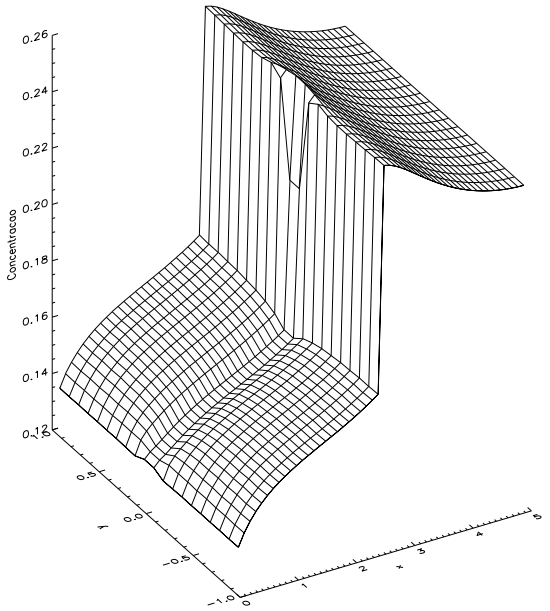

Fig. 7.9 – Campo de concentração em um instante intermediário da solidificação realizada à velocidade de 1 mm/h.

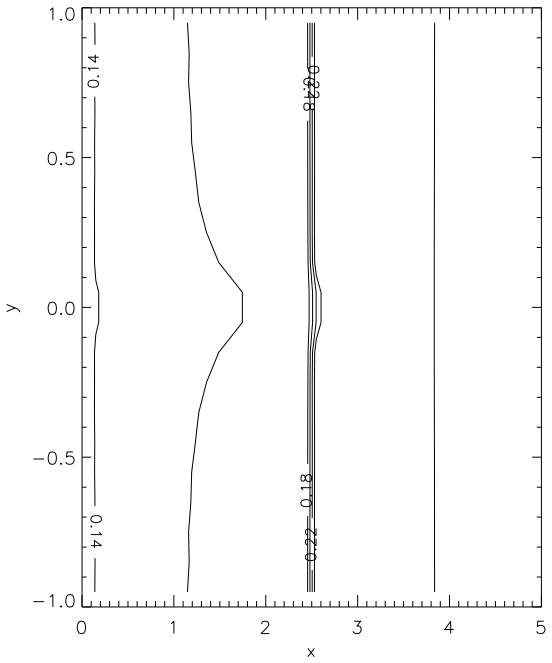

Fig. 7.10 – Linhas de isoconcentração em um instante intermediário da solidificação realizada à velocidade de 1 mm/h.

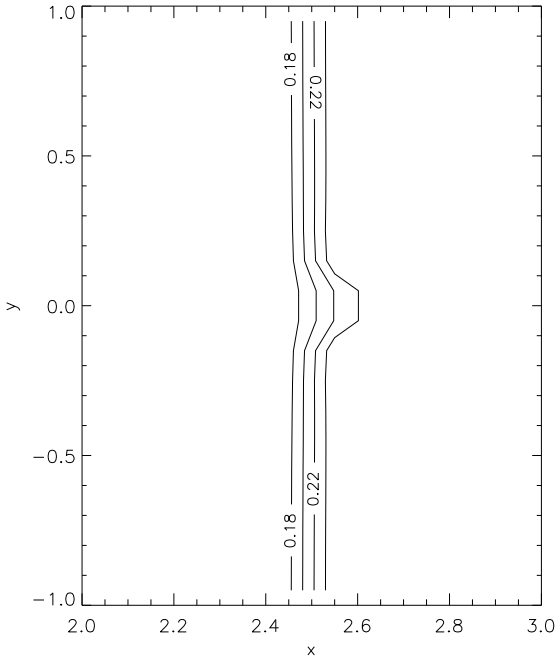

Fig. 7.11 – Linhas de isoconcentração em um instante intermediário da solidificação realizada à velocidade de 1 mm/h, na região da interface.

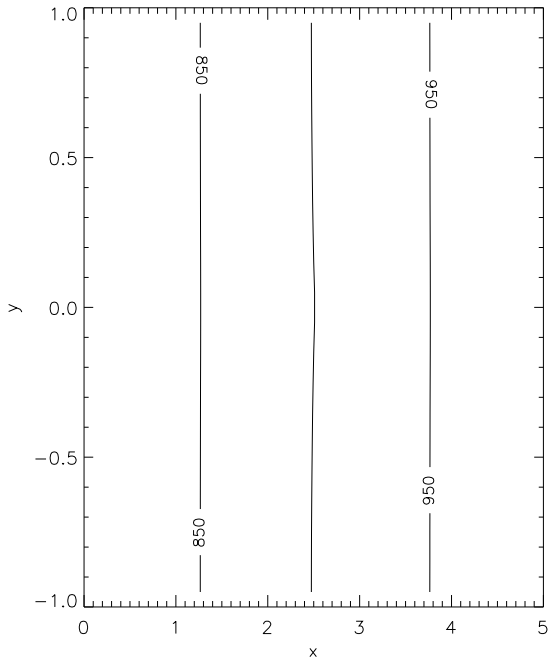

Fig. 7.12 – Isotermas em um instante intermediário da solidificação realizada à velocidade de 1 mm/h.

