Some personal computer programs for Transmission Electron Microscopy in Mineralogy — Mineral identification , ED pattern simulation and HRTEM image simulation —

by

Toshihiko Takahasi* and Junji Akai **

abstract

Personal computer programs for Transmission Electron Microscopy in Mineralogy were newly written. They are Mineral identification and ED pattern simulation programs. Mineral identification program can be conveniently used for interpretation and identification of ED pattern of mineral. ED pattern simulation program can be used for handling ED pattern by rotating it. HRTEM image simulation program based on kinematical theory written by Ishizuka was used on personal computer by transforming the original program to personal computer usage. Using this, estimation of unknown mineral structure which was found in thermally metamorphosed carbonaceous chondrite was carried out.

key words : personal computer program, Electorn Microscopy, ED pattern, HRTEM image simulation

Introduction

Many calculations are needed in examination of minerals by TEM (e.g. Boisen and Gibbs, 1985; Jackson, 1988). The calculation is in some cases, complicated. Furthermore, graphical presentation is also helpfull for understanding 3 dimensional or complicated situation of the crystalline materials. Some softwares has already been prepared but they are sometimes for special purposes or for special machines. It is convenient to prepare combined personal computer programs fitting for his TEM or his own necessity. We developed some programs convenient for TEM study: We use JEOL

* Public Office of Yoshida Town , Hinode-cho, Yoshida Town 959- 02, Niigata Prefecture, Japan .

** Department of Geology, Faculty of Science, Niigata University, 950-21 Ikarashi, 2-nocho 8050, Niigtata 950-21, Japan .

(Manuscript received 20 December, 1994 , accepted 2 March, 1995)

HRTEM JEM- 200CX which is settled at Department of Geology, Faculty of Science, Niigata University and we often carry out identification of fine grained minerals in TEM, and carry out observation of fine structures of rock forming minerals and have HRTEM images. For these purposes, the following necessary softwares are, for example, listed;

1 Programs for identification of fine grained minerals in TEM size
  Calculation of possible indexing for electron diffraction (ED) spots
  (Calculation of angles between two reciprocal lattice points < diffraction spots>)
  ED pattern simulation program
2 HRTEM image simulation programs
3 Crystal shape drawing of fine mineral grains in TEM observation
4 Crystallographic direction analysis of two or more mineral grains which coexist maintaining crystallographic relationship
5 Lattice parameter calculation in case of ED pattern using least square method
6 Display of crystal structure : static displaying and/or moving images
7 Related programs for basic mineralogical calculation

Some of them or similar types of programs developed independently have already been prepared but combining them as a whole may be more convenient and more usefull in practice.

Among the above listed programs, we briefly report the programs developed at this time and demonstrate the usage (application) of these programs in this paper,.

Programs for identification of fine mineral grains

Programs for identification of fine mineral grains in TEM size which can be said as ED pattern interpretation program in another word are composed of the following parts;

Calculation of possible indexing for electron diffraction (ED) spots
(Calculation of angles between two reciprocal lattice points
<diffraction spots>) (*d* value calculation)
ED pattern simulation program

Processing data of SAED (Selected Area Electron DIffraction) pattern which can be easily obtained in TEM analysis may enable identification of minute minerals in TEM order scale can be identified.

The followings are procedures to identify the minerals in TEM

(1) To obtain AEM spectra

Listing up possible mineral species.

Also taking into crystal habit (shapes, size, paragenesis, ,,,)

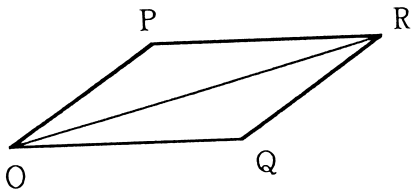

Fig.1. Fundamental elements in ED pattern. O represents origin. P,Q and R are diffraction spots.

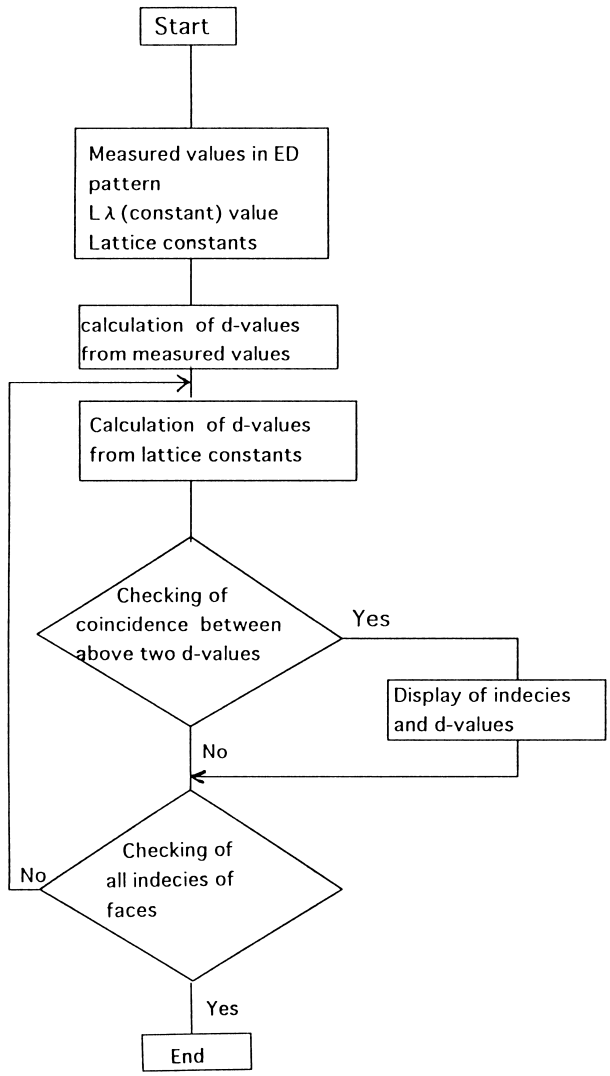

Fig.2. Flow chart of indexing program for ED pattern.

(2) To obtain ED pattern
Measuring necessary distances (and angles), d-values can be easily calculated
(3) To caluculate possible indices of diffraction spots in ED pattern
Testing on first possible mineral
(4) To check matching of d-values and angles between spots
OK or NOT
(5) Testing second possible mineral
OK or NOT → continue
(6) Confirmation of identification and handling the pattern in more detail by rotation and tilting of specimens etc.
The followings are the programs corresponding to each part of this procedure.

#1 **Indexing program for ED pattern**

Fundamental situation of ED pattern is shown in Fig.1.
Necessary numerical data for this program are
1 distances OP , OQ and OR
(angles ; $\angle$ POR and $\angle$ POQ) *
3 $L\lambda$ value : camera length L multiplied by wave length of electron beam λ
4 lattice constants of the testing minerals

Fig.2 shows the flow chart of this procedure. Here, we demonstrate the examples of usage of this program: at ED pattern , measuring diffraction pattern components, OP , OQ and OR values (for example, OP = 2.2mm ,OQ = 3.8mm , OR = 4.4mm), are obtined.

