REFERENCES

Abrikosov, A., Fundamental Theory of Metals, pp.315-316, North-Holland, 1988.

_____., Fundamental Theory of Metals, pp.412-414, North-Holland, 1988.

_____., A.A., Sov. Phys. JEPT., 5, 1174-1182, 1957.

Allen, M.P. and D.J. Tildesley, Computer Simulation of Liquids , pp.5, Oxford U. Press, New York, 1987.

Arfken, G., Mathematical Methods for Physicists, 3rd ed., pp.944, Academic Press, Inc., 1985.

Ashcroft, N.W., and N.D. Mermin, Solid State Physic, pp. 739, Holt, Rienhart and Winston, Inc., New York, 1976.

Bardeen, J., *et al.*, Phys. Rev. 79, 167, 1950.

_____., *et al.*, Phys. Rev. 80, 567, 1950.

_____., *et al.*, Phys. Rev., 108, 1175-1204, 1957.

Baxter, R.J., *et al.*, Exactly Solved Models in Statistical Mechanics, Academic Press, London, 1982.

Bednorz, J.G. and K.A. Muller, Z. Phys. B 64, 189-193, 1986.

Brandt, E.M., Phys. Stat. Solidi. B 51, 345, 1972.

Cooper, L.N., Phys. Rev. 104, 1189, 1956.

_____., Phys. Rev. 145, 526, 1956.

Daver, B.S. and W.M. Fairbank, Phys. Rev. Lett., 7, 43, 1961.

de Gennes, P.G., Superconductivity of Metal and Alloys, 2nd ed., pp.51, Addison-Wesley Publishing Company,Inc., New York, 1989.

_____., Superconductivity of Metal and Alloys, 2nd ed., pp.55, Addison-Wesley Publishing Company,Inc., New York, 1989.

_____., Superconductivity of Metal and Alloys, 2nd ed., pp.66-69, Addison-Wesley Publishing Company,Inc., New York, 1989.

Doria, M.M., *et al.*, Phys. Rev. B 41, 6335, 1990.

Du, Q., *et al.*, Phys. Rev. B 46, 9027, 1992.

Duzer, T.V. and C.W. Turner, Principles of Superconductive Devices and Circuits, pp. 277, Elsevier North Holland, Inc., 1981.

Fetter, L.F. and J.D. Walecka, Quantum Theory of Many-particles Systems, pp. 416, McGraw Hill, Inc., New York, 1971.

_____., Quantum Theory of Many Particles Systems, pp.415, McGraw-Hill, New york, 1971.

_____., Quantum Theory of Many Particles Systems, pp.434, McGraw-Hill, New york, 1971.

_____., Quantum Theory of Many Particles Systems, pp.435, McGraw-Hill, New york, 1971.

File, F. and R.G. Mills, Phys. Rev. Lett., 10, 93, 1963.

Fröhlich, H., Phys. Rev. 79, 845, 1950.

Garner, J., and R. Benedeck, Phys. Rev. 42 , 6027, 1990.

Ginzburg, V.L. and L.D. Landau, Zh. Eksp. Teor. Fiz., 20, 1064, 1950.

Gorkov, L.P., Sov. Phys. JETP., 9, 1364, 1959.

Jackson, J.D, Classical Electrodynamics, pp. 177, John Wiley & Sons, Inc., New York, 1962.

_____., Classical Electrodynamics, pp. 138, John Wiley & Sons, Inc., New York, 1962.

_____., Classical Electrodynamics, pp.312, John Wiley & Sons, Inc., New York, 1962.

Keesom, W.H. and J.A. Kok, Comm. Phys. Lab. Univ. Leiden, 221e, 1932.

Kirkpatrick, S., *et al.*, Science, 220, 671, 1983.

Kittel, C., Introduction to Solid state Physics, 6th ed., pp. 335, John Wiley & Sons, Inc., 1986.

_____., Introduction to Solid state Physics, 6th ed., pp. 342, John Wiley & Sons, Inc., 1986.

_____., Introduction to Solid state Physics, 6th ed., pp. 321, John Wiley & Sons, Inc., 1986.

Kleiner, W.H., *et al.*, Phys. Rev. 133 , No. 5A, A1126 , 1964.