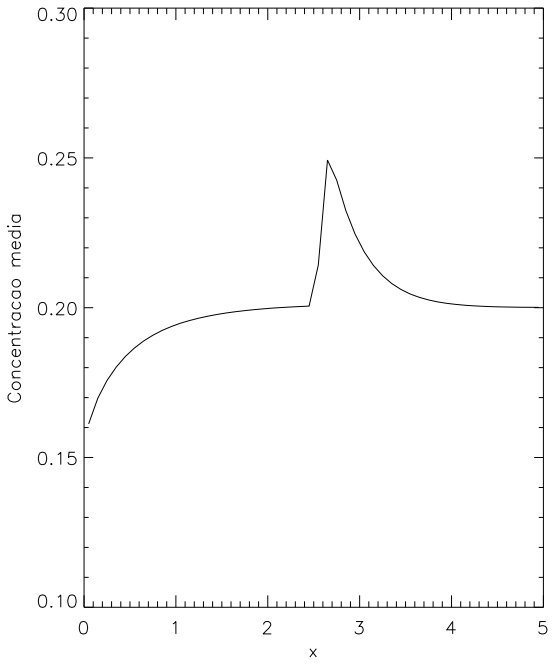

Fig. 7.13 – Concentração média radial para um instante intermediário da solidificação realizada à velocidade de 10 mm/h.

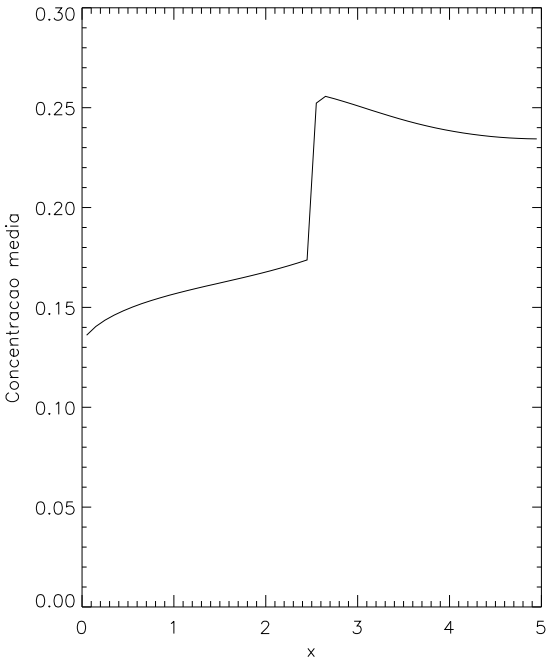

Fig. 7.14 – Concentração média radial para um instante intermediário da solidificação realizada à velocidade de 1 mm/h.

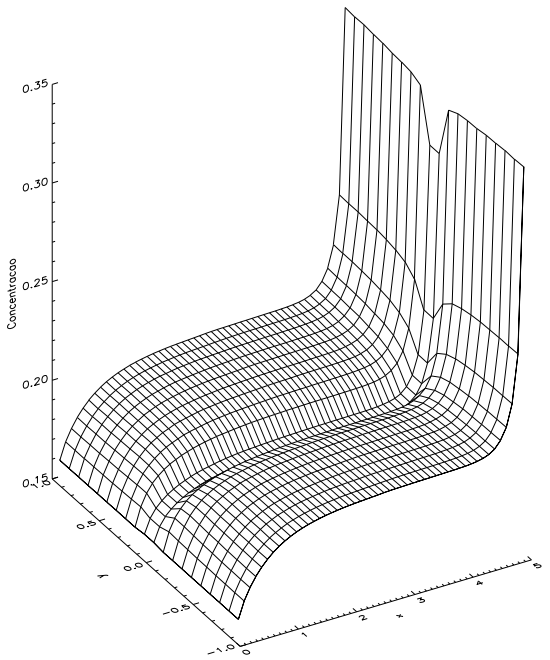

Fig. 7.15 – Campo de concentração final do sólido obtido através de uma solidificação realizada à velocidade de 10 mm/h.

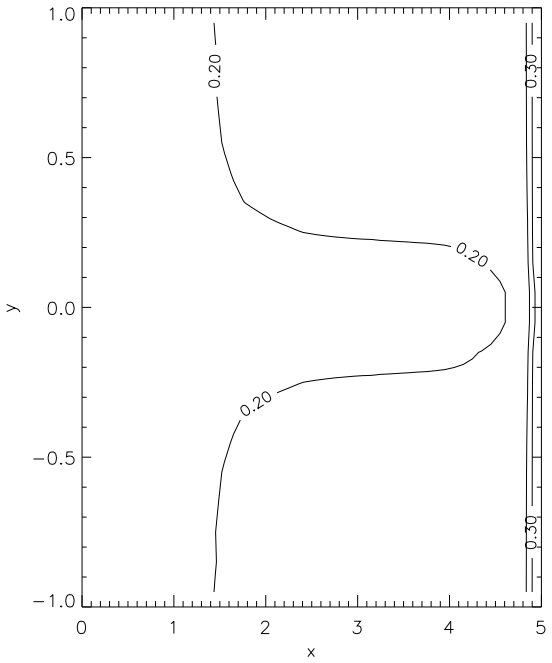

Fig. 7.16 – Linhas de isoconcentração do sólido obtido através de uma solidificação realizada à velocidade de 10 mm/h.