Fig.3 shows example of Input data for olivine and Fig.4 shows Output examples for this input data.

The program list whose comments were originally written in Japanese is shown in Appendix 1.

---

* Program for caluculation of angles between two reciprocal lattice points< diffraction spots> is contained in program #1 but this part can also be used independently to check the ED pattern analysis. This part picked up will become subprogram of basic calculation.
Necessary data for calculation are
1 lattice constants of the testing minerals
2 two indices of the two spots in the ED pattern
Comparing the obtained angle value and the observed value identification procedure proceeds.

```
Input the distances OP OQ OR ( in Fig. 1)
      ：ＯＰ，ＯＱ，ＯＲの値（距離）を入力して下さい

        OP=2.2
        OQ=3.8
        OR=4.4

Input Lλ value (camera length X wave length of electron
      beam)  ：Ｌλ値（カメラ長　Ｘ　波長）を入力し
   て下さい
          Lλ= 22.5
Input lattice constants   ：格子定数を入力してください
          a=10.195
          b=5.981
          c=4.756
          α = 90.0
          β =90.0
          γ =90.0
```

Fig.3. Input data form for indexing of ED pattern obtained from olivine.

```
(1) The following possibilities are present：次の条件で
可能性有り
P:(-1  0  0)  dp= 10.195
Q:(0  -1  0)  dq = 5.981
R:(-1  -1  0) dr = 5.159

(2) The following possibilities are present：次の条件で
可能性有り
P:(-1  0  0)  dp= 10.195
Q:(0  1  0)  dq = 5.981
R:(-1  1  0) dr = 5.159

(3) The following possibilities are present：次の条件で
可能性有り
P:(1  0  0)  dp= 10.195
Q:(0  -1  0)  dq = 5.981
R:(1  -1  0) dr = 5.159

(4) The following possibilities are present：次の条件で
可能性有り
P:(1  0  0)  dp= 10.195
Q:(0  1  0)  dq = 5.981
R:(1  1  0) dr = 5.159
```

Fig.4. Output form for the data of Fig.3 in ED pattern indexing.

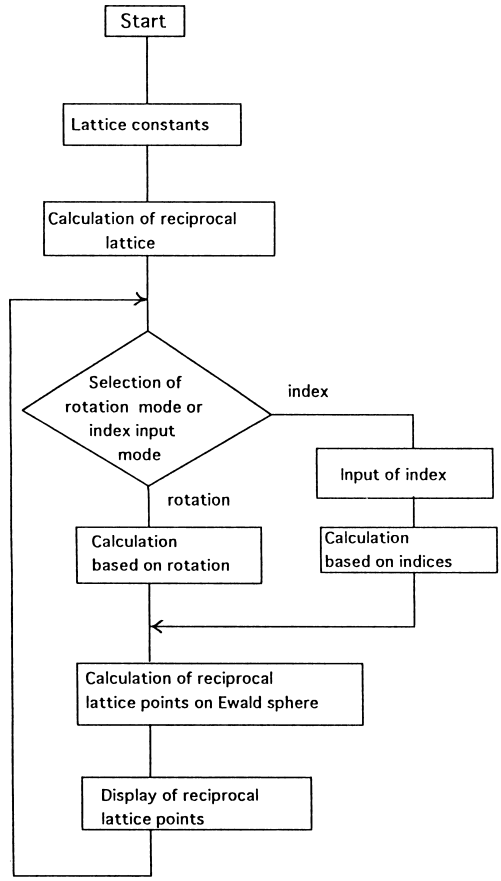

Fig.5. Flow chart of ED pattern simulation program

#2 ED pattern simulation program

Identified ED pattern which is the same as reciprocal lattice of the mineral can be graphically represented. The program now developed runs on SHARP X 68030: the ED pattern figure can rotate around 2 rotationary axes; Using 2 and 8 in ten key, the figure rotates around first axis. Using 4 and 6 on ten key the figure roatates around second axis. Rotation step is by 0.001 degrees by each. If 0 is typed in on ten key, it will become Ewald sphrere mode . In this case, 2 indicies are needed to define the reciprocal plane.

The flow chart of this program is shown in Fig.5. This program is composed of three subprograms; Recipr.C , CLS.S and KEYIN.S. CLS.S and KEYIN.S. are written in assembly language, so, only program list of Recipr.C which was originally written in Japanese is shown in Appendix 2.

Fig.6 shows the demonstration of Input form for olivine structure and Fig.7a,b,c and d show

```
Input lattice constants:  格子定数を入力してください

a=  10.195
b=  5.981
c=  4.756
  α =  90.0
  β=  90.0
  γ = 90.0
```

```
Input  2 vectors defining  reciplocal plane
( Plane  approximation of Ewald sphere )
        ：逆格子面を定義する２つのベクトル（面指数）を
       入力して下さい
Input  vector 1  (if hk0  plane is defined, input  100  )
        ：ベクトル1の入力（ｈｋ面を表示する場合（100）
     を入力します
h=1
k=0
l=0

Input vector 2 (if hk0  plane is defined, input  010  )
        ：ベクトル2の入力（ｈｋ面を表示する場合（010）
     を入力します
h=0
k=1
l=0
```

Fig.6. Input data form of olivine structure for ED pattern simulation.

examples of Output forms which represent a series of rotated images.

HRTEM image simulation programs

HRTEM image is formed through double Fourier Transform of real structure. To interprete HRTEM or structure image, image simulation is often essential. It is necessary to design programs based on dynamical theory for precise image calculation taking into acount of specimen thickness. Some programs for these purposes are already prepared(Ishizuka and Uyeda, 1977; Ishizuka, 1980; Ishizuka, 1982; O'keefe and Sanders ,1975 ; O'keefe and Buseck, 1979) . For example, MacHREM wirtten by Ishizuka(1993) is one of excellent examples and is now on sale.

For these programs, important Input data needed are (1) unit cell parameters, (2) atomic parameters (atomic scattering factors, atomic coordiantions, isotropic thermal factors, and occupancy factors) , (3) symmetry operations, (4) spherical aberration coefficient, (5) defocus values, (6) aperture size, (7) envelop function parameters (partial coherence) , (8) specimen orientation, (9) incident beam direction, (10) wave length of electron beam, (11) specimen thickness and (12) half tone

