Computer Algebra calculation of XRMS polarisation dependence in the non spherical case

Thomas A. Wood*, Alessandro MIRONE
ESRF, BP 220, 38043 Grenoble, France

September 12, 2018

Abstract

Simple analytical formulae, directly relating the experimental geometry and sample orientation to the measured R(M)XS scattered intensity are very useful to design experiments and analyse data. Such formulae can be obtained by the contraction of an expression containing the polarisations and crystal field tensors, and where the magnetisation vector acts as a rotation derivative [3]. The result of a contraction contains a scalar product of (rotated) polarisation vectors and the crystal field axis. As an example, the dipole-dipole RXS scattering amplitude by Jahn-Teller distortion is calculated as:

$$\begin{aligned}\mathbb{C}(\epsilon'(2z^2-x^2-y^2)\epsilon) &= 2\epsilon'\overbrace{z\overbrace{z\epsilon}} + 2\epsilon'\overbrace{z\overbrace{z\epsilon}} + ... \\ &= 4\epsilon'_z\epsilon_z - 2\epsilon'_x\epsilon_x - 2\epsilon'_y\epsilon_y\end{aligned}$$

The contraction rules give rise to combinatorial algorithms which can be efficiently treated by computers. In this work we provide and discuss a concise Mathematica code along with a few example applications to non-centrosymmetric magnetic systems.

1 Introduction

A tensorial contraction method has been developed [3] to obtain analytical formulae for X-ray resonant magnetic scattering, working from the established formulae for RMXS amplitude in the spherical atom approximation [2]. Here this method will be implemented in a Mathematica code and used to investigate electric dipole and quadrupole scattering in the general case of a non-spherical system.

*Van Mildert College, University of Durham, DH1 3LH, England

To illustrate the capabilities of the code, we consider the case of *spiral antiferromagnetic holmium*. This has an HCP structure and the atoms are embedded in a local D_{3h} symmetry environment. The magnetic field direction of the holmium varies helically according to the crystal layer.

The Mathematica code executes contractions of the tensors of an incoming and outgoing photon with the field tensor of the crystal system, to obtain expressions for the dipole-dipole, quadrupole-quadrupole and mixed dipole-quadrupole corrections to the scattering amplitude for the spherical case.

2 Theory

2.1 Spherical case

We consider the interaction between matter and a photon described by a wave vector $\vec{k}$ and a polarisation vector $\vec{\epsilon}$, discarding both elastic Thompson scattering and the spin-magnetic field interaction. Our atom is spherical [3] but perturbed by a magnetic exchange field. We take the angular momentum quantisation axis $\vec{\xi}$ along the magnetic field. Since dipolar and quadrupolar terms will not mix due to parity conservation, the scattering amplitude is given by [3]:

$$F_{\epsilon,k\to\epsilon',k'} = \sum_{q=-1}^{q=1} F_{1,q}\epsilon_q^{1*}\epsilon_q^1 + \sum_{q=-2}^{q=2} F_{2,q}(k\epsilon)_q^*(k\epsilon)_q \tag{1}$$

where $(k\epsilon)_q$ denotes the rank 2 spherical tensor components of $\vec{k}\otimes\vec{\epsilon}$, and ϵ_q^1 represents $\vec{\epsilon}$ in rank 1 spherical tensor form. The scattering coefficients $F_{1,q}$ and $F_{2,q}$ can be derived from the analysis in [3].

2.2 Spherical case in Cartesian representation

It is, however, more convenient to keep the vectors $\vec{k}$ and $\vec{\epsilon}$ in Cartesian form and re-express Equation 1 in Cartesian space. Using a different set of coefficients $F_1'^n$, $F_2'^n$ (which we will define in Section 4.4), and substituting q in Equation 1 by $\vec{\xi}.\vec{L}$ where $\vec{L}$ is the angular momentum operator, we re-express Equation 1 as

$$F_{\epsilon,k\to\epsilon',k'} = \mathbb{C}\left(\epsilon'\sum_{n=0}^{n=2}(i\xi\times)^n F_1'^n\epsilon\right)+\mathbb{C}\left((k'\otimes\epsilon')\sum_{n=0}^{n=4}(i\xi\times)^n F_2'^n(k\otimes\epsilon)\right)/2 \tag{2}$$

where $\mathbb{C}$ signifies a sum over all possible contractions (the contractions are defined in Section 2.4). This formulation can be used to calculate directly the dipolar and quadrupolar scattering coefficients for a spherical atom [3], which are in agreement with the formulae obtained previously by Hill & McMorrow [2]. The implementation of $\mathbb{C}$ as a computer-efficient algorithm is one of the objectives of this code.

2.3 Non-spherical case

We use the extension of this formulation for the case of a non-spherical atom [3]. We represent the non-sphericity of the atomic environment by a crystal potential $T(x, y, z)$ which is added to the atomic Hamiltonian, which is given as a superposition of spherical harmonics [3]:

$$T(x, y, z) = \sum_{l,q} t_{l,q} T_q^l(x, y, z) \tag{3}$$

where T must also have the same symmetry as the crystal system [1].

The field tensor must now be included in Equation 2. The scattering amplitude for the non-spherical case is therefore given by the contraction sum

$$F_{\epsilon,k\to\epsilon',k'} = \mathbb{C}\left(\epsilon' \sum_{n=0}^{n=2} (i\xi\times)^n F_1'^n T\epsilon\right) + \mathbb{C}\left((k'\otimes\epsilon') \sum_{n=0}^{n=4} (i\xi\times)^n F_2'^n T(k\otimes\epsilon)\right)/2 \tag{4}$$

Later we will construct some code to evaluate $F_{\epsilon,k\to\epsilon',k'}$ in the general case of a field tensor in x, y and z, and then investigate the effect that this will have on the scattering peaks measured.

2.4 Contractions

First we define mathematically what is meant by a *contraction* [4] and the function $\mathbb{C}$, which represents the sum of all possible contractions between the tensors it is operating on, that result in a scalar. In Section 3 this will be implemented as a computer code.

Single contractions A single contraction of two tensors involves taking the inner product between an index of the first tensor $\vec{a}_1 \otimes \vec{a}_2$ and one of the second tensor $\vec{b}_1 \otimes \vec{b}_2$, thus reducing the total rank of the expression by two. So, for two tensors $\vec{a}_1 \otimes \vec{a}_2$ and $\vec{b}_1 \otimes \vec{b}_2$, each of rank 2, the possible single contractions are

$$(\vec{a}_1.\vec{b}_1)\vec{a}_2\vec{b}_2 \quad (\vec{a}_1.\vec{b}_2)\vec{a}_2\vec{b}_1 \quad (\vec{a}_2.\vec{b}_1)\vec{a}_1\vec{b}_2 \quad (\vec{a}_2.\vec{b}_2)\vec{a}_1\vec{b}_1 \tag{5}$$

i.e. there are four possible single contractions of $\vec{a}_1 \otimes \vec{a}_2$ with $\vec{b}_1 \otimes \vec{b}_2$, each one a tensor of rank 2.

Double contractions However, we are interested in contracting the two tensors to form a scalar. For the case of the two rank-2 tensors $\vec{a}_1 \otimes \vec{a}_2$ and $\vec{b}_1 \otimes \vec{b}_2$, we can execute the first single contraction and then contract the two indices that have not already been contracted. Working from Equation 5, we see that this results in a total of four possible (double) contractions:

$$(\vec{a}_1.\vec{b}_1)(\vec{a}_2.\vec{b}_2) \quad (\vec{a}_1.\vec{b}_2)(\vec{a}_2.\vec{b}_1) \quad (\vec{a}_2.\vec{b}_1)(\vec{a}_1.\vec{b}_2) \quad (\vec{a}_2.\vec{b}_2)(\vec{a}_1.\vec{b}_1) \tag{6}$$

Therefore, the contraction sum of $\vec{a}_1 \otimes \vec{a}_2$ and $\vec{b}_1 \otimes \vec{b}_2$, which is the sum of all possible contractions, is

$$\mathbb{C}(A_{ij}B_{kl}) = 2(\vec{a}_1.\vec{b}_1)(\vec{a}_2.\vec{b}_2) + 2(\vec{a}_2.\vec{b}_1)(\vec{a}_1.\vec{b}_2) \tag{7}$$

Note the factors of 2 which arise from the repeated terms in Equation 6.

Multiple contractions We would like to construct an iterative computer algorithm to take a set of tensors and repeatedly contract them until the result becomes a scalar.

We begin by taking all possible single contractions (for the case of two tensors of rank N, there are $2N$ possible single contractions; however we would also like to contract three tensors with each other).

Then we take this result and take all possible single contractions on it again—this continues until either the expression becomes a scalar (in which case the contraction function is no longer called, and the result is output as the answer), or a tensor of rank 1 (which will be ignored by the program, as it cannot be contracted to a scalar).

3 The program

Here we describe the workings of the Mathematica notebook. The program input is presented in a different colour for easy differentiation from the rest of the document. Occasionally the program output has been stylised slightly for this report, but everything listed here was produced more or less directly by Mathematica.