Landau, L.D. and E.M. Lifshitz, Quantum Mechanics, pp. 454, Pergamon Press, 1977.

Lifshitz, E.M. and L.P. Pitaevskii, Statistical Physics, Part 2, pp. 181, Pergamon Press, 1980.

London, F. , Superfluids, vol.1, Dover Publications, Inc., New York, 1961.

_____., and H. London, Proc. Roy. Soc. London, A 149, 71, 1935.

_____., Superfluids vol I, pp. 152, John Wiley, & Sons, New York, 1950.

_____., Superfluids, Dover Publications, Inc., New york, 1974.

Meissner, N. and Ochsenfeld, R., Naturwissenschaften, 21, 787, 1933.

Metropolis, N., J. Chem. Phys. 21, 1087, 1953.

Murphy, D.W., *et al.*, Phys. Rev. Lett., 581, 1888, 1987.

Onnes, H.K., Comm. Phys. Lab. Univ. Leiden, 119, 120, 1911.

_____., H.K., Comm. Phys. Lab. Univ. Leiden, 139f, 1914.

Parkin, S.S.P. *et al.*, Phys. Rev. Lett. 60, 2539, 1988.

Poulter, J. , The Lecture Notes on Ginzburg-Landau Theory, Forum for Theoretical Science, Chulalongkorn University, 1991.

Press, W.H. and S. A. Teukolsky, Computer in Physics, 5, No4, 426, 1991.

_____., *et al.*, Numerical Recipes: The Art of Scientific Computing, pp. 289-334, Cambridge U. Press., New York, 1986.

Rose-Innes, A.C., and E.H. Rhoderich, Introduction to Superconductivity, 2nd ed., pp.183-185, Pergamon Press, New York, 1978.

_____., Introduction to Superconductivity, 2nd ed., pp.19, Pergamon Press, New York, 1978.

_____., Introduction to Superconductivity, 2nd ed., pp.35, Pergamon Press, New York, 1978.

_____., Introduction to Superconductivity, 2nd ed., pp.74, Pergamon Press , New York, 1978.

_____., Introduction to Superconductivity, 2nd ed., pp.78, Pergamon Press, New York, 1978.

_____., Introduction to Superconductivity, 2nd ed., pp.83, Pergamon Press, New York, 1978.

_____., Introduction to Superconductivity, 2nd ed., pp.189, Pergamon Press, New York, 1978.

Ruffolo, D. and P. Boolchand, Phys. Rev. Lett. 55, 242, 1985.

Sheng, Z.Z., and A.M. Herman, Nature 332, 55, 1988.

Shoenberg, D., Superconductivity, 2nd ed., Cambridge University Press, Cambridge, 1965.

Silsbee, F.B., J. Wash. Acad. Sci. 6, 597, 1916.

Silverman, A. and J. Addler, Computer in Physics, 6, No3, 277, 1992.

Subramanian, M.A., *et al.*, Nature 239, 1015, 1988.

Tarascon, J.M., *et al.*, Phys. Rev. B 38, 8885, 1988.

Taylor, P.L., A Quantum Approach to the Solid State, pp.242., Prentice-Hall, Inc., 1970.

Tinkham, M., Introduction to Superconductivity, pp. 3, Science Publisher, New York, 1975.

_____., Introduction to Superconductivity, pp. 114-115, Science Publisher, New York, 1975.

_____., Introduction to Superconductivity, pp. 144-157, Science Publisher,

New York, 1975.

_____., Introduction to Superconductivity, pp. 144-157, Science Publisher, New York, 1975.

Torardi, C.C., *et al.*, Nature 240, 631, 1988.

Träuble, H. and U. Essmann, J. App. Phys. 39, No. 9, 4052-4059, 1968.

Vecchi, M.P. and S. Kirkpatrick, IEEE Trans. Comput. Aided Design. CAD-2, 215, 1983.

Wang, Z.D. and C.R. Hu, Phys. Rev. B. 44, 11918, 1991.

Wu, M.K., *et al.*, Phys. Rev. Lett. 58, 908, 1987.