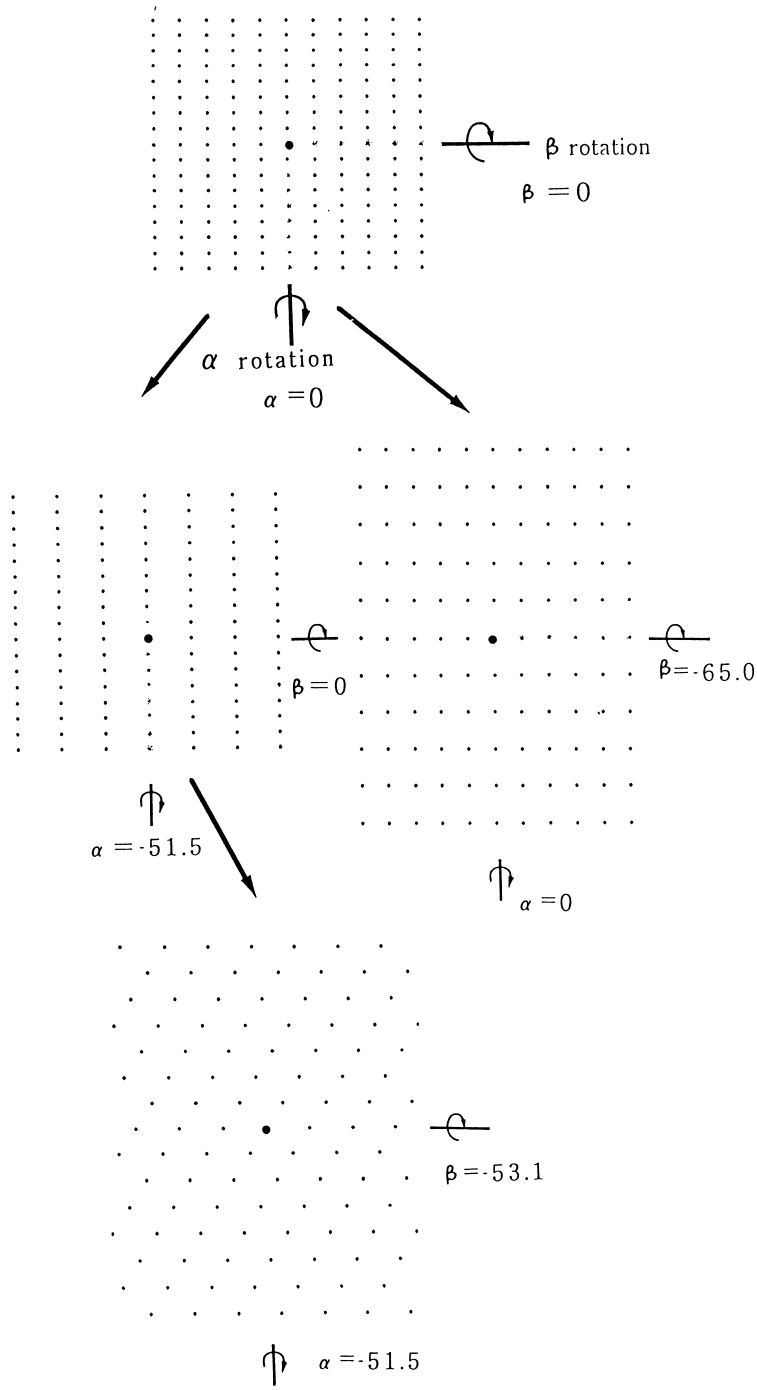

Fig.7. Output examples of ED pattern simulation for the data of Fig.6. a,b,c and d represent only several figures in rotating images.

levels, and so on.

Image calculation program based on kinematical theory is also more fundamental one, and can be used as an ideal case or as the first approximation. For kinematical calculations, needed input data are also the same for multislice calculation except specimen thickness. For example, Fortran program, EMI (Electron Microscopic Image with Abberation) developed by K. Ishizuka was one of typical programs based on kinematical theory and could be used by his kindness at this time. In this study, we used his program by transforming the program written in Fortran language into personal computer program, and calculated typical images of standard mineral structures and carried out some unknown structure estimation.

Image simulaiton for structure estimation can be carried out as follows;

(1) Determination of chemical composition

(2) To take HRTEM images of through focus series for thin specimen

(3) Construction of Structure model

(4) Calculation of TEM image simulation

(5) Comparison between TEM image and simulation image

(6) Change of model and recaluculation

In this programs many parameters are needed as mentioned above and these are listed up as defining file style.

Testing of image simulations for ideal known structures .

Here, only the results are shown. Results tested for known structures(serpentine, talc and olivine) are shown in Figs. 8, 9, 10 and 11. Different defocus images for serpentine structure are shown in Fig. 8. Images for olivine structure and talc structure were also calculated as 1 dimensional and 2 dimensional images. Figs. 9, 10, 11 show Scherzer focus images (under focus of 107nm; cf Akai (1987)) of 1 and 2 dimensional one for serpentine, talc and olivine structure. These results suggest resolution limits and image characteristics to be obtained.

Trial of estimation for unknown structure .

Testing for unknown structure as first approximataion was carried out

: Intermediate structure in transformation from serpentine to olivine through thermal metamorphism was examined. This mineral was found in Antarctic carbonaceous chondrite Yamato-793321(Akai, 1984, 1987, 1988, 1990, 1992, Akai, 1994). On the other hand, artificial thermal changes of serpentine minerals have been investigated (Brindley and Zussman, 1957; Ball and Taylor, 1963; and Brindley and Hayami, 1965; Akai, 1990, 1992, etc.)

As the first approximation, model of structure being very similar to olivine structure with some vacant sites was considered, because in HRTEM image, new periodicities of about 10 - 13 A are often observed and this can be due to some superstructure in pseudo-olivine structure.

