GAMMA

Rank 2 Interactions

$$\hat{\hat{\Gamma}}$$

Author: Dr. Scott A. Smith

Date: August 24, 1999

Table of Contents

1 Introduction

1.1 Overview

GAMMA is used to facilitate simulations of magnetic resonance phenomena. This book details the classes and modules which treat spin and spin-spin interactions of rank 2. In particular, shift anisotropy, dipolar, and quadrupolar interactions.

1.2 Why Read This Documentation

The classes described herein are NOT particularly useful in themselves unless one is interested in a single interaction associated with a single spin or spin-pair. Rather, they are used by GAMMA's spin system class(es) to build up a spin interaction picture over sets of spins. Thus, these classes form the foundation for many caluclations of oriented spin systems through use of a spin system, normally without the user dealing directly with specific interactions or their associated class.

If you wish to know the full details of how individual rank 2 interactions work (perhaps you wish to add in another one?) read on. However, there are few things you can do direcltly with these classes that you cannot do in a more general context (and more intuitively) using the proper spin system class. For primitives just use a single or two-spin system with only a single interaction defined.

1.3 Chemical Shifts

One of the most commonly measured features in NMR is the chemical shift. For any particular spin in a static magnetic field $\boldsymbol{B}_o$, the resonance frequency of a particular spin in the laboratory frame is given by

$$\Omega = \gamma \boldsymbol{B}_o (1 - \sigma_{iso}) \tag{0-1}$$

where γ is the gyromagnetic ratio of the spin and σ_{iso} the isotropic component of the spin's shielding tensor. In this context σ_{iso} is a unitless quantity, typically on the order of 10^{-6}, and at common field strengths Ω will be hundreds of MHz[1].

Rather than view NMR from the laboratory frame, it is more convenient (both mathematically and

1. A spin which has no shielding, $\sigma_{iso} = 0$, corresponds to the case when there is a bare nucleus.

experimentally) to work in rotating frames, rather than the laboratory frame. Measured frequencies are referenced to a specific frequency, or multiple specified frequencies in a heteronuclear experiment, and the user normally sees only the differences between the absolute frequency and the reference frequency. We shall label such rotating frame frequencies using small ω as opposed to large Ω in the lab frame. The relationship between the two is given by

$$\omega = \Omega - \Omega_{ref} \tag{0-2}$$

The reference frequency Ω_{ref} is normally placed close to the larmor frequency of the isotope type being detected, $\Omega_{ref} \sim \gamma \boldsymbol{B}_o = \Omega_0$, so that the measured frequencies are relatively small. Obviously, $\omega_{ref} = 0$.

It is common practice to refer to ω as the **chemical shift** although these values are dependent on the reference frequency and only indirectly related to the chemical shielding, the latter being that which is the cause of chemical shifts.

Because, even for a particular isotope, there are many different reference frequencies and because the absolute frequencies measured will depend upon the applied field strength, the IUPAC recommended standard is to report chemical shifts in PPM. We shall call this the “delta scale” and follow the discussion found in Duncan[1]. The frequency is then given in PPM by

$$\delta(PPM) = \frac{\Omega - \Omega_{ref}}{\Omega_{ref}} \times 10^6 = \frac{\omega}{\Omega_{ref}} \times 10^6 \tag{0-3}$$

These values are still dependent upon the chosen reference but are “nearly” independent of the applied field strength, as seen by substitution

$$\begin{aligned}\delta(PPM) &= \frac{\Omega - \Omega_{ref}}{\Omega_{ref}} \times 10^6 = \frac{\gamma \boldsymbol{B}_o(1-\sigma_{iso}) - \gamma \boldsymbol{B}_o(1-\sigma_{iso,ref})}{\Omega_{ref}} \times 10^6 \\ &= \frac{\gamma \boldsymbol{B}_o[\sigma_{iso,ref} - \sigma_{iso}] - \gamma \boldsymbol{B}_o(1-\sigma_{iso,ref})}{\gamma \boldsymbol{B}_o(1-\sigma_{iso,ref})} \times 10^6 = \frac{\sigma_{iso,ref} - \sigma_{iso}}{1-\sigma_{iso,ref}} \times 10^6\end{aligned}$$

Notice that the field strength, $\boldsymbol{B}_o$ is absent from the last form. We can write

$$\delta = \frac{\sigma_{iso,ref} - \sigma_{iso}}{1-\sigma_{iso,ref}} \times 10^6 \approx (\sigma_{iso,ref} - \sigma_{iso}) \times 10^6 \tag{0-4}$$

where the approximation shown assumes that $\sigma_{reference} \ll 1$, most often true because the isotropic

1. T. Michael Duncan, “A Compilation of Chemical Shift Anisotropies”, The Farragut Press, Chicago, 1990.

values of most shielding tensors are small, typically on the order of 10^{-6}. The figure below illustrates the relationship between the differing aspects of chemical shifts and isotropic shieldings.

Comparison of Shielding vs. Reported Shifts in a ^{13}C Spectrum

Downfield
High Frequency
Less Shielding

Upfield
Low Frequency
More Shielding

	Bare ^{13}C Nucleus	Arbitrary Transition	Reference Transition
Ω	***100.018543 MHz***	***100.010000 MHz***	***100.000000 MHz***
$\sigma \times 10^6$	***0.00***	***85.41***	***185.40***
ω	***18.543 kHz***	***10.00 kHz***	***0.00***
δ	***185.43 PPM***	***100.0 PPM***	***0.00 PPM***

Figure 19-1 : Comparison of different shifts in a ^{13}C spectrum. The static magnetic field is 9.4 Tesla (^{1}H = 400 MHz, ^{13}C =100.02 MHz) and TMS is used for the standard reference (185.4 PPM).

Explicit computation of the observed absolute frequencies in the previous figures are

$$\Omega_{bare} = \gamma \boldsymbol{B}_o = 100.018543 \text{ MHz}$$

$$\Omega_{trans} = \gamma \boldsymbol{B}_o(1-\sigma) = 100.018543 \text{ MHz}(1 - 85.41 \times 10^{-6}) = 100.010000 \text{ MHz}$$

$$\Omega_{ref} = \gamma \boldsymbol{B}_o(1-\sigma) = (100.018543 \text{ MHz}(1 - 185.40 \times 10^{-6}) = 100.000000 \text{ MHz})$$

and a quick calculation of the reported shifts is

$$\delta_{bare} \approx \sigma_{reference} - \sigma_{bare} = 185.40 \times 10^{-6} - 0 = 185.4 \text{ PPM}$$

$$\delta_{trans} \approx \sigma_{reference} - \sigma_{trans} = 185.40 \times 10^{-6} - 84.51 \times 10^{-6} = 99.9 \text{ PPM}$$

As a rule of thumb, the larger the chemical shielding the lower the resonance frequency.

2 Rank 2 Interaction Spatial Tensors

2.1 Overview

The class ***IntRank2AA*** contains the essence an irreducible rank 2 spatial tensor. The class serves as the base class for specific rank 2 interactions (e.g. quadrupolar, shift anisotropy, dipolar...) of interest.

2.2 Available Functions

I/O Functions

2.3 Theory

2.4 Figures

2.5 Literature Comparisons

2.6 Examples

2.7 Example Programs

2.8 Constructors

2.8.1 IntRank2AA

Usage:

```
void IntRank2A::IntRank2A(double eta=0, double theta=0, double phi=0)
void IntRank2A::IntRank2A(const IntRank2A& R2)
void IntRank2A::IntRank2A(ParameterAVLSet& pset, int idx=-1)
```

Description:

The function ***IntRank2A*** is used to create a new rank 2 interaction.

1. IntRank2A(double delzz, double eta=0.0, double theta=0.0, double phi=0.0) - This will construct a new rank 2 interaction. The value of ***eta*** can be optionally input and this set the asymmetry of the interaction. It is restricted to be within the range ***[0, 1]***. Two angles, ***theta*** and ***phi***, can be optionally specified in ***degrees***. This will set the orientation of the rank 2 interaction relative to its PAS.
2. IntRank2A(const IntRank2A& R2) - Called with another rank 2 interaction, a new rank 2 interaction is constructed which is identical to the input interaction.
3. IntRank2A(const coord& AxAyAz, double theta=0.0, double phi=0.0) -A new rank 2 interaction is constructed which has its Cartesian PAS components set from AxAyAz and is oriented at angles ***theta*** and ***phi*** (specified in degrees).
4. IntRank2A(ParameterAVLSet& pset, int idx=-1) - This will construct a new rank 2 interaction from parameters found in the parameter set ***pset***. If the optional index ***idx*** has been set ***>=0*** the rank 2 parameters scanned in ***pset*** will be assumed to have a ***(idx)*** appended to their names.

Return Value:

Void. It is used strictly to create a rank 2 interaction.

Examples:

```
#include <gamma.h>
main()
  {
  IntRank2A R2;                    // An empty rank 2 interaction.
  IntRank2A R2a(0.2, 45., 30.);    // Interaction with η=0.2, θ=45.0, ϕ=30.0
  IntRank2A R2b(R2a);              // Another interaction, here equal to R2a
  }
```

See Also: =, read, ask_read

2.8.2 =

Usage:

```
void IntRank2A::operator= (const IntRank2A& R2)
```

Description:

The operator = is assign one rank 2 interaction to another.

Return Value:

Void.

Example:

```
#include <gamma.h>
main()
  {
  IntRank2A R2a(0.2, 45., 30.);          // Interaction with η=.2, θ=45, ϕ=30
  IntRank2A R2= R2a;                     // Now R2 is the same as R2a
  }
```

See Also: constructor, read, ask_read

2.9 Spherical Spatial Tensor Functions

2.9.1 A0, A20

Usage:

```
#include <IntRank2A.h>
complex IntRank2A::A0() const
complex IntRank2A::A20() const
complex IntRank2A::A0(double theta, double phi) const
complex IntRank2A::A20(double theta, double phi) const
```

Description:

The functions ***A0*** and ***A20*** are used to obtain the rank 2 interaction spatial tensor component $A_{2,0}$. If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments ***theta*** and ***phi*** are given the returned tensor component is for the orientation at ***theta*** degrees down from the interactions PAS z-axis and ***phi*** degrees over from the interactions PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***Hz***.

$$A_{2,0}(\theta,\varphi) = \sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]$$

Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type[1]. This component can also be related to the Cartesian tensor components for any arbitrary orientation.

$$A_{2,0} = \sqrt{6}[3A_{zz} - Tr\{A\}]$$

Return Value:

A complex number.

Example:

```
#include <gamma.h>
main()
  {
  IntRank2A R2(0.2, 45.0, 45.0);        // Make a rank 2 interaction.
  complex v20 = R2.A20();               // This is at theta=phi=45 degrees
  cout << R2.A20(15.6, 99.3);           // This is at theta=15.6 and phi=99.3 degrees.
  }
```

See Also: A1, A21, Am1, A2m1, A2, A22, Am2, A2m2

---

1. To accommodate different interaction types, GAMMA scaling on all spatial tensors is chosen to be independent of the interaction. Rather, the spatial tensors are related directly to the familiar rank two spherical harmonics $A_{2,m}(\theta,\varphi)\big|_{\eta=0} = Y_m^2(\theta,\varphi)$. Also, the sign on the term(s) involving η will have opposite sign if the common alternative definition of the PAS orientation ($|A_{zz}| \geq |A_{xx}| \geq |A_{yy}|$) is used rather that the definition used in GAMMA ($|A_{zz}| \geq |A_{yy}| \geq |A_{xx}|$)

2.9.2 A1, A21

Usage:

```
#include <IntRank2A.h>
complex IntRank2A::A1() const
complex IntRank2A::A21() const
complex IntRank2A::A1(double theta, double phi) const
complex IntRank2A::A21(double theta, double phi) const
```

Description:

The functions ***A1*** and ***A21*** are used to obtain the rank 2 interaction spatial tensor component $A_{2,1}$. If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments ***theta*** and ***phi*** are given the returned tensor component is for the orientation at ***theta*** degrees down from the interactions PAS z-axis and ***phi*** degrees over from the interactions PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***Hz***.

$$A_{2,1}(\theta, \varphi) = \sqrt{\frac{5}{24\pi}} \sin\theta [3\cos\theta - \eta(\cos\theta\cos 2\varphi - i\sin 2\varphi)]$$

Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type[1]. This component can also be related to the Cartesian tensor components for any arbitrary orientation.

$$A_{21} = -\frac{1}{2}[A_{xz} + A_{zx} + i(A_{yz} + A_{zy})]$$

Return Value:

A complex number.

Example:

```
#include <gamma.h>
main()
  {
  IntRank2A R2(0.2, 45.0, 45.0);        // Make a rank 2 interaction.
  complex v1 = R2.A21();                // This is at theta=phi=45 degrees
  cout << R2.A21(15.6, 99.3);           // This is at theta=15.6 and phi=99.3 degrees.
  }
```

See Also: A1, A21, Am1, A2m1, A2, A22, Am2, A2m2

1. Because the GAMMA platform accommodates different interaction types, the scaling on all spatial tensors is chosen to be independent of the interaction. Rather, the spatial tensors are related directly to the familiar rank two spherical harmonics $A_{2,m}(\theta, \varphi)\big|_{\eta = 0} = Y_m^2(\theta, \varphi)$. Also, the sign on the term(s) involving η will have opposite sign if the common alternative definition of the PAS orientation ($|A_{zz}| \geq |A_{xx}| \geq |A_{yy}|$) is used rather that the definition used in GAMMA ($|A_{zz}| \geq |A_{yy}| \geq |A_{xx}|$)

2.9.3 Am1, A2m1

Usage:

```
#include <IntRank2A.h>
complex IntRank2A::Am1() const
complex IntRank2A::A2m1() const
complex IntRank2A::Am1(double theta, double phi) const
complex IntRank2A::Am21(double theta, double phi) const
```

Description:

The functions ***Am1*** and ***A2m1*** are used to obtain the rank 2 interaction spatial tensor component $A_{2,1}$. If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments ***theta*** and ***phi*** are given the returned tensor component is for the orientation at ***theta*** degrees down from the interactions PAS z-axis and ***phi*** degrees over from the interactions PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***Hz***.

$$A_{2,-1}(\theta,\varphi) = -\sqrt{\frac{5}{24\pi}}\sin\theta[3\cos\theta - \eta(\cos\theta\cos 2\varphi + i\sin 2\varphi)] = -A_{2,1}{}^*(\theta,\varphi)$$

Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type[1]. This component can also be related to the Cartesian tensor components for any arbitrary orientation.

$$A_{2,-1} = \frac{1}{2}[A_{xz} + A_{zx} + i(A_{yz} - A_{zy})]$$

Return Value:

A complex number.

Example:

```
#include <gamma.h>
main()
  {
  IntRank2A R2( 0.2, 45.0, 45.0);        // Make a rank 2 interaction.
  complex vm1 = R2.A2m1();               // This is at theta=phi=45 degrees
  cout << R2.Am1(15.6, 99.3);            // This is at theta=15.6 and phi=99.3 degrees.
  }
```

See Also: A1, A21, Am1, A2m1, A2, A22, Am2, A2m2

1. Because the GAMMA platform accommodates different interaction types, the scaling on all spatial tensors is chosen to be independent of the interaction. Rather, the spatial tensors are related directly to the familiar rank two spherical harmonics $A_{2,m}(\theta,\varphi)\big|_{\eta=0} = Y_m^2(\theta,\varphi)$. Also, the sign on the term(s) involving η will have opposite sign if the common alternative definition of the PAS orientation ($|A_{zz}| \geq |A_{xx}| \geq |A_{yy}|$) is used rather that the definition used in GAMMA ($|A_{zz}| \geq |A_{yy}| \geq |A_{xx}|$)

2.9.4 A2, A22

Usage:

```
#include <IntRank2A.h>
complex IntRank2A::A2() const
complex IntRank2A::A22() const
complex IntRank2A::A2(double theta, double phi) const
complex IntRank2A::A22(double theta, double phi) const
```

Description:

The functions ***A2*** and ***A22*** are used to obtain the rank 2 interaction spatial tensor component $A_{2,1}$. If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments ***theta*** and ***phi*** are given the returned tensor component is for the orientation at ***theta*** degrees down from the interactions PAS z-axis and ***phi*** degrees over from the interactions PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***Hz***.

$$A_{2,\,2}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}\frac{1}{2}[3\sin^2\theta + \eta[\cos 2\varphi(1+\cos^2\theta) - i2\sin 2\varphi\cos\theta]]$$

Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type[1]. This component can also be related to the Cartesian tensor components for any arbitrary orientation.

$$A_{2,\,2} = \frac{1}{2}[A_{xx} - A_{yy} + i(A_{xy} + A_{yx})]$$

Return Value:

A complex number.

Example:

```
#include <gamma.h>
main()
  {
  IntRank2A R2(0.2, 45.0, 45.0);        // Make a rank 2 interaction.
  complex v22 = R2.A22();               // This is at theta=phi=45 degrees
  cout << R2.A2(15.6, 99.3);            // This is at theta=15.6 and phi=99.3 degrees.
  }
```

See Also: A1, A21, Am1, A2m1, A2, A22, Am2, A2m2

1. Because the GAMMA platform accommodates different interaction types, the scaling on all spatial tensors is chosen to be independent of the interaction. Rather, the spatial tensors are related directly to the familiar rank two spherical harmonics $A_{2,\,m}(\theta,\varphi)\big|_{\eta\,=\,0} = Y_m^2(\theta,\varphi)$. Also, the sign on the term(s) involving η will have opposite sign if the common alternative definition of the PAS orientation ($|A_{zz}| \geq |A_{xx}| \geq |A_{yy}|$) is used rather that the definition used in GAMMA ($|A_{zz}| \geq |A_{yy}| \geq |A_{xx}|$)

2.9.5 Am2, A2m2

Usage:

```
#include <IntRank2A.h>
complex IntRank2A::Am2() const
complex IntRank2A::A2m2() const
complex IntRank2A::Am2(double theta, double phi) const
complex IntRank2A::A2m2(double theta, double phi) const
```

Description:

The functions ***Am2*** and ***A2m2*** are used to obtain the rank 2 interaction spatial tensor component $A_{2,-2}$. If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments ***theta*** and ***phi*** are given the returned tensor component is for the orientation at ***theta*** degrees down from the interactions PAS z-axis and ***phi*** degrees over from the interactions PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***degrees***.

$$A_{2,-2}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}\frac{1}{2}[3\sin^2\theta + \eta[\cos 2\varphi(1+\cos^2\theta) + i2\sin 2\varphi\cos\theta]] = A_{2,2}{}^*(\theta,\varphi)$$

Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type[1]. This component can also be related to the Cartesian tensor components for any arbitrary orientation.

$$A_{2,-2} = \frac{1}{2}[A_{xx} + (-A_{yy}) - i(A_{xy} + A_{yx})]$$

Return Value:

A complex number.

Example:

```
#include <gamma.h>
main()
  {
  IntRank2A R2(0.2, 45.0, 45.0);        // Make a rank 2 interaction.
  complex v2m2 = R2.A2m2();             // This is at theta=phi=45 degrees
  cout << R2.Am2(15.6, 99.3);           // This is at theta=15.6 and phi=99.3 degrees.
  }
```

See Also: A1, A21, Am1, A2m1, A2, A22, Am2, A2m2

1. Because the GAMMA platform accommodates different interaction types, the scaling on all spatial tensors is chosen to be independent of the interaction. Rather, the spatial tensors are related directly to the familiar rank two spherical harmonics $A^Q_{2,m}(\theta,\varphi)\Big|_{\eta=0} = Y^2_m(\theta,\varphi)$. Also, the sign on the term(s) involving η will have opposite sign if the common alternative definition of the PAS orientation ($|A_{zz}| \geq |A_{xx}| \geq |A_{yy}|$) is used rather that the definition used in GAMMA ($|A_{zz}| \geq |A_{yy}| \geq |A_{xx}|$)

2.10 Cartesian Spatial Tensor Functions

2.10.1 Axx

Usage:

```
#include <IntRank2A.h>
complex IntRank2A::Axx() const
complex IntRank2A::Axx(double theta, double phi) const
```

Description:

The functions ***Axx*** is used to obtain the rank 2 interaction spatial tensor component A_{xx}. If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments ***theta*** and ***phi*** are given the returned tensor component is for the orientation at ***theta*** degrees down from the interactions PAS z-axis and ***phi*** degrees over from the interactions PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***Hz***.

$$A_{xx}(\theta, \varphi) = \sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]$$

Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type. This component can also be related to the spherical tensor components for any arbitrary orientation.

$$A_{xx} = \frac{1}{2}(A_{2,2} + A_{2,-2}) - \frac{1}{\sqrt{6}}A_{2,0}$$

Return Value:

A complex number.