Zimmerman, J.E. and J.E. Mercereau, Phys. Rev. Lett., 14, 887, 1965.

APPENDIX A

```
C  ***************************************************************
C  ## PROGRAM NAME:    GL1DI.FOR
C  ## Author:          Tanin Nutaro
C  ## Advisors:        Dr. Julian Poulter, Dr. David Ruffulo,
C                      Prof. Virulh Sa-yakanit.
C  ## Last update:     July/19/1992
C  ## Purpose:         minimizing the Gibbs free energy
C
C     g=-|psi|^2+0.5|psi|^4+grad(psi)^2+q^2psi^2+k^2(curl(q)-H)^2.
C
C  ## Output file 1> GL1DI.out :: psi, q, Hint data file.
C                 2> Est.out   :: initial input data and the best
C                                 free energy estimatimation.
C  Method of integration :: Trapezoid rule.
C  ***************************************************************
C  A1,A2 are the location points of the sample in coherence length unit
C  PSI    is the orderparameter
C  Q      is the supervelocity :(Q <- A+grad(phi))
C  HEXT   is the external magnetic field
C  Hint   is the internal magnetic field
C  N      is the number of strip for integration
C  KAPPA  is the Ginzburg-Landau parameter(k)
C  TOL    is the tolerance
C
       REAL *8 F,A1,A2,X,SUM,H,est,tol,z,psi,psi0,Q,Q0,HINT,HEXT,KAPPA
       DIMENSION PSI(5009),psi0(5009),A(5009)
       DIMENSION Q(5009),Q0(5009),HINT(5009)
C
       INTEGER N,np
C  make a decision to start program from the real initial condition
C  or to read the previous data files(GL1DI.out & EST.out).
C
       logical start
       DATA N/100/
       PRINT *,'Input A1,A2'
       READ *, A1,A2
       print*,'Input hext, kappa'
       read*,hext,kappa
       print*,'Type 'T' if you want to start program'
       print*,'Type 'F' if you want to read data files'
       read*,start
       np=n+1
C
C  generate the coefficient of trapezoid rule for integration
C
       do 1 i=2,np
1       a(i)=2.d0
        a(1)=1.d0
        a(np)=1.d0
       h=(a2-a1)/dfloat(n)
```

```
C
        tol=1.d-1
        IF(start)then
C
C  generate initial values for psi=1.0 ans q=0.0 everywhere.
C
        est=1.d50
        DO 5 i=1,np
            psi(i)=1.d0
            psi0(i)=psi(i)
            q(i)=0.d0
            q0(i)=q(i)
5        continue
        ELSE
C
C the second condition to read the previous data files
C
            open(7,file= 'est.out')
                read(7,*)est
            close(7)
             open(8,file= 'gl1di.out')
                DO 12 j=1,np
                  read(8,123)i,psi(j),q(j),hint(j)
                   psi0(j)=psi(j)
12                 Q0(j)=Q(j)
             close(8)
C
        ENDIF
C
C  start counting
C
        kount=0
        kx=0
 500    call func(psi,Q,np,tol)
C
        hint(1)=(q(2)-q(1))/h
        do 796 i=2,(np-1)
 796    hint(i)=.5d0*(q(i+1)-q(i-1))/h
        hint(np)=(q(np)-q(np-1))/h
C
C  SUM is the result of integration
        SUM=0.D0
        DO 2 I=1,NP
 2      SUM=SUM+F(PSI,Q,NP,I,H,HEXT,KAPPA,hint(i))*a(i)
C
        SUM=H*SUM/2.d0
C
C  if SUM is less than the previous one set kx to kx+1
        if(sum.lt.est)then
           kx=kx+1
           print *, 'better estimate  ',sum,TOL
           kount=0
           est=sum
        DO 550 i=1,np
           psi0(i)=psi(i)
550        q0(i)=q(i)
        if(kx.eq.100)then
           kx=0
```

```
            open(7,file='est.out')
              write(7,*)est,A1,A2,n,kappa,hext,tol
            close(7)
            open(8,file='gl1di.out')
          DO 600 i=1,np
 600        write(8,123)i,psi(i),Q(I),HINT(I)
            close(8)
          endif
         else
C  set Kount to Kount+1
         kount=kount+1
          DO 700 i=1,np
             psi(i)=psi0(i)
700       q(i)=q0(i)
         endif
C
C  if more than 250 tries but can't get the better result,
C  change tolurence

         if(kount.gt.250)then