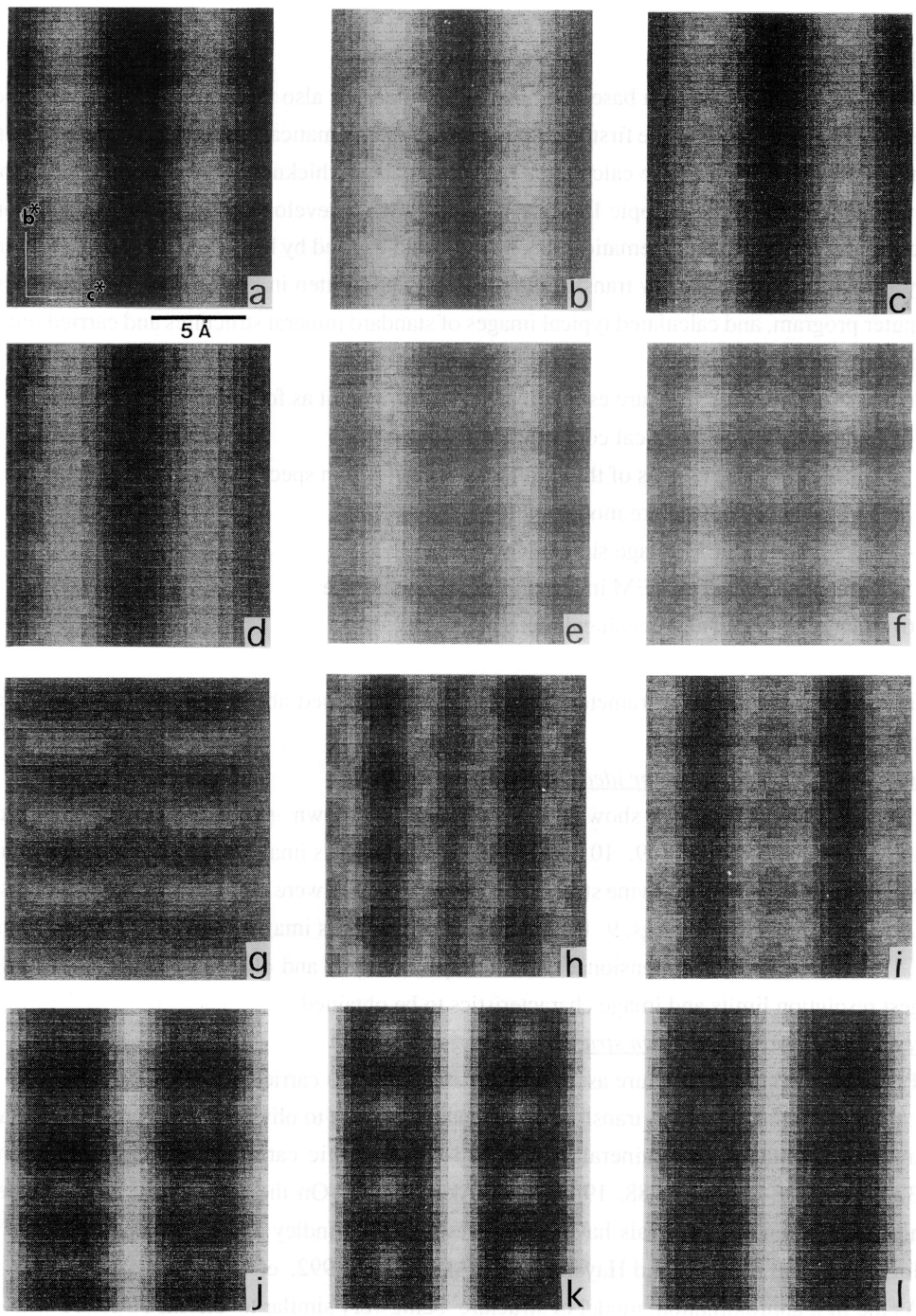

Fig.8. Simulated images with different defocus values for serpentine calculated. Negative sign indicates over focus. (2 dimensional images). a (- 100nm : < over focus >), b (-80nm), c (-60 nm), d(-40 nm), e (-20 nm), f (0 nm : < just focus >), g (+ 20 nm),h (+40 nm), i (+60 nm), j (+80 nm), k (+100 nm), l (+120 nm : < under focus >)

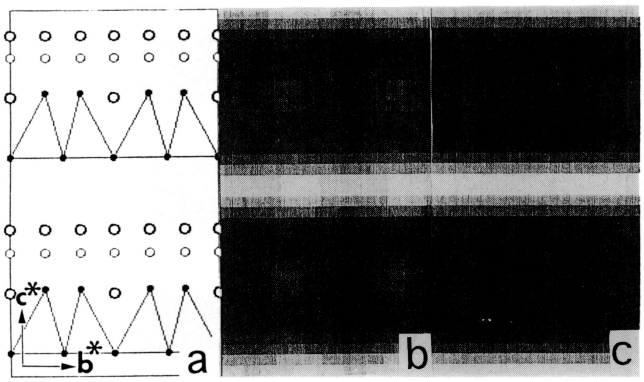

Fig.9. Simulated Scherzer focus images (107nm underfocus)for serpentine structure.
a : serpentine structure model
b : 2 dimensional image
c : 1 dimensional image

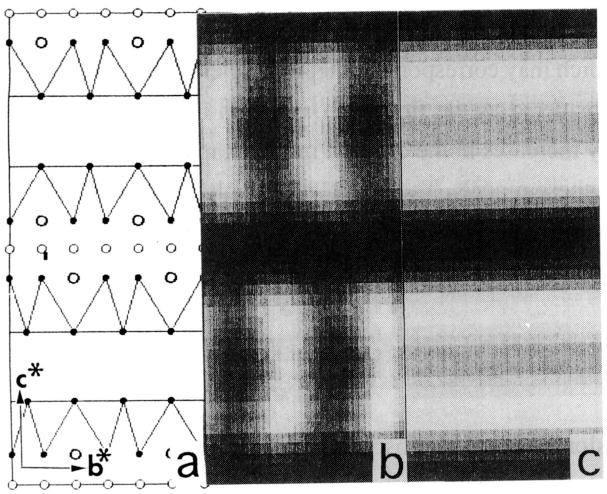

Fig.10. Simulated Scherzer focus images (107nm underfocus) for talc structure
a : talcc structure model
b : 2 dimensional image
c : 1 dimensional image

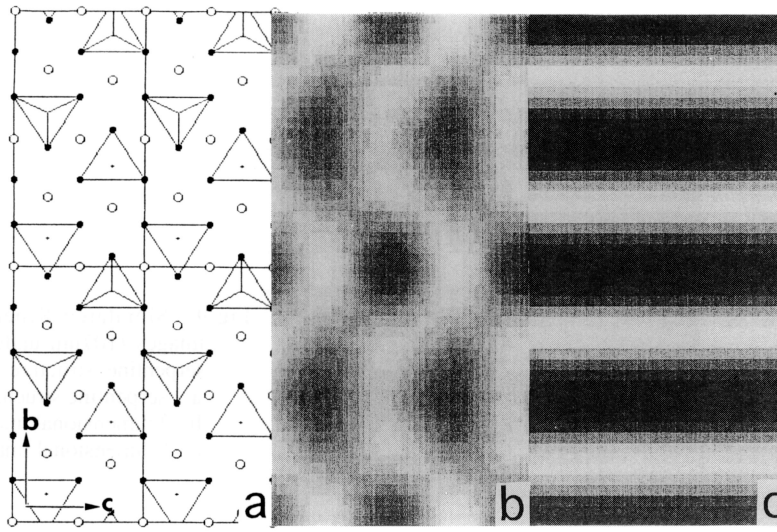

Fig.11. Simulated Scherzer focus images (107nm underfocus) for olivine structure.
a : olivine structure model
b : 2 dimensional image
c : 1 dimensional image

The first approximation models are shown in Fig. 12a and image simulations corresponding to them are also shown in Figs. 12b, c and d. Figs. 12b, c and d correspond to three models (model 1, model 2 and model 3) with slightly different cation occupancies. Model 1 indicates a structure with vacant site in M2 site of olivine structure, which may correspond to metal deficiency. In this modeling no precise structural construction is considered as the first approximation but only effect by cation deficiency is cheked. Cation deficiency increases in the order of model 1, model 2 and model 3. About 10 A periodicities really appear in these models. The effect is strongly found in model 3 than in model. The other image contrast, however, does not always well coincide with the HRTEM iamges but the whole result may suggest some cation deficiency structure.

