Help for StrucFact

D.J. Goossens

September 2, 2013

1 Introduction

`StrucFact` or `StrucFact.exe` is a crude program to calculate expected intensities for neutron diffraction (magnetic and nuclear) and X-ray diffraction (in the version from 2013). It is distributed as source code because any serious user will likely need to modify the source to make sensible use of it.

It's mathematics is essentially taken from *Neutron Diffraction* by George Bacon and H. M. Rietveld *J. Appl. Cryst.* (1969) **2** 65-71, and its mandate is limited; its job is to calculate F^2 for neutron scattering for arbitrary cells, magnetic or nuclear, and more recently for X-ray diffraction. It is 'developed' solely on an 'as needed' basis, which means I add 'features' when I need them to solve some problem I am working on. The inverted commas may see gratuitous, but they are not!

I am sure there are better tools out there for everything that this program does, and I advise against using it.

There should be a README.TXT and the code itself distributed with this file.

Please Note

1. It does not work for incommensurates (unless you want to define an enormous cell).
2. It treats every cell as $P1$ (i.e. you have to give it *all* atoms explicitly).
3. The nuclear and magnetic cells must be the same size, which means that is one is bigger than the other (usually magnetic bigger than nuclear) you have to put the atoms from the smaller cell into the bigger, including inserting all redundant copies of atoms.
4. *No* corrections for intensities (e.g. not even Lorentz), no B-factors beyond the isotropic.

In other words, it is remarkably limited. Why anyone would want to use it I do not know when FullProf and GSAS and the like are around. Having said that, it is quite simple (in the sense that everything has to be done explicitly, so it is laborious but not as conceptually demanding) compared to such programs, and the (minimally tested) code is here available.

2 Compiling

This is Fortran90 code that does not need any extra libraries. To use `g95` running under `cygwin` to make a statically-linked executable, type:

```
g95 -mno-cygwin -fstatic -o StrucFact.exe strucfact.f90
```

where the source is in `strucfact.f90`.

Non-static is even easier (`g95 -o StrucFact.exe strucfact.f90`).

Please report errors in the code to:

`goossens@rsc.anu.edu.au`.

3 Running the Program

Here is an example input file. The fields are:

1. Unit cell parameters a, b, c, α, β and γ
2. Number of atoms (in total, not in asymmetric unit, and in the biggest cell you are using.)
3. Then N rows, each of which is: `TYPE`, x, y, z, *occ*, B_{iso}, `MAGFLAG`. Where `TYPE` is 3 characters, if need be padded out to 3 characters with 'x' (for example phosphorus would be 'Pxx'). Three characters to allow for the magnetic form factor, which for example might be Mn^{2+} (labelled 'Mn2') or Mn^{3+} etc. x etc are fractional coords, and *occ* is as a fraction of unity — not referenced to general positions or anything (seeing as we have no asymmetric unit, you can say everything is a general position and we are working in $P1$ and with a multiplicity of 1 all the time). The `MAGFLAG` indicates if the magnetic moment of that atom is to be specified below.
4. If `MAGFLAG` $= 1$ for some atoms, say `nmagatom` atoms, here we specify the magnetic moments. The fields are *moment*, θ and ϕ, where *moment* is the moment magnitude in μ_B (Bohr magnetons), θ is the angle the moment makes to the x axis (parallel to the a axis) and ϕ is the angle the moment makes to the z axis (the z axis is perpendicular to the ab plane, so in the direction of c^*).

   In other words, these lines specify the magnetic structure.
5. Then we give the number of reflections whose F and F^2 we are going to calculate.
6. Then we give a list of *hkl*s.

That is all.