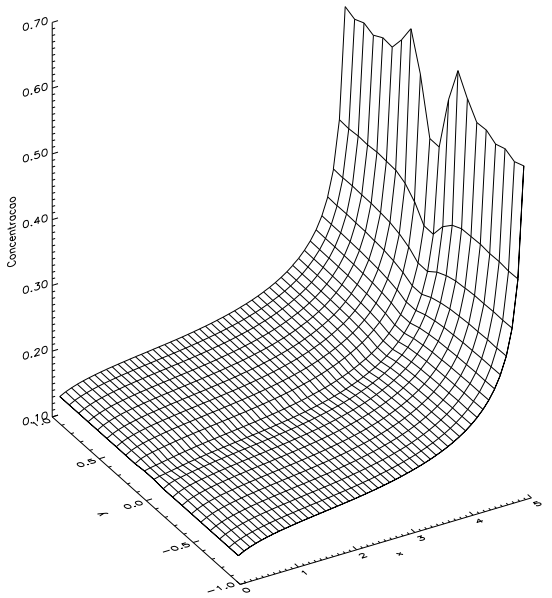

Fig. 7.17 – Campo de concentração final do sólido obtido através de uma solidificação realizada à velocidade de 1 mm/h.

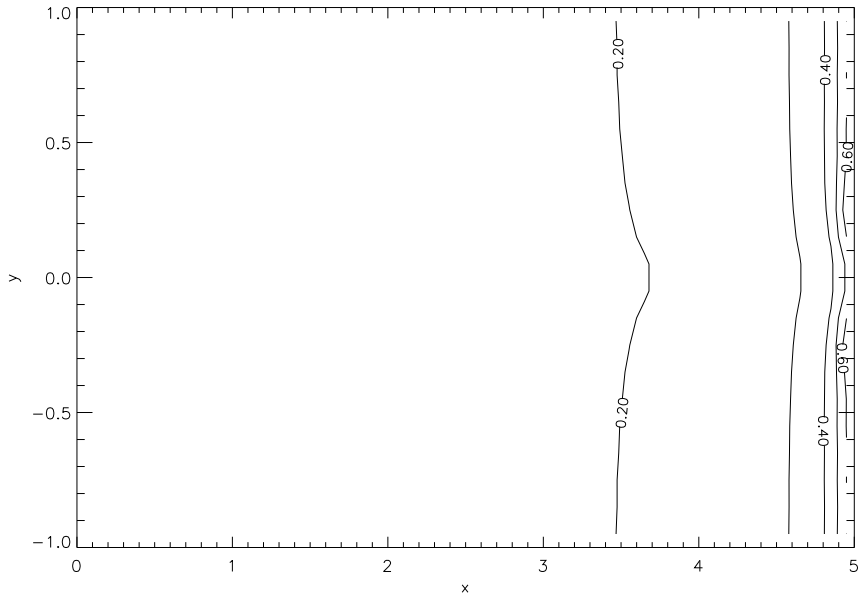

Fig. 7.18 – Linhas de isoconcentração do sólido obtido através de uma solidificação realizada à velocidade de 1 mm/h.

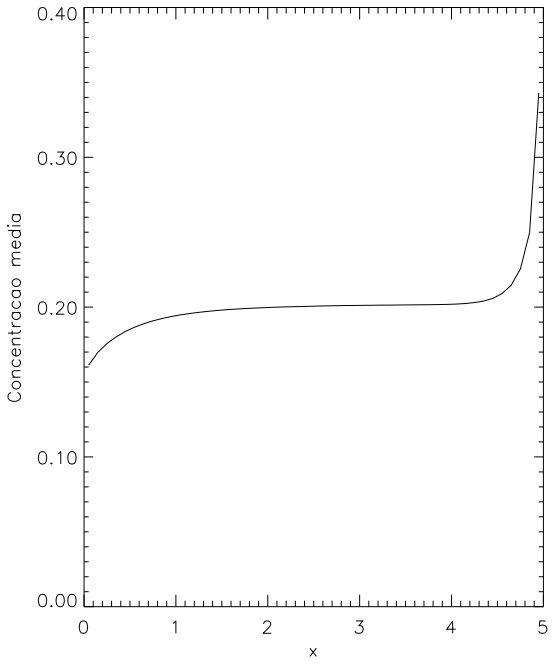

Fig. 7.19 – Concentração média radial do sólido obtido através de uma solidificação realizada à velocidade de 10 mm/h.