In this section we will define the contraction mechanisms for calculating the magnetic diffraction amplitude for the general case of a material with a known field tensor.

In Section 4 we will then apply this mechanism for the specific case of spiral antiferromagnetic holmium, and show how this can be used to calculate the expected intensities of the satellite diffraction peaks of Bragg order $n \pm q$.

3.1 Tensor contraction

First we set up a function to execute tensorial contractions (these are needed to calculate the scattering amplitudes for a non-spherical atom).

Contracting two tensors First we define a function `Con` which will find the sum of all possible contractions for two tensors rank N. Each tensor is given in the form $\texttt{T}[\vec{a}_1, \vec{a}_2 ... \vec{a}_N]$ (representing the tensor product $\vec{a}_1 \otimes \vec{a}_2 \otimes ... \otimes \vec{a}_N$ of a set of vectors $\vec{a}_1$, $\vec{b}_2$, ... $\vec{a}_N$). For example, the contraction sum of two tensors $\vec{a}_1 \otimes \vec{a}_2$ and $\vec{b}_1 \otimes \vec{b}_2$ will be entered as $\texttt{Con}[\texttt{T}[\vec{a}_1, \vec{a}_2], \texttt{T}[\vec{b}_1, \vec{b}_2]]$, and is equivalent to, mathematically,

$$\mathbb{C}((\vec{a}_1 \otimes \vec{a}_2)(\vec{b}_1 \otimes \vec{b}_2)) = (\vec{a_1}.\vec{b}_1)(\vec{a}_2.\vec{b}_2) + (\vec{a}_1.\vec{b}_2)(\vec{a}_2.\vec{b}_1) \tag{8}$$

i.e. each vector in the left hand tensor is contracted with each one on the right, thus generating all possible contractions.

The contraction function `Con` for two tensors is given below. It loops round, gradually reducing the expression by eliminating pairs of vectors and calling itself again to contract the remaining expression. This creates a complicated stack of one function calling itself again many times with different arguments—to prevent infinite loops in the case of an error, in this definition `Con` will not call itself directly but will call a temporary (as yet undefined) function `ConTmp` which can later be set equal to `Con`.

First we state that the function `Con` and our tensor product operator `T` are both *orderless*, so it does not matter in which order their arguments (vectors and tensors) are given.

```
Attributes[Con] = {Orderless};
Attributes[T] = {Orderless};
```

Now we create the contraction algorithm. The function takes two tensors and returns the sum of all possible single contractions between them.

```
Con[x__, 0]:=0;
Con[T[a_, al__], T[bl__]]:=Module[{res}, res = 0;
 n = Length[List[bl]];
 Do[
 res =
  res + scal [a/. {x → e1, y → e2, z → e3} ,
   List[bl][[i]]/. {x → e1, y → e2, z → e3}]
   × ConTmp[T[al], Delete[T[bl], i]]
 , {i, 1, n}
 ];
 res
];
```

where we introduce the variables $\vec{e}_1$, $\vec{e}_2$ and $\vec{e}_3$ to represent the basis vectors in the $\vec{x}$, $\vec{y}$ and $\vec{z}$ directions (this will be useful later for when we will need to take scalar products).

Some other definitions are necessary to enable the program to stop looping when it has finished the contractions. The contraction of a single tensor must evaluate to zero; to contract anything with 1, we can ignore the 1; and the contraction of nothing must evaluate to 1.

```
Con[T[c___]]:=0;
Con[1, t__]:=Con[t];
Con[]:=1;
T[] = 1;
T[{}] = 1;
T[{x_}] = T[x];
```

Contracting three tensors Now a contraction function has been defined for two tensors, we can start to think about $\mathbb{C}$ for three tensors. This is more complicated, as the sum of possible contractions is much greater. The function works in the same way as the `Con` function for the case of two tensors, eliminating vector pairs one by one and calling itself (via `ConTmp`) repeatedly. Once one of the three tensors has been fully contracted with vector elements from the other two, the expression will have been reduced to a sum of contractions of *two* tensors—which is a much simpler problem and has already been defined above.

```
Con[T[al__], T[bl__], T[cl__]]:=
Module[{res, m, n}, res = 0;
 m = Length[T[bl]];
 n = Length[T[cl]];
 Do[
  res =
   res + scal[List[al][[1]]/. {x → e1, y → e2, z → e3},
    List[bl][[i]]/. {x → e1, y → e2, z → e3}]
    × ConTmp[Delete[T[al], 1], Delete[T[bl], i], T[cl]]
  , {i, 1, m}
  ];    Do[
  res =
   res + scal[List[al][[1]]/. {x → e1, y → e2, z → e3},
    List[cl][[i]]/. {x → e1, y → e2, z → e3}]
    × ConTmp[Delete[T[al], 1], T[bl], Delete[T[cl], i]]
  , {i, 1, n}
  ];
  res
 ];
```

As a temporary measure, we also make all callings of `ConTmp` equivalent to calling the function `Con` itself (thus creating a stack)—in the case of errors or infinite loops, this line can be removed.

```
ConTmp = Con
```

3.2 Extending the contraction functions to work with polynomials

The field tensor in polynomial representation Now we need to make a function which will contract a Cartesian tensor in the form of a polynomial in x, y and z. This is needed for scattering amplitude calculations, where the crystal field tensor is typically expressed in a polynomial form $T = f(x, y, z)$. For example, the quadrupole-quadrupole scattering formula for the particular case of holmium has the form:

$$F^{q,q}_{\epsilon,k\to\epsilon',k'} = \mathbb{C}((\epsilon' \otimes k')(t_2(2z^2 - x^2 - y^2) \pm t_3(x^3 - 3xy^2))(\epsilon \otimes k)) \quad (9)$$

We define a new function $\mathbb{C}(A, f(x, y, z), C)$ which re-expresses a contraction of tensors A and C and the polynomial f in terms of sums and products of contractions of three tensors which can be executed by the function `Con`.

Reducing the polynomial via linearity We want to be able to contract some quite complicated polynomials, so it is useful to first reduce the polynomials to a more manageable form using some basic mathematical properties of the function $\mathbb{C}$. For example, the contraction sum operation is linear, so

$$\mathbb{C}(A(f(x, y, z) + g(x, y, z))B) = \mathbb{C}(Af(x, y, z)B) + \mathbb{C}(Ag(x, y, z)B) \quad (10)$$

So we can start constructing the function $\mathbb{C}$ by creating a `Rule`[1] that any polynomial to be contracted must first be split into terms and the final contraction will be a sum of contractions of these terms:

```
ℂ[{al__}, Plus[x_, y__], {cl__}]:=
 Module[{i}, xy = Expand[x + y];
  Sum[ℂ[{al}, xy[[i]], {cl}], {i, 1, Length[xy]}]]
```

Removing prefactors from the individual terms Now we need to find an efficient way of taking each term and extracting the xyz dependent components for contraction, while leaving numbers and constants untouched by the `Con` function. Here we use another property of the linearity of $\mathbb{C}$:

$$\mathbb{C}(A(kf(x, y, z))B) = k\mathbb{C}(Af(x, y, z)B) \quad (11)$$

So first we separate out any occurrences of xyz from the terms of the polynomial (which has already been split up into a sum of parts):

```
ℂ[{al__},
 Times[prefactor_/;FreeQ[FullForm[prefactor], x]/;
 FreeQ[FullForm[prefactor], y]/;
 FreeQ[FullForm[prefactor], z], xyz__], {cl__}]:=
 Times[prefactor]ℂ[{al}, Times[xyz], {cl}];
```

Converting a polynomial term to a tensor Now that individual terms in xyz have been separated, they can be converted into a tensorial form (such as `T[x, x, y, z]` in Mathematica) that can be used directly by the function `Con`.

[1] In Mathematica, a `Rule` defining $\mathbb{C}$ means that every time $\mathbb{C}$ is referred to in subsequent code, the code within the `Rule` is invoked. Parameters on the left side of a definition of a `Rule` must be followed by underscores (e.g. `A_`), to show that they can take any value.