Summary

1. Personal computer programs for Transmission Electron Microscopy in Mineralogy; (that is, Mineral identification, ED pattern simulation) were newly written.
2. Mineral identification program can be conveniently used for interpretation and identification of ED pattern of mineral.
3. ED pattern simulation program can be used for checking ED pattern by handling ED pattern by

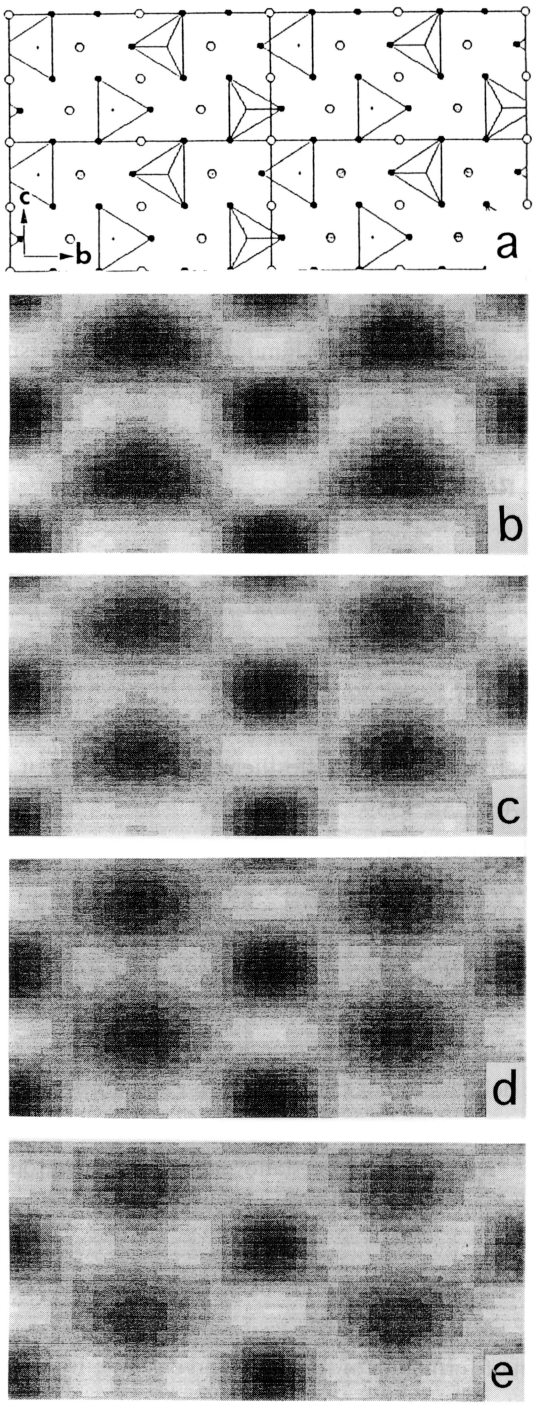

Fig.12 Simulated HRTEM image for model structure of defect olivine structures with cation vacancies, which are the first approximation model structures of intermediate structure in transformation from serpentine to olivine.
a: model structure with vacant sites in different quantities in olivine structure,
b: 2 dimensional image corresponding to original olivine structure,
c: calculated image for model 1 (25% of M2 is vacant),
d: calculated image for model 2 (50% of M2 is vacant),
e: calculated image for model 3 (75 % of M2 is vacant).

rotating it.

4. HRTEM image simulation program based on kinematical theory written by Ishizuka was used on personal computer by transforming the original program to personal computer usage. At first, basic checks to calculate HRTEM images of known structures were carried out.

5 . Estimation of unknown structure which was found in thermally metamorphosed carbonaceous chondrite was carried out as the first approximation.

Acknowledgments

We acknowledge Dr. Kazuo Ishizuka for his kindly giving us permission to use TEM image simulation program. We also express our thanks to Prof. T. Yoshimura of Niigata University for his encouragements and critical reading of this manuscript.

References

Akai, J., 1984, Mineralogical characterization of matrix materials in carbonaceous chondrite, Yamato-793321 and Belgica-7904. *Paper presnted to 9th Symp. Antarct. Meteor.* 59.

———, 1987, Recent progress in characterization of phyllosilicates and related minerals in meteorites by electron microscope. *Journal of Clay Science Society of Japan* (Nendo Kagaku), **27**, 104 (in Japanese).

———, 1988, Incompletely transformed serpentine-type phyllosilicates in the matrix of Antarctic CM chondrites. *Geochim. Cosmochim, Acta*, **52**, 1539.

———, 1990, Mineralogical evidence of heating events in Antarctic carbonaceous chondrites Y-86720 and Y- 82162. *Proc. NIPR Symp. Antarctic Meteor.*, **3**, 55.

———, 1992, T-T-T diagram of serpentine and estimation of metamorphic heating degree of Antarctic carbonaceous chondrites *Proc. NIPR Symp. Antarc. Meteor.*, **5**, 120.

——— and Sekine, T., 1994, Shock effect experiments on serpentine and thermal metamorphic conditions in Antarctic carbonaceous chondrite. *Proc. NIPR Symp. Antarc. Meteor.*, **7**, 101.

Ball, M.C. and Taylor, H.F.W., 1963, The dehydration of chrysotile in air and under hydrothermal conditions. *Mineral. Mag.*, **33**, 467.

Boisen, M. B. Jr. and Gibbs, G. V., 1985, Mathematical crystallography — An Introduction to mathematical foundations of crystallography. *Reviews in Mineralogy* volume 15, Min. Soc. Amer.

———, Brindley, G.W. and Zussman , J., 1957, Structural study of the mineral transformation of serpentine minerals to forsterite. *Amer. Mineral.*, **42**, 461.

——— and Hayami, R., 1965, Mechanism of formation of forsterite and enstatite from serpentine. *Mineral. Mag.*, **35**, 189.

Horiuchi, S., 1988, *High Resolution Electron Microscopy — principles and its application* . (in Japanese) Kyoritsu Shuppan.

Ishizuka, K., 1980, Contrast transfer of crystal image in TEM. *Ultramicroscopy*, **5**, 55.

———, 1982, Multislice formula for inclined crystals. *Acta Cryst.* **A38**, 773.

———, and Uyeda N., 1977, A new theoretical and practical approach to the multislice method. *Acta Cryst*, **A33**, 740.

Jackson, A. G., 1988, *Handbook of crystalography for electron microscopist and others*. Springer-Verlag.

O'keefe, M. A. and Sanders, V. J., 1975, n-beam lattice images VI. Degradation of image resolution by combination of incident beam divergence and spherical abberation. *Acta Cryst.* **A31**, 307.

———, and Buseck, P. R., 1979, Computation of high resolution TEM images of minerals. *Trans. Amer. Cryst. Assoc.*, **15**, 27.

Wenk, H. -R. (ed), 1976, *Electron Microscopy in Mineralogy*. Springer-Verlag.