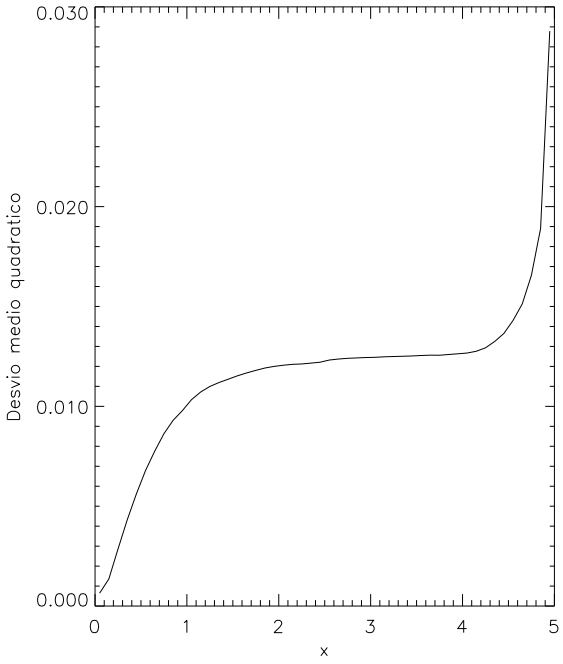

Fig. 7.20 – Desvio médio quadrático do campo de concentração em relação à concentração média radial, do sólido obtido através de uma solidificação realizada à velocidade de 10 mm/h.

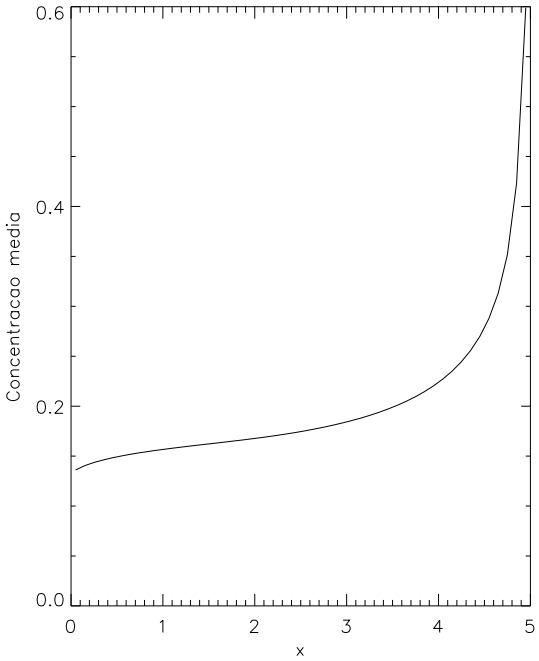

Fig. 7.21 – Concentração média radial do sólido obtido através de uma solidificação realizada à velocidade de 1 mm/h.

Os testes preliminares foram realizados com uma malha de discretização de 10 linhas por 50 colunas, uma malha pequena, para validação e testes básicos, na máquina multiprocessada primeiramente. Após isso foram feitos testes em outras máquinas com resoluções maiores.

Inicialmente o método numérico proposto foi implementado e compilado em *Fortran 90*, utilizando alocação dinâmica de memória, para que o programa pudesse ser executado para qualquer tamanho de malha, fornecendo os resultados esperados, em aproximadamente 11 minutos de processamento.

O mesmo código foi compilado em HPF, para se avaliar o que o compilador faria sem ter nenhuma indicação do programador. Também foram produzidos os resultados corretos levando praticamente o mesmo tempo, ou seja, o compilador não realizou nenhuma paralelização.

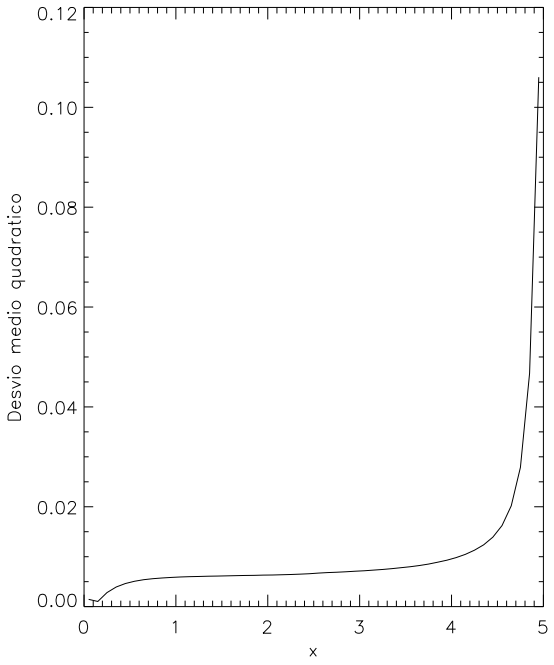

Fig. 7.22 – Desvio médio quadrático do campo de concentração em relação à concentração média radial, do sólido obtido através de uma solidificação realizada à velocidade de 1 mm/h.

O próximo passo tomado foi paralelizar o código explicitamente, iniciando-se com a inclusão da diretiva de paralelização INDEPENDENT em um simples laço de inicialização de dados, o que fez com que o programa fosse logo abortado sem produzir qualquer resultado. Essa diretiva foi retirada e substituída por uma diretiva de distribuição de dados DISTRIBUTE(BLOCK,*). O programa ficou em execução por mais de um dia sem produzir nenhuma saída, de modo que foi interrompido.

Diante desses problemas, procurou-se trabalhar na implementação do programa, gerando novas versões, as quais seriam novamente testadas. Para diminuir a complexidade do programa, a alocação dinâmica de memória, responsável pelas principais matrizes de cálculo foram substituídas por matrizes estáticas. Essa nova versão foi testada em Fortran 90 e em HPF, produzindo os resultados esperados levando praticamente os mesmos tempos entre si e se comparados com a versão anterior, a qual trabalhava com alocação dinâmica.