Expressing $x^\alpha y^\beta z^\gamma$ in tensorial form First we make a `Rule` that $\mathbb{C}(A(x^\alpha y^\beta z^\gamma)B)$ will be executed as `Con[T[A],T[B],T[C]]` where `B` is a sequence of the variables `x`, `y` and `z`. For example, we want the expression

$$\mathbb{C}[\mathtt{T[A]},(\mathtt{x}^2\mathtt{yz}),\mathtt{T[C]}] \equiv \mathbb{C}(A(x^2yz)B)\ (\textit{mathematical notation}) \tag{12}$$

to be converted to

$$\mathtt{Con[T[A],T[x,x,y,z],T[C]}] \tag{13}$$

and then contracted in the usual way by the function `Con`. The code to execute this is as follows:

```
ℂ[{al__},Times[xyz1_,xyz2__],{cl__}]:=
 Module[{i,power,n,xyz},
  xyz:=List[xyz1,xyz2]/.x_^power-- >Table[x,{i,1,power}];
  n = Length[xyz];
  Con[T[al],T[Flatten[Table[xyz[[i]],{i,1,n}]]],T[cl]]
 ]
```

Expressing x^α in tensorial form We also need a `Rule` to interpret powers of a single variable, such as $\mathbb{C}(A(x^\alpha)B)$, since this case is not covered by the above `Rule` (although the interpretation is much the same).

```
ℂ[{al__},Power[xyz_,power_],{cl__}]:=
 Module[{i,n},
  Con[T[al],T[Table[xyz,{i,1,power}]],T[cl]]
 ]
```

3.3 Defining the scalar product symbolically

The contraction function defined so far has produced results in terms of the function `scal`$[\vec{a},\vec{b}]$, as yet undefined. We must now define `scal` so that a product of a Cartesian vector with one of the basis vectors $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$ and $\hat{\mathbf{z}}$ will be re-expressed as the correct component of that vector (at least for the moment).

```
Clear[scal,ξ];
Attributes[scal] = {Orderless};
scal[vector_,e_i_] :=vector.{x̂,ŷ,ẑ}[[i]];
```

4 Application: spiral antiferromagnetic holmium

The contraction mechanism which has been defined will work in general on any kind of polynomial tensor. Now we present an example of its application, for the specific case of spiral antiferromagnetic holmium

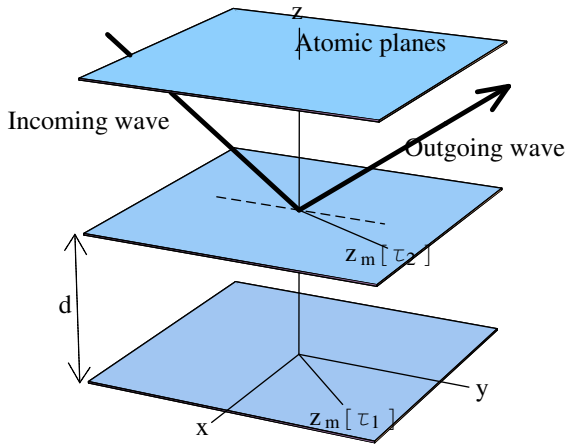

Figure 1: The geometry of the holmium system. Since τ is different for different crystal planes, $\vec{\hat{z}}_m$ has a spiral dependence on z.

4.1 Defining the geometry of the system

Defining frame X First, we define a simple Cartesian co-ordinate system. Initially we would like to work in the frame X with co-ordinates (x, y, z) oriented around the crystal geometry (z points upwards through the crystal planes), as shown in Figure 4.1.

The magnetic field direction $\vec{\hat{z}}_m$ of spiral antiferromagnetic holmium varies helically according to the crystal layering, as

$$\vec{\hat{z}}_m(\tau) = (\cos\tau, \sin\tau, 0) \tag{14}$$

where the angle $\tau = \frac{2\pi q z}{d}$ depends on the height z (d is the interplanar spacing and q is the antiferromagnetic wavevector). In Mathematica code this is expressed as (see Figure 4.1):

```
ẑₘ[τ_] = {Cos[τ], Sin[τ], 0};
```

The incoming/outcoming photons Now we define the characteristics in frame X of an incoming photon, at angle θ from the crystal planes. Its wavevector is

```
kx[θ_] = {Cos[θ], 0, -Sin[θ]};
```

and its polarisation vectors in X are (with the indices 0 and 1 representing polarisation orientations σ and π respectively):

$\epsilon\mathtt{x}_0[\theta\_] = \{0, 1, 0\};$
$\epsilon\mathtt{x}_1[\theta\_] = \{\mathtt{Sin}[\theta], 0, \mathtt{Cos}[\theta]\};$
$\mathtt{vector}\_\sigma := \mathtt{vector}_0;$
$\mathtt{vector}\_\pi := \mathtt{vector}_1;$

4.2 The magnetic co-ordinate system

Defining M It is more convenient for our purposes to introduce a new co-ordinate system M with basis vectors $(\vec{\hat{x}}_m, \vec{\hat{y}}_m, \vec{\hat{z}}_m)$ here, constructed from the (rotating) magnetic moment $\vec{\hat{z}}_m$. We set $\vec{\hat{x}}_m(\tau)$ to the most convenient possibility:

$\hat{\mathtt{x}}_\mathtt{m}[\tau\_] = \{0, 0, 1\};$

And we can now derive $\vec{\hat{y}}_m(\tau)$ from $\vec{\hat{x}}_m(\tau)$ and $\vec{\hat{z}}_m(\tau)$:

$\hat{\mathtt{y}}_\mathtt{m}[\tau\_] = \hat{\mathtt{x}}_\mathtt{m}[\tau] \times \hat{\mathtt{z}}_\mathtt{m}[\tau];$

So our new basis is

$$\left(\vec{\hat{x}}_m | \vec{\hat{y}}_m | \vec{\hat{z}}_m\right) = \begin{pmatrix} 0 & -\sin\tau & \cos\tau \\ 0 & \cos\tau & \sin\tau \\ 1 & 0 & 0 \end{pmatrix} \tag{15}$$

Re-expressing wave parameters in the magnetic co-ordinate system Now we can redefine $\vec{k}$ and $\vec{\epsilon}_{\pi,\sigma}$ in the frame M from their definitions `kx` and $\epsilon\mathtt{x}_{\pi,\sigma}$ in frame X (the subscript `pol` stands for either π or σ):

$\mathtt{k}[\theta\_, \tau\_] := \{\mathtt{kx}[\theta].\hat{\mathtt{x}}_\mathtt{m}[\tau], \mathtt{kx}[\theta].\hat{\mathtt{y}}_\mathtt{m}[\tau], \mathtt{kx}[\theta].\hat{\mathtt{z}}_\mathtt{m}[\tau]\};$
$\epsilon_{\mathtt{pol}}\_[\theta\_, \tau\_] := \{\epsilon\mathtt{x}_{\mathtt{pol}}[\theta].\hat{\mathtt{x}}_\mathtt{m}[\tau], \epsilon\mathtt{x}_{\mathtt{pol}}[\theta].\hat{\mathtt{y}}_\mathtt{m}[\tau], \epsilon\mathtt{x}_{\mathtt{pol}}[\theta].\hat{\mathtt{z}}_\mathtt{m}[\tau]\};$

So the wave vector in this new basis is

$$\vec{k}(\theta, \tau) = \begin{pmatrix} -\sin(\theta) \\ -\cos(\theta)\sin(\tau) \\ \cos(\theta)\cos(\tau) \end{pmatrix} \tag{16}$$

And the polarisation vectors are

$$\vec{\epsilon}_\sigma(\theta, \tau) = \begin{pmatrix} 0 \\ \cos(\tau) \\ \sin(\tau) \end{pmatrix}; \vec{\epsilon}_\pi(\theta, \tau) = \begin{pmatrix} \cos(\theta) \\ -\sin(\theta)\sin(\tau) \\ \cos(\tau)\sin(\theta) \end{pmatrix} \tag{17}$$

The reflected wave We define the reflected wave parameters $\vec{k}'$, $\vec{\epsilon}'$ by changing the sign of θ:

$\epsilon'_{\mathtt{pol}\_}[\theta\_, \tau\_] := \epsilon_{\mathtt{pol}}[-\theta, \tau]$;
$\mathtt{k}'[\theta\_, \tau\_] := \mathtt{k}[-\theta, \tau]$;

4.3 Applying the contraction mechanism for the Holmium case

Using our expression for the Holmium field tensor (ignoring the $\pm$ alternation),

$$T = t2(2z^2 - x^2 - y^2) + t_3(x^3 - 3xy^2) \tag{18}$$

we can use the functions $\mathbb{C}$ and `Con` to derive some properties of the diffraction pattern, that arise from the non-sphericity of the holmium atom. We are interested in particular in effects that are unique for the non-spherical case—*corrections* to the spherical scattering formulae are what we are after. We will analyse separately the dipole-dipole, quadrupole-quadrupole and mixed terms, as the whole expression is rather complicated.

Calculating the dipole-dipole scattering correction First we look at the simplest case: the dipole-dipole scattering correction. Here we only consider the contraction of ϵ' and ϵ (the outgoing and incoming photon polarisation tensors respectively) with T, ignoring k' and k: that is, we ignore effects of the finiteness of the speed of light. We will attempt to find this function's proportionality (i.e. we are not interested in prefactors for now).