Example input file; this is the magnetic structure of $MnPS_3$, wherein there are four magnetic atoms in the unit cell (the Mn), each with all neighbours of opposite spin, and the moments are of around 4.5 μ_B and are (anti)parallel to the c^* axis (see for example K. C. Rule, et al., *Applied Physics A: Materials Science & Processing* **74** (2002) s811-s813 and references therein):

```
 6.0770 10.5240 6.7960 90.0000 107.3500 90.0000
20
Mn2 0. 0.3326 0. 1. 1.22 1
Mn2 0. 0.6674 0. 1. 1.22 1
Mn2 0.5 0.8326 0. 1. 1.22 1
Mn2 0.5 0.1674 0. 1. 1.22 1
Pxx 0.0556 0. 0.1686 1. 0.77 0
Pxx 0.9444 0. 0.8314 1. 0.77 0
Pxx 0.5556 0.5 0.1686 1. 0.77 0
Pxx 0.4444 0.5 0.8314 1. 0.77 0
Sxx 0.7593 0. 0.2497 1. 0.96 0
Sxx 0.2407 0. 0.7503 1. 0.96 0
Sxx 0.2593 0.5 0.2497 1. 0.96 0
Sxx 0.7407 0.5 0.7503 1. 0.96 0
Sxx 0.2438 0.1612 0.2516 1. 0.96 0
Sxx 0.7562 0.1612 0.7484 1. 0.96 0
Sxx 0.7562 0.8388 0.7484 1. 0.96 0
Sxx 0.2438 0.8388 0.2516 1. 0.96 0
Sxx 0.7438 0.6612 0.2516 1. 0.96 0
Sxx 0.2562 0.6612 0.7484 1. 0.96 0
Sxx 0.2562 0.3388 0.7484 1. 0.96 0
Sxx 0.7438 0.3388 0.2516 1. 0.96 0
4.5   90.0  0.0
4.5   90.0  180.0
4.5   90.0  0.0
4.5   90.0  180.0
750
0 1 0
0 0 1
1 0 0
0 1 -1
0 1 1
0 2 0
1 0 -1
1 -1 0
1 1 0
1 -1 -1
```

```
1 1 -1
0 2 -1
...
etc
...
```

4 Code

Here is the code as of 1 Sept 2013. Key points are:

- The arrays of scattering factors (scattering lengths for nuclear scattering and form factor parameters for magnetic and X-ray) are read in from three text files which *must* therefore be distributed with the executable. On the other hand, they can easily be edited if you don't like the numbers or whatever.
- The nuclear calc uses one approach, the magnetic another (in terms of calculation of scattering vectors and the like) and there could be more unification in the code and it could be more efficient, but it is not(!).
- It is perhaps not very efficient code, but I believe it is pedagogically useful as it calculates most quantities fairly directly from their most common definitions.

```
program strucfact

!
! 01 Sept 2013
!
! This code is usually distributed in an archive
! with a name something like StrucFact_Files.tar.gz
!
! If so, this archive usually contains:
!
! strucfact.f90 -- Fortran90 source code (this file).
! StrucFact.pdf -- Documentation of a sort.
! StrucFact.exe -- a Windows executable that might work.
! StrucFact     -- a Mac OS X (10.6) executable that might work.
! three .txt files that contain the scattering factors.  Easily edited
! if you want to change them.
! MnPS3_mag.inp -- An example input file.
! README.txt    -- Directs users to read this header.
!
! If for some strange reason you make published use of this code, please
! just reference:
!
```

```
! D. J. Goossens, StrucFact, 2011, http://rsc.anu.edu.au/~goossens/StrucFact_Files.zip
!
! or something similar.  The code is in the public domain and is made
! available for incorporation into other programs, modifcation, extension etc.
! An acknowledgement would be nice, and a copy of the derived program too.
!
! Darren
!
! goossens@rsc.anu.edu.au

  implicit none

  character*3 :: atype(100),matype(100),ion(220),nion(200),mion(200)
  character*90 :: header

  integer     :: natom, nmagatom
  integer     ::  atypenum(100)
  integer     :: nrefn,atom
  integer     :: i, mflag, j, k,itype

  real        :: xyz(100,3), mxyz(100,3), occ(100)
  real        :: Biso(100), mocc(100), mBiso(100)
  real        :: bcoh(200), msf(200,7), Xsf(220,9)
  real        :: param(6), kappa,eg2mc,s,formf,tempval
  real        :: hkl(5000,3),h(3),Fnucl,Fmag,x(3),Fin,Frn,Xrn,Xin
  real        :: hxkylz, dw, edotK, ivec(3)
  real        :: a,b,c,al,be,ga,sa2,sb2,sg2,ca2,cb2,cg2,ca,cb,cg,sa,sb,sg
  real        :: d, d1,d2,d3,d4,d5,d6,d7,d8
  real        :: pi, rpart,ipart,angle,cvol,q(3),uq(3),qmag
  real        :: mom(100),theta(100),phi(100),umom(100,3)
  real        :: ccel(3,3),crec(3,3),pref,axb(3),bxc(3),cxa(3)
  real        :: Ay,Ax,Az,Bx,By,Bz,Psquared,edotPsquared
  real        :: edotPreal, edotPimag


  open(unit=1,file="x-ray_coefficients_from_int_tables_edited.txt",status="old")
  read(1,'(a90)')header
  write(6,*)'Reading X-ray scattering coefficients'
  write(6,*)header
   do i = 1,187
    read(1,*)ion(i),(Xsf(i,j),j=1,9)
    write(6,*)ion(i),(Xsf(i,j),j=1,9)