Novamente foi incluída a diretiva de paralelização INDEPENDENT e executou-se o programa sucessivamente com 1, 2, 3 e 4 processadores, obtendo-se resultados corretos, mas sem conseguir redução de tempo, o que era de se esperar visto que os laços eram executados poucas vezes e teriam pouca influência nos resultados. Esses laços foram paralelizados apenas para conferir o comportamento da execução do programa com a presença de diretivas *HPF*. Outro teste incluiu a utilização de diretivas de distribuição de dados, o qual foi executado, primeiramente, com apenas um processador e levou aproximadamente 20 minutos de processamento e, posteriormente, quando se utilizou 2 processadores demorou horas até produzir os primeiros resultados.

Após a realização desses testes, deduziu-se que o compilador não realiza paralelização de um código mais complexo, no qual o paralelismo não possa ser facilmente detectado. Optou-se por trabalhar com matrizes estáticas e verificar resultados de novos testes para que o processo de paralelização pudesse ser realizado passo-a-passo.

Além de se buscar a paralelização do código procurou-se, também, otimizá-lo para que o tempo gasto na execução do programa sequencial pudesse ser reduzido ao máximo, antes de realizar a paralelização de fato.

Um dos grandes fatores de influência do tempo levado pelo programa é o passo de tempo considerado. Devido às particularidades do problema físico, estudadas na

seção 5.5.3, foi visto que o passo de tempo deveria ser suficientemente pequeno para que as alterações provocadas nos campos de temperatura, concentração e energia se mantivessem consistentes e os resultados produzidos fossem coerentes. Porém um passo de tempo muito pequeno produziria os resultados corretos, mas de forma lenta uma vez que todos os cálculos seriam realizados inúmeras vezes a cada passo adotado e a progressão no tempo ocorreria de maneira mais demorada. Um passo de tempo maior, faria com que a solidificação, simulada, evoluísse de maneira mais rápida.

Devido ao algoritmo implementado, caso o tempo seja maior que do que o aceitável, a rotina faz a redução do intervalo de tempo e refaz os cálculos para confirmar se o novo intervalo satisfaz as exigências de convergência. Esses cálculos, testes e a recalculagem, quando necessária, torna o program mais lento. Assim, verificou-se que um passo de tempo médio deveria ser adotado, não muito pequeno que implicasse em muitas iterações, porém não grande tal que o intervalo tivesse que ser diminuído, várias vezes, para satisfazer o processo de solificação.

São gravados em um arquivo de dados o tempo no qual não foi possível atingir a precisão requerida e o número de vezes que foi necessário fazer a diminuição do valor do passo de tempo. Com base na análise desse arquivo e também considerando-se o tempo gasto na execução do programa pôde-se avaliar qual o Δt mais adequado para cada tamanho de malha testado. Isso foi analisado para programas em Fortran 90, sequenciais, executados no microcomputador Pentium III e os melhores valores de Δt foram destacados na tabela 7.1.

TABELA 7.1 – VALORES DE Δt ADEQUADOS A CADA TAMANHO DE MALHA E TEMPOS DE EXECUÇÃO DO PROGRAMA

tamanho da malha	Δt adequado	tempo consumido
10 x 50	40s	54,048s
20 x 100	10s	8m39,952s
30 x 150	4,5s	33m30,271s
40 x 200	2,6s	91m19,475s
50 x 250	1,7s	207m33,449s
60 x 300	1,1s	361m49,137s

O tempo consumido relatado é o *elapsed time*, que aproxima o tempo real de execução do programa, uma vez que os testes foram realizados individualmente

e sem que a máquina fosse utilizada por outros usuários. Esses testes não foram realizados para malhas maiores devido ao tempo tomado para cada teste, sendo que foram realizadas, no mínimo, 3 execuções do programa para cada tamanho de malha, considerando passos de tempo diferentes, para que fosse selecionado o melhor deles. Observa-se que, dentro da amostra dos testes realizados, o Δt adequado é diretamente proporcional ao volume de cada célula de discretização. Pode-se observar que ao dobrar o tamanho da malha em cada dimensão ou seja aumentar por um fator de 4 o volume da célula (dividir a resolução em X por 2 e em Y por 2), pode-se fazer o mesmo com o passo de tempo, o que se mostra perfeitamente coerente.

Utilizou-se esses valores de Δt obtidos para que os testes posteriores pudessem ser realizados e produzir resultados dentro de um tempo hábil.

Apesar de se obter tempos menores com valores de Δt adequados, notou-se que o tempo consumido ainda era grande e aumentava consideravelmente com o refinamento da malha. De maneira que foram feitas pequenas modificações no código do programa, visando obter tempo de processamento menor.

O código foi recompilado, na máquina multiprocessada, para permitir a geração do arquivo *pgprof.out* durante a execução. Os dados obtidos foram analisados com a ferramenta *pgprof* e comprovou-se que a função que realiza cálculo e atualização dos campos de temperatura, concentração e energia é a que mais consome tempo de processamento, pois o núcleo do programa se concentra nesse ponto, no qual o avanço do processo de solidificação ocorre devido às alterações desses campos. Utilizando o recurso de análise de linhas da função selecionada foi possível ver quais os pontos críticos dessa função.