The contraction is (taking only the first order term in the operator $(i\xi\times)^n$):

$$\mathbb{C}(\epsilon' T(i\xi \times \epsilon)) - [\epsilon' \leftrightarrow \epsilon] \tag{19}$$

To encode this contraction in Mathematica, we ignore for now the symmetry term $-[\epsilon' \leftrightarrow \epsilon]$, and enter:

$\mathbb{C}\left[\{\epsilon'\}, \mathtt{t2}\left(2\mathtt{z}^2 - \mathtt{x}^2 - \mathtt{y}^2\right) + \mathtt{t3}\left(\mathtt{x}^3 - 3\mathtt{x}\mathtt{y}^2\right), \{\mathtt{i}\xi \times \epsilon\}\right] /. \mathtt{t2} \to 1$

$$-2\epsilon'.\hat{\mathbf{x}}(i\xi \times \epsilon).\hat{\mathbf{x}} - 2\epsilon'.\hat{\mathbf{y}}(i\xi \times \epsilon).\hat{\mathbf{y}} + 4\epsilon'.\hat{\mathbf{z}}(i\xi \times \epsilon).\hat{\mathbf{z}} \tag{20}$$

Condensing the spherical components into a single term gives (still ignoring the symmetry term)

`Simplify`[%, $(\epsilon'.\hat{\mathbf{x}}(\mathtt{i}\xi \times \epsilon).\hat{\mathbf{x}} + \epsilon'.\hat{\mathbf{y}}(\mathtt{i}\xi \times \epsilon).\hat{\mathbf{y}} + \epsilon'.\hat{\mathbf{z}}(\mathtt{i}\xi \times \epsilon).\hat{\mathbf{z}}) ==$
$\epsilon'.(\mathtt{i}\xi \times \epsilon)$]

$$-2\epsilon'.(i\xi \times \epsilon) + 6\epsilon'.\hat{\mathbf{z}}(i\xi \times \epsilon).\hat{\mathbf{z}} \tag{21}$$

Subtracting the symmetry term $[\epsilon' \leftrightarrow \epsilon]$, and removing constant prefactors, we see that the dipole-dipole scattering correction is proportional to:

```
Simplify[%/2 - symm]//TraditionalForm
```

$$-symm - \epsilon'.(i\xi \times \epsilon) + 3\epsilon'.\hat{\mathbf{z}}(i\xi \times \epsilon).\hat{\mathbf{z}} \tag{22}$$

This is Equation 29 in [3]. The $\epsilon'.(i\xi \times \epsilon)$ part merges with the term that appears in the spherical formulae, but the other term adds complexity to the amplitude dependence on experimental geometry.

Calculating the quadrupole-quadrupole scattering correction Mathematically, the contraction we are interested in is

$$\begin{gathered}\mathbb{C}((k' \otimes \epsilon')T(i\xi\times)(k \otimes \epsilon)) - [k', \epsilon' \leftrightarrow k, \epsilon] \\ = [\mathbb{C}((k' \otimes \epsilon')T((i\xi \times k) \otimes \epsilon)) + \mathbb{C}((k' \otimes \epsilon')T(k \otimes (i\xi \times \epsilon)))] \\ -[k', \epsilon' \leftrightarrow k, \epsilon]\end{gathered} \tag{23}$$

In Mathematica, this is encoded as (ignoring the $\{i\xi \times k, \epsilon\}$ term, and also the symmetry terms, for now):

```
ℂ[{k′, ϵ′}, t2 (2z^2 - x^2 - y^2) + t3 (x^3 - 3xy^2), {k, iξ × ϵ}] /.
t2 → 1
```

which gives output

$$\begin{gathered}-2\epsilon'.\hat{\mathbf{x}}(i\xi \times \epsilon).\hat{\mathbf{x}}\ \texttt{scal}[k, k'] - 2\epsilon'.\hat{\mathbf{y}}(i\xi \times \epsilon).\hat{\mathbf{y}}\ \texttt{scal}[k, k'] - \\ 2k'.\hat{\mathbf{x}}(i\xi \times \epsilon).\hat{\mathbf{x}}\ \texttt{scal}[k, \epsilon'] - 2k'.\hat{\mathbf{y}}(i\xi \times \epsilon).\hat{\mathbf{y}}\ \texttt{scal}[k, \epsilon'] - \\ 2k.\hat{\mathbf{x}}(\epsilon'.\hat{\mathbf{x}}\ \texttt{scal}[k', i\xi \times \epsilon] + k'.\hat{\mathbf{x}}\ \texttt{scal}[\epsilon', i\xi \times \epsilon]) - \\ 2k.\hat{\mathbf{y}}(\epsilon'.\hat{\mathbf{y}}\ \texttt{scal}[k', i\xi \times \epsilon] + k'.\hat{\mathbf{y}}\ \texttt{scal}[\epsilon', i\xi \times \epsilon]) + \\ 2(2\epsilon'.\hat{\mathbf{z}}(i\xi \times \epsilon).\hat{\mathbf{z}}\ \texttt{scal}[k, k'] + 2k'.\hat{\mathbf{z}}(i\xi \times \epsilon).\hat{\mathbf{z}}\ \texttt{scal}[k, \epsilon'] + \\ 2k.\hat{\mathbf{z}}(\epsilon'.\hat{\mathbf{z}}\ \texttt{scal}[k', i\xi \times \epsilon] + k'.\hat{\mathbf{z}}\ \texttt{scal}[\epsilon', i\xi \times \epsilon]))\end{gathered} \tag{24}$$

Condensing the spherical components into a single term,

```
Simplify[%,
{ϵ′.x̂(iξ × ϵ).x̂ + ϵ′.ŷ(iξ × ϵ).ŷ + ϵ′.ẑ(iξ × ϵ).ẑ == ϵ′.(iξ × ϵ),
k′.x̂(iξ × ϵ).x̂ + k′.ŷ(iξ × ϵ).ŷ + k′.ẑ(iξ × ϵ).ẑ == k′.(iξ × ϵ),
k.x̂ϵ′.x̂ + k.ŷϵ′.ŷ + k.ẑϵ′.ẑ == k.ϵ′,
k.x̂k′.x̂ + k.ŷk′.ŷ + k.ẑk′.ẑ == k.k′}]
```

$$\begin{gathered}-2(\epsilon'.(i\xi \times \epsilon)\ \texttt{scal}[k, k'] + k'.(i\xi \times \epsilon)\ \texttt{scal}[k, \epsilon'] - \\ 3k'.\hat{\mathbf{z}}(i\xi \times \epsilon).\hat{\mathbf{z}}\ \texttt{scal}[k, \epsilon'] + k.\epsilon'\ \texttt{scal}[k', i\xi \times \epsilon] - \\ 3\epsilon'.\hat{\mathbf{z}}((i\xi \times \epsilon).\hat{\mathbf{z}}\ \texttt{scal}[k, k'] + k.\hat{\mathbf{z}}\ \texttt{scal}[k', i\xi \times \epsilon]) + \\ k.k'\ \texttt{scal}[\epsilon', i\xi \times \epsilon] - 3k.\hat{\mathbf{z}}k'.\hat{\mathbf{z}}\ \texttt{scal}[\epsilon', i\xi \times \epsilon])\end{gathered} \tag{25}$$

and now taking only the non-spherical terms in F,

```
Simplify[%/.{ϵ′.(iξ × ϵ) → 0, k′.(iξ × ϵ) → 0, k.ϵ′ → 0, k.k′ → 0}]
```

$$6(\epsilon'.\hat{\mathbf{z}}((i\xi\times\epsilon).\hat{\mathbf{z}}\ \mathtt{scal}[k,k'] + k.\hat{\mathbf{z}}\ \mathtt{scal}[k',i\xi\times\epsilon]) + \\ k'.\hat{\mathbf{z}}((i\xi\times\epsilon).\hat{\mathbf{z}}\ \mathtt{scal}[k,\epsilon'] + k.\hat{\mathbf{z}}\ \mathtt{scal}[\epsilon',i\xi\times\epsilon])) \tag{26}$$

Adding the $\{i\xi\times k,\epsilon\}$ term back in now (by exchanging k and $i\xi\times\epsilon$ and adding the result on to the original formula), and rewriting `scal` in the dot-product notation, we find that the quadrupole-quadrupole scattering correction is proportional to:

```
Simplify[% + (%/.k → tmp/.ϵ → k/.tmp → ϵ)]/6/.
scal[vec1_, vec2_] → vec1.vec2//TraditionalForm
```

$$\epsilon'.\hat{\mathbf{z}}(k.\hat{\mathbf{z}}k'.(i\xi\times\epsilon) + \\ k'.(i\xi\times k)\epsilon.\hat{\mathbf{z}} + k'.\epsilon(i\xi\times k).\hat{\mathbf{z}} + k.k'(i\xi\times\epsilon).\hat{\mathbf{z}}) + \\ k'.\hat{\mathbf{z}}(\epsilon.\hat{\mathbf{z}}\epsilon'.(i\xi\times k) + k.\hat{\mathbf{z}}\epsilon'.(i\xi\times\epsilon) + \\ \epsilon.\epsilon'(i\xi\times k).\hat{\mathbf{z}} + k.\epsilon'(i\xi\times\epsilon).\hat{\mathbf{z}}) \tag{27}$$

which is equivalent to Equation 30 in [3].

Calculating the mixed (quadrupole-dipole) scattering correction For the quadrupole-quadrupole and dipole-dipole scattering corrections calculated above, only the t_2 (quadratic order) terms of T had any effect. The alternating term $\pm t_3(x^3-3xy^2)$ contributes to the quadrupole-dipole and dipole-quadrupole scattering factor corrections, due to its cubic order.