Appendix 1

The program list of indexing program of ED pattern, whose comments were originally written in Japanese

```
#include <stdio.h>
#include <math.h>

#define sgn(x) (((x)<0.0) ? -1.0 : 1.0)

/* 浮動小数点ゼロ判定 */
#define Zero 0.0001
#define DZero 0.1
#define PMAX 5
#define PI 3.141592653589

typedef struct {
    float X;
    float Y;
    float Z;
} VECTOR;

/* グローバル変数 */
/* 格子定数 */
float a,b,c,alpha,beta,gamma;
/* 平行六面体の体積 */
float ABC;

/* 3点の距離 */
float LenP,LenQ,LenR;
float DP,DQ,DR;

/* カメラ長×電子線波長 */
float Lrda;

/* 実格子ベクトル */
VECTOR A = { 0.0,0.0,0.0 };
VECTOR B = { 0.0,0.0,0.0 };
VECTOR C = { 0.0,0.0,0.0 };
/* 逆格子ベクトル */
VECTOR AStar = { 0.0,0.0,0.0 };
VECTOR BStar = { 0.0,0.0,0.0 };
VECTOR CStar = { 0.0,0.0,0.0 };

/* 投影平面決定のための格子点の位置ベクトル */
VECTOR LP1 = { 1.0,0.0,0.0 };
VECTOR LP2 = { 0.0,1.0,1.0 };


void GAISEKI(VECTOR *AB,VECTOR *A,VECTOR *B,float C)
{
    AB->X = ((A->Y*B->Z)-(A->Z*B->Y))/C;
    AB->Y = ((A->Z*B->X)-(A->X*B->Z))/C;
    AB->Z = ((A->X*B->Y)-(A->Y*B->X))/C;
}

float KAKUDO(VECTOR *A,VECTOR *B)
{
    float AxB,AB;
    /* 外積の絶対値 */
    AxB = sqrt(
            ((A->Y*B->Z)-(A->Z*B->Y))*((A->Y*B->Z)-(A->Z*B->Y))
          + ((A->Z*B->X)-(A->X*B->Z))*((A->Z*B->X)-(A->X*B->Z))
          + ((A->X*B->Y)-(A->Y*B->X))*((A->X*B->Y)-(A->Y*B->X))
        );
    /* 内積 */
    AB = (A->X*B->X)+(A->Y*B->Y)+(A->Z*B->Z);
    if(fabs(AB)<Zero) return sgn(AB)*PI/2.0;
    return atan( AxB / AB );
}


void main(void)
{
    int PH,PK,PL;
    int QH,QK,QL;
    int RH,RK,RL;
    int number=0;
    float u,v,w;
    float dpr,dqr,drr;
    float sqr;

    printf("面指数計算プログラム¥n");
    printf("¥n");
    printf("OP,OQ,ORそれぞれの値を入力して下さい¥n");
    printf("OP = ");
    scanf("%f",&LenP);
    printf("OQ = ");
    scanf("%f",&LenQ);
    printf("OR = ");
    scanf("%f",&LenR);
    printf("¥n");
    printf("(カメラ長×電子線波長)を入力してください。¥n");
    printf("Lλ = ");
    scanf("%f",&Lrda);
    printf("¥n");
    printf("格子定数を入力してください。¥n");
    printf("a = ");
    scanf("%f",&a);
    printf("b = ");
    scanf("%f",&b);
    printf("c = ");
    scanf("%f",&c);
    printf("α= ");
    scanf("%f",&alpha);
```

```
    printf("β= ");
    scanf("%f",&beta);
    printf("γ= ");
    scanf("%f",&gamma);

    printf("¥n");

    /* 角をラジアンに変換する */
    alpha = alpha*PI/180.0;
    beta  = beta *PI/180.0;

    gamma = gamma*PI/180.0;


/* 格子定数から直交座標系3ベクトルを計算する */

    /* ベクトルA の計算 */
    A.X=a;
    A.Y=0.0;
    A.Z=0.0;
    /* ベクトルB の計算 */
    B.X=b*cos(gamma);
    B.Y=b*sin(gamma);
    B.Z=0.0;
    /* ベクトルC の計算 */
    C.X=cos(beta);
    C.Y=((-cos(beta)+cos(gamma+alpha))/tan(gamma)+sin(gamma+alpha));
    C.Z=sqrt(1.0 -C.X*C.X -C.Y*C.Y);
    C.X=c*C.X;
    C.Y=c*C.Y;
    C.Z=c*C.Z;

/* 計算した3ベクトルから逆格子ベクトルを計算する */
    /* スカラー三重積 の計算 */
    ABC = A.X*(B.Y*C.Z-B.Z*C.Y)
        + A.Y*(B.Z*C.X-B.X*C.Z)
        + A.Z*(B.X*C.Y-B.Y*C.X);

    /* 逆格子ベクトルの計算 */
    GAISEKI(&AStar,&B,&C,ABC);
    GAISEKI(&BStar,&C,&A,ABC);
    GAISEKI(&CStar,&A,&B,ABC);

    /* 3点のd値を計算 */
    DP = Lrda/LenP;
    DQ = Lrda/LenQ;
    DR = Lrda/LenR;

    for(PH=-PMAX ; PH<=PMAX ; PH++){
      for(PK=-PMAX ; PK<=PMAX ; PK++){
        for(PL=-PMAX ; PL<=PMAX ; PL++){
          u = AStar.X*(float)PH
            + BStar.X*(float)PK
            + CStar.X*(float)PL;
          v = AStar.Y*(float)PH
            + BStar.Y*(float)PK
            + CStar.Y*(float)PL;
          w = AStar.Z*(float)PH
            + BStar.Z*(float)PK
            + CStar.Z*(float)PL;
          sqr = sqrt(u*u+v*v+w*w);
          if(sqr<Zero) continue;
          dpr = 1.0/sqr;
          if(fabs(DP-dpr) > DZero) continue;
          for(QH=-PMAX ; QH<=PMAX ; QH++){
            for(QK=-PMAX ; QK<=PMAX ; QK++){
              for(QL=-PMAX ; QL<=PMAX ; QL++){
                u = AStar.X*(float)QH
                  + BStar.X*(float)QK
                  + CStar.X*(float)QL;
                v = AStar.Y*(float)QH
                  + BStar.Y*(float)QK
                  + CStar.Y*(float)QL;
                w = AStar.Z*(float)QH
                  + BStar.Z*(float)QK
                  + CStar.Z*(float)QL;
                sqr = sqrt( u*u + v*v + w*w );
                if(sqr<Zero) continue;
                dqr = 1.0 / sqr;
                if(fabs(DQ-dqr) > DZero) continue;
                RH=PH+QH;
                RK=PK+QK;
                RL=PL+QL;
                if(RH==0 && RK==0 && RL==0) continue;
                u = AStar.X*(float)RH
                  + BStar.X*(float)RK
                  + CStar.X*(float)RL;
                v = AStar.Y*(float)RH
                  + BStar.Y*(float)RK
                  + CStar.Y*(float)RL;
                w = AStar.Z*(float)RH
                  + BStar.Z*(float)RK
                  + CStar.Z*(float)RL;
                sqr = sqrt(u*u+v*v+w*w);
                if(sqr<Zero) continue;
                drr = 1.0/sqr;
                if(fabs(DR-drr) > DZero) continue;
                printf("(%d)次の条件で可能性があります。¥n",++number);
                printf("P : ( %d %d %d ) dp = %f ¥n",PH,PK,PL,dpr);
                printf("Q : ( %d %d %d ) dq = %f ¥n",QH,QK,QL,dqr);
                printf("R : ( %d %d %d ) dr = %f ¥n",RH,RK,RL,drr);
                printf("¥n");
              }
            }
          }
        }
      }
    }
    if(number==0) printf("該当する面指数条件はありませんでした。¥n");
}
```