            TOL=0.5D0*TOL
            write(6,*)tol
            kount=0
          endif
C
C  until the TOL < 1.0e-6 :save the final result and the stop program
C
         IF (TOL .LT.1.D-6) then
            open(7,file='est.out')
              write(7,*)est,a1,a2,n,kappa,hext,tol
            close(7)
            open(8,file='gl1di.out')
          DO 450 i=1,np
 450        write(8,123)i,psi(i),Q(i),HINT(I)
123        Format (I4,2x,3(f15.7,2x))
            close(8)
          STOP
         ENDIF
          goto 500
         STOP
         END

C   @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
       SUBROUTINE FUNC(PSI,Q,NP,TOL)
C   @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
C   The main purpose of this subroutine is to adjust the value
C   of psi, q  randomly
C
         REAL*8 A,B,tol,psi,Q,y,z
         REAL*4 URAND
         INTEGER SEED
         DIMENSION PSI(NP),Q(NP)
                save seed
         DATA SEED/0/
         rand=urand(seed)
         DO 3 I=1,NP
             if(rand.le..1)then
               y=psi(i+1)-2.0*psi(i)+psi(i-1)
```

```
                a=.25d0*y
              else
             A=2.D0*(URAND(SEED)-.5)*TOL
              endif
             PSI(I)=PSI(I)+A
             psi(1)=0.d0
             psi(np)=0.d0
3         IF(PSI(I).LT.0.D0)PSI(I)=0.D0
         DO 2 I=1,np
            if(rand.le..1.and.i.ne.1.and.i.ne.np) then
              z=q(i+1)-2.0*q(i)+q(i-1)
              a=.25d0*z
            else
            A=2.D0*(URAND(SEED)-.5)*TOL
            endif
2           Q(I)=Q(I)+A
         RETURN
         END
C
C  ------------------------------------------------------------
         FUNCTION F(PSI,Q,NP,INDEX,H,HEXT,KAPPA,hint)
C  ------------------------------------------------------------
C  The main function for free energy calculation

         REAL*8 F,A,B,C,D,E,h,psi,Q,HEXT,KAPPA,curl,hint
         DIMENSION PSI(NP),Q(NP)
          A=PSI(INDEX)
          B=PSI(INDEX+1)
            D=Q(INDEX)
            E=Q(INDEX+1)
         if(index.eq.np)b=0.d0
         if(index.eq.np)e=d
                    F=A*A
           C=(B-A)/H
           CURL=hint-HEXT
          F=-F+.5D0*F*F+C*C+D*D*F+KAPPA*KAPPA*CURL*CURL
         RETURN
         END
C  ------------------------------------------------------------
C  ************************************************************
      FUNCTION URAND(IY)
C  ************************************************************
C           UNIFORM RANDOM NUMBER GENERATOR.
C           INITIALISE IY PRIOR TO FIRST CALL.
C           CALLING PROGRAMME MUST NOT ALTER IY BETWEEN CALLS.
C           VALUES OF URAND ARE RETURNED IN THE INTERVAL (0,1).
C

      REAL*8 HALFM
      SAVE IA,IC,M2,S
      DATA M2/0/,ITWO/2/
      IF(M2.NE.0)GOTO 20
C
C           IF FIRST ENTRY, COMPUTE MACHINE INTEGER WORD LENGTH.
C
      M=1
   10 M2=M
      M=ITWO*M2
```

```
      IF(M.GT.M2)GOTO 10
      HALFM=M2
C
C         COMPUTE MULTIPLIER AND INCREMENT FOR LINEAR CONGRUENTIAL
C         METHOD.
C
      IA=8*IDINT(HALFM*DATAN(1.D0)/8.D0)+5
      IC=2*IDINT(HALFM*(.5D0-DSQRT(3.D0)/6.D0))+1
      MIC=(M2-IC)+M2
C
C         S IS THE SCALE FACTOR FOR CONVERTING TO FLOATING POINT.
C
      S=.5/HALFM
C
C         COMPUTE NEXT RANDOM NUMBER
C
   20 IY=IY*IA
C
C         THE FOLLOWING STATEMENT IS FOR COMPUTERS WHICH DO NOT
C         ALLOW INTEGER OVERFLOW ON ADDITION.
C
C     IF(IY.GT.MIC)IY=(IY-M2)-M2
C
      IY=IY+IC
C
C         THE FOLLOWING STATEMENT IS FOR COMPUTERS WHERE THE
C         WORD LENGTH FOR ADDITION IS GREATER THAN FOR MULTIPLICATION.
C
C     IF(IY/2.GT.M2)IY=(IY-M2)-M2
C
C         THE FOLLOWING STATEMENT IS FOR COMPUTERS WHERE INTEGER
C         OVERFLOW AFFECTS THE SIGN BIT.
C
      IF(IY.LT.0)IY=(IY+M2)+M2
C
      URAND=FLOAT(IY)*S
      RETURN
      END
C
```