Example:

```
#include <gamma.h>
main()
  {
  IntRank2A R2(0.2, 45.0, 45.0);        // Make a rank 2 interaction.
  complex vxx = R2.Axx();               // This is at theta=phi=45 degrees
  cout << R2.Axx(15.6, 99.3);           // This is at theta=15.6 and phi=99.3 degrees.
  }
```

See Also: Ayy, Azz, Axy, Axz, Ayx, Ayz, Azx, Azy

2.10.2 Ayy

Usage:

```
#include <IntRank2A.h>
complex IntRank2A::Ayy() const
complex IntRank2A::Ayy(double theta, double phi) const
```

Description:

The functions ***Ayy*** is used to obtain the rank 2 interaction spatial tensor component A_{yy}. If no arguments are given

the functions return the value of the tensor component at the current interaction orientation. If the arguments ***theta*** and ***phi*** are given the returned tensor component is for the orientation at ***theta*** degrees down from the interactions PAS z-axis and ***phi*** degrees over from the interactions PAS x-a!a

567s. The values of ***theta*** and ***phi*** are assumed in ***Hz***.

$$A_{yy}(\theta, \varphi) = -\sqrt{\frac{5}{24\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]$$

Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type. This component can also be related to the spherical tensor components for any arbitrary orientation.

$$A_{yy} = \frac{-1}{2}(A_{2,2} + A_{2,-2}) - \frac{1}{\sqrt{6}}A_{2,0}$$

Return Value:

A complex number.

Example:

```
#include <gamma.h>
main()
  {
  IntRank2A R2(10.2, 45.0, 45.0);        // Make a rank 2 interaction.
  complex vyy = R2.Ayy();                // This is at theta=phi=45 degrees
  cout << R2.Ayy(15.6, 99.3);            // This is at theta=15.6 and phi=99.3 degrees.
  }
```

See Also: Axx, Azz, Axy, Axz, Ayx, Ayz, Azx, Azy

2.10.3 Azz

Usage:

```
#include <IntRank2A.h>
complex IntRank2A::Azz() const
complex IntRank2A::Azz(double theta, double phi) const
```

Description:

The functions ***Azz*** is used to obtain the rank 2 interaction spatial tensor component A_{zz}. If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments ***theta*** and ***phi*** are given the returned tensor component is for the orientation at ***theta*** degrees down from the interactions PAS z-axis and ***phi*** degrees over from the interactions PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***Hz***.

$$A_{zz}(\theta, \varphi) = \sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]$$

Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type. This component can also be related to the spherical tensor components for any arbitrary orientation.

$$A_{zz} = \sqrt{\frac{2}{3}} A_{2,0}$$

Return Value:

A complex number.

Example:

```
#include <gamma.h>
main()
  {
  IntRank2A R2(0.2, 45.0, 45.0);        // Make a rank 2 interaction.
  complex vzz = R2.A20();               // This is at theta=phi=45 degrees
  cout << R2.A20(15.6, 99.3);           // This is at theta=15.6 and phi=99.3 degrees.
  }
```

See Also: Axx, Ayy, Axy, A2xz, Ayx, Ayz, Azx, Azy

2.10.4 Axy, Ayx

Usage:

```
#include <IntRank2A.h>
complex IntRank2A::Axy() const
complex IntRank2A::Axy(double theta, double phi) const
complex IntRank2A::Ayx() const
complex IntRank2A::Ayx(double theta, double phi) const
```

Description:

The functions ***Axy*** and ***Ayx*** are used to obtain the rank 2 interaction spatial tensor component $A_{xy} = A_{yx}$. If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments ***theta*** and ***phi*** are given the returned tensor component is for the orientation at ***theta*** degrees down from the interactions PAS z-axis and ***phi*** degrees over from the interactions PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***Hz***.

$$A_{xy}(\theta, \varphi) = \sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]$$

Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type. This component can also be related to the spherical tensor components for any arbitrary orientation.

$$A_{xy} = -\frac{i}{2}(A_{2,2} - A_{2,-2}) = A_{yx}$$

Return Value:

A complex number.

Example:

```
IntRank2A R2(0.2, 45.0, 45.0);        // Make a rank 2 interaction.
complex vxy = Q.Axy();                // This is at theta=phi=45 degrees
```

```
cout << R2.Ayx(15.6, 99.3);          // This is at theta=15.6 and phi=99.3 degrees.
```

See Also: Axx, Ayy, Azz, Axz, Ayz, Azx, Azy

2.10.5 Axz, Azx

Usage:

```
#include <IntRank2A.h>
complex IntRank2A::Axz() const
complex IntRank2A::Axz(double theta, double phi) const
complex IntRank2A::Azx() const
complex IntRank2A::Azx(double theta, double phi) const
```

Description:

The functions ***Axz*** and ***Azx*** are used to obtain the rank 2 interaction spatial tensor component $A_{xz} = A_{zx}$. If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments ***theta*** and ***phi*** are given the returned tensor component is for the orientation at ***theta*** degrees down from the interactions PAS z-axis and ***phi*** degrees over from the interactions PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***Hz***.

$$A_{xy}(\theta, \varphi) = \sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]$$

Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type. This component can also be related to the spherical tensor components for any arbitrary orientation.

$$A_{xz} = -\frac{1}{2}[(A_{2,1} - A_{2,-1})] = A_{zx}$$

Return Value:

A complex number.

Example:

```
#include <gamma.h>
main()
  {
  IntRank2A R2(0.2, 45.0, 45.0);     // Make a rank 2 interaction.
  complex vxz= R2.A20();             // This is at theta=phi=45 degrees
  cout << R2.Azx(15.6, 99.3);        // This is at theta=15.6 and phi=99.3 degrees.
  }
```

See Also: Axx, Ayy, Azz, Axz, Ayz, Azx, Azy

2.10.6 Ayz, Azy

Usage:

```
#include <IntRank2A.h>
complex IntRank2A::Ayz() const
complex IntRank2A::Ayz(double theta, double phi) const
complex IntRank2A::Azy() const
complex IntRank2A::Azy(double theta, double phi) const
```

Description:

The functions ***Ayz*** and ***Azy*** are used to obtain the rank 2 interaction spatial tensor component $A_{yz} = A_{zy}$. If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments ***theta*** and ***phi*** are given the returned tensor component is for the orientation at ***theta*** degrees down from the interactions PAS z-axis and ***phi*** degrees over from the interactions PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***Hz***.

$$A_{xy}(\theta, \varphi) = \sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]$$

Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type. This component can also be related to the spherical tensor components for any arbitrary orientation.

$$A_{yz} = \frac{i}{2}[(A_{2,1} + A_{2,-1})] = A_{zy}$$

Return Value:

A complex number.

Example:

```
#include <gamma.h>
main()
  {
  IntRank2A R2( 0.2, 45.0, 45.0);        // Make a rank 2 interaction.
  complex vyz = R2.Ayz();                // This is at theta=phi=45 degrees
  cout << R2.Azy(15.6, 99.3);            // This is at theta=15.6 and phi=99.3 degrees.
  }
```

See Also: Axx, Ayy, Azz, Axz, Ayz, Azx, Azy

2.11 Powder Average Facilitator Functions

2.11.1 A0A, A20A

Usage:

```
row_vector IntRank2A::A0A(int Ntheta)
row_vector IntRank2A::A20A(int Ntheta)
```

Description:

The functions ***A0A*** and ***A20A*** are equivalent. They are used to obtain part of rank 2 interaction spatial tensor component $A_{2,0}$ for a series of evenly incrmented θ values.

$$A_{2,0}A(\theta) = \sqrt{\frac{5}{16\pi}}(3\cos^2\theta - 1) = A_{2,0}(\theta, \varphi)\Big|_{\eta = \varphi = 0}$$

Given a number of angle increments, ***Ntheta***, a row vector of dimension ***Ntheta*** will be returned which contains the η independent terms of $A_{2,0}$ at evenly spaced increments of θ starting at the +z PAS ($\theta = 0$) alignment and finishing at -z PAS ($\theta = 180$) alignment.

$$\langle v|i\rangle = A_{2,0}A(\theta_i) \qquad \theta_i = \frac{180i}{(\mathrm{Ntheta} - 1)}$$

Note that to obtain the full $A_{2,0}$ terms (if they are η dependent) they must be properly combined with the values from the function ***A20B.***

Return Value:

A vector.

Example:

```
#include <gamma.h>
main()
  {
  IntRank2A Q(1.5, 3.e5, 0.2, 45.0, 45.0);  // Make a rank 2 interaction.
  row_vector A20s = Q.A20A(720);            // Get 720 A20A values spanning [0, 180]
  }
```

See Also: A21A, A22A, A20B, A21B, A22B, A2As, A2Bs, A2s

2.11.2 A1A, A21A

Usage:

```
row_vector IntRank2A::A1A(int Ntheta)
row_vector IntRank2A::A21A(int Ntheta)
```

Description:

The functions ***A1A*** and ***A21A*** are equivalent. They are used to obtain part of rank 2 interaction spatial tensor com-

ponent $A_{2,1}$ for a series of evenly incrmented θ values.

$$A_{2,1}A(\theta) = 3\sqrt{\frac{5}{24\pi}}\sin\theta\cos\theta = A_{2,1}(\theta,\varphi)\Big|_{\eta=0}$$

Given a number of angle increments, ***Ntheta***, a row vector of dimension ***Ntheta*** will be returned which contains the η independent terms of $A_{2,1}$ at evenly spaced increments of θ starting at the +z PAS ($\theta = 0$) alignment and finishing at -z PAS ($\theta = 180$) alignment.

$$\langle v|i\rangle = A_{2,1}A(\theta_i) \qquad \theta_i = \frac{180i}{(\mathrm{Ntheta}-1)}$$

Note that to obtain the full $A_{2,1}$ terms (if they are η dependent) they must be properly combined with the values from the function ***A21B.***

Return Value:

A vector.

Example:

```
#include <gamma.h>
main()
  {
  IntRank2A Q(1.5, 3.e5, 0.2);          // Make a rank 2 interaction.
  row_vector A21s = Q.A21A(181);        // Get 181 A20A values spanning [0, 180]
  }
```

See Also: A20A, A22A, A20B, A21B, A22B, A2As, A2Bs, A2s

2.11.3 A2A, A221A

Usage:

```
row_vector IntRank2A::A2A(int Ntheta)
row_vector IntRank2A::A22A(int Ntheta)
```

Description:

The functions ***A2A*** and ***A22A*** are equivalent. They are used to obtain part of rank 2 interaction spatial tensor component $A_{2,2}$ for a series of evenly incremented θ values.

$$A_{2,2}A(\theta) = \frac{3}{2}\sqrt{\frac{5}{24\pi}}\sin^2\theta = 3\sqrt{\frac{5}{96\pi}}\sin^2\theta = A_{2,2}(\theta,\varphi)\Big|_{\eta=0}$$

Given a number of angle increments, ***Ntheta***, a row vector of dimension ***Ntheta*** will be returned which contains the η independent terms of $A_{2,2}$ at evenly spaced increments of θ starting at the +z PAS ($\theta = 0$) alignment and finishing at -z PAS ($\theta = 180$) alignment.

$$\langle v|i\rangle = A_{2,2}A(\theta_i) \qquad \theta_i = \frac{180i}{(\text{Ntheta} - 1)}$$

Note that to obtain the full $A_{2,2}$ terms (if they are η dependent) they must be properly combined with the values from the function ***A22B.***

Return Value:

A vector.

Example:

```
#include <gamma.h>
main()
  {
  IntRank2A Q(1.5, 3.e5, 0.2);          // Make a rank 2 interaction.
  row_vector A22s = Q.A22A(181);        // Get 181 A22A values spanning [0, 180]
  }
```

See Also: A20A, A21A, A20B, A21B, A22B, A2As, A2Bs, A2s

2.11.4 A0B, A20B

Usage:

```
row_vector IntRank2A::A0B(int Nphi)
row_vector IntRank2A::A20B(int Nphi)
```

Description:

The functions ***A0B*** and ***A20B*** are equivalent. They are used to obtain part of rank 2 interaction spatial tensor component $A_{2,0}$ for a series of evenly incrmented φ values.

$$A_{2,0}B(\varphi) = \sqrt{\frac{5}{16\pi}}\eta\cos 2\varphi = \frac{1}{\sin^2\theta}[A_{2,0}(\theta,\varphi) - A_{2,0}(\theta,\varphi)|_{\eta=0}]$$

Given a number of angle increments, ***Nphi***, a row vector of dimension ***Nphi*** will be returned which contains θ independent terms of $A_{2,0}$ at evenly spaced increments of φ starting at the +x PAS ($\varphi = 0$) alignment and finishing at +x PAS ($\varphi = 360$) alignment.

$$\langle v|i\rangle = A_{2,0}A(\varphi_i) \qquad \varphi_i = \frac{360i}{\text{Nphi}}$$

Note that to obtain the full $A_{2,0}$ terms they must be properly combined with the values from the function ***A20A.***

$$A_{2,0}(\theta,\varphi) = \sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]$$

$$A_{2,1}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}\sin\theta[3\cos\theta - \eta(\cos\theta\cos 2\varphi - i\sin 2\varphi)] = -A_{2,-1}{}^*(\theta,\varphi)$$

$$A_{2,2}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}\frac{1}{2}[3\sin^2\theta + \eta[\cos 2\varphi(1+\cos^2\theta) - i2\sin 2\varphi\cos\theta]] = A_{2,-2}{}^*(\theta,\varphi)$$

Return Value:

A vector.

Example:

```
#include <gamma.h>
main()
  {
  IntRank2A Q(1.5, 3.e5, 0.2);          // Make a rank 2 interaction.
  row_vector A20s = Q.A20B(120);        // Get 120 A20B values spanning [0, 360)
  }
```

See Also: A20A, A21A, A22A, A21B, A22B, A2As, A2Bs, A2s

2.11.5 A1B, A21B

Usage:

```
row_vector IntRank2A::A1B(int Nphi)
row_vector IntRank2A::A21B(int Nphi)
```

Description:

The functions ***A1B*** and ***A21B*** are equivalent. They are used to obtain part of rank 2 interaction spatial tensor component $A_{2,1}$ for a series of evenly incrmented φ values.

$$A_{2,1}B(\varphi) = -\sqrt{\frac{5}{24\pi}}\eta(\cos 2\varphi - i\sin 2\varphi)$$

where

$$A_{2,1}(\theta,\varphi) = \sin\theta\cos\theta Re(A_{2,1}B(\varphi)) + i\sin\theta Im(A_{2,1}B(\varphi)) + A_{2,1}(\theta,\varphi)\Big|_{\eta=0}$$

Given a number of angle increments, ***Nphi***, a row vector of dimension ***Nphi*** will be returned which contains θ independent terms of $A_{2,1}$ at evenly spaced increments of φ starting at the +x PAS ($\varphi = 0$) alignment and finishing at +x PAS ($\varphi = 360$) alignment.

$$\langle v|i\rangle = A_{2,1}A(\varphi_i) \qquad \varphi_i = \frac{360i}{\text{Nphi}}$$

Note that to obtain the full $A_{2,1}$ terms they must be properly combined with the values from the function ***A21A.***

Return Value:

A vector.

Example:

```
#include <gamma.h>
main()
  {
  IntRank2A Q(1.5, 3.e5, 0.2);          // Make a rank 2 interaction.
  row_vector A21s = Q.A21B(120);        // Get 120 A21B values spanning [0, 360)
  }
```

See Also: A20A, A21A, A22A, A20B, A22B, A2As, A2Bs, A2s

2.11.6 A2B, A22B

Usage:

```
row_vector IntRank2A::A2B(int Nphi)
row_vector IntRank2A::A22B(int Nphi)
```

Description:

The functions ***A1B*** and ***A21B*** are equivalent. They are used to obtain part of rank 2 interaction spatial tensor component $A_{2,2}$ for a series of evenly incrmented φ values.

$$A_{2,2}B(\varphi) = \sqrt{\frac{5}{96\pi}}\eta[\cos 2\varphi - i2\sin 2\varphi]$$

where

$$A_{2,2}(\theta,\varphi) = (1+\cos^2\theta)Re(A_{2,2}B(\varphi)) + i\cos\theta Im(A_{2,2}B(\varphi)) + A_{2,2}(\theta,\varphi)\Big|_{\eta=0}$$

Given a number of angle increments, ***Nphi***, a row vector of dimension ***Nphi*** will be returned which contains θ independent terms of $A_{2,2}$ at evenly spaced increments of φ starting at the +x PAS ($\varphi = 0$) alignment and finishing at +x PAS ($\varphi = 360$) alignment.

$$\langle v|i\rangle = A_{2,2}A(\varphi_i) \qquad \varphi_i = \frac{360i}{\text{Nphi}}$$

Note that to obtain the full $A_{2,2}$ terms they must be properly combined with the values from the function ***A22A.***

Return Value:

A vector.

Example:

```
#include <gamma.h>
main()
  {
  IntRank2A Q(1.5, 3.e5, 0.2);          // Make a rank 2 interaction.
  row_vector A22s = Q.A22B(120);        // Get 120 A22B values spanning [0, 360)
  }
```

See Also: A20A, A21A, A22A, A20B, A21B, A2As, A2Bs, A2s

2.11.7 A2s

Usage:

matrix IntRank2A::A2s(int Ntheta, int Nphi)

Description:

The function ***A2s*** is used to construct the rank 2 interaction spatial tensor components $A_{2,\,m}$ for a series of evenly incrmented θ and φ values. Given arguments for the number of angle increments, ***Ntheta*** and ***Nphi*** the function will return a matrix of dimenstion (8 x nc) where nc is the larger of the two input arguments. The matrix columns, indexed by j, will then correspond either to an angle θ or an angle φ where

$$\theta_j = \frac{180j}{(\mathrm{Ntheta} - 1)} \qquad \varphi_j = \frac{360j}{\mathrm{Nphi}}$$

depending upon which row is being accessed. Rows 0-2 of the array will correspond to the the η independent terms of $A_{2,\,\{0,\,1,\,2\}}$ at evenly spaced increments of θ starting at the +z PAS ($\theta = 0$) alignment and finishing at -z PAS ($\theta = 180$) alignment. Rows 3-5 of the array will correspond to θ independent parts of the interaction spatial tensor components $A_{2,\,\{0,\,1,\,2\}}$ at evenly spaced increments of φ starting at the +x PAS ($\varphi = 0$) alignment and finishing at +x PAS ($\varphi = 360$) alignment. The final three array columns will contain θ dependent terms that are used to blend with the other rows to form the full $A_{2,\,m}(\theta, \varphi)$ values. Reconstruction of full $A_{2,\,m}(\theta, \varphi)$ values is based on

$$A_{2,\,0}(\theta, \varphi) = A_{2,\,0}(\theta, \varphi)\big|_{\eta = 0} + \sin^2\theta A_{2,\,0}B(\varphi)$$

$$A_{2,\,1}(\theta, \varphi) = A_{2,\,1}(\theta, \varphi)\big|_{\eta = 0} + \sin\theta\cos\theta Re(A_{2,\,1}B(\varphi)) + i\sin\theta Im(A_{2,\,1}B(\varphi))$$

$$A_{2,\,2}(\theta, \varphi) = A_{2,\,2}(\theta, \varphi)\big|_{\eta = 0} + (1 + \cos^2\theta)Re(A_{2,\,2}B(\varphi)) + i\cos\theta Im(A_{2,\,2}B(\varphi))$$

A particular $A_{2,\,m}(\theta_k, \varphi_l)$ can be reconstructed according to the analogous discrete equations.

$$A_{2,\,0}(\theta_k, \varphi_l) = \langle 0|mx|k\rangle + \langle 6|mx|k\rangle^2 \langle 3|mx|l\rangle$$

$$A_{2,\,1}(\theta_k, \varphi_l) = \langle 1|mx|k\rangle + \langle 6|mx|k\rangle[\langle 7|mx|k\rangle Re\langle 4|mx|l\rangle + iIm\langle 4|mx|l\rangle]$$

$$A_{2,\,2}(\theta_k, \varphi_l) = \langle 2|mx|k\rangle + (1 + \langle 7|mx|k\rangle^2) Re\langle 5|mx|l\rangle + i\langle 7|mx|k\rangle Im\langle 5|mx|l\rangle$$

The components with m negative are obtained from the relationship .

$$A_{2,\,-m} = (-1)^m A_{2,\,m}$$

Return Value:

An array.

Example:

```
#include <gamma.h>
main()
  {
  IntRank2A Q(1.5, 3.e5, 0.2);          // Make a rank 2 interaction.
  matrix As = Q.A2x(720, 360);          // Get array for values spanning [0, 180] & [0, 360)
  }
```

See Also: A20A, A21A, A22A, A20B, A21B, A2As, A2Bs, A2s

2.12 Asymmetry Functions

2.12.1 eta

Usage:

```
#include <IntRank2A.h>
double IntRank2A::eta() const;
void IntRank2A::eta(double Eta);
```

Description:

The function ***eta*** is used to either obtain or set the asymmetry of the rank 2 interaction. The asymmetry is defined (in terms of Cartesian components) as

$$\eta = \frac{A_{xx} - A_{yy}}{A_{zz}} \qquad |A_{zz}| \geq |A_{yy}| \geq |A_{xx}|$$

Return Value:

None.

Example:

```
#include <gamma.h>
main()
  {
IntRank2A Q(1.5, 3.e5, 0.2, 45.0, 45.0);    // Make a rank 2 interaction.
Q.setPAS();                                 // As if we used Q(1.5,3.e5,0.2,0,0)
  }
```

See Also: theta, phi, orient

2.13 Auxiliary Functions

2.13.1 setPAS

Usage:

```
#include <IntRank2A.h>
void IntRank2A::setPAS()
```

Description:

The functions ***setPAS*** is used to orient the rank 2 interaction into it's principal axis system. All 5 spatial tensor components will be set to PAS values and the internal orientation angles set to zero.

Return Value:

None.