The mixed scattering correction is (ignoring for now the extra terms in $[\epsilon\leftrightarrow k]$ and $[k\leftrightarrow k',\epsilon\leftrightarrow\epsilon']$):

```
Expand[ℂ[{ϵ′}, t2(2z^2 − x^2 − y^2) + t3(x^3 − 3xy^2), {iξ × k, ϵ}]/.
{t2 → 0, t3 → 1}] + symm//TraditionalForm
```

$$symm + 6\epsilon.\hat{\mathbf{x}}\epsilon'.\hat{\mathbf{x}}(i\xi\times k).\hat{\mathbf{x}} - 6\epsilon.\hat{\mathbf{y}}\epsilon'.\hat{\mathbf{y}}(i\xi\times k).\hat{\mathbf{x}} - \\ 6\epsilon.\hat{\mathbf{y}}\epsilon'.\hat{\mathbf{x}}(i\xi\times k).\hat{\mathbf{y}} - 6\epsilon.\hat{\mathbf{x}}\epsilon'.\hat{\mathbf{y}}(i\xi\times k).\hat{\mathbf{y}} \tag{28}$$

where `symm` represents $[\epsilon\leftrightarrow k'] + [k\leftrightarrow k',\epsilon\leftrightarrow\epsilon']$.

After rearrangement according to the rules of the scalar triple product, this is equivalent to Equation 31 in [3]. This is the scattering factor contribution from the alternating term $\pm t_3(x^3-3xy^2)$, which is a dipole-quadrupole term.

4.4 The physical meaning of the contraction results

Extra peaks arising from asphericity of T The asphericity of the holmium atom results in extra peaks of Bragg order $2n\pm q$ (where $n\in\mathbb{N}$ and q is the antiferromagnetic wavevector) appearing alongside the peaks of order $2n$ that one would normally expect to observe. Here we will examine the magnetic contribution to the scattering amplitude of the peak appearing at $2n+q$, for four polarisation channels ($\sigma\to\sigma$, $\pi\to\sigma$, $\sigma\to\pi$ and $\pi\to\pi$).

From [3], the contribution to the scattering amplitude at this peak, for a given field tensor T, is given by

$$F_{\epsilon,k\to\epsilon',k'} = \mathbb{C}\left(\epsilon' T \sum_{n=1} (i\xi\times)^n F_1'^n \epsilon\right) + \mathbb{C}\left((k'\otimes\epsilon')T \sum_{n=1,3} (i\xi\times)^n F_2'^n (k\otimes\epsilon)\right)/2 \tag{29}$$

which originates from Equation 4, where only the terms in odd n are considered.

The F prefactors are defined as follows:

$$\begin{aligned}
\mathtt{F1'_0} &= \mathtt{F_{10}};\\
\mathtt{F1'_1} &= \tfrac{(\mathtt{F_{11}}-\mathtt{F_{1-1}})}{2};\\
\mathtt{F1'_2} &= \tfrac{(2\mathtt{F_{10}}-\mathtt{F_{11}}-\mathtt{F_{1-1}})}{2};\\
\mathtt{F2'_0} &= \mathtt{F_{20}};\\
\mathtt{F2'_1} &= \tfrac{(\mathtt{F_{2-2}}-\mathtt{F_{22}}+8\mathtt{F_{21}}-8\mathtt{F_{2-1}})}{12};\\
\mathtt{F2'_2} &= \tfrac{(16\mathtt{F_{21}}+16\mathtt{F_{2-1}}-\mathtt{F_{2-2}}-\mathtt{F_{22}}-30\mathtt{F_{20}})}{24};\\
\mathtt{F2'_3} &= \tfrac{(\mathtt{F_{22}}-\mathtt{F_{2-2}}+2\mathtt{F_{2-1}}-2\mathtt{F_{21}})}{12};\\
\mathtt{F2'_4} &= \tfrac{(6\mathtt{F_{20}}-\mathtt{F_{22}}-2\mathtt{F_{2-2}}-4\mathtt{F_{21}}-4\mathtt{F_{2-1}})}{24};
\end{aligned}$$

where the $F_{1,q}$ and $F_{2,q}$ originate from the treatment of the Holmium atom as the spherical case with a magnetic field perturbation, although we are not concerned with their definition here.

Redefining the scalar product We now want to create another definition of `scal`, so that it decomposes $\mathtt{scal}[\vec{a},\vec{b}]$ into $a_xb_x+a_yb_y+a_zb_z$—this is necessary as we will be using the definitions from Equations 16 and 17 to calculate the τ and θ dependence of the scattering amplitude.

First we define the scalar product function for various combinations of vectors, basis vectors and cross products with $\vec{\xi}$. We are making a new definition of `scal`, and so we remove the previous one.

$$\begin{aligned}
&\mathtt{Clear}[\mathtt{scal},\xi];\\
&\mathtt{Attributes}[\mathtt{scal}] = \{\mathtt{Orderless}\};
\end{aligned}$$

Now we define the scalar product of various expressions involving a cross-product with the magnetic basis vector $\vec{\xi}$, and any of the basis vectors $\vec{e}_1$, $\vec{e}_2$ or $\vec{e}_3$. In our co-ordinate system, $\vec{\xi} = (0,0,1) = \vec{e}_3$, so `scal` has been defined accordingly.

$$\begin{aligned}
&\mathtt{scal}\left[\mathtt{i}\xi\times(\mathtt{i}\xi\times(\mathtt{i}\xi\times\mathtt{vector_})),\mathtt{e_i_}\right] :=\\
&\mathtt{i}\left\{-\mathtt{vector_y},\mathtt{vector_x},0\right\}[[\mathtt{i}]];\\
&\mathtt{scal}\left[\mathtt{i}\xi\times(\mathtt{i}\xi\times\mathtt{vector_}),\mathtt{e_i_}\right] := \left\{\mathtt{vector_x},\mathtt{vector_y},0\right\}[[\mathtt{i}]];\\
&\mathtt{scal}\left[\mathtt{i}\xi\times\mathtt{vector_},\mathtt{e_i_}\right] :=\mathtt{i}\left\{-\mathtt{vector_y},\mathtt{vector_x},0\right\}[[\mathtt{i}]];\\
&\mathtt{scal}\left[\mathtt{vector_},\mathtt{e_i_}\right] :=\mathtt{vector}_{\{\mathtt{x,y,z}\}[[\mathtt{i}]]};
\end{aligned}$$

Now we make a similar definition for scalar products with other vectors, such as $\vec{k}$ or $\vec{\epsilon}$.

```
scal[iξ × (iξ × (iξ × vec1_)), vec2_]:=
i (−vec1_y vec2_x + vec1_x vec2_y) ;
scal[iξ × (iξ × vec1_), vec2_]:= (vec1_x vec2_x + vec1_y vec2_y) ;
scal[iξ × vec1_, vec2_]:=i (−vec1_y vec2_x + vec1_x vec2_y) ;
scal[vec1_, vec2_]:=
Sum [vec1_{x,y,z}[[i]] vec2_{x,y,z}[[i]], {i, 1, 3}] ;
```

Considering the expected form of the solution Knowing that for the case of this particular peak the magnetic contribution to the scattering amplitude will have terms in $i(F_{1,1} - F_{1,-1})$, $i(F_{2,1} - F_{2,-1})$ and $i(F_{2,2} - F_{2,-2})$, we can initialise a small array, `magterm`, to store these three terms.

```
magterm = {0, 0, 0};
```

Calculating the first part of the scattering amplitude expression Now we will try and find the first term, which will have a prefactor of $i(F_{1,1} - F_{1,-1})$). This term derives from the dipole-dipole contraction:

```
magterm[[1]] =
 Simplify[
 F1′1ℂ [{ϵ′}, t2 (2z^2 − x^2 − y^2) + t3 (x^3 − 3xy^2) , {iξ × ϵ}] /.
  t2 → 1/.F11 − F1−1 → 1/i]
```

After simplification, the result is

$$\epsilon_y \epsilon'_x - \epsilon_x \epsilon'_y \tag{30}$$

i.e. there is a contribution to the scattering amplitude of

$$i\left(F_{1,1} - F_{1,-1}\right)\left(\epsilon_y \epsilon'_x - \epsilon_x \epsilon'_y\right) \tag{31}$$

This will be evaluated later on in more detail from the definitions of ϵ, ϵ', k and k' in Equations 16 and 17, and re-expressed in terms of the incident angle θ and the antiferromagnetic parameters of the holmium system.