Notou-se que a função intrínseca RESHAPE, do Fortran 90, a qual trabalha com matrizes, gasta tempo considerável de processamento. Esta foi eliminada graças à flexibilidade da linguagem para fazer acessos a seções de matrizes, obtendo-se ganho de tempo.

Dois blocos, um para o cálculo do campo de concentração e outro para o campo de temperatura, são os mais dispendiosos no programa e a principal modificação realizada foi a quebra de cada bloco em nove blocos separados. Cada bloco único original continha um laço bidimensional e efetuava operações condicionais de acordo com a vizinhança do ponto calculado, realizava verificação para comprovar se pontos

vizinhos estariam ou não na fronteira. Com a modificação , cada um dos nove blocos é específico para cada caso especial de fronteira.

Devido a essas alterações o programa ficou mais extenso, porém os últimos testes realizados, na máquina multiprocessada, comprovaram que se obteve um ganho notável, no tempo de execução, graças à eliminação de vários testes condicionais (IF).

Essas modificações no código foram realizadas para aumentar a eficiência do programa sequencial e também para tornar o algoritmo mais adequado Ă implementação e execução em um ambiente paralelo.

Os testes foram feitos na máquina multiprocessada e também no multicomputador do SLB/DAS. Inicialmente foi implementado em Fortran 90 e feitos testes dos programas sequenciais, os quais foram executados normalmente e forneceram os resultados corretos. O mesmo código foi compilado em HPF, sem a presença de diretivas. Foram obtidos os mesmos resultados dentro de um tempo de processamento praticamente igual, ou seja, o compilador não efetuou paralelizaçãodo código.

Posteriormente foram feitos testes com opções de auto-paralelização do compilador. Também não houve alteração no tempo de execução do programa. Com opções especiais de compilação podem ser verificados quais os trechos que são paralelizados automaticamente pelo compilador. Verificou-se que os trechos que são executados em paralelo, apresentam pouco destaque em relação ao tempo total de execução , pois ou não sao dispendiosos ou são realizados poucas vezes.

A inserção de diretivas HPF em laços simples como inicialização de algumas variáveis, alguns dos quais detectados pelas opções de auto-paralelização do compilador, faz com que sejam executados em paralelo. Embora facilmente paralelizáveis, esses tipos de laços, como explicado anteriormente, possuem pouca influência no tempo total de processamento.

Ao se tentar distribuir os dados principais, por exemplo, a temperatura, a concentração e a energia ou tentar paralelizar os laços que fazem a atualização desses campos, ou seja, os principais trechos de cálculo do programa, verificou-se que ocorrem problemas originados pela dependência de dados entre processadores, possivelmente devido à complexidade dos laços nos quais os cálculos são realizados.

A paralelização não foi efetuada mesmo com alterações no código, por exemplo, a evolução da interface passou a ser armazenada em variáveis e não em arquivos para permitir a paralelização, uma vez que a entrada e saída de dados é realizada de forma sequencial. Somente ao final de todos os cálculos os dados são passado das variáveis aos arquivos de resultados.

Com base nos testes efetuados, comprova-se que o programa sequencial funciona de modo correto, dentro de um tempo razoável, porém o algoritmo não é adequado para a execução em ambiente paralelo. Isso se deve ao problema físico em si, que é difícil de modelar visando um algoritmo conveniente para paralelização.

Um resumo das atividades realizadas, das principais considerações do problema e das conclusões obtidas é dado no próximo capítulo.

CAPÍTULO 8

Conclusões

O método numérico proposto, para a simulação bidimensional de crescimento de ligas binárias com malha fixa, foi implementado e, executado de forma sequencial, de maneira satisfatória.

Trata-se de um problema não-linear, ou seja, os campos de temperatura, concentração e energia são interdependentes. De forma que, por exemplo, para o cálculo da temperatura, em cada ponto da malha, faz-se necessário conhecer a concentração e a energia, que por sua vez, são calculadas com base no valor da temperatura no ponto e nas células vizinhas. E, cada campo é dependente de propriedades não-lineares, por exemplo, a concentração depende dos coeficientes de difusão, os quais são dependentes dos valores da concentração.

Como não seria possível realizar os cálculos, uma única vez, e fazer o acoplamento perfeito de todos os campos e todas as propriedades ou variáveis dentro de cada campo, os cálculos são realizados iterativamente. Cada campo é calculado de forma ímplicita em uma direção e explícita na outra, para que sistemas tridiagonais possam ser resolvidos. E cada campo utiliza valores, da iteração anterior, de suas propriedades. Esse cálculo é repetido até atingir uma precisão estipulada, para garantir que o campo esteja correto, ou seja, para corrigir os erros causados pela utilização de dados que foram gerados de forma explícita.

Após os cálculos de todos os campos, ainda é necessário realizar o acoplamento entre um campo e outro, ou seja, verificar, por exemplo, se o campo de energia obtido, quando utilizado para calcular o campo de temperatura, gera o campo de temperatura obtido na iteração corrente.

Além desses fatores, deve-se observar que, como se trata de um processo de mudança de fase, todas as equações devem ser calculadas para pontos que eventualmente passam de uma fase para outra. E cada ponto é dependente do processo que está ocorrendo nas quatro células vizinhas.