  end do
  close(1)

  open(unit=1,file="neutron_bcoh_from_international_tables_edited.txt",status="old")
```

```
   read(1,'(a90)')header
  write(6,*)'Reading neutron nuclear scattering coefficients'
  write(6,*)header
   do i = 1,187
    read(1,*)nion(i),tempval
    bcoh(i)=tempval * 0.1
    if(nion(i).ne.(ion(i))) then
        write(6,*)nion(i),ion(i),'!!!!!'
        write(6,*)'Labels in x-ray scatt factor file must agree with'
        write(6,*)'those in neutron scattering length file'
        stop
    end if
    write(6,*)nion(i),bcoh(i)
  end do
  close(1)

! International tables Table 4.4.5.1 (j0)
  open(unit=1,file="neutron_mag_from_international_tables_edited.txt",status="old")
   read(1,'(a90)')header
  write(6,*)'Reading neutron magnetic scattering coefficients'
  write(6,*)header
   do i = 1,95
    read(1,*)mion(i),(msf(i,j),j=1,7)
    write(6,*)mion(i),(msf(i,j),j=1,7)
  end do
  close(1)

 !  write(6,*)"relative scales uncertain as of now.... Sept 2013"



  pi = 3.14159265358979
  eg2mc = 0.54 / 2.0

  ! e^2gamma/(mc^2)(J. Appl. Cryst. (1969). 2, 65-71., Rietveld)
  ! Factor of 2 is a question?  IN units of 10-12 cm (to scale with bcoh)
  ! Rietveld uses factor of 2 on bottom line, Bacon does not...?
  ! Strufac calcs follow this paper by Rietveld

  write(6,*) '---------------------------'
  write(6,*) 'Mag struc fac calcs toolbox'
  write(6,*) 'D. J. Goossens, August 2011'
  write(6,*) '---------------------------'

  write(6,*) 'Lattice params, a b c al be ga (angles in deg)'
  read(5,*) param(1),param(2),param(3),param(4),param(5),param(6)
```

```
write(6,*) 'num atoms (in largest cell):'
read(5,*) natom

do i = 4,6
   param(i) = param(i) * pi / 180.0
end do
write(6,*)param(1),param(2),param(3),param(4),param(5),param(6)

write(6,*) 'type x y z occ Biso Mag'
write(6,*)'pad type to 3 chars with x eg C = Cxx'
write(6,*)'write Fe2 for Fe2+ (for mag calc)'
matype = ''
mxyz = 0.
mBiso = 0.
mocc = 0.
j = 0
do i = 1,natom
   mflag = -999
   read(5,*) atype(i),xyz(i,1),xyz(i,2),xyz(i,3),occ(i),Biso(i),mflag
   if (mflag.eq.1) then
      j = j + 1
      matype(j) = atype(i)
      mxyz(j,1) = xyz(i,1)
      mxyz(j,2) = xyz(i,2)
      mxyz(j,3) = xyz(i,3)
      mocc(j) = occ(i)
      mBiso(j) = Biso(i)
      mflag = -999
   end if
end do
nmagatom = j
write(6,*)'Found ',nmagatom,' mag atoms.'

if(nmagatom.gt.0) then
   write(6,*)'So specify their moments...'
   write(6,*)'moment (mu_B), angle to x, angle to z'
   write(6,*)'where x is along a and z is along c*'
end if

mom = 0.
theta = 0.
phi = 0.

do j = 1,nmagatom
   read(5,*)mom(j),theta(j),phi(j)
end do
```

```
  ! convert to radians
  do j = 1,nmagatom
     theta(j) = theta(j) * pi / 180.0
     phi(j) = phi(j) * pi / 180.0
  end do

  !bcoh(1) = 2.847 ! guess!!!; 1 = S
  !bcoh(2) = 5.13  ! P
  !bcoh(3) = 9.45  ! Fe
  !bcoh(4) = 10.3   ! Ni
  !bcoh(5) = -3.73 ! Mn
  !bcoh(1) = .2847 ! guess!!!; 1 = S
 ! bcoh(2) = .513  ! P
 ! bcoh(3) = .945  ! Fe
 ! bcoh(4) = 1.03   ! Ni