Calculating the second and third parts of the scattering amplitude expression We can now calculate the terms in $i(F_{2,1} - F_{2,-1})$ and $i(F_{2,2} - F_{2,-2})$. These originate from the contraction

$$\left(F_2'^1 \mathbb{C}\left((\vec{k}' \otimes \vec{\epsilon}')T(i\vec{\xi}\times)(\vec{k}' \otimes \vec{\epsilon}')\right) + F_2'^3 \mathbb{C}\left((\vec{k}' \otimes \vec{\epsilon}')T(i\vec{\xi}\times)^3(\vec{k}' \otimes \vec{\epsilon}')\right)\right)/2 \tag{32}$$

where an operation of $i\vec{\xi}\times$ on a tensor is expanded binomially as a sum of operations on the vector components of the tensor. In Mathematica, we calculate

```
baseformagterms2and3 =
 FullSimplify[
  F2′1 (ℂ [{k′, ϵ′}, t2 (2z^2 − x^2 − y^2) + t3 (x^3 − 3xy^2),
  {k, iξ × ϵ}]+
  ℂ [{k′, ϵ′}, t2 (2z^2 − x^2 − y^2) + t3 (x^3 − 3xy^2),
  {iξ × k, ϵ}])+
  F2′3
   (ℂ [{k′, ϵ′}, t2 (2z^2 − x^2 − y^2) + t3 (x^3 − 3xy^2),
  {k, iξ × (iξ × (iξ × ϵ))}]+
  3ℂ [{k′, ϵ′}, t2 (2z^2 − x^2 − y^2) + t3 (x^3 − 3xy^2),
  {iξ × k, iξ × (iξ × ϵ)}]+
  3ℂ [{k′, ϵ′}, t2 (2z^2 − x^2 − y^2) + t3 (x^3 − 3xy^2),
  {iξ × (iξ × k), iξ × ϵ}]+
  ℂ [{k′, ϵ′}, t2 (2z^2 − x^2 − y^2) + t3 (x^3 − 3xy^2),
  {iξ × (iξ × (iξ × k)), ϵ}])/.t2 → 1]/2
```

which gives the rather complicated result

$$\begin{gathered}\tfrac{1}{2}i\left(k_z\left(k'_z\left(\epsilon_y\epsilon'_x-\epsilon_x\epsilon'_y\right)+\left(-k'_y\epsilon_x+k'_x\epsilon_y\right)\epsilon'_z\right)\left(F_{2,-1}-F_{2,1}\right)+\right.\\ k_x\left(k'_z\epsilon_z\epsilon'_y\left(-F_{2,-1}+F_{2,1}\right)+\right.\\ k'_y\left(\epsilon_z\epsilon'_z\left(-F_{2,-1}+F_{2,1}\right)+2\left(\epsilon_x\epsilon'_x+\epsilon_y\epsilon'_y\right)\left(F_{2,-2}-F_{2,2}\right)\right)+\\ \left.2k'_x\left(\epsilon_y\epsilon'_x-\epsilon_x\epsilon'_y\right)\left(-F_{2,-2}+F_{2,2}\right)\right)+\\ k_y\left(k'_z\epsilon_z\epsilon'_x\left(F_{2,-1}-F_{2,1}\right)+\right.\\ k'_x\left(\epsilon_z\epsilon'_z\left(F_{2,-1}-F_{2,1}\right)-2\left(\epsilon_x\epsilon'_x+\epsilon_y\epsilon'_y\right)\left(F_{2,-2}-F_{2,2}\right)\right)+\\ \left.\left.2k'_y\left(\epsilon_y\epsilon'_x-\epsilon_x\epsilon'_y\right)\left(-F_{2,-2}+F_{2,2}\right)\right)\right)\end{gathered}\tag{33}$$

which can be used to calculate the second and third terms in F.

Calculating the second part of the scattering amplitude expression First we use Equation 33 to calculate the scattering amplitude term in $i(F_{2,1}-F_{2,-1})$. This involves taking only the coefficients of $iF_{2,1}$ (by symmetry, the $-iF_{2,-1}$ terms are identical), and discarding the other terms.

```
magterm[[2]] =
 FullSimplify[
   baseformagterms2and3/.F21 → 1/i/.F2−1 → 0/.F22 → 0/.F2−2 → 0]
```

$$\begin{gathered}\tfrac{1}{2}\left(k_z\left(k'_z\left(-\epsilon_y\epsilon'_x+\epsilon_x\epsilon'_y\right)+\left(k'_y\epsilon_x-k'_x\epsilon_y\right)\epsilon'_z\right)+\right.\\ \left.\epsilon_z\left(-k_y\left(k'_z\epsilon'_x+k'_x\epsilon'_z\right)+k_x\left(k'_z\epsilon'_y+k'_y\epsilon'_z\right)\right)\right)\end{gathered}\tag{34}$$

Calculating the third part of the scattering amplitude expression And now we find the term in $i(F_{2,2} - F_{2,-2})$, by the same method.

```
magterm[[3]] =
 FullSimplify[
   baseformagterms2and3/.F21 → 0/.F2−1 → 0/.F22 → 1/i/.F2−2 → 0]
```

$$\begin{array}{l} k_y\left(k'_y\left(\epsilon_y\epsilon'_x - \epsilon_x\epsilon'_y\right) + k'_x\left(\epsilon_x\epsilon'_x + \epsilon_y\epsilon'_y\right)\right) - \\ k_x\left(k'_x\left(-\epsilon_y\epsilon'_x + \epsilon_x\epsilon'_y\right) + k'_y\left(\epsilon_x\epsilon'_x + \epsilon_y\epsilon'_y\right)\right) \end{array} \tag{35}$$

Reducing the scattering amplitude expression The three terms in F that we have just calculated are:

```
magterm//TableForm
```

$$\epsilon_y\epsilon'_x - \epsilon_x\epsilon'_y \tag{36}$$

$$\begin{array}{l} \frac{1}{2}\left[k_z\left(k'_z\left(-\epsilon_y\epsilon'_x + \epsilon_x\epsilon'_y\right) + \left(k'_y\epsilon_x - k'_x\epsilon_y\right)\epsilon'_z\right)\right. \\ \left.+\epsilon_z\left(-k_y\left(k'_z\epsilon'_x + k'_x\epsilon'_z\right) + k_x\left(k'_z\epsilon'_y + k'_y\epsilon'_z\right)\right)\right] \end{array} \tag{37}$$

$$\begin{array}{l} k_y\left(k'_y\left(\epsilon_y\epsilon'_x - \epsilon_x\epsilon'_y\right) + k'_x\left(\epsilon_x\epsilon'_x + \epsilon_y\epsilon'_y\right)\right) \\ -k_x\left(k'_x\left(-\epsilon_y\epsilon'_x + \epsilon_x\epsilon'_y\right) + k'_y\left(\epsilon_x\epsilon'_x + \epsilon_y\epsilon'_y\right)\right) \end{array} \tag{38}$$

We introduce a shorthand notation and re-express the tensor products.

```
i {F11 − F1−1, F21 − F2−1, F22 − F2−2}
  × FullSimplify[magterm,
  {ϵxky + ϵykx==kϵxy, ϵxkz + ϵzkx==kϵxz, ϵykz + ϵzky==kϵyz,
 ϵ'xk'y + ϵ'yk'x==k'ϵ'xy, ϵ'xk'z + ϵ'zk'x==k'ϵ'xz,
 ϵ'yk'z + ϵ'zk'y==k'ϵ'yz, k'xϵ'x − k'yϵ'y==k'ϵ'x2my2,
  kxϵx − kyϵy==kϵx2my2}] //TableForm//TraditionalForm
```

So the reduced (simplified) form of the terms is

$$i\left(\epsilon_y\epsilon'_x - \epsilon_x\epsilon'_y\right)\left(F_{1,1} - F_{1,-1}\right) \tag{39}$$

$$\tfrac{1}{2}i\left(F_{2,1} - F_{2,-1}\right)\left(k\epsilon_{x,z}k'\epsilon'_{y,z} - k\epsilon_{y,z}k'\epsilon'_{x,z}\right) \tag{40}$$

$$i\left(F_{2,2} - F_{2,-2}\right)\left(k'\epsilon'_{x2my2}k\epsilon_{x,y} - k\epsilon_{x2my2}k'\epsilon'_{x,y}\right) \tag{41}$$

Finding the scattering amplitude matrices Now we will evaluate the expression for the scattering amplitude. Since ϵ and ϵ' can both be in either the π or the σ polarisation, we construct a 2×2 matrix to represent the four combinations of $\epsilon \to \epsilon'$ in terms of θ and τ.