Para que os campos se mantenham consistentes, o processo iterativo é repetido até que se atinja a convergência estipulada, ou seja, os cálculos são realizados inúmeras vezes. À medida que a malha é refinada, esses cálculos aumentam muito, implicando

em um tempo de processamento crescente, de O(n^2).

O código foi otimizado a partir de informações de *profiling*, com o uso da ferramenta *pgprof*. Ainda assim, estima-se com base nos testes realizados que uma malha 200 x 1000, em uma máquina monoprocessada, demandaria vários dias de execução.

A partir do algoritmo otimizado foram feitas tentativas de paralelização. Esta pode ser restrita pelas características do algoritmo implementado, o qual apresenta laços que incluem testes de condições, inclusive relativas a pontos vizinhos (iterações próximas). Por sua vez, o algoritmo é restrito pela modelagem do problema físico em si.

No problema de mudança de fase abordado, a modelagem realizada, o algoritmo desenvolvido e restrições do High Performance Fortran (HPF), utilizado para a implementação, não possibilitaram que a simulação fosse realizada em um ambiente paralelo. Estas restrições são devidas ao caráter "automático" com o qual o HPF distribui os dados e aloca processadores considerando as diretivas de paralelização como sugestões, as quais podem ser seguidas ou não, ou seja, o programador não tem domínio completo da paralelização.

Para o caso estudado, a não linearidade do problema, associada ao acoplamento de vários campos interdependentes, faz com que o processo iterativo tenha que ser repetido até que se atinja a consistência. Os cálculos referentes a um determinado campo (temperatura, concentração ou energia) podem ser efetuados em paralelo, através do Método de Gauss-Seidel de linhas, porém são dependentes dos valores dos outros campos que já foram obtidos, na mesma iteração ou na iteração anterior. Há, portanto, no cálculo de um determinado campo, forte dependência dos valores calculados para os demais campos.

Assim, conclui-se que o problema de solidificação de ligas binárias possui características que dificultam a utilização de técnicas paralelas, mas pode ser modelado, implementado e executado em passos, os quais são independentes e podem ser realizados em paralelo.

Essas características restritivas poderiam ser contornadas por alterações na modelagem do problema, no desenvolvimento do algoritmo e/ou na utilização de outras linguagens e/ou bibliotecas para a paralelização , como o MPI e o PVM, que ficariam como sugestões de trabalhos futuros.

Pretende-se, fora do escopo desse trabalho, inserir no código Fortran 90 chamadas à biblioteca de comunicação por troca de mensagens MPI para tentar viabilizar a paralelização. À continuação, pretende-se também utilizar um multicomputador mais eficiente, com dez nós de 800 MHz, que estará disponível a curto prazo.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Alexiades, V.; Geist, G. A.; Solomon, A. D. *Numerical simulation of a HgCdTe solidification process.* Tennessee: NASA, 1985. 28p. (ORNL-6127/UC-32).

[2] Alexiades, V.; Solomon, A. D. *Mathematical modeling of melting and freezing processes.* Washington: Hemisphere Publishing Corporation, 1993. 323p.

[3] Alexiades, V.; Wilson, D. G.; Solomon, A. D. Macroscopic global modeling of binary alloy solidification process. *Quarterly of Applied Mathematics*, v.43, n.2, p.143-158, July 1985.

[4] Andreeta, J. P. *Diagramas de fase: princípios e aplicações ao crescimento de cristais.* In: I Escola de Verão em Crescimento de Cristais. São Paulo: Instituto de Pesquisas Energéticas e Nucleares (IPEN), 1997.

[5] Arai, N. N.; Fabbri, M.; Stephany S. Simulação numérica e visualização gráfica de crescimento de macrocristais. In: Seminários de Iniciação Científica INPE/PIBIC/CNPq, 2., 3., 4., 5., São José dos Campos, Maio 1996, Jun. 1997, Jul. 1998 e Jul. 1999. *Resumos.* São José dos Campos: INPE, 1996, 1997, 1998 e 1999.

[6] Bird, R. B.; Stewart, W. E.; Lightfoot, E. N. *Transport phenomena.* New York: John Wiley & Sons, 1960. 780p.

[7] Boston, B. K. *Front Tracking of Complex Wave Interactions.* New York. Thesis (Phd in Applied Mathematics and Statistics) - State University of New York, 1995.

[8] Brice, J. C. *Crystal growth processes.* New York: John Wiley and Sons, 1986. 298p.

[9] Calawa, A. R, et al. Crystal growth, annealing, and diffusion of lead-tin chalcogenides. *Transactions of the Metallurgical Society of AIME*, v.242, n.3, p.374-383, Mar. 1968.

[10] Campos Filho, M.P., Davies, G.J. *Solidificação e fundição de metais e suas ligas.* São Paulo: Editora da Universidade de São Paulo, 1978. 246p.

[11] Capper, P.; Gosney, J. J. G.; Jones, C. L.; Quelch M. J. T. Quenching studies in bridgman-grown CdHgTe. *Journal of Crystal Growth*, v.63, n.1, p.154-164, May 1983.

[12] Chalmers, B. *Principles of solidification.* New York: John Wiley & Sons, 1964. 319p.