Appendix 2

The program list of ED pattern simulation program, whose comments were originally written in Japanese

```
#include <stdio.h>
#include <math.h>
#include <basic0.h>
#include <graph.h>

#define sgn(x) (((x)<0.0) ? -1.0 : 1.0)

/* プロトタイプ宣言 */
void CLS(void);  /* 高速画面消去 */
int KeyIN(void); /* 高速キー入力 */

/* 逆格子点の数 */
#define NuMbEr 16
/* 浮動小数点ゼロ判定 */
#define Zero 0.0001
#define SCALE 200.0
#define HV 4.0/3.0
#define Hankei 0.01

typedef struct {
    float X;
    float Y;
    float Z;
} VECTOR;

/* グローバル変数 */
/* 格子定数 */
float a,b,c,alpha,beta,gamma;
/* 平行六面体の体積 */
float ABC;
/* 回転角度 */
float Halpha,Hbeta;
/* 回転角度指定モード用回転角度 */
float Palpha,Pbeta;
/* 逆格子点直交座標 */
float GX,GY,GZ;
/* ディスプレイ座標 */
float PX,PY,PZ;

/* 実格子ベクトル */
VECTOR A = { 0.0,0.0,0.0 };
VECTOR B = { 0.0,0.0,0.0 };
VECTOR C = { 0.0,0.0,0.0 };
/* 逆格子ベクトル */
VECTOR AStar = { 0.0,0.0,0.0 };
VECTOR BStar = { 0.0,0.0,0.0 };
VECTOR CStar = { 0.0,0.0,0.0 };

/* 投影平面（エワルド球）の法線ベクトル */
VECTOR Housen = { 0.0,0.0,1.0 };
/* 投影平面決定のための格子点の位置ベクトル */
VECTOR LP1 = { 1.0,0.0,0.0 };
VECTOR LP2 = { 0.0,1.0,1.0 };

/* 法線ベクトルの傾きを求めるためのワークベクトル */
VECTOR WHou = { 0.0,0.0,0.0 };

/* ディスプレイ画面法線ベクトル */
VECTOR Display = { 0.0,0.0,1.0 };

/* 座標変換回転行列 */
float YZ[4];
float ZX[4];

/* 逆格子点テーブル */
char lattice[NuMbEr+1][NuMbEr+1][NuMbEr+1];
/* 投影逆格子点テーブル */
char project[NuMbEr+1][NuMbEr+1][NuMbEr+1];

/* 逆格子点座標テーブル */
float pointX[NuMbEr+1][NuMbEr+1][NuMbEr+1];
float pointY[NuMbEr+1][NuMbEr+1][NuMbEr+1];
float pointZ[NuMbEr+1][NuMbEr+1][NuMbEr+1];


/* 画面表示 */
void PLOT(void)
{
    int h,k,l,m;
    int col;
    char *PR;
    m=NuMbEr/2;
    PR=project;
    CLS();
    for(h=0;h<=NuMbEr;h++){
        for(k=0;k<=NuMbEr;k++){
            for(l=0;l<=NuMbEr;l++){
                if((*PR++)==0) continue;
                GX = pointX[h][k][l];
                GY = pointY[h][k][l];
                GZ = pointZ[h][k][l];
                /* YZ面をX軸回転 */
                PY = GY*YZ[0] + GZ*YZ[1];
                PZ = GY*YZ[2] + GZ*YZ[3];
                GY = PY;
```

```
                GZ = PZ;
                /* ZX面をY軸回転 */
                PX = GX*ZX[0] + GZ*ZX[1];
                PZ = GX*ZX[2] + GZ*ZX[3];
                GX = PX;
                GZ = PZ;
                GY = GY*HV;
                /* プロットカラーの選択 */
                col= 0xffff;
                if( h>m && k==m && l==m) col&=0x003f;
                if(h==m &&  k>m && l==m) col&=0x07c1;
                if(h==m && k==m &&  l>m) col&=0xf801;
                if(h==m && k==m && l==m) col =0xf83f;
                circle(GX+256,-GY+256,
                       (0.011-fabs(GZ))*300,

                       col,0,360,341
                );
                paint(GX+256,-GY+256,col);
            }
        }
    }
}

void GAISEKI(VECTOR *AB,VECTOR *A,VECTOR *B,float C)
{
    AB->X = ((A->Y*B->Z)-(A->Z*B->Y))/C;
    AB->Y = ((A->Z*B->X)-(A->X*B->Z))/C;
    AB->Z = ((A->X*B->Y)-(A->Y*B->X))/C;
}

float KAKUDO(VECTOR *A,VECTOR *B)
{
    float AxB,AB;
    /* 外積の絶対値 */
    AxB = sqrt(
          ((A->Y*B->Z)-(A->Z*B->Y))*((A->Y*B->Z)-(A->Z*B->Y))
        + ((A->Z*B->X)-(A->X*B->Z))*((A->Z*B->X)-(A->X*B->Z))
        + ((A->X*B->Y)-(A->Y*B->X))*((A->X*B->Y)-(A->Y*B->X))
        );
    /* 内積 */
    AB = (A->X*B->X)+(A->Y*B->Y)+(A->Z*B->Z);
    if(fabs(AB)<Zero) return sgn(AB)*PID2;
    return atan( AxB/AB );
}

/* 回転行列の作成 */
void ROTATION(float *ROT,float theta)
{
    ROT[0] = cos(theta);
    ROT[1] = -sin(theta);
    ROT[2] = sin(theta);
    ROT[3] = cos(theta);
}

/* 消滅則を考慮した逆格子点テーブルの作成 */
void GYAKUKOUSI(void)
{
    int h,k,l,m;
    m=NuMbEr/2;
    for(h=-m;h<=m;h++){
      for(k=-m;k<=m;k++){
        for(l=-m;l<=m;l++){
          /* ここに消滅則処理を入れる */
          lattice[h+m][k+m][l+m]=1;
          pointX[h+m][k+m][l+m] = h*AStar.X+k*BStar.X+l*CStar.X;
          pointY[h+m][k+m][l+m] = h*AStar.Y+k*BStar.Y+l*CStar.Y;
          pointZ[h+m][k+m][l+m] = h*AStar.Z+k*BStar.Z+l*CStar.Z;
        }
      }
    }
}

/* 平面にのる逆格子点テーブルの作成 */
void EWLD(struct VECTOR *Housen)
{
    int h,k,l,m;
    float HX,HY,HZ;
    float atai;
    float *p,*q,*r;
    char *PR,*LA;
    m=NuMbEr/2;
    HX=Housen->X;
    HY=Housen->Y;
    HZ=Housen->Z;
    p=pointX;
    q=pointY;
    r=pointZ;