First we evaluate the matrix $F = \begin{pmatrix} F_{\sigma'\leftarrow\sigma} & F_{\sigma'\leftarrow\pi} \\ F_{\pi'\leftarrow\sigma} & F_{\pi'\leftarrow\pi} \end{pmatrix}$ or rather its three constituent terms which originate from the scattering amplitude expression calculated above.

```
Fmatrix:=
 Table[
  Simplify[
   magterm[[i]]/. {vector_xyz_ → vector[θ, τ][[xyz]]} /.
     {ϵ → ϵπσ, ϵ' → ϵ'π'σ'} /.{x → 1, y → 2, z → 3}] , {i, 1, 3},
  {π'σ', 0, 1}, {πσ, 0, 1}];
```

The three terms are

```
Table[Fmatrix[[i]]//MatrixForm, {i, 1, 3}]//TableForm//
 TraditionalForm
```

$$\begin{pmatrix} 0 & -\mathrm{c}(\theta)\mathrm{c}(\tau) \\ \mathrm{c}(\theta)\mathrm{c}(\tau) & -2\mathrm{c}(\theta)\mathrm{s}(\theta)\mathrm{s}(\tau) \end{pmatrix}$$
$$\begin{pmatrix} -\frac{1}{2}\mathrm{c}(2\tau)\mathrm{s}(2\theta)\mathrm{s}(\tau) \\ \frac{1}{8}\mathrm{c}(\theta)\mathrm{c}(\tau)(-2\mathrm{c}(2\theta)+3\mathrm{c}(2(\theta-\tau))-2\mathrm{c}(2\tau)+3\mathrm{c}(2(\theta+\tau))+2) \\ \frac{1}{8}\mathrm{c}(\theta)\mathrm{c}(\tau)(2\mathrm{c}(2\theta)-3\mathrm{c}(2(\theta-\tau))+2\mathrm{c}(2\tau)-3\mathrm{c}(2(\theta+\tau))-2) \\ \frac{1}{2}\mathrm{c}^2(\tau)\mathrm{s}(4\theta)\mathrm{s}(\tau) \end{pmatrix} \quad (42)$$
$$\begin{pmatrix} -2\mathrm{c}(\theta)\mathrm{c}^2(\tau)\mathrm{s}(\theta)\mathrm{s}(\tau) \\ \mathrm{c}(\theta)\mathrm{c}(\tau)\left(\mathrm{s}^2(\theta)\left(2\mathrm{s}^2(\tau)+1\right)-\mathrm{c}^2(\theta)\mathrm{s}^2(\tau)\right) \\ \mathrm{c}(\theta)\mathrm{c}(\tau)\left((\mathrm{c}(2\tau)-2)\mathrm{s}^2(\theta)+\mathrm{c}^2(\theta)\mathrm{s}^2(\tau)\right) \\ \frac{1}{8}\mathrm{s}(4\theta)(\mathrm{s}(3\tau)-7\mathrm{s}(\tau)) \end{pmatrix}$$

where the last two 4×4 matrices have been condensed into a columnar form, and the abbreviations s and c stand for sin and cos respectively.

Fourier analysis of the F matrix Since we are interested in the $+q$ satellite, we need to find the $e^{i\tau}$ Fourier component of the matrix F. The following code breaks F into $e^{in\tau}$ components and extracts the first Fourier coefficients.

```
Fourier1 =
Table[
 TrigReduce[
  PadRight[CoefficientList[
   Expand[TrigExpand[Fmatrix[[i, var1, var2]]]]/.
    Sin[n_.τ] → 1/(2i) (e^(inτ) − e^(−inτ)) /.
     Cos[n_.τ] → 1/2 (e^(inτ) + e^(−inτ))] /.e^(iτ) → TERM, TERM] , 2] [[
   2]]], {i, 1, 3}, {var1, 1, 2}, {var2, 1, 2}];
```

So the coefficient of the scattering amplitude matrix multiplying the $e^{i\tau}$ Fourier

component is:

```
Do[Print[" + ", i {F11 − F1−1, F21 − F2−1, F22 − F2−2}[[i]]//
  TraditionalForm, Fourier1[[i]]//MatrixForm//
  TraditionalForm], {i, 1, 3}]
```

$$
\begin{aligned}
& i\,(F_{1,1}-F_{1,-1})\begin{pmatrix} 0 & -\frac{\mathrm{c}(\theta)}{2} \\ \frac{\mathrm{c}(\theta)}{2} & \frac{1}{2}i\mathrm{s}(2\theta) \end{pmatrix} \\
+ & i\,(F_{2,1}-F_{2,-1})\begin{pmatrix} -\frac{1}{8}i\mathrm{s}(2\theta) & \frac{1}{32}(3\mathrm{c}(\theta)+\mathrm{c}(3\theta)) \\ \frac{1}{32}(-3\mathrm{c}(\theta)-\mathrm{c}(3\theta)) & -\frac{1}{16}i\mathrm{s}(4\theta) \end{pmatrix} \\
+ & i\,(F_{2,2}-F_{2,-2})\begin{pmatrix} \frac{1}{8}i\mathrm{s}(2\theta) & \frac{1}{32}(3\mathrm{c}(\theta)-7\mathrm{c}(3\theta)) \\ \frac{1}{32}(7\mathrm{c}(3\theta)-3\mathrm{c}(\theta)) & \frac{7}{16}i\mathrm{s}(4\theta) \end{pmatrix}
\end{aligned}
\tag{43}
$$

The above expression is the scattering amplitude matrix for the $n+q$ satellite diffraction peak of holmium[2]. This can be used to obtain theoretical predictions for the amplitudes of the various scattering peaks for photons of different energies—which can then be compared with measurements made at the ESRF.

5 Conclusions

We have taken the contraction method established in [3] and implemented it as an efficient computer algorithm in a Mathematica program. The program was designed to take a general (polynomial) field tensor and contract it with various combinations of $\vec{k}$ and $\vec{\epsilon}$ with the magnetic vector $\vec{\xi}$ acting as a rotation derivative. The program output scattering amplitude expressions in terms of components of $\vec{k}$ and $\vec{\epsilon}$.

An example of the program operation was presented here for the case of spiral antiferromagnetic holmium, where the vector $\vec{\xi}$ rotates round through the crystal layering structure with an antiferromagnetic wavevector q. Using the non-spherical field tensor for this material, the dipole-dipole, quadrupole-quadrupole and mixed dipole-quadrupole corrections to the scattering factors which were obtained for the case of a spherical atom [2] were calculated.

The contraction method was then used to find the amplitude of the diffraction peak with Bragg number $2n \pm q$, for the $\sigma' \leftarrow \sigma$, $\sigma' \leftarrow \pi$, $\pi' \leftarrow \sigma$ and $\pi' \leftarrow \pi$ polarisations, via a Fourier analysis of the scattering amplitude function. These scattering amplitudes could then be used to predict the intensities of the satellite peaks observed in RMXS experiments on Holmium.

Further work This program can be developed further to allow for more complex forms of field tensors, such as those involving cross products with the

[2]This can be shown by considering components of a wave incident at angle θ being reflected off adjacent crystal layers z_1, z_2 with scattering coefficients $e^{i\tau_1}$ and $e^{i\tau_2}$. The electric fields of the two reflected waves will be given by $e^{i\tau_1}e^{ikx}$ and $e^{i\tau_2}e^{ik(x+2d\sin\theta)}$. The condition for these to be in phase leads to $2d\sin\theta = q\lambda$, a modified Bragg diffraction condition.

photon tensors (see Appendix A). The reliability of the contraction method [3] in predicting scattering peaks for other materials is yet to be tested.

The extension of the program to cope with contractions of more than three tensors is relatively trivial, involving only the addition of a few more rules to reduce the number of tensors in the same way that the present program reduces it from three to two.

A more complicated problem would be to contract a set of tensors with restrictions on the possible contractions, such as in Equation 25 of [3].

6 Acknowledgements

I am grateful to Alessandro Mirone for his advice on the programming and for his patience in explaining the theoretical details of his paper [3].

A Extension of the program

A.1 Tensor contraction

Two tensors First we define the contraction sum for two tensors of any rank.

```
Attributes[Con] = {Orderless};
Attributes[T] = {Orderless};
Con[x__, 0]:=0;

Con[T[a_, al___], T[bl___]]:=Module[{res}, res = 0;
n = Length[List[bl]];
Do[
res = res + scal [a/. {x → e1, y → e2, z → e3} , List[bl][[i]]/. {x → e1, y → e2, z → e3}]
ConTmp[T[al], Delete[T[bl], i]]
, {i, 1, n}
];
res
];
Con[T[c___]]:=0;
Con[1, t___]:=Con[t];
Con[]:=1;
T[] = 1;
T[{}] = 1;
T[{x_}] = T[x];
```

Three tensors Now we define the contraction sum for three tensors (by first eliminating one of the three, and then reverting to the two-tensor contraction).

```
Con[T[al__], T[bl__], T[cl__]]:=Module[{res, m, n}, res = 0;
m = Length[T[bl]];
n = Length[T[cl]];
Do[
res =
res + scal [List[al][[1]]/. {x → e1, y → e2, z → e3} ,
List[bl][[i]]/. {x → e1, y → e2, z → e3}]
ConTmp[Delete[T[al], 1], Delete[T[bl], i], T[cl]]
, {i, 1, m}
];
Do[
res =
res + scal [List[al][[1]]/. {x → e1, y → e2, z → e3} ,
List[cl][[i]]/. {x → e1, y → e2, z → e3}]
ConTmp[Delete[T[al], 1], T[bl], Delete[T[cl], i]]
, {i, 1, n}
];
res
];
```

As a temporary measure (for debugging), we make all `ConTmp` be executed as `Con`.

```
ConTmp = Con;
```

A.2 Contracting polynomials with cross products

We will construct several functions, all defined by rules, along which the contraction is passed. Each function executes its own simplifications and contractions, and then invokes the next one in the sequence.