Example:

```
#include <gamma.h>
main()
  {
  IntRank2A Q(1.5, 3.e5, 0.2, 45.0, 45.0);  // Make a rank 2 interaction.
  Q.setPAS();                               // As if we used Q(1.5,3.e5,0.2,0,0)
  }
```

See Also: theta, phi, orient

2.13.2 symmetric

Usage:

```
#include <IntRank2A.h>
int IntRank2A::symmetric() const
```

Description:

The functions ***symmetric*** is used to check if the rank 2 interaction has any asymmetry. The function will return true if the interaction is symmetric and false if there is some asymmetry (non-zero eta value).

Return Value:

An integer

Example:

```
#include <gamma.h>
main()
  {
  IntRank2A Q(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a rank 2 interaction.
  if(Q.symmetric()) cout << "Yep";        // We should get No for Q because eta=0.2)
  else                   << "Nope";
  }
```

See Also: eta

2.13.3 PAS

Usage:

int IntRank2A::PAS) const

Description:

The function ***PAS*** is used to check if the rank 2 interaction is oriented in its PAS or not. The function will return true if the interaction is PAS aligned and false if not).

Return Value:

An integer

Example:

```
#include <gamma.h>
main()
  {
  IntRank2A Q(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a rank 2 interaction.
  if(Q.PAS()) cout << "Yep";                    // We should get No for Q because neither θ or ϕ is 0)
  else                        << "Nope";
  }
```

See Also: eta

2.14 I/O Functions

2.14.1 read

Usage:

```
void IntRank2A::read(const String& filename, const spin_sys) const
void IntRank2A::read(const String& filename, const spin_sys) const
void IntRank2A::read(const String& filename, const spin_sys) const
void IntRank2A::read(const String& filename, const spin_sys) const
```

Description:

The function ***delzz*** is used to either obtain or set the interaction rank 2 coupling constant. With no arguments the function returns the coupling in Hz. If an argument, ***dz***, is specified then the coupling constant for the interaction is set. It is assumed that the input value of ***dz*** is in units of ***Hz***. The function is overloaded with the name ***delz*** for convenience. Note that setting of ***delzz*** will alter the (equivalent) value of the rank 2 coupling ***QCC***/***NQCC*** as well as the rank 2 frequency.

Return Value:

Either void or a floating point number, double precision.

Example(s):

```
#include <gamma.h>
main()
  {
  IntRank2A Q();                    // Empty rank 2 interaction.
  Q.delzz(100000.0);                // Set QCC to 100 KHz.
  cout << Q.delz ();                // Write coupling constant to std output.
  }
```

See Also: QCC, NQCC, wQ

2.14.2 ask

Usage:

```
#include <IntRank2A.h>
double IntRank2A:: () const
double IntRank2A::delz () const
double IntRank2A::delzz (double dz) const
double IntRank2A::delz (double dz) const
```

Description:

The function ***delzz*** is used to either obtain or set the interaction rank 2 coupling constant. With no arguments the function returns the coupling in Hz. If an argument, ***dz***, is specified then the coupling constant for the interaction is set. It is assumed that the input value of ***dz*** is in units of ***Hz***. The function is overloaded with the name ***delz*** for convenience. Note that setting of ***delzz*** will alter the (equivalent) value of the rank 2 coupling ***QCC***/***NQCC*** as well

as the rank 2 frequency.

Return Value:

Either void or a floating point number, double precision.

Example(s):

```
#include <gamma.h>
main()
  {
  IntRank2A Q();                    // Empty rank 2 interaction.
  Q.delzz(100000.0);                // Set QCC to 100 KHz.
  cout << Q.delz ();                // Write coupling constant to std output.
  }
```

See Also: QCC, NQCC, wQ

2.14.3 askset

Usage:

```
#include <IntRank2A.h>
double IntRank2A:: () const
double IntRank2A::delz () const
double IntRank2A::delzz (double dz) const
double IntRank2A::delz (double dz) const
```

Description:

The function ***delzz*** is used to either obtain or set the interaction rank 2 coupling constant. With no arguments the function returns the coupling in Hz. If an argument, ***dz***, is specified then the coupling constant for the interaction is set. It is assumed that the input value of ***dz*** is in units of ***Hz***. The function is overloaded with the name ***delz*** for convenience. Note that setting of ***delzz*** will alter the (equivalent) value of the rank 2 coupling ***QCC***/***NQCC*** as well as the rank 2 frequency.

Return Value:

Either void or a floating point number, double precision.

Example(s):

```
#include <gamma.h>
main()
  {
  #include <IntRank2A.h>
  IntRank2A Q();                    // Empty rank 2 interaction.
  Q.delzz(100000.0);                // Set QCC to 100 KHz.
  cout << Q.delz ();                // Write coupling constant to std output.
  }
```

See Also: QCC, NQCC, wQ

2.14.4 print

Usage:

```
#include <IntRank2A.h>
ostream& IntRank2A::print (ostream& ostr, int fflag=-1)
```

Description:

The function ***print*** is used to write the interaction rank 2 coupling constant to an output stream ***ostr***. An additional flag ***fflag*** is set to allow some control over how much information is output. The default (***fflag !=0***) prints all information concerning the interaction. If ***fflag*** is set to zero only the basis parameters are printed.

Return Value:

The ostream is returned.

Example:

```
#include <gamma.h>
main()
  {
  IntRank2A Q(2.5, 2.e6, 0.2, 45.7, 15.0);  // Make a rank 2 interaction.
  cout << Q;                                // Write the interaction to standard output.
  }
```

See Also: <<

2.14.5 <<

Usage:

```
#include <IntRank2A.h>
friend ostream& operator << (ostream& out, IntRank2A& Q)
```

Description:

The operator << defines standard output for the interaction rank 2 coupling constant.

Return Value:

The ostream is returned.

Example:

```
#include <gamma.h>
main()
  {
  #include <IntRank2A.h>
  IntRank2A Q(1.5, 3.e5, 0.2);              // Make a rank 2 interaction.
  cout << Q;                                // Write the interaction to standard output.
  }
```

See Also: print

2.14.6 printSpherical

Usage:

```
#include <IntRank2A.h>
ostream& IntRank2A::print (ostream& ostr, int fflag=-1)
```

Description:

The function ***print*** is used to write the interaction rank 2 coupling constant to an output stream ***ostr***. An additional flag ***fflag*** is set to allow some control over how much information is output. The default (***fflag !=0***) prints all information concerning the interaction. If ***fflag*** is set to zero only the basis parameters are printed.

Return Value:

The ostream is returned.

Example:

```
#include <gamma.h>
main()
  {
  IntRank2A Q(2.5, 2.e6, 0.2, 45.7, 15.0);  // Make a rank 2 interaction.
  cout << Q;                                // Write the interaction to standard output.
  }
```

See Also: <<

2.14.7 printCartesian

Usage:

```
#include <IntRank2A.h>
ostream& IntRank2A::print (ostream& ostr, int fflag=-1)
```

Description:

The function ***print*** is used to write the interaction rank 2 coupling constant to an output stream ***ostr***. An additional flag ***fflag*** is set to allow some control over how much information is output. The default (***fflag !=0***) prints all information concerning the interaction. If ***fflag*** is set to zero only the basis parameters are printed.

Return Value:

The ostream is returned.

Example:

```
#include <gamma.h>
main()
  {
  IntRank2A Q(2.5, 2.e6, 0.2, 45.7, 15.0);  // Make a rank 2 interaction.
  cout << Q;                                // Write the interaction to standard output.
  }
```

See Also: <<

2.15 Description

2.15.1 Overview

There are several interactions of rank 2 that are important in the treatment of magnetic resonance. Such interactions modify system energy levels depeding upon their orientation in 3-dimensional space. The concern herein are the irreducible rank 2 ***spatial tensor components*** of such interactions. The spatial tensor components are embodied in the class IntRank2A soas to be independent of the interaction type. Class IntRank2A then serves as a base class for specific rank 2 interaction types when combined with appropriate spin tensor components and an interaction strength.

2.15.2 Coordinate Systems

We will shortly concern ourselves with the mathematical representation of rank 2 interactions, in particular their description in terms of spatial and spin tensors. The spatial tensors will be cast in both Cartesian and spherical coordinates and we will switch between the two when convenient. The figure below relates the orientation angles theta and phi to the standard right handed coordinate system in all GAMMA treatments.

Cartesian and Spherical Coordinate Systems

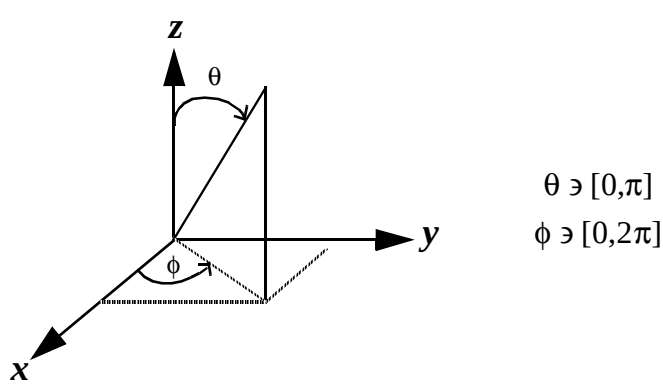


Figure 2-1 The right handed Cartesian axes with the spherical angles and radius.

2.15.3 Internal Structure

The internal structure of class ***IntRank2A*** contains the quantities listed in the following table (names shown are also internal).

Table 1: : Internal Structure of Class IsoRank2

Name	Description	Type	Name	Description	Type
ETA	Spatial Tensor η	double	THETA	Orientation Angle	double
PHI	Orientation Angle	double	Asph	Spatial Tensor Values	complex*

Two values (δ_{zz} & η) are required to specify the rank 2 interaction strength. In GAMMA the value of δ_{zz} is factored out of the spatial tensor such that all rank two interactions have the same scaling.

Thus the class contains only the value ***ETA*** to track η.

Two angles (θ & ϕ) are required to specify the interaction orentation. These are maintained in the values ***THETA*** and ***PHI*** indicate how the rank 2 interaction is aligned relative to the interaction principal axes (PAS). These are one in the same as the angles shown in Figure 2-1 when the Cartesian axes are those of the PAS with the origin vaguely being the center of the nucleus. These are intrinsically tied into the values in the array ***Asph***.

There are five values in the complex vector ***Asph*** and these are irreducible spherical components of the rank 2 spatial tensor oriented at angle ***THETA*** down from the PAS z-axis and over angle ***PHI*** from the PAS x-axis. Note that these 5 values are not only orientation dependent, they are also ***ETA*** dependent. If either of the three the interaction values {***ETA***, ***THETA, PHI***} are altered these components will all be reconstructed. The values in ***Asph*** will be scaled such that they are consistent with other rank 2 spatial tensors in GAMMA which are independent of the interaction type.

Structure of a Variable of Class IntRank2A

doubles

DELZZ	*THETA*
ETA	*PHI*

Figure 2-2 **Depiction of class IntRank2A contents, i.e. what each GAMMA defined rank 2 interaction contains. The values of both Xi and DELZZ are maintained for convenience (one being deduced from the other via I). Tsph will contain 5 matrices which dimension will be 2*I+1 and Asph will contain 5 complex numbers.**

The vector of complex numbers relate to the spherical spatial tensor components via

Asph:	[0]	[1]	[2]	[3]	[4]
$A_{2,m}$:	$A_{2,0}$	$A_{2,1}$	$A_{2,-1}$	$A_{2,2}$	$A_{2,-2}$

2.15.4 Cartesian Spatial Tensors

We begin by examining the 3x3 array which represents a general (reducible) rank 2 Cartesian tensor. The form on the left (below) is a generic representation whereas the tensor is written as a sum over tensors of ranks 0 - 2 on the right.

$$\hat{A} = \begin{bmatrix} A_{xx} & A_{xy} & A_{xz} \\ A_{yx} & A_{yy} & A_{yz} \\ A_{zx} & A_{zy} & A_{zz} \end{bmatrix} = \underbrace{A_{iso}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Rank 0}} + \underbrace{\begin{bmatrix} 0 & \alpha_{xy} & \alpha_{xz} \\ -\alpha_{xy} & 0 & \alpha_{yz} \\ -\alpha_{xz} & -\alpha_{yz} & 0 \end{bmatrix}}_{\text{Rank 1}} + \underbrace{\begin{bmatrix} \delta_{xx} & \delta_{xy} & \delta_{xz} \\ \delta_{yx} & \delta_{yy} & \delta_{yz} \\ \delta_{zx} & \delta_{zy} & \delta_{zz} \end{bmatrix}}_{\text{Rank 2}}$$

where

$$A_{iso} = \frac{1}{3}Tr\{\hat{A}\} \qquad \alpha_{xy} = \frac{1}{2}(A_{xy} - A_{yx}) \qquad \delta_{xy} = \frac{1}{2}(A_{xy} + A_{yx} - 2A_{iso})$$

If the rank 2 spatial tensor is symmetric, the anti-symmetric rank 1 terms are zero. If the rank 2 spatial tensor is traceless then the isotropic rank 1 term will be zero. If both are true (symmetric and traceless) then the spatial tensor is equivalent to the irreducible rank 2 tensor.

$$\hat{A} = \left.\begin{bmatrix} A_{xx} & A_{xy} & A_{xz} \\ A_{yx} & A_{yy} & A_{yz} \\ A_{zx} & A_{zy} & A_{zz} \end{bmatrix}\right|_{\substack{traceless \\ symmetric}} = \begin{bmatrix} \delta_{xx} & \delta_{xy} & \delta_{xz} \\ \delta_{yx} & \delta_{yy} & \delta_{yz} \\ \delta_{zx} & \delta_{zy} & \delta_{zz} \end{bmatrix} = \hat{A}_2$$

2.15.5 Rank 2 Spatial Tensor PAS Components

Any rank 2 spatial tensor can be specified in its principal axis system, the set of axes in which the irreducible rank 2 component is diagonal[1].

$$\hat{A}_2(PAS) = \begin{bmatrix} \delta_{xx} & 0 & 0 \\ 0 & \delta_{yy} & 0 \\ 0 & 0 & \delta_{zz} \end{bmatrix}$$

Symmetric rank 2 spatial tensors are commonly specified in their principal axis system by the three components: the isotropic value A_{iso}, the anisotropy ΔA, and the asymmetry η. These are given by

1. The rank 2 principal axis system is set such that $|\delta_{zz}| \geq |\delta_{yy}| \geq |\delta_{xx}|$. The orientation of the x and y axes are inconsequential if η is zero.

$$A_{iso} = \frac{1}{3}Tr\{A\}, \qquad \Delta A = A_{zz} - \frac{1}{2}(A_{xx} + A_{yy}) = \frac{3}{2}\delta_{zz} \qquad \eta = (\delta_{xx} - \delta_{yy})/\delta_{zz}$$

and we can then write

$$\hat{A}_2(PAS) = \delta_{zz}\begin{bmatrix} -\frac{1}{2}(1-\eta) & 0 & 0 \\ 0 & -\frac{1}{2}(1+\eta) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \delta_{xx} & 0 & 0 \\ 0 & \delta_{yy} & 0 \\ 0 & 0 & \delta_{zz} \end{bmatrix} \tag{0-5}$$

In addition, a set of Euler angles $\{\alpha, \beta, \gamma\}$ is used to relate the interaction at an arbitrary orientation to the spatial tensor principle axes.

2.15.6 Rank 2 Spherical Spatial Tensor General Components

The 9 irreducible spherical components of a rank 2 spatial tensor, $A_{l,\,m}$, are related to the Cartesian spatial components by the following general formulas[1].

$$\begin{gathered} A_{0,\,0} = \frac{-1}{\sqrt{3}}[A_{xx} + A_{yy} + A_{zz}] = \frac{-1}{\sqrt{3}}Tr\{A\} \\ A_{1,\,0} = \frac{-i}{\sqrt{2}}[A_{xy} - A_{yx}] \qquad A_{1,\,\pm1} = \frac{-1}{2}[A_{zx} - A_{xz} \pm i(A_{zy} - A_{yz})] \\ A_{2,\,0} = \sqrt{6}[3A_{zz} - (A_{xx} + A_{yy} + A_{zz})] = \sqrt{6}[3A_{zz} - Tr\{A\}] \\ A_{2,\,\pm1} = \mp\frac{1}{2}[A_{xz} + A_{zx} \pm i(A_{yz} + A_{zy})] \qquad A_{2,\,\pm2} = \frac{1}{2}[A_{xx} - A_{yy} \pm i(A_{xy} + A_{yx})] \end{gathered} \tag{0-6}$$

The subscript l spans the rank as l = [0, 2], and the subscript m = [-l, l]. The irreducible rank 2 spherical elements of the tensor, $A_{2,\,m}$, in the principal axis system are obtained by placement of (0-5) into (0-6).

$$A_{2,\,0}(PAS) = \sqrt{3/2}\delta_{zz} \qquad A_{2,\,1}(PAS) = 0 \qquad A_{2,\,2}(PAS) = A_{2,\,-2}(PAS) = \frac{1}{2}\delta_{zz}\eta$$

These will be the only terms if the tensor is symmetric and has no isotropic component (e.g. a dipolar interaction). Most interactions ignore the antisymmetric terms (l=1) so at most the above terms with $A_{0,\,0}$ suffice to describe everything. Herein we are concerned only with the irreducible rank 2 terms.

Unscaled Irreducible Speherical Rank 2 Spatial Tensor PAS Components

$$\boxed{A_{2,\,0}(PAS) = \sqrt{3/2}\delta_{zz} \qquad A_{2,\,\pm1}(PAS) = 0 \qquad A_{2,\,\pm2}(PAS) = \frac{1}{2}\delta_{zz}\eta}$$

1. See GAMMA Class Documentation on Spatial Tensors.

2.15.7 Oriented Irreducible Rank 2 Spherical Spatial Tensor Components

We can express the irreducible rank 2 spatial tensor components $A_{2,m}$ relative to any arbitrary axis system (AAS) by a rotation from the principal axes to the new axes *via* the formula[1]

$$A_{2,m}(AAS) = \sum_{m'}^{\pm 2} D^2_{m'm}(\Omega) A_{2,m'}(PAS) \tag{0-7}$$

where $D^2_{m'm}(\Omega)$ are the rank 2 Wigner rotation matrix elements and Ω the set of three Euler angles which relate the principal axes of the rank 2 spatial tensor to the arbitrary axes. For the treatment liquid isotropic systems these Euler angles are time dependent and averaged. For the treatment of solids they may be static (powder) or time dependent (MAS). Often only two angles suffice to orient the interaction relative to its PAS.
We can expand the components of the oriented spatial tensor in terms of the Wigner rotation elements as well as the reduced Wigner rotation elements $d^2_{mm'}$ which are given by

$$\boldsymbol{D}^2_{m,n}(\alpha, \beta, \gamma) = e^{-i\alpha m} d^2_{m,n}(\beta) e^{-i\gamma n} \tag{0-8}$$

The reduced (rank 2) Wigner rotation matrix elements supplied by this function are given in the following figure[2].

Reduced Rank 2 Wigner Rotation Matrix Elements

$d^2_{m,n}(\beta)$

m \ n	2	1	0	-1	-2
2	$\cos^4(\beta/2)$	$-\frac{1}{2}(1+\cos\beta)\sin\beta$	$\sqrt{3/8}\sin^2\beta$	$\frac{1}{2}(\cos\beta-1)\sin\beta$	$\sin^4(\beta/2)$
1	$\frac{1}{2}\sin\beta(\cos\beta+1)$	$\cos^2\beta-\frac{1}{2}(1-\cos\beta)$	$-\sqrt{3/2}\sin\beta\cos\beta$	$\frac{1}{2}(1+\cos\beta)-\cos^2\beta$	$\frac{1}{2}\sin\beta(\cos\beta-1)$
0	$\sqrt{3/8}\sin^2\beta$	$\sqrt{3/8}\sin(2\beta)$	$\frac{1}{2}(3\cos^2\beta-1)$	$-\sqrt{3/8}\sin(2\beta)$	$\sqrt{3/8}\sin^2\beta$
-1	$-\frac{1}{2}(\cos\beta-1)\sin\beta$	$\frac{1}{2}(1+\cos\beta)-\cos^2\beta$	$\sqrt{3/2}\sin\beta\cos\beta$	$\cos^2\beta-\frac{1}{2}(1-\cos\beta)$	$-\frac{1}{2}(1+\cos\beta)\sin\beta$
-2	$\sin^4(\beta/2)$	$-\frac{1}{2}(\cos\beta-1)\sin\beta$	$\sqrt{3/8}\sin^2\beta$	$\frac{1}{2}(1+\cos\beta)\sin\beta$	$\cos^4(\beta/2)$

Figure 2-3 **The reduced Wigner rotation matrix elements of rank 2. The angle β is the standard**

1. This rotation takes the PAS into the oriented coordinate axes.
2. See Brink and Satchler, page 24, TABLE 1.

Euler angle which is equal to the spherical phi angle.