APPENDIX B

```
#include  <math.h>
#include  <stdio.h>
#define  NROWS   19
#define  NDIM    1121   /* Must be 3*NROWS^2 + 2*NROWS. */
#define  FIRST_T  1
#define  FRACTION  0.5
#define  LAST_T   0.00001
/*  Last modified :: 30 SEP 1992
   initial condition : psi[]              = 0.0
                       qx[] , qy[]        = 0.0
                         length           = 20.0
                         kappa            = 3.0
                         hext             = 0.5
        number of change points 300
*/

double  hext, kappa, length;
double  bests[NDIM+1], s[NDIM+1], ds[NDIM+1];
double  ss;
unsigned  long ia, ic, iy = 2847, m2 = 0;
long    better;
int     iff=0;

main()
{
  double  dy, newfe, oldfe, qx ;
  int     ans, i, j,select;
  long    total;
  double  update(), f();
  void    printout(),previous();
  length=20.0;
  kappa=3.0;
  hext=0.50;
```

```
  dy = length/((double)NROWS+1);
  printf("\n This is the best program for simulated annealing \n");
  printf("\n");
  printf("enter 0 to start program from initial condition\n");
  printf("      1 to read the previous data files \n");
   scanf("%d", &select);
 /*       select=1; */
          if(select == 1){ previous();}
    else {
          for (j=1;j<=NROWS*NROWS;j++) {
    s[j] = bests[j] = 0.5 ;
    range[j]=0.01;
  }
  for(i=(NROWS*NROWS)+1,qx= 0.0;i<=2*(NROWS*NROWS)+NROWS;i++) {
    bests[i] = s[i] = qx;
    if (!((i-NROWS*NROWS) % (NROWS+1))) qx = 0.0;
    range[i]=0.01;
  }
  for(i=2*(NROWS*NROWS)+NROWS+1;i<=3*(NROWS*NROWS)+2*
NROWS;i++){
    bests[i] = s[i] = 0.0 ;
    range[i]=0.01;
  }

  t = FIRST_T;
  }

/*   Start here  */
   for(i=1; i<=NDIM+1; i++) { range[i] = 0.01 ;}
  better = 0;
  total  = 0;
  oldfe  = f(s);
  printf("oldfe ====%lf\n",f(s));
  for (t = FIRST_T;t >= LAST_T;) {
          for(better = 0;better <= 10*NDIM;) {
```

```
    if (ans=update(&newfe,&oldfe,&t)) {
      printf("b ");
      oldfe = newfe;
      for (i=1;i<=NDIM;i++) bests[i] = s[i];
      better++;
      total++;
      if (!(better % 100)) {
        printf(" better = %d, total = %d, newfe = %lf\n",
          better,total,newfe);
        printout();
        sleep(2);
      }
    } else {
      printf("w ");
      total++;
    }
  }
        t *=FRACTION;
 }
}

double  update(newfe,oldfe,t)
double  *newfe, *oldfe, *t;
{
  double        de, random;
  int           i, k;
  static long int  seed=1278;

 double  ran();
  double  f();
for(i=1;i<=300;i++){
  random=NDIM*ran(&seed);
  k=random+1;
  if(random+1-k<0.5){
    s[k]=bests[k]+range[k];
  } else {
```

```
    s[k]=bests[k]-range[k];
                }
 if (k <= NROWS*NROWS) s[k] = fabs(s[k]);