 ! bcoh(5) = -0.373 ! Mn
!
!  msf = -99999.9
!  ! http://www.ill.eu/sites/ccsl/ffacts/ffactnode3.html
!  ! Mag scatt facs, Fe2+
!  msf(3,1) = 0.0263
!  msf(3,2) = 34.9597
!  msf(3,3) = 0.3668
!  msf(3,4) = 15.9435
!  msf(3,5) = 0.6188
!  msf(3,6) = 5.5935
!  msf(3,7) = -0.0119
!
!  ! Mag scatt facs, Ni2+
!  msf(4,1) = 0.0163
!  msf(4,2) = 35.8826
!  msf(4,3) = 0.3916
!  msf(4,4) = 13.2233
!  msf(4,5) = 0.6052
!  msf(4,6) = 4.3388
!  msf(4,7) = -0.0133
!
!  ! Mag scatt facs, Mn2+
!  msf(5,1) = 0.4220
!  msf(5,2) = 17.6840
!  msf(5,3) = 0.5948
!  msf(5,4) = 6.0050
!  msf(5,5) = 0.0043
!  msf(5,6) = -0.6090
!  msf(5,7) = -0.0219
  write(6,*) 'How many refns?'
```

```
read(5,*)nrefn
write(6,*)'hkl (reals)'
do i = 1,nrefn
   read(5,*)hkl(i,1),hkl(i,2),hkl(i,3)
end do

!     bcoh(1) = 2.3   ! guess!!!; 1 = S
!     bcoh(2) = 2.7   ! P
!     bcoh(3) = 9.23  ! Fe
!     bcoh(4) = 11.23  ! Ni
!     bcoh(5) = 0.23  ! Mn

a = param(1)
b = param(2)
c = param(3)
al = param(4)
be = param(5)
ga = param(6)

sa = sin(al)
sb = sin(be)
sg = sin(ga)
sa2 = (sin(al))**2
sb2 = (sin(be))**2
sg2 = (sin(ga))**2
ca2 = (cos(al))**2
cb2 = (cos(be))**2
cg2 = (cos(ga))**2
ca = (cos(al))
cb = (cos(be))
cg = (cos(ga))

!q is the magnetic interaction vector, equal to E( e. K) - K, where E is the
!unit scattering vector, and K is a unit vector in the direction of the magnetic
!moment of the ion.
!
! ok, so this needs some serious geometry.  We need a*, b* and c* and
! a, b c in terms of x, y and z.

! a  = a x^
! b = b sin gamma y^ + b cos gamma x^
! c =
ccel = 0.0
ccel(1,1) = a
ccel(2,1) = b * cg
ccel(2,2) = b * sg
ccel(3,1) = c * cb
ccel(3,2) = c * ca
ccel(3,3) = c * sqrt(1.0 - cb2 - ca2)

write(6,*)'Cell in Cartesian'
write(6,'(3f8.4)')(ccel(1,j),j=1,3)
write(6,'(3f8.4)')(ccel(2,j),j=1,3)
write(6,'(3f8.4)')(ccel(3,j),j=1,3)

call crossprod(ccel(1,:),ccel(2,:),axb)
call crossprod(ccel(2,:),ccel(3,:),bxc)
call crossprod(ccel(3,:),ccel(1,:),cxa)
write(6,*)'b x c, c x a, a x b'
write(6,'(3f8.4)')bxc
write(6,'(3f8.4)')cxa
write(6,'(3f8.4)')axb
cvol = ccel(1,1)*bxc(1)+ccel(1,2)*bxc(2)+ccel(1,3)*bxc(3)
write(6,*)'Cell vol, a.bxc',cvol
cvol = ccel(2,1)*cxa(1)+ccel(2,2)*cxa(2)+ccel(2,3)*cxa(3)
write(6,*)'Cell vol, b.cxa',cvol
cvol = ccel(3,1)*axb(1)+ccel(3,2)*axb(2)+ccel(3,3)*axb(3)
write(6,*)'Cell vol, c.axb',cvol
crec(1,1) = 2.0 * pi * bxc(1) / cvol
crec(1,2) = 2.0 * pi * bxc(2) / cvol
crec(1,3) = 2.0 * pi * bxc(3) / cvol
crec(2,1) = 2.0 * pi * cxa(1) / cvol
crec(2,2) = 2.0 * pi * cxa(2) / cvol
crec(2,3) = 2.0 * pi * cxa(3) / cvol
crec(3,1) = 2.0 * pi * axb(1) / cvol
crec(3,2) = 2.0 * pi * axb(2) / cvol
crec(3,3) = 2.0 * pi * axb(3) / cvol
write(6,*)'Recip cell, Cartesian'
write(6,'(3f8.4)')(crec(1,j),j=1,3)
write(6,'(3f8.4)')(crec(2,j),j=1,3)
write(6,'(3f8.4)')(crec(3,j),j=1,3)

! unit vectors in moment directions
write(6,*)'Unit vectors for mag moments'
do j = 1,nmagatom
   umom(j,1) = cos(theta(j))*sin(phi(j))
   umom(j,2) = sin(theta(j))*sin(phi(j))
   umom(j,3) = cos(phi(j))