Other useful relationships concerning these elements are

$$d^2_{m,\,n}(\beta) \;=\; (-1)^{m-n} d^2_{n,\,m}(\beta) \;=\; (-1)^{m-n} d^2_{-n,\,-m}(\beta) \tag{0-9}$$

Putting these into the formula for rotating the spatial tensor

$$\begin{aligned}
A_{2,\,m}(\theta,\varphi) \;&=\; \sum_{m'}^{\pm 2} D^2_{m'm}(0,\theta,\varphi) A_{2,\,m'}(PAS) \\
A_{2,\,m}(\theta,\varphi) \;&=\; A_{2,\,0}(PAS) D^2_{0m}(0,\theta,\varphi) + A_{2,\,2}(PAS)[D_{2m}(0,\theta,\varphi) + D_{-2m}(0,\theta,\varphi)] \\
A_{2,\,m}(\theta,\varphi) \;&=\; \sqrt{\frac{3}{2}}\delta_{zz} D^2_{0m}(0,\theta,\varphi) + \frac{1}{2}\delta_{zz}\eta[D_{2m}(0,\theta,\varphi) + D_{-2m}(0,\theta,\varphi)] \\
A_{2,\,m}(\theta,\varphi) \;&=\; \sqrt{\frac{3}{2}}\delta_{zz} d^2_{0m}(\theta) + \frac{1}{2}\delta_{zz}\eta[e^{-i2\varphi} d^2_{2m}(\theta) + e^{i2\varphi} d^2_{-2m}(\theta)]
\end{aligned} \tag{0-10}$$

Of the five components, two may be generated by symmetry. The remaining three are listed in the next equation.

$$\begin{aligned}
A_{2,\,0}(\theta,\varphi) \;&=\; \sqrt{\frac{3}{2}}\delta_{zz} d^2_{00}(\theta) + \frac{1}{2}\delta_{zz}\eta[e^{-i2\varphi} d^2_{20}(\theta) + e^{i2\varphi} d^2_{-20}(\theta)] \\
A_{2,\,1}(\theta,\varphi) \;&=\; \sqrt{\frac{3}{2}}\delta_{zz} d^2_{01}(\theta) + \frac{1}{2}\delta_{zz}\eta[e^{-i2\varphi} d^2_{21}(\theta) + e^{i2\varphi} d^2_{-21}(\theta)] \;=\; -A_{2,\,-1}{}^*(\theta,\varphi) \\
A_{2,\,2}(\theta,\varphi) \;&=\; \sqrt{\frac{3}{2}}\delta_{zz} d_{02}(\theta) + \frac{1}{2}\delta_{zz}\eta[e^{-i2\varphi} d_{22}(\theta) + e^{i2\varphi} d^2_{-22}(\theta)] \;=\; A_{2,\,-2}{}^*(\theta,\varphi)
\end{aligned} \tag{0-11}$$

For the m=0 component we can use the relationship $d^2_{-20}(\theta) \;=\; d^2_{20}(\theta)$

$$\begin{aligned}
A_{2,\,0}(\theta,\varphi) \;&=\; \sqrt{\frac{3}{2}}\delta_{zz} d^2_{00}(\theta) + \frac{1}{2}\delta_{zz}\eta[e^{-i2\varphi} d^2_{20}(\theta) + e^{i2\varphi} d^2_{-20}(\theta)] \\
&=\; \sqrt{\frac{3}{2}}\delta_{zz} d^2_{00}(\theta) + \frac{1}{2}\delta_{zz}\eta d^2_{20}(\theta)[e^{-i2\varphi} + e^{i2\varphi}] \\
&=\; \sqrt{\frac{3}{2}}\delta_{zz} d^2_{00}(\theta) + \frac{1}{2}\delta_{zz}\eta d^2_{20}(\theta) 2\cos 2\varphi \\
&=\; \sqrt{\frac{3}{2}}\delta_{zz}\left[\frac{1}{2}(3\cos^2\theta - 1) + \sqrt{\frac{2}{3}}\eta[\sqrt{3/8}\sin^2\theta]\cos 2\varphi\right] \\
&=\; \sqrt{\frac{3}{2}}\delta_{zz}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]
\end{aligned} \tag{0-12}$$

For the m=1 component

$$
\begin{aligned}
A_{2,1}(\theta,\varphi) &= \sqrt{\frac{3}{2}}\delta_{zz}d^2_{01}(\theta) + \frac{1}{2}\delta_{zz}\eta[e^{-i2\varphi}d^2_{21}(\theta) + e^{i2\varphi}d^2_{-21}(\theta)] \\
&= \sqrt{\frac{3}{2}}\delta_{zz}[\sqrt{3/2}\sin\theta\cos\theta] + \frac{1}{2}\delta_{zz}\eta\left[e^{-i2\varphi}\left[-\frac{1}{2}(1+\cos\theta)\sin\theta\right] + e^{i2\varphi}\left[\frac{1}{2}(1-\cos\theta)\sin\theta\right]\right] \\
&= \sqrt{\frac{3}{2}}\delta_{zz}\left[\sqrt{3/2}\sin\theta\cos\theta + \sqrt{\frac{1}{24}}\eta\sin\theta[e^{-i2\varphi}(-1-\cos\theta) + e^{i2\varphi}(1-\cos\theta)]\right] \\
&= \sqrt{\frac{3}{2}}\delta_{zz}\left[\sqrt{3/2}\sin\theta\cos\theta + \sqrt{\frac{1}{24}}\eta\sin\theta(2i\sin 2\varphi - 2\cos\theta\cos 2\varphi)\right] \\
&= \sqrt{\frac{3}{2}}\delta_{zz}\sin\theta[3\cos\theta - \eta(\cos\theta\cos 2\varphi - i\sin 2\varphi)]
\end{aligned}
\tag{0-13}
$$

For the m=2 component

$$
\begin{aligned}
A_{2,2}(\theta,\varphi) &= \sqrt{\frac{3}{2}}\delta_{zz}d_{02}(\theta) + \frac{1}{2}\delta_{zz}\eta[e^{-i2\varphi}d_{22}(\theta) + e^{i2\varphi}d^2_{-22}(\theta)] \\
&= \sqrt{\frac{3}{2}}\delta_{zz}[\sqrt{3/8}\sin^2\theta] + \frac{1}{2}\delta_{zz}\eta\left[e^{-i2\varphi}\left[\frac{1}{4}(1+\cos\theta)^2\right] + e^{i2\varphi}\left[\frac{1}{4}(1-\cos\theta)^2\right]\right] \\
&= \sqrt{\frac{3}{2}}\delta_{zz}\left[\sqrt{3/8}\sin^2\theta + \sqrt{\frac{1}{6}}\frac{\eta}{4}[e^{-i2\varphi}(1+\cos\theta)^2 + e^{i2\varphi}(1-\cos\theta)^2]\right] \\
&= \sqrt{\frac{3}{2}}\delta_{zz}\left[\sqrt{3/8}\sin^2\theta + \sqrt{\frac{1}{6}}\frac{\eta}{4}[e^{-i2\varphi}(1+2\cos\theta+\cos^2\theta) + e^{i2\varphi}(1-2\cos\theta+\cos^2\theta)]\right] \\
&= \sqrt{\frac{3}{2}}\delta_{zz}\left[\sqrt{3/8}\sin^2\theta + \sqrt{\frac{1}{6}}\frac{\eta}{4}[2\cos 2\varphi(1+\cos^2\theta) - i4\sin 2\varphi\cos\theta]\right] \\
&= \sqrt{\frac{1}{16}}\delta_{zz}[3\sin^2\theta + \eta[\cos 2\varphi(1+\cos^2\theta) - i2\sin 2\varphi\cos\theta]]
\end{aligned}
\tag{0-14}
$$

To summarize the results of these rotations:

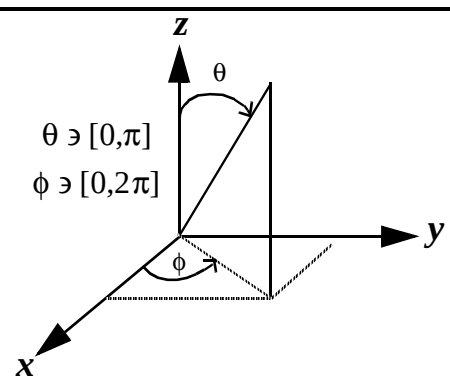
Unscaled Oriented Irreducible Rank 2 Spatial Tensor Components

$$A_{2,0}(PAS) = \sqrt{3/2}\delta_{zz}$$

$$A_{2,\pm1}(PAS) = 0 \qquad A_{2,\pm2}(PAS) = \frac{1}{2}\delta_{zz}\eta$$

$$\theta \ni [0,\pi]$$

$$\phi \ni [0,2\pi]$$



$$A_{2,0}(\theta,\varphi) = \sqrt{\frac{3}{2}}\delta_{zz}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]$$

$$A_{2,1}(\theta,\varphi) = \sqrt{\frac{3}{2}}\delta_{zz}\sin\theta[3\cos\theta - \eta(\cos\theta\cos 2\varphi - i\sin 2\varphi)] = -A_{2,-1}{}^*(\theta,\varphi)$$

$$A_{2,2}(\theta,\varphi) = \sqrt{\frac{1}{16}}\delta_{zz}[3\sin^2\theta + \eta[\cos 2\varphi(1+\cos^2\theta) - i2\sin 2\varphi\cos\theta]] = A_{2,-2}{}^*(\theta,\varphi)$$

Figure 2-4 **Equations relevant to the Rank 2 Hamiltonian spatial spherical tensor components when oriented at angles θ & φ from the PAS.**

Since scaling is arbitrary on these tensors (i.e. the value of δ_{zz} does not affect the tensor mathematics), we have choosen a standard scaling based on normalized speherical harmonics. In GAMMA we set

$$A_{2,\pm m}(AAS)\big|_{\eta=0} = Y_{2,\pm m}(\theta,\phi)$$

Comparison with the spatial tensors defined in GAMMA

$$Y_{2,0}(\theta,\phi) = \sqrt{\frac{5}{16\pi}}(3\cos^2\theta - 1) \qquad Y_{2,\pm 1}(\theta,\phi) = \mp\sqrt{\frac{15}{8\pi}}\cos\theta\sin\theta e^{\pm i\phi}$$

$$Y_{2,\pm 2}(\theta,\phi) = \sqrt{\frac{15}{32\pi}}\sin^2\theta e^{\pm 2i\phi}$$

It is evident that

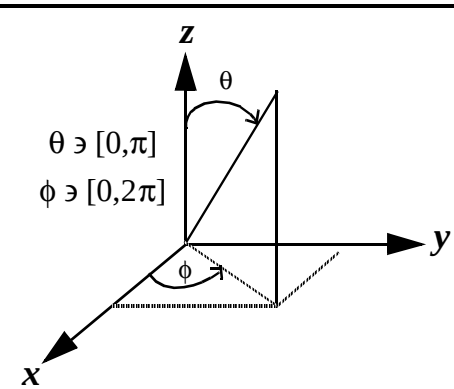
$$\delta_{zz} = \sqrt{2/3}\cdot\sqrt{\frac{5}{4\pi}} = \sqrt{\frac{5}{6\pi}}$$

Substitution into the previous table of equations produces the spatial tensor components that will be used in GAMMA.

GAMMA Scaled Oriented Rank 2 Spatial Tensor Components

$$A_{2,0}(PAS) = \sqrt{\frac{5}{4\pi}}$$

$$A_{2,\pm 1}(PAS) = 0 \qquad A_{2,\pm 2}(PAS) = \sqrt{\frac{5}{24\pi}}\eta$$



$$A_{2,0}(\theta,\varphi) = \sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]$$

$$A_{2,1}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}\sin\theta[3\cos\theta - \eta(\cos\theta\cos 2\varphi - i\sin 2\varphi)] = -A_{2,-1}{}^*(\theta,\varphi)$$

$$A_{2,2}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}\frac{1}{2}[3\sin^2\theta + \eta[\cos 2\varphi(1+\cos^2\theta) - i2\sin 2\varphi\cos\theta]] = A_{2,-2}{}^*(\theta,\varphi)$$

Figure 2-5 **Equations relevant to the Rank 2 Hamiltonian spatial spherical tensor components when oriented at angles θ & ϕ from the PAS[1].**

1. The scaling on both A_{2m} is arbitrary, so GAMMA uses a scaling which independent of whatever the tensor is applied to (the interaction type). The $\{A_{2m}\}$ are scaled so that rotations by angles θ & ϕ produce spherical harmonics for a symmetric interaction (η = 0). For specific treatments, a scaling factor (interaction constant) will likely be required.

2.15.8 Scaled Rank 2 Cartesian Spatial Tensor Components

For the sake of completeness we shall rewrite the Cartesian tensor components for the oriented rank 2 spatial tensor. These are used in the literature and may form components of equations found therein. We begin with the generalized relations between the Cartesian and irreducible spherical rank 2 tensor components[1].

$$A_{xx} = \frac{1}{2}(A_{2,2} + A_{2,-2}) - \frac{1}{\sqrt{6}}A_{2,0} \qquad A_{xy} = A_{yx} \qquad A_{xz} = A_{zx}$$

$$A_{yx} = -\frac{i}{2}(A_{2,2} - A_{2,-2}) \qquad A_{yy} = \frac{-1}{2}(A_{2,2} + A_{2,-2}) - \frac{1}{\sqrt{6}}A_{2,0} \qquad A_{yz} = A_{zy} \tag{0-15}$$

$$A_{zx} = -\frac{1}{2}[(A_{2,1} - A_{2,-1})] \qquad A_{zy} = \frac{i}{2}[(A_{2,1} + A_{2,-1})] \qquad A_{zz} = \sqrt{\frac{2}{3}}A_{2,0}$$

Generation of the Cartesian components thus involves the substitution of the previous formulae for the oriented spherical components into the above equations.

$$\begin{aligned}
A_{xx} &= \frac{1}{2}(A_{2,2} + A_{2,-2}) - \frac{1}{\sqrt{6}}A_{2,0} = \frac{1}{2}(A_{2,2} + A_{2,2}^*) - \frac{1}{\sqrt{6}}A_{2,0} = Re(A_{2,2}) - \frac{1}{\sqrt{6}}A_{2,0} \\
&= \sqrt{\frac{5}{24\pi}}\frac{1}{2}[3\sin^2\theta + \eta\cos 2\varphi(1 + \cos^2\theta)] - \frac{1}{\sqrt{6}}\sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right] \\
&= \sqrt{\frac{5}{24\pi}}\frac{1}{2}[3\sin^2\theta + \eta\cos 2\varphi(1 + \cos^2\theta) - (3\cos^2\theta - 1) - \eta\sin^2\theta\cos 2\varphi] \\
&= \sqrt{\frac{5}{24\pi}}\frac{1}{2}[3\sin^2\theta - 3\cos^2\theta + 1 + \eta\cos 2\varphi(1 + \cos^2\theta - \sin^2\theta)] \\
&= \sqrt{\frac{5}{24\pi}}\frac{1}{2}[6\sin^2\theta - 2 + 2\eta\cos 2\varphi\cos^2\theta] = \sqrt{\frac{5}{24\pi}}[3\sin^2\theta - 1 + \eta\cos 2\varphi\cos^2\theta] \\
A_{yy} &= \frac{-1}{2}(A_{2,2} + A_{2,-2}) - \frac{1}{\sqrt{6}}A_{2,0} = \frac{-1}{2}(A_{2,2} + A_{2,2}^*) - \frac{1}{\sqrt{6}}A_{2,0} = Re(A_{2,2}) - \frac{1}{\sqrt{6}}A_{2,0} \\
&= -\sqrt{\frac{5}{24\pi}}\frac{1}{2}[3\sin^2\theta + \eta\cos 2\varphi(1 + \cos^2\theta)] - \frac{1}{\sqrt{6}}\sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right] \\
&= -\sqrt{\frac{5}{24\pi}}\frac{1}{2}[3\sin^2\theta + \eta\cos 2\varphi(1 + \cos^2\theta) + (3\cos^2\theta - 1) + \eta\sin^2\theta\cos 2\varphi] \\
&= -\sqrt{\frac{5}{24\pi}}\frac{1}{2}[3\sin^2\theta + 3\cos^2\theta - 1 + \eta\cos 2\varphi(1 + \cos^2\theta + \sin^2\theta)] = -\sqrt{\frac{5}{24\pi}}\frac{1}{2}[2 + 2\eta\cos 2\varphi] = -\sqrt{\frac{5}{24\pi}}[1 + \eta\cos 2\varphi] \\
A_{zz} &= \sqrt{\frac{2}{3}}A_{2,0} = \sqrt{\frac{2}{3}}\sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right] = \sqrt{\frac{5}{24\pi}}[3\cos^2\theta - 1 + \eta\sin^2\theta\cos 2\varphi] \\
A_{xy} &= -\frac{i}{2}(A_{2,2} - A_{2,-2}) = -\frac{i}{2}(A_{2,2} - A_{2,2}^*) = Im(A_{2,2}) = -\sqrt{\frac{5}{24\pi}}\eta\sin 2\varphi\cos\theta = A_{yx} \\
A_{xz} &= -\frac{1}{2}[(A_{2,1} - A_{2,-1})] = -\frac{1}{2}(A_{2,1} + A_{2,1}^*) = -Re(A_{2,1}) = -\sqrt{\frac{5}{24\pi}}\sin\theta\cos\theta[3 - \eta\cos 2\varphi] = A_{zx} \\
A_{yz} &= \frac{i}{2}(A_{2,1} + A_{2,-1}) = \frac{i}{2}(A_{2,1} - A_{2,1}^*) = -Im(A_{2,1}) = -\sqrt{\frac{5}{24\pi}}\sin\theta[-\eta(-\sin 2\varphi)] = -\sqrt{\frac{5}{24\pi}}\eta\sin\theta\sin 2\varphi = A_{zy}
\end{aligned}$$

1. See the GAMMA documentation on spatial tensors, class space_T.

GAMMA Scaled Oriented Rank 2 Cartesian Spatial Tensor Components

$$A_{xx}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}[3\sin^2\theta - 1 + \eta\cos 2\varphi\cos^2\theta]$$

$$A_{yy}(\theta,\varphi) = -\sqrt{\frac{5}{24\pi}}[1 + \eta\cos 2\varphi]$$

$$A_{zz}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}[3\cos^2\theta - 1 + \eta\sin^2\theta\cos 2\varphi]$$

$$A_{xz}(\theta,\varphi) = -\sqrt{\frac{5}{24\pi}}\sin\theta\cos\theta[3 - \eta\cos 2\varphi] = A_{zx}$$

$$A_{xy}(\theta,\varphi) = -\sqrt{\frac{5}{24\pi}}\eta\sin 2\varphi\cos\theta = A_{yx} \qquad A_{yz}(\theta,\varphi) = -\sqrt{\frac{5}{24\pi}}\eta\sin\theta\sin 2\varphi = A_{zy}$$

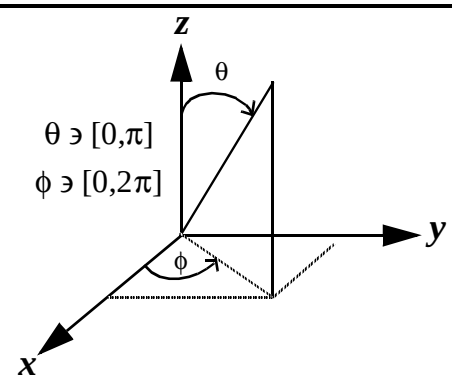


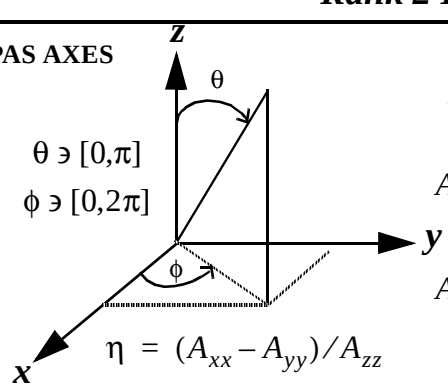
Figure 2-6 **Equations relevant to the Rank 2 spatial Cartesian tensor components when oriented at angles θ & ϕ from the PAS.**

2.15.9 Rank 2 PAS Equations

When the rank 2 interaction has alignment along its principal axes system virtually all of the rank 2 equations simplify. The following figure collects all of these for convenience.

Rank 2 Equations Involving the PAS

PAS AXES



$$A_{2,0}(PAS) = \sqrt{6}[3A_{zz} - Tr\{A\}]\big|_{PAS} = \sqrt{\frac{5}{4\pi}}$$

$$A_{2,\pm1}(PAS) = \mp\frac{1}{2}[A_{xz} + A_{zx} \pm i(A_{yz} + A_{zy})]\Big|_{PAS} = 0$$

$$A_{2,\pm2}(PAS) = \frac{1}{2}[A_{xx} - A_{yy} \pm i(A_{xy} + A_{yx})]\big|_{PAS} = \sqrt{\frac{5}{24\pi}}\eta$$

$$\eta = (A_{xx} - A_{yy})/A_{zz}$$

$$|A_{zz}| \geq |A_{yy}| \geq |A_{xx}|$$

$$A_{xx}(PAS) = \sqrt{\frac{5}{24\pi}}[\eta - 1] \qquad A_{yy}(PAS) = -\sqrt{\frac{5}{24\pi}}[1 + \eta] \qquad A_{zz}(\theta,\varphi) = \sqrt{\frac{5}{6\pi}}$$

$$A_{xz}(PAS) = 0 = A_{zx}(PAS) = A_{xy}(PAS) = A_{yx}(PAS) = A_{yz}(PAS) = A_{zy}(PAS)$$

Figure 2-7 **Rank 2 interaction equations in its principal axis orientation (PAS).**

Included are the general relationships between the (GAMMA scaled) Cartesian tensor components to the irreducible spherical components. They are valid when η is defined accordingly! If η is defined by the other common convention ($|A_{zz}| \geq |A_{xx}| \geq |A_{yy}|$) then the sign on the $A_{2,\pm2}$ will change as will the sign on the Hamiltonian term multiplied by η.