    }

  *newfe = f(s);
 /*  printf("NeW fe of up date==%lf\n",f(s)); */
  de = *newfe - *oldfe;
  return(de < 0.0 || ran(&seed) < exp(-de/ *t));
}

/* RAN --> UNIFORM RANDOM DEVIATE BETWEEN 0 AND 1.

    NEGATIVE ARGUMENT REINITIALIZES SEQUENCE.

    THIS ROUTINE IS FAST, PORTABLE, AND FREE OF
    SEQUENTIAL CORRELATIONS.  OUTPUT IS LIMITED
    INTEGERS DIVIDED BY 714,025.

    from NUMERICAL RECIPES IN C (ran2)
*/

#define     M       714025
#define     IA      1366
#define     IC      150889

double  ran(idum)
long    *idum;
{
  int    j;
  void   nrerror();

  if (*idum < 0 || iff == 0) {

                /* INITIALIZE SEQUENCE  */
```

```
    iff=1;
    if ((*idum=(IC-(*idum)) % M) < 0) *idum = -(*idum);
    for (j=1;j<=97;j++) {
        *idum=(IA*(*idum)+IC) % M;
        ir[j]=(*idum);
    }
    *idum=(IA*(*idum)+IC) % M;
    iy=(*idum);
  }

   /* START HERE UNLESS INITIALIZING.      */
   /* RANDOMLY SELECT NUMBER FROM SEQUENCE. */

  j=1 + 97.0*iy/M;
  if (j > 97 || j < 1) nrerror("RAN: This cannot happen.");
  iy=ir[j];
  *idum=(IA*(*idum)+IC) % M;
  ir[j]=(*idum);
  return (double) iy/M;
}

double f(s)
double *s;
{
  double   dq1dy, dq2dx, dsdx, dsdy, dx, dy, fe=0.0, *q1, *q2, feinc;
  double  s2, qx2, qy2,sud,slr,qll,qlr,qul,qur,qle,que,qre;
  double  curl1, curl2;
  int     jx, jy, qxoffset, qyoffset, soffset;

   /* Set the pointer q to point to s+n.  Thus the second half of
        the vector s is referred to as q.
   */

  q1 = s + NROWS*NROWS;
  q2 = s + 2*NROWS*NROWS + NROWS;
```

```
dy = dx = length / (double)(NROWS+1);

  /* Contribution from lattice points: inside the boundary */

for (jy=1;jy<=NROWS;jy++) {
 soffset = (jy-1)*NROWS;
  qxoffset = (jy-1)*(NROWS+1);
  qyoffset = (jy-1)*NROWS;
  for (jx=1;jx<=NROWS;jx++) {
       s2 = s[soffset+jx]*s[soffset+jx];
       qx2 = (q1[qxoffset+jx]+q1[qxoffset+jx+1])
          * (q1[qxoffset+jx]+q1[qxoffset+jx+1]) / 4;
       qy2 = (q2[qyoffset+jx]+q2[qyoffset+jx+NROWS])
          * (q2[qyoffset+jx]+q2[qyoffset+jx+NROWS]) / 4;
       fe += -s2 + 0.5*s2*s2 + s2*qx2 + s2*qy2;
  }
}

  /* Contribution from lattice points: on the boundary
       Boundary conditions: s = 0 at boundary
  */

  /* No contribution from boundary points! psi(edge) = 0. */

/*  For Grad(psi)^2 : ((Sup-Sdown)/dy)^2 + ((Sright-Sleft)/dx)^2  */
 for (jx=1; jx<=NROWS-1;jx++)
 {
       for(jy=1;jy<=NROWS-1;jy++)
       {
       sud=((s[(jy)*NROWS+jx]-s[(jy-1)*NROWS+jx])/dy); /*sud=s(up-down)/dy*/
       slr=((s[(jy-1)*NROWS+jx+1]-s[(jy-1)*NROWS+jx])/dx);/*slr=sleft-right*/
       fe += (sud*sud) + (slr*slr);
       }
 }

/* at the boundary: only edges */
```