[13] Crank, J. *Free and moving boundary problems.* Oxford: Clarendon, 1984. 425p.

[14] Digital Equipment Corporation. *Digital Fortran: language reference manual.* [online]. <http://bay.grove.ufl.edu/usr/share/doclib/f90/dflrm.htm>. Jan. 2001.

[15] Dutch, S. *Bravais lattices.* [online]. <http://gbms01.uwgb.edu/ dutchs/symmetry/bravais.htm>. Dez. 1998.

[16] Ewing, A. K. et al. *Writing data parallel programmes with High Performance Fortran.* Edinburgh: University of Edinburgh, 1995. 156p.

[17] Fabbri, M. An Alternative approach for the modelling and simulation of binary alloy solidification using a fixed-grid control-volume technique. In: Wrobel, L.C.; Brebbia, C.A. ed. *Computational Modelling of Free and Moving Boundary Problems.* Southampton: Computational Mechanics Publications. V.2, pp.59-76, 1991.

[18] Fazenda, A. L. *O método de campo de fase aplicado a problemas de solidificação.* São José dos Campos. Dissertação (Mestrado em Computação Aplicada) - Instituto Nacional de Pesquisas Espaciais, 1997.

[19] Foster, I. *Designing and building parallel programs: concepts and tools for parallel software engineering.* Massachusetts: Addison Wesley, 1995. 381p.

[20] Gilman, J. J. *The art and science of growing crystals.* New York: John Wiley & Sons, 1963. 493p.

[21] Hernandes, A. C. *Fenômenos de segregação em crescimento de cristais.* In: I Escola de Verão em Crescimento de Cristais, São Paulo: Instituto de Pesquisas Energéticas e Nucleares (IPEN), 1997.

[22] Hoffman, J. D. *Numerical methods for engineers and scientists.* Singapoure: McGraw-Hill, 1993. 825p.

[23] Horack, J. *Protein crystal growth.* [online]. <http://wwwssl.msfc.nasa.gov/msL1/pcg_why.htm>. Jan. 1997.

[24] Huang, Y.; Debram, W. J.; Fripp, A. L. Interface shapes during vertical bridgman growth of (Pb,Sn)Te crystal. *Journal of Crystal Growth*, v.104,

n.2, p.315-326, Jan. 1990.

[25] Jasinski, T.; Witt, A. F. On control of the crystal-melt interface shape during growth in a vertical bridgman configuration. *Journal of Crystal Growth*, v.71, n.2, p.295-304, Fev. 1985.

[26] Jasinski, T.; Witt, A. F.; Rohsenow, W. M. Heat transfer analysis of the Bridgman - Stockbarger configuration for crystal growth II. Analytical treatment of radial temperature variations. *Journal of Crystal Growth*, v.67, n.2, p.173-184, Abr. 1984.

[27] Kinoshita, K., Kiyomasa, S. PbTe-SnTe mutual diffusion coeficient at just above the $Pb_{0.8}Sn_{0.2}Te$ solidus temperature. *Journal of Crystal Growth*, v.67, n.2, p.375-379, Abr. 1984.

[28] Kittel, C. *Introduction to solid state physics.* New York: John Wiley & Sons, 1976. 648p.

[29] Marshall, A. C. *HPF programming course notes.* Liverpool: The University of Liverpool, 1997. 256p.

[30] Modi, J. J. *Parallel algorithms and matrix computation.* Oxford: Claredon, 1988. 260 p. (Applied Mathematics and Computing Science Series)

[31] Motakef, S. Interference of buoyancy-induced convection with segregation during directional solidification: scaling laws. *Journal of Crystal Growth*, v.102, n.1-2, p.197-213, Jan. 1990.

[32] Naumann, R. J.; Lehoczky, S. L. Effect of variable thermal conductivity on isotherms in bridgman growth. *Journal of Crystal Growth*, v.61, n.3, p.707-710, Jan. 1983.

[33] Pamplin, B. R. *Crystal growth.* Oxford: Pergamon, 1975. 672p.

[34] Patankar, S. V. *Numerical heat transfer and fluid flow* New York: McGraw Hill, 1980. 197p.

[35] Pfann, W. G. *Zone melting.* New York: John Wiley & Sons, 1958. 310p.

[36] Research System. *IDL Version 5.0.* USA, 1997.

[37] Rosenberger, F. E. *Fundamentals of crystal growth I.* Berlin: Springer-Verlag, 1979. 530p.

[38] Smith, G.D. *Numerical solution of partial differential equations: finite difference methods.* Oxford: Claredon, 1978. 304p.

[39] Solomon, A. D.; Alexiades, V.; Wilson, D. G. A numerical simulation of a binary alloy solidification process. *Society for Industrial and Applied Mathematics*, v.6, n.4, p.911-922, Oct 1985.

[40] Trava-Airoldi, V. J., Corat, E. J. *Diamond chemical vapour deposition.* In: I Escola de Verão em Crescimento de Cristais. São Paulo: Instituto de Pesquisas Energéticas e Nucleares (IPEN), 1997.

[41] Zeghbroeck, B. J. V. *Bravais lattices.* [online]. <http://ece-www.colorado.edu/~bart/book/bravais.htm>. Jan. 2001.