2.15.10 Reoriented Irreducible Rank 2 Spherical Spatial Tensor Components

It is often the case that multiple rotations on a particular tensor must be performed.We can express the irreducible rank 2 spatial tensor components $A_{2,\,m}$ relative to any arbitrary axis system (AAS) by a rotation from the principal axes to the new axes *via* the formula[1]

$$A_{2,\,m}(AAS) = \sum_{m'}^{\pm 2} D^2_{m'm}(\Omega) A_{2,\,m'}(PAS) \tag{0-16}$$

where $D^2_{m'm}(\Omega)$ are the rank 2 Wigner rotation matrix elements and Ω the set of three Euler angles which relate the principal axes of the rank 2 spatial tensor to the arbitrary axes. For the treatment liquid isotropic systems these Euler angles are time dependent and averaged. For the treatment of solids they may be static (powder) or time dependent (MAS). Often only two angles suffice to orient the interaction relative to its PAS.
We can expand the components of the oriented spatial tensor in terms of the Wigner rotation elements as well as the reduced Wigner rotation elements $d^2_{mm'}$ which are given by

$$\boldsymbol{D}^2_{m,\,n}(\alpha, \beta, \gamma) = e^{-i\alpha m} d^2_{m,\,n}(\beta) e^{-i\gamma n} \tag{0-17}$$

Putting these into the formula for rotating the spatial tensor

$$A_{2,\,m}(\theta, \varphi) = \sum_{m'}^{\pm 2} D^2_{m'm}(0, \theta, \varphi) A_{2,\,m'}(PAS) \tag{0-18}$$

1. This rotation takes the PAS into the oriented coordinate axes.

2.16 Rank 2 Interaction Parameters

This section describes how an ASCII file may be constructed that is self readable by a rank 2 interaction. The file can be created with any editor and is read with the rank 2 interaction member function “read”. An example of one such file is given in its entirety at the end of this section. Keep in mind that parameter ordering in the file is arbitrary. Other parameters are allowed in the file which do not relate to rank 2 interactions.

Table 2: Rank 2 Interaction Parameters

Parameter	Units	Examples Parameter (Type[a]) : Value - Statement
R2eta	none	R2eta (1) : 0.33 - Rank 2 Asymmetry[b]
R2theta	degrees	R2theta (1) : 127.2 - Orientation from PAS z (deg)[c]
R2phi	degrees	R2phi (1) : 270.9 - Orientation from PAS x (deg)[d]

a. Parameter type 1 indicates an double precision number parameter.
b. The asymmetry parameter must be within the range of [0, 1]. This parameter does not need to be set for a rank 2 interaction definition, it will be assumed 0 if unspecified.
c. The angle theta which relates the rank 2 interactions orientation down from the z-axis of its PAS may be set. This is not essential and will be taken as zero in left unspecified
d. The angle phi which relates the rank 2 interactions orientation over from the x-axis of its PAS may be set. This is not essential and will be taken as zero in left unspecified

2.17 Literature Comparisons

2.17.1 P.P. Man's "Rank 2 Interactions"

The following figure lists some of the equations found in Pascal P. Man's article[1] along with the corresponding GAMMA equations. Aside from difference in scaling factors, GAMMA is in full agreement with these equations.

Comparison of GAMMA & P.P. Man's Rank 2 Equations

Γ's Equations

$$A^{Q}_{2,0}(PAS) = \sqrt{\frac{5}{4\pi}} \qquad A^{Q}_{2,\pm1}(PAS) = 0 \qquad A^{Q}_{2,\pm2}(PAS) = \sqrt{\frac{5}{24\pi}}\eta$$

$$\eta = (A_{xx} - A_{yy})/A_{zz} \qquad |A_{zz}| \geq |A_{yy}| \geq |A_{xx}|$$

1. "Rank 2 Interactions", P.P. Man, Encyclopedia of Nuclear Magnetic Resonance, Editors-in-Chief D.M.Grant and R.K. Harris, Vol. 6, Ped-Rel, pgs. 3838-3948.

2.18 Rank 2 Interaction Examples

2.18.1 Zero Field Transitions, First Order Spectra

2.19 References

[1] D.M.Grant and R.K. Harris, Eds. in Chief, (1996), *Encyclopedia of Nuclear Magnetic Resonance*, John Wiley & Sons, New York.

[2] Brink, D.M. and Satchler, G.R. (1962), Angular Momentum, Clarendon Press, Oxford.

0.1 Programs and Input Files

IntQu_LC0.cc

```
/* IntQu_LC0.cc ************************************************************-*-c++-*-
**                                                                            **
** Update:     10/11/96                                                       **
** Version:    3.6                                                            **
** Copyright:   S. Smith.  You can modify this program for personal use, but  **
**                            you must leave it intact if you re-distribute it.  **
**                                                                            **
*******************************************************************************/

#include <gamma.h>                          // Include GAMMA

main (int argc, char* argv[])
{
//                          Set Up The Rank 2 Interaction

  int qn=1;
  double I;
  query_parameter(argc, argv, qn++,                  // Read in the coupling
     {
```

3 Dipole-Dipole Interactions

3.1 Overview

The class IntDip contains a fully functional dipole-dipole interaction defined between two spins. The class allows for the definition and manipulation of such interactions, in particular it allows for the construction of oriented dipolar Hamiltonians.

3.2 Available Functions

Constructors

IntDip	- Dipolar interaction constructor	page 3-65
=	- Assignment operator	page 3-66

Basic Functions

delzz	- Get or set the Dipolar spatial tensor delzz value	page 3-67
DCC	- Get or set the Dipolar coupling constant	page 3-67
eta	- Get or set the Dipolar spatial tensor asymmetry	page 3-68
xi	- Get the Dipolar interaction constant	page 3-70

Spherical Spatial Tensor Functions (Inherited[1])

A0, A20	- Get Dipolar m=0 spherical tensor component)	page 3-71
A1, A21	- Get Dipolar m=1 spherical tensor component	page 3-71
Am1, A2m1	- Get Dipolar m=-1 spherical tensor component	page 3-71
A2, A22,	- Get Dipolar m=2 spherical tensor component	page 3-71
Am2, A2m2	- Get Dipolar m=-2 spherical tensor component	page 3-71

Cartesian Spatial Tensor Functions (Inherited[2])

Axx, Ayy, Azz	- Get the xx, yy, zz Cartesian tensor component	page 3-73
Axy, Ayx, Axz,	- Get the xy, yx, xz Cartesian tensor component	page 3-73
Azx, Ayz, Azy	- Get the zx, yz, zy Cartesian tensor component	page 3-73

Spherical Spatial Tensor Functions For Averaging

A0A, A20A	- Get Dipolar m=0 tensor component constructs over sphere	page 3-75
A1A, A21A	- Get Dipolar m=1 tensor component constructs over sphere	page 3-75
A2A, A221A	- Get Dipolar m=2 tensor component constructs over sphere	page 3-76
A0B, A20B	- Get Dipolar m=0 tensor component constructs over sphere	page 3-77
A1B, A21B	- Get Dipolar m=1 tensor component constructs over sphere	page 3-78
A2B, A22B	- Get Dipolar m=2 tensor component constructs over sphere	page 3-79

1. These functions are inherited from the base class IntRank2.
2. These functions are inherited from the base class IntRank2.

Spherical Spin Tensor Functions

Auxiliary Functions

Hamiltonian Functions

Input Functions

Output Functions

3.3 Theory Sections

3.4 Chapter Figures

3.5 Literature Comparisons

3.6 Examples

3.7 Example Programs

3.8 Constructors

3.8.1 IntDip

Usage:

void IntDip::IntDip()
void IntDip::IntDip(const IntDip& D1)
void IntDip::IntDip(const String& II, const String& IS, double dz, double theta=0, double phi=0, double eta=0)
void IntDip::IntDip(const String& II, const String& IS, const coord& cI, const coord& cS)
void IntDip::IntDip(double qI, double qS, double dz, double theta=0, double phi=0, double eta=0)
void IntDip::IntDip(ParameterAVLSet& pset, int idx=-1, int warn=2)
void IntDip::IntDip(int idxI, int idxS, ParameterAVLSet& pset, int warn=2)

Description:

The function ***IntDip*** is used to create a new Dipolar interaction.

1. IntDip() - Without arguments the function creates a NULL dipolar interaction.
2. IntDip(const IntDip& D1) - With another dipolar interaction makes a new identical dipolar interaction
3. IntDip(const String& II, const String& IS, double dz, double theta=0, double phi=0, double eta=0) - This will form a dipolar interaction between two spins of type ***II*** & ***IS*** (e.g. 1H, 13C, 14N,...) having a dipolar coupling constant of ***dz*** (in Hz). The dipolar orientation (the vector between the two spins) will be set at angle ***theta*** (down from +z) and ***phi*** (over from +x) input in degees with respect to the laboratory frame. An optional asymmetry ***eta*** ([0,1]) can be input, but this is normally 0 for dipolar interactions. Note that neither ***II*** or ***IS*** are allowed to be an electron (e-), either results in a fatal error.
4. IntDip(double qI, double qS, double dz, double theta=0, double phi=0, double eta=0) - Similar to previous overload (3.) except that spin quantum values ***qI*** and ***qS*** are input rather than isotope labels. The spin quantum values must be positive integer multiples of 1/2 (i.e. 1, 1.5, 2.5, 3.0....)
5. IntDip(const String& II, const String& IS, const coord& cI, const coord& cS) - Forms a dipolar interaction between two spins of type ***II*** & ***IS*** (e.g. 1H, 13C, 14N,...) where the spins are positioned in the laboratory frame at points ***cI*** and ***cS***.
6. IntDip(ParameterAVLSet& pset, int idx=-1, int warn=2) - Constructs a new dipolar interaction from interaction indexed parameters found in the parameter set ***pset***. If the optional interaction index ***idx*** has been set ***!=-1*** the dipolar parameters scanned in ***pset*** will be assumed to have a ***(idx)*** appended to their names. See valid parameter descriptions later in this chapter. The optional flag warn sets how the constructor treats failures to find proper parameters (0=no warnings, 1=warnings, >1=fatal errors). Note that since this constructor uses a single interaction index (idx) it will NOT utilize isotope labels and coordinate specifications. Spin quantum values must be positive integer multiples of 1/2 (i.e. 1, 1.5, 2.5, 3.0....)
7. IntDip(double qI, double qS, ParameterAVLSet& pset, int idx=-1, int warn=2) - Same as the previous constructor except that the spin quantum numbers are directly specified with ***qI*** and ***qS.***
8. IntDip(int idxI, int idxS, ParameterAVLSet& pset, int warn=2) - Similar to earlier constructor uses parameters having spin indices ***idxI*** and ***idxS.*** Preferentially takes spin isotope labels and spin coordinates.

Return Value:

Void. It is used strictly to create a dipolar interaction.

Examples:

```
#include <gamma.h>
main()
  {
  IntDip D;                                  // An empty Dipolar interaction.
  IntDip D1(1.5, 3.e5,.2, 45., 30.);         // D. Int. for I=3/2, DCC=300kHz, η=.2, θ=45, ϕ=30
  IntDip D2(D1);                             // Another Dip. Interaction, here equal to D1
  D = D2;                                    // Now D is the same as D1 and D2
  }
```

See Also: =, read, ask_read

3.8.2 =

Usage:

void IntDip::operator= (const IntDip& D1)

Description:

The operator = is assign one Dipolar interaction to another.

Return Value:

Void.

Example:

```
#include <gamma.h>
main()
  {
  IntDip D;                                  // An empty Dipolar interaction.
  IntDip D1(1.5, 3.e5,.2, 45., 30.);         // D. Int. for I=3/2, DCC=300kHz, η=.2, θ=45, ϕ=30
  D = D1;                                    // Now D is the same as D1
  }
```

See Also: constructor, read, ask_read

3.9 Basic Functions

3.9.1 delzz

Usage:

```
#include <IntDip.h>
double IntDip::delzz () const
double IntDip::delz () const
double IntDip::delzz (double dz) const
double IntDip::delz (double dz) const
```

Description:

The function ***delzz*** is used to either obtain or set the interaction Dipolar coupling constant. With no arguments the function returns the coupling in Hz. If an argument, ***dz***, is specified then the coupling constant for the interaction is set. It is assumed that the input value of ***dz*** is in units of ***Hz***. The function is overloaded with the name ***delz*** for convenience. Note that setting of ***delzz*** will alter the (equivalent) value of the Dipolar coupling ***DCC*** as well as the dipolar frequency.

$$\delta_{zz}^{D} = \frac{h^2\gamma_i\gamma_j}{r_{ij}^3} = DCC$$

Return Value:

Either void or a floating point number, double precision.

Examples:

```
#include <gamma.h>
main()
  {
  IntDip D();                    // Empty Dipolar interaction.
  D.delzz(100000.0);             // Set DCC to 100 KHz.
  cout << D.delz ();             // Write coupling constant to std output.
  }
```

See Also: DCC, wD

3.9.2 DCC

Usage:

```
#include <IntDip.h>
double IntDip::DCC () const
double IntDip::DCC (double dz) const
```

Description:

The function ***DCC*** is used to either obtain or set the interaction dipolar coupling constant. With no arguments the function returns the coupling in Hz. If an argument, ***dz***, is specified then the coupling constant for the interaction is set. It is assumed that the input value of ***dz*** is in units of ***Hz***. Note that setting of ***DCC*** will alter the (equivalent) value of the dipolar spatial tensor ***delzz*** value as well as the dipolar frequency. This function has identical functionality as ***delzz*** and ***delz***.

$$\delta_{zz}^{D} = \frac{h^2\gamma_i\gamma_j}{r_{ij}^3} = DCC$$

Return Value:

Either void or a floating point number, double precision.

Examples:

```
#include <gamma.h>
main()
  {
  IntDip D();                    // Empty Dipolar interaction.
  D.NDCC(100000.0);              // Set DCC to 100 KHz.
  cout << D.DCC ();              // Write coupling constant to std output.
  }
```

See Also: delz, delzz, wD

3.9.3 eta

Usage:

```
#include <IntDip.h>
double IntDip::eta () const
double IntDip::eta (double Deta) const
```

Description:

The function ***eta*** is used to either obtain or set the Dipolar interaction asymmetry. With no arguments the function returns the asymmetry (unitless). If an argument, ***Deta***, is specified then the asymmetry for the interaction is set.The input value is **restricted to the range [0,1]** and is related to the Dipolar spatial tensor Cartesian components according to

$$\eta = (A_{xx} - A_{yy})/A_{zz} \qquad |A_{zz}| \geq |A_{yy}| \geq |A_{xx}|$$

Note that setting ***eta*** will alter the 5 internal irreducible spherical spatial tensor components of the interaction. In most treatments[1] of dipolar interactions, ***eta=0.***

Return Value:

Either void or a floating point number, double precision.

Examples:

```
#include <gamma.h>
main()
  {
  IntDip D();                    // Empty Dipolar interaction.
  D.eta(0.75);                   // Set eta to 0.75.
  double Deta = D.eta();         // Set Deta to current eta value
  }
```

See Also: delz, delzz, wD

1. Non-zero eta values have been used to mimic exchange effects where the interaction doesn't reside along the spin-spin internuclear vector but between it and the internuclear vector with a third spin exchange partner.

3.9.4 wD

Usage:

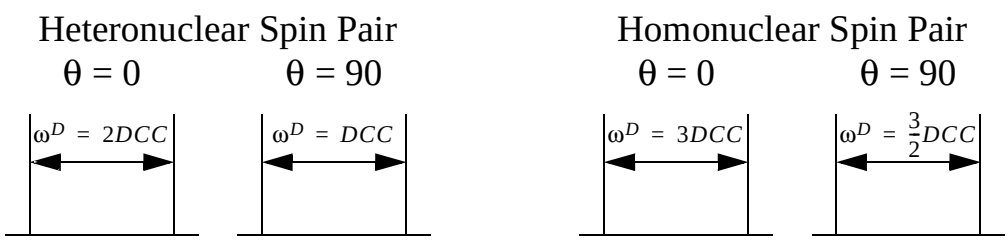
```
#include <IntDip.h>
double IntDip::wD() const
double IntDip::wD(double W) const
```

Description:

The function ***wD*** is used to either obtain or set the interaction dipolar frequency in Hz. This frequency will be the ***observed*** splitting between transitions under the dipolar interaction[1] and varies with orientation[2] according to

$$\omega^D(\theta, \varphi) = \frac{1}{2}\omega_o^D[3\cos^2\theta - 1 - \eta\sin^2\theta\cos 2\varphi]\Big|_{\substack{\eta = 0 \\ \theta = 0}} = \omega_o^D$$

The frequency may be quickly related to the dipolar coupling constant at $\theta = 0, \pm 90$ and $\eta = 0$.



With no arguments the function returns the frequency at the current interaction orientation. If an argument, ***W***, is specified then the frequency for the interaction is set.

Return Value:

Either void or a floating point number, double precision.

Examples:

```
#include <gamma.h>
main()
  {
  IntDip D();                    // Empty Dipolar interaction.
  D.wD(1.4e5);                   // Set Dip. frequency to 140 KHz.
  cout << D.wD();                // Write frequency to std output.
  }
```

See Also: delz, delzz, DCC, NDCC, xi

1. This is true when the Larmor frequency is large (or Bo field strong) relative to the dipolar coupling constant. If the external Bo field is weak, the dipolar interaction will compete with the Zeeman interaction and alter the observed frequencies. Additional interactions will may also alter these frequencies.
2. There are variations in the literature as to what the Dipolar frequency is. The definition in GAMMA is set such that the dipolar interaction will split the observed NMR transitions by ω^D when the Zeeman interaction is strong (i.e. high field, first-order dipolar interaction). This definition is analogous to that of an isotropic scalar (J) coupling between a spin pair, although in the dipolar case the splittig varies with orientation.

3.9.5 xi

Usage:

double IntDip::xi() const

Description:

The function ***xi*** is used to either obtain the GAMMA defined dipolar interaction constant. The constant is used to scale the interaction such that both its spatial and spin tensors are “independent” of the interaction type.

$$\xi^D = -2\sqrt{\frac{6\pi}{5}}\frac{h^2\gamma_i\gamma_j}{r_{ij}^3} = -2\sqrt{\frac{6\pi}{5}}\delta_{zz}^D = \sqrt{\frac{2\pi}{15}}DCC = \sqrt{\frac{2\pi}{15}}\omega_{o,\,\gamma_i \neq \gamma_j}^D$$

This will be used in the formulation of Dipolar Hamiltonians according to.

$$\boldsymbol{H}^D(\theta, \varphi) = \xi^D \sum_m^{\pm 2} (-1)^m A_{2,\,-m}^D(\theta, \varphi) \bullet \boldsymbol{T}_{2,\,m}^D$$

Return Value:

A floating point number, double precision.

Examples:

```
#include <gamma.h>
main()
  {
  IntDip D(1.5, 3.e5, 0.2, 45.0, 45.0);        // Make a Dipolar interaction.
  double Xi = D.xi();                          // Get Dip. interaction constant.
  }
```

3.10 Spherical Spatial Tensor Functions

3.10.1 A0, A20
3.10.2 A1, A21
3.10.3 Am1, A2m1
3.10.4 A2, A22,
3.10.5 Am2, A2m2

Usage:

```
#include <IntDip.h>
complex IntRank2A::A0() const
complex IntRank2A::A20() const
complex IntRank2A::A0(double theta, double phi) const
complex IntRank2A::A20(double theta, double phi) const
complex IntRank2A::A1() const
complex IntRank2A::A21() const
complex IntRank2A::A1(double theta, double phi) const
complex IntRank2A::A21(double theta, double phi) const
complex IntRank2A::Am1() const
complex IntRank2A::A2m1() const
complex IntRank2A::Am1(double theta, double phi) const
complex IntRank2A::Am21(double theta, double phi) const
complex IntRank2A::A2() const
complex IntRank2A::A22() const
complex IntRank2A::A2(double theta, double phi) const
complex IntRank2A::A22(double theta, double phi) const
complex IntRank2A::Am2() const
complex IntRank2A::A2m2() const
complex IntRank2A::Am2(double theta, double phi) const
complex IntRank2A::A2m2(double theta, double phi) const
```

Description:

The functions ***AM*** and ***A2M*** are used to obtain the dipolar interaction spatial tensor component $A_{2,m}$. Here, the names are mapped to the spherical tensor components as follows:

$$\{A0, A20\} \rightarrow A_{2,\,0}$$

$$\{A1, A21\} \rightarrow A_{21} \qquad \{Am1, A2m1\} \rightarrow A_{2,\,-1}$$

$$\{A2, A22\} \rightarrow A_{2,\,2} \qquad \{Am2, A2m2\} \rightarrow A_{2,\,-2}$$

If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments ***theta*** and ***phi*** are given, the returned tensor component is for the interaction orientation having its PAS z-axis at ***theta*** degrees down from laboratory z-axis and its PAS x-axis at ***phi*** degrees over from the labframe x-axis. The values of ***theta*** and ***phi*** are assumed in ***degrees***.

$$A_{2,0}(\theta,\varphi) = \sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]$$

$$A_{2,1}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}\sin\theta[3\cos\theta - \eta(\cos\theta\cos 2\varphi - i\sin 2\varphi)] = -A_{2,1}{}^*(\theta,\varphi)$$

$$A_{2,2}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}\frac{1}{2}[3\sin^2\theta + \eta[\cos 2\varphi(1+\cos^2\theta) - i2\sin 2\varphi\cos\theta]] = A_{2,-2}{}^*(\theta,\varphi)$$

Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type[1]. This component can also be related to the Cartesian tensor components for any arbitrary orientation.

$$A_{2,0} = \sqrt{6}[3A_{zz} - Tr\{A\}]$$

$$A_{21} = -\frac{1}{2}[A_{xz} + A_{zx} + i(A_{yz} + A_{zy})] \qquad A_{2,-1} = \frac{1}{2}[A_{xz} + A_{zx} + i(A_{yz} - A_{zy})]$$

$$A_{2,2} = \frac{1}{2}[A_{xx} - A_{yy} + i(A_{xy} + A_{yx})] \qquad A_{2,-2} = \frac{1}{2}[A_{xx} + (-A_{yy}) - i(A_{xy} + A_{yx})]$$

Return Value:

A complex number.