    PR=project;
    LA=lattice;
    for(h=0;h<=NuMbEr;h++){
      for(k=0;k<=NuMbEr;k++){
        for(l=0;l<=NuMbEr;l++){
          atai = (HX)*(*p++)+(HY)*(*q++)+(HZ)*(*r++);
          if(fabs(atai)<Hankei){
              *PR=*LA;
          }else{
              *PR=0;
          }
          PR++;
          LA++;
        }
      }
    }
}

void vectorHousen(void)
{
    float u,v,w;
    /* 投影平面を決定する２ベクトルを入力する */
    printf("¥n");
    printf("(エワルド球)平面を定義する２ベクトル(面指数)を入力してください。¥n");
    printf("¥n");
    printf("ベクトル１の入力 (h-k平面を表示する場合 (100) を入力します)¥n");
    printf("    h = ");
```

```
    scanf("%f",&LP1.X);
    printf("    k = ");
    scanf("%f",&LP1.Y);
    printf("    l = ");
    scanf("%f",&LP1.Z);
    printf("¥n");
    printf("ベクトル２の入力 (h-k平面を表示する場合 (010) を入力します)¥n");
    printf("    h = ");
    scanf("%f",&LP2.X);
    printf("    k = ");
    scanf("%f",&LP2.Y);

    printf("    l = ");
    scanf("%f",&LP2.Z);

    /* (仮)ベクトルを実際の直交座標におけるベクトルへ変換する */

    u = LP1.X;
    v = LP1.Y;
    w = LP1.Z;
    LP1.X = u*AStar.X + v*BStar.X + w*CStar.X;
    LP1.Y = u*AStar.Y + v*BStar.Y + w*CStar.Y;
    LP1.Z = u*AStar.Z + v*BStar.Z + w*CStar.Z;

    u = LP2.X;
    v = LP2.Y;
    w = LP2.Z;
    LP2.X = u*AStar.X + v*BStar.X + w*CStar.X;
    LP2.Y = u*AStar.Y + v*BStar.Y + w*CStar.Y;
    LP2.Z = u*AStar.Z + v*BStar.Z + w*CStar.Z;

    /* 投影する平面 (エワルド球) の法線ベクトルを求める */
    GAISEKI(&Housen,&LP1,&LP2,1.0);

    /* 法線ベクトルがひっくり返っている時の処理 */
    if(Housen.Z < 0.0){
        Housen.X=-Housen.X;
        Housen.Y=-Housen.Y;
        Housen.Z=-Housen.Z;
    }

    /* 法線ベクトルと(001)との傾きを２成分で計算する */
    /* ワークベクトルの設定 */
    WHou.X = 0.0;
    WHou.Y = Housen.Y;
    WHou.Z = Housen.Z;
    /* Halpha = (001)^(0vw) の計算 */
    Palpha = KAKUDO(&Display,&WHou)*sgn(Housen.Y);
    /* Hbeta  = (0vw)^(uvw) の計算 */
    Pbeta  = KAKUDO(&WHou,&Housen)*sgn(Housen.X);

}

void main(void)
{
    int K;
    screen(1,3,1,1);
```

```
    printf("逆格子像表示プログラム¥n");
    printf("¥n");
    printf("格子定数を入力してください。¥n");
    printf("a = ");
    scanf("%f",&a);
    printf("b = ");
    scanf("%f",&b);
    printf("c = ");
    scanf("%f",&c);
    printf("α= ");
    scanf("%f",&alpha);
    printf("β= ");
    scanf("%f",&beta);
    printf("γ= ");
    scanf("%f",&gamma);

    /* 度をラジアンに変換する */
    alpha = alpha*PI/180.0;
    beta  = beta *PI/180.0;
    gamma = gamma*PI/180.0;


/* 格子定数から直交座標系３ベクトルを計算する */

    /* ベクトルA の計算 */
    A.X=a;
    A.Y=0.0;
    A.Z=0.0;
    /* ベクトルB の計算 */
    B.X=b*cos(gamma);
    B.Y=b*sin(gamma);
    B.Z=0.0;
    /* ベクトルC の計算 */
    C.X=cos(beta);
    C.Y=((-cos(beta)+cos(gamma+alpha))/tan(gamma)+sin(gamma+alpha));
    C.Z=sqrt(1.0 -C.X*C.X -C.Y*C.Y);
    C.X=c*C.X;
    C.Y=c*C.Y;
    C.Z=c*C.Z;

/* 計算した３ベクトルから逆格子ベクトルを計算する */
    /* スカラー三重積 の計算 */
    ABC = A.X*(B.Y*C.Z-B.Z*C.Y)
        + A.Y*(B.Z*C.X-B.X*C.Z)
        + A.Z*(B.X*C.Y-B.Y*C.X);
    ABC = ABC / SCALE;

    /* 逆格子ベクトルの計算 */
    GAISEKI(&AStar,&B,&C,ABC);
    GAISEKI(&BStar,&C,&A,ABC);
    GAISEKI(&CStar,&A,&B,ABC);


/* 消滅則を考慮した逆格子点テーブルの作成 */
    GYAKUKOUSI();
```

```
/* 変数初期化 */
    Palpha=0.0*PI/180.0;
    Pbeta=0.0*PI/180.0;

/* mainループ */
    for(;;){

/* キー入力 */
        while((K=KeyIN())==0);

        if(K=='2') Palpha=Palpha-PI/180.0/1000.0;
        if(K=='4') Pbeta=Pbeta-PI/180.0/1000.0;
        if(K=='6') Pbeta=Pbeta+PI/180.0/1000.0;
        if(K=='8') Palpha=Palpha+PI/180.0/1000.0;
        if(K=='0') vectorHousen();
        if(fabs(Palpha)>=90.0*PI/180.0) break;
        if(fabs(Pbeta)>=90.0*PI/180.0) break;

/* 回転角度指定モード */
        ROTATION(ZX,-Pbeta);
        ROTATION(YZ,-Palpha);

        GX=Display.X;
        GY=Display.Y;
        GZ=Display.Z;

        /* ZX面をY軸回転 */
        PX = GX*ZX[0] + GZ*ZX[1];
        PZ = GX*ZX[2] + GZ*ZX[3];
        GX = PX;
        GZ = PZ;
        /* YZ面をX軸回転 */
        PY = GY*YZ[0] + GZ*YZ[1];
        PZ = GY*YZ[2] + GZ*YZ[3];
        GY = PY;
        GZ = PZ;
        Housen.X = GX;
        Housen.Y = GY;
        Housen.Z = GZ;

    /* 平面にのる逆格子点のテーブルを作成 */
        EWLD(&Housen);

    /* 座標変換行列の作成 */
        ROTATION(YZ,Palpha);
        ROTATION(ZX,Pbeta);

    /* 画面表示 */
        PLOT();
        printf(" Kakudo α=%f, β=%f¥n",Palpha*180.0/PI,Pbeta*180.0/PI);
    }
}
```