   write(6,'(i3,3f8.4)')j,(umom(j,i),i=1,3)
end do

write(6,*) 'Output:'
```

```
  write(6,*) '--------------------------'
  write(6,'(3a5,5a7,4a8,a12)')'h','k','l','d','qx','qy','qz','|q|'     &
       ,'|Fn|^2','|Fm|^2','|FX|^2'
  !       ,'Frn','Fin','|Fn|^2','|Fm|^2'
  do j = 1,nrefn
     Fin = 0.
     Frn = 0.
     Xin = 0.
     Xrn = 0.
     h(1) = hkl(j,1)
     h(2) = hkl(j,2)
     h(3) = hkl(j,3)

     !  scattering vector is q(3)
     q(1) = h(1)*crec(1,1) + h(2)*crec(2,1) + h(3)*crec(3,1)
     q(2) = h(1)*crec(1,2) + h(2)*crec(2,2) + h(3)*crec(3,2)
     q(3) = h(1)*crec(1,3) + h(2)*crec(2,3) + h(3)*crec(3,3)
     ! Unitised scattering vector is uq(3)
     qmag = sqrt((q(1))**2+(q(2))**2+(q(3)**2))
     uq(1) = q(1)/qmag
     uq(2) = q(2)/qmag
     uq(3) = q(3)/qmag
     d1 = h(1)**2*sa2/a**2
     d2 = h(2)**2*sb2/b**2
     d3 = h(3)**2*sg2/c**2
     d4 = 2.*h(1)*h(2)*(ca*cb-cg)/(a*b)
     d5 = 2.*h(2)*h(3)*(cb*cg-ca)/(b*c)
     d6 = 2.*h(3)*h(1)*(cg*ca-cb)/(c*a)
     d7 = 1.-ca2-cb2-cg2
     d8 = 2.*ca*cb*cg

     d = 1.0 / sqrt((d1+d2+d3+d4+d5+d6)/(d7+d8))
     kappa = 2.0*pi/d
     s = kappa/(4.0*pi)
! Neutron nuclear, and X-ray
     do i = 1,natom
!        if(atype(i).eq.'Sxx')atom=1
!        if(atype(i).eq.'Pxx')atom=2
!        if(atype(i).eq.'Fe2')atom=3
!        if(atype(i).eq.'Ni2')atom=4
!        if(atype(i).eq.'Mn2')atom=5
         atom = -9999
         do itype = 1,187
           if(atype(i).eq.nion(itype))atom=itype
         end do
         if (atom.eq.-9999) then
```

```
              write(6,*)atype(i),' is unknown symbol'
              stop
         end if
        x(1) = xyz(i,1)
        x(2) = xyz(i,2)
        x(3) = xyz(i,3)
        hxkylz = 0.
        do k = 1,3
           hxkylz = hxkylz + h(k)*x(k)
        end do
        dw = exp(-1.0*Biso(i)*kappa**2/(4.0*pi)**2)
        ! a(cos theta + i sin theta)
        angle = 2.0*pi*hxkylz
        rpart = cos(angle)
        ipart = sin(angle)
! Neutron
        Frn = Frn + rpart*bcoh(atom)*dw*occ(i)
        Fin = Fin + ipart*bcoh(atom)*dw*occ(i)
! X-ray
        formf = Xsf(atom,1)*exp(-1.0*Xsf(atom,2)*s**2) +  &
             Xsf(atom,3)*exp(-1.0*Xsf(atom,4)*s**2) +  &
             Xsf(atom,5)*exp(-1.0*Xsf(atom,6)*s**2) +  &
             Xsf(atom,7)*exp(-1.0*Xsf(atom,8)*s**2) +  &
             Xsf(atom,9)
         Xrn = Xrn + rpart*formf*dw*occ(i)
         Xin = Xin + ipart*formf*dw*occ(i)

     end do


     ! Magnetic part...

     Ax = 0.0
     Ay = 0.0
     Az = 0.0
     Bx = 0.0
     By = 0.0
     Bz = 0.0
     do i = 1,nmagatom
        edotK = uq(1)*umom(i,1)+uq(2)*umom(i,2)+uq(3)*umom(i,3)
        ivec(1) = uq(1) * edotK - umom(i,1)
        ivec(2) = uq(2) * edotK - umom(i,2)
        ivec(3) = uq(3) * edotK - umom(i,3)
!        if(matype(i).eq.'Sxx')atom=1
!        if(matype(i).eq.'Pxx')atom=2
!        if(matype(i).eq.'Fe2')atom=3
!        if(matype(i).eq.'Ni2')atom=4
!        if(matype(i).eq.'Mn2')atom=5
         atom = -9999
         do itype = 1,95