Example:

```
#include <gamma.h>
main()
  {
  IntDip D(0.5,1.5, 3.e4, 45.0, 45.0);      // Make a Dipolar interaction (I=1/2, S=3/2, DCC=30kHz).
  complex A20 = D.A20();                    // This is at theta=phi=45 degrees
  cout << D.A20(15.6, 99.3);                // This is at theta=15.6 and phi=99.3 degrees.
  }
```

See Also:Axx, Axy

1. Because the GAMMA platform accommodates different interaction types, the scaling on all spatial tensors is chosen to be independent of the interaction. Rather, the spatial tensors are related directly to the familiar rank two spherical harmonics $A_{2,m}(\theta,\varphi)\big|_{\eta=0} = Y_m^2(\theta,\varphi)$. Also, the sign on the term(s) involving η will have opposite sign if the common alternative definition of the PAS orientation ($|A_{zz}| \geq |A_{xx}| \geq |A_{yy}|$) is used rather that the definition used in GAMMA ($|A_{zz}| \geq |A_{yy}| \geq |A_{xx}|$)

3.11 Cartesian Spatial Tensor Functions

3.11.1 Axx, Ayy, Azz

3.11.2 Axy, Ayx, Axz,

3.11.3 Azx, Ayz, Azy

Usage:

```
#include <IntDip.h>
complex IntRank2::Axx() const
complex IntRank2::Axx(double theta, double phi) const
complex IntRank2::Ayy() const
complex IntRank2::Ayy(double theta, double phi) const
complex IntRank2::Azz() const
complex IntRank2::Azz(double theta, double phi) const
complex IntRank2::Axy() const
complex IntRank2::Axy(double theta, double phi) const
complex IntRank2::Ayx() const
complex IntRank2::Ayx(double theta, double phi) const
complex IntRank2::Axz() const
complex IntRank2::Axz(double theta, double phi) const
complex IntRank2::Azx() const
complex IntRank2::Azx(double theta, double phi) const
complex IntRank2::Ayz() const
complex IntRank2::Ayz(double theta, double phi) const
complex IntRank2::Azy() const
complex IntRank2::Azy(double theta, double phi) const
```

Description:

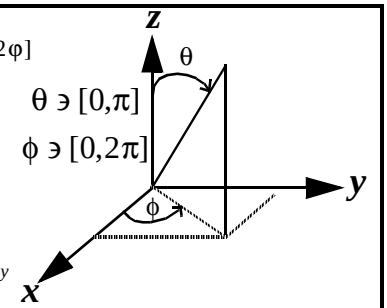
The functions ***Auv*** are used to obtain the dipolar interaction spatial tensor Cartesian components A_{uv}. If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments ***theta*** and ***phi*** are given, the returned tensor component is for the interaction orientation having its PAS z-axis at ***theta*** degrees down from laboratory z-axis and its PAS x-axis at ***phi*** degrees over from the labframe x-axis. The values of ***theta*** and ***phi*** are assumed in ***degrees***. The formulae for these components are given below (see Figure 19-7, page 2-53).

$$A_{xx}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}[3\sin^2\theta - 1 + \eta\cos 2\varphi\cos^2\theta] \qquad A_{yy}(\theta,\varphi) = -\sqrt{\frac{5}{24\pi}}[1 + \eta\cos 2\varphi]$$

$$A_{zz}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}[3\cos^2\theta - 1 + \eta\sin^2\theta\cos 2\varphi]$$

$$A_{xz}(\theta,\varphi) = -\sqrt{\frac{5}{24\pi}}\sin\theta\cos\theta[3 - \eta\cos 2\varphi] = A_{zx}$$

$$A_{xy}(\theta,\varphi) = -\sqrt{\frac{5}{24\pi}}\eta\sin 2\varphi\cos\theta = A_{yx} \qquad A_{yz}(\theta,\varphi) = -\sqrt{\frac{5}{24\pi}}\eta\sin\theta\sin 2\varphi = A_{zy}$$



Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type.

Return Value:

A complex number.

Example:

```
#include <gamma.h>
main()
  {
  IntDip D("13C","2H", 2.e3);          // Make a Dipolar interaction (I=1/2, S=1, DCC=2kHz).
  complex Azz = D.Azz();               // This is Axx @ theta=phi=45 degrees
  cout << D.Axx(15.6, 99.3);           // This is Axx @ theta=15.6 and phi=99.3 degrees.
  }
```

See Also: A20, A22

3.12 Powder Average Facilitator Functions

3.12.1 A0A, A20A

Usage:

row_vector IntDip::A0A(int Ntheta)
row_vector IntDip::A20A(int Ntheta)

Description:

The functions ***A0A*** and ***A20A*** are equivalent. They are used to obtain part of Dipolar interaction spatial tensor component $A_{2,0}$ for a series of evenly incrmented θ values.

$$A_{2,0}A(\theta) = \sqrt{\frac{5}{16\pi}}(3\cos^2\theta - 1) = A_{2,0}(\theta, \varphi)\Big|_{\eta = \varphi = 0}$$

Given a number of angle increments, ***Ntheta***, a row vector of dimension ***Ntheta*** will be returned which contains the η independent terms of $A_{2,0}$ at evenly spaced increments of θ starting at the +z PAS ($\theta = 0$) alignment and finishing at -z PAS ($\theta = 180$) alignment.

$$\langle v|i\rangle = A_{2,0}A(\theta_i) \qquad \theta_i = \frac{180i}{(\text{Ntheta} - 1)}$$

Note that to obtain the full $A_{2,0}$ terms (if they are η dependent) they must be properly combined with the values from the function ***A20B.***

Return Value:

A vector.

Example:

```
#include <gamma.h>
main()
  {
  IntDip D(1.5, 3.e5, 0.2, 45.0, 45.0);        // Make a Dipolar interaction.
  row_vector A20s = D.A20A(720);               // Get 720 A20A values spanning [0, 180]
  }
```

See Also: A21A, A22A, A20B, A21B, A22B, A2As, A2Bs, A2s

3.12.2 A1A, A21A

Usage:

row_vector IntDip::A1A(int Ntheta)
row_vector IntDip::A21A(int Ntheta)

Description:

The functions ***A1A*** and ***A21A*** are equivalent. They are used to obtain part of Dipolar interaction spatial tensor component $A_{2,1}$ for a series of evenly incrmented θ values.

$$A_{2,1}A(\theta) = 3\sqrt{\frac{5}{24\pi}}\sin\theta\cos\theta = A_{2,1}(\theta,\varphi)\big|_{\eta=0}$$

Given a number of angle increments, ***Ntheta***, a row vector of dimension ***Ntheta*** will be returned which contains the η independent terms of $A_{2,1}$ at evenly spaced increments of θ starting at the +z PAS ($\theta = 0$) alignment and finishing at -z PAS ($\theta = 180$) alignment.

$$\langle v|i\rangle = A_{2,1}A(\theta_i) \qquad \theta_i = \frac{180i}{(\mathrm{Ntheta}-1)}$$

Note that to obtain the full $A_{2,1}$ terms (if they are η dependent) they must be properly combined with the values from the function ***A21B.***

Return Value:

A vector.

Example:

```
#include <gamma.h>
main()
  {
  IntDip D(1.5, 3.e5, 0.2);                    // Make a Dipolar interaction.
  row_vector A21s = D.A21A(181);               // Get 181 A20A values spanning [0, 180]
  }
```

See Also: A20A, A22A, A20B, A21B, A22B, A2As, A2Bs, A2s

3.12.3 A2A, A221A

Usage:

row_vector IntDip::A2A(int Ntheta)
row_vector IntDip::A22A(int Ntheta)

Description:

The functions ***A2A*** and ***A22A*** are equivalent. They are used to obtain part of Dipolar interaction spatial tensor component $A_{2,2}$ for a series of evenly incremented θ values.

$$A_{2,2}A(\theta) = \frac{3}{2}\sqrt{\frac{5}{24\pi}}\sin^2\theta = 3\sqrt{\frac{5}{96\pi}}\sin^2\theta = A_{2,2}(\theta,\varphi)\big|_{\eta=0}$$

Given a number of angle increments, ***Ntheta***, a row vector of dimension ***Ntheta*** will be returned which contains the η independent terms of $A_{2,2}$ at evenly spaced increments of θ starting at the +z PAS ($\theta = 0$) alignment and finishing at -z PAS ($\theta = 180$) alignment.

$$\langle v|i\rangle = A_{2,2}A(\theta_i) \qquad \theta_i = \frac{180i}{(\mathrm{Ntheta}-1)}$$

Note that to obtain the full $A_{2,2}$ terms (if they are η dependent) they must be properly combined with the values

from the function ***A22B.***

Return Value:

A vector.

Example:

```
#include <gamma.h>
main()
  {
  IntDip D(1.5, 3.e5, 0.2);                    // Make a Dipolar interaction.
  row_vector A22s = D.A22A(181);               // Get 181 A22A values spanning [0, 180]
  }
```

See Also: A20A, A21A, A20B, A21B, A22B, A2As, A2Bs, A2s

3.12.4 A0B, A20B

Usage:

row_vector IntDip::A0B(int Nphi)
row_vector IntDip::A20B(int Nphi)

Description:

The functions ***A0B*** and ***A20B*** are equivalent. They are used to obtain part of Dipolar interaction spatial tensor component $A_{2,0}$ for a series of evenly incrmented φ values.

$$A_{2,0}B(\varphi) = \sqrt{\frac{5}{16\pi}}\eta\cos 2\varphi = \frac{1}{\sin^2\theta}[A_{2,0}(\theta,\varphi) - A_{2,0}(\theta,\varphi)|_{\eta=0}]$$

Given a number of angle increments, ***Nphi***, a row vector of dimension ***Nphi*** will be returned which contains θ independent terms of $A_{2,0}$ at evenly spaced increments of φ starting at the +x PAS ($\varphi = 0$) alignment and finishing at +x PAS ($\varphi = 360$) alignment.

$$\langle v|i\rangle = A_{2,0}A(\varphi_i) \qquad \varphi_i = \frac{360i}{\mathrm{Nphi}}$$

Note that to obtain the full $A_{2,0}$ terms they must be properly combined with the values from the function ***A20A.***

$$A_{2,0}(\theta,\varphi) = \sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]$$

$$A_{2,1}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}\sin\theta[3\cos\theta - \eta(\cos\theta\cos 2\varphi - i\sin 2\varphi)] = -A_{2,-1}{}^*(\theta,\varphi)$$

$$A_{2,2}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}\frac{1}{2}[3\sin^2\theta + \eta[\cos 2\varphi(1+\cos^2\theta) - i2\sin 2\varphi\cos\theta]] = A_{2,-2}{}^*(\theta,\varphi)$$

Return Value:

A vector.

Example:

```
#include <gamma.h>
main()
  {
  IntDip D(1.5, 3.e5, 0.2);                    // Make a Dipolar interaction.
  row_vector A20s = D.A20B(120);               // Get 120 A20B values spanning [0, 360)
  }
```

See Also: A20A, A21A, A22A, A21B, A22B, A2As, A2Bs, A2s

3.12.5 A1B, A21B

Usage:

row_vector IntDip::A1B(int Nphi)
row_vector IntDip::A21B(int Nphi)

Description:

The functions ***A1B*** and ***A21B*** are equivalent. They are used to obtain part of Dipolar interaction spatial tensor component $A_{2,1}$ for a series of evenly incrmented φ values.

$$A_{2,1}B(\varphi) = -\sqrt{\frac{5}{24\pi}}\eta(\cos 2\varphi - i\sin 2\varphi)$$

where

$$A_{2,1}(\theta,\varphi) = \sin\theta\cos\theta Re(A_{2,1}B(\varphi)) + i\sin\theta Im(A_{2,1}B(\varphi)) + A_{2,1}(\theta,\varphi)\Big|_{\eta=0}$$

Given a number of angle increments, ***Nphi***, a row vector of dimension ***Nphi*** will be returned which contains θ independent terms of $A_{2,1}$ at evenly spaced increments of φ starting at the +x PAS ($\varphi = 0$) alignment and finishing at +x PAS ($\varphi = 360$) alignment.

$$\langle v|i\rangle = A_{2,1}A(\varphi_i) \qquad \varphi_i = \frac{360i}{\text{Nphi}}$$

Note that to obtain the full $A_{2,1}$ terms they must be properly combined with the values from the function ***A21A.***

Return Value:

A vector.

Example:

```
#include <gamma.h>
main()
  {
  IntDip D(1.5, 3.e5, 0.2);                    // Make a Dipolar interaction.
  row_vector A21s = D.A21B(120);               // Get 120 A21B values spanning [0, 360)
  }
```

See Also: A20A, A21A, A22A, A20B, A22B, A2As, A2Bs, A2s

3.12.6 A2B, A22B

Usage:

row_vector IntDip::A2B(int Nphi)
row_vector IntDip::A22B(int Nphi)

Description:

The functions ***A1B*** and ***A21B*** are equivalent. They are used to obtain part of Dipolar interaction spatial tensor component $A_{2,2}$ for a series of evenly incrmented φ values.

$$A_{2,2}B(\varphi) = \sqrt{\frac{5}{96\pi}}\eta[\cos 2\varphi - i2\sin 2\varphi]$$

where

$$A_{2,2}(\theta,\varphi) = (1+\cos^2\theta)Re(A_{2,2}B(\varphi)) + i\cos\theta Im(A_{2,2}B(\varphi)) + A_{2,2}(\theta,\varphi)\Big|_{\eta=0}$$

Given a number of angle increments, ***Nphi***, a row vector of dimension ***Nphi*** will be returned which contains θ independent terms of $A_{2,2}$ at evenly spaced increments of φ starting at the +x PAS ($\varphi = 0$) alignment and finishing at +x PAS ($\varphi = 360$) alignment.

$$\langle v|i\rangle = A_{2,2}A(\varphi_i) \qquad \varphi_i = \frac{360i}{\text{Nphi}}$$

Note that to obtain the full $A_{2,2}$ terms they must be properly combined with the values from the function ***A22A.***

Return Value:

A vector.

Example:

```
#include <gamma.h>
main()
  {
  IntDip D(1.5, 3.e5, 0.2);                    // Make a Dipolar interaction.
  row_vector A22s = D.A22B(120);               // Get 120 A22B values spanning [0, 360)
  }
```

See Also: A20A, A21A, A22A, A20B, A21B, A2As, A2Bs, A2s

3.12.7 A2s

Usage:

matrix IntDip::A2s(int Ntheta, int Nphi)

Description:

The function ***A2s*** is used to construct the Dipolar interaction spatial tensor components $A_{2,\,m}$ for a series of evenly incrmented θ and φ values. Given arguments for the number of angle increments, ***Ntheta*** and ***Nphi*** the function will return a matrix of dimenstion (8 x nc) where nc is the larger of the two input arguments. The matrix columns, indexed by j, will then correspond either to an angle θ or an angle φ where

$$\theta_j = \frac{180j}{(\mathrm{Ntheta} - 1)} \qquad \varphi_j = \frac{360j}{\mathrm{Nphi}}$$

depending upon which row is being accessed. Rows 0-2 of the array will correspond to the the η independent terms of $A_{2,\,\{0,\,1,\,2\}}$ at evenly spaced increments of θ starting at the +z PAS ($\theta = 0$) alignment and finishing at -z PAS ($\theta = 180$) alignment. Rows 3-5 of the array will correspond to θ independent parts of the interaction spatial tensor components $A_{2,\,\{0,\,1,\,2\}}$ at evenly spaced increments of φ starting at the +x PAS ($\varphi = 0$) alignment and finishing at +x PAS ($\varphi = 360$) alignment. The final three array columns will contain θ dependent terms that are used to blend with the other rows to form the full $A_{2,\,m}(\theta,\,\varphi)$ values. Reconstruction of full $A_{2,\,m}(\theta,\,\varphi)$ values is based on

$$A_{2,\,0}(\theta,\,\varphi) = A_{2,\,0}(\theta,\,\varphi)\big|_{\eta\,=\,0} + \sin^2\theta A_{2,\,0}B(\varphi)$$

$$A_{2,\,1}(\theta,\,\varphi) = A_{2,\,1}(\theta,\,\varphi)\big|_{\eta\,=\,0} + \sin\theta\cos\theta Re(A_{2,\,1}B(\varphi)) + i\sin\theta Im(A_{2,\,1}B(\varphi))$$

$$A_{2,\,2}(\theta,\,\varphi) = A_{2,\,2}(\theta,\,\varphi)\big|_{\eta\,=\,0} + (1 + \cos^2\theta)Re(A_{2,\,2}B(\varphi)) + i\cos\theta Im(A_{2,\,2}B(\varphi))$$

A particular $A_{2,\,m}(\theta_k,\,\varphi_l)$ can be reconstructed according to the analogous discrete equations.

$$A_{2,\,0}(\theta_k, \varphi_l) = \langle 0|mx|k\rangle + \langle 6|mx|k\rangle^2 \langle 3|mx|l\rangle$$

$$A_{2,\,1}(\theta_k, \varphi_l) = \langle 1|mx|k\rangle + \langle 6|mx|k\rangle[\langle 7|mx|k\rangle Re\langle 4|mx|l\rangle + iIm\langle 4|mx|l\rangle]$$

$$A_{2,\,2}(\theta_k, \varphi_l) = \langle 2|mx|k\rangle + (1 + \langle 7|mx|k\rangle^2) Re\langle 5|mx|l\rangle + i\langle 7|mx|k\rangle Im\langle 5|mx|l\rangle$$

The components with m negative are obtained from the relationship .

$$A_{2,\,-m} = (-1)^m A_{2,\,m}$$

Return Value:

An array.

Example:

```
#include <gamma.h>
main()
  {
  IntDip D(1.5, 3.e5, 0.2);          // Make a Dipolar interaction.
  matrix As = D.A2x(720, 360);       // Get array for values spanning [0, 180] & [0, 360)
  }
```

See Also: A20A, A21A, A22A, A20B, A21B, A2As, A2Bs, A2s

3.13 Spin Space Access Functions

3.13.1 Izval

3.13.2 Szval

Usage:

```
#include <IntRank2T.h>
double IntRank2T::I() const
double IntRank2T::S() const
```

Description:

The function ***I*** is used to obtain the spin quantum number of the first (or only) spin associated with the interaction. The function ***S*** is used to obtain the spin quantum number of the second spin associated with the interaction. Either value will be an integer multiple of 1/2. For single spin interactions the value returned by ***S*** will be 0.

Return Value:

A double.

Example:

```
#include <gamma.h>
main()
  {
  IntRank2T ID("1H", "2H", DIP);          // Make a dipolar rank 2 spin tensor.
  cout << "\n\tI (1H) is: " << ID.I();    // This should be 1/2
  cout << "\n\tI (2H) is: " << ID.S();    // This should be 1
  }
```

See Also: IV, SV, HS

3.13.3 IV

3.13.4 SV

Usage:

```
#include <IntRank2T.h>
int IntRank2T::IV() const
int IntRank2T::SV() const
```

Description:

The function ***IV*** is used to obtain the spin Hilbert space of the first (or only) spin associated with the interaction. The function ***SV*** is used to obtain the spin Hilbert space of the second spin associated with the interaction. The spin Hilbert space is reated to the spin quantum number I as $IV = 2I + 1$ .For single spin interactions the value returned by ***SV*** will be 0.

Return Value:

A double.

Example:

```
#include <gamma.h>
main()
  {
  IntRank2T D1("1H", "2H", 100.e3);        // Make a dipolar rank 2 spin tensor.
  cout << "\n\tHS (1H) is: " << D1.IV();    // This should be 2
  cout << "\n\tHS (2H) is: " << D1.SV();    // This should be 3
  }
```

See Also: I, S, HS

3.13.5 HS

Usage:

```
#include <IntRank2T.h>
int IntRank2T:HS() const
```

Description:

The function ***HS*** is used to obtain the spin Hilbert space associated with the interaction. The spin Hilbert space is reated to the spin quantum number(s). For a dipolar interaction $HS = (2I+1)(2S+1)$.

Return Value:

A double.