Expanding brackets The function ℂ rewrites the polynomial in expanded form, and passes the contraction to ℂ2, the second contracting function.

```
ℂ[al_, anything_, cl_]:=ℂ2[al, Expand[anything], cl];
```

Expanding contractions of sums If we are a contracting a sum of terms, then ℂ3 splits it into a sum of contractions...

```
ℂ2[al_, Plus[thing1_, morethings__], cl_]:=
Sum[ℂ2[al, Join[List[thing1], List[morethings]][[i]], cl],
{i, 1, Length[Join[List[thing1], List[morethings]]]}];
```

...until we we have nothing left to expand into sums: then it passes the contraction on to ℂ3, the third contraction function

```
ℂ2[al_, Sum[onething_], cl_]:=ℂ3[al, onething, cl];
```

Taking sums outside $\times$ operator Now ℂ3 expands the sums within cross products into sums of contractions

```
ℂ3[al_,
Times[prefactors__, vec1_ × Plus[vec2a_, vec2b_]], cl_]:=
Times@@Complement[{prefactors}, {x, y, z, x, y, z, x-, y-, z-}]
Times[
Sum[ℂ3[al, Times@@Intersection[{prefactors}, {x, y, z, x, y, z, x-, y-, z-}]
vec1 × (vec2a + vec2b)[[i]], cl], {i, 1, Length[vec2a + vec2b]}]];
ℂ3[al_,
Times[prefactors__, Plus[vec1a_, vec1b_] × vec2_], cl_]:=
Times@@Complement[{prefactors}, {x, y, z, x, y, z, x-, y-, z-}]
Times[
Sum[ℂ3[al, Times@@Intersection[{prefactors}, {x, y, z, x, y, z, x-, y-, z-}]
(vec1a + vec1b)[[i]] × vec2, cl], {i, 1, Length[vec1a + vec1b]}]];
```

Taking powers outside $\times$ operator Now ℂ3 can also expand powers within cross products, and re-write the expression as a sum of cross products with each of the variable on the right of the cross-product operator.

```
ℂ3[al_, Times[prefactors__, Cross[a_, Power[xyz1_, power_]]], cl_]:=
Module[{i, xyz}, xyz = Table[xyz1, {i, 1, power}];
Times@@Complement[{prefactors}, {x, y, z, x, y, z, x-, y-, z-}]
Sum[
ℂ3[al,
Times[Times@@Intersection[{prefactors},
 {x, y, z, x, y, z, x-, y-, z-}], Times@@Delete[xyz, i], cross[a, xyz[[i]]]], cl],
{i, 1, Length[xyz]}]
]
```

Taking products outside $\times$ operator ℂ3 also expands products within cross products, similarly re-expressing the polynomial as a sum of cross product operations on each of the variables on the right of the $\times$ sign.

```
ℂ3[al_, Times[prefactors__, Cross[a_, Times[xyz1_, xyz2_]]], cl_]:=
Module[{i, xyz},
xyz = Flatten[Join[List[xyz1], List[xyz2]]/.
{anything_power- → Table[anything, {i, 1, power}]}];
Times@@Complement[{prefactors}, {x, y, z, x, y, z, x-, y-, z-}]
Sum[
ℂ3[al,
Times[Times@@Intersection[{prefactors},
 {x, y, z, x, y, z, x-, y-, z-}], Times@@Delete[xyz, i], cross[a, xyz[[i]]]], cl],
{i, 1, Length[xyz]}]
```

```
]
```

Introducing the `cross` operator We now re-express any simple cross product within a contraction as a function of the cross product operator.

```
ℂ3[al_, Times[prefactors___, Cross[a_, b_]], cl_]:=
Times@@Complement[{prefactors}, {x, y, z, x, y, z, x⁻, y⁻, z⁻}]
ℂ3[al, Times@@Intersection[{prefactors}, {x, y, z, x, y, z, x⁻, y⁻, z⁻}] cross[a, b],
cl]
```

Defining the cross product operator

```
cross[a_, b_]:=
Cross[(a/.{x → x, y → y, z → z}/.{x → {1, 0, 0}, y → {0, 1, 0}, z → {0, 0, 1}}),
(b/.{x → x, y → y, z → z}/.{x → {1, 0, 0}, y → {0, 1, 0}, z → {0, 0, 1}})].{x, y, z}
```

Correcting a small glitch We will get some dodgy terms arising from the program attempting to also contract minus signs as separate entities. We can patch over this with the following rule:

```
cross[_, −1]:=0
```

A.3 Contracting polynomials now that the cross-products have been removed

Now the cross products in contractions have been all re-expressed as simple sums of scalar products, we can continue with the contractions as normal.

Taking summations outside The result from all the above simplifications might still be expandable: so any summations must therefore be taken outside the contraction sign again.

```
ℂ3[{al__}, Plus[x_, y__], {cl__}]:=
Module[{i}, xy = Expand[x + y]; Sum[ℂ3[{al}, xy[[i]], {cl}], {i, 1, Length[xy]}]]
```

Executing the contractions Now we need to find an efficient way to take each term and extract the xyz-dependent components.

First we separate out the terms containing xyz from the rest—this is so constant factors *etc.* don't get accidentally contracted when they shouldn't.

```
ℂ3[{al__},
Times[prefactor_/;FreeQ[FullForm[prefactor], x]/;FreeQ[FullForm[prefactor], y]/;
FreeQ[FullForm[prefactor], z], xyz__], {cl__}]:=
```

```
Times[prefactor]ℂ3[{al}, Times[xyz], {cl}];
```

Now we convert the polynomial (which is now in xyz form) into a tensorial form that can be used directly by the function `Con`.

```
ℂ3[{al_}, Times[xyz1_, xyz2_], {cl_}]:=Module[{i, power, n, xyz},
xyz:=List[xyz1, xyz2]/.x_^power-- >Table[x, {i, 1, power}];
n = Length[xyz];
Con[T[al], T[Flatten[Table[xyz[[i]], {i, 1, n}]]], T[cl]]
]
ℂ3[{al_}, Power[xyz_, power_], {cl_}]:=Module[{i, n},
Con[T[al], T[Table[xyz, {i, 1, power}]], T[cl]]
]
ℂ3[al_, 0, cl_]:=0;
```

A.4 Defining the scalar product symbolically

```
Clear[scal, ξ];
Attributes[scal] = {Orderless};
scal [vector_, e_i_] :=vector.{x, y, z}[[i]];
scal[vec1_, vec2_]:=vec1.vec2;
```

A.5 Application: chiral system

The contraction mechanism has been defined to work on any kind of polynomial tensor involving a cross-product with another vector. Now we test it on the case of a chiral system, which has the field tensor

$$T = (1 + z\alpha\mathbf{z}\times)\left(x^2 - y^2\right) \tag{44}$$

Dipole-dipole correction We can see that this result is what you would expect from a contraction of $x^2 - y^2$ with ϵ' and ϵ—the cross-product term in α has no effect as it has rank 3:

```
ℂ [{ϵ′}, (x^2 − y^2) + αzz × (x^2 − y^2) , {ϵ}]
```

$$2\epsilon.\mathbf{x}\epsilon'.\mathbf{x} - 2\epsilon.\mathbf{y}\epsilon'.\mathbf{y} \tag{45}$$

Mixed (dipole-quadrupole) correction Here only the term in α affects the result.

```
ℂ [{ϵ′}, (x^2 − y^2) + αzz × (x^2 − y^2) , {k, ϵ}]
```

$$\begin{aligned}\alpha\,(&4(k.\mathbf{z}\epsilon.\mathbf{y} + k.\mathbf{y}\epsilon.\mathbf{z})\epsilon'.\mathbf{x} + 4(k.\mathbf{z}\epsilon.\mathbf{x} + k.\mathbf{x}\epsilon.\mathbf{z})\epsilon'.\mathbf{y}\\ &+4(k.\mathbf{y}\epsilon.\mathbf{x} + k.\mathbf{x}\epsilon.\mathbf{y})\epsilon'.\mathbf{z})\end{aligned} \tag{46}$$

Quadrupole-quadrupole correction Now we are looking at the whole expression for the field tensor

$$\mathbb{C}\left[\{\mathtt{k}', \epsilon'\}, (\mathtt{x}^2 - \mathtt{y}^2) + \alpha \mathtt{zz} \times (\mathtt{x}^2 - \mathtt{y}^2), \{\mathtt{k}, \epsilon\}\right]$$

$$\begin{array}{c} 2k.\epsilon' k'.\mathbf{x}\epsilon.\mathbf{x} - 2k.\epsilon' k'.\mathbf{y}\epsilon.\mathbf{y} + 2k.k'\epsilon.\mathbf{x}\epsilon'.\mathbf{x} \\ +2k.\mathbf{x}(k'.\mathbf{x}\epsilon.\epsilon' + k'.\epsilon\epsilon'.\mathbf{x}) - 2k.k'\epsilon.\mathbf{y}\epsilon'.\mathbf{y} - 2k.\mathbf{y}(k'.\mathbf{y}\epsilon.\epsilon' + k'.\epsilon\epsilon'.\mathbf{y}) \end{array} \tag{47}$$

References

[1] P. Carra, B.T. Thole, Phys. Rev. B. **43**, 13401 (1991)

[2] J.P. Hill, D.F. McMorrow, Acta Cryst. A **52**, 236 (1996)

[3] A. Mirone, L. Bouchenoire, S. Brown, `http://arxiv.org/abs/cond-mat/0606144`

[4] R. Snieder, *A Guided Tour of Mathematical Methods for the Physical Sciences*, C.U.P. (2001)