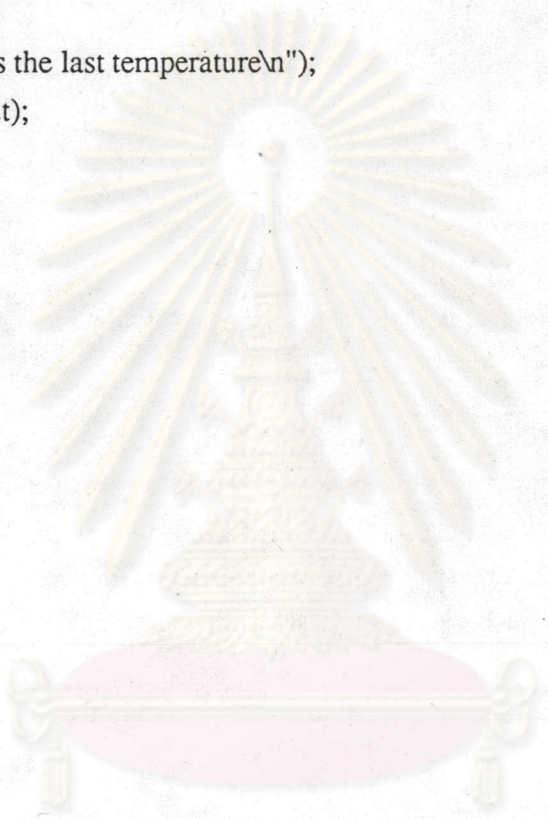
```
                /* Lower edge */
for (jx=1; jx<=NROWS;jx++)
{
        sud=((s[jx]-0) / dy); /*sud=s(up-down)/dy*/
        fe += (sud*sud);
}

                /* Upper edge */

for (jx=1; jx<=NROWS;jx++)
{
        sud=((0-s[(NROWS-1)*NROWS+jx])/dy); /*sud=s(up-down)/dy*/
        fe += (sud*sud);
}

          /* Left edge */

        for(jy=1;jy<=NROWS;jy++)
        {
        slr=((s[(jy-1)*NROWS+1] - 0) / dx);/*slr=sleft-right*/
        fe += (slr*slr);
        }

          /* Left edge */

        for(jy=1;jy<=NROWS;jy++)
        {
        slr=((0-s[(jy-1)*NROWS+NROWS]) / dx);/*slr=sleft-right*/
        fe += (slr*slr);
        }


/* For the term k^2*(curlQ-hext)*/

 for (jy=1;jy<=NROWS-1;jy++) {
   soffset = (jy-1)*NROWS;
```

```
    qxoffset = (jy-1)*(NROWS+1);
    qyoffset = jy*NROWS;
    for (jx=1;jx<=NROWS-1;jx++) {
        curl1=(q2[qyoffset+jx+1]-q2[qyoffset+jx]) ;
        curl2= (q1[jy*(NROWS+1)+jx+1]-q1[qxoffset+jx+1]);
        feinc =kappa*kappa*(((curl1-curl2)/dx)-hext)*(((curl1-curl2)/dx)-hext);
      fe += feinc;
    }
  }
 /* For the boundary  */

 /* Corners, upper edge, lower edge -> all 0 (Qx free to make them zero) */

  fe *= dx * dy;
/*   printf("f: fe = %12lf\n",fe);     */
  return(fe);
}


void printout()
{
  FILE     *fp_s,*fp_q1,*fp_q2,*fp_h ,*fp_e;
  int i,j,k,l;
  double   dx, hint, *s, *q1, *q2;

        void nrerror();
  dx=length/(double)(NROWS+1);

   /* printf("   %lf\n",f(p)); */

                        fp_s  = fopen("psi3.dat","w");
                        fp_q1 = fopen("qone3.dat","w");
                        fp_q2 = fopen("qtwo3.dat","w");
                        fp_h  = fopen("hint3.dat","w");
           fp_e  = fopen("free3.dat","a");
      q1 = p + NROWS*NROWS;
```

```
        q2 = p + 2*NROWS*NROWS + NROWS;

        for (k=1;k<=NROWS;k++) {
             for (l=1;l<=NROWS;l++) {
        fprintf(fp_s," %4d %4d %12lf\n",k,l,bests[(l-1)*NROWS+k]);
          }
        }
        for (k=1;k<=NROWS+1;k++) {
             for (l=1;l<=NROWS;l++) {
             fprintf(fp_q1," %4d %4d %12lf\n",k,l,q1[(l-1)*(NROWS+1)+k]);
          }
        }
        for (k=1;k<=NROWS;k++) {
          for (l=1;l<=NROWS+1;l++) {
             fprintf(fp_q2," %4d %4d %12lf\n",k,l,q2[(l-1)*NROWS+k]);
          }