           if(matype(i).eq.mion(itype))atom=itype
         end do
         if (atom.eq.-9999) then
             write(6,*)matype(i),' is unknown symbol'
             stop
         end if
        x(1) = mxyz(i,1)
        x(2) = mxyz(i,2)
        x(3) = mxyz(i,3)
        hxkylz = 0.
        do k = 1,3
           hxkylz = hxkylz + h(k)*x(k)
        end do
        dw = exp(-1.0*mBiso(i)*kappa**2/(4.0*pi)**2)
        ! a(cos theta + i sin theta)
        angle = 2.0*pi*hxkylz
        rpart = cos(angle)
        ipart = sin(angle)
        ! spin-only form factor for now:
        formf = msf(atom,1)*exp(-1.0*msf(atom,2)*s**2) +  &
             msf(atom,3)*exp(-1.0*msf(atom,4)*s**2) +  &
             msf(atom,5)*exp(-1.0*msf(atom,6)*s**2) +  &
             msf(atom,7)
        Ax = Ax + rpart * formf * mom(i) * mocc(i) * umom(i,1) * dw
        Ay = Ay + rpart * formf * mom(i) * mocc(i) * umom(i,2) * dw
        Az = Az + rpart * formf * mom(i) * mocc(i) * umom(i,3) * dw
        Bx = Bx + ipart * formf * mom(i) * mocc(i) * umom(i,1) * dw
        By = By + ipart * formf * mom(i) * mocc(i) * umom(i,2) * dw
        Bz = Bz + ipart * formf * mom(i) * mocc(i) * umom(i,3) * dw
        !write(6,*)'ipart,rpart,mom(i),mocc(i),umom(i,:),dw'
        !write(6,*)ipart,rpart,mom(i),mocc(i),umom(i,:),dw
     end do

     ! So we have the magnetic sum for hkl, now we need to
     ! turn it into a single number.
     ! Rietveld says we need |P|^2 - |e.P|^2 where
     ! P = (Ax+Ay+Az) + i(Bx+By+Bz)
     ! and e = unit vector along kappa = uq(3)
     ! So...
     edotPreal = uq(1)*Ax + uq(2)*Ay +uq(3)*Az
     edotPimag = uq(1)*Bx + uq(2)*By +uq(3)*Bz
     ! squared magnitude of edotPreal + i edotPimag = ...
     edotPsquared  =  edotPreal**2 + edotPimag**2
     ! Now, |P| is not quite so clear, but,
     ! I assume |A|^2 = Ax^2 + Ay^2 +Az^2 etc
     ! so that
     Psquared = Ax**2 + Ay**2 + Az**2 + Bx**2 + By**2 + Bz**2
     Fmag = sqrt(Psquared - edotPsquared)* eg2mc
     Fnucl = sqrt(Frn**2+Fin**2)
     write(6,'(3f5.1,5f7.3,2f8.3,f12.3)')h,d,q,kappa,Fnucl**2,Fmag**2,   &
            (Xrn**2+Xin**2)
     !   write(6,*)'edotPreal,edotPimag,edotPsquared,Psquared,eg2mc'
     !   write(6,*)edotPreal,edotPimag,edotPsquared,Psquared,eg2mc
     !   write(6,*)Fnucl,Fmag,Fnucl**2,Fmag**2
  end do

  stop

end program strucfact

SUBROUTINE crossprod(A, B, C)      ! cross product (right-handed)

  IMPLICIT NONE

  real, DIMENSION(3), INTENT (IN)    :: A
  real, DIMENSION(3), INTENT (IN)    :: B
  real, DIMENSION(3), INTENT (OUT)   :: C

  C(1) = A(2)*B(3) - A(3)*B(2)
  C(2) = A(3)*B(1) - A(1)*B(3)
  C(3) = A(1)*B(2) - A(2)*B(1)

  RETURN

END SUBROUTINE crossprod
```

5 Notes

Ultimately, this could form the core of a program for magnetic diffuse calculations, by hybridising with `DIFFUSE` by Butler and Welberry. But probably it never will.

Direct any comments to the universe at large, but if you really feel the need, I can be contacted at goossens@rsc.anu.edu.au.