Example:

```
#include <gamma.h>
main()
  {
  IntRank2T D1("1H", "2H", 100.e3);        // Make a dipolar rank 2 spin tensor.
  cout << "\n\tHS is: " << D1.HS();         // This should be 6
  }
```

See Also: I, S, IV, SV

3.14 Spin Tensor Functions

3.14.1 Tcomp

Usage:

```
#include <IntDip.h>
matrix IntDip::Tcomp(int m)
```

Description:

The function ***Tcomp*** is used to obtain a dipolar interaction spin tensor component $T^D_{2,m}$. The component desired is specified by the argument ***m*** which spans [-2, 2]. The spin components are given by

$$T^D_{2,0} = \frac{1}{\sqrt{6}}[3I_{iz}I_{jz} - (\boldsymbol{I}_i \bullet \boldsymbol{I}_j)] \qquad T^D_{2,\pm1} = \mp\frac{1}{2}[I_{i\pm}\ I_{jz} + I_{iz}I_{j\pm}\ ] \qquad T^D_{2,\pm2} = \frac{1}{2}[I_{i\pm}\ I_{j\pm}\ ]$$

and are returned as matrices of dimension ***(2I+1)(2S+1)*** where ***I*** & ***S*** are the associated spin quantum numbers.

Return Value: A matrix.

Example:

```
#include <gamma.h>
main()
  {
  IntDip D("1H", "1H", 450.0);        // Make a Dipolar interaction.
  matrix T20 = D.Tcomp(0);            // This is the T20 spin tensor component
  cout << D.Tcomp(-1);                // Have a look at T2-1 on screen.
  }
```

3.14.2 T0het

Usage:

```
#include <IntDip.h>
matrix IntDip::T0het()
```

Description:

Function ***T0het*** returns the dipolar interaction spin tensor component $T^D_{2,0}$ assuming that 1.) the two spins are heteronuclear, and 2.) The high-field approximation applies. This is given by

$$T^D_{2,0} = \frac{1}{\sqrt{6}}[2I_{iz}I_{jz}]$$

and are returned as a matrix of dimension ***(2I+1)(2S+1)*** where ***I*** & ***S*** are the associated spin quantum numbers.

Return Value: A matrix.

Example:

```
#include <gamma.h>
main()
  {
```

```
IntDip D("1H", "2H", 450.0);        // Make a Dipolar interaction.
matrix T20 = D.T0het();             // This is the T20 spin tensor component
cout << T20;                        // Have a look at it on screen.
}
```

3.15 Auxiliary Functions

3.15.1 setPAS

Usage:

```
#include <IntDip.h>
void IntDip::setPAS()
```

Description:

The functions ***setPAS*** is used to orient the Dipolar interaction into it's principal axis system. All 5 spatial tensor components will be set to PAS values and the internal orientation angles set to zero.

Return Value:

None.

Example:

```
#include <gamma.h>
main()
  {
  IntDip D(1.5, 3.e5, 0.2, 45.0, 45.0);      // Make a Dipolar interaction.
  D.setPAS();                                // As if we used D(1.5,3.e5,0.2,0,0)
  }
```

See Also: theta, phi, orient

3.15.2 symmetric

Usage:

```
#include <IntDip.h>
int IntDip::symmetric() const
```

Description:

The functions ***symmetric*** is used to check if the Dipolar interaction has any asymmetry. The function will return true if the interaction is symmetric and false if there is some asymmetry (non-zero eta value). It is a rare case when the assymetry of a dipolar interaction is non-zero, so this will most often return TRUE.

Return Value:

An integer

Example:

```
#include <gamma.h>
main()
  {
  IntDip D(1.5, 3.e5, 0.2, 45.0, 45.0);      // Make a Dipolar interaction.
  if(D.symmetric()) cout << "Yep";           // We should get No for D because eta=0.2)
  else                   << "Nope";
  }
```

See Also: eta

3.15.3 PAS

Usage:

int IntDip::PAS) const

Description:

The function ***PAS*** is used to check if the Dipolar interaction is oriented in its PAS or not. The function will return true if the interaction is PAS aligned and false if not).

Return Value:

An integer

Example:

```
IntDip D(1.5, 3.e5, 0.2, 45.0, 45.0);      // Make a Dipolar interaction.
if(D.PAS()) cout << "Yep";                 // We should get No for D because neither θ or ϕ is 0)
else                    << "Nope";
```

See Also: eta

3.15.4 Dn

Usage:

#include <IntDip.h>

double IntDip::Dn() const

Description:

The functions ***Dn*** is used to obtain the Dipolar interaction spin quantum number. The function will return a double which will be an integer multiple of 0.5 which is not less than 1 (1.0, 1.5, 2.5, 3.0,.....).

Return Value:

A double

Example:

```
#include <gamma.h>
main()
  {
  IntDip D(1.5, 3.e5, 0.2, 45.0, 45.0);      // Make a Dipolar interaction.
  double HS = 2.*D.Dn()+1.;                  // The spin Hilbert space of D
  }
```

See Also: none

3.15.5 wD2DCC

Usage:

#include <IntDip.h>

friend double wD2DCC(double wD, double I)

Description:

The functions ***wD2DCC*** is used to convert a Dipolar frequency ***wD*** for a spin with quantum number ***I*** to a Dipolar coupling constant. The two are related in GAMMA by

$$QCC = e^2qQ = \frac{2I(2I-1)\omega^Q}{3} = 2I(2I-1)\sqrt{\frac{5}{6\pi}}\xi^Q$$

Return Value:

A double

Example:

```
#include <gamma.h>
main()
  {
  double wD = 450.e3;                    // Dip. frequency of 450 kHz.
  double NDCC = wD2DCC(wD, 1.5);         // Dip. coupling if I=3/2
  }
```

See Also: DCC2wD

3.15.6 DCC2wD

Usage:

#include <IntDip.h>

friend double DCC2wD(double DCC, double I)

Description:

The functions ***DCC2wD*** is used to convert a Dipolar coupling constant to a Dipolar frequency. The two are related in GAMMA by

$$\omega^D = \frac{3e^2qQ}{2I(2I-1)} = \frac{3QCC}{2I(2I-1)} = \sqrt{\frac{15}{2\pi}}\xi^Q$$

Return Value:

A double

Example:

```
double DCC = 450.e3;                     // Dip. coupling constant of 450 kHz.
double wD = DCC2wD(wD, 1.5);             // Dip. frequency if I=3/2
}
```

See Also: wD2DCC

3.16 Hamiltonian Functions

3.16.1 H0

Usage:

#include <IntDip.h>are

matrix IntDip::H0() const

matrix IntDip::H0(double theta, double phi) const

Description:

The function ***H0*** is used to obtain the dipolar Hamiltonian as a first order perturbation to the Zeeman Hamiltonian. As such, the returned matrix is "secular" and commutes with both $\boldsymbol{F_z}$ and $\boldsymbol{R_z}$. It will be valid in a rotating frame about the z-axis. It will not be valid unless the Zeeman Hamiltonian (which it is meant to be added to[1]) is much stronger. The return array will have units of ***Hz***. The dimension of the array will be 2I+1 where I is the spin quantum value associated with the interaction. If the input arguments ***heta*** and ***phi*** are given the returned Hamiltonian is for the orientation at ***theta*** degrees down from the interaction PAS z-axis and ***phi*** degrees over from the interaction PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***degrees.***

In GAMMA the first order Dipolar Hamiltonian is given by

$$H_D^{(0)} = \xi^D A_{2,0}(\theta,\phi) T_{2,0}^D = \frac{\omega^D}{12}[3\cos^2\theta - 1][3I_z^2 - I(I+1)]$$

Return Value:

A matrix.

Example:

```
#include <gamma.h>
main()
  {
  IntDip D(1.5, 3.e5, 0.2, 45.0, 45.0);    // Make a Dipolar interaction.
  matrix H = D.H0();                          // Here's the 1st order Dip. Hamiltonian
  cout << H;                                  // Have a look at the Hamiltonian.
  }
```

See Also: H1, Hsec, H

1. A spin in a strong magnetic field will evolve under the influence of both the Zeeman Hamiltonian, H_Z and the Dipolar Hamiltonian H_D. When the Zeeman interaction is much strong than the Dipolar interaction it suffices to use H0 instead of H_D. This is often nice to use because then the two Hamiltonians commute. In evolving a density operator one may then work in the rotating frame at a spin's Larmor frequency by simply removing the Zeeman Hamiltonian and evolving under only H0.

3.16.2 H1

Usage:

#include <IntDip.h>

matrix IntDip::H1() const

matrix IntDip::H1(double theta, double phi) const

Description:

The function ***H1*** is used to obtain the second order Dipolar Hamiltonian as a perturbation to the Zeeman Hamiltonian. As such, the returned matrix is "secular" and commutes with both $\boldsymbol{F_z}$ and $\boldsymbol{R_z}$. It will be valid in a rotating frame about the z-axis. It will not be valid unless the Zeeman Hamiltonian (to which it is meant to be added[1]) is much stronger. The return array will have units of ***Hz***. The dimension of the array will be 2I+1 where I is the spin quantum value associated with the interaction. If the input arguments ***heta*** and ***phi*** are given the returned Hamiltonian is for the orientation at ***theta*** degrees down from the interaction PAS z-axis and ***phi*** degrees over from the interaction PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***degrees***

In GAMMA the second order Dipolar Hamiltonian is given by

$$H_D^{(1)} = -\frac{\xi^2}{2\Omega_o} I_z \{ A_{0,1} A_{0,-1} [4I(I+1) - 8I_z^2 - 1] + A_{0,2} A_{0,-2} [2I(I+1) - 2I_z^2 - 1] \}$$

Return Value:

A matrix.

Example:

```
#include <gamma.h>
main()
  {
  IntDip D(1.5, 3.e5, 0.2, 45.0, 45.0);    // Make a Dipolar interaction.
  matrix H = D.H1();                          // Here's the 2nd order Dip. Hamiltonian
  cout << H;                                  // Have a look at the Hamiltonian.
  }
```

See Also: DCC, NDCC, wD

3.16.3 Hsec

Usage:

#include <IntDip.h>

matrix IntDip::Hsec() const

Description:

The function ***Hsec*** is used to obtain the sum of the first and second order Dipolar Hamiltonians as a perturbation to the Zeeman Hamiltonian. As such, the returned matrix is "secular" and commutes with both $\boldsymbol{F_z}$ and $\boldsymbol{R_z}$. It will be valid in a rotating frame about the z-axis. It will not be valid unless the Zeeman Hamiltonian is much stronger. The return array will have units of ***Hz***. The dimension of the array will be 2I+1 where I is the spin quantum value

1. In the rotating frame the effective Hamiltonian my have all Zeeman contributions removed. Note that the function does not include the 1st order terms, so should be added to the return from the function H0! The function Hsec will do that automatically.

associated with the interaction.

Return Value:

A matrix.

Example:

```
#include <gamma.h>
main()
  {
  IntDip D(1.5, 3.e5, 0.2, 45.0, 45.0);    // Make a Dipolar interaction.
  matrix H = D.H1();                        // Here's the 2nd order Dip. Hamiltonian
  cout << H;                                // Have a look at the Hamiltonian.
  }
```

See Also: DCC, NDCC, wD

3.16.4 H

Usage:

#include <IntDip.h>

matrix IntDip::H() const

Description:

The function ***H*** is used to obtain the Dipolar Hamiltonian. Most likely this will NOT commute with $\boldsymbol{R_z}$. Thus it will be time independent in the laboratory frame (and time dependent in a frame rotating about the z-axis). The return array will have units of ***Hz***. The dimension of the array will be 2I+1 where I is the spin quantum value associated with the interaction.

Return Value:

A matrix.

Example:

```
#include <gamma.h>
main()
  {
  IntDip D(1.5, 3.e5, 0.2, 45.0, 45.0);    // Make a Dipolar interaction.
  matrix H = D.H1();                        // Here's the 2nd order Dip. Hamiltonian
  cout << H;                                // Have a look at the Hamiltonian.
  }
```

See Also: DCC, wD

3.17 Input Functions

3.17.1 read

Usage:

void IntDip::read(const String& filename, int idxI, int idxS)

void IntDip::read(ParameterAVLSet& pset, int idxI, int idxS)

void IntDip::read(const String& filename, double I, double S, int idx)

void IntDip::read(const String& filename, const spin_sys) const

Description:

The function ***read*** is used to create a Dipolar interaction from parameters in either an external ASCII file, ***filename,*** or in a GAMMA parameter set, ***pset***. Additional arguments can be added to specify a dipole index, spin pair indices, or spin quantum numbers. Used of spin pair indices allows for the most diversity in allowed parameters.

Return Value:

Either void or a floating point number, double precision.

Examples:

```
#include <gamma.h>
main()
  {
  IntDip D();                          // Empty Dipolar interaction.
  D.delzz(100000.0);                   // Set DCC to 100 KHz.
  cout << D.delz ();                   // Write coupling constant to std output.
  }
```

See Also: DCC, NDCC, wD

3.17.2 ask

Usage:

#include <IntDip.h>

double IntDip:: () const

double IntDip::delz () const

double IntDip::delzz (double dz) const

double IntDip::delz (double dz) const

Description:

The function ***delzz*** is used to either obtain or set the interaction Dipolar coupling constant. With no arguments the function returns the coupling in Hz. If an argument, ***dz***, is specified then the coupling constant for the interaction is set. It is assumed that the input value of ***dz*** is in units of ***Hz***. The function is overloaded with the name ***delz*** for convenience. Note that setting of ***delzz*** will alter the (equivalent) value of the Dipolar coupling ***DCC***/***NDCC*** as well as the Dipolar frequency.

Return Value:

Either void or a floating point number, double precision.

Example(s):

```
#include <gamma.h>
main()
  {
  IntDip D();                    // Empty Dipolar interaction.
  D.delzz(100000.0);             // Set DCC to 100 KHz.
  cout << D.delz ();             // Write coupling constant to std output.
  }
```

See Also: DCC, NDCC, wD

3.17.3 askset

Usage:

```
#include <IntDip.h>
double IntDip:: () const
double IntDip::delz () const
double IntDip::delzz (double dz) const
double IntDip::delz (double dz) const
```

Description:

The function ***delzz*** is used to either obtain or set the interaction Dipolar coupling constant. With no arguments the function returns the coupling in Hz. If an argument, ***dz***, is specified then the coupling constant for the interaction is set. It is assumed that the input value of ***dz*** is in units of ***Hz***. The function is overloaded with the name ***delz*** for convenience. Note that setting of ***delzz*** will alter the (equivalent) value of the Dipolar coupling ***DCC***/***NDCC*** as well as the Dipolar frequency.

Return Value:

Either void or a floating point number, double precision.

Example(s):

```
#include <gamma.h>
main()
  {
  IntDip D();                    // Empty Dipolar interaction.
  D.delzz(100000.0);             // Set DCC to 100 KHz.
  cout << D.delz ();             // Write coupling constant to std output.
  }
```

See Also: DCC, NDCC, wD

3.18 Output Functions

3.18.1 print

Usage:

```
#include <IntDip.h>
ostream& IntDip::print (ostream& ostr, int fflag=-1) const
```

Description:

The function ***print*** is used to write the dipolar interaction to an output stream ***ostr***. An additional flag ***fflag*** is set to allow some control over how much information is output. The default (***fflag !=0***) prints all information concerning the interaction. If ***fflag*** is set to zero only the basic parameters are printed. Note that this function can be easily set to output to an external file.

Return Value:

The ostream is returned.

Example:

```
#include <gamma.h>
main()
  {
  IntDip D(2.5, 2.e6, 0.2, 45.7, 15.0);      // Make a Dipolar interaction.
  D.print(cout);                             // Write the interaction to standard output.
  D.print(cout, 1);                          // Write again, but with more output.
  }
```

See Also: <<, write

3.18.2 <<

Usage:

```
#include <IntDip.h>
friend ostream& operator << (ostream& out, const IntDip& D)
```

Description:

The operator << defines standard output for the dipolar interaction.

Return Value:

The ostream is returned.

Example:

```
#include <gamma.h>
main()
  {
  IntDip D(1.5, 3.e5, 0.2);                  // Make a Dipolar interaction.
  cout << D;                                 // Write the interaction to standard output.
  }
```

See Also: print, write

3.18.3 printSpherical

Usage:

#include <IntDip.h>

ostream& IntDip::print (ostream& ostr, int fflag=-1)

Description:

The function ***print*** is used to write the interaction Dipolar coupling constant to an output stream ***ostr***. An additional flag ***fflag*** is set to allow some control over how much information is output. The default (***fflag !=0***) prints all information concerning the interaction. If ***fflag*** is set to zero only the basis parameters are printed.

Return Value:

The ostream is returned.

Example:

```
#include <gamma.h>
main()
  {
  IntDip D(2.5, 2.e6, 0.2, 45.7, 15.0);      // Make a Dipolar interaction.
  cout << D;                                 // Write the interaction to standard output.
  }
```

See Also: <<

3.18.4 printCartesian

Usage:

#include <IntDip.h>

ostream& IntDip::print (ostream& ostr, int fflag=-1)

Description:

The function ***print*** is used to write the interaction Dipolar coupling constant to an output stream ***ostr***. An additional flag ***fflag*** is set to allow some control over how much information is output. The default (***fflag !=0***) prints all information concerning the interaction. If ***fflag*** is set to zero only the basis parameters are printed.

Return Value:

The ostream is returned.

Example:

```
#include <gamma.h>
main()
  {
  IntDip D(2.5, 2.e6, 0.2, 45.7, 15.0);      // Make a Dipolar interaction.
  cout << D;                                 // Write the interaction to standard output.
  }
```

See Also: <<

3.18.5 write

Usage:

#include <IntDip.h>

int IntDip::write (const String& filename, int idx=-1, int pfx=-1, int warn=2) const

int IntDip::write (ofstream& ostr, int idx=-1, int pfx=-1, int warn=2) const

Description:

The function ***write*** is used to write the dipolar interaction to either an external ASCII file, ***filename***, or to an output filestream, ***ostr***. The interaction is written in terms of dipolar interaction parameters which should match those used in the class ***read*** functions. The output parameter names will contain (#) as a suffix where #=***idx*** if ***idx*** is set to other than its default value of -1. The output parameter names will contain [#] as a prefix where #=***pfx*** if ***pfx*** is set to other than its default value of -1. The function returns ***TRUE/FALSE*** (0 or 1) depending upon whether the output was successful. The flag ***warn*** is used to set the action taken by the function when the output fails. If ***warn*** > 0 the the function will output messages to standard output that it has failed. If the value of ***warn*** > 1 the function will abort the program before ***FALSE*** is returned. Note: becas

Return Value:

An integer (0=FALSE, 1=TRUE) is returned.

Example:

```
#include <gamma.h>
main()
  {
  IntDip D(0.5, 1.0, 2037.2, 45.7, 15.0);   // Make a Dipolar interaction (DCC=2.037 kHz).
  String fname("newD.pset");                // File name we'll use for output
  write(fname);                             // Write interaction to ASCII file named newD.pset.
  fname = String("Ds.pset");                // Another file name we'll use
  ofstream out(fname);                      // Open an output stream for file named Ds.pset
  D.write(out,0);                           // Write D into Ds.pset with suffix (0)
  IntDip D2("1H", "13C", 127e3);            // Another dipolar interaction (DCC=127 kHz)
  D2.write(out,1);                          // Write D2 into Ds.pset with suffix (1)
  out.close();                              // Close the output stream
  }
```

See Also: read, print, <<

3.19 Description

3.19.1 Overview

There is an orientational energy dependence that occurs between two point charges (two nuclei) in an externally applied magnetic field. This dipolar interaction is of rank 2 and symmetric about the internuclear axis. It produces relaxation effects in liquid NMR and orientationally dependent shifts in solids.

3.19.2 Coordinate Systems

We will shortly concern ourselves with the mathematical representation of dipole-dipole interactions, in particular their description in terms of spatial and spin tensors. The spatial tensors will be cast in both Cartesian and spherical coordinates and we will switch between the two when convenient. The figure below relates the orientation angles theta and phi to the standard right handed coordinate system in all GAMMA treatments.

Cartesian and Spherical Coordinate Systems

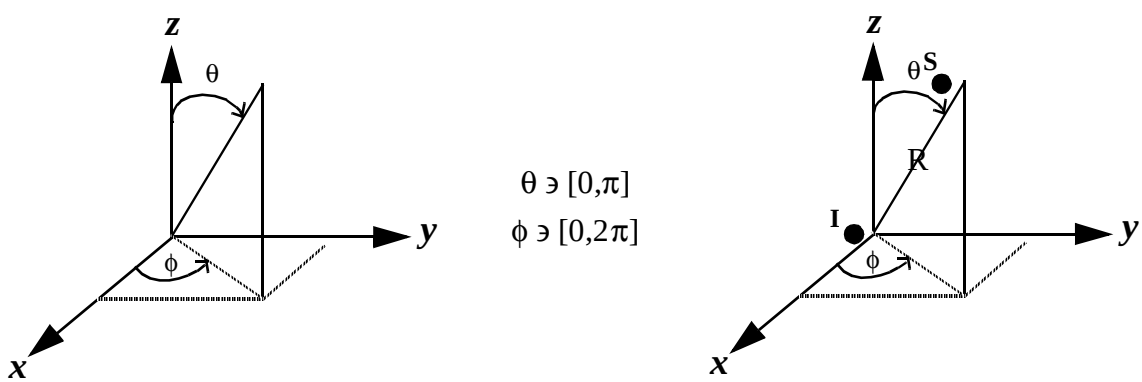


Fig. 3-1 **The right handed Cartesian axes with the spherical angles and radius. For the treatment of dipole-dipole interactions "spins" I and S have been placed on the right hand figure and the radius between them labeled a R.**

3.19.3 Internal Structure

The internal structure of class ***IntDip*** contains the quantities listed in the following table (names shown are also internal).