        }
        for (k=1;k<=NROWS-1;k++) {
          for (l=1;l<=NROWS-1;l++) {
             hint = (q2[l*NROWS+k+1]-q2[l*NROWS+k]
               -q1[l*(NROWS+1)+k+1]+q1[(l-1)*(NROWS+1)+k+1])/dx;
             fprintf(fp_h," %4d %4d %12lf\n",k,l,hint);
          }
          }
                  fprintf(fp_e,"%6d\t %lf\t %f \n",(better/100),f(p),t);
        fclose(fp_s);
        fclose(fp_q1);
        fclose(fp_q2);
           fclose(fp_h);
          fclose(fp_e);
}

void previous()
 {   FILE *fsc_s, *fsc_q1, *fsc_q2 ;
    int i, j, k, l;
    double *q1,  *q2 ;
```

```
   void nrerror();


printf("Hello..\n");
printf("hereeeeeeee");
fflush(stdout);
        fsc_s = fopen("psi3.dat","r");
        fsc_q1 = fopen("qone3.dat","r");
        fsc_q2 = fopen("qtwo3.dat","r");

        for(k=1;k<=NROWS;k++){
         for(l=1; l<=NROWS; l++){
         for (i=1; i<= NROWS*NROWS; i++) {
                fscanf(fsc_s,"%4d%4d%12lf",&k,&l,&s[i]);
  /*   printf("s[%d] = %lf\n ", i, s[i]);   */
             bests[i] = s[i];
                 }
       }
    }
   for(k=1; k<=NROWS+1;k++){
        for(l=1; l<=NROWS; l++){
      for(i=(NROWS*NROWS)+1; i<=2*(NROWS*NROWS)+NROWS ; i++)
      {
           fscanf(fsc_q1,"%4d%4d%12lf",&k,&l,&s[i]);
        /*    printf("s[%d] = %lf\n", i,s[i]); */
           bests[i] = s[i];
      }
        }
    }
   for(k=1;k<=NROWS ; k++){
        for(l=1; l<=NROWS+1; l++){
      for(i=2*(NROWS*NROWS)+NROWS+1 ; i<= 3*(NROWS*NROWS)+(2*
NROWS); i++)
          {
           fscanf(fsc_q2,"%4d%4d%12lf",&k,&l,&s[i]);
           bests[i] = s[i];
```

```
/*          printf("s[%d] = %lf\n", i, s[i]); */
        }
       }
     }
    fclose(fsc_s);
    fclose(fsc_q1);
    fclose(fsc_q2);
     printf("What is the last temperature\n");
     scanf("%lf",&t);
}
```



CURRICULUM VITAE

Mr. Tanin Nutaro was born on September 10, 1961 in Surin. He received his B.Ed. degree in Physics from Srinakharinwirot Bangkhen University in 1984. During his study for a M.Sc. degree in physics at Chulalongkorn University, he received a scholarship from the University Development Commission of the National Council (U.D.C.) and he worked as a research assistant for the high-temperature superconductivity project, which was supported by the Science and Technology Development Board (STDB).