Universidade de São Paulo

Escola Politécnica

Departamento de Engenharia Metalúrgica e de Materiais

Modelamento por CVM do diagrama de fases do sistema Co–Cr–Al cúbico de corpo centrado

Luiz Tadeu Fernandes Eleno

Orientador Prof. Dr. Hélio Goldenstein

Co–orientador Dr. Cláudio Geraldo Schön

Índice geral

Modelamento por CVM do diagrama de fases do sistema Co–Cr–Al (Cobalto–Cromo–Alumínio) cúbico de corpo centrado

1. Introdução

1.1. Objetivos

A termodinâmica do sistema Co–Cr–Al cúbico de corpo centrado foi modelada por meio do CVM – método variacional de clusters, com base em dados experimentais tomados da literatura (tie–lines, temperaturas críticas de ordenamento e valores de atividade do alumínio). Como resultado, foram obtidas seções isotermas do diagrama de fases ternário, bem como os três binários do sistema, em condições de interesse tecnológico.

1.2. Revisão bibliográfica

Intermetálicos ordenados baseados no supereticulado B2 (tipo FeAl) têm atraído o interesse dos pesquisadores em engenharia e ciência dos materiais devido a um elevado potencial para o emprego como materiais estruturais em aplicações de alta temperatura (como por exemplo, palhetas de turbinas de aviões a jato ou de centrais termo–elétricas). Esses materiais aliam elevada resistência mecânica a altas temperaturas (~700°C), baixa densidade (~6g/cm^3) e elevada resistência à corrosão (1).

O sistema Co–Al apresenta uma fase estável tipo B2 nas composições vizinhas a 50%at de Al com ponto de fusão congruente de 1640°C a 50%at (2). Esse alto ponto de fusão torna materiais baseados nessa fase bons candidatos para aplicações estruturais. A adição de cromo a essas ligas apresenta alto potencial para o desenvolvimento desses materiais, pois ambos os sistemas binários, Al–Cr e Co–Cr, apresentam amplas áreas de estabilidade da fase cúbica de corpo centrado (CCC), o que permite supor que equilíbrios envolvendo o supereticulado B2 serão observados em grandes faixas de concentração no sistema ternário. O cromo também auxiliaria no incremento da resistência à oxidação dessas ligas.

Recentemente, Ishikawa, Ise et al. (3) determinaram experimentalmente quatro seções isotérmicas do sistema ternário, nas regiões de composição de interesse para o presente trabalho. Esses autores observaram que a fase B2 – (Co, Cr)Al apresenta–se em equilíbrio com a solução sólida CCC desordenada (A2) em uma ampla faixa de concentrações. O equilíbrio é de segunda ordem para teores baixos de cromo e torna–se de primeira ordem a partir de um ponto tricrítico,

cuja composição é função da temperatura. O domo de miscibilidade envolvendo as fases B2 e A2 fecha–se com o aumento da temperatura, mas seu desaparecimento total não foi observado na faixa de temperaturas estudada.

As transformações de ordem–desordem envolvendo o supereticulado B2 e a fase desordenada A2 podem ser modelados pelo método variacional de clusters (Cluster Variation Method, CVM), fornecendo um formalismo teórico para a descrição termodinâmica desses sistemas. O procedimento a ser adotado no presente trabalho foi usado recentemente para o modelamento do diagrama de fases Fe–Ti–Al (4) e será revisado mais adiante.

O objetivo do presente trabalho é modelar as transformações ordem–desordem envolvendo os supereticulados baseados no reticulado CCC por meio do CVM, utilizando–se os dados experimentais da ref. (3) para a determinação dos parâmetros de interação necessários ao cálculo.

1.3. Metodologia

A metodologia empregada no presente trabalho foi desenvolvida por C. G. Schön em sua tese de doutoramento na Universidade de Dortmund, Alemanha (5), e foi empregada no modelamento do sistema CCC Fe–Ti–Al (4). Nesse trabalho, informações sobre os equilíbrios envolvendo as fases A2, B2 e $D0_3$ (ordenada tipo Fe_3Al) foram empregadas na determinação dos parâmetros de interação dos sistemas binários Fe–Al e Ti–Fe CCC. Os parâmetros de interação do sistema Ti–Al CCC foram obtidos por comparação das temperaturas críticas de ordenamento no sistema ternário, na região rica em ferro. Uma fundamentação teórica mais aprofundada e mais abrangente sobre a ordenação atômica é fornecida por Inden e Pitsch (9)

No caso do sistema Co–Cr–Al, um procedimento alternativo foi empregado, visto que não há equilíbrios estáveis observados entre as fases A2, B2 e $D0_3$ nos binários Al–Cr, Co–Cr e Co–Al. Dados confiáveis de atividade do alumínio nos sistemas binários Al–Cr (6) e Co–Al (7), entretanto, encontram–se disponíveis na literatura e podem ser usados na determinação dos parâmetros de interação, conforme procedimento desenvolvido e exemplificado em (5) pelo modelamento do sistema Ti–Al–Cr CCC. Dados para o sistema Co–Cr foram obtidos por Havrankova et al. (8), através de espectrometria de massa em célula de Knudsen; esses dados serão utilizados para o modelamento desse sistema.

O presente trabalho propôs–se a combinar as informações termodinâmicas disponíveis nos sistemas binários e os dados experimentais obtidos no sistema ternário para a determinação dos parâmetros de interação para o modelamento do sistema Co–Cr–Al CCC por CVM.

1.4. Visão geral do presente trabalho

O objetivo principal desse trabalho de formatura, como o próprio título deixa claro, é o modelamento do sistema Co–Cr–Al CCC utilizando o método variacional de clusters – CVM, a partir de dados termodinâmicos obtidos da literatura. Portanto, o autor desse trabalho de formatura não realizou experimentos para a obtenção de dados termodinâmicos do sistema Co–Cr–Al. No entanto, esses dados estão totalmente baseados determinações experimentais obtidas na literatura. Os dados utilizados para o ajuste dos parâmetros do CVM são todos relativos aos três sub–sistemas binários. Isso significa que apenas parâmetros binários estão sendo utilizados. Isso nos leva a outro objetivo do presente trabalho: mostrar que com apenas os parâmetros dos sistemas binários, é possível fornecer uma boa descrição do sistema ternário.

Nessa primeira etapa do trabalho, portanto, faremos uma descrição do método variacional de clusters, seguida da sua aplicação para o sistema Co–Cr–Al CCC. Isso nos tomára a maior parte do trabalho. Entretanto, devido ao caráter fortemente computacional desse projeto, também descrevemos alguns dos conceitos práticos relacionados ao uso do CVM para cálculos em sistemas binários e ternários. Nessa segunda etapa, também descrevemos a utilização do software GnuPlot para a confecção de diagramas de fase ternários, já que acreditamos que esse esforço deve ser registrado de alguma forma no presente trabalho. Relacionado a isso, é ainda abordada uma forma de suavização de curvas, que foi necessária para a elaboração dos diagramas ternários, que mostravam uma certa dispersão dos pontos.

2. O CVM – Método variacional de clusters

2.1.Definições

O CVM foi proposto inicialmente como um formalismo geral para aproximar a entropia de sistemas cristalinos [5]. O princípio fundamental do método é definir um cluster básico, que inclua todas as interações entre átomos que devem ser levadas em consideração no cálculo. Uma descrição pormenorizada dos aspectos termodinâmicos relacionados à ordem configuracional está além do escopo do presente trabalho. O leitor pode encontrar tal descrição na referência [9]

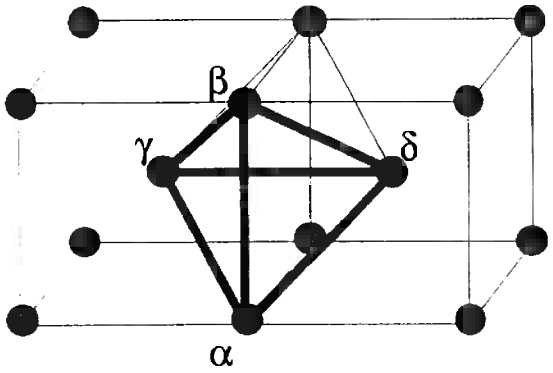


Figura 2.1. Tetraedro irregular na estrutura CCC. Estão representados dois tetraedros para enfatizar as intersecções entre os clusters.

No presente trabalho foi utilizado como cluster básico o tetraedro irregular, mostrado em linhas fortes na figura 2.1. O tetraedro particiona o reticulado CCC em quatro sub–reticulados, denotados pelas letras gregas α, β, γ *e* δ. O tetraedro irregular é o cluster tridimensional mais simples a levar em conta as interações entre pares de primeiros (α–γ, α–δ, β–γ e β–δ) e segundos (α–β e γ–δ) vizinhos [10], bem como interações tetraédricas (α–β–γ–δ) [5]. Os componentes do sistema – Co, Cr e Al – podem ocupar qualquer uma das posições do tetraedro. Uma configuração do tetraedro é representada por *{i, j, k, l}*, onde *i, j, k* e *l* indicam qual componente ocupa as posições α, β, γ e δ, respectivamente, com a seguinte relação: 1 = Co, 2 = Cr e 3 = Al.

Subclusters de um cluster são todos os conjuntos de pontos, pares e triângulos contidos no mesmo, incluindo o próprio tetraedro. As configurações de um subcluster são definidas de maneira semelhante às do cluster básico, indicada no parágrafo anterior. Agora, devemos definir as probabilidades de ocupação. Seja um subcluster λ = *{α, β, ...}* qualquer. A probabilidade de uma configuração *{i, j, ...}* para esse subcluster, indicada por $\rho^{[\lambda]}_{i,j,\ldots}$, é dada por [5]:

configuração *{i, j, ...}* para esse subcluster, indicada por $\rho^{\{\lambda\}}_{i,j,\ldots}$, é dada por [5]:

$$\rho^{\{\lambda\}}_{i,j,\ldots} = \frac{N^{\{\lambda\}}_{i,j,\ldots}}{q^{\{\lambda\}} N} , \tag{2.1}$$

onde:

- $q^{\{\lambda\}}$ é o número de coordenação do subcluster, isto é, o número de subclusters λ por posição no reticulado da estrutura cristalina. Os valores de $q^{\{\lambda\}}$ são fornecidos na tabela 2.1, para todos os subclusters do tetraedro irregular.
- $N^{\{\lambda\}}_{i,j,\ldots}$ é o número de subclusters λ com a dada configuração *{i, j,...}*;
- *N* é o número total de posições da estrutura cristalina.

Em particular, as probabilidades de ponto são dadas por $\rho^{\{\alpha\}}_{i}$. Indicaremos o cluster básico, no caso o tetraedro irregular, pela letra Λ.

As probabilidades dos sub–clusters estão relacionadas às probabilidades de tetraedro pelas chamadas "relações de redução". Por exemplo, para a configuração *{i,j}* do par *{α,β}*, essa relação é dada por:

$$\rho^{\{\alpha\beta\}}_{ij} = \sum_{k'l'} \rho^{\{\alpha\beta\gamma\delta\}}_{ijk'l'} \tag{2.2}$$

Tabela 2.1. Números de coordenação $q^{\{\lambda\}}$ e coeficientes de Kikuchi–Barker $a^{\{\lambda\}}$ para os subclusters do tetraedro irregular no reticulado CCC. Tetr Irr = tetraedro irregular, Tr iso = triângulo isósceles, 2viz = par de segundos vizinho e 1viz = par de primeiros vizinhos.

subcluster λ	configurações	$q^{\{\lambda\}}$	$a^{\{\lambda\}}$
Tetr Irr	{αβγδ}	6	1
Tr Iso	{αβγ}, {αγδ}, {αβγ}, {βγδ}	3	−1
2viz	{αβ}, {γδ}	2	1
1viz	{αγ}, {αδ}, {βγ}, {βδ}	¾	1
Ponto	{α}, {β}, {γ}, {δ}	¼	−1

Usando o tetraedro irregular no sistema CCC, é possível definir algumas estruturas nesse reticulado, com base nas probabilidades definidas acima. Essas estruturas são chamadas e supereticulados. Os supereticulados, juntamente com a sua notação Strukturbericht, estão definidos na tabela 2.2.

Tabela 2.2: Supereticulados no reticulado CCC utilizando o tetraedro irregular como cluster básico, com relação às probabilidades de ponto.

Estado fundamental	Probabilidades de ponto
A2	$\rho_i^{(\alpha)}=\rho_i^{(\beta)}=\rho_i^{(\gamma)}=\rho_i^{(\delta)}$
B2	$\rho_i^{(\alpha)}=\rho_i^{(\beta)}\neq\rho_i^{(\gamma)}=\rho_i^{(\delta)}$
B32	$\rho_i^{(\alpha)}=\rho_i^{(\gamma)}\neq\rho_i^{(\beta)}=\rho_i^{(\delta)}$
$D0_3$	$\rho_i^{(\alpha)}\neq\rho_i^{(\beta)}\neq\rho_i^{(\gamma)}=\rho_i^{(\delta)}$

2.2. Formulação

2.2.1. Energia interna

Definido o cluster e suas probabilidades, devemos escrever a energia livre do sistema em função dessas probabilidades. A energia interna do sistema é dada por [5]:

$$U^{\{\Lambda\}} = q^{\{\Lambda\}} N \sum_{i,j,k,l} \varepsilon_{i,j,k,l}^{\{\Lambda\}} \rho_{i,j,k,l}^{\{\Lambda\}} \tag{2.3}$$

onde os termos $\varepsilon_{i,j,k,l}^{\{\Lambda\}}$ são os parâmetros de interação do tetraedro irregular, dados por:

$$\varepsilon_{i,j,k,l}^{\{\Lambda\}} = +\frac{1}{6}\left(\varepsilon_{i,k}^{(1)}+\varepsilon_{i,l}^{(1)}+\varepsilon_{j,k}^{(1)}+\varepsilon_{j,l}^{(1)}\right)+\frac{1}{4}\left(\varepsilon_{i,j}^{(2)}+\varepsilon_{k,l}^{(2)}\right)+\frac{1}{2}\left(\tilde{\varepsilon}_{i,k,j}+\tilde{\varepsilon}_{i,l,j}+\tilde{\varepsilon}_{k,i,l}+\tilde{\varepsilon}_{k,j,l}\right)+\tilde{\varepsilon}_{i,j,k,l} \tag{2.4}$$

Nessa expressão, as interações entre pares são indicadas pelos sobrescritos (1) para primeiros vizinhos e (2) para segundos vizinhos. Os outros parâmetros são correções devidas às interações entre triângulos e entre conjuntos de quatro átomos, indicadas por um til (~). As frações são necessárias porque os pares de primeiros vizinhos são compartilhados entre 6 tetraedros, os pares de segundos vizinhos são compartilhados entre 4 tetraedros e, finalmente, os triângulos são compartilhados entre 2 tetraedros.

Tomando como estado de referência a mistura mecânica dos componentes, os parâmetros $\varepsilon_{i,j,k,l}^{\{\lambda\}}$ são substituídos por:

$$
\begin{aligned}
w_{i,j,k,l} &= \tilde{\varepsilon}_{i,j,k,l} - \frac{1}{4} \sum_{m=i,j,k,l} \tilde{\varepsilon}_{m,m,m,m} \\
w_{i,j,k} &= \tilde{\varepsilon}_{i,j,k} - \frac{1}{3} \sum_{n=i,j,k} \tilde{\varepsilon}_{n,n,n} \\
w_{i,j}^{(d)} &= \tilde{\varepsilon}_{i,j}^{(d)} - \frac{1}{2}\tilde{\varepsilon}_{i,i}^{(d)} - \frac{1}{2}\tilde{\varepsilon}_{j,j}^{(d)}
\end{aligned}
\qquad (5)
$$

Com essas relações, as energias de formação dos diferentes supereticulados definidos na tabela 2 podem ser escritas em função dos parâmetros de interação de maneira trivial. Entretanto, no reticulado CCC, apenas quatro parâmetros de interação são necessários; os demais são escritos como combinações lineares dos primeiros. Por razões históricas, prefere–se trabalhar com as contribuições dos pares, completando–se o sistema de equações com duas interações relativas a tetraedros. A escolha feita na referência [5], mantida no presente trabalho, está indicada pelo sistema 2.6:

$$
\begin{pmatrix} U^f_{DO_3\,(A,B)} \\ U^f_{B2} \\ U^f_{B32} \\ U^f_{DO_3\,(AB_3)} \end{pmatrix}_{T=0K} = 6N \begin{pmatrix} 0 & 0 & 1/4 & 1/3 \\ 0 & 0 & 0 & 2/3 \\ 1 & 0 & 1/2 & 1/3 \\ 0 & 1 & 1/4 & 1/3 \end{pmatrix} \times \begin{pmatrix} w_{ABAB} & w_{ABBB} & w_{AB}^{(2)} & w_{AB}^{(1)} \end{pmatrix} \qquad (2.6)
$$

O sistema de equações 2.6 fornece 12 parâmetros de interação em um sistema ternário A–B–C (quatro para cada ternário). Supereticulados ternários também são possíveis, fornecendo mais seis parâmetros. Neste trabalho, entretanto, assumiu–se que os termos de correção destes sub–reticulados são negligíveis.

2.2.2. Entropia

A entropia do sistema é dada por [5]:

$$
S = -q^{\{\lambda\}} N k_B \sum_{i,j,k,l} \rho^{\{\lambda\}}_{i,j,k,l} \ln \rho^{\{\lambda\}}_{i,j,k,l} - N k_B \sum_{\nu \subset \lambda} q^{\nu} a^{\nu} \sum_{conf} \rho^{\nu} \ln \rho^{\nu} \qquad (2.7)
$$

onde $k_B = 8{,}31451\ \mathrm{Jmol^{-1}K^{-1}}$ é a constante de Boltzmann. A primeira somatória leva em conta todas as configurações do tetraedro, enquanto a segunda diz respeito a todos os subclusters. Os a^{ν} são os coeficientes de Kikuchi–Barker, definidos na referência [5] e indicados na tabela 2.1. Desenvolvendo a equação 2.7 com os valores da tabela 2.1, chegamos à expressão para a entropia

do sistema baseada no tetraedro irregular. A expressão obtida é bastante complicada, e será omitida aqui por restrições de espaço. Entretanto, ela se encontra na referência [9], com uma notação um pouco diferente.

2.3. Minimização da energia interna

A energia livre do sistema é dada pela expressão convencional $G = U + PV - TS$, onde G é a energia livre de Gibbs, P é a pressão, V o volume e T a temperatura, sendo U e S a energia interna e a entropia, já definidas anteriormente. Em sistemas sólidos, no entanto, é comum negligenciarmos o termo PV, já que a variação de volume é desprezível face às outras variáveis [5]. Desta forma, podemos utilizar a expressão $A = U - TS$, onde A indica agora a energia livre de Helmholtz.

A energia livre do sistema é função do número de átomos de cada espécie – logo, é função da composição – e da temperatura. Entretanto, a dependência com a composição não é adequada ao cálculo de diagramas de fases, já que usualmente existe mais de uma fase em equilíbrio no sistema, cada uma com a sua composição e com a sua energia livre. Em um equilíbrio heterogêneo, o que se mantém constante em todas as fases é o potencial químico μ_i de cada espécie. Por exemplo, se existem duas fases em equilíbrio termodinâmico, ϕ e ξ *em um sistema binário* $A-B$, devemos ter:

$$\begin{aligned} \mu_A^{\phi} &= \mu_A^{\xi} \\ &e \\ \mu_B^{\phi} &= \mu_B^{\xi} \end{aligned} \qquad (2.8)$$

Portanto, é desejável trabalhar com a energia livre em função de μ_i, ao invés de x_i. Para essa conversão, procede-se com a transformada de Legendre da energia livre [5]. Esse procedimento, porém, requer o conhecimento da dependência da composição com o potencial químico. A formulação do CVM não fornece essa relação de forma analítica. Devemos nesse caso adotar um procedimento alternativo. Fazemos isso definindo a função F tal que:

$$F = A - N\sum_{i=1}^{n} \mu_i x_i \qquad (2.9)$$

onde n é o número de componentes do sistema. Devemos observar que os potenciais químicos para uma dada composição não são todos independentes, já que estão relacionados com a tangente à curva da energia livre em função da composição, sendo que essa tangente é única [5]. Além disso, o CVM, na formulação utilizada no presente trabalho, não modela lacunas [10]. Desse modo,

podemos definir as variáveis μ_i^* , dadas por [5]:

$$\mu_i^* = \mu_i - \frac{1}{n}\sum_{i'=1}^{n} \mu_{i'} \tag{2.10}$$

tais que $\sum_{i=1}^{n} \mu_i = 0$. Com essa nova variável, a função F é reescrita na forma:

$$F = A - N\sum_{i=1}^{n} \mu_i^* x_i + const. \tag{2.11}$$

Portanto, minimizar a energia livre em função da composição equivale a minimizar F em função dos potenciais μ_i^* . Um método numérico para esse procedimento é o método de iteração natural (*Natural Iteration Method – NIM*), desenvolvido por Kikuchi [10], no qual a função F é minimizada com relação às probabilidades do tetraedro por um algoritmo iterativo autoconsistente [10]. O NIM está descrito nas referências [5] e [10].

3. O sistema Co–Cr CCC

3.1 Ajuste dos parâmetros de interação

3.1.1. Introdução

Para o ajuste dos parâmetros a serem utilizados no programa CVM, foram obtidos dados de energia livre de excesso (ref. 8). O método utilizado para a obtenção desses dados experimentais foi a espectrometria de massa em célula de Knudsen. Antes de partir para o ajuste dos dados, vamos inicialmente fornecer uma visão geral sobre a técnica experimental e uma descrição um pouco mais detalhada sobre o modo de tratamento matemático dos dados termodinâmicos obtidos pela referência citada.

Em primeiro lugar, abordamos as funções de termodinâmicas de excesso e suas relações com as demais funções termodinâmicas. O próximo passo é descrever o tratamento matemático desses dados experimentais, chegando a expressões para as funções de excesso, obtidas por uma regressão pelo método dos mínimos quadrados.

De posse de expressões para as funções de excesso, a descrição dos dados experimentais estará completa. A próxima etapa é ajustar esses dados para uso no CVM. Isso é feito através de uma aproximação da solução real por uma solução regular, obtendo assim os valores iniciais dos parâmetros de interação. Esses parâmetros são posteriormente ajustados, por um método de tentativa e erro, até obtermos os parâmetros de interação definitivos do sistema Co–Cr CCC para uso nesse trabalho.

3.1.2. Funções de excesso

A energia livre de Gibbs molar de mistura para uma solução sólida real é dada por:

$$\Delta G_m^M = \sum_i x_i G_m^i \qquad (3.1)$$

onde x_i é a fração molar do componente i e G_m^i é a energia livre molar parcial do componente i. O sub–índice m indica que as grandezas são molares, isto é, dadas por mol de solução, e o sobre–índice M indica que se trata de uma mistura, ou, no caso particular do sistema Co–Cr, de uma solução sólida. Nesse caso, G_m^i corresponde ao potencial químico μ_i do componente i, de forma

que podemos reescrever:

$$\Delta G_m^M = \sum_i x_i \mu_i \qquad (3.2)$$

É comum referir–se não à energia livre de mistura, e sim à energia livre de excesso, que é dada por:

$$G_m^E = \Delta G_m^M - \Delta G_m^{id} \qquad (3.3)$$

A introdução dessa grandeza é importante, pois é facilmente obtida experimentalmente (ver, por exemplo, [11]). Dessa forma, a energia livre molar de excesso G_m^E é a diferença entre a energia livre da solução real e a energia livre de uma solução ideal, ΔG_m^{id} .

Recordamos que uma solução ideal é uma solução para a qual a entalpia molar de mistura ΔH_m^M é igual a zero, e os potenciais químicos são dados por:

$$\mu_i^{id} = \mu_i^0 + RT \ln x_i \qquad (3.4)$$

sendo R a constante universal dos gases e T a temperatura. O valor μ_i^0 é o *estado de referência* do componente *i*. Para um elemento puro, o valor de μ_i^0 é zero, por convenção. Portanto, para o sistema Co–Cr, devemos ter $\mu_{Co}^0 = 0 e \mu_{Cr}^0 = 0$.

Desse modo, para uma solução ideal, utilizando–se as equações (3.2) e (3.4), chegamos a uma expressão para a energia livre molar ideal:

$$\Delta G_m^{id} = RT \sum_i x_i \ln x_i \qquad (3.5)$$

Para uma solução real, o potencial químico é dado por:

$$\mu_i = \mu_i^0 + RT \ln a_i \qquad (3.6)$$

Nessa expressão, introduzimos a *atividade* do elemento *i*, a_i. Novamente, para um elemento puro, $\mu_i^0 = 0$.

É possível obter o valor da atividade em função do potencial químico de excesso; para tanto, façamos a diferença entre as equações (3.6) e (3.4). Essa operação resulta em:

$$\mu_i - \mu_i^{id} = RT \ln a_i - RT \ln x_i \qquad (3.7)$$

Entretanto, podemos definir uma outra função de excesso, o *potencial químico de excesso,* μ_i^E , que é dado por:

$$\mu_i^E = \mu_i - \mu_i^{id} \qquad (3.8)$$

Voltando com essa nova variável na equação (3.7) e isolando a atividade a_i, obtemos:

$$a_i = x_i \exp\left(\frac{\mu_i^E}{RT}\right) \qquad (3.9)$$

Como veremos, os dados experimentais obtidos na referência 8 resultam em valores para os potenciais químicos de excesso μ_i^E . Dessa forma, é fundamental obtermos uma expressão para essa grandeza. Esses cálculos serão feitos mais adiante, através de um ajuste por mínimos quadrados.

3.1.3. Dados experimentais

Os dados da referência 8 foram obtidos por espectrometria de massa em célula de Knudsen. A espectrometria de massa para a determinação de funções molares de excesso é realizada através de medidas das intensidades iônicas de alguns isótopos característicos correspondentes aos

elementos da liga. Para uma descrição do método experimental, ver a referência 8.

Vamos nos restringir à liga que estamos usando, Co–Cr. As intensidades iônicas nesse caso são J_{Co} e J_{Cr}, e os isótopos rastreados foram o ^{58}Co e o ^{52}Cr. Essas intensidades iônicas são funcões da temperatura e da composição da liga. A relação entre as intensidades iônicas e a diferença entre os potenciais químicos de excesso na célula de Knudsen é (ref. 8):

$$RT\cdot\left[\ln\left(\frac{J_{Cr}}{J_{Co}}\right)-\ln\left(\frac{x_{Cr}}{x_{Co}}\right)\right]=\mu_{Cr}^{E}-\mu_{Co}^{E}+C_0^G(T) \qquad (3.10)$$

Nessa expressão, $C_0^G(T)$ é uma constante de calibração dependente da temperatura, levando em conta características dos isótopos e da espectrometria de massa. Os estados de referência, tanto para o cobalto quanto para o cromo são os elementos puros na mesma temperatura da liga (1673 K).

O segundo membro da equação 3.10 contém as grandezas que se deseja determinar indiretamente. Os dados experimentais são relativos às intensidades iônicas. Um modo mais conveniente de tratar os dados termodinâmicos é através de uma regressão dos dados de intensidade iônica. Uma equação adequada para isso é:

$$\ln\left(\frac{J_{Cr}}{J_{Co}}\right)=d^0(x_{Cr})+\frac{d^1(x_{Cr})}{T} \qquad (3.11)$$

As constantes $d^0(x_{Cr})$ e $d^1(x_{Cr})$ são os parâmetros de ajuste. Esses são fundamentalmente os dados experimentais obtidos na referência 8, reproduzidos na tabela 3.1, juntamente com as composições das diversas ligas utilizadas para a obtenção desses dados.

Tabela 3.1

Constantes $d^0(x_{Cr})$ e $d^1(x_{Cr})$ da equação 3.11, em função da fração de cromo x_{Cr}.

Fase CCC, T = 1673 K

No	x_{Cr}	$d^0(x_{Cr})$	$d^1(x_{Cr})$
1	0,6742	4,2857	–3263,9
2	0,6782	3,5966	–2656,5
3	0,6981	0,9569	–2822,9
4	0,7320	4,1723	–3016,3
5	0,7420	3,7708	–2492,0
6	0,7651	4,1912	–2956,1
7	0,7940	3,9016	–2621,9
8	0,7940	4,0067	–2317,3
9	0,8163	4,2225	–2649,0
10	0,8409	3,8306	–2138,1
11	0,8658	4,1318	–1971,0
12	0,9227	4,1924	–1827,2

3.1.4. Tratamento matemático dos dados experimentais

Com os dados da tabela 3.1 e com as equações (3.10) e (3.11), obtemos valores para a *diferença* entre os potenciais químicos de excesso. O que desejamos é uma equação para a diferença entre os potenciais químicos *reais* da solução. Isso é feito através das equações 3.4 e 3.7. Com essas duas equações, chegamos à expressão procurada, da seguinte forma: temos, com as equações 3.4 e 3.7:

$$\mu_{Co} = RT \ln x_{Co} + \mu_{Co}^{E}$$
$$\mu_{Cr} = RT \ln x_{Cr} + \mu_{Cr}^{E}$$

A diferença entre essas duas equações nos fornece a expressão para a diferença entre os potenciais químicos:

$$\mu_{Co}-\mu_{Cr}=RT\ln\left(\frac{x_{Co}}{x_{Cr}}\right)+\left(\mu_{Co}^{E}-\mu_{Cr}^{E}\right) \tag{3.12}$$

Entretanto, no capítulo 2, havíamos definido os potenciais $\{\mu^{*}\}$:

$$\mu_{A}^{*} = \mu_{A}-\frac{\mu_{A}+\mu_{B}}{2} = \frac{\mu_{A}-\mu_{B}}{2} \tag{3.13}$$

Dessa forma, já temos como relacionar os potenciais $\{\mu^{*}\}$, obtidos diretamente do CVM, com os dados experimentais, através das equações 3.10, 3.11 e 3.12. Entretanto, não temos um valor para a constante de calibração C_0^G . Para obter esse valor, precisamos de alguma forma obter uma expressão algébrica para a diferença entre os potenciais químicos de excesso. Essa expressão é obtida em função de alguns parâmetros a serem ajustados pelo método dos mínimos quadrados.

Um modo conveniente de obter a expressão para a diferença entre os potenciais químicos de excesso é usar séries de potências para o ajuste das funções termodinâmicas de excesso, de acordo com a expressão abaixo:

$$Z_m^E=x_{Co}\sum_{n=1}^{N} C_n^Z x_{Cr}^n \tag{3.14}$$

Nessa expressão, Z_m^E indica tanto a entalpia, a entropia ou a energia livre de Gibbs, todas molares (Z = G, H, S). N é o número de coeficientes C_n^Z da série de potências. De posse da expressão para a energia livre molar de excesso, G_m^E , obtemos as expressões procuradas para os potenciais químicos de excesso.

A expressão da energia livre molar de excesso, de acordo com a equação 3.14, é:

$$G_m^E=x_{Co}\sum_{n=1}^{N} C_n^G x_{Cr}^n \tag{3.15}$$

O potencial químico de excesso da espécie *i* é dado por:

$$\mu_i^E = \left(\frac{\partial G^E}{\partial n_i}\right)_{n_j \neq n_i} = \left(\frac{\partial (n_T G_m^E)}{\partial n_i}\right)_{n_j \neq n_i} \tag{3.16}$$

onde n_T é o número total de mols da liga e n_i é o número de mols do elemento *i*. Desenvolvendo essa expressão em função das frações molares, e já colocando em função dos elementos da liga (Co e Cr), chegamos às equações 3.17a e 3.17b. Para a obtenção dessas expressões, bem como das expressões (3.18a) e (3.18b), ver o apêndice A.

$$\mu_{Co}^E = G_m^E - x_{Cr}\frac{\partial G_m^E}{\partial x_{Cr}} \tag{3.17a}$$

e $$\mu_{Cr}^E = G_m^E + x_{Co}\frac{\partial G_m^E}{\partial x_{Cr}} \tag{3.17b}$$

Realizando esses cálculos, obtemos finalmente as expressões para os potenciais químicos de excesso:

$$\mu_{Cr}^E = x_{Co}^2 \sum_{n=1}^{N} n C_n^G x_{Cr}^{(n-1)} \tag{3.18a}$$

$$\mu_{Co}^E = \sum_{n=1}^{N} C_n^G x_{Cr}^n (1 - n + n x_{Cr}) \tag{3.18b}$$

Agora substituimos os valores dos potenciais químicos de excesso na expressão (3.10), juntamente com os parâmetros d^0 e d^1 definidos na expressão (3.11). O resultado é a expressão utilizada para a determinação dos coeficientes da série de potências para a energia livre molar de excesso, ou seja, os coeficientes C_n^G :

$$RT\left[d^0+\frac{d^1}{T}-\ln\left(\frac{x_{Cr}}{x_{Co}}\right)\right]=C_0^G+\sum_{n=1}^{N} C_n^G\cdot x_{Cr}^{n-1}\cdot\left[n-(1+n)x_{Cr}\right] \qquad (3.19)$$

Os coeficientes C_n^G são então ajustados através de um método de regressão adequado. O método escolhido na referência 8 foi um ajuste por mínimos quadrados com N =2. Os valores dos coeficientes obtidos estão listados na tabela 3.2. A concordância dessa regressão está indicada na figura 3.1.