Table 3-1: Internal Structure of Class IntDip

Name	Description	Type	Name	Description	Type
Inherited From Class IntRank2			**Inherited From Class IntRank2T**		
DELZZ	Spatial Tensor δ_{zz}	double	Tsph	Spin Tensor Values	matrix*
I	Spin Quantum Number	double	Ival	Spin I Hilbert Space	int
S	Spin Quantum Number	double	Sval	Spin S Hilbert Space	int
Inherited From Class IntRank2A					
Asph	Spatial Tensor Values	complex*	THETA	Orientation Angle θ	double
ETA	Spatial Tensor η	double	PHI	Orientation Angle ϕ	double

Fig. 3-2 **Depiction of class IntDip contents, i.e. what each GAMMA defined dipolar interaction contains. All values are inherited from the base class IntRank2 which is derived from classes IntRank2T and IntRank2A. Array Tsph will contain 5 matrices which dimension will be (2*I+1)(2*S+1) and Asph will contain 5 complex numbers. The value of η is normally zero for a dipolar interaction (but left in for ad-hoc computations of dipolar tensors in exchange!).**

The values of ***I*** and ***S*** are the spin quantum number of the two spins involved in the dipole-dipole interaction and both will have values which are positive non-zero integer multiples of 1/2. These dictate how many energy levels (and transitions) are associated with the dipolar interaction. These are intrinsically tied into the values and dimensions of the matrices in the vector ***Tsph*** as well as the spin Hilbert space values of ***Ival*** and ***Sval***.

The value ***DELZZ*** is used to specify the dipolar interaction strength. In GAMMA this is factored out of the spatial tensor such that all rank two interactions (such as the dipolar interaction) have the same spatial tensor scaling. For dipolar interacton ***DELZZ***=DCC and is maintained in Hz.

The two angles ***THETA*** and ***PHI*** indicate how the dipolar interaction (internuclear vector) is aligned relative to the interaction principal axes (PAS). These are one in the same as the angles shown in Fig. 3-1 when the Cartesian axes are those of the PAS with the origin vaguely being the center of the nucleus I. These are intrinsically tied into the values in the array ***Asph***.

The asymmetry ***ETA*** indicates how the diolar interaction varies with the angle phi. Normally this is zero because the interaction is symmetric about the internuclear vector. However, users may set this to a nonzero value in cases where the dipolar vector is modulated (due to exchange for example) and it may be advantageous to use a single averaged diipole.

There are five values in the complex vector ***Asph*** and these are irreducible spherical components of the dipolar spatial tensor oriented at angle ***THETA*** down from the PAS z-axis and over angle ***PHI*** from the PAS x-axis. Note that these 5 values are not only orientation dependent, they are also ***ETA*** dependent. If either of the three the interaction values {***ETA***, ***THETA, PHI***} are altered these components will all be reconstructed. The values in ***Asph*** will be scaled such that they are consistent with other rank 2 spatial tensors in GAMMA which are independent of the interaction type.

The vector of matrices relates to the sperical spin tensor components according to:

Tsph:	[0]	[1]	[2]	[3]	[4]
$T^D_{2,m}$:	$T^D_{2,0}$	$T^D_{2,1}$	$T^D_{2,-1}$	$T^D_{2,2}$	$T^D_{2,-2}$

and the vector of complex numbers relate to the spherical spatial tensor components via

Asph:	[0]	[1]	[2]	[3]	[4]
$A_{2,m}$:	$A_{2,0}$	$A_{2,1}$	$A_{2,-1}$	$A_{2,2}$	$A_{2,-2}$

3.19.4 Classical Dipole-Dipole Treatment

The classical interaction energy between two dipoles, $\vec{\mu}_i$ and $\vec{\mu}_j$, separated by a distance r is[1]

$$E^D_{i,j} = \frac{\vec{\mu}_i \bullet \vec{\mu}_j}{r^3_{ij}} + 3\frac{(\vec{\mu}_i \bullet \vec{r}_{ij})(\vec{\mu}_j \bullet \vec{r}_{ij})}{r^5_{ij}}$$

where $\vec{\mu}$ is the magnetic moment, i and j spin indices, E the energy, and $\vec{r}_{ij}$ the vector connecting the two spins. The superscript D is used to denote a dipolar interaction.

3.19.5 Quantum Mechanical Formulation

The associated Hamiltonian is obtained from substitution of $h\gamma\vec{I}_i$ for $\vec{\mu}_i$ (here $h = 2\pi h$).

$$\boldsymbol{H}^D_{i,j} = \frac{h^2\gamma_i\gamma_j}{r^3_{ij}}\left[\vec{I}_i \bullet \vec{I}_j - \frac{3}{r^2_{ij}}(\vec{I}_i \bullet \vec{r}_{ij})(\vec{I}_j \bullet \vec{r}_{ij})\right]$$

Using normalized unit vectors pointing in the direction of $\vec{r}_{ij}$, $\vec{e}_{ij} = \vec{r}_{ij}/r_{ij}$, the equation becomes

$$\boldsymbol{H}^D_{i,j} = \frac{h^2\gamma_i\gamma_j}{r^3_{ij}}[\vec{I}_i \bullet \vec{I}_j - 3(\vec{I}_i \bullet \vec{e}_{ij})(\vec{I}_j \bullet \vec{e}_{ij})] = \frac{h^2\gamma_i\gamma_j}{r^3_{ij}}[\vec{I}_i \bullet \mathbf{1} \bullet \vec{I}_j - 3(\vec{I}_i \bullet \hat{\boldsymbol{E}}_{ij} \bullet \vec{I}_j)]$$

where $\hat{\boldsymbol{E}}_{ij}$ is the matrix formed from the dyadic product of the two $\vec{e}_{ij}$ unit vectors. A dipolar tensor $\hat{\boldsymbol{D}}_{ij}$ between the two spins can be defined as

$$\hat{\boldsymbol{D}}_{ij} = \mathbf{1} - 3\hat{\boldsymbol{E}}_{ij} \tag{1}$$

and thus the dipolar Hamiltonian for a spin pair i & j, $\boldsymbol{H}^D_{i,j}$, given by

$$\boldsymbol{H}^D_{i,j} = \frac{h^2\gamma_i\gamma_j}{r^3_{ij}}[\vec{I}_i \bullet \hat{\boldsymbol{D}}_{ij} \bullet \vec{I}_j]$$

where $\vec{I}_i$ is the spin angular momentum operator of spin i and $\hat{\boldsymbol{D}}_{ij}$ the dipolar tensor between the two spins. In expanded matrix form this equation looks like

$$\boldsymbol{H}^D_{i,j} = \frac{h^2\gamma_i\gamma_j}{r^3_{ij}}\begin{bmatrix}\boldsymbol{I}_{ix} & \boldsymbol{I}_{iy} & \boldsymbol{I}_{iz}\end{bmatrix} \bullet \begin{bmatrix} D_{xx} & D_{xy} & D_{xz} \\ D_{yx} & D_{yy} & D_{yz} \\ D_{zx} & D_{zy} & D_{zz}\end{bmatrix}_{ij} \bullet \begin{bmatrix}\boldsymbol{I}_{jx} \\ \boldsymbol{I}_{jy} \\ \boldsymbol{I}_{jz}\end{bmatrix} \tag{2}$$

An equivalent equation explicitly showing the matrix multiplication is (with $u, v \in \{x, y, z\}$)

$$\boldsymbol{H}^D_{i,j} = \frac{h^2\gamma_i\gamma_j}{r^3_{ij}}\sum_{u}^{axes}\sum_{v}^{axes}\langle 1|\vec{I}_i|u\rangle\langle u|\hat{\boldsymbol{D}}_{ij}|v\rangle\langle v|\vec{I}_j|1\rangle . \tag{3}$$

1. See Slichter, page 66, equation (3.2).

3.19.6 Cartesian Tensor Formulation

Equation (3) can be rearranged to produce an equation involving two rank 2 Cartesian tensors by taking the dyadic product of the vectors $\boldsymbol{I}_i$ and $\boldsymbol{I}_j$.

$$\boldsymbol{H}_{i,j}^D = \frac{h^2\gamma_i\gamma_j}{r_{ij}^3}\sum_u^{axes}\sum_v^{axes}\langle u|\hat{D}_{ij}|v\rangle\langle v|\hat{\boldsymbol{I}}_j|1\rangle\langle 1|\hat{\boldsymbol{I}}_i|u\rangle = \frac{h^2\gamma_i\gamma_j}{r_{ij}^3}\sum_u^{axes}\sum_v^{axes}\langle u|\hat{D}_{ij}|v\rangle\langle v|\hat{\boldsymbol{I}}_j\hat{\boldsymbol{I}}_i|u\rangle$$

The dyadic product to produce $\hat{\boldsymbol{I}}_j\hat{\boldsymbol{I}}_i$ is explicitly done *via*

$$\begin{bmatrix}\boldsymbol{I}_{jx}\\ \boldsymbol{I}_{jy}\\ \boldsymbol{I}_{jz}\end{bmatrix}\bullet\begin{bmatrix}\boldsymbol{I}_{ix} & \boldsymbol{I}_{iy} & \boldsymbol{I}_{iz}\end{bmatrix} = \begin{bmatrix}\boldsymbol{I}_{jx}\boldsymbol{I}_{xi} & \boldsymbol{I}_{jx}\boldsymbol{I}_{yi} & \boldsymbol{I}_{jx}\boldsymbol{I}_{zi}\\ \boldsymbol{I}_{jy}\boldsymbol{I}_{xi} & \boldsymbol{I}_{jy}\boldsymbol{I}_{yi} & \boldsymbol{I}_{jy}\boldsymbol{I}_{zi}\\ \boldsymbol{I}_{jz}\boldsymbol{I}_{xi} & \boldsymbol{I}_{jz}\boldsymbol{I}_{yi} & \boldsymbol{I}_{jz}\boldsymbol{I}_{zi}\end{bmatrix}.$$

and from Equation (1) the matrix $\hat{D}_{ij}$ in the principle axis system ($\hat{\boldsymbol{e}}_{ij} = \hat{k}$) given by

$$\hat{D}_{ij}\Big|_{PAS} = 1 - 3\hat{E}_{ij}\Big|_{PAS} = \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix} - 3\begin{bmatrix}0\\ 0\\ 1\end{bmatrix}\bullet\begin{bmatrix}0 & 0 & 1\end{bmatrix} = \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & -2\end{bmatrix}. \tag{4}$$

Letting $\hat{\boldsymbol{T}}_{ij} = \hat{\boldsymbol{I}}_j\hat{\boldsymbol{I}}_i$, the Hamiltonian is expressed as a scalar product of two rank 2 Cartesian tensors.

$$\boldsymbol{H}_{i,j}^D = \frac{h^2\gamma_i\gamma_j}{r_{ij}^3}\hat{D}_{ij}\bullet\hat{\boldsymbol{T}}_{ij}$$

or equivalently (4-1)

$$\boldsymbol{H}_{i,j}^D = \frac{h^2\gamma_i\gamma_j}{r_{ij}^3}\sum_u^{axes}\sum_v^{axes}\langle u|\hat{D}_{ij}|v\rangle\langle v|\hat{\boldsymbol{T}}_{ij}|u\rangle$$

3.19.7 Spherical Tensor Formulation

Equation (4-1) can be rewritten in terms of irreducible spherical components rather than the current Cartesian components[1] using the substitution

$$\sum_{l=0}^{2}\sum_m^{\pm l}(-1)^m A_{l-m}\hat{\boldsymbol{T}}_{lm}^D = \sum_u^{axes}\sum_v^{axes}\langle u|D|v\rangle\langle v|\hat{\boldsymbol{T}}^D|u\rangle \tag{5}$$

1. The purpose of this step is to place $\boldsymbol{H}_{i,j}^D$ in a format which facilitates rotations on its coordinate system. For a more detailed explanation see the description in Class Spin Tensor.

The result is

$$\boldsymbol{H}^D_{i,j} = \frac{h^2\gamma_i\gamma_j}{r^3_{ij}} \sum_{l=0}^{2} \sum_{m}^{\pm l} (-1)^m D_{lm}(ij) \boldsymbol{T}^D_{l-m}(ij) \tag{6}$$

3.19.8 Dipole-Dipole Spherical Tensor Spin Components

We can thus obtain the 9 irreducible spherical components of the dipolar spin tensor (rank 2), $\boldsymbol{T}^D_{l-m}(ij)$, directly from the Cartesian components, $\langle v|\boldsymbol{T}_{ij}|u\rangle$, as indicated in GAMMA Class Documentation on Spin Tensors. The nomenclature used here for a tensor component is

$$\boldsymbol{T}_{l,m},$$

where the subscript *l* spans the rank (in this case 2) as *l* = [0, 2], and the subscript *m* spans +/- *l*, *m* = [-*l*, *l*]. The nine formulas for these quantities a listed in the following figure.

Dipolar Irreducible Spherical Spin Tensor Components

$$T^D_{0,0}(ij) = \frac{-1}{\sqrt{3}}\left[I_{iz}I_{jz} + \frac{1}{2}(I_{i+}I_{j-} + I_{i-}I_{j+})\right]$$

$$T^D_{1,0}(ij) = \frac{-1}{2\sqrt{2}}[I_{i+}I_{j-} + I_{i-}I_{j+}] \qquad T^D_{1,\pm1}(ij) = \frac{-1}{2\sqrt{2}}[I_{i\pm}\ I_{jz} + I_{iz}I_{j\pm}\ ]$$

$$T^D_{2,0}(ij) = \frac{1}{\sqrt{6}}[3I_{iz}I_{jz} - (\boldsymbol{I}_i \bullet \boldsymbol{I}_j)]$$

$$T^D_{2,\pm1}(ij) = \mp\frac{1}{2}[I_{i\pm}\ I_{jz} + I_{iz}I_{j\pm}\ ] \qquad T^D_{2,\pm2}(ij) = \frac{1}{2}[I_{i\pm}\ I_{j\pm}\ ]$$

Figure 3-3 **The rank 2 spin tensor components for a dipolar interaction.**

The matrix representation of these nine tensor components will depend upon the matrix representations of the individual spin operators from which they are constructed[1]. These in turn depend upon the spin quantum numbers of the two spins involved. For a treatment of two spin 1/2 particles the dipolar tensor components are expressed in their matrix form (spanning the composite Hilbert space of the two spins) in the default product basis of GAMMA as follows[2] (spin indices implicit).

1. Note that the spin tensors are invariably constructed in the laboratory coordinate system. Here the z-axis corresponds to the direction of the spectrometer static magnetic field and the coordinate system is right-handed.
2. The GAMMA program DipSpinT.cc on page 118 generated these matrices.

Dipolar Spin Tensor Component Matrix Representations

$$T^D_{0,0} = \frac{-1}{4\sqrt{3}}\begin{bmatrix}1&0&0&0\\0&-1&2&0\\0&2&-1&0\\0&0&0&1\end{bmatrix} \qquad T^D_{1,0} = \frac{1}{2\sqrt{2}}\begin{bmatrix}0&0&0&0\\0&0&-1&0\\0&1&0&0\\0&0&0&0\end{bmatrix} \qquad T^D_{1,-1} = \frac{1}{4}\begin{bmatrix}0&0&0&0\\1&0&0&0\\-1&0&0&0\\0&1&-1&0\end{bmatrix} \qquad T^D_{1,1} = \frac{1}{4}\begin{bmatrix}0&1&-1&0\\0&0&0&1\\0&0&0&-1\\0&0&0&0\end{bmatrix}$$

$$T^D_{2,0} = \frac{1}{2\sqrt{6}}\begin{bmatrix}1&0&0&0\\0&-1&-1&0\\0&-1&-1&0\\0&0&0&1\end{bmatrix} \quad T^D_{2,1} = \frac{1}{4}\begin{bmatrix}0&-1&-1&0\\0&0&0&1\\0&0&0&1\\0&0&0&0\end{bmatrix} \quad T^D_{2,-1} = \frac{1}{4}\begin{bmatrix}0&0&0&0\\1&0&0&0\\1&0&0&0\\0&-1&-1&0\end{bmatrix} \quad T^D_{2,-2} = \frac{1}{2}\begin{bmatrix}0&0&0&0\\0&0&0&0\\0&0&0&0\\1&0&0&0\end{bmatrix} \quad T^D_{2,2} = \frac{1}{2}\begin{bmatrix}0&0&0&1\\0&0&0&0\\0&0&0&0\\0&0&0&0\end{bmatrix}$$

Figure 3-4 **Rank 2 spin tensor components for a dipolar interaction between two spin 1/2 nuclei.**

3.19.9 Dipole-Dipole Spherical Spatial Tensor Components

The 9 irreducible spherical components of a rank two spatial tensor, $A^{(2)}_{lm}$, are related to its Cartesian components by the following formulas[1].

$$\begin{gathered}
A_{0,0} = \frac{-1}{\sqrt{3}}[A_{xx}+A_{yy}+A_{zz}] = \frac{-1}{\sqrt{3}}Tr\{A\} \\
A_{1,0} = \frac{-i}{\sqrt{2}}[A_{xy}-A_{yx}] \qquad A_{1,\pm1} = \frac{-1}{2}[A_{zx}-A_{xz}\pm i(A_{zy}-A_{yz})] \\
A_{2,0} = \sqrt{6}[3A_{zz}-(A_{xx}+A_{yy}+A_{zz})] = \sqrt{6}[3A_{zz}-Tr\{A\}] \\
A_{2,\pm1} = \mp\frac{1}{2}[A_{xz}+A_{zx}\pm i(A_{yz}+A_{zy})] \qquad A_{2,\pm2} = \frac{1}{2}[A_{xx}-A_{yy}\pm i(A_{xy}+A_{yx})]
\end{gathered} \tag{7}$$

Again the subscript l spans the rank as l = [0, 2], and the subscript m spans +/- l, m = [-l, l]. In this dipolar treatment we then have components $D_{l,m}(ij)$ as indicated in equation (6). Thus, the irreducible spherical tensor components can be obtained by substituting the Cartesian elements of the dipolar spatial tensor, $\hat{D}$ from equation (2), into equations (7).

$$\begin{gathered}
D_{0,0} = \frac{-1}{\sqrt{3}}[D_{xx}+D_{yy}+D_{zz}] = \frac{-1}{\sqrt{3}}Tr\{\hat{D}\} \\
D_{1,0} = \frac{-i}{\sqrt{2}}[D_{xy}-D_{yx}] \qquad D_{1,\pm1} = \frac{-1}{2}[D_{zx}-D_{xz}\pm i(D_{zy}-D_{yz})] \\
D_{2,0} = \sqrt{6}[3D_{zz}-(D_{xx}+D_{yy}+D_{zz})] = \sqrt{6}[3D_{zz}-Tr\{\hat{D}\}] \\
D_{2,\pm1} = \mp\frac{1}{2}[D_{xz}+D_{zx}\pm i(D_{yz}+D_{zy})] \qquad D_{2,\pm2} = \frac{1}{2}[D_{xx}-D_{yy}\pm i(D_{xy}+D_{yx})]
\end{gathered} \tag{8}$$

However, it is more convenient to rewrite the general rank two Cartesian tensor in terms of a sum

1. See the GAMMA Class Documentaion on Rank 2 Interactions.

over tensors of ranks 0 through 2 as follows,

$$\hat{A} = \begin{bmatrix} A_{xx} & A_{xy} & A_{xz} \\ A_{yx} & A_{yy} & A_{yz} \\ A_{zx} & A_{zy} & A_{zz} \end{bmatrix} = \underbrace{A_{iso}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Rank 0}} + \underbrace{\begin{bmatrix} 0 & \alpha_{xy} & \alpha_{xz} \\ -\alpha_{xy} & 0 & \alpha_{yz} \\ -\alpha_{xz} & -\alpha_{yz} & 0 \end{bmatrix}}_{\text{Rank 1}} + \underbrace{\begin{bmatrix} \delta_{xx} & \delta_{xy} & \delta_{xz} \\ \delta_{yx} & \delta_{yy} & \delta_{yz} \\ \delta_{zx} & \delta_{zy} & \delta_{zz} \end{bmatrix}}_{\text{Rank 2}} \tag{9}$$

where

$$A_{iso} = \frac{1}{3}Tr\{\hat{A}\} \qquad \alpha_{xy} = \frac{1}{2}(A_{xy} - A_{yx}) \qquad \delta_{xy} = \frac{1}{2}(A_{xy} + A_{yx} - 2A_{iso}) \tag{10}$$

The rank 0 part is isotropic (scalar), the rank 1 part is antisymmetric and traceless, and the rank 2 part traceless and symmetric. We shall apply this same nomeclature to our dipolar spatial tensor in order to produce

$$\hat{D} = \begin{bmatrix} D_{xx} & D_{xy} & D_{xz} \\ D_{yx} & D_{yy} & D_{yz} \\ D_{zx} & D_{zy} & D_{zz} \end{bmatrix} = D_{iso}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & \alpha_{xy} & \alpha_{xz} \\ -\alpha_{xy} & 0 & \alpha_{yz} \\ -\alpha_{xz} & -\alpha