Tabela 3.2

Parâmetros C_n^Z e a constante de calibração C_0^Z para o sistema Co–Cr CCC $\left(C_n^G=C_n^H-T\,C_n^S\right)$

n	C_n^H (J/mol)	C_n^S (J/mol)	C_n^G (J/mol)
0	−18520	−20,62	15979
1	19500	19,40	−12866
2	−29500	−32,0	24573

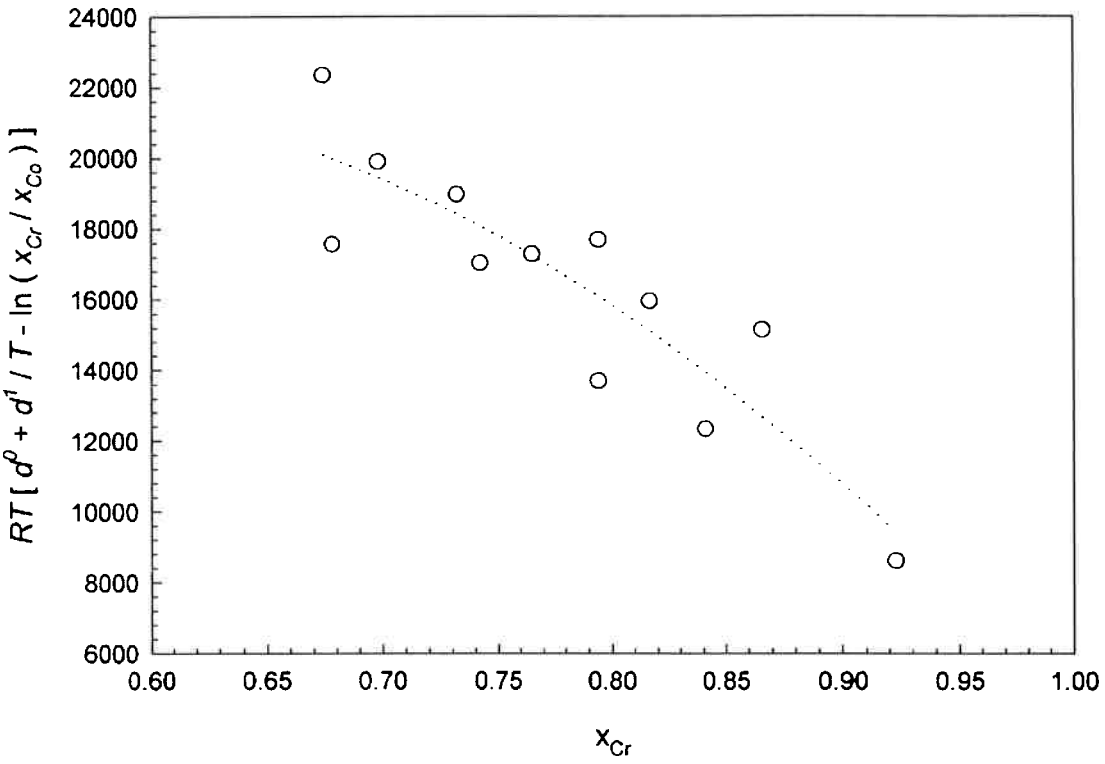


Figura 3.1 Ajuste dos parâmetros da tabela 3.2. Pontos: resultados experimentais [9]; linha: cálculo CVM do presente trabalho

Com o parâmetro C_0^G dado na tabela 3.2, podemos voltar à expressão 3.10 e calcular os valores de $\mu_{Cr}^E - \mu_{Co}^E$. A seguir, calculamos os valores de $\mu_{Cr} - \mu_{Co}$ através da expressão 3.12. E, finalmente, calculamos $\mu_{Co}^{\bullet} = -\frac{\mu_{Cr} - \mu_{Co}}{2}$, a partir da equação 3.13. O resultado desses cálculos está reproduzido na tabela 3.3.

Tabela 3.3

Potenciais químicos para o sistema Co–Cr CCC a 1673K

$(1\,k_B \cdot K = 8{,}31451\,J/mol)$

x_{Cr}	$\mu^E_{Cr} - \mu^E_{Co}$ (J/mol)	$\mu^*_{Co} = -\frac{\mu^E_{Cr} - \mu^E_{Co}}{2}$ (J/mol)	μ^*_{Co} $(k_B \cdot K)$
0,6742	6382,03	16498,11	–992,13
0,6782	1592,64	11962,84	–719,4
0,6981	3930,71	15591,14	–937,59
0,7320	3002,51	16979,37	–1021,07
0,7420	1059,16	15753,73	–947,36
0,7651	1317,85	17743,63	–1067,03
0,7940	–2274,58	16493,13	–991,83
0,7940	1719,98	20487,69	–1232,04
0,8163	–15,14	20731,58	–1246,71
0,8409	–3631,54	19528,06	–1174,34
0,8658	–825,89	25107,16	–1509,84
0,9227	–7346,07	27145,75	–1632,43

Nesse ponto, a descrição dos dados experimentais pertinentes ao sistema Co–Cr está completa. Todas as variáveis termodinâmicas necessárias ao uso do CVM estão calculadas ou estão expressas algebricamente. O próximo passo será o ajuste dos parâmetros de interação iniciais. Isso é feito na próxima seção.

3.1.5.Energias de formação

Todos os dados termodinâmicos disponíveis são relativos à funções de excesso, que foram tratados para obtermos os dados de potencial químico. A temperatura de 1673K, temperatura na qual os dados foram obtidos, é alta o suficiente para que possamos tratar a solução sólida como uma solução regular. Para uma solução regular, temos que a entropia tem o mesmo valor que para uma solução ideal, e portanto a entropia de excesso é nula. A entalpia é algum valor dependente das energias de formação entre pares. Desse modo, podemos escrever:

$$H^E_m = \left(\Delta U_m - \Delta U^{id}_m\right) = \Delta U_m \approx \Delta U^{reg}_m \qquad (3.20)$$

Nessa equação, já foi feita a aproximação $H_m \approx U_m$, sendo U a energia interna, já que o volume não é uma variável empregada no CVM. Além disso, relembramos que $\Delta U_m^{id}=0$.

No estado desordenado, as probabilidades de tetraedros podem ser reescritas na forma:

$$\rho_{i,j,k,l}^{\{\alpha,\beta,\gamma,\delta\}} \approx \rho_i^{\alpha}\rho_j^{\beta}\rho_k^{\gamma}\rho_l^{\delta}=x_i x_j x_k x_l \tag{3.21}$$

Essa aproximação leva a:

$$\Delta U_m^{reg}=\left(U_{AAAA}\right)x_A^4+\left(4U_{AAAB}\right)x_A^3 x_B+\left(4U_{ABAB}+2U_{AABB}\right)x_A^2 x_B^2+\left(4U_{ABBB}\right)x_A x_B^3+ \\ +\left(U_{BBBB}\right)x_B^4 \tag{3.22}$$

Os estados de referência são os elementos puros; portanto, U_{AAAA} e U_{BBBB} são nulos. A equação 22 pode dessa forma ser reescrita como:

$$\Delta U_m^{reg}=x_A x_B\left[4U_{AAAB}x_A^2+\left(4U_{ABAB}+2U_{AABB}\right)x_A x_B+4U_{ABBB}x_B^2\right] \tag{3.23}$$

Fazendo a substituição $x_B = 1-x_A$ e definindo $6U_{AB} = 4U_{ABAB} + 2U_{AABB}$, chegamos a:

$$\Delta U_m^{reg}=x_a\left(1-x_A\right)\left[\left(4U_{AAAB}-6U_{AB}+4U_{ABBB}\right)x_A^2+\left(6U_{AB}-8U_{ABBB}\right)x_A+\left(4U_{ABBB}\right)\right] \tag{3.24}$$

A equação 3.24 servirá ao ajuste em função dos dados experimentais apresentados no item anterior. Antes disso, é necessária apenas uma pequena transformação:

$$\frac{\Delta U_m^{reg}}{x_A(1-x_A)} = \left[\left(4U_{AAAB}-6U_{AB}+4U_{ABBB}\right)x_A^2+\left(6U_{AB}-8U_{ABBB}\right)x_A+\left(4U_{ABBB}\right)\right] \quad (3.24a)$$

De acordo com a equação 3.21, a entalpia molar de excesso é numericamente igual à energia interna de mistura. Mas a equação 3.12 nos fornece uma expressão para o cálculo da entalpia de excesso:

$$H_m^E = x_{Co}\sum_{n=1}^{N} C_n^H x_{Cr}^n = \Delta U_m^{reg} \quad (3.25)$$

No entanto, foram usados apenas dois termos na obtenção da série ($N = 2$). Desenvolvendo a expressão acima, temos:

$$\Delta U_m^{reg} = x_{Co}x_{Cr}\left(C_1^H + x_{Cr}^2 C_2^H\right) \quad (3.26)$$

Continuando o desenvolvimento da expressão acima em termos de x_{Co}, chegamos a:

$$\frac{\Delta U_m^{reg}}{x_{Co}x_{Cr}} = (-C_2^H)x_{Co} + (C_1^H + C_2^H) \quad (3.27)$$

Comparando as equações 3.24a e 3.27, estamos aptos a determinar as energias de formação. Essas energias serão utilizadas posteriormente para o cálculo dos parâmetros de interação iniciais a serem usados no CVM. Dessa forma, somos levados a:

$$\begin{aligned} 4U_{AAAB}-6U_{AB}+4U_{ABBB} &= 0 \\ 6U_{AB}-8U_{ABBB} &= -C_2^H \\ 4U_{ABBB} &= C_1^H + C_2^H \end{aligned} \quad (3.28)$$

Relembrando, os valores numéricos dos coeficientes C_1^H e C_2^H são, respectivamente, 19500 e –29500 J/mol, de acordo com a tabela 3.2. Resolvendo esse sistema, chegamos a:

$$\boxed{\begin{aligned} U_{AAAB} &= +4875\,J/mol = +586k_B.K \\ U_{AB} &= +1583\,J/mol = +190k_B.K \\ U_{ABBB} &= -2500\,J/mol = -298k_B.K \end{aligned}} \qquad (3.29)$$

De posse desses valores, determinaremos os parâmetros de interação iniciais, conforme explicitado na próxima seção.

3.1.6. Parâmetros de interação iniciais

A matriz relacionando os parâmetros do CVM com as energias de formação de um sistema bínário é dada por (*N* é o número total de átomos do sistema):

$$\begin{pmatrix} U^f_{DO_3(A_3B)} \\ U^f_{B2} \\ U^f_{B32} \\ U^f_{DO_3(AB_3)} \end{pmatrix}_{T=0K} = 6N \begin{pmatrix} 0 & 0 & 1/4 & 1/3 \\ 0 & 0 & 0 & 2/3 \\ 1 & 0 & 1/2 & 1/3 \\ 0 & 1 & 1/4 & 1/3 \end{pmatrix} \times \begin{pmatrix} w_{ABAB} & w_{ABBB} & w^{(2)}_{AB} & w^{(1)}_{AB} \end{pmatrix} \qquad (3.30)$$

A matriz do lado esquerdo da equação acima deve receber os valores das energias calculadas na seção anterior. Nessa equação, $U^f_{DO_3(A_3B)}$ corresponde a U_{AAAB}, $U^f_{DO_3(AB_3)}$ corresponde a U_{ABBB}. U^f_{B2} corresponde a U_{AABB} e U^f_{B32} corresponde a U_{ABAB}. Entretanto, naquela seção foi definida a variável $6U_{AB} = 4U_{ABAB} + 2U_{AABB}$. Dessa forma, não temos os valores individuais das energias de formação das fases B2 e B32, nesse primeiro momento. Note que, com essas relações, devemos desconsiderar *N* na expressão 3.30, já que as grandezas da seção anterior eram todas molares.

Os parâmetros a serem usados no CVM são as variáveis dadas na última matriz, w_{ABAB}, w_{ABBB}, $w^{(2)}_{AB}$ e $w^{(1)}_{AB}$. Nesse ponto, todos esses valores devem ser determinados. Porém, se analisarmos esse sistema, veremos que temos cinco equações (as quatro do sistema 3.30 mais a

equação $6U_{AB} = 4U_{ABAB} + 2U_{AABB}$, definida anterirmente), com oito incógnitas; portanto, o sistema não está determinado. O procedimento a ser adotado aqui é fixar em zero o valor de dois dos parâmetros de interação e calcular os demais. A escolha de quais parâmetros devem ser mantidos em zero baseia–se na experiência; entretanto, isso não fará grande diferença nesse estágio, já que os valores serão posteriormente ajustados aos dados experimentais.

A escolha feita foi manter os parâmetros w_{ABAB} e $w^{(2)}_{AB}$ em zero. Substituindo os valores das energias, já calculados na seção anterior, no sistema (3.30), mais a equação $6U_{AB} = 4U_{ABAB} + 2U_{AABB}$, temos:

$$\begin{aligned} 586\,k_B.K &= 2\,w^{(1)}_{AB} \\ U^f_{B2} &= 4\,w^{(1)}_{AB} \\ U^f_{B32} &= 2\,w^{(1)}_{AB} \\ -298\,k_B.K &= 6\,w_{ABBB} + 2\,w^{(1)}_{AB} \\ 190\,k_B.K &= 4\,U^f_{B32} + 2\,U^f_{B2} \end{aligned} \qquad (3.31)$$

Resolvendo o sistema 3.31, encontramos os parâmetros de interação desejados, juntamente com as energias de formação das fases B2 e B32. O resultado está esquematizado na tabela 3.5.

Tabela 3.5
Parâmetros de interação iniciais e energias de formação para o sistema Co–Cr CCC

$w_{CoCrCoCr}$	$w_{CoCrCrCr}$	$w^{(2)}_{Co,Cr}$	$w^{(1)}_{Co,Cr}$
0	−148 $[k_B.K]$	0	+293 $[k_B.K]$
$U^f_{B32(CoCr)_{T=0K}}$	$U^f_{DO_3(CoCr_3)_{T=0K}}$	$U^f_{DO_3(Co_3Cr)_{T=0K}}$	$U^f_{B2(CoCr)_{T=0K}}$
+586 $[k_B.K]$	−301$[k_B.K]$	+586 $[k_B.K]$	1173 $[k_B.K]$

3.1.7. Ajuste dos dados experimentais – parâmetros de interação definitivos para o sistema Co–Cr CCC

Nesse estágio, temos os primeiros dados de entrada para um cálculo usando o CVM, fornecidos na tabela 3.5. Para verificar a concordância dos cálculos que faremos a seguir com os dados experimentais, usaremos os valores obtidos para os potenciais químicos, apresentados na

tabela 3.4. O valor a ser ajustado é $\mu^*_{Co}=\frac{\mu_{Co}-\mu_{Cr}}{2}$, que já havia sido definido anteriormente. Os valores desse potencial estão na tabela 3.3. Essa expressão é uma variável do CVM, e portanto é facilmente ajustada a dados experimentais, caso seja possível obter dados para $\{\mu^*\}$ da literatura. No caso dos dados da referência 8, isso é plenamente possível, conforme já visto, e portanto vamos utilizá–los.

A figura 3.2 mostra os dados experimentais para $\{\mu^*\}$ fornecidos na tabela 3.4, juntamente com a curva obtida com o CVM utilizando os parâmetros de interação inicais fornecidos na tabela 3.5. O ajuste, como se percebe, não é satisfatório. Isso é esperado, pois esses parâmetros são apenas um "chute", uma vez que deliberadamente fixamos dois dos parâmetros em zero, sem nenhuma justificativa a não ser a experiência. O próximo passo agora é obter um ajuste melhor, através da variação dos parâmetros de interação. O método para isso é um processo de tentativa e erro, já que não há, até presente momento, nenhum processo mais elaborado de avaliação desses parâmetros. O ajuste definitivo também está traçado na figura 3.2, mostrando uma concordância bem mais aceitável. A tabela 3.6 fornece os valores dos parâmetros de interação utilizados para o ajuste definitivo.

As energias de formação mostradas na tabela 3.6 foram calculadas em função dos novos parâmetros de interação. O processo é exatamente o inverso do que foi feito na seção anterior: agora, entramos no sistema (3.30) com os parâmetros de interação e calculamos as energias de formação.

Tabela 3.6

Parâmetros de interação definitivos e energias de formação para o sistema Co–Cr CCC

$w_{CoCrCoCr}$	$w_{CoCrCrCr}$	$w^{(2)}_{Co,Cr}$	$w^{(1)}_{Co,Cr}$
0	+50 $[k_B.K]$	−380 $[k_B.K]$	+293 $[k_B.K]$
$U^f_{B32(Fe_3Cr)_{T\to 0K}}$	$U^f_{DO_3(FeCr_3)_{T\to 0K}}$	$U^f_{DO_3(Fe_3Cr)_{T\to 0K}}$	$U^f_{B2(Fe_3Cr)_{T\to 0K}}$
−554 $[k_B.K]$	+316$[k_B.K]$	+16 $[k_B.K]$	+1173 $[k_B.K]$

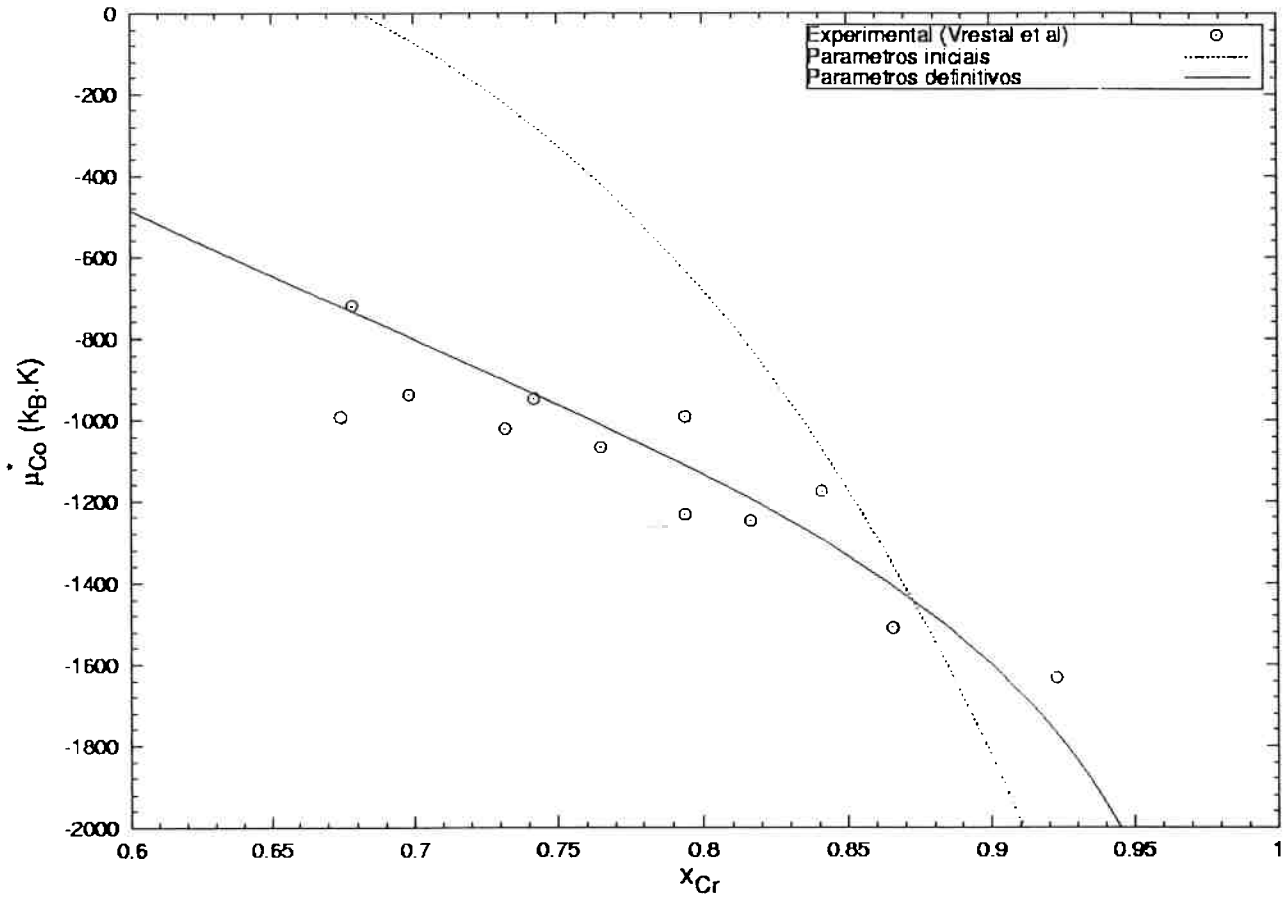


Figura 3.2 Ajuste dos parâmetros de interação do CVM. A linha cheia mostra o ajuste definitivo, enquanto a linha tracejada foi obtida com os valores iniciais obtidos para esses parâmetros. Os dados experimentais foram fornecidos na tabela 3.3.

3.2. Diagrama de fases

3.2.1. Introdução

O primeiro passo paro o cálculo do diagrama de fases é a análise de quais fases estarão presentes nesse diagrama. Isso é feito através de um gráfico chamado *Diagrama de estados fundamentais.* As fases mais estáveis fornecidas por esse gráfico serão as encontradas no diagrama de fases. Nesse tópico, forneceremos um maior detalhamento dos procedimentos para o cálculo de diagramas binários em geral, usando o exemplo do sistema Co–Cr. A teoria relacionada ao cálculo dos equilíbrios e aos diagramas de estados fundamentais é fornecida no capítulo 2.

3.2.2. Diagrama de estados fundamentais

Para o caso de um sistema binário, o diagrama de estados fundamentais é obtido plotando as energias de formação a 0 K contra a composição da liga. As energias de formação já foram calculadas e estão na tabela 3.1. A 0 K, a composição das fases B2 e B32 é de 50at–%, ou seja, essas fases apresentarão a sua composição estequiométrica típica. As composições das fases $D0_3$ serão de 25at% Cr para a fase Co_3Cr e 75at%Cr para a fase $CoCr_3$, pelo mesmo motivo. O diagrama está mostrado na figura 3.3.

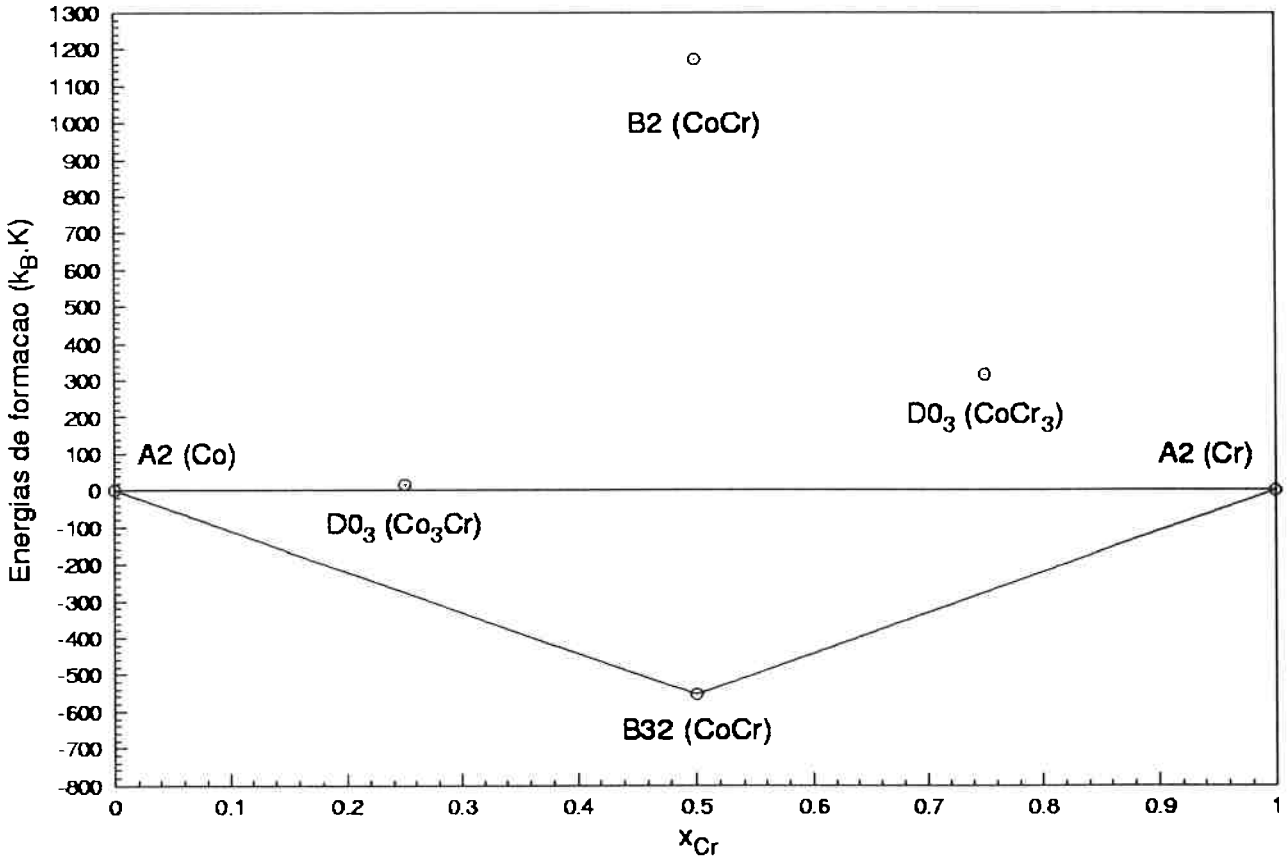


Figura_3.3 Diagrama de estados fundamentais para o sistema Co–Cr CCC usando os valores para as energias de formação a 0 K fornecidos na tabela 3.6.

Através do diagrama de estados fundamentais, percebemos que a fase mais estável é a fase B32, já que ela apresenta a menor energia de formação. Logo, o diagrama de fases deverá apresentar um equilíbrio entre as fases B32 e A2.

3.2.3. Diagrama de fases Co–Cr

O diagrama de fases calculado para o sistema Co–Cr–Al CCC está apresentado abaixo.

Nesse diagrama, podemos notar um estreito campo de estabilidade da fase B32. A transição A2/B32 é de 1ª ordem, e o campo de separação de fases é bastante amplo, apesar de ir diminuindo com o aumento da temperatura. O pico de temperatura esta a 942 K, com $x_{Co} = 0{,}53$. Uma forte assimetria pode ser verificada no sistema, mostrando que a fase desordenada A2 é mais estável a baixas temperaturas para composições mais ricas em cobalto. O mesmo acontece com o campo da fase B32, deslocado para maiores teores de cobalto.

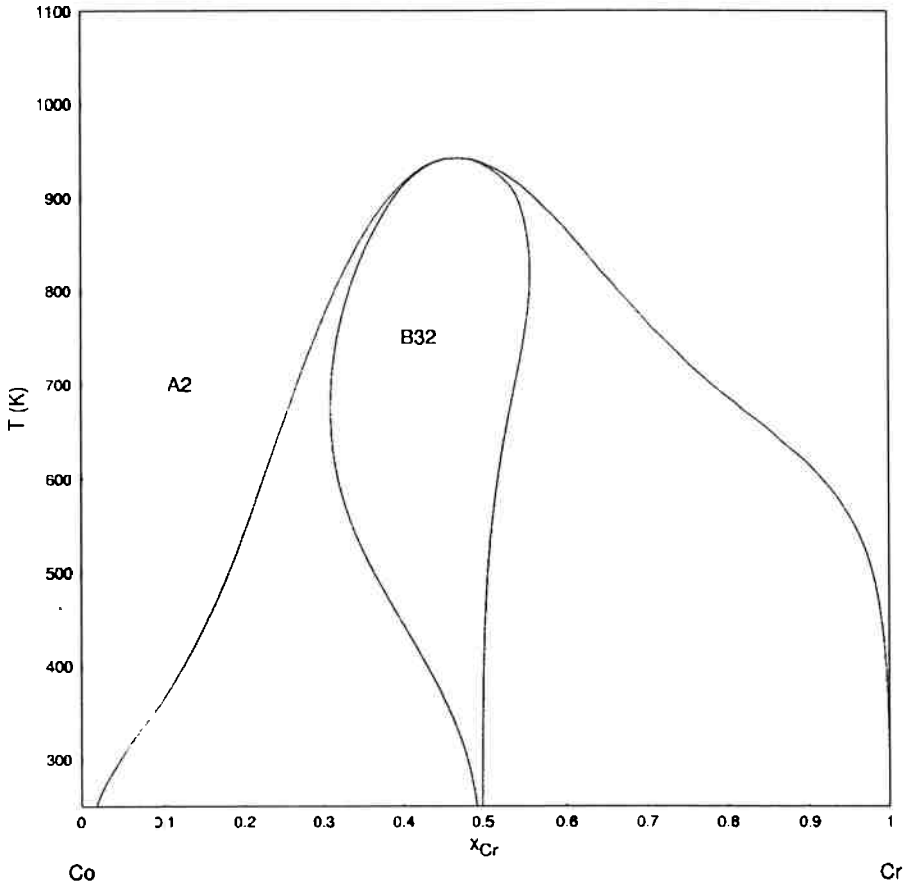


Figura 3.4 Diagrama de fases do sistema Co–Cr calculado com os parâmetros de interação da tabela 3.6

4. O sistema Co–Al CCC

4.1. Introdução

O sistema Co–Al CCC foi tratado recentemente por Colinet, Inden e Kikuchi [10], e foi modelado usando o CVM, como parte do sistema ternário Fe–Co–Al. Os autores apresentam uma discussão a respeito dos parâmetros de interação utilizados, e calculam os diagramas binários do sistema, juntamente com seis seções isotérmicas e três isopletas. Para o presente trabalho, os parâmetros de interação foram modificados. Nesse capítulo, temos por objetivo discutir os parâmetros de interação originais, bem como a alteração desses parâmetros. A seguir, calcularemos o diagrama de fases do sistema Co–Al CCC através do CVM.

4.2.Parâmetros de interação

Os parâmetros de interação $w^{(k)}_{CoAl}$ obtidos por Colinet et al. [10] não puderam ser determinados diretamente a partir das temperaturas críticas de ordenamento no sistema binário, como foi o caso do sistema Fe–Al, porque a estrutura CCC é estável apenas na forma da estrutura ordenada B2 com $x_{Al} \approx 0{,}5$. Por essa razão, o método de determinação dos parâmetros de interação foi baseado na comparação com o diagrama de fases experimental do sistema Co–Al. Esse diagrama [10] impõe um limite superior ao parâmetro $w^{(2)}_{CoAl}$, já que a temperatura crítica $T^{DO_3 \rightarrow B2}$ não deve entrar no campo de estabilidade da fase B2. Isso estabelece que $w^{(2)}_{CoAl} \geq -750 k_B K$, aproximadamente. Colinet et. al. utilizaram esse limite para o cálculo.

O valor do parâmetro $w^{(1)}_{CoAl}$ foi determinado a partir das temperaturas críticas de ordenamento $T^{B2 \rightarrow A2}$ observadas no diagrama de fases experimental nas composições próximas ao ternário Fe–Co (do sistema Fe–Co–Al tratado por Colinet et al.). O valor desse parâmetro foi estimado em $-1800 k_B K$

O resultado do cálculo mostrou que, apesar do discutido acima, a linha $DO_3 - B2$ penetra no campo de estabilidade da fase B2 obtido experimentalmente. Esse fato justifica, no presente trabalho, a adoção de parâmetros ligeiramente modificados, com base nos parâmetros de Colinet et al. O critério para esses novos parâmetros foi o mesmo que o desses autores. Com esses novos parâmetros, o campo de estabilidade da fase DO_3 (calculado) localiza–se totalmente fora do campo da fase B2 (experimental). Na verdade, o único valor que precisamos alterar é o parâmetro

$w_{CoAl}^{(2)}$, já que é o parâmetro responsável pela incompatibilidade entre os diagramas calculado e experimental. O novo valor estabelecido para esse parâmetro foi $w_{CoAl}^{(2)} = -600 k_B K$.

Os autores da referência [10] utilizaram apenas parâmetros de interação entre pares de primeiros e segundos vizinhos. Não foram usados parâmetros relativos aos tetraedros. Essa restrição nos leva a obter um diagrama de fases simétrico com relação a $x_{Al}=0,5$, conforme veremos a seguir.

4.3.Diagrama de fases

No caso do sistema Co–Cr, discutido no capítulo 3, foi necessário o uso de um diagrama de estados fundamentais, já que não sabíamos que fases estariam presentes no modelamento. No caso presente, isso não é necessário, uma vez que os parâmetros foram determinados em função das fases que estariam presentes no sistema. Essas fases são B2 e $D0_3$. O resultado está representado na figura 4.1.

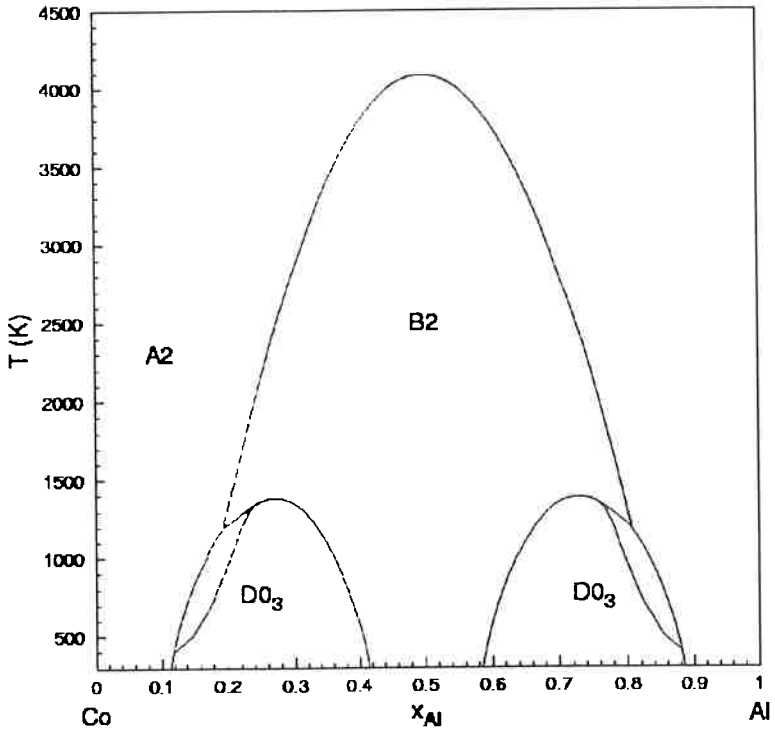


Figura 4.1 Diagrama de fases do sistema Co–Al CCC, calculado com $w_{CoAl}^{(1)}=-1800 k_B K$ *e* $w_{CoAl}^{(2)}=-600 k_B K$. *Como não foram utilizados parâmetros de interação entre tetraedros, o diagrama resulta simétrico em relação a* $x_{Al}=0,5$.

Com relação ao diagrama de fases calculado na referência [10], percebemos um abaixamento em torno de 200K com relação ao pico da região de estabilidade da fase D03. Entretanto, a estrutura B2 tornou–se mais estável, com um pico em torno de 4100K, 500K mais alto que o obtido por Colinet et al. Percebe–se a simetria do diagrama em relação a $x_{Al}=0,5$. A transição de fases A2/B2 é de segunda ordem, enquanto que a transição A2/$D0_3$ é de primeira ordem em uma ampla faixa de composições. A transição B2/$D0_3$ é de segunda ordem, com apenas um pequeno trecho próximo à transição A2/$D0_3$, que é de primeira ordem.

5. O sistema Cr–Al CCC

5.1. Introdução

Os dados relativos ao sistema Cr–Al foram extraídos de [6], tendo sido utilizados no modelamento do sistema Cr–Al–Ti CCC [5]. Esses dados baseiam–se em valores experimentais de atividade do alumínio no sistema Cr–Al a 1000°C, que foram ajustados com o CVM, em um procedimento muito semelhante àquele descrito na secção 3.1.6 do presente trabalho. Neste capítulo vamos descrever os dados experimentais de [6] e os procedimentos utilizados em [5] para a obtenção dos parâmetros do CVM relativos a esse sistema.

5.2. Dados experimentais

A atividade do alumínio em diversas ligas no sistema Cr–Al foi determinada usando uma técnica isopiestica entre 890 e 1126°C, com a fração de Al variando entre 0,13 e 0,80 [6]. Os autores detectaram um amplo campo de estabilidade da fase A2 na temperatura de 1000°C, extendendo–se até $x_{Al} \approx 0.42$. Esses valores experimentais, para as frações de alumínio correspondentes ao campo da fase A2 estável, encontram–se na tabela 5.1, e estão representados graficamente na figura 5.1

Tabela 5.1. Valores experimentais para a atividade do alumínio no sistema Cr–Al a 1273K [6]

x_{Al} (at–%)	$-\log a_{Al}$	x_{Al} (at–%)	$-\log a_{Al}$	x_{Al} (at–%)	$-\log a_{Al}$	x_{Al} (at–%)	$-\log a_{Al}$
37,96	1,11	18,55	2,32	28,52	1,81	26,51	1,83
36,48	1,22	17,07	2,38	26,01	1,96	24,17	1,95
34,04	1,36	15,81	2,46	23,82	2,09	20,95	2,07
33,51	1,46	14,69	2,53	21,37	2,21	21,04	2,17
30,24	1,59	13,88	2,58	17,96	2,31	41,23	1,08
29,10	1,70	42,16	0,97	17,02	2,40	39,81	1,18
26,62	1,81	39,91	1,12	41,12	1,02	37,87	1,29
24,78	1,90	38,28	1,25	38,51	1,14	36,74	1,39
24,81	1,98	36,59	1,35	36,66	1,29	32,14	1,59
22,98	2,08	34,50	1,45	34,06	1,41	30,77	1,67
21,56	2,14	31,89	1,58	31,58	1,55	26,50	1,86
18,65	2,24	29,98	1,70	27,59	1,69		

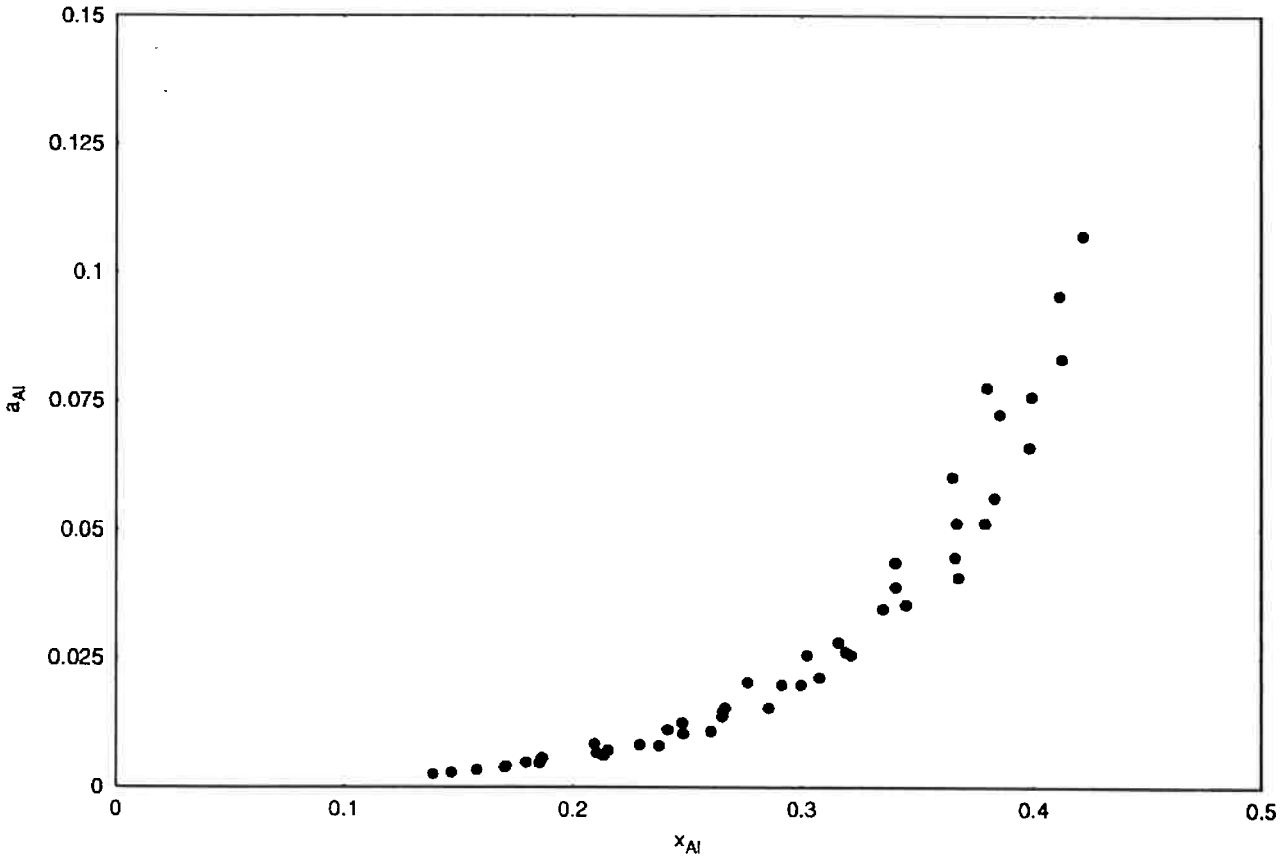


Figura 5.1. Atividades do alumínio no sistema Cr–Al a 1273K [6]

5.3.Ajuste dos dados ao CVM

O ajuste foi feito por tentativa e erro, em um procedimento semelhante àquele feito para o ajuste das energias livres de excesso do sistema Co–Cr (seção 3.1.6). Esse ajuste foi realizado em [5], na análise do sistema Cr–Al–Ti. O resultado do ajuste é fornecido na figura 5.2. Os parâmetros do CVM obtidos através desse ajuste estão na tabela 5.2.

Tabela 5.2. Parâmetros de interação iniciais e energias de formação para o sistema Co–Cr CCC

$w_{CrAlCrAl}$	$w_{CrAlAlAl}$	$w^{(2)}_{Cr,Al}$	$w^{(1)}_{Cr,Al}$
$-60\ [k_B.K]$	0	$-290\ [k_B.K]$	$-660\ [k_B.K]$
$U^f_{B32(CrAl)_{T=0K}}$	$U^f_{DO_3(Al_3Cr)_{T=0K}}$	$U^f_{DO_3(Cr_3Al)_{T=0K}}$	$U^f_{B2(CrAl)_{T=0K}}$
$-2550\ [k_B.K]$	$-1755.67[k_B.K]$	$-1755.67\ [k_B.K]$	$-2640\ [k_B.K]$

O parâmetro de interação $w_{CrAlAlAl}$ foi mantido em zero, de forma a manter o diagrama de

fases (calculado pelo CVM) simétrico, com relação a $x_{Al}=0,5$. Isso é justificado pelo fato de os dados experimentais limitarem–se a uma fração molar de alumínio de até 0,43, ou seja, apenas metade do diagrama.

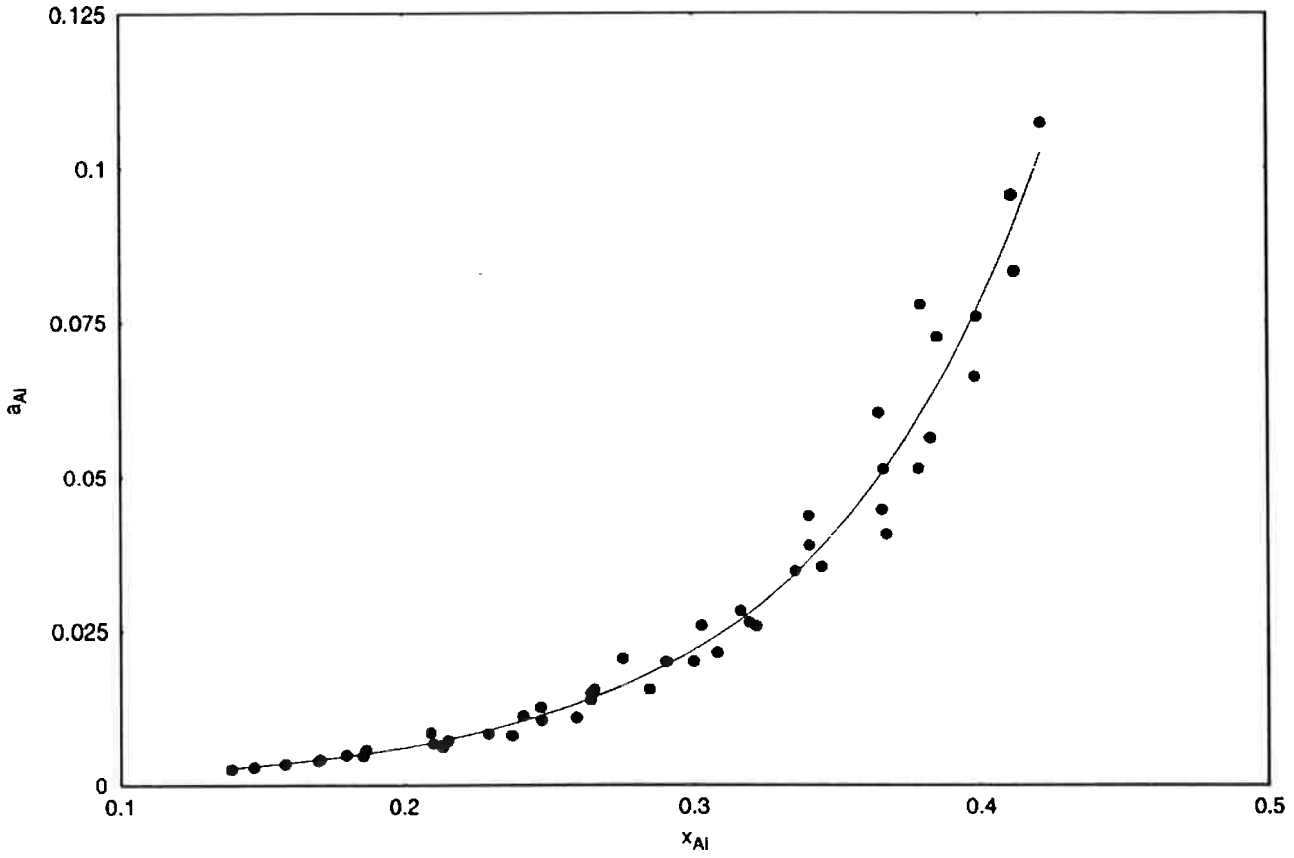


Figura 5.2. Ajuste dos dados experimentais da tabela 5.1 [6] usando para a obtenção dos parâmetros de interação da tabela 5.2 [5]

5.4Diagrama de fases

Os parâmetros da tabela 5.2 foram utilizados para o cálculo do diagrama de fases, exibido na figura 5.3. Nota–se a simetria em relação à linha central do diagrama (fração de alumínio igual a 0.5), como era esperado. O máximo de temperatura da fase B2 encontra–se a 999K. As regiões de estabilidade da fase $D0_3$ apresentam um máximo à temperatura de 612K, com $x_{Al}=0.34$. O cálculo mostra que há uma região de equilíbrio de primeira ordem A2/ $D0_3$.

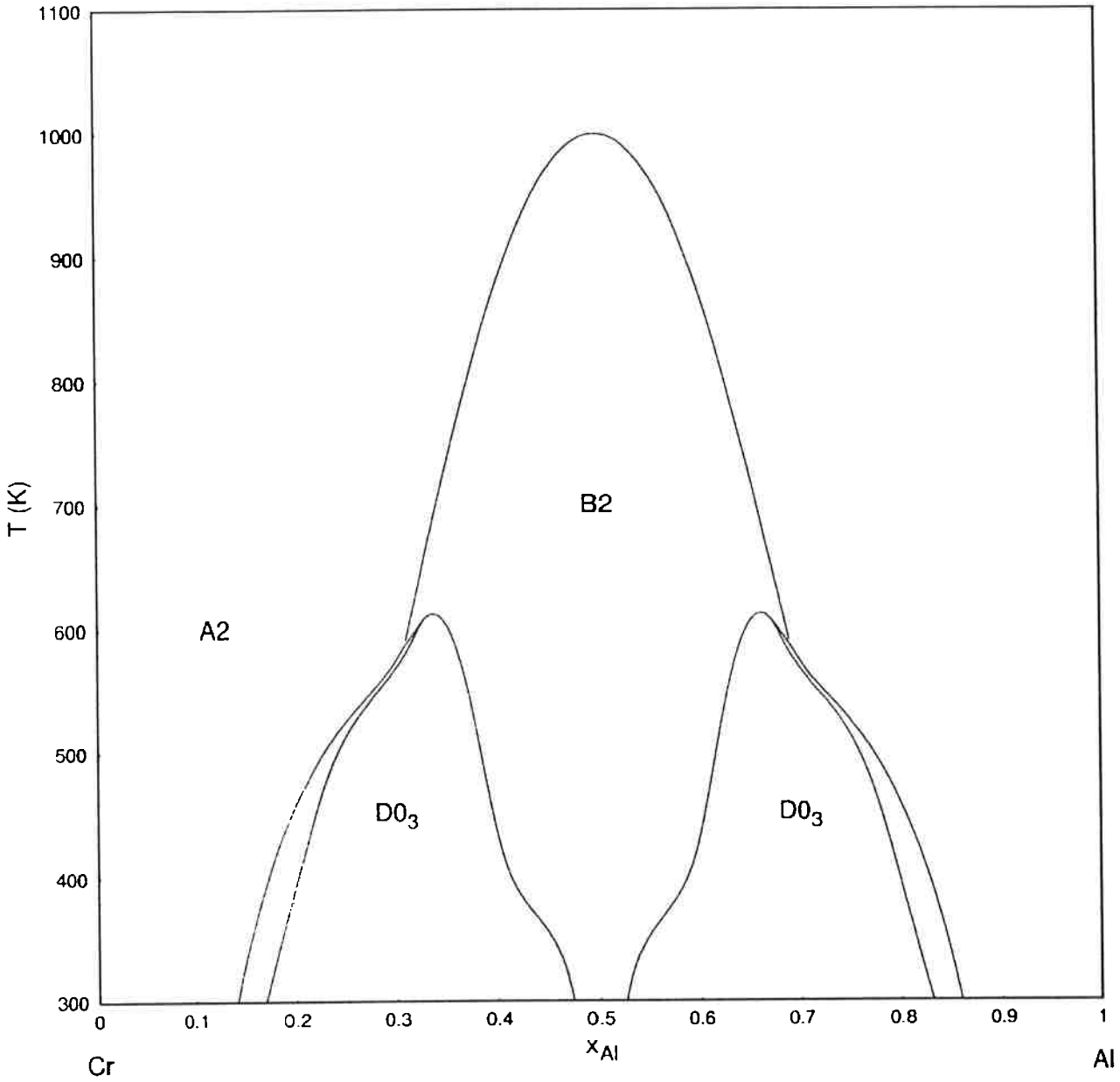


Figura 5.3. Diagrama de fases do sistema Cr–Al CCC calculado a partir dos parâmetros de interação fornecidos pela tabela 5.2 [5]

6. O sistema Co–Cr–Al CCC

O diagrama de fases ternário Co–Cr–Al CCC foi calculado a partir dos parâmetros de interação obtidos através dos procedimentos descritos nos capítulos 3, 4 e 5. Os resultados são mostrados juntamente com algumas tie–lines calculadas e outras experimentais obtidas da referência [3]. Também estão representados alguns pontos experimentais de segunda ordem, também determinados pela referência 3. As temperaturas das seções isotérmicas foram escolhidas a partir dessa mesma referência, para a comparação dos resultados. A figura 6.1 apresenta resumidamente as quatro seções isotérmicas calculadas, a 1273K, 1473K, 1573K e 1623K. Os parâmetros de interação estão na tabela 6.1. Detalhes dos cálculos e da confecção dos diagramas são fornecidos mais adiante.

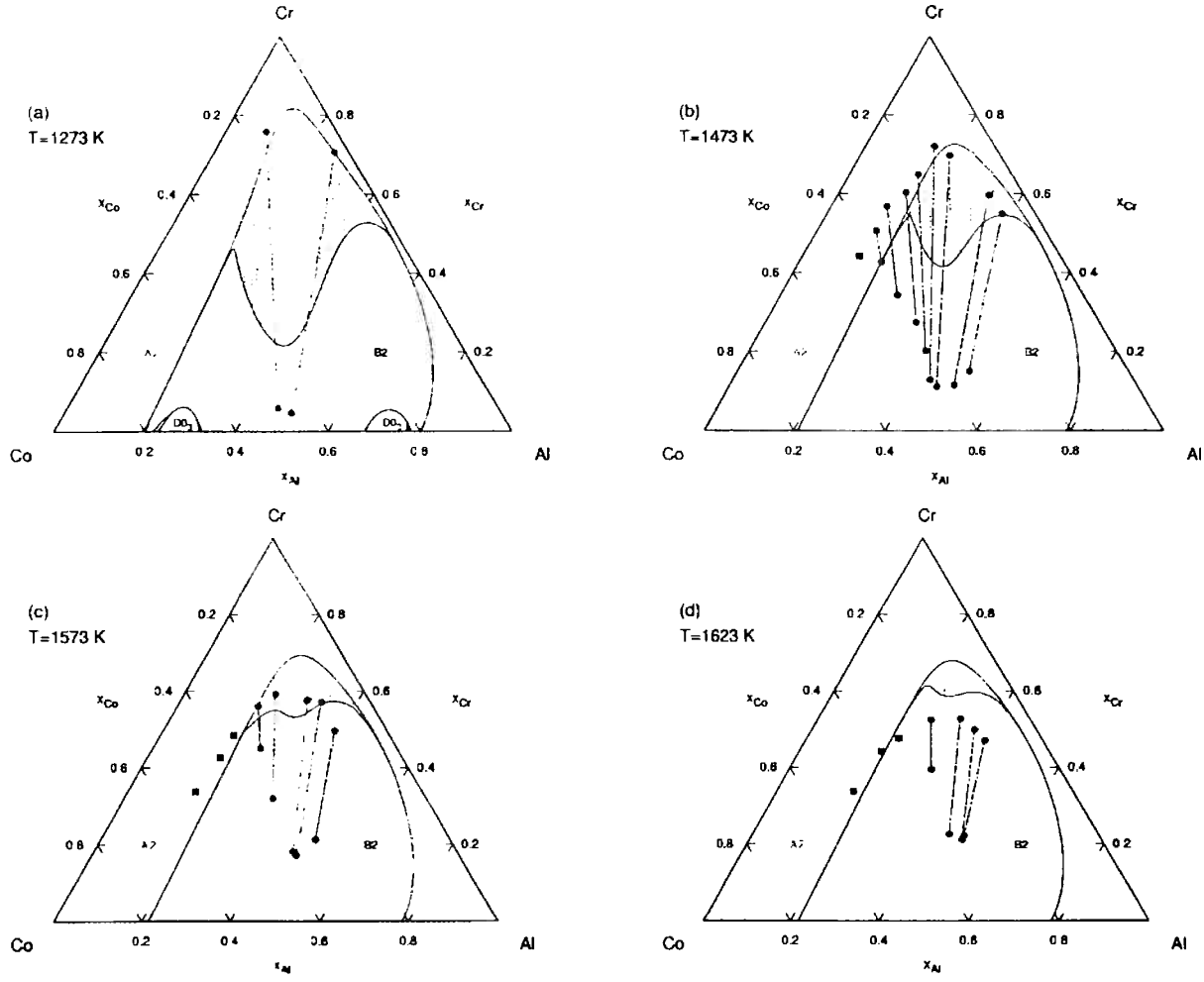


Figura 6.1. Seções isotérmicas do sistema Co–Cr–Al CCC calculados com os parâmetros da tabela 6.1. Linhas pontilhadas: tie–lines calculadas; círculos: tie–lines experimentais [3]; quadrados: pontos de 1a ordem experimentais [3]. Temperaturas: (a) 1000°C; (b) 1200°C; (c) 1300°C; (d) 1350°C;

Tabela 6.1. Parâmetros de interação utilizados no cálculo dos diagramas de fase do sistema Co–Cr–Al CCC

$w_{CrAlCrAl}$	$w_{CrAlAlAl}$	$w^{(2)}_{Cr,Al}$	$w^{(1)}_{Cr,Al}$
$-60\ [k_B.K]$	0	$-290\ [k_B.K]$	$-660\ [k_B.K]$
$w_{CoAlCoAl}$	$w_{CoAlAlAl}$	$w^{(2)}_{Co,Al}$	$w^{(1)}_{Co,Al}$
0	0	$-600\ [k_B.K]$	$-1800\ [k_B.K]$
$w_{CoCrCoCr}$	$w_{CoCrCrCr}$	$w^{(2)}_{Co,Cr}$	$w^{(1)}_{Co,Cr}$
$-60\ [k_B.K]$	$50\ [k_B.K]$	$-380\ [k_B.K]$	$+293\ [k_B.K]$

Observamos na figura 6.1 que apenas equilíbrios entre as fases B2 e A2 são encontrados nos diagramas, com apenas um pequeno campo de estabilidade da fase DO_3 a 1273K. Se analisarmos os diagramas binários dos capítulos 3, 4 e 5, perceberemos que, nas temperaturas das seções isotérmicas da figura 6.1 apenas as fases B2 e A2 são estáveis. A 1273K, a fase DO_3 ainda é estável no sistema Co–Al, e portanto deverá ser observada no diagrama ternário. É o que percebemos na figura 6.1.a.

As tabelas 6.2 e 6.3 apresentam os dados experimentais usados para a comparação da figura 6.1, obtidos da referência 3.

Tabela 6.2. Composições de equilíbrio entre as fases A2 e B2 no sistema Co–Cr–Al [3].

Temperatura	A2		B2	
(°C)	x_{Cr}	x_{Al}	x_{Cr}	x_{Al}
1000	0,758	0,091	0,058	0,465
	0,706	0,264	0,046	0,499
1200	0,506	0,130	0,427	0,180
	0,568	0,122	0,343	0,256
	0,603	0,145	0,274	0,331
	0,649	0,149	0,203	0,387
	0,720	0,149	0,129	0,433
	0,697	0,194	0,112	0,456
	0,596	0,329	0,116	0,492
	0,549	0,380	0,151	0,507
1300	0,561	0,185	0,451	0,246
	0,592	0,210	0,320	0,339
	0,576	0,286	0,182	0,452
	0,572	0,319	0,172	0,464
	0,496	0,388	0,213	0,485
1350	0,527	0,320	0,227	0,445
	0,498	0,366	0,213	0,480
	0,470	0,403	0,222	0,480
	0,524	0,256	0,395	0,321

Tabela 6.3. Composições críticas correspondentes à transição de segunda ordem entre as fases B2 e A2 no sistema Co–Cr–Al [3]

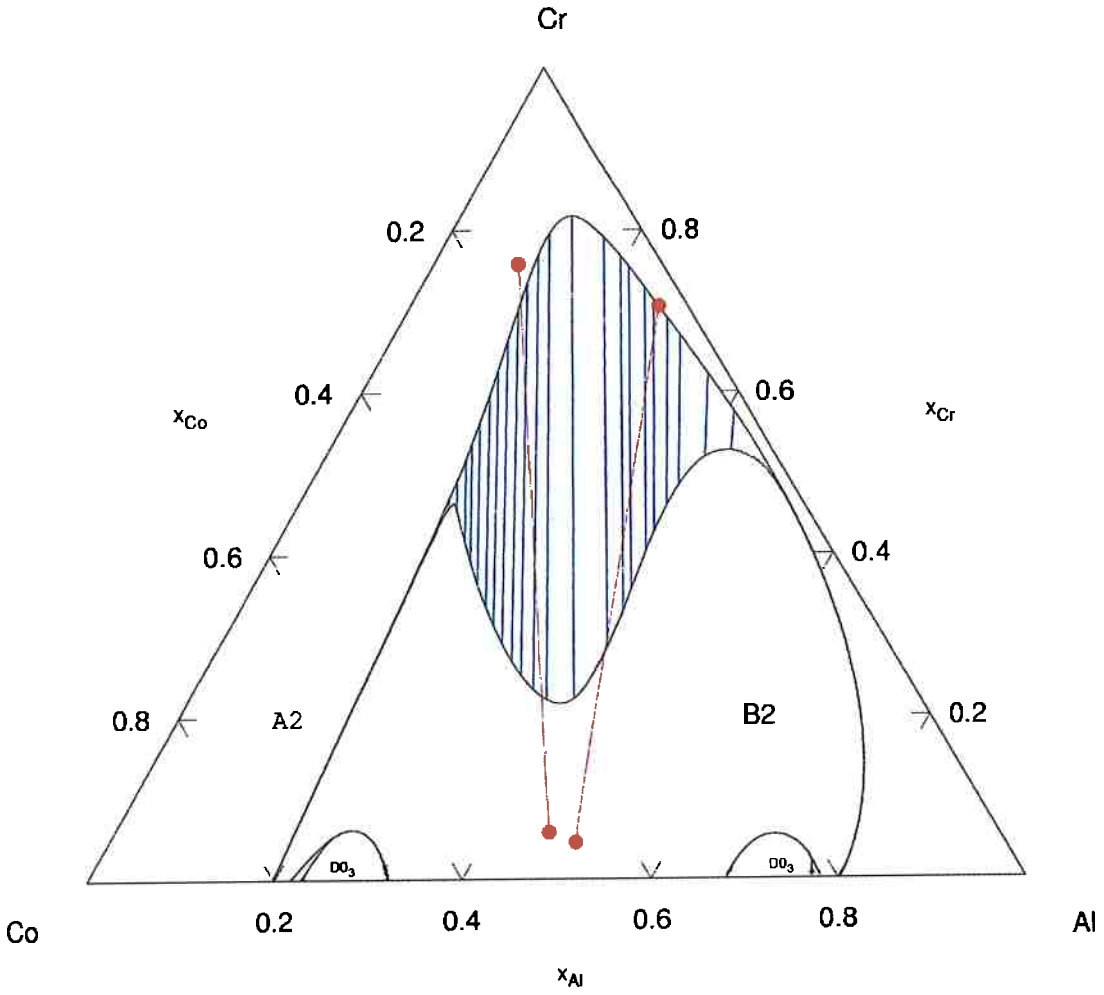
Temperatura (°C)	Composição crítica x_{Cr}	x_{Al}
1200	0,442	0,125
1300	0,338	0,154
	0,426	0,166
	0,484	0,169
1350	0,337	0,175
	0,442	0,185
	0,477	0,206

A seguir fornecemos algumas representações mais ampliadas dos diagramas de fases calculados, os mesmos que os da figura 6.1. Para o cálculo desses diagramas, foi necessário o uso de um método de suavização de curvas. Esse método está descrito no apêndice C. Essa suavização foi necessária porque em algumas regiões os diagramas apresentaram uma perturbação nas linhas dos equilíbrios de primeira ordem. Essa perturbação provavelmente foi resultado do grande número de pontos determinados no cálculo, muito próximos entre si. Essa é uma característica do algoritmo do programa CVM usado no cálculo. A convenção adotada nesses gráficos é a mesma da figura 6.1.

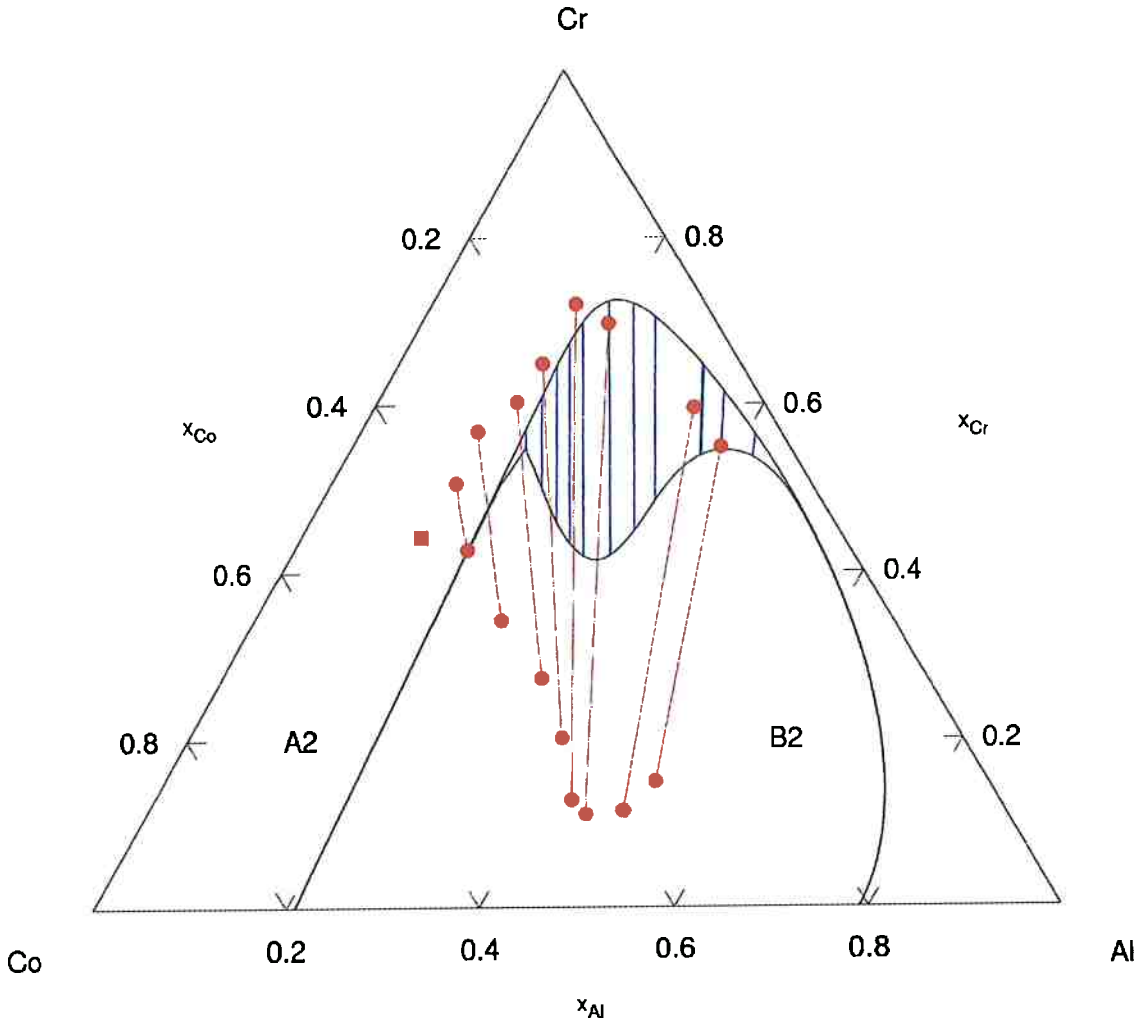
Os resultados mostraram um ajuste razoável, comparados aos dados de atividade e energias livres de excesso nos sistemas binários. O sistema ternário apresentou um bom ajuste também, apesar de o campo de estabilidade da fase B2 verificado experimentalmente ser menos amplo que o calculado.

Um ajuste mais aceitável pode ser obtido para esses diagramas, alterando os parâmetros do sistema Cr–Al. Esse sistema, assim como o sistema Co–Al, mostra–se simétrico com relação a $x_{Al}=0.5$. A razão para esse fato é que os parâmetros w_{ABBB} desses dois sistemas binários foram fixados em zero. No sistema Cr–Al, podemos variar esse parâmetro, bem como o parâmetro w_{ABAB} , de forma a aprimorar o ajuste. Além disso, os dados experimentais para os equilíbrios de primeira ordem se referem ao equilíbrio incoerente entre as fases B2 e A2 (i. e. o volume molar pode relaxar para um valor diferente para cada fase no equilíbrio termodinâmico), alterando o formato de domo de imiscibilidade. Como o presente formalismo CVM não considera a dependência no volume da energia livre, este efeito não pode ser modelado. O paralelismo entre as tie–lines (que são efetivamente linhas de isoatividade do sistema) demonstra entretanto que o modelamento termodinâmico do sistema é aceitável com os presentes parâmetros.

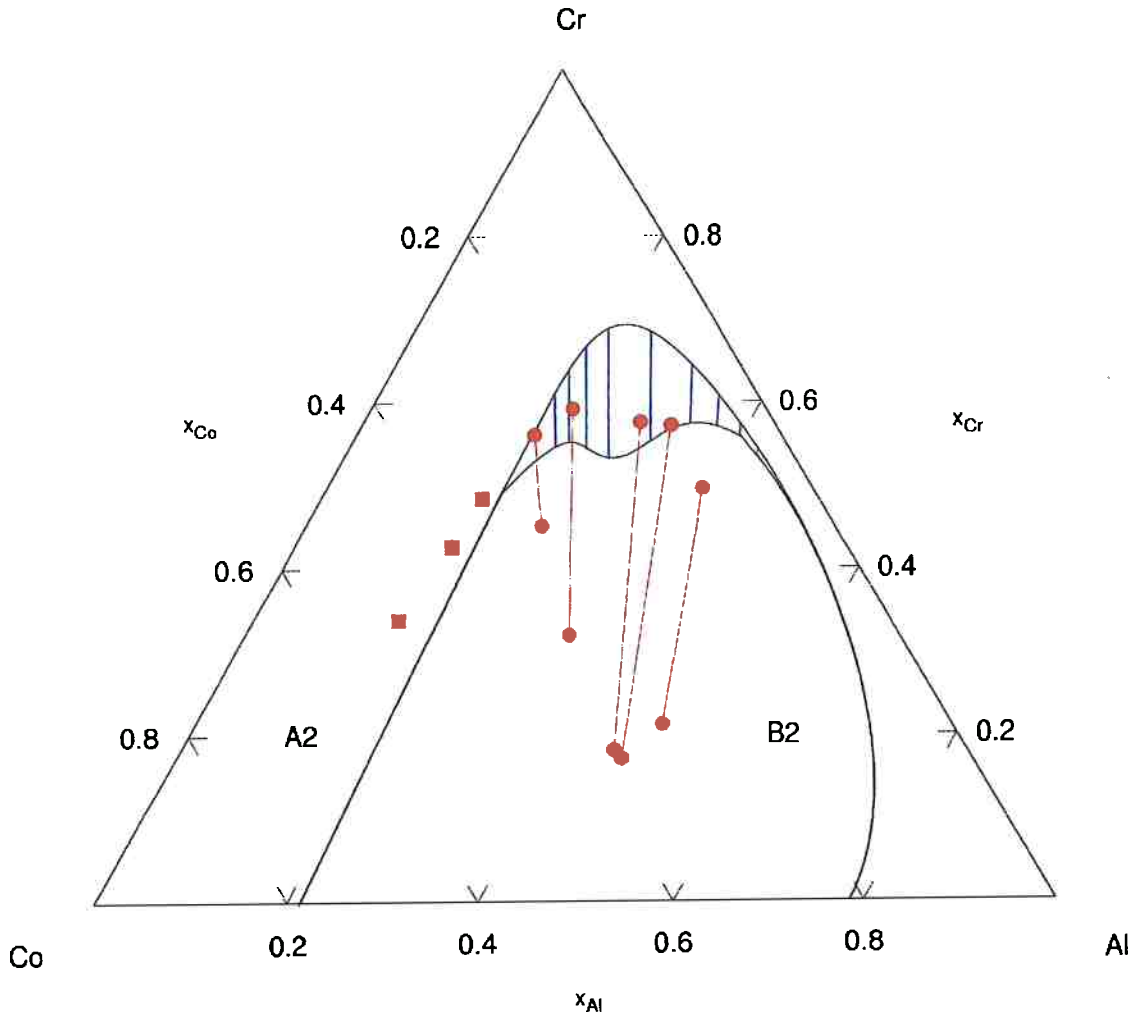
T=1000°C



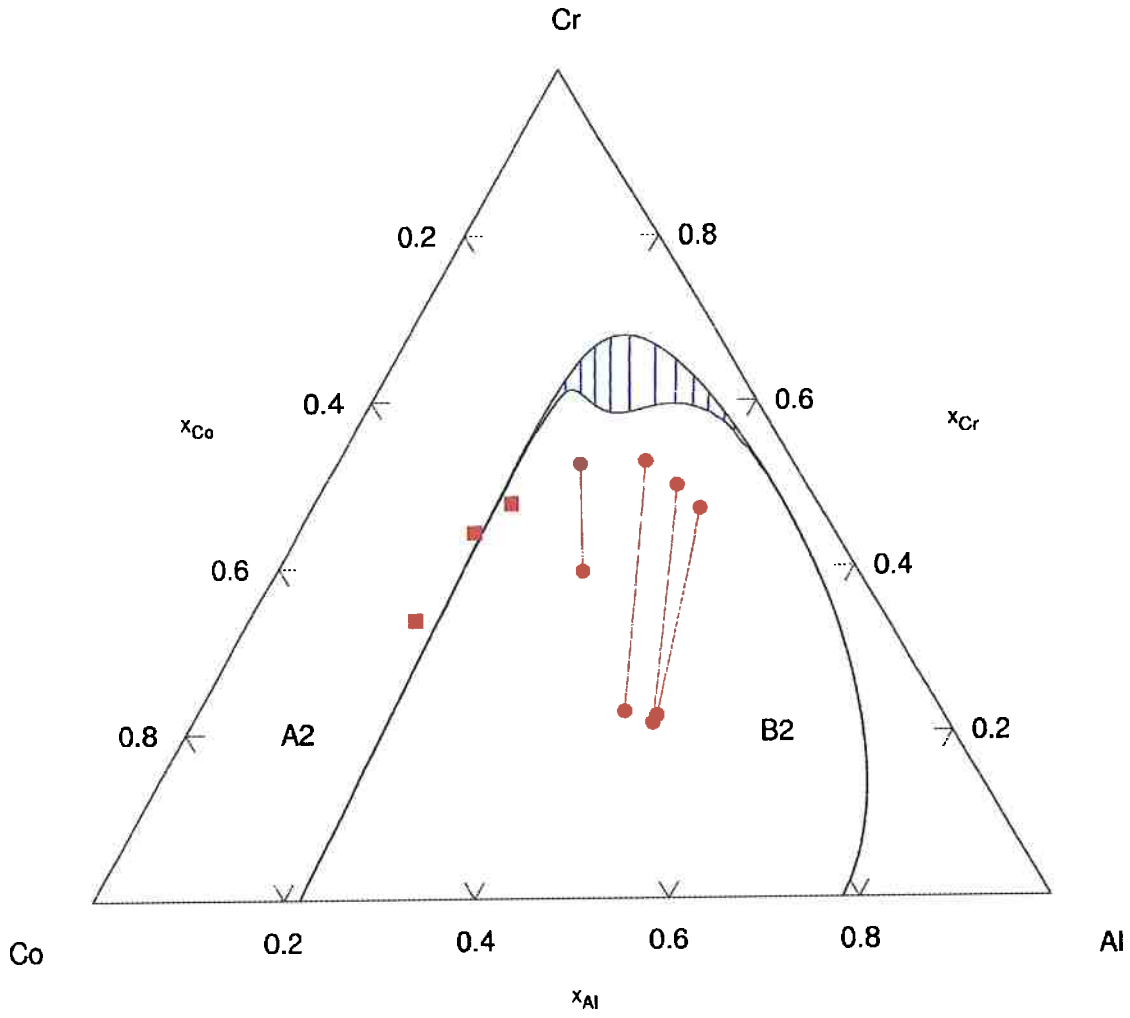
T=1200℃



T=1300℃



T=1350℃



Apêndice A: Potenciais químicos de excesso

Os nossos objetivos nesse apêndice são:

- deduzir expressões genéricas para os potenciais químicos de excesso em função da composição (Eqs. 3.17a e 3.17b), para uma liga binária;
- com as expressões para os potenciais químicos de excesso, deduzir as expressões 3.18a e 3.18b, que são o ajuste por série de potências para o sistema Co–Cr.

A.1. Potenciais químicos de excesso

O potencial químico de excesso do elemento *i é*, de acordo com a equação 3.16:

$$\mu_i^E = \left(\frac{\partial G^E}{\partial n_i} \right)_{T,n_j \neq n_i} = \left(\frac{\partial \left(n_T G_m^E \right)}{\partial n_i} \right)_{T,n_j \neq n_i} \tag{A.1}$$

Nessa equação, G^E é a energia livre da solução e G_m^E é a energia livre molar dessa solução. n_i é o número de mols da espécie *i* e n_T é o número total de mols da solução. O que nos propomos a fazer é deduzir uma expressão para os potenciais químicos de excesso em função das *frações molares* dos elementos da liga e da energia livre *molar* de excesso.

Utilizando a regra da cadeia, obtemos:

$$\mu_i^E = \left(\frac{\partial n_T}{\partial n_i} \right) G_m^E + n_T \left(\frac{\partial G_m^E}{\partial n_i} \right) \tag{A.2}$$

Nessa equação, as grandezas mantidas constantes foram omitidas para maior clareza. A partir desse ponto, vamos começar a nos referir aos elementos da liga, ao invés de tratá–los genericamente pelo sub–índice *i*.

Usando o fato de que $n_T = n_{Co} + n_{Cr}$, chegamos a:

$$\frac{\partial n_T}{\partial n_{Co}} = 1 \quad \text{e} \quad \frac{\partial n_T}{\partial n_{Cr}} = 1$$

Assim, o primeiro termo do segundo membro da equação A.2 está determinado. Resta–nos

definir o segundo termo. Para isso, devemos expandir a derivada parcial em termos da fração molar de cobalto da liga, usando a regra da cadeia:

$$\frac{\partial G_m^E}{\partial n_{Co}} = \frac{\partial G_m^E}{\partial x_{Cr}} \cdot \frac{\partial x_{Cr}}{\partial x_{Co}} \cdot \frac{\partial x_{Co}}{\partial n_{Co}} \tag{A.3}$$

O objetivo dessa expansão é deixar a derivada da energia livre em função da fração de cromo da liga. O motivo é manter a coerência com os métodos do capítulo 3.

Prosseguimos notando que, como $x_{Co}+x_{Cr}=1$, devemos ter $\frac{\partial x_{Cr}}{\partial x_{Co}} = -1$.

Para a última derivada parcial, lembramos que:

$$x_{Co}=\frac{n_{Co}}{n_{Co}+n_{Cr}}$$

e portanto:

$$\frac{\partial x_{Co}}{\partial n_{Co}}=\frac{1\cdot\left(n_{Cr}+n_{Co}\right)-n_{Co}\cdot 1}{\left(n_{Co}+n_{Cr}\right)^2}=\frac{n_{Cr}}{\left(n_{Co}+n_{Cr}\right)^2}=\frac{x_{Cr}}{n_T}$$

Com esses últimos resultados, obtemos a expressão procurada para o potencial químico de excesso do primeiro componente da liga:

$$\mu_{Co}^E=G_m^E-x_{Cr}\frac{\partial G_m^E}{\partial x_{Cr}} \tag{A.4}$$

Para o potencial químico de excesso do cromo, utilizamos a relação:

$$G_m^E=x_{Co}\mu_{Co}^E+x_{Cr}\mu_{Cr}^E \tag{A.5}$$

Com essa expressão e a equação A.4, chegamos a:

$$\mu_{Cr}^{E} = G_m^E + x_{Co}\frac{\partial G_m^E}{\partial x_{Cr}} \qquad (A.6)$$

As equações A.4 e A.6 são as equações procuradas, e são as mesmas equações 3.17a e 3.17b da seção 3.1.4.

A.2. Obtenção de equações algébricas para os potenciais químicos de excesso

As expressões A.4 e A.6 nos fornecem um meio de obtermos os potenciais químicos de excesso para os elementos de uma liga binária. Entretanto, aquela equação traz uma derivada parcial da energia livre de excesso em função da composição. Essas expressões só podem ser expandidas se tivermos equações para a energia livre. Caso contrário, devemos resolver numericamente a derivada.

No caso do ajuste do sistema Co–Cr, havíamos obtido expressões em série de potencias para a energia livre de excesso (eq. 3.15), dadas por:

$$G_m^E = x_{Co}\sum_{n=1}^{N} C_n^G x_{Cr}^n \qquad (A.7)$$

O potencial químico do cobalto é dado pela equação A.4. Para desenvolvê–la, devemos derivar a expressão acima em relação à fração de cromo, x_{Cr}. Isso é feito sem maiores problemas, lembrando que $\frac{\partial x_{Co}}{\partial x_{Cr}} = -1$ e usando a regra da cadeia:

$$\frac{\partial G_m^E}{\partial x_{Cr}} = -\sum_{n=1}^{N} C_n^G x_{Cr}^n + x_{Co}\sum_{n=1}^{N} n\, C_N^G x_{Cr}^{n-1}$$

Com essa expressão na equação A.4, obtemos a primeira das equações 3.18:

$$\mu_{Co}^{E} = x_{Co}\sum_{n=1}^{N} C_n^G x_{Cr}^n + x_{Cr}\sum_{n=1}^{N} C_n^G x_{Cr}^n - x_{Co}x_{Cr}\sum_{n=1}^{N} n C_n^G x_{Cr}^n \qquad (A.8)$$

ou:

$$\mu_{Co}^{E} = \sum_{n=1}^{N} C_n^G x_{Cr}^n \left(x_{Co} + x_{Cr} - n x_{Co}\right)$$

e, finalmente:

$$\mu_{Co}^{E} = \sum_{n=1}^{N} C_n^G x_{Cr}^n \left(1 - n x_{Co}\right) = \sum_{n=1}^{N} C_n^G x_{Cr}^n \left(1 - n + n x_{Cr}\right) \qquad (A.9)$$

A última expressão acima é a equação 3.18b. Para a outra equação, podemos repetir o procedimento feito para obter a equação 3.18b, mas usando a equação A.6 ao invés da equação A.4. Um procedimento alternativo é usar a equação A.5. Vamos utilizar essa última expressão:

$$x_{Co}\sum_{n=1}^{N} C_n^G x_{Cr}^n = x_{Co}\sum_{n=1}^{N} C_n^G x_{Cr}^n \left(1 - n x_{Co}\right) + x_{Cr}\mu_{Cr}^{E}$$

$$x_{Cr}\mu_{Cr}^{E} = \sum_{n=1}^{N}\left[C_n^G x_{Cr}^n \left(x_{Co} - x_{Co} + n x_{Co}^2\right)\right]$$

O resultado é a equação 3.18a:

$$\mu_{Cr}^{E} = x_{Co}^2 \sum_{n=1}^{N} n C_n^G x_{Cr}^{n-1} \qquad (A.10)$$

A.3. Referências

McGlashan, M. L., *Chemical Thermodynamics*, Academic Press, London, 1979.

Havrankova, J. Vrestal, J. Tomiska, *Berichte Bunsenges. Phys. Chem.* **102**, 1225–1230 (1998).

Apêndice B: Diagramas ternários no GNUPLOT

Grande parte dos diagramas apresentados nesse trabalho foi obtida através do software GNUPLOT v. 3.7, rodando no ambiente UNIX. O objetivo desse apêndice é listar os procedimentos necessários para a elaboração de diagramas ternários usando o GNUPLOT.

B.1. Macros e saída gráfica

A grande vantagem do GNUPLOT é a possibilidade de carregar informações através de macros, ou seja, arquivos contendo os comandos necessários para a confecção do gráfico. Outra vantagem importante é a saída do programa: existe a opção de salvar o gráfico no formato postscript. Os gráficos nesse formato podem ser inseridos diretamente em processadores de texto ou convertidos em arquivos GIF ou JPEG, se necessário.

A principal desvantagem na utilização do GNUPLOT para diagramas de fase ternários é que esse programa não inclui o triângulo de Gibbs como um tipo de gráfico padrão. Portanto, os eixos, as linhas de grade e os títulos devem ser todos inseridos "manualmente". Da mesma forma, a conversão entre o sistema de coordenadas triangulares (usados no triângulo) e as coordenadas cartesianas usuais também deve ser fornecida na macro de entrada do programa. Essa transformação de coordenadas também precisa ser calculada e introduzida "manualmente" no GNUPLOT.

B.2. Coordenadas triangulares

A figura B.1 abaixo mostra esquematicamente um diagrama de fases ternário A–B–C. O ponto P apresenta uma composição x_B do elemento B e x_C do elemento C, em termos de frações molares. Vamos partir do seguinte: temos as coordenadas triangulares do ponto P, ou seja, sabemos a sua composição (x_B e x_C), e queremos calcular as coordenadas cartesianas (x_P e y_P) desse ponto.

Para realizar a transformação, vamos considerar o triângulo PMN da figura B.1. Nesse triângulo, temos:

$$sen(60°) = \frac{\overline{PN}}{\overline{PM}} = \frac{y_P}{x_C} \qquad \text{(B.1)}$$

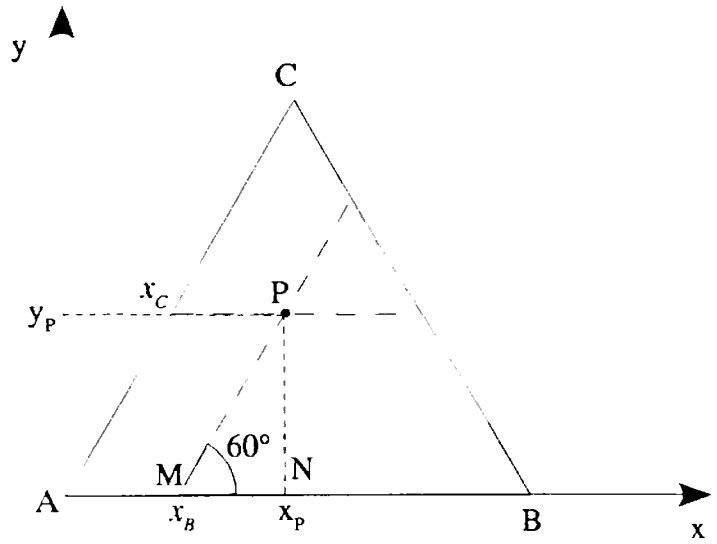


Figura B.1: Diagrama de fases ternário (esquemático)

Isolando y_P na expressão B.1, ficamos com:

$$y_P = x_C \; sen(60°) \tag{B.2}$$

Do mesmo triângulo, temos:

$$\cos(60°) = \frac{\overline{MN}}{\overline{PM}} = \frac{x_P - x_B}{x_C} \tag{B.3}$$

Substituindo o valor de y_P encontrado na expressão B.2 e isolando x_P, chegamos a:

$$x_P = x_B + x_C \cos(60°) \tag{B.4}$$

Portanto, a transformação de coordenadas é dada por:

$$\boxed{\begin{aligned} x_P &= x_B + x_C \cos(60°) \\ y_P &= x_C \; sen(60°) \end{aligned}} \tag{B.5}$$

Com a transformação B.5, estamos preparados para usar os diagramas triangulares no GNUPLOT. As linhas de grade e os eixos podem ser traçados usando essas transformações da seguinte forma:

- o eixo A–B é simplesmente o eixo x $\left(x_C=0\right)$;
- o eixo A–C é a imagem da reta $x_B=0$ em relação à transformação B.5;
- da mesma forma, o eixo B–C é a imagem da reta $x_B+x_C=1$;
- as linhas de grade paralelas ao eixo x são a imagem das retas $x_C=cte$;
- a imagem das linhas $x_B=cte$ correspondem às linhas de grade paralelas ao eixo A–C;
- finalmente, as linhas de grade correspondentes a $x_A=cte$ são a imagem das retas $x_B+x_C=1-cte$, uma vez que $x_A+x_B+x_C=1$.

B.3. Uso do GNUPLOT

Uma descrição dos comandos do GNUPLOT foge aos objetivos do presente trabalho. Portanto, vamos nos restringir ao suficiente para a plotagem de um diagrama ternário usando esse aplicativo.

B.3.1. Descrição dos comandos do GNUPLOT

A entrada dos comandos, como já foi dito, pode ser feita através de uma macro de comandos. Essa macro pode ser gravada em qualquer editor de textos, e o arquivo correspondente pode ter qualquer extensão. No caso, o nome escolhido foi `'ternary.gnu'`. Esse arquivo está reproduzido no final desse apêndice. Aqui, nos limitaremos à descrição dos comandos mais importantes.

- Primeiramente, o caracter # indica um comentário; o programa ignora qualquer coisa a partir desse símbolo até o fim da linha.
- Os comandos `set` definem algumas das opções do programa, como o formato de ângulo (graus, radianos), títulos, legenda, razão entre eixos, etc.
- é possível usar funções definidas pelo usuário. No caso do diagrama ternário, queremos entrar com a transformação B.5. Essas funções estão definidas logo no começo da macro, e foram

chamadas de `ternx(x,y)` e `terny(y)`; elas nada mais são que as equações B.5:

$$ternx(x,y) = x + y*\cos(60°)$$
$$terny(y) = y*\sin(60°)$$

- o comando `set arrow` traça uma seta entre dois pontos; a opção `nohead` indica que a seta se degenera em uma linha reta.
- o comando `set label` posiciona um texto qualquer em uma posição definida.
- o comando `plot "nome do arquivo" {opções}` traça a curva dada pelos pontos no arquivo. Para incluir outra curva no mesmo gráfico, usamos replot ao invés de plot.
- `set output "nome.ps"` muda a saída para o arquivo postscript correspondente. Desse modo, podemos salvar o gráfico.
- para ativar a macro, digitamos o comando `load 'ternary.gnu'` na linha de comando do GNUPLOT.
- o arquivo com os dados (no caso chamado de `'CoCrAl.txt'`) contém os dados para os pontos do diagrama de fases. Dentro dos comandos `plot` e `replot`, há a opção `using ($col1):($col2)`, que indica que, das várias colunas que o arquivo de dados possa ter, o gráfico será feito usando a `col1` no eixo x e a `col2` no eixo y.
- na área rotulada "User–defined" são feitas as modificações necessárias, ou seja, os títulos do gráfico e dos eixos e os nomes dos arquivos de entrada e saída.

B.3.2. A macro 'ternary.gnu'

```
# batch file ternary.gnu
# Gibbs triangle for GNUPLOT
# Latest version by Luiz Eleno July 26th 2000

# transformations for triangular coordinates
ternx(x,y)=x+y*cos(60)
terny(y)=y*sin(60)

reset
set angle degrees
set xrange [-0.15:1]
set yrange [-0.15:1]
set clip
set nokey
set noborder
set nozeroaxis
set noytics
set noxtics
set nogrid
set size square
```

```
# grid:
set linestyle 1 lt 0
# axes:
set linestyle 2 lt -2

#grid ticks
c1=0.1
c2=0.2
c3=0.3
c4=0.4
c5=0.5
c6=0.6
c7=0.7
c8=0.8
c9=0.9

# axes
set arrow 1 from  ternx(0,0),terny(0) to ternx(1.0,0),terny(0) nohead ls 2
set arrow 2 from  ternx(0,0),terny(0) to ternx(0,1.0),terny(1.0) nohead ls 2
set arrow 3 from  ternx(0,1.0),terny(1.0) to ternx(1.0,0),terny(0) nohead ls 2

# grid
set arrow 4 from  ternx(c1,0),terny(0) to ternx(c1,1-c1),terny(1-c1) nohead ls 1
set arrow 5 from  ternx(0,c1),terny(c1) to ternx(1-c1,c1),terny(c1) nohead ls 1
set arrow 6 from  ternx(c1,0),terny(0) to ternx(0,c1),terny(c1) nohead ls 1

set arrow 7 from  ternx(c2,0),terny(0) to ternx(c2,1-c2),terny(1-c2) nohead ls 1
set arrow 8 from  ternx(0,c2),terny(c2) to ternx(1-c2,c2),terny(c2) nohead ls 1
set arrow 9 from  ternx(c2,0),terny(0) to ternx(0,c2),terny(c2) nohead ls 1

set arrow 10 from  ternx(c3,0),terny(0) to ternx(c3,1-c3),terny(1-c3) nohead ls 1
set arrow 11from  ternx(0,c3),terny(c3) to ternx(1-c3,c3),terny(c3) nohead ls 1
set arrow 12 from  ternx(c3,0),terny(0) to ternx(0,c3),terny(c3) nohead ls 1

set arrow 13 from  ternx(c4,0),terny(0) to ternx(c4,1-c4),terny(1-c4) nohead ls 1
set arrow 14 from  ternx(0,c4),terny(c4) to ternx(1-c4,c4),terny(c4) nohead ls 1
set arrow 15 from  ternx(c4,0),terny(0) to ternx(0,c4),terny(c4) nohead ls 1

set arrow 16 from  ternx(c5,0),terny(0) to ternx(c5,1-c5),terny(1-c5) nohead ls 1
set arrow 17 from  ternx(0,c5),terny(c5) to ternx(1-c5,c5),terny(c5) nohead ls 1
set arrow 18 from  ternx(c5,0),terny(0) to ternx(0,c5),terny(c5) nohead ls 1

set arrow 19 from  ternx(c6,0),terny(0) to ternx(c6,1-c6),terny(1-c6) nohead ls 1
set arrow 20 from  ternx(0,c6),terny(c6) to ternx(1-c6,c6),terny(c6) nohead ls 1
set arrow 21 from  ternx(c6,0),terny(0) to ternx(0,c6),terny(c6) nohead ls 1

set arrow 22 from  ternx(c7,0),terny(0) to ternx(c7,1-c7),terny(1-c7) nohead ls 1
set arrow 23 from  ternx(0,c7),terny(c7) to ternx(1-c7,c7),terny(c7) nohead ls 1
set arrow 24 from  ternx(c7,0),terny(0) to ternx(0,c7),terny(c7) nohead ls 1

set arrow 25 from  ternx(c8,0),terny(0) to ternx(c8,1-c8),terny(1-c8) nohead ls 1
set arrow 26 from  ternx(0,c8),terny(c8) to ternx(1-c8,c8),terny(c8) nohead ls 1
set arrow 27 from  ternx(c8,0),terny(0) to ternx(0,c8),terny(c8) nohead ls 1

set arrow 28 from  ternx(c9,0),terny(0) to ternx(c9,1-c9),terny(1-c9) nohead ls 1
set arrow 29 from  ternx(0,c9),terny(c9) to ternx(1-c9,c9),terny(c9) nohead ls 1
set arrow 30 from  ternx(c9,0),terny(0) to ternx(0,c9),terny(c9) nohead ls 1
```

```
#ticks
set label 4 "1.0" at ternx(–0.06,0.0),terny(0.0)
set label 5 "0.0" at ternx(0.0,–0.04),terny(–0.04)
set label 6 "0.0" at ternx(1.02,0.0),terny(0.0)

set label 7 "0.9" at ternx(–0.06,c1),terny(c1)
set label 8 "0.1" at ternx(c1,–0.04),terny(–0.04)
set label 9 "0.1" at ternx(1–c1+0.02,c1),terny(c1)

set label 10 "0.8" at ternx(–0.06,c2),terny(c2)
set label 11 "0.2" at ternx(c2,–0.04),terny(–0.04)
set label 12 "0.2" at ternx(1–c2+0.02,c2),terny(c2)

set label 13 "0.7" at ternx(–0.06,c3),terny(c3)
set label 14 "0.3" at ternx(c3,–0.04),terny(–0.04)
set label 15 "0.3" at ternx(1–c3+0.02,c3),terny(c3)

set label 16 "0.6" at ternx(–0.06,c4),terny(c4)
set label 17 "0.4" at ternx(c4,–0.04),terny(–0.04)
set label 18 "0.4" at ternx(1–c4+0.02,c4),terny(c4)

set label 19 "0.5" at ternx(–0.06,c5),terny(c5)
set label 20 "0.5" at ternx(c5,–0.04),terny(–0.04)
set label 21 "0.5" at ternx(1–c5+0.02,c5),terny(c5)

set label 22 "0.4" at ternx(–0.06,c6),terny(c6)
set label 23 "0.6" at ternx(c6,–0.04),terny(–0.04)
set label 24 "0.6" at ternx(1–c6+0.02,c6),terny(c6)

set label 25 "0.3" at ternx(–0.06,c7),terny(c7)
set label 26 "0.7" at ternx(c7,–0.04),terny(–0.04)
set label 27 "0.7" at ternx(1–c7+0.02,c7),terny(c7)

set label 28 "0.2" at ternx(–0.06,c8),terny(c8)
set label 29 "0.8" at ternx(c8,–0.04),terny(–0.04)
set label 30 "0.8" at ternx(1–c8+0.02,c8),terny(c8)

set label 31 "0.1" at ternx(–0.06,c9),terny(c9)
set label 32 "0.9" at ternx(c9,–0.04),terny(–0.04)
set label 33 "0.9" at ternx(1–c9+0.02,c9),terny(c9)

set label 34 "0.0" at ternx(–0.06,1.0),terny(1.0)
set label 35 "1.0" at ternx(1.0,–0.04),terny(–0.04)
set label 36 "1.0" at ternx(0.02,1.0),terny(1.0)

# ————————————————User–defined area————————————————————

# Titles
set title "Co–Al–Cr\nT=1473K"
set label 1 "Co" at ternx(–0.06,–0.06),terny(–0.06)
set label 37  "X"  at 0.5,–0.1
set label 38 "Al" at 0.52,–0.12

set label 2 "Al" at ternx(1.08,–0.06),terny(–0.06)
set label 39  "X"  at 0.9,0.5
set label 40 "Cr" at 0.92,0.48

set label 3 "Cr" at ternx(–0.02,1.06),terny(1.06) center
```

```
set label 41  "X"  at 0.1,0.5
set label 42 "Co" at 0.12,0.48

#B2–A2
plot 'CoCrAl.txt' using (ternx($3,$2)):(terny($2)) w l ls 2
replot 'CoCrAl.txt' using (ternx($6,$5)):(terny($5)) w l ls 2

set output 'CoCrAl1200C.ps'

# ------------------------------------------------------------------------

set terminal postscript color solid 12
replot

set output
set terminal x11

# End of file
```

O resultado dessa série de comandos é a figura B.2 abaixo (o diagrama obtido é o resultado dos pontos fornecidos no arquivo 'CoCrAl.txt').

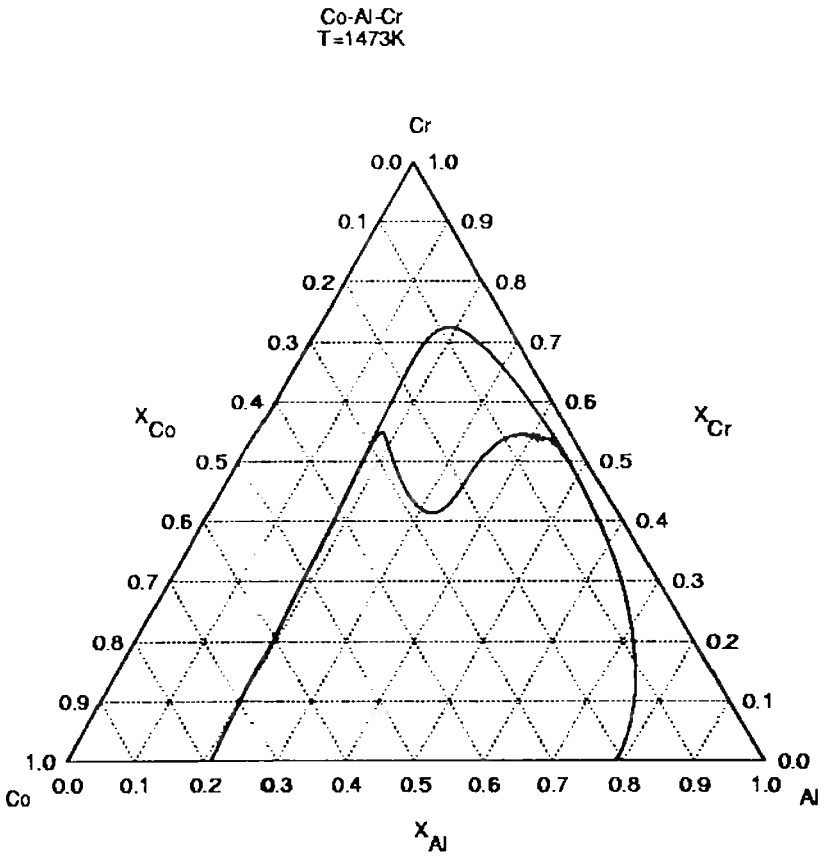


Figura B.2: Diagrama de fases ternário (exemplo)

Para esse gráfico poder ser utilizado no processador de texto em que foi redigido o presente trabalho (StarOffice 5.1) ele teve que ser convertido para o formato GIF. Para tanto, foi feito uso do programa *convert (convert nome.ps nome.gif)*, para transformá–lo. A seguir, o programa *Gimp*

foi usado para as demais conversões necessárias, como por exemplo, uma rotação de 90 graus para exibi–lo na posição adequada.

Como é possível perceber da análise da figura B.2, a resolução desse gráfico não é das melhores. Para melhorar a resolução, nos demais gráficos do presente trabalho foi utilizado o comando *set size 5* dentro do GNUPLOT, ou seja, ampliamos todas as figuras cinco vezes. Conforme pode ser visto nos demais gráficos do presente trabalho, a resolução torna–se bem melhor.

Apêndice C: Suavização de curvas

O gráfico C.1 representa uma relação qualquer entre duas grandezas, obtida experimentalmente ou calculada por um algoritmo qualquer. Por algum motivo, os dados estão bastante dispersos. Mesmo assim, percebemos uma clara linha de tendência pela qual esses pontos podem ser ajustados por um método de regressão, entre eles o método dos mínimos quadrados.

Para um ajuste por uma curva, é necessário que a função através da qual se queira ajustar os dados seja conhecida, ou que pelo menos se tenha uma idéia razoável do seu formato. Por exemplo, para os dados da figura C.1, um ajuste com uma função do tipo y=a+bx não é satisfatório. Da mesma forma, um ajuste através de uma equação do segundo grau não é suficiente, porque a curva apresenta um ponto de máximo e um de mínimo. Portanto, devemos realizar o ajuste com alguma função mais elaborada. Se não tivermos algumas indicações teóricas, ou pelo menos baseadas na experiência, do formato dessa função, temos um problema. A opção de obter uma curva ajustada por splines cúbicos (as curvas do Excel passando por todos os pontos do gráfico utilizam esse método) também é impraticável, já que o que desejamos é um ajuste suave, e não uma curva que ligue necessariamente todos os pontos.

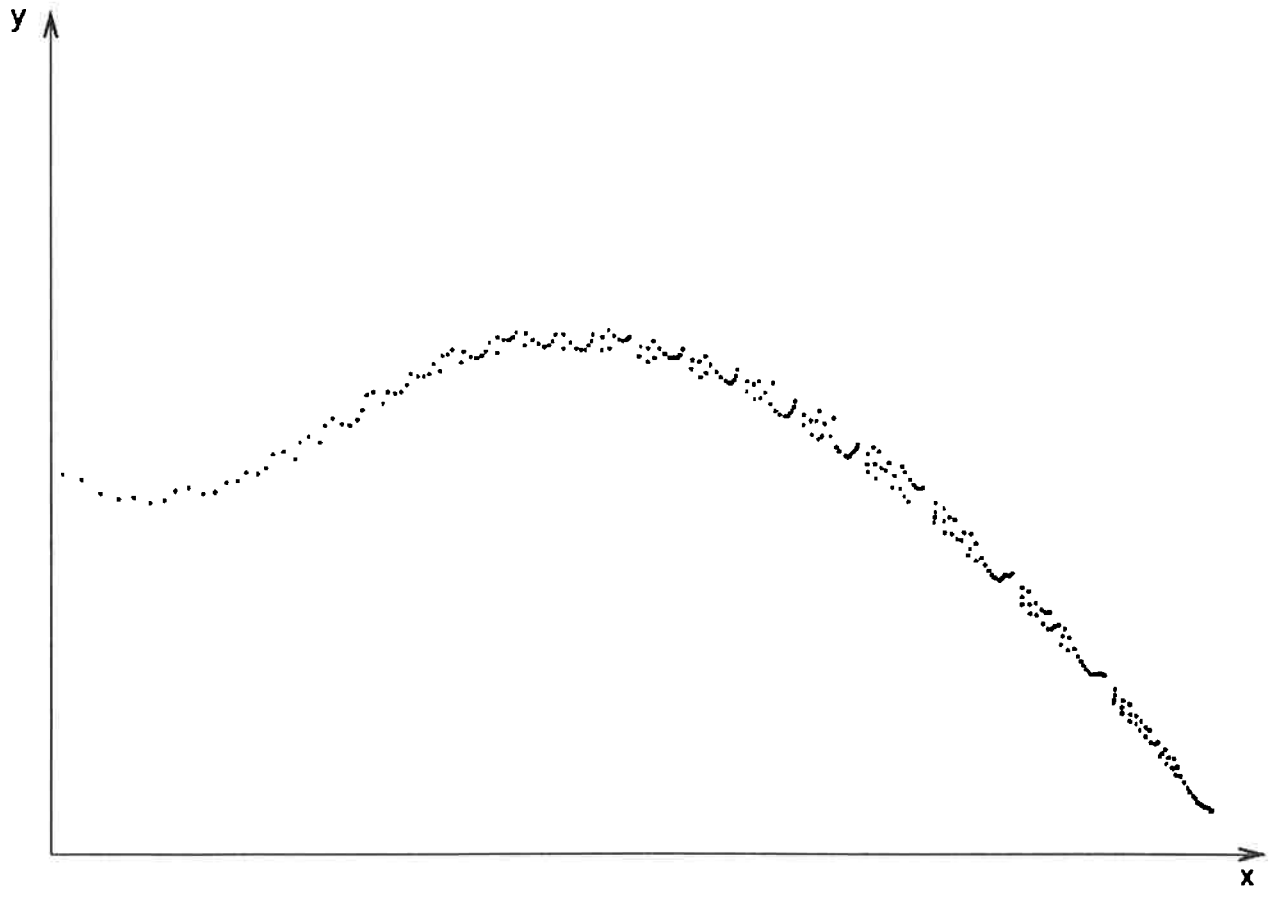


Figura C.1 Exemplo de dados experimentais dispersos

Para tentar resolver essa dificuldade, introduzimos aqui um método de suavização de curvas, que foi utilizado no presente trabalho para a suavização de algumas regiões dos diagramas ternários do capítulo 6. Esse método foi desenvolvido inteiramente pelo autor desse trabalho, apesar de não ser inédito. De fato, ele pode ser encontrado em livros a respeito de tratamento de imagens ou em alguns poucos livros de cálculo numérico.

O método se baseia no seguinte algoritmo: cada ponto é substituído pela média ponderada de si próprio e dos dois pontos adjacentes. Para facilitar a compreensão, pode–se observar a figura C.2.

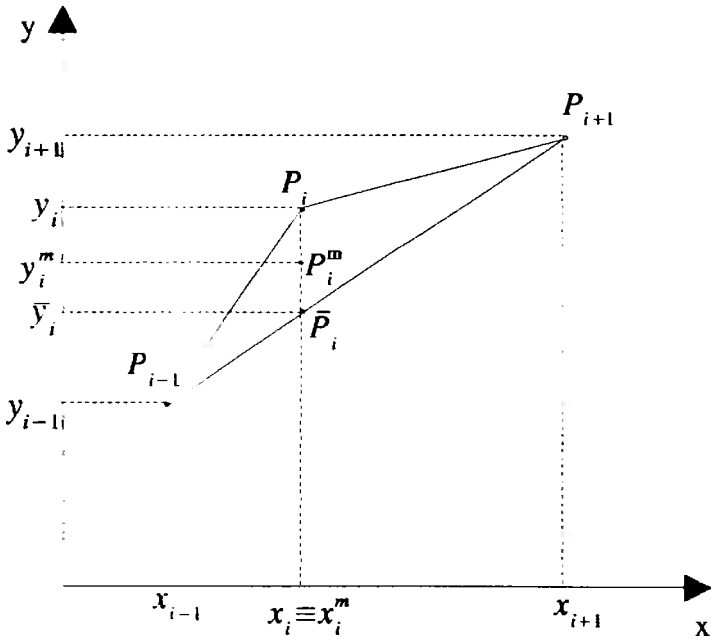


Figura C.2. Princípio do método de suavização de curvas

Os pontos P_{i-1}, P_i *e* P_{i+1} são três pontos consecutivos, pertencentes ao conjuntos de dados que se quer suavizar. As coordenadas desses pontos são $\{(x_j, y_j)\}$, com $j=\{i-1,\ i,\ i+1\}$, conforme indicado na figura C.2. O ponto P_i^m é o ponto em consideração para o ajuste. Ele é obtido da seguinte forma: primeiramente, calculamos as coordenadas do ponto $\bar{P}_i$, que pertence à reta que passa pelos pontos P_{i-1} *e* P_{i+1} e tem a mesma abcissa que o ponto P_i; a seguir, calculamos as coordenadas do ponto P_i^m, situado no ponto médio do segmento $P_i\bar{P}_i$. Desse modo, o ponto P_i^m também apresenta a mesma abcissa que o ponto P_i

.

Vamos agora obter uma expressão matemática para as coordenadas do ponto P_i^m . O método é o seguinte: calculamos inicialmente as coordenadas do ponto $\bar{P}_i$, e então calculamos o ponto P_i^m pela média com as coordenadas do ponto P_i . Como as abcissas são sempre as mesmas, podemos nos concentrar apenas nas ordenadas desses pontos.

Os pontos P_{i-1}, $\bar{P}_i$ *e* P_{i+1} pertencem à mesma reta; portanto, podemos escrever:

$$\frac{y_{i+1}-y_{i-1}}{x_{i+1}-x_{i-1}} = \frac{\bar{y}_i-y_{i-1}}{\bar{x}_i-x_{i-1}} \tag{C.1}$$

Usando o fato de que $\bar{x}_i=x_i$ e isolando $\bar{y}_i$, chegamos à ordenada do ponto $\bar{P}_i$:

$$\bar{y}_i = y_{i-1} + \frac{x_i-x_{i-1}}{x_{i+1}-x_{i-1}}\left(y_{i+1}-y_{i-1}\right) \tag{C.2}$$

A ordenada do ponto P_i^m , de acordo com o discutido acima, é:

$$y_i^m = \frac{y_i + \bar{y}_i}{2} \tag{C.3}$$

Substituindo o valor de $\bar{y}_i$ da equação C.2 na equação C.3 e rearranjando:

$$y_i^m = \frac{y_{i-1}(x_{i+1}-x_i) + y_i(x_{i+1}-x_{i-1}) + y_{i+1}(x_i-x_{i-1})}{2(x_{i+1}-x_{i-1})} \tag{C.4}$$

Para simplificar a expressão acima, podemos definir a variável $\Delta_x^{(i)}$ tal que:

$$\Delta_x^{(i)} = x_i-x_{i-1} \tag{C.5}$$

Além disso, se observarmos que

$$x_{i+1}-x_{i-1} = (x_{i+1}-x_i)+(x_i-x_{i-1}) = \Delta_x^{(i+1)} + \Delta_x^{(i)} , \tag{C.6}$$

a expressão C.4 torna–se, depois de mais alguns ajustes:

$$y_i^m = \frac{\Delta_x^{(i)}(y_{i+1}+y_i) + \Delta_x^{(i+1)}(y_i+y_{i-1})}{2(\Delta_x^{(i+1)}+\Delta_x^{(i)})} \qquad \text{(C.7)}$$

Finalmente, definindo as variáveis $\Sigma_y^{(i)}$ e $\Sigma_{\Delta x}^{(i)}$ tais que:

$$\Sigma_y^{(i)} = y_i + y_{i-1} \qquad \text{(C.8)}$$

e

$$\Sigma_{\Delta x}^{(i+1)} = \Delta_x^{(i+1)} + \Delta_x^{(i)} , \qquad \text{(C.9)}$$

podemos rescrever a equação C.4, chegando finalmente à fórmula para a suavização:

$$\boxed{y_i^m = \frac{\Delta_x^{(i)}\Sigma_y^{(i+1)} + \Delta_x^{(i+1)}\Sigma_y^{(i)}}{2\,\Sigma_{\Delta x}^{(i+1)}}} \qquad \text{(C.10)}$$

Essa expressão é válida para $2 \leqslant i \leqslant n-1$, onde n é o número total de pontos a serem suavizados. Para os pontos extremos, com $i=1$ e $i=n$, temos que adotar um procedimento alternativo. Existem duas opções:

a) fixar as coordenadas desses pontos extremos. Nessa caso, devemos adotar $y_1^m = y_1$ e $y_n^m = y_n$;

b) encontrar a reta de mínimos quadrados $y^{mq}(x_i)$ relativa aos dados e adotar para os pontos extremos os valores $y_1^m = y^{mq}(x_1)$ e $y_n^m = y^{mq}(x_n)$, fixando–os.

No presente trabalho, a opção a) foi utilizada.

Para exemplificar o uso da expressão C.10, podemos utilizá–la para suavizar uma curva usando uma planilha, como a do Excel. A tabela C.1 mostra um esboço de uma planilha que pode ser usada para esse objetivo.

Tabela C.1

Exemplo de uma planilha para a suavização de curvas

	Dados de entrada		Variáveis auxiliares			Dados suavizados	
i	x_i	y_i	$\Delta_x^{(i)}$	$\Sigma_y^{(i)}$	$\Sigma_{\Delta x}^{(i)}$	x_i^m	y_i^m
1	x_1	y_1	–	–	–	$=x_1$	$=y_1$
2	x_2	y_2	$= x_2 - x_1$	$= y_2 + y_1$	–	$=x_2$	=<eq. C.10>
3	x_3	y_3	$= x_3 - x_2$	$= y_3 + y_2$	$= \Delta_x^{(3)} + \Delta_X^{(2)}$	$=x_3$	=<eq. C.10>
...	...	...	...	...	...	...	...
$n-2$	x_{n-2}	y_{n-2}	$= x_{n-2} - x_{n-3}$	$= y_{n-2} + y_{n-3}$	$= \Delta_x^{(n-2)} + \Delta_X^{(n-3)}$	$=x_{n-2}$	=<eq. C.10>
$n-1$	x_{n-1}	y_{n-1}	$= x_{n-1} - x_{n-2}$	$= y_{n-1} + y_{n-2}$	$= \Delta_x^{(n-1)} + \Delta_X^{(n-2)}$	$=x_{n-1}$	=<eq. C.10>
n	x_n	y_n	$= x_n - x_{n-1}$	$= y_n + y_{n-1}$	$= \Delta_x^{(n)} + \Delta_X^{(n-1)}$	$=x_n$	$=y_n$

Para ilustrar o método, vamos aplicá–lo para os dados da figura C.1. O resultado está mostrado na figura C.3.

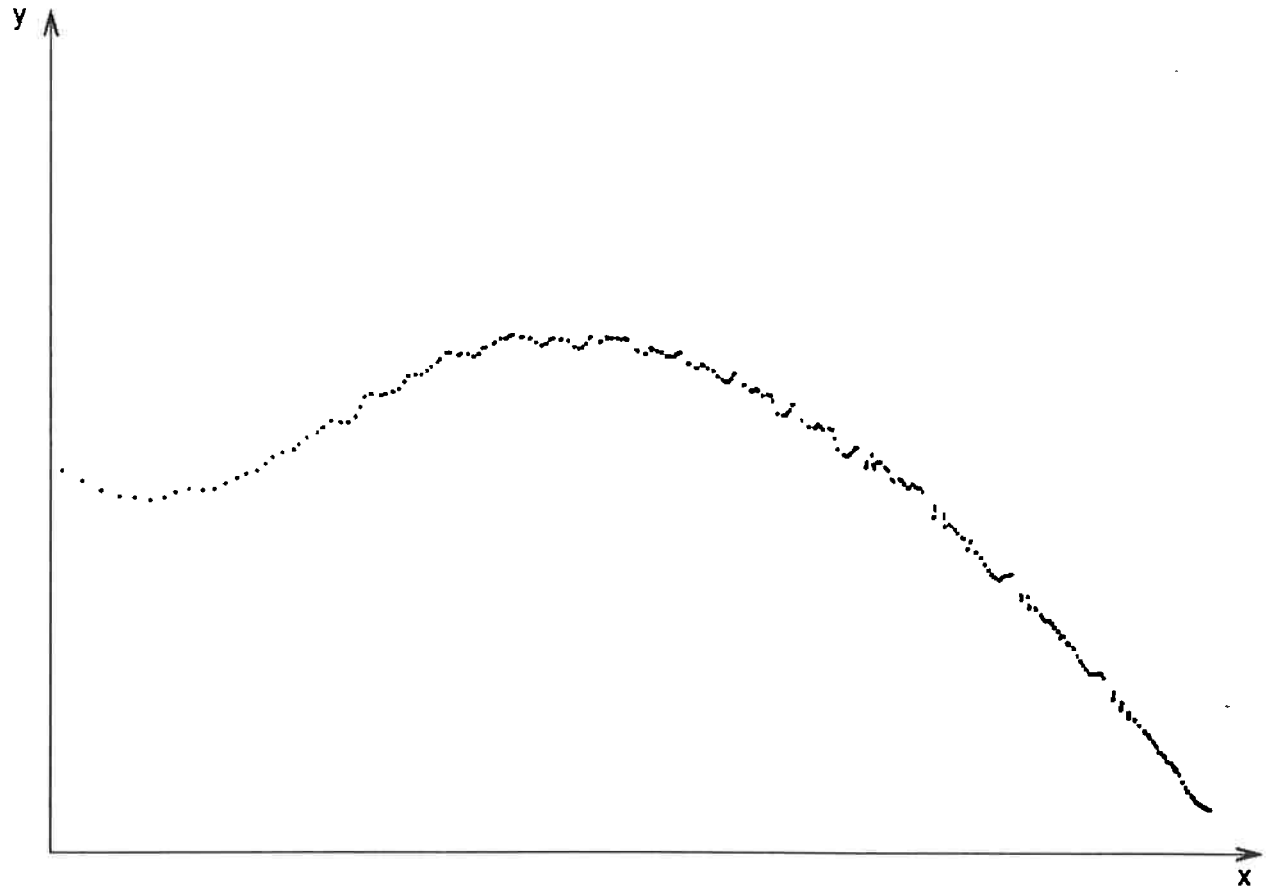


Figura C.3 Suavização dos dados da figura C.1

A escala dos gráficos das figuras C.1 e C.3 é a mesma. Desse modo, podemos perceber que o método resultou em uma suavização considerável, apesar de ainda não ser satisfatória, pelo

menos nessa escala. Ressaltamos que o método, a partir desse ponto, pode se tornar iterativo, com os valores de y_i^m substituindo os de y_i e repetindo o procedimento até que a suavização torne-se aceitável. Para isso, o método da planilha descrito acima não é conveniente; a melhor opção é utilizar um algoritmo que permita as iterações. O Visual Basic for Applications (VBA) é adequado se for preferível continuar trabalhando com a planilha, através do uso de macros; caso contrário, deve-se usar alguma outra linguagem.

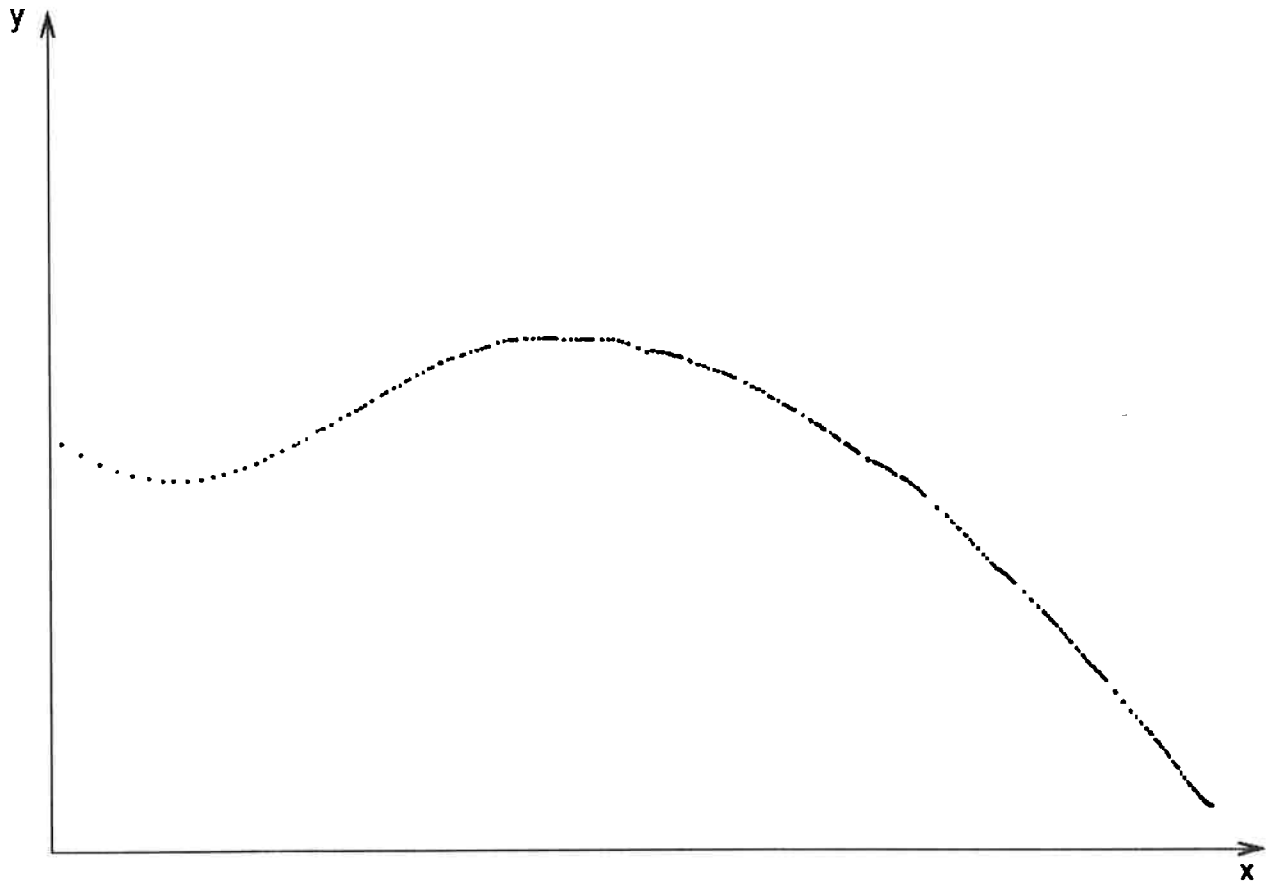


Figura C.4 Dados suavizados utilizando o algoritmo iterativo

No presente trabalho, não foi necessário o uso de mais de uma iteração. Entretanto, apenas como exemplo, isso foi feito para os dados desse apêndice. O resultado está indicado na figura C.4. Foram necessárias algumas dezenas de iterações para se atingir essa suavização; poderia-se atingir uma suavização melhor com mais algumas iterações. Entretanto, à medida em que o número de iterações aumenta, a suavização tende à curva mais suave possível, ou seja, uma linha reta. No caso (a), utilizado nesse apêndice e citado anteriormente, a curva tende para a reta ligando os pontos extremos; no caso (b), a suavização reproduz a reta de mínimos quadrados. Logo, deve-se tomar cuidado com a suavização excessiva, que pode descaracterizar os dados de partida.

Um caso particular da expressão C.10 é quando os pontos a serem suavizados estiverem igualmente espaçados no eixo x, ou seja, $\Delta x^{(i)}=cte$. Nesse caso, a expressão C10 simplifica-se bastante, após ser convenientemente rearranjada:

$$y_i^m=\frac{y_{i+1}+2y_i+y_{i-1}}{4} \qquad \text{(C.11)}$$

Salientamos que, no presente trabalho, a equação utilizada foi de fato a equação C.10, ou seja, a equação para o caso geral, no qual os pontos não estão espalhados uniformemente pelo eixo das abcissas.

Apêndice D: Código do programa (cvm17bf2.f) utilizado nos cálculos

```
C*****************************************************************C
      program cvm17bf2
C*****************************************************************C
C     created by C. G. Schoen (7 August 1998)
C     based on the program cvm17bf (C. G. Schoen)
C     first compiled at zirconio.pmt.usp.br
C     (Pentium X2 Linux workstation)
C     with fort77 compiler
C
C     Compiled in 27th October 1999 at zirconio.pmt.usp.br
C     with the newer g77 (egcs) gnu FORTRAN compiler
C
C     last modified by C. G. Schoen (27th October 1999)
C
C     Substituted obsolete real*8 and integer*4
C     types by real(kind=2) and integer(kind=2)
C*****************************************************************C
C     CVM calculations for systems up to 17 species (either spin
C     states or alloy components) in the tetrahedron approximation
C     for the FCC and BCC lattices.
C     Natural iteration minimization
C     Constrained equilibria (not yet implemented)
C     True chemical potentials
C     APB energy calculation
C*****************************************************************C

      Parameter(NN=17)

C*****************************************************************C
C     Declarations
C*****************************************************************C

      logical*2 useclap,useread,state_f,constrained

      integer(kind=2) i,j,k,l,m
      integer(kind=2) i2
      integer(kind=2) j2,k2,l2
      integer(kind=2) Nel,Nmag,N
      integer(kind=2) nspin,ncomp
      integer(kind=2) Jph,Iph
      integer(kind=2) elmt
      integer(kind=2) ibryf,iversion
      integer(kind=2) ibry,icount
      integer(kind=2) ifp,ifn,inp
      integer(kind=2) nrec
      integer(kind=2) DimSys
      integer(kind=2) num_c,c_comp
      integer(kind=2) Iphase
      integer(kind=2) nph
      integer(kind=2) deg_f
      integer(kind=2) itr,itrD,itrO

      dimension nspin(NN),c_comp(NN-1)
      dimension nph(2)
      dimension itr(2)

      real(kind=2) epsb,epcb,epxb
      real(kind=2) epsf,epcf
      real(kind=2) ref_stI,ref_stJ
      real(kind=2) CP,CPX1,CPX2,CP1,CP2
      real(kind=2) mju,acr
      real(kind=2) g,spin
      real(kind=2) r_CP,delCPX,delCP
C      real(kind=2) rotCP
C      real(kind=2) r_CP_old
      real(kind=2) TK,RTK,delTK
      real(kind=2) fl,dfCP,flp,fln
      real(kind=2) dGPp,dGPn
      real(kind=2) Zp,ZO,ZD
      real(kind=2) xal,xbt,xgm,xdt
      real(kind=2) c_al,c_bt,c_gm,c_dt
      real(kind=2) mag,m_al,m_bt,m_gm,m_dt
      real(kind=2) si
      real(kind=2) sj,sk,sl
      real(kind=2) constr
      real(kind=2) GP,GPD,GPO,dGP
      real(kind=2) Fgy,FgyD,FgyO
      real(kind=2) Etpy,EtpyD,EtpyO
      real(kind=2) Egy,EgyD,EgyO
C      real(kind=2) Gref,echr
C      real(kind=2) GPref,PCref
      real(kind=2) ecp
      real(kind=2) pl
      real(kind=2) distance
      real(kind=2) unit_vector
C      real(kind=2) ortho_vector
      real(kind=2) k_B,m_B
      real(kind=2) DE_apb,s_apb
      real(kind=2) pABGD
      real(kind=2) pABG,pABD,pAGD,pBGD
      real(kind=2) pAB,pAG,pAD,pBG,pBD,pGD
      real(kind=2) pA,pB,pG,pD
      real(kind=2) part1,part2,part3,part4,part5,part6



      dimension epsb(NN,NN,NN,NN),epcb(NN,NN,NN,NN)
      dimension epxb(NN,NN,NN,NN)
      dimension epsf(NN,NN,NN,NN),epcf(NN,NN,NN,NN)
      dimension ref_stI(NN,8),ref_stJ(NN,8)
      dimension CP(NN),CPX1(NN),CPX2(NN),CP1(NN),CP2(NN)
      dimension mju(NN),acr(NN)
      dimension g(NN),constr(NN-1)
      dimension spin(4)
      dimension r_CP(NN)
      dimension ZO(NN,NN,NN,NN),ZD(NN,NN,NN,NN)
      dimension Zp(2,NN,NN,NN,NN)
      dimension GP(2),Fgy(2),Etpy(2),Egy(2),mag(2)
      dimension xal(2,NN),xbt(2,NN),xgm(2,NN),xdt(2,NN)
      dimension m_al(2,NN),m_bt(2,NN),m_gm(2,NN),m_dt(2,NN)
      dimension c_al(2,NN),c_bt(2,NN),c_gm(2,NN),c_dt(2,NN)
C      dimension Gref(2,NN),PCref(2,NN),echr(NN,NN,NN,NN)
      dimension pl(2,NN)
      dimension ecp(NN,NN,NN,NN)
      dimension unit_vector(NN)
C      dimension ortho_vector(NN)
      dimension delCPX(NN),delCP(NN)
      dimension DE_apb(2,2),s_apb(2,2)
      dimension pABGD(NN,NN,NN,NN)
      dimension pABG(NN,NN,NN),pABD(NN,NN,NN)
      dimension pAGD(NN,NN,NN),pBGD(NN,NN,NN)
      dimension pAB(NN,NN),pAG(NN,NN),pAD(NN,NN)
      dimension pBG(NN,NN),pBD(NN,NN),pGD(NN,NN)
      dimension pA(NN),pB(NN),pG(NN),pD(NN)

      character*1 use_abs_G,print_eqm
      character*2 component,Mode
      character*3 structJ,structJ,BCC,FCC,struct
      character*15 Filename1,Filename2,Filename3
      character*15 Filename4
      character*20 Sysname

      dimension component(NN)
      dimension struct(2)

      Common/Csys/Nel,Nmag,ncomp,nspin,component
      Common/CBCC/epsb,epcb,epxb
      Common/CFCC/epsf,epcf
      Common/Cpot/ecp,TK,RTK
      Common/Cord/ZO
      Common/Cdis/ZD

      Data icountf,dflf,dGPf/10000,1.0e-4,0.01/

C---------------------------------------------------------------C
C     Main
C---------------------------------------------------------------C

C-------------------------------------------------------C
C     Physical constants:
C     k_B: Boltzmann constant [J/mol.K]
C     m_B: Bohr magneton [J/mol.T]
C-------------------------------------------------------C

      k_B=8.3145d+0
      m_B=5.585d+0

      BCC='BCC'
      FCC='FCC'

C---------------------------------------------------------------C
C     Opening input and output files
C     Reading and writing file headers
C---------------------------------------------------------------C

      Open(Unit=5,File='cvm17bf2.inp',status='old',form='formatted')

 10   Format(4X,I2,6X,I2,6X,A20)
      Read(5,10) Nel,Nmag,Sysname


 20   Format(A15)
      Read(5,20) Filename1
      Read(5,20) Filename2
      Read(5,20) Filename3
      Read(5,20) Filename4

      Open(Unit=10,File=Filename3,Status='New',form='Formatted')
      Open(Unit=12,File=Filename1,Status='New',form='Formatted')
      Open(Unit=14,File=Filename2,Status='New',form='Formatted')
      Open(Unit=15,File=Filename4,Status='New',form='Formatted')

 25   Format(1X,'######## cvm17bf2 calculation report #######'/)
      Write(10,25)

C---------------------------------------------------------------C
C     Fundamental definitions:
C
C     Sysname: A label for the system
C     Nel: Number of species (spin states and alloy components)
C     Nmag: Number of magnetic components
C     nspin: Number of spin states for each magnetic componen
C     nspin(i)= 2*(s(i))+1
C     N: Number of non-magnetic alloy components
C     ncomp: Number of alloy components
C     components: The labels for the components
C
C     StructI,StructJ:Lattice types (BCC or FCC)
C     Iph,Jph:Phase labels
C---------------------------------------------------------------C

 30   Format(1X,A20,2X,',','Nel=',I2,2X,',','Nmag=',I2,',')
      Write(12,30) Sysname,Nel,Nmag
      Write(14,30) Sysname,Nel,Nmag
      Write(15,30) Sysname,Nel,Nmag

      do i=1,Nel
         nspin(i)=1
         g(i)=0.0
      enddo

 40   Format(17I2)
 45   Format(8E10.4)

      if (Nmag.ne.0) then
         Read(5,40) (nspin(i),i=1,Nmag)
         Read(5,45) (g(j),j=1,Nmag)
         N=Nel
         do i=1,Nmag
            N=N-nspin(i)
         enddo
      else
         N=Nel
      endif



 50   Format(1X,'Inconsistency: Nel<sum(nspin(i))')
      if (N.lt.0) then
         write(10,50)
         goto 5000
      endif

      ncomp=N+Nmag

c      do i=1,ncomp
c         write(*,40) nspin(i)
c      enddo

 60   Format(17(A2,1X))
      Read(5,60) (component(i),i=1,ncomp)

 70   Format(1X,'System: ',A20)
      Write(10,70) sysname

 80   Format(1X,'component(',I2,')= ',A2)
 90   Format(1X,'component(',I2,')= ',A2,':(2s+1)= ',I2)
      do i=1,ncomp
         if (i.le.Nmag) then
            Write(10,90) i,component(i),nspin(i)
         else
```

```
          Write(10,80) i,component(i)
        endif
      enddo

      DimSys=Nel**4

 95   Format(1X,'giromagnetic ratio:')
 96   Format(1X,'g(',A2,')= ',F6.4)

      if (Nmag.gt.0) then
         Write(10,95)
         do i=1,Nmag
            Write(10,96) component(i),g(i)
         enddo
      endif

100   Format(A3,I2/A3,I2)
      Read(5,100) structJ,Jph,structI,Iph

      if ((structJ.eq.'bcc').or.(structJ.eq.'Bcc')) then
         structJ=BCC
      endif
      if ((structJ.eq.'fcc').or.(structJ.eq.'Fcc')) then
         structJ=FCC
      endif
      if ((structI.eq.'bcc').or.(structI.eq.'Bcc')) then
         structI=BCC
      endif
      if ((structI.eq.'fcc').or.(structI.eq.'Fcc')) then
         structI=FCC
      endif


      nph(1)=Jph
      struct(1)=structJ

      nph(2)=Iph
      struct(2)=structI

110   Format(1X,'Phase1: structure',1X,A3,I2,4X,',',
     *     'Phase2: structure',1X,A3,I2)

      Write(10,110) StructJ,Jph,StructI,Iph
      Write(12,110) StructJ,Jph,StructI,Iph
      Write(14,110) StructJ,Jph,StructI,Iph

      state_f=(Iph.eq.Jph).and.(structI.eq.structJ)

120   Format(28X,A1)

      Read(5,120) print_enm

C-----------------------------------------------------------------C
C     pr_enm='Y'->print complete energy energy matrix
C-----------------------------------------------------------------C

      if(print_enm.eq.'y') then
         print_enm='Y'
      endif

130   Format(34X,A1)

      Read(5,130) use_abs_G

C-----------------------------------------------------------------C
C     use_abs_G= 'Y'-> uses the absolute reference states.
C     The procedure is automatic in two phase equilibria when
C     the two phases have different crystal structures
C-----------------------------------------------------------------C

      if(use_abs_G.eq.'y') then
         use_abs_G='Y'
      endif

C-----------------------------------------------------------------C
C     Reading the eigenenergy matrix
C-----------------------------------------------------------------C

      if (structI.eq.structJ) then
         if (structJ.eq.BCC) then
            Call energyb(print_enm)
         else
            Call energyf(print_enm)
         endif
         if (use_abs_G.eq.'Y') then
            call refgp(structI,ref_stI)
            if (Iph.ne.Jph) then
               do i=1,ncomp
                  do j=1,8
                     ref_stJ(i,j)=ref_stI(i,j)
                  enddo
               enddo
            endif
         endif
      else
         Call energyb(print_enm)
         Call energyf(print_enm)
         Call refgp(StructI,ref_stI)
         Call refgp(StructJ,ref_stJ)
      endif

C-----------------------------------------------------------------C
C     Reading initial set of parameters
C
C     CPX1(i):Chemical potential of component i
C     at the first point (correspondent to phase 1)
C
C     CPX2(i):Chemical potential of component i
C     at the second point (correspondent to phase 2)
C
C     H_ext1:External magnetic field at the first point
C
C     H_ext2:External magnetic field at the second point
C
C     CP1(j):Generalized potential field of specie j
C     (component=elmt,spin number=spin) at the first point
C
C     CP2(j):idem at the second point
C-----------------------------------------------------------------C

 140  Format(10X,E13.7,11X,E13.7)
 150  Format(1X,'CPX1(',A2,')= ',E13.7,1X,'CPX2(',A2,')= ',E13.7)

 155  Format('Reading chemical potentials')

      CPX1(ncomp)=0.0
      CPX2(ncomp)=0.0

      if (ncomp.ge.2) then
         do i=1,(ncomp-1)
            Write(*,155)
            Read(5,140) CPX1(i),CPX2(i)
            Write(10,150) component(i),CPX1(i),
     *           component(i),CPX2(i)
            CPX1(ncomp)=CPX1(ncomp)-CPX1(i)
            CPX2(ncomp)=CPX2(ncomp)-CPX2(i)
         enddo
         Write(10,150) component(ncomp),CPX1(ncomp),
     *        component(ncomp),CPX2(ncomp)
      endif

160   Format(2X,'H_ext1= ',E12.7,3X,'H_ext2= ',E12.7)

170   Format('Reading magnetic fields')

      if (Nmag.gt.0) then
         Write(*,170)
         Read(5,140) H_ext1,H_ext2
         Write(10,160) H_ext1,H_ext2
      endif

      distance = 0.0

      do i=1,ncomp-1
         distance = distance + (CPX2(i)-CPX1(i))**2
      enddo

      distance = distance**(0.5)

      do i=1,ncomp
         if (distance.ne.(0.0)) then
            unit_vector(i)=(CPX2(i)-CPX1(i))/distance
         else
            unit_vector(i)=0.0e+0
         endif
      enddo


C-----------------------------------------------------------------C
C     orthogonal unit vector to r_CP
C     implemeted only for binary and ternary systems
C-----------------------------------------------------------------C

c      if (ncomp.eq.2) then
c         ortho_vector(2)=unit_vector(2)
c         ortho_vector(1)=unit_vector(1)
c      else
c         if (ncomp.eq.3) then
c            ortho_vector(2)=unit_vector(3)
c            ortho_vector(3)=-unit_vector(2)
c            ortho_vector(1)= -(ortho_vector(2)+ortho_vector(3))
c         endif
c      endif


      do i=1,Nel
         if (Nmag.gt.0) then
            call indexing(i,elmt,spin(1))
            CP1(i)=(CPX1(elmt)/nspin(elmt))
     *           +(g(elmt)*m_B*spin(1)*H_ext1
     *           *(nspin(elmt)-1.0)/(2.0*k_B))
            CP2(i)=(CPX2(elmt)/nspin(elmt))
     *           +(g(elmt)*m_B*spin(1)*H_ext2
     *           *(nspin(elmt)-1.0)/(2.0*k_B))
         else
            CP1(i)=CPX1(i)
            CP2(i)=CPX2(i)
         endif
         r_CP(i)=CP2(i)-CP1(i)
         CP(i)=CP1(i)
      enddo


C-----------------------------------------------------------------C
C     'Thermal history' of the calculation:
C
C     TK= temperature in K
C     delCP = translation of the chemical potential vectors
C     delTK = temperature change between steps
C     dfCP= initial value of df1
C
C-----------------------------------------------------------------C

 180  Format(3F10.4)
 185  Format(F10.4)

      Write(*,190)
      Read(5,180) TK,delTK,dfCP

      delCPX(ncomp)=0.0d+0
      do i=1,ncomp-1
         Read(5,185) delCPX(i)
         delCPX(ncomp)=delCPX(ncomp)-delCPX(i)
      enddo

      if (Nmag.gt.0) then
         elmt=0
         do i=1,ncomp
            do j=1,nspin(i)
               elmt=elmt+1
               delCP(elmt)=delCPX(i)/nspin(i)
            enddo
         enddo
      else
         do i=1,Nel
            delCP(i)=delCPX(i)
         enddo
      endif

 190  Format('Reading thermal history')

      Write(10,210) TK,delTK,dfCP
      do i=1,ncomp
         Write(10,215) component(i),delCPX(i)
      enddo

 210  Format(/1X,'TK= ',F8.2,1X,'delTK= ',E11.4,1X,'dfCP= ',E11.4)
 215  Format(1X,'delCP(',A2,')= ',E11.4)

C-----------------------------------------------------------------C
C     Control variables:
C
```

```
C     ibryf= number phase boundary points/one-phase equilibria
C
C     iversion= version of the search algorithm for the
C     phase boudary points
C
C     useclap= uses clapeyron relation to get a phase
C     boundary point in a different temperature
C
C     useread= read the initial configuration from file
C     bint.inp
C
C     constrained= calculates a constrained equilibrium
C     (for example, an isopleth)
C
C-----------------------------------------------------------------C

 220  Format('Reading control variables')

      Write(*,220)
 230  Format(I4,I2,3L2)

      Read(5,230) ibryf,iversion,useclap,useread,constrained

 240  Format(1X,'ibryf= ',I4,1X,'iversion= ',I2,1X,
     *     'useclap= ',L2,1X,'useread= ',L2,1X,
     *     'constrained= ',L2)

      Write(10,240) ibryf,iversion,useclap,useread,constrained

 245  Format(1X,'Sorry, this choice does not make sense')
 246  Format(1X,'set constrained = f in the control line')

      if (constrained.and.(ncomp.eq.1)) then
         Write(10,245)
         Write(10,246)
         Write(*,245)
         Write(*,246)
         goto 5000
      endif

 247  Format(1X,'Number of degrees of freedon is greater than zero')
 248  Format(1X,'set constrained = t in the control line')

      if (.not.constrained.and.(ncomp.gt.3)) then
         Write(10,247)
         Write(10,248)
         Write(*,247)
         Write(*,248)
         Goto 5000
      endif


C-----------------------------------------------------------------C
C     Reading initial configurations:
C
C     if useread is true the configurations are read from
C     a random access file (bint.inp), if not, they are
C     calculated as the product of the point probabilities
C     read in unit 10
C-----------------------------------------------------------------C

 250  Format(4F7.5)
 255  Format(1X,'inconsistency: Point probability <= 0')
 256  Format(1X,'Reading initial configurations')

      Write(*,256)

      if (useread) then
         Open(Unit=8,File='bint.inp',access='direct',
     *        recl=DimSys,status='old',form='unformatted')
         icount=icount+1
         nrec=0
         do i=1,Nel
            do j=1,Nel
               do k=1,Nel
                  do l=1,Nel
                     nrec=nrec+1
                     Read(8,rec=nrec) Zp(1,i,j,k,l),
     *                    Zp(2,i,j,k,l)
                  enddo
               enddo
            enddo
         enddo
         Close(Unit=8)
      else
         xal(1,Nel)=1.000
         xbt(1,Nel)=1.000
         xgm(1,Nel)=1.000
         xdt(1,Nel)=1.000
         xal(2,Nel)=1.000
         xbt(2,Nel)=1.000
         xgm(2,Nel)=1.000
         xdt(2,Nel)=1.000
         do i=1,Nel-1
            Read(5,250) xal(1,i),xbt(1,i),
     *           xgm(1,i),xdt(1,i)
            xal(1,Nel)=xal(1,Nel)-xal(1,i)
            xbt(1,Nel)=xbt(1,Nel)-xbt(1,i)
            xgm(1,Nel)=xgm(1,Nel)-xgm(1,i)
            xdt(1,Nel)=xdt(1,Nel)-xdt(1,i)
         enddo
         if ((xal(1,Nel).le.(0.0)).or.
     *       (xbt(1,Nel).le.(0.0)).or.
     *       (xgm(1,Nel).le.(0.0)).or.
     *       (xdt(1,Nel).le.(0.0))) then
            Write(10,255)
            Write(*,255)
            Goto 5000
         endif
         if (.not.state_f) then
            do i=1,Nel-1
               Read(5,250) xal(2,i),xbt(2,i),
     *              xgm(2,i),xdt(2,i)
               xal(2,Nel)=xal(2,Nel)-xal(2,i)
               xbt(2,Nel)=xbt(2,Nel)-xbt(2,i)
               xgm(2,Nel)=xgm(2,Nel)-xgm(2,i)
               xdt(2,Nel)=xdt(2,Nel)-xdt(2,i)
            enddo
            if ((xal(2,Nel).le.(0.0)).or.
     *          (xbt(2,Nel).le.(0.0)).or.
     *          (xgm(2,Nel).le.(0.0)).or.
     *          (xdt(2,Nel).le.(0.0))) then
               Write(10,255)
               Write(*,255)
               Goto 5000
            endif
         endif
         do i=1,Nel
            do j=1,Nel
               do k=1,Nel
                  do l=1,Nel
                     Zp(1,i,j,k,l)=xal(1,i)*xbt(1,j)
     *                    *xgm(1,k)*xdt(1,l)
                     if (state_f) then
                        Zp(2,i,j,k,l)=Zp(1,i,j,k,l)
                     else
                        Zp(2,i,j,k,l)=xal(2,i)*xbt(2,j)
     *                       *xgm(2,k)*xdt(2,l)
                     endif
                  enddo
               enddo
            enddo
         enddo
      endif

      Close(Unit=5)

 259  Format(/1X,'Phase= ',A3,I2)
 261  Format(1X,'xal(',I2,') = ',F6.4,1X,'xbt(',I2,')= ',F6.4,1X,
     *     'xgm(',I2,')= ',F6.4,1X,'xdt(',I2,')= ',F6.4)

      Write(10,259) structJ,Jph
      do i=1,Nel
         Write(10,261) i,xal(1,i),i,xbt(1,i),i,xgm(1,i),
     *        i,xdt(1,i)
      enddo


      if (.not.state_f) then
         Write(10,259) structI,Iph
         do i=1,Nel
            Write(10,261) i,xal(2,i),i,xbt(2,i),i,xgm(2,i),
     *           i,xdt(2,i)
         enddo
      endif


C-----------------------------------------------------------------C
C     Constrained calculations:
C
C     Mode='x'-> isopleth calculation
C     Mode='m'-> constant potential calculation
C     num_c= number of constraints
C     c_comp= compound to which the constraint should be applied
C     constr= value of the constraint
C-----------------------------------------------------------------C
      deg_f = 0

      if(constrained) then
         Open(Unit=5,file='constr.inp',status='old',
     *        form='formatted')
         Read(5,270) Mode
         Write(10,280) Mode
         Read(5,290) num_c
         if (Nmag.ne.0) then
            deg_f = ncomp-num_c-3
         else
            deg_f = ncomp-num_c-2
         endif
         Write(10,295) num_c
         if(num_c.ge.ncomp) then
            Write(10,245)
            Write(*,245)
            Goto 5000
         endif
         do i=1,num_c
            Read(5,310) c_comp(i),constr(i)
            Write(10,320) Mode,component(c_comp(i)),
     *           constr(i)
         enddo
         Close(Unit=5)
      endif

 270  Format(A2)
 280  Format(1X,'Constrained calculation: Mode= ',A2)
 290  Format(I2)
 295  Format(1X,'Number of constraints= ',I2)
 310  Format(I2,F6.4)
 320  Format(1X,'contraint of ',A2,'(',A2,')',' = ',F6.4)


C*****************************************************************C
C     Search Algorithm start
C*****************************************************************C

 260  Format('Algorithm start')
      Write(*,260)

C-----------------------------------------------------------------C
C     Initializing variables and counters
C-----------------------------------------------------------------C

 265  Format(/'iteration= ',I4/'TK= ',F11.2)
 275  Format('CP: ',17(E12.5,','))
 276  Format('dCP:',17(E12.5,','))
 277  Format('Tr: ',17(E12.5,','))

C-----------------------------------------------------------------
C     mju(i) is the 'true' chemical potential of i
C
C     acr(i) is the activity of component i in the reference
C     state TK and StructJ
C-----------------------------------------------------------------

      do i=1,NN
         mju(i)=0.0d+0
         acr(i)=0.0d+0
      enddo


      do ibry=1,ibryf

         RTK=1.0d+0/TK
C-----------------------------------------
C     Introduce here the actualization of the
C     reference state
C-----------------------------------------

         do i=1,Nel
            if (Nmag.gt.0) then
               call indexing(i,elmt,spin(1))
               CP1(i)=(CPX1(elmt)/nspin(elmt))
```

```
     *              +(g(elmt)*m_B*spin(1)*H_ext1
     *              *(nspin(elmt)-1.0)/(2.0*k_B))
            CP2(i)=(CPX2(elmt)/nspin(elmt))
     *              +(g(elmt)*m_B*spin(1)*H_ext2
     *              *(nspin(elmt)-1.0)/(2.0*k_B))
          else
             CP1(i)=CPX1(i)
             CP2(i)=CPX2(i)
          endif
          r_CP(i)=CP2(i)-CP1(i)
          CP(i)=CP1(i)
        enddo

        mag(1)=0.0
        mag(2)=0.0

        do i=1,Nel
           xal(1,i)=0.0
           xbt(1,i)=0.0
           xgm(1,i)=0.0
           xdt(1,i)=0.0
           xal(2,i)=0.0
           xbt(2,i)=0.0
           xgm(2,i)=0.0
           xdt(2,i)=0.0
           c_al(1,i)=0.0
           c_al(2,i)=0.0
           c_bt(1,i)=0.0
           c_bt(2,i)=0.0
           c_gm(1,i)=0.0
           c_gm(2,i)=0.0
           c_dt(1,i)=0.0
           c_dt(2,i)=0.0
           m_al(1,i)=0.0
           m_al(2,i)=0.0
           m_bt(1,i)=0.0
           m_bt(2,i)=0.0
           m_gm(1,i)=0.0
           m_gm(2,i)=0.0
           m_dt(1,i)=0.0
           m_dt(2,i)=0.0
        enddo

        dGP = 1.0
        f1 = 0.0
        df1 = dfCP

C-------------------------------------------------------------------C
C     initializing the flags for the search of the phase boundary
C-------------------------------------------------------------------C

        ifp = 0                         !iversion = 1 (Kikuchi)
        ifn = 0

        inp = 0                         !iversion = 2 (Colinet)
        f1p = 0.0
        f1n = 0.0

        icount = 0

        Write(*,265) ibry,TK
        Write(*,277) (CP(i),i=1,Nel)

        do 2000 while ((dGP.gt.dGPf).and.(icount.lt.icountf))

           icount=icount+1
           do i=1,Nel
              do j=1,Nel
                 do k=1,Nel
                    do l=1,Nel
                       ecp(i,j,k,l)=CP(i)+CP(j)+CP(k)+CP(l)
                       epcb(i,j,k,l)= (-epsb(i,j,k,l)
     *                      +ecp(i,j,k,l)/24.0d+0)*RTK
                       epcf(i,j,k,l)= (-epsf(i,j,k,l)
     *                      +ecp(i,j,k,l)/8.0d+0)*RTK
                    enddo
                 enddo
              enddo
           enddo

C-------------------------------------------------------------------C
C     Realizing one equilibrium calculation
C-------------------------------------------------------------------C

           if (.not.state_f) then
              if(Jph.eq.1) then
                 Iphase=Jph
                 do i=1,Nel
                    do j=1,Nel
                       do k=1,Nel
                          do l=1,Nel
                             ZD(i,j,k,l)=Zp(1,i,j,k,l)
                          enddo
                       enddo
                    enddo
                 enddo
                 if(structJ.eq.BCC) then
                    call DISBCC(itrD,GPD,FgyD,
     *                   EtpyD,EgyD)
                 else
                    if(structJ.eq.FCC) then
                       call DISFCC(itrD,GPD,FgyD,
     *                      EtpyD,EgyD)
                    endif
                 endif
                 itr(1)=itrD
                 GP(1)=k_B*GPD
                 Fgy(1)=k_B*FgyD
                 Etpy(1)=k_B*EtpyD
                 Egy(1)=k_B*EgyD
                 do i=1,Nel
                    do j=1,Nel
                       do k=1,Nel
                          do l=1,Nel
                             Zp(1,i,j,k,l)=ZD(i,j,k,l)
                          enddo
                       enddo
                    enddo
                 enddo
              else
                 Iphase=Jph
                 do i=1,Nel
                    do j=1,Nel
                       do k=1,Nel
                          do l=1,Nel
                             ZO(i,j,k,l)=Zp(1,i,j,k,l)
```

```
                          enddo
                       enddo
                    enddo
                 enddo
                 if(structJ.eq.BCC) then
                    call ORDBCC(Iphase,itrO,GPO,FgyO,
     *                   EtpyO,EgyO)
                 else
                    if(structJ.eq.FCC) then
                       call ORDFCC(Iphase,itrO,GPO,FgyO,
     *                      EtpyO,EgyO)
                    endif
                 endif
                 itr(1)=itrO
                 GP(1)=k_B*GPO
                 Fgy(1)=k_B*FgyO
                 Etpy(1)=k_B*EtpyO
                 Egy(1)=k_B*EgyO
                 do i=1,Nel
                    do j=1,Nel
                       do k=1,Nel
                          do l=1,Nel
                             Zp(1,i,j,k,l)=ZO(i,j,k,l)
                          enddo
                       enddo
                    enddo
                 enddo

              endif
           endif
           if(Iph.eq.1) then
              Iphase=Iph
              do i=1,Nel
                 do j=1,Nel
                    do k=1,Nel
                       do l=1,Nel
                          ZD(i,j,k,l)=Zp(2,i,j,k,l)
                       enddo
                    enddo
                 enddo
              enddo
              if(structI.eq.BCC) then
                 call DISBCC(itrD,GPD,FgyD,
     *                EtpyD,EgyD)
              else
                 if(structI.eq.FCC) then
                    call DISFCC(itrD,GPD,FgyD,
     *                   EtpyD,EgyD)
                 endif
              endif
              itr(2)=itrD
              GP(2)=k_B*GPD
              Fgy(2)=k_B*FgyD
              Etpy(2)=k_B*EtpyD
              Egy(2)=k_B*EgyD
              do i=1,Nel
                 do j=1,Nel
                    do k=1,Nel
                       do l=1,Nel
                          Zp(2,i,j,k,l)=ZD(i,j,k,l)
                          if (state_f) then
                             Zp(1,i,j,k,l)=ZD(i,j,k,l)
                          endif
                       enddo
                    enddo
                 enddo
              enddo
              if(state_f) then
                 itr(1)=itr(2)
                 GP(1)=GP(2)
                 Fgy(1)=Fgy(2)
                 Etpy(1)=Etpy(2)
                 Egy(1)=Egy(2)
              endif
           else
              Iphase=Iph
              do i=1,Nel
                 do j=1,Nel
                    do k=1,Nel
                       do l=1,Nel
                          ZO(i,j,k,l)=Zp(2,i,j,k,l)
                       enddo
                    enddo
                 enddo
              enddo
              if(StructI.eq.BCC) then
                 call ORDBCC(Iphase,itrO,GPO,FgyO,
     *                EtpyO,EgyO)
              else
                 if(StructI.eq.FCC) then
                    call ORDFCC(Iphase,itrO,GPO,FgyO,
     *                   EtpyO,EgyO)
                 endif
              endif
              itr(2)=itrO
              GP(2)=k_B*GPO
              Fgy(2)=k_B*FgyO
              Etpy(2)=k_B*EtpyO
              Egy(2)=k_B*EgyO
              do i=1,Nel
                 do j=1,Nel
                    do k=1,Nel
                       do l=1,Nel
                          Zp(2,i,j,k,l)=ZO(i,j,k,l)
                          if (state_f) then
                             Zp(1,i,j,k,l)=ZO(i,j,k,l)
                          endif
                       enddo
                    enddo
                 enddo
              enddo

              if(state_f) then
                 itr(1)=itr(2)
                 GP(1)=GP(2)
                 Fgy(1)=Fgy(2)
                 Etpy(1)=Etpy(2)
                 Egy(1)=Egy(2)
              endif
           endif

C-------------------------------------------------------------------C
C     Searching for the phase boundary in iteration ibry
C-------------------------------------------------------------------C

           dGP=GP(2)-GP(1)
```

```
            if(.not.state_f) then

               if((dGP.lt.0).and.(icount.eq.1)) then
                  Write(10,311)
                  Write(*,311)
                  Goto 5000
               endif

               if(icount.gt.icountf) then
                  Write(10,321)
                  Write(*,321)
                  Goto 5000
               endif

               if(abs(df1).lt.df1f) then
                  Write(10,330)
                  goto 5000
               endif

               if(iversion.eq.1) then !Kikuchi
                  if(dGP.lt.0) then
                     ifn=1
                     dGPn=dGP
                     fln=f1
                     if(ifp.eq.1) then
                        df1= (flp-fln)*dGPn/(dGPp-dGPn)
                     endif
                     f1 = f1 - df1
                  else
                     ifp=1
                     dGPp=dGP
                     flp=f1
                     if(ifn.eq.1) then
                        df1= (flp-fln)*dGPp/(dGPp-dGPn)
                     endif
                     f1=f1+df1
                  endif

               else              !iversion = 2 (Colinet)

                  if(icount.eq.2.and.dGP.lt.0) then
                     inp = 1
                  endif

                  if (inp.eq.1) then
                     if (dGP.lt.0) then
                        f1=f1 - df1
                     else
                        inp = 0
                        df1 = df1/2.0
                        f1 = f1 + df1
                     endif
                  endif

                  if (dGP.lt.0) then
                     f1 = f1 - df1
                     df1 = df1/4.0
                     f1 = f1 + df1
                  else
                     if (inp.eq.1) then
                        df1 = df1/2.0
                     endif
                     f1 = f1 + df1
                  endif
               endif
               do i=1,Nel
                  CP(i)=CP(i)+(f1*r_CP(i))
               enddo

            endif

 2000    enddo


         Write(*,275) (CP(i),i=1,Nel)

C-----------------------------------------------------------------C
C     One iteration is ready
C     storing the cluster probabilities in file bintl.out
C-----------------------------------------------------------------C


         Open(Unit=8,File='bintl.out',access='direct',
     *        recl=DimSys,status='unknown',form='unformatted')
         icount=icount+1
         nrec=0
         do i=1,Nel
            do j=1,Nel
               do k=1,Nel
                  do l=1,Nel
                     nrec=nrec+1
                     Write(8,rec=nrec) Zp(1,i,j,k,l),
     *                    Zp(2,i,j,k,l)
                  enddo
               enddo
            enddo
         enddo
         Close(Unit=8)

C-----------------------------------------------------------------C
C     Calculating point probabilities and subl. magnetizations
C-----------------------------------------------------------------C

         do i=1,Nel
            do j=1,Nel
               do k=1,Nel
                  do l=1,Nel
                     xal(1,i)=xal(1,i)+Zp(1,i,j,k,l)
                     xal(2,i)=xal(2,i)+Zp(2,i,j,k,l)
                     xbt(1,i)=xbt(1,i)+Zp(1,j,i,k,l)
                     xbt(2,i)=xbt(2,i)+Zp(2,j,i,k,l)
                     xgm(1,i)=xgm(1,i)+Zp(1,k,j,i,l)
                     xgm(2,i)=xgm(2,i)+Zp(2,k,j,i,l)
                     xdt(1,i)=xdt(1,i)+Zp(1,l,j,k,i)
                     xdt(2,i)=xdt(2,i)+Zp(2,l,j,k,i)
                     if (Nmag.gt.0) then
                        call indexing(i,i2,si)
                        m_al(1,i2)=m_al(1,i2)+(si*Zp(1,i,j,k,l))
                        m_al(2,i2)=m_al(2,i2)+(si*Zp(2,i,j,k,l))
                        m_bt(1,i2)=m_bt(1,i2)+(si*Zp(1,j,i,k,l))
                        m_bt(2,i2)=m_bt(2,i2)+(si*Zp(2,j,i,k,l))
                        m_gm(1,i2)=m_gm(1,i2)+(si*Zp(1,k,j,i,l))
                        m_gm(2,i2)=m_gm(2,i2)+(si*Zp(2,k,j,i,l))
                        m_dt(1,i2)=m_dt(1,i2)+(si*Zp(1,l,j,k,i))
                        m_dt(2,i2)=m_dt(2,i2)+(si*Zp(2,l,j,k,i))
                     else
                        i2=i
                     endif
                     c_al(1,i2)=c_al(1,i2)+Zp(1,i,j,k,l)
                     c_al(2,i2)=c_al(2,i2)+Zp(2,i,j,k,l)
                     c_bt(1,i2)=c_bt(1,i2)+Zp(1,j,i,k,l)
                     c_bt(2,i2)=c_bt(2,i2)+Zp(2,j,i,k,l)
                     c_gm(1,i2)=c_gm(1,i2)+Zp(1,k,j,i,l)
                     c_gm(2,i2)=c_gm(2,i2)+Zp(2,k,j,i,l)
                     c_dt(1,i2)=c_dt(1,i2)+Zp(1,l,j,k,i)
                     c_dt(2,i2)=c_dt(2,i2)+Zp(2,l,j,k,i)
                  enddo
               enddo
            enddo
         enddo

         do i=1,ncomp
            p1(1,i)=(c_al(1,i)+c_bt(1,i)+c_gm(1,i)+c_dt(1,i))/4.0
            p1(2,i)=(c_al(2,i)+c_bt(2,i)+c_gm(2,i)+c_dt(2,i))/4.0
            if (Nmag.gt.0) then
               mag(1)=mag(1)+((g(i)*(nspin(i)-1.0d+0)/2.0d+0)*
     *              (m_al(1,i)+m_bt(1,i)+
     *              m_gm(1,i)+m_dt(1,i))/4.0)
               mag(2)=mag(2)+((g(i)*(nspin(i)-1.0d+0)/2.0d+0)*
     *              (m_al(2,i)+m_bt(2,i)+
     *              m_gm(2,i)+m_dt(2,i))/4.0)
            endif
         enddo

C-----------------------------------------------------------------C
C     Calculating the APB surface tensions for
C     BCC-based superlattices
C
C     DE_apb(i,j): Energy change per atom caused by introducing an
C     APB of Burgers vector 'i' in the plane 'j' not allowing
C     chemical relaxation (mechanical APB)
C
C     s_apb(i,j): surface tension (times a0**2, where a0 is the
C     lattice parameter of the disordered BCC crystal) of an
C     APB of Burgers vector 'i' in the plane 'j'
C
C     i = 1 => a0<1,0,0>
C     i = 2 => (a0/2)<1,1,1>
C
C     j = 1 => (0,0,1)
C     j = 2 => (0,1,-1)
C-----------------------------------------------------------------C

         if(state_f.and.(StructJ.eq.BCC)) then

            do i=1,Nel
               do j=1,Nel
                  do k=1,Nel
                     do l=1,Nel
                        pABGD(i,j,k,l)=0.0d+0
                     enddo
                     pABG(i,j,k)=0.0d+0
                     pABD(i,j,k)=0.0d+0
                     pAGD(i,j,k)=0.0d+0
                     pBGD(i,j,k)=0.0d+0
                  enddo
                  pAB(i,j)=0.0d+0
                  pAG(i,j)=0.0d+0
                  pAD(i,j)=0.0d+0
                  pBG(i,j)=0.0d+0
                  pBD(i,j)=0.0d+0
                  pGD(i,j)=0.0d+0
               enddo
               pA(i)=0.0d+0
               pB(i)=0.0d+0
               pG(i)=0.0d+0
               pD(i)=0.0d+0
            enddo

            do i=1,Nel
               do j=1,Nel
                  do k=1,Nel
                     do l=1,Nel
                        if(Nmag.gt.0) then
                           call indexing(i,i2,si)
                           call indexing(j,j2,sj)
                           call indexing(k,k2,sk)
                           call indexing(l,l2,sl)
                        else
                           i2=i
                           j2=j
                           k2=k
                           l2=l
                        endif
                        pABGD(i2,j2,k2,l2) =
     *                       pABGD(i2,j2,k2,l2) +
     *                       Zp(1,i,j,k,l)
                     enddo
                  enddo
               enddo
            enddo

            do i=1,ncomp
               do j=1,ncomp
                  do k=1,ncomp
                     do l=1,ncomp
                        pABG(i,j,k)=pABG(i,j,k)+pABGD(i,j,k,l)
                        pABD(i,j,k)=pABD(i,j,k)+pABGD(i,j,l,k)
                        pAGD(i,j,k)=pAGD(i,j,k)+pABGD(i,l,j,k)
                        pBGD(i,j,k)=pBGD(i,j,k)+pABGD(l,i,j,k)
                        pAB(i,j)=pAB(i,j)+pABGD(i,j,k,l)
                        pAG(i,j)=pAG(i,j)+pABGD(i,k,j,l)
                        pAD(i,j)=pAD(i,j)+pABGD(i,k,l,j)
                        pBG(i,j)=pBG(i,j)+pABGD(k,i,j,l)
                        pBD(i,j)=pBD(i,j)+pABGD(k,i,l,j)
                        pGD(i,j)=pGD(i,j)+pABGD(k,l,i,j)
                        pA(i)=pA(i)+pABGD(i,j,k,l)
                        pB(i)=pB(i)+pABGD(j,i,k,l)
                        pG(i)=pG(i)+pABGD(j,k,i,l)
                        pD(i)=pD(i)+pABGD(j,k,l,i)
                     enddo
                  enddo
               enddo
            enddo

            do i=1,2
               do j=1,2
                  DE_apb(i,j)=0.0d+0
                  s_apb(i,j)=0.0d+0
               enddo
            enddo
```

```
      do i=1,ncomp
         do j=1,ncomp
            do k=1,ncomp
               do l=1,ncomp
                  part1=0.0d+0
                  part2=0.0d+0
                  part3=0.0d+0
                  part4=0.0d+0
                  do m=1,ncomp
                     part1 = part1 +
     *                    (pABGD(i,j,k,m)*
     *                    ((pABGD(i,j,l,m)/pABD(i,j,m))
     *                    +(pAGD(i,l,m)/pAD(i,m))
     *                    +(pBGD(j,l,m)/pBD(j,m))
     *                    +(pGD(l,m)/pD(m))))
                     part2 = part2 +
     *                    (pABGD(i,j,m,l)*
     *                    ((pABGD(i,j,m,k)/pABG(i,j,m))
     *                    +(pAGD(i,m,k)/pAG(i,m))
     *                    +(pBGD(j,m,k)/pBG(j,m))
     *                    +(pGD(m,k)/pG(m))))
                     part3 = part3 +
     *                    (pABGD(j,m,l,k)*
     *                    ((pABGD(i,m,l,k)/pBGD(m,l,k))
     *                    +(pABG(i,m,l)/pBG(m,l))
     *                    +(pABD(i,m,k)/pBD(m,k))
     *                    +(pAB(i,m)/pB(m))))
                     part4 = part4 +
     *                    (pABGD(m,i,l,k)*
     *                    ((pABGD(m,j,l,k)/pAGD(m,l,k))
     *                    +(pABG(m,j,l)/pAG(m,l))
     *                    +(pABD(m,j,k)/pAD(m,k))
     *                    +(pAB(m,j)/pA(m))))
                  enddo
                  part1 = part1 - (4.0d+0*pABGD(i,j,k,l))
                  part2 = part2 - (4.0d+0*pABGD(i,j,k,l))
                  part3 = part3 - (4.0d+0*pABGD(i,j,k,l))
                  part4 = part4 - (4.0d+0*pABGD(i,j,k,l))
                  part5 = pGD(l,k)/(pG(l)*pD(k))*
     *                 ((pG(l)*pABD(i,j,k))
     *                 +(pD(k)*pABG(i,j,l)))
                  part5 = part5 - (2.0d+0*pABGD(i,j,k,l))
                  part6 = pAB(i,j)/(pA(i)*pB(j))*
     *                 ((pA(i)*pBGD(j,l,k))
     *                 +(pB(j)*pAGD(i,l,k)))
                  part6 = part6 - (2.0d+0*pABGD(i,j,k,l))
                  DE_apb(1,1) = DE_apb(1,1) +
     *                 ((epxb(i,j,k,l)*(part1 + part2 +
     *                 part3 + part4 + part5 + part6))/
     *                 4.0d+0)

                  part1 = (pA(i)*pABG(k,l,j)) +
     *                 (pB(j)*pABD(k,l,i)) +
     *                 (pA(k)*pABD(i,j,l)) +
     *                 (pB(l)*pABG(i,j,k)) +
     *                 (pAB(i,j)*pAB(k,l))
                  part1 = part1 - (5.0d+0*pABGD(i,j,k,l))
                  DE_apb(2,1)= DE_apb(2,1) +
     *                 (part1*epxb(i,j,k,l))

                  part1 = (
     *                 (pAD(i,k)*((pAD(i,l)*pAD(j,k)) +
     *                 (pABD(i,j,l)*pD(k)) +
     *                 (pAGD(j,l,k)*pA(i)))/(pA(i)*pD(k))) +
     *                 (pBG(j,l)*((pABG(i,j,k)*pG(l)) +
     *                 (pBG(j,k)*pBG(i,l)) +
     *                 (pBGD(i,l,k)*pB(j)))/(pB(j)*pG(l))) +
     *                 (pAG(j,k)*((pAGD(i,k,l)*pA(j)) +
     *                 (pAG(i,k)*pAG(j,l)) +
     *                 (pABG(j,i,l)*pG(k)))/(pA(j)*pG(k))) +
     *                 (pBD(i,l)*((pBGD(j,k,l)*pB(i)) +
     *                 (pBD(j,l)*pBD(i,k)) +
     *                 (pABD(j,i,k)*pD(l)))/(pB(i)*pD(l))))
                  part1 = part1 - (1.2d+1*pABGD(i,j,k,l))
                  DE_apb(1,2) = DE_apb(1,2) +
     *                 ((epxb(i,j,k,l)*part1)/2.0d+0)

                  part1 = (
     *                 (pAD(i,l)*pABD(i,k,j)/pA(i))+
     *                 (pBG(j,k)*pABG(l,j,i)/pB(j))+
     *                 (pAG(i,k)*pAGD(l,k,j)/pG(k))+
     *                 (pBD(j,l)*pBGD(k,i,l)/pD(l))+
     *                 (pBD(k,j)*pABD(i,k,l)/pB(k))+
     *                 (pAG(l,i)*pABG(l,j,k)/pA(l))+
     *                 (pAD(l,j)*pAGD(i,k,j)/pD(j))+
     *                 (pBG(k,i)*pBGD(j,i,l)/pG(i)))

                  part1 = part1 - (8.0d+0*pABGD(i,j,k,l))
                  part2 = 0.0d+0
                  part3 = 0.0d+0
                  do m=1,ncomp
                     part2 = part2 + (
     *                    (pABGD(i,j,m,l)
     *                    *pABG(i,k,m)/pAG(i,m))+
     *                    (pABGD(i,j,k,m)
     *                    *pABD(l,j,m)/pBD(j,m))+
     *                    (pABGD(i,m,k,l)
     *                    *pBGD(m,k,j)/pBG(m,k))+
     *                    (pABGD(m,j,k,l)
     *                    *pAGD(m,i,l)/pAD(m,l))+
     *                    (pABGD(l,k,m,j)
     *                    *pABG(i,k,m)/pBG(k,m))+
     *                    (pABGD(l,k,i,m)
     *                    *pABD(l,j,m)/pAD(l,m))+
     *                    (pABGD(l,m,i,j)
     *                    *pBGD(m,k,j)/pBD(m,j))+
     *                    (pABGD(m,k,i,j)
     *                    *pAGD(m,i,l)/pAG(m,i)))

                     part3 = part3 +
     *                    (pABGD(i,j,m,l)*
     *                    pABGD(i,k,m,l)/pAGD(i,m,l))+
     *                    (pABGD(i,j,k,m)*
     *                    pABGD(l,j,k,m)/pBGD(j,k,m))+
     *                    (pABGD(i,m,k,l)*
     *                    pABGD(i,m,k,j)/pABG(i,m,k))+
     *                    (pABGD(m,j,k,l)*
     *                    pABGD(m,j,i,l)/pABD(m,j,l))+
     *                    (pABGD(l,m,k,j)*
     *                    pABGD(l,m,i,j)/pABD(l,m,j))+
     *                    (pABGD(m,k,i,l)*
     *                    pABGD(m,k,i,j)/pABG(m,k,i))+
     *                    (pABGD(l,j,i,m)*
     *                    pABGD(l,k,i,m)/pAGD(l,i,m))+
     *                    (pABGD(i,k,m,j)*
     *                    pABGD(i,m,j)/pAGD(i,m,j))+
     *                    (pABGD(i,j,m,k)*
     *                    pABGD(l,j,m,k)/pBGD(j,m,k))+
     *                    (pABGD(i,j,k,m)*
     *                    pABGD(l,k,m,j)/pBGD(k,m,j))+
     *                    (pABGD(i,m,k,j)*
     *                    pABGD(i,m,l,k)/pABG(i,m,k))+
     *                    (pABGD(m,j,k,l)*
     *                    pABGD(m,j,i,l)/pABD(m,j,l))+
     *                    (pABGD(l,m,k,j)*
     *                    pABGD(l,m,i,j)/pABD(l,m,j))+
     *                    (pABGD(m,k,i,j)*
     *                    pABGD(m,k,i,l)/pABG(m,k,i))+
     *                    (pABGD(l,j,i,m)*
     *                    pABGD(l,k,i,m)/pAGD(l,i,m))+
     *                    (pABGD(i,k,m,j)*
     *                    pABGD(l,k,m,j)/pBGD(k,m,j))
                     enddo
                     part2 = part2 - (8.0d+0*pABGD(i,j,k,l))
                     part3 = part3 - (8.0d+0*pABGD(i,j,k,l))

                     DE_apb(2,2) = DE_apb(2,2) +
     *                    ((epxb(i,j,k,l)*(part1 + part2 +
     *                    part3))/4.0d+0)
                  enddo
               enddo
            enddo
         enddo
         s_apb(1,1)=DE_apb(1,1)
         s_apb(2,1)=DE_apb(2,1)
         s_apb(1,2)=DE_apb(1,2)*sqrt(2.0d+0)
         s_apb(2,2)=DE_apb(2,2)*sqrt(2.0d+0)

      endif
C-----------------------------------------------------------------C
C     Writing outputs
C-----------------------------------------------------------------C

      do i=1,Nel
         CP1(i)=CP(i)
         CP2(i)=CP(i)+r_CP(i)
      enddo

C      Write(*,277) (CP1(i),i=1,Nel)


      if (state_f) then
         Write(10,340)
         Write(10,370) TK,structI,Iph,itr(2),GP(2)
         Write(10,360)
         do i=1,ncomp
            mju(i)=GP(1)
            CPX1(i)=0.0d+0
            CPX2(i)=0.0d+0
         enddo
         if (Nmag.gt.0) then
            do i=1,Nel
               call indexing(i,i2,si)
               mju(i2)=mju(i2)+(k_B*CP1(i))
               CPX1(i2)=CPX1(i2)+CP1(i)
               CPX2(i2)=CPX2(i2)+CP2(i)
            enddo
         else
            do i=1,Nel
               mju(i)=mju(i)+(k_B*CP1(i))
               CPX1(i)=CP1(i)
               CPX2(i)=CP2(i)
            enddo
         endif
         do i=1,ncomp
            acr(i)=exp(mju(i)/(k_B*TK))
            if (Nmag.eq.0) then
               Write(10,365) component(i),CP(i),
     *              component(i),mju(i),
     *              component(i),acr(i)
            else
               Write(10,365) component(i),CPX1(i),
     *              component(i),mju(i),
     *              component(i),acr(i)
            endif
         enddo
         Write(10,420) structJ,Jph,Fgy(1),Egy(1),Etpy(1),mag(1)

         if(StructJ.eq.BCC) then
            Write(10,470)
            Write(10,475)
            Write(10,480)
            Write(10,490)
            Write(10,510) '(0,0,1)',s_apb(1,1),s_apb(2,1)
            Write(10,510) '(0,1,-1)',s_apb(1,2),s_apb(2,2)
            Write(10,520)
            Write(15,530) TK,',',s_apb(1,1),',',s_apb(1,2),',',
     *           s_apb(2,1),',',s_apb(2,2)
         endif

         Write(10,385)
         Write(10,388)
         do k=1,ncomp
            Write(10,390) component(k),pl(1,k),c_al(1,k),
     *           c_bt(1,k),c_gm(1,k),c_dt(1,k)
         enddo
         if (Nmag.ne.0) then
            Write(10,398)
            do k=1,Nmag
               Write(10,410) component(k),m_al(1,k),
     *              m_bt(1,k),m_gm(1,k),m_dt(1,k)
            enddo
         endif
         Write(10,385)
         Write(12,430) TK,',',(xal(1,i),',',
     *        xbt(1,i),',',xgm(1,i),',',xdt(1,i),',',i=1,Nel-1)
         Write(14,460) TK,',',GP(1),',',Egy(1),',',
     *        Etpy(1),',',(CP(i),',',pl(1,i),',',i=1,Nel)


c         do i=1,Nel
c            if(Nmag.eq.0) then
c               CP1(i)=CP1(i)+delCP(i)
c            else
c               call indexing(i,i2,si)
c               CP1(i)=CP1(i)+(delCP(i2)/nspin(i2))
c            endif
c         enddo

         do i=1,ncomp
            CPX1(i)=CPX1(i)+delCPX(i)
         enddo


      else
         Write(10,350)
         Write(10,380) TK,icount,structJ,Jph,itr(1),
     *        GP(1),dGP,structI,Iph,itr(2),GP(2),f1
         Write(10,360)

         do i=1,ncomp
            mju(i)=GP(2)
            CPX1(i)=0.0d+0
            CPX2(i)=0.0d+0
         enddo

         if (Nmag.gt.0) then
```

```
               do i=1,Nel
                  call indexing(i,i2,si)
                  mju(i2)=mju(i2)+(k_B*CP1(i))
                  CPX1(i2)=CPX1(i2)+CP1(i)
                  CPX2(i2)=CPX2(i2)+CP2(i)
               enddo
            else
               do i=1,Nel
                  mju(i)=mju(i)+(k_B*CP(i))
                  CPX1(i)=CP1(i)
                  CPX2(i)=CP2(i)
               enddo
            endif
            do i=1,ncomp
               acr(i)=exp(mju(i)/(k_B*TK))
               if (Nmag.eq.0) then
                  Write(10,365) component(i),CP(i),
     *                 component(i),mju(i),
     *                 component(i),acr(i)
               else
                  Write(10,365) component(i),CPX1(i),
     *                 component(i),mju(i),
     *                 component(i),acr(i)
               endif
            enddo
            do j=1,2
               Write(10,420) struct(j),nph(j),Fgy(j),Egy(j),
     *              Etpy(j),mag(j)
               Write(10,385)
               Write(10,388)
               do k=1,ncomp
                  Write(10,390) component(k),pl(j,k),
     *                 c_al(j,k),c_bt(j,k),c_gm(j,k),c_dt(j,k)
               enddo
               Write(10,385)
               if (Nmag.gt.0) then
                  Write(10,398)
                  do k=1,Nmag
                     Write(10,410) component(k),m_al(j,k),
     *                    m_bt(j,k),m_gm(j,k),m_dt(j,k)
                  enddo
               endif
               Write(10,385)
            enddo
            Write(12,440) TK,',',((xal(j,i),',',
     *           xbt(j,i),',',xgm(j,i),',',
     *           xdt(j,i),',',i=1,Nel-1),j=1,2)

            Write(14,450) TK,',',(CP(j),',',Fgy(j),',',Egy(j),',',
     *           Etpy(j),',',j=1,2),(CP(i),',',i=1,Nel-1)
         endif

C-----------------------------------------------------------------------
C
C     Changing external parameters (TK,CP...)
C-----------------------------------------------------------------------
C

C-----------------------------------------------------------------------C
C     Changes the chemical potentials
C-----------------------------------------------------------------------C

         if(.not.state_f) then
            if(Nmag.eq.0) then
               do i=1,Nel
                  CP1(i)=CP1(i)+delCP(i)
                  CP2(i)=CP2(i)+delCP(i)
                  r_CP(i)=CP2(i)-CP1(i)
                  CP(i)=CP1(i)
                  CPX1(i)=CP1(i)
                  CPX2(i)=CP2(i)
               enddo
            else
               do i2=1,ncomp
                  CPX1(i2)=0.0d+0
                  CPX2(i2)=0.0d+0
               enddo
               do i=1,Nel
                  call indexing(i,i2,si)
                  CP1(i)=CP1(i)+delCP(i)
                  CP2(i)=CP2(i)+delCP(i)
                  CP(i)=CP1(i)
                  r_CP(i)=CP2(i)-CP1(i)
                  CPX1(i2)=CPX1(i2)+CP1(i)
                  CPX2(i2)=CPX2(i2)+CP2(i)
               enddo
C               Write(*,277) (CP(i),i=1,Nel)
            endif
         endif

         Write(*,276) (r_CP(i),i=1,Nel)

         TK=TK+delTK
         RTK=1.0d+0/TK

      enddo

 311     Format(/1X,'Iph-phase is more stable, choose diff. CPs')

 321     Format(1X,'icount > icountf: No phase boundary')

 330     Format(1X,'df1 < df1f: No phase boundary')

 340  Format(/1X,15(1H-),'Grand Potential calculation',26(1H-))

 350  Format(/1X,15(1H*),'PHASE BOUNDARY CALCULATION',27(1H*))

 360  Format(/1X,'CP [K/at]:                mju [J/mol]:',
     *     '                activity:')
 365  Format(1X,A2,' -> ',E12.5,7X,A2,'  -> ',E12.5,1X,
     *     7X,A2,' -> ',F12.10)

 370  Format(/1X,'TK= ',F8.2,1X,'Iph= ',A3,I2,1X,'itr= ',I5,
     *     5X,'GP= ',E12.6)

 380  Format(/1X,'TK= ',F8.2,1X,'icount= ',I4/
     *     /1X,'Jph= ',A3,I2,1X,'itr1= ',I5,5X,
     *     'GP1= ',E12.6,3X,'dGP= ',E10.3,
     *     /1X,'Iph= ',A3,I2,1X,'itr2= ',I5,5X,
     *     'GP2= ',E12.6,4X,'f1= ',E10.3)

 385  Format(1X,59(1H-))

 388  Format(1X,'|',2X,'Elmt',1X,'|',3X,'pl',3X,'|',3X,'c_al',2X,'|',
     *     3X,'c_bt',2X,'|',3X,'c_gm',2X,'|',3X,'c_dt',2X,'|')

 390  Format(1X,'|',3X,A2,2X,'|',1X,F6.4,1X,'|',1X,F7.5,1X,'|',
     *     1X,F7.5,1X,'|',1X,F7.5,1X,'|',1X,F7.5,1X,'|')

 398  Format(1X,'|',2X,'Elmt',10X,'|',3X,'m_al',2X,'|',
     *     3X,'m_bt',2X,'|',3X,'m_gm',2X,'|',3X,'m_dt',2X,'|')

 410  Format(1X,'|',3X,A2,11X,'|',1X,F7.5,1X,'|',
     *     1X,F7.5,1X,'|',1X,F7.5,1X,'|',1X,F7.5,1X,'|')

 420  Format(/1X,'Phase= ',A3,I2,1X,'Fgy= ',E12.6,' J/mol',/1X,
     *     'Egy= ',E12.6,' J/mol',5X,'Etpy= ',F8.5,' J/mol.K'
     *     ,5X,'mag= ',F8.4,' m_B'/)

 430  Format(F6.0,A1,(64(F5.4,A1)))

 440  Format(F6.0,A1,(40(F5.4,A1)))

 450  Format(F6.0,A1,(8(E14.8,A1)),(5(E10.4,A1)))

 460  Format(F6.0,A1,(3(E14.8,A1)),(16(E10.4,A1,E12.7,A1)))

 470  Format(/1X,'################APB energies################')
 475  Format(1X,'|          sigma*(a0**2) in [K]              |')

 480  Format(1X,'| Vector ->  |  a0<1,0,0>    a0/2<1,1,1> |')

 490  Format(1X,'|   Plane    |                             |')

 510  Format(1X,'| ',A9,' | ',E11.5,3X,E11.5,' |')
 520  Format(1X,'#############################################')
 530  Format(F6.0,A1,4(E14.8,A1))

 540  Format(E20.14)

 5000 continue
      close(Unit=10)
      close(unit=12)
      close(unit=14)
      close(unit=15)
      stop
      end

C*************************************************************C
      subroutine energyb(pr_enm)
C*************************************************************C

      Parameter (NN=17)

C-------------------------------------------------------------C
C     Based on a similar subroutine in program bfcc.for
C     (C. Colinet)
C     Adapted by Claudio G. Schoen in the actual form
C     (July 17th 1997)
C
C     For BCC phases in the irregular tetrahedron cluster
C     approximation:
C
C     Reads interaction parameters in unit 10 and calculates
C     the eigenenergy matrix (epsilon) for the system
C-------------------------------------------------------------C


      Common/Csys/Nel,Nmag,ncomp,nspin,component
      Common/CBCC/epsb,epcb,epxb

      integer(kind=2) i,j,k,l
      integer(kind=2) Nel,Nmag,ncomp,nspin
      integer(kind=2) ni
      integer(kind=2) nj,nk,nl

      dimension nspin(NN)

      real(kind=2) epsb,epcb,epxb
      real(kind=2) wlc,w2c,wlm,w2m
      real(kind=2) etetra
      real(kind=2) si,sj,sk,sl

      dimension epsb(NN,NN,NN,NN),epcb(NN,NN,NN,NN)
      dimension epxb(NN,NN,NN,NN)
      dimension etetra(NN,NN,NN,NN)
      dimension wlc(NN,NN)
      dimension wlm(NN,NN)
      dimension w2c(NN,NN)
      dimension w2m(NN,NN)

      character*2 component
      character*1 pr_enm

      dimension component(NN)

C-------------------------------------------------------------C
C     Reading chemical interactions
C-------------------------------------------------------------C

 10   Format(2F8.1)
 20   Format(F8.1)
 30   Format('Reading chemical interactions')

      do i=1,NN
         do j=1,NN
            wlc(i,j)=0.0
            w2c(i,j)=0.0
            wlm(i,j)=0.0
            w2m(i,j)=0.0
            do k=1,NN
               do l=1,NN
                  etetra(i,j,k,l)=0.0
               enddo
            enddo
         enddo
      enddo

 50   Format(/1X,'***********Chemical Interactions:***********')
 60   Format(/1X,'***********Magnetic Interactions:***********')
 70   Format(1X,'wlc(',A2,',',A2,')= ',F12.5)
 80   Format(1X,'w2c(',A2,',',A2,')= ',F12.5)
 90   Format(1X,'etetra(',A2,',',A2,',',
     *     A2,',',A2,')=',F12.5)

      if(ncomp.gt.1) then
         Write(10,50)
         Write(*,30)
         do i=1,ncomp
            do j=i,ncomp
               read(5,10) wlc(i,j),w2c(i,j)
```

```
            Write(10,70) component(i),component(j),
     *           wlc(i,j)
            Write(10,80) component(i),component(j),
     *           w2c(i,j)
            wlc(j,i)=wlc(i,j)
            w2c(j,i)=w2c(i,j)
          enddo
        enddo
        do i=1,ncomp
          do j=i,ncomp
            do l=j,ncomp
              do k=i,l
                Read(5,20) etetra(i,j,k,l)
                Write(10,90) component(i),component(j),
     *               component(k),component(l),
     *               etetra(i,j,k,l)
                etetra(j,i,k,l)=etetra(i,j,k,l)
                etetra(i,j,l,k)=etetra(i,j,k,l)
                etetra(j,i,l,k)=etetra(i,j,k,l)
                etetra(k,l,i,j)=etetra(i,j,k,l)
                etetra(k,l,j,i)=etetra(i,j,k,l)
                etetra(l,k,i,j)=etetra(i,j,k,l)
                etetra(l,k,j,i)=etetra(i,j,k,l)
              enddo
            enddo
          enddo
        enddo
      endif

C-----------------------------------------------------------------C
C     Reading magnetic interactions
C-----------------------------------------------------------------C


 110  Format(1X,'wlm(',A2,',',A2,')=',F12.5)
 120  Format(1X,'w2m(',A2,',',A2,')=',F12.5)
 130  Format(1X,'Reading magnetic interactions')

      if(Nmag.gt.0) then
        Write(10,60)
        write(*,130)
        do i=1,Nmag
          do j=1,Nmag
            read(5,10) wlm(i,j),w2m(i,j)
            write(10,110) component(i),component(j),
     *           wlm(i,j)
            write(10,120) component(i),component(j),
     *           w2m(i,j)
            wlm(j,i)=wlm(i,j)
            w2m(j,i)=w2m(i,j)
          enddo
        enddo
      endif

C-----------------------------------------------------------------C
C     Calculating eigenenergy matrix
C-----------------------------------------------------------------C

 140  Format(1X,21(1H*),'Energy Matrix:',20(1H*))
 150  Format(1X,'*ni   si   nj   sj   nk   sk',3X,
     *     'nl   sl      epsilon   *')
 160  Format(1X,'*',4(A2,1X,F6.3,1X),1X,E12.5,'*')
 170  Format(1X,55(1H*))

      if (pr_enm.eq.'Y') then
        write(10,140)
        write(10,150)
      endif

      if (Nmag.eq.0) then
        do i=1,Nel
          do j=1,Nel
            do k=1,Nel
              do l=1,Nel
                epsb(i,j,k,l)=etetra(i,j,k,l)
     *               +(wlc(i,k)+wlc(i,l)
     *               +wlc(j,k)+wlc(j,l))/6.
     *               +(w2c(i,j)+w2c(k,l))/4.
                epxb(i,j,k,l)=epsb(i,j,k,l)
                if(pr_enm.eq.'Y') then
                  Write(10,160) component(i),0.0,
     *                 component(j),0.0,
     *                 component(k),0.0,
     *                 component(l),0.0,
     *                 epsb(i,j,k,l)
                endif
              enddo
            enddo
          enddo
        enddo
      else
        do i=1,Nel
          do j=1,Nel
            do k=1,Nel
              do l=1,Nel
                call indexing(i,ni,si)
                call indexing(j,nj,sj)
                call indexing(k,nk,sk)
                call indexing(l,nl,sl)
                epxb(ni,nj,nk,nl) = etetra(ni,nj,nk,nl)
     *               +(wlc(ni,nk)+wlc(ni,nl)
     *               + wlc(nj,nk)+wlc(nj,nl))/6.
     *               +(w2c(ni,nj)+w2c(nk,nl))/4.
                epsb(i,j,k,l) = epxb(ni,nj,nk,nl)
     *               + ((wlm(ni,nk)*si*sk)
     *               + (wlm(ni,nl)*si*sl)
     *               + (wlm(nj,nk)*sj*sk)
     *               + (wlm(nj,nl)*sj*sl))/6.
     *               +((w2m(ni,nj)*si*sj)
     *               + (w2m(nk,nl)*sk*sl))/4.
                if(pr_enm.eq.'Y') then
                  write(10,160) component(ni),
     *                 ((nspin(ni)-1)*si/2.),
     *                 component(nj),((nspin(nj)-1)*sj/2.),
     *                 component(nk),((nspin(nk)-1)*sk/2.),
     *                 component(nl),((nspin(nl)-1)*sl/2.),
     *                 epsb(i,j,k,l)
                endif
              enddo
            enddo
          enddo
        enddo
      endif

      if (pr_enm.eq.'Y') then
        write(10,170)
      endif

      return
      end

C*************************************************************C
      subroutine energyf(pr_enm)
C*************************************************************C

      Parameter (NN=17)

C-------------------------------------------------------------C
C     Based on a similar subroutine in program bfcc.for
C     (C. Colinet)
C     Adapted by Claudio G. Schoen in the actual form
C     (July 17th 1997)
C
C     For FCC phases in the regular tetrahedron cluster
C     approximation
C
C     Reads interaction parameters in unit 10 and calculates
C     the eigenenergy matrix (epsilon) for the system
C-------------------------------------------------------------C

      Common/Csys/Nel,Nmag,ncomp,nspin,component
      Common/CFCC/epsf,epcf

      integer(kind=2) i,j,k,l
      integer(kind=2) Nel,Nmag,ncomp,nspin
      integer(kind=2) ni,nj,nk,nl

      dimension nspin(NN)

      real(kind=2) epsf,epcf
      real(kind=2) wlc,wlm
      real(kind=2) etetra
      real(kind=2) si,sj,sk,sl

      dimension epsf(NN,NN,NN,NN),epcf(NN,NN,NN,NN)
      dimension etetra(NN,NN,NN,NN)
      dimension wlc(NN,NN)
      dimension wlm(NN,NN)

      character*2 component
      character*1 pr_enm

      dimension component(NN)

C-------------------------------------------------------------C
C     Reading chemical interactions
C-------------------------------------------------------------C

 10   Format(F8.1)
 20   Format(F8.1)
 30   Format('Reading chemical interactions')

      do i=1,NN
        do j=1,NN
          wlc(i,j)=0.0
          wlm(i,j)=0.0
          do k=1,NN
            do l=1,NN
              etetra(i,j,k,l)=0.0
            enddo
          enddo
        enddo
      enddo

 50   Format(/1X,'***********Chemical Interactions:**********')
 60   Format(/1X,'***********Magnetic Interactions:**********')
 70   Format(1X,'wlc(',A2,',',A2,')= ',F12.5)
 90   Format(1X,'etetra(',A2,',',A2,',',
     *     A2,',',A2,')=',F12.5)

      if(ncomp.gt.1) then
        Write(10,50)
        Write(*,30)
        do i=1,ncomp
          do j=i,ncomp
            read(5,10) wlc(i,j)
            Write(10,70) component(i),component(j),
     *           wlc(i,j)
            wlc(j,i)=wlc(i,j)
          enddo
        enddo
        do i=1,ncomp
          do j=i,ncomp
            do k=j,ncomp
              do l=k,ncomp
                Read(5,20) etetra(i,j,k,l)
                Write(10,90) component(i),component(j),
     *               component(k),component(l),
     *               etetra(i,j,k,l)
                etetra(j,i,k,l)=etetra(i,j,k,l)
                etetra(i,j,l,k)=etetra(i,j,k,l)
                etetra(j,i,l,k)=etetra(i,j,k,l)
                etetra(k,l,i,j)=etetra(i,j,k,l)
                etetra(k,l,j,i)=etetra(i,j,k,l)
                etetra(l,k,i,j)=etetra(i,j,k,l)
                etetra(l,k,j,i)=etetra(i,j,k,l)
                etetra(i,k,l,j)=etetra(i,j,k,l)
                etetra(i,l,j,k)=etetra(i,j,k,l)
                etetra(i,l,k,j)=etetra(i,j,k,l)
                etetra(i,k,j,l)=etetra(i,j,k,l)
                etetra(j,k,l,i)=etetra(i,j,k,l)
                etetra(j,l,i,k)=etetra(i,j,k,l)
                etetra(j,k,i,l)=etetra(i,j,k,l)
                etetra(j,l,k,i)=etetra(i,j,k,l)
                etetra(k,i,j,l)=etetra(i,j,k,l)
                etetra(k,j,l,i)=etetra(i,j,k,l)
                etetra(k,j,i,l)=etetra(i,j,k,l)
                etetra(k,i,l,j)=etetra(i,j,k,l)
                etetra(l,i,j,k)=etetra(i,j,k,l)
                etetra(l,j,k,i)=etetra(i,j,k,l)
                etetra(l,i,k,j)=etetra(i,j,k,l)
                etetra(l,j,i,k)=etetra(i,j,k,l)
              enddo
            enddo
          enddo
        enddo
      endif

C-------------------------------------------------------------C
C     Reading magnetic interactions
C-------------------------------------------------------------C
```

```
110   Format(1X,'wlm(',A2,',',A2,')=',F12.5)

130   Format('Reading magnetic interactions')

      if(Nmag.gt.0) then
         write(10,60)
         write(*,130)
         do i=1,Nmag
            do j=i,Nmag
               read(5,10) wlm(i,j)
               write(10,110) component(i),component(j),
     *              wlm(i,j)
               wlm(j,i)=wlm(i,j)
            enddo
         enddo
      endif

C-----------------------------------------------------------------C
C     Calculating eigenenergy matrix
C-----------------------------------------------------------------C

140   Format(1X,21(1H*),'Energy Matrix:',20(1H*))
150   Format(1X,'*ni   si   nj   sj   nk   sk',3X,
     *     'nl   sl      epsilon   *')
160   Format(1X,'*',4(A2,1X,F6.3,1X),1X,E12.5,'*')
170   Format(1X,55(1H*))

      if (pr_enm.eq.'Y') then
         write(10,140)
         write(10,150)
      endif

      if (Nmag.eq.0) then
         do i=1,Nel
            do j=1,Nel
               do k=1,Nel
                  do l=1,Nel
                     epsf(i,j,k,l)=etetra(i,j,k,l)
     *                    -(wlc(i,k)+wlc(i,l)
     *                    + wlc(j,k)+wlc(j,l)
     *                    + wlc(i,j)+wlc(k,l))/2.
                     if(pr_enm.eq.'Y') then
                        write(10,160) component(i),0.0,
     *                       component(j),0.0,
     *                       component(k),0.0,
     *                       component(l),0.0,
     *                       epsf(i,j,k,l)
                     endif
                  enddo
               enddo
            enddo
         enddo
      else
         do i=1,Nel
            do j=1,Nel
               do k=1,Nel
                  do l=1,Nel
                     call indexing(i,ni,si)
                     call indexing(j,nj,sj)
                     call indexing(k,nk,sk)
                     call indexing(l,nl,sl)
                     epsf(i,j,k,l)=etetra(ni,nj,nk,nl)
     *                    + (wlc(ni,nk)+wlc(ni,nl)
     *                    +  wlc(nj,nk)+wlc(nj,nl)
     *                    +  wlc(ni,nj)+wlc(nk,nl))/2.
     *                    -((wlm(ni,nk)*si*sk)
     *                    + (wlm(ni,nl)*si*sl)
     *                    + (wlm(nj,nk)*sj*sk)
     *                    + (wlm(nj,nl)*sj*sl)
     *                    + (wlm(ni,nj)*si*sj)
     *                    + (wlm(nk,nl)*sk*sl))/2.
                     if(pr_enm.eq.'Y') then
                        write(10,160) component(ni),
     *                       ((nspin(ni)-1)*si/2.),
     *                       component(nj),((nspin(nj)-1)*sj/2.),
     *                       component(nk),((nspin(nk)-1)*sk/2.),
     *                       component(nl),((nspin(nl)-1)*sl/2.),
     *                       epsf(i,j,k,l)
                     endif
                  enddo
               enddo
            enddo
         enddo
      endif

      if (pr_enm.eq.'Y') then
         write(10,170)
      endif

      return
      end

C*****************************************************************C
      Subroutine indexing(a,na,sa)
C*****************************************************************C

      Parameter(NN=17)

C-----------------------------------------------------------------C
C     Created by Claudio G. Schoen (July 10th 1997)
C     Converts the absolute index of the alloy species (a)
C     into the index of the alloy component (na) and its
C     reduced spin: sa=s(a)/s
C-----------------------------------------------------------------C

      Common/Csys/Nel,Nmag,ncomp,nspin,component

      integer(kind=2) Nel,Nmag,ncomp,nspin
      integer(kind=2) a,na
      integer(kind=2) contr,contr1,i
      integer(kind=2) uppb

      character*2 component

      Dimension nspin(NN)
      Dimension component(NN)

      real(kind=2) sa

      uppb=0

      do i=1,Nmag
         uppb=uppb+nspin(i)
      enddo

      contr=a
      if (a.gt.uppb) then
         i=a-uppb+Nmag
      else
         i=0
         do 100 while(contr.gt.0)
            contr1=contr
            contr=contr-nspin(i+1)
            i=i+1
100      enddo
      endif
      na=i
      if (na.le.Nmag) then
         sa=1.0d+0-((2.0d+0*(contr1-1))/(nspin(i)-1))
      else
         sa=0.0d+0
      endif

      return
      end

C*****************************************************************C
      subroutine refgp(Phasel,reference)
C*****************************************************************C

      Parameter(NN=17)

C-----------------------------------------------------------------C
C     Created by Claudio G. Schoen (July 11th 1997)
C     Reads the reference states for the free energy in the
C     text file (SGTE.txt)
C     Phasel: Label for the structure
C-----------------------------------------------------------------C

      Common/Csys/Nel,Nmag,ncomp,nspin,component

      integer(kind=2) Nel,Nmag,ncomp,nspin
      integer(kind=2) i,j
      integer(kind=2) flag1

      dimension nspin(NN)

      real(kind=2) prmts
      real(kind=2) reference

      dimension prmts(8)
      dimension reference(NN,8)

      character*2 component,name
      character*3 Phasel,phase

      dimension component(NN)

      do i=1,NN
         do j=1,8
            reference(i,j)=0.0
         enddo
      enddo

      Open(Unit=9,File='SGTE.txt',Status='Old',Form='Formatted')

10    Format(A2,1X,A3,1X,8E15.10)

      flag1=0

      Read(9,10,END=100) name,phase,(prmts(i),i=1,8)

      do i=1,ncomp
         if (name.eq.component(i)) then
            if (phase.eq.Phasel) then
               do j=1,8
                  reference(i,j)=prmts(j)
               enddo
               flag1=flag1+1
            endif
         endif
      enddo

20    Format('Incomplete list of parameters in file SGTE.txt')

      if (flag1.lt.ncomp) then
         Write(10,20)
      endif

30    Format('Warning: There is something wrong in file SGTE.txt!')

      if (flag1.gt.ncomp) then
         Write(*,30)
         Write(10,30)
      endif

100   continue

      Close(Unit=9)

      return
      end


C*****************************************************************C
      subroutine ORDBCC(Iphase,itr,GP,Fgy,Etpy,Egy)
C*****************************************************************C

C-----------------------------------------------------------------C
C     Based on a similar subroutine in program bfcc.for
C     (C. Colinet)
C     Adapted by Claudio G. Schoen in the actual form
C     (July 18th 1998)
C     Last changed by C. G. Schoen in August 12th 1998
C
C     For BCC ordered phases in the irregular tetrahedron
C     cluster approximation
C
C     Natural Iteraction minimization of the free energy
C-----------------------------------------------------------------C

      Parameter (NN=17)

      Common/Csys/Nel,Nmag,ncomp,nspin,component
      Common/CBCC/epsb,epcb,epxb
      Common/Cpot/ecp,TK,RTK
      Common/Cord/ZO

      integer(kind=2) i,j,k,l
      integer(kind=2) Iphase,itr
      integer(kind=2) Nel,Nmag,ncomp,nspin
```

```
      integer(kind=2) Ichem,Imag
      integer(kind=2) a,b,i1

      dimension nspin(NN)
      dimension i1(NN)

      real(kind=2) epsb,epcb,ecp,ZO,GP,Fgy,Etpy,Egy
      real(kind=2) epxb
      real(kind=2) TK,RTK
      real(kind=2) Zln,Zlnin,dZ
      real(kind=2) Zh,Zhln
      real(kind=2) Cptm
      real(kind=2) VAL,VBT,VGM,VDT
      real(kind=2) Y1AG,Y1AD,Y1BG,Y1BD
      real(kind=2) Y2AB,Y2GD
      real(kind=2) XAL,XBT,XGM,XDT
      real(kind=2) VALln,VBTln,VGMln,VDTln
      real(kind=2) Y1AGln,Y1ADln,Y1BGln,Y1BDln
      real(kind=2) Y2ABln,Y2GDln
      real(kind=2) XALln,XBTln,XGMln,XDTln
      real(kind=2) ZVln,ZY1ln,ZY2ln,ZXln
      real(kind=2) Emgp,Egp,GPln
      real(kind=2) Ztransf

      character*2 component

      Dimension component(NN)

      dimension epsb(NN,NN,NN,NN),epcb(NN,NN,NN,NN)
      dimension epxb(NN,NN,NN,NN)
      dimension ecp(NN,NN,NN,NN)
      dimension ZO(NN,NN,NN,NN),Zln(NN,NN,NN,NN)
      dimension Zlnin(NN,NN,NN,NN)
      dimension Zhln(NN,NN,NN,NN),Zh(NN,NN,NN,NN)
      dimension Zz(NN,NN,NN,NN)
      dimension VAL(NN,NN,NN),VBT(NN,NN,NN)
      dimension VGM(NN,NN,NN),VDT(NN,NN,NN)
      dimension Y1AG(NN,NN),Y1AD(NN,NN)
      dimension Y1BG(NN,NN),Y1BD(NN,NN)
      dimension Y2AB(NN,NN),Y2GD(NN,NN)
      dimension XAL(NN),XBT(NN),XGM(NN),XDT(NN)
      dimension VALln(NN,NN,NN),VBTln(NN,NN,NN)
      dimension VGMln(NN,NN,NN),VDTln(NN,NN,NN)
      dimension Y1AGln(NN,NN),Y1ADln(NN,NN)
      dimension Y1BGln(NN,NN),Y1BDln(NN,NN)
      dimension Y2ABln(NN,NN),Y2GDln(NN,NN)
      dimension XALln(NN),XBTln(NN),XGMln(NN),XDTln(NN)

      Data itrf,testZ/20000,1.0e-4/

      Ichem=mod(Iphase,10)

      if (Nmag.gt.0) then
         Imag=Iphase/10
      else
         Imag=0
      endif

      if (Imag.gt.0) then
         i1(1)=1
         if (Nmag.gt.1) then
            do i=2,Nmag
               i1(i)=i1(i-1)+nspin(i-1)
            enddo
         endif
      endif


      do i=1,Nel
         do j=1,Nel
            do k=1,Nel
               do l=1,Nel
                  Zln(i,j,k,l)=dlog(ZO(i,j,k,l))
               enddo
            enddo
         enddo
      enddo


      itr=0
      dZ=2.0


      do 100 while(.not.((dZ.lt.testZ).or.(itr.ge.itrf)))
         itr=itr+1

C-----------------------------------------------------------------------
--------C
C     Reduction relations
C
C     VABG(i,j,k)= alfa,beta,gamma triangle cluster probability
C
C     VABD(i,j,l)= alfa,beta,delta triangle cluster probability
C
C     VGDA(k,l,i)= gamma,delta,alfa triangle cluster probability
C
C     VGDB(k,l,j)= gamma,delta,beta triangle cluster probability
C
C     Y1AG(i,k)= alfa,gamma nearest neighbour pair cluster probability
C
C     Y1AD(i,l)= alfa,delta nearest neighbour pair cluster probability
C
C     Y1BG(j,k)= beta,gamma nearest neighbour pair cluster probability
C
C     Y1BD(j,l)= beta,delta nearest neighbour pair cluster probability
C
C     Y2AB(i,j)= alfa,beta next nearest neighbour pair cluster
probability
C
C     Y2GD(k,l)= gamma,delta next nearest neighbour pair cluster
probability
C
C     XAL(i)= alfa point cluster probability
C
C     XBT(j)= beta point cluster probability
C
C     XGM(k)= gamma point cluster probability
C
C     XDT(l)= delta point cluster probability
C-----------------------------------------------------------------------
--------C

         do i=1,Nel
            XAL(i)=0.0d+0
            XBT(i)=0.0d+0
            XGM(i)=0.0d+0
            XDT(i)=0.0d+0
            do j=1,Nel
               Y1AG(i,j)=0.0d+0
               Y1AD(i,j)=0.0d+0
               Y1BG(i,j)=0.0d+0
               Y1BD(i,j)=0.0d+0
               Y2AB(i,j)=0.0d+0
               Y2GD(i,j)=0.0d+0
               do k=1,Nel
                  VAL(i,j,k)=0.0d+0
                  VBT(i,j,k)=0.0d+0
                  VGM(i,j,k)=0.0d+0
                  VDT(i,j,k)=0.0d+0
                  do l=1,Nel
                     Zlnin(i,j,k,l)=Zln(i,j,k,l)
                     VAL(i,j,k)=VAL(i,j,k)+ZO(i,l,j,k)
                     VBT(i,j,k)=VBT(i,j,k)+ZO(l,i,j,k)
                     VGM(i,j,k)=VGM(i,j,k)+ZO(j,k,i,l)
                     VDT(i,j,k)=VDT(i,j,k)+ZO(j,k,l,i)
                     Y1AG(i,j)=Y1AG(i,j)+ZO(i,k,j,l)
                     Y1AD(i,j)=Y1AD(i,j)+ZO(i,k,l,j)
                     Y1BG(i,j)=Y1BG(i,j)+ZO(k,i,j,l)
                     Y1BD(i,j)=Y1BD(i,j)+ZO(k,i,l,j)
                     Y2AB(i,j)=Y2AB(i,j)+ZO(i,j,k,l)
                     Y2GD(i,j)=Y2GD(i,j)+ZO(k,l,i,j)
                     XAL(i)=XAL(i)+ZO(i,j,k,l)
                     XBT(i)=XBT(i)+ZO(l,i,j,k)
                     XGM(i)=XGM(i)+ZO(k,l,i,j)
                     XDT(i)=XDT(i)+ZO(j,k,l,i)
                  enddo
                  VALln(i,j,k)=dlog(VAL(i,j,k))
                  VBTln(i,j,k)=dlog(VBT(i,j,k))
                  VGMln(i,j,k)=dlog(VGM(i,j,k))
                  VDTln(i,j,k)=dlog(VDT(i,j,k))
               enddo
               Y1AGln(i,j)=dlog(Y1AG(i,j))
               Y1ADln(i,j)=dlog(Y1AD(i,j))
               Y1BGln(i,j)=dlog(Y1BG(i,j))
               Y1BDln(i,j)=dlog(Y1BD(i,j))
               Y2ABln(i,j)=dlog(Y2AB(i,j))
               Y2GDln(i,j)=dlog(Y2GD(i,j))
            enddo
            XALln(i)=dlog(XAL(i))
            XBTln(i)=dlog(XBT(i))
            XGMln(i)=dlog(XGM(i))
            XDTln(i)=dlog(XDT(i))
         enddo


         Emgp=0.0d+0

         do i=1,Nel
            do j=1,Nel
               do k=1,Nel
                  do l=1,Nel
                     ZVln=VALln(i,k,l)+VBTln(j,k,l)
     .                    +VGMln(k,i,j)+VDTln(l,i,j)
                     ZY1ln=Y1AGln(i,k)+Y1ADln(i,l)
     .                    +Y1BGln(j,k)+Y1BDln(j,l)
                     ZY2ln=Y2ABln(i,j)+Y2GDln(k,l)
                     ZXln=XALln(i)+XBTln(j)+XGMln(k)+XDTln(l)
                     Zhln(i,j,k,l)=epcb(i,j,k,l)+(ZVln/2.0d+0)
     .                    -(ZY1ln/6.0d+0)-(ZY2ln/4.0d+0)
     .                    +(ZXln/24.0d+0)
                     Zh(i,j,k,l)=dexp(Zhln(i,j,k,l))
                     Emgp= Emgp+Zh(i,j,k,l)
                  enddo
               enddo
            enddo
         enddo


         Egp=(1.0d+0)/Emgp
         GPln=dlog(Egp)
         GP=(6.0d+0)*GPln*TK

         do i=1,Nel
            do j=1,Nel
               do k=1,Nel
                  do l=1,Nel
                     ZO(i,j,k,l)=Zh(i,j,k,l)*Egp
                  enddo
               enddo
            enddo
         enddo

C-----------------------------------------------------------------------C
C     Imposing superlattice symmetry:
C     Ichem= 2 or 3 -> B2 symmetry
C     Ichem= 4 or 5 -> DO3(L21) symmetry
C     Ichem= 6 -> B32 symmetry
C     Ichem= 7 -> ABAC symmetry
C     Ichem= 8 or 9 -> no symmetry imposed
C
C     for all cases above:
C     Imag <> 0 -> paramagnetic phase
C
C     Observation: Use options 8 and 9 unless you are having problems
C     to find second order lines. The symmetry conditions above may
C     hidden domains of existence for a priori unknown phases in the
C     ph. diagram
C-----------------------------------------------------------------------C

         do i=1,Nel
            do j=1,Nel
               do k=1,Nel
                  do l=1,Nel
                     if((Ichem.eq.2).or.(Ichem.eq.3)) then
                        Zz(i,j,k,l)=(ZO(i,j,k,l)+ZO(j,i,k,l)
     .                       +ZO(i,j,l,k)+ZO(j,i,l,k))/4.0d+0
                     else
                        if((Ichem.eq.4).or.(Ichem.eq.5)) then
                           Zz(i,j,k,l)= (ZO(i,j,k,l)
     .                          +ZO(j,i,k,l))/2.0d+0
                        else
                           if(Ichem.eq.6) then
                              Zz(i,j,k,l)=(ZO(i,j,k,l)
     .                             +ZO(k,j,i,l)
     .                             +ZO(i,l,k,j)
     .                             +ZO(k,l,i,j))/4.0d+0
                           else
                              if(Ichem.eq.7) then
                                 Zz(i,j,k,l)=(ZO(i,j,k,l)
     .                                +ZO(k,j,i,l))/2.0d+0
                              else
```

```
                                Zz(i,j,k,l)=ZO(i,j,k,l)
                              endif
                            endif
                          endif
                        endif
                      enddo
                    enddo
                  enddo
                enddo

C-----------------------------------------------------------------C
C     imposing paramagnetism
C-----------------------------------------------------------------C

        if (Imag.gt.0) then
           do a=1,Nmag
              do b=1,(nspin(a)/2)
                 do j=1,Nel
                    do k=1,Nel
                       do l=1,Nel
                          Ztransf=(Zz(il(a)+b-1,j,k,l)+
     *                         Zz(il(a)+nspin(a)-b,j,k,l))/2.0d+0
                          Zz(il(a)+b-1,j,k,l)=Ztransf
                          Zz(il(a)+nspin(a)-b,j,k,l)=Ztransf

                          Ztransf=(Zz(j,il(a)+b-1,k,l)+
     *                         Zz(j,il(a)+nspin(a)-b,k,l))/2.0d+0
                          Zz(j,il(a)+b-1,k,l)=Ztransf
                          Zz(j,il(a)+nspin(a)-b,k,l)=Ztransf

                          Ztransf=(Zz(j,k,il(a)+b-1,l)+
     *                         Zz(j,k,il(a)+nspin(a)-b,l))/2.0d+0
                          Zz(j,k,il(a)+b-1,l)=Ztransf
                          Zz(j,k,il(a)+nspin(a)-b,l)=Ztransf

                          Ztransf=(Zz(j,k,l,il(a)+b-1)+
     *                         Zz(j,k,l,il(a)+nspin(a)-b))/2.0d+0
                          Zz(j,k,l,il(a)+b-1)=Ztransf
                          Zz(j,k,l,il(a)+nspin(a)-b)=Ztransf
                       enddo
                    enddo
                 enddo
              enddo
           enddo
        endif

C-----------------------------------------------------------------C
C     Calculation of the potentials
C     Egy= internal energy
C     Fgy= free energy (Helmholtz potential)
C     Etpy= entropy
C-----------------------------------------------------------------C

        dZ=0.0
        Egy=0.0
        Cptm=0.0


        do i=1,Nel
           do j=1,Nel
              do k=1,Nel
                 do l=1,Nel
                    ZO(i,j,k,l)=Zz(i,j,k,l)
                    Zln(i,j,k,l)=dlog(ZO(i,j,k,l))
                    dZ=dZ+abs(Zln(i,j,k,l)-Zlnin(i,j,k,l))
                    Egy=Egy+(6.0d-0*epsb(i,j,k,l)*ZO(i,j,k,l))
                    Cptm= Cptm+(ecp(i,j,k,l)*ZO(i,j,k,l))
                 enddo
              enddo
           enddo
        enddo

        Cptm= Cptm/(4.0d+0)
        Fgy=GP+Cptm
        Etpy=(-Fgy+Egy)*RTK

 100  enddo


 10   Format(/1X,'Iphase= ',I2,1X,'No convergence in ORD BCC after',
     *     1X,I5,1X,'iterations:',/,1X,'test= ',E12.5)

      if (itr.ge.itrf) then
         Write(10,10) Iphase,itr,dZ
      endif


      return
      end


C*****************************************************************C
      subroutine DISBCC(itr,GP,Fgy,Etpy,Egy)
C*****************************************************************C

C-----------------------------------------------------------------C
C     Based on a similar subroutine in program bfcc.for
C     (C. Colinet)
C     Adapted by Claudio G. Schoen in the actual form
C     (July 17th 1997)
C     Last changed by C. G. Schoen in 12th August 1998
C
C     For BCC disordered phases in the irregular tetrahedron
C     cluster approximation
C
C     Natural Iteration minimization of the free energy
C-----------------------------------------------------------------C

      Parameter (NN=17)

      Common/Csys/Nel,Nmag,ncomp,nspin,component
      Common/CBCC/epsb,epcb,epxb
      Common/Cpct/ecp,TK,RTK
      Common/Cdis/ZD

      integer(kind=2) i,j,k,l
      integer(kind=2) itr
      integer(kind=2) Nel,Nmag,ncomp,nspin

      Dimension nspin(NN)

      real(kind=2) epsb,epcb,ecp,ZD,GP,Fgy,Etpy,Egy
      real(kind=2) epxb
      real(kind=2) TK,RTK
      real(kind=2) Zln,Zlnin,dZ
      real(kind=2) V,Vln,Y1,Y1ln,Y2,Y2ln,X,Xln
      real(kind=2) VVln,YY1ln,YY2ln,XXln,Zhln,Zh,Emgp,EGP,GPln
      real(kind=2) Cptm

      dimension epsb(NN,NN,NN,NN),epcb(NN,NN,NN,NN)
      dimension epxb(NN,NN,NN,NN)
      dimension ecp(NN,NN,NN,NN)
      dimension ZD(NN,NN,NN,NN),Zln(NN,NN,NN,NN)
      dimension Zlnin(NN,NN,NN,NN)
      dimension V(NN,NN,NN),Vln(NN,NN,NN)
      dimension Y1(NN,NN),Y1ln(NN,NN)
      dimension Y2(NN,NN),Y2ln(NN,NN)
      dimension X(NN),Xln(NN)
      dimension Zhln(NN,NN,NN,NN),Zh(NN,NN,NN,NN)

      character*2 component

      Dimension component(NN)

      Data itrf,testZ/20000,1.0e-4/


      do i=1,Nel
         do j=1,Nel
            do k=1,Nel
               do l=1,Nel
                  Zln(i,j,k,l)=dlog(ZD(i,j,k,l))
               enddo
            enddo
         enddo
      enddo

      dZ=2.0
      itr=0

      do 100 while(.not.((dZ.lt.testZ).or.(itr.ge.itrf)))
         itr=itr+1

C-----------------------------------------------------------------C
C     Reduction relations
C     V(i,j,k)= (isotropic) alfa,beta,gamma triangle cluster
C     probability
C     Y1(i,j)= (isotropic) nearest neighbour pair cluster
C     probability
C     Y2(i,j)= (isotropic) next nearest neighbour pair
C     cluster probability
C     X(i)= (isotropic) point cluster probability
C-----------------------------------------------------------------C

         do i=1,Nel
            X(i)=0.0
            do j=1,Nel
               Y1(i,j)=0.0
               Y2(i,j)=0.0
               do k=1,Nel
                  V(i,j,k)=0.0
                  do l=1,Nel
                     Zlnin(i,j,k,l)=Zln(i,j,k,l)
                     V(i,j,k)=V(i,j,k)+ZD(i,j,k,l)
                     Y1(i,j)=Y1(i,j)+ZD(i,k,j,l)
                     Y2(i,j)=Y2(i,j)+ZD(i,j,k,l)
                     X(i)=X(i)+ZD(i,j,k,l)
                  enddo
                  Vln(i,j,k)=dlog(V(i,j,k))
               enddo
               Y1ln(i,j)=dlog(Y1(i,j))
               Y2ln(i,j)=dlog(Y2(i,j))
            enddo
            Xln(i)=dlog(X(i))
         enddo

C-----------------------------------------------------------------C
C     One NIM iteration
C-----------------------------------------------------------------C

         Emgp=0.0

         do i=1,Nel
            do j=1,Nel
               do k=1,Nel
                  do l=1,Nel
                     VVln=Vln(i,k,j)+Vln(i,l,j)+
     *                    Vln(k,i,l)+Vln(k,j,l)
                     YY1ln=Y1ln(i,k)+Y1ln(i,l)+Y1ln(j,k)
     *                    +Y1ln(j,l)
                     YY2ln=Y2ln(i,j)+Y2ln(k,l)
                     XXln=(Xln(i)+Xln(j)+Xln(k)+Xln(l))
                     Zhln(i,j,k,l)=epcb(i,j,k,l)
     *                    +(VVln/2.0d+0)-(YY2ln/4.0d+0)
     *                    -(YY1ln/6.0d+0)+(XXln/24.0d+0)
                     Zh(i,j,k,l)=dexp(Zhln(i,j,k,l))
                     Emgp=Emgp+Zh(i,j,k,l)
                  enddo
               enddo
            enddo
         enddo

         EGP=1.0d+0/Emgp
         GPln=dlog(EGP)
         GP=2.0d+0*GPln*TK

         Egy= 0.0
         Cptm= 0.0

         dZ=0.0
         do i=1,Nel
            do j=1,Nel
               do k=1,Nel
                  do l=1,Nel
                     Zln(i,j,k,l)=Zhln(i,j,k,l)+GPln
                     ZD(i,j,k,l)=Zh(i,j,k,l)*EGP
                     dZ=dZ+abs(Zln(i,j,k,l)-Zlnin(i,j,k,l))
                     Egy=Egy+(2.0d+0*epsb(i,j,k,l)*ZD(i,j,k,l))
                     Cptm=Cptm+(ecp(i,j,k,l)*ZD(i,j,k,l)/4.0d+0)
                  enddo
               enddo
            enddo
         enddo

 100  enddo

      Fgy=GP+Cptm
      Etpy=(-Fgy+Egy)*RTK

 10   Format(/1X,'No convergence in DIS BCC after',1X,I5,1X,
     *     'iterations:',/1X,'test= ',E12.5)
```

```
      if (itr.ge.itrf) then
         Write(10,10) itr,dZ
      endif

      return
      end

C*****************************************************************C
      subroutine ORDFCC(Iphase,itr,GP,Fgy,Etpy,Egy)
C*****************************************************************C

C-----------------------------------------------------------------C
C     Based on a similar subroutine in program bfcc.for
C     (C. Colinet)
C     Adapted by Claudio G. Schoen in the actual form
C     (July 21th 1997)
C
C     For FCC ordered phases in the regular tetrahedron
C     cluster approximation
C
C     Natural Iteraction minimization of the free energy
C-----------------------------------------------------------------C

      Parameter (NN=17)

      Common/Csys/Nel,Nmag,ncomp,nspin,component
      Common/CFCC/epsf,epcf
      Common/Cpot/ecp,TK,RTK
      Common/Cord/ZO

      integer(kind=2) i,j,k,l
      integer(kind=2) Iphase,itr
      integer(kind=2) Nel,Nmag,ncomp,nspin
      integer(kind=2) Ichem,Imag
      integer(kind=2) a,b,il

      dimension nspin(NN)
      dimension il(NN)

      real(kind=2) epsf,epcf,ecp,ZO,GP,Fgy,Etpy,Egy
      real(kind=2) TK,RTK
      real(kind=2) Zln,Zlnin,dZ
      real(kind=2) Zh,Zhln
      real(kind=2) Cptm
      real(kind=2) YlAG,YlAD,YlBG,YlBD,YlAB,YlGD
      real(kind=2) XAL,XBT,XGM,XDT
      real(kind=2) YlAGln,YlADln,YlBGln,YlBDln,YlABln,YlGDln
      real(kind=2) XALln,XBTln,XGMln,XDTln
      real(kind=2) ZYlln,ZXln
      real(kind=2) Emgp,Egp,GPln
      real(kind=2) Ztransf

      dimension epsf(NN,NN,NN,NN),epcf(NN,NN,NN,NN)
      dimension ecp(NN,NN,NN,NN)
      dimension ZO(NN,NN,NN,NN),Zln(NN,NN,NN,NN)
      dimension Zlnin(NN,NN,NN,NN)
      dimension Zhln(NN,NN,NN,NN),Zh(NN,NN,NN,NN)
      dimension Zz(NN,NN,NN,NN)
      dimension YlAG(NN,NN),YlAD(NN,NN)
      dimension YlBG(NN,NN),YlBD(NN,NN)
      dimension YlAB(NN,NN),YlGD(NN,NN)
      dimension XAL(NN),XBT(NN),XGM(NN),XDT(NN)
      dimension YlAGln(NN,NN),YlADln(NN,NN)
      dimension YlBGln(NN,NN),YlBDln(NN,NN)
      dimension YlABln(NN,NN),YlGDln(NN,NN)
      dimension XALln(NN),XBTln(NN),XGMln(NN),XDTln(NN)

      character*2 component

      Dimension component(NN)

      Data itrf,test2/20000,1.0e-4/

      Ichem=mod(Iphase,10)

      if (Nmag.gt.0) then
         Imag=Iphase/10
      else
         Imag=0
      endif

      if (Imag.gt.0) then
         il(1)=1
         if (Nmag.gt.1) then
            do i=2,Nmag
               il(i)=il(i-1)+nspin(i-1)
            enddo
         endif
      endif


      do i=1,Nel
         do j=1,Nel
            do k=1,Nel
               do l=1,Nel
                  Zln(i,j,k,l)=dlog(ZO(i,j,k,l))
               enddo
            enddo
         enddo
      enddo

      dZ=2.0
      itr=0

      do 100 while(.not.((dZ.lt.test2).or.(itr.ge.itrf)))
         itr=itr+1

C-----------------------------------------------------------------
--------C
C     Reduction relations
C
C     YlAG(i,k)= alfa,gamma nearest neighbour pair cluster probability
C
C     YlAD(i,l)= alfa,delta nearest neighbour pair cluster probability
C
C     YlBG(j,k)= beta,gamma nearest neighbour pair cluster probability
C
C     YlBD(j,l)= beta,delta nearest neighbour pair cluster probability
C
C     YlAB(i,j)= alfa,beta nearest neighbour pair cluster probability
C
C     YlGD(k,l)= gamma,delta nearest neighbour pair cluster
probability
C
C     XAL(i)= alfa point cluster probability
C
C     XBT(j)= beta point cluster probability
C
C     XGM(k)= gamma point cluster probability
C
C     XDT(l)= delta point cluster probability
C-----------------------------------------------------------------
--------C

         do i=1,Nel
            XAL(i)=0.0
            XBT(i)=0.0
            XGM(i)=0.0
            XDT(i)=0.0
            do j=1,Nel
               YlAG(i,j)=0.0
               YlAD(i,j)=0.0
               YlBG(i,j)=0.0
               YlBD(i,j)=0.0
               YlAB(i,j)=0.0
               YlGD(i,j)=0.0
               do k=1,Nel
                  do l=1,Nel
                     Zlnin(i,j,k,l)=Zln(i,j,k,l)
                     YlAG(i,j)=YlAG(i,j)+ZO(i,k,j,l)
                     YlAD(i,j)=YlAD(i,j)+ZO(i,k,l,j)
                     YlBG(i,j)=YlBG(i,j)+ZO(k,i,j,l)
                     YlBD(i,j)=YlBD(i,j)+ZO(k,i,l,j)
                     YlAB(i,j)=YlAB(i,j)+ZO(i,j,k,l)
                     YlGD(i,j)=YlGD(i,j)+ZO(k,l,i,j)
                     XAL(i)=XAL(i)+ZO(i,j,k,l)
                     XBT(i)=XBT(i)+ZO(l,i,j,k)
                     XGM(i)=XGM(i)+ZO(k,l,i,j)
                     XDT(i)=XDT(i)+ZO(j,k,l,i)
                  enddo
               enddo
               YlAGln(i,j)=dlog(YlAG(i,j))
               YlADln(i,j)=dlog(YlAD(i,j))
               YlBGln(i,j)=dlog(YlBG(i,j))
               YlBDln(i,j)=dlog(YlBD(i,j))
               YlABln(i,j)=dlog(YlAB(i,j))
               YlGDln(i,j)=dlog(YlGD(i,j))
            enddo
            XALln(i)=dlog(XAL(i))
            XBTln(i)=dlog(XBT(i))
            XGMln(i)=dlog(XGM(i))
            XDTln(i)=dlog(XDT(i))
         enddo

         Emgp=0.0

         do i=1,Nel
            do j=1,Nel
               do k=1,Nel
                  do l=1,Nel
                     ZYlln=YlAGln(i,k)+YlADln(i,l)
     *                    +YlBGln(j,k)+YlBDln(j,l)
     *                    +YlABln(i,j)+YlGDln(k,l)
                     ZXln=XALln(i)+XBTln(j)+XGMln(k)+XDTln(l)
                     Zhln(i,j,k,l)=epcf(i,j,k,l)+(ZYlln/2.0d+0)
     *                    -(5.0d+0*ZXln/8.0d+0)
                     Zh(i,j,k,l)=dexp(Zhln(i,j,k,l))
                     Emgp= Emgp+Zh(i,j,k,l)
                  enddo
               enddo
            enddo
         enddo

         Egp=(1.0d+0)/Emgp
         GPln=dlog(Egp)
         GP=(2.0d+0)*GPln*TK

         do i=1,Nel
            do j=1,Nel
               do k=1,Nel
                  do l=1,Nel
                     ZO(i,j,k,l)=Zh(i,j,k,l)*Egp
                  enddo
               enddo
            enddo
         enddo


C-----------------------------------------------------------------C
C     Imposing superlattice symmetry:
C     Ichem= 2 or 3 -> L10(AABB) symmetry
C     Ichem= 4 or 5 -> L12(AAAB) symmetry
C     Ichem= 6 or 7 -> AABC symmetry
C     Ichem= 8 or 9 -> no symmetry imposed
C
C     for all cases above:
C     Imag <> 0 -> paramagnetic phase
C
C     Observation: Use options 8 and 9 unless you are having problems
C     to find second order lines. The symmetry conditions above may
C     hidden domains of existence for a priori unknown phases in the
C     ph. diagram
C-----------------------------------------------------------------C

         do i=1,Nel
            do j=1,Nel
               do k=1,Nel
                  do l=1,Nel
                     if((Ichem.eq.2).or.(Ichem.eq.3)) then
                        Zz(i,j,k,l)=(ZO(i,j,k,l)+ZO(j,i,k,l)
     *                       +ZO(i,j,l,k)+ZO(j,i,l,k))/4.0d+0
                     else
                        if((Ichem.eq.4).or.(Ichem.eq.5)) then
                           Zz(i,j,k,l)= (ZO(i,j,k,l)
     *                          +ZO(j,i,k,l)+ZO(j,k,i,l)
     *                          +ZO(i,k,j,l)+ZO(k,i,j,l)
     *                          +ZO(k,j,i,l))/6.0d+0
                        else
                           if((Ichem.eq.6).or.(Ichem.eq.7)) then
                              Zz(i,j,k,l)=(ZO(i,j,k,l)
     *                             +ZO(j,i,k,l))/2.0d+0
                           else
                              Zz(i,j,k,l)=ZO(i,j,k,l)
                           endif
                        endif
                     endif
                  enddo
               enddo
            enddo
         enddo

C-----------------------------------------------------------------C
```

```
C     imposing paramagnetism
C-----------------------------------------------------------------------C

         if (Imag.gt.0) then
            do a=1,Nmag
               do b=1,(nspin(a)/2)
                  do j=1,Nel
                     do k=1,Nel
                        do l=1,Nel
                           Ztransf=(Zz(il(a)+b-1,j,k,l)+
     *                          Zz(il(a)+nspin(a)-b,j,k,l))/2.0d+0
                           Zz(il(a)+b-1,j,k,l)=Ztransf
                           Zz(il(a)+nspin(a)-b,j,k,l)=Ztransf

                           Ztransf=(Zz(j,il(a)+b-1,k,l)+
     *                          Zz(j,il(a)+nspin(a)-b,k,l))/2.0d+0
                           Zz(j,il(a)+b-1,k,l)=Ztransf
                           Zz(j,il(a)+nspin(a)-b,k,l)=Ztransf

                           Ztransf=(Zz(j,k,il(a)+b-1,l)
     *                          +Zz(j,k,il(a)+nspin(a)-b,l))/2.0d+0
                           Zz(j,k,il(a)+b-1,l)=Ztransf
                           Zz(j,k,il(a)+nspin(a)-b,l)=Ztransf

                           Ztransf=(Zz(j,k,l,il(a)+b-1)+
     *                          Zz(j,k,l,il(a)+nspin(a)-b))/2.0d+0
                           Zz(j,k,l,il(a)+b-1)=Ztransf
                           Zz(j,k,l,il(a)+nspin(a)-b)=Ztransf
                        enddo
                     enddo
                  enddo
               enddo
            enddo
         endif

C-----------------------------------------------------------------------C
C     Calculation of the potentials
C     Egy= internal energy
C     Fgy= free energy (Helmholtz potential)
C     Etpy= entropy
C-----------------------------------------------------------------------C

         Egy=0.0
         Cptm=0.0
         dZ=0.0

         do i=1,Nel
            do j=1,Nel
               do k=1,Nel
                  do l=1,Nel
                     ZO(i,j,k,l)=Zz(i,j,k,l)
                     Zln(i,j,k,l)=dlog(ZO(i,j,k,l))
                     dZ=dZ+abs(Zln(i,j,k,l)-Zlnin(i,j,k,l))
                     Egy=Egy+(2.0d+0*epsf(i,j,k,l)*ZO(i,j,k,l))
                     Cptm= Cptm+(ecp(i,j,k,l)*ZO(i,j,k,l))
                  enddo
               enddo
            enddo
         enddo

         Cptm= Cptm/(4.0d+0)
         Fgy=GP+Cptm
         Etpy=(-Fgy+Egy)*RTK

 100  enddo

 10   Format(/1X,'Iphase= ',I2,1X,'No convergence in ORD FCC after',
     *     1X,I5,1X,'iterations:',5X,'test= ',E12.5)

      if (itr.ge.itrf) then
         Write(10,10) Iphase,itr,test
      endif

      return
      end


C*********************************************************************C
      subroutine DISFCC(itr,GP,Fgy,Etpy,Egy)
C*********************************************************************C

C-----------------------------------------------------------------------C
C     Based on a similar subroutine in program bfcc.for
C     (C. Colinet)
C     Adapted by Claudio G. Schoen in the actual form
C     (July 17th 1997)
C
C     For FCC disordered phases in the regular tetrahedron
C     cluster approximation
C
C     Natural Iteration minimization of the free energy
C-----------------------------------------------------------------------C

      Parameter (NN=17)

      Common/Csys/Nel,Nmag,ncomp,nspin,component
      Common/CFCC/epsf,epcf
      Common/Cpot/ecp,TK,RTK
      Common/Cdis/ZD

      integer(kind=2) i,j,k,l
      integer(kind=2) itr
      integer(kind=2) Nel,Nmag,ncomp,nspin

      Dimension nspin(NN)

      real(kind=2) epsf,epcf,ecp,ZD,GP,Fgy,Etpy,Egy
      real(kind=2) TK,RTK
      real(kind=2) Zln,Zlnin,dZ
      real(kind=2) Yl,Ylln,X,Xln
      real(kind=2) YYln,XXln,Zhln,Zh,Emgp,EGP,GPln
      real(kind=2) Cptm

      dimension epsf(NN,NN,NN,NN),epcf(NN,NN,NN,NN)
      dimension ecp(NN,NN,NN,NN)
      dimension ZD(NN,NN,NN,NN),Zln(NN,NN,NN,NN)
      dimension Zlnin(NN,NN,NN,NN)
      dimension Yl(NN,NN),Ylln(NN,NN)
      dimension X(NN),Xln(NN)
      dimension Zhln(NN,NN,NN,NN),Zh(NN,NN,NN,NN)

      character*2 component

      Dimension component(NN)

      Data itrf,testZ/20000,1.0e-4/

      do i=1,Nel
         do j=1,Nel
            do k=1,Nel
               do l=1,Nel
                  Zln(i,j,k,l)=dlog(ZD(i,j,k,l))
               enddo
            enddo
         enddo
      enddo

      dZ=2.0
      itr=0

      do 100 while(.not.((dZ.lt.testZ).or.(itr.ge.itrf)))
         itr=itr+1

C-----------------------------------------------------------------------C
C     Reduction relations
C     Y(i,j)= (isotropic) pair cluster probability
C     X(i)= (isotropic) point cluster probability
C-----------------------------------------------------------------------C

         do i=1,Nel
            X(i)=0.0
            do j=1,Nel
               Yl(i,j)=0.0
               do k=1,Nel
                  do l=1,Nel
                     Zlnin(i,j,k,l)=Zln(i,j,k,l)
                     Yl(i,j)=Yl(i,j)+ZD(i,j,k,l)
                     X(i)=X(i)+ZD(i,j,k,l)
                  enddo
               enddo
               Ylln(i,j)=dlog(Yl(i,j))
            enddo
            Xln(i)=dlog(X(i))
         enddo

C-----------------------------------------------------------------------C
C     One NIM iteration
C-----------------------------------------------------------------------C

         Emgp=0.0

         do i=1,Nel
            do j=1,Nel
               do k=1,Nel
                  do l=1,Nel
                     YYln=(Ylln(i,k)+Ylln(i,l)+Ylln(j,k)+
     *                    Ylln(j,l)+Ylln(i,j)+Ylln(k,l))
                     XXln=(Xln(i)+Xln(j)+Xln(k)+Xln(l))
                     Zhln(i,j,k,l)=epcf(i,j,k,l)
     *                    +(YYln/2.0d+0)-(5.0d+0*XXln/8.0d+0)
                     Zh(i,j,k,l)=dexp(Zhln(i,j,k,l))
                     Emgp=Emgp+Zh(i,j,k,l)
                  enddo
               enddo
            enddo
         enddo
         EGP=1.0d+0/Emgp
         GPln=dlog(EGP)
         GP=2.0d+0*GPln*TK

         Egy= 0.0
         Cptm= 0.0

         dZ=0.0

         do i=1,Nel
            do j=1,Nel
               do k=1,Nel
                  do l=1,Nel
                     Zln(i,j,k,l)=Zhln(i,j,k,l)+GPln
                     ZD(i,j,k,l)=Zh(i,j,k,l)*EGP
                     dZ=dZ+abs(Zln(i,j,k,l)-Zlnin(i,j,k,l))
                     Egy=Egy+(2.0d+0*epsf(i,j,k,l)*ZD(i,j,k,l))
                     Cptm=Cptm+(ecp(i,j,k,l)*ZD(i,j,k,l)/4.0d+0)
                  enddo
               enddo
            enddo
         enddo

 100  enddo

      Fgy=GP+Cptm
      Etpy=(-Fgy+Egy)*RTK

 10   Format(/1X,'No convergence in DIS FCC after',
     *     1X,I5,1X,'iterations:',/1X,'test= ',E12.5)

      if (itr.ge.itrf) then
         Write(10,10) itr,dZ
      endif

      return
      end
```

Referências bibliográficas

. C. T. Liu, J. Stringer, J. N. Mundy, L. L. Horton, P. Angelini, "Ordered intermetallic alloys: an assessment", *Intermetallics 5, 579* (1997).

. A. J. McAlister, *Binary Alloy Phase Diagrams*, T. B. Massalsky, Ed (ASM International, Metals Park–OH, USA, 1992), p. 136.

. K. Ishikawa, M. Ise, I. Ohnuma, R. Kainuma, K. Ishida, "Phase Equilibria and Stability of the BCC Aluminide in the Co–Cr–Al System", *Berichte der Bunsengesellschaft für physikalische Chemie* **102**, 1 (1998).

. I. Ohnuma, C. G. Schön, R. Kainuma, G. Inden, K. Ishida, "Ordering and Phase Separation in the BCC Phase Diagram of the Fe–Al–Ti System", *Acta Materialia* **46**, 2083 (1998).

. C. G. Schön, Thermodynamics of multicomponent systems with chemical and magnetic interactions. (Jan 14 1998). Universität Dortmund: Ph. D.

. W. Johnson, K. Komarek, E. Miller, "Thermodynamic Properties of Solid Cr–Al Allosy at 1000°C", *Transactions of the metallurgical society of AIME* **242**, 1685 (1968).

. M. Ettenberg, K. Komarek, E. Miller, "Thermodinamic Properties and Ordering in CoAl", *Transactions of the metallurgical society of AIME* **242**, 1801 (1968).

. J. Havrankova, J. Vrestal, J. Tomiska, "Thermodynamics of Solid Co–Cr Alloys by Knudsen Cell Mass Spectrometry and Computation of the Phase Diagram". *Berichte Bunsenges. Phys. Chem.* **102**, 1225–1230 (1998).

. G. Inden, W. Pitsch, "Atomic Ordering", in P. Haasen (Ed.), *Phase transformations in materials* (pp. 499–552). Weinheim: VCH.

0.C. Colinet, G. Inden, R. Kikuchi, "CVM Calculation of the Phase Diagram of BCC Fe–Co–Al", *Acta Materialia* **46**, 1109 (1993).

1.McGlashan, M. L., *Chemical Thermodynamics*, Academic Press, London, 1979.

2.Pelton, A. D., "Thermodynamics and phase diagrams of materials". in Haasen, P. (Ed.), *Phase transformations in materials* (pp. 1–73). Weinheim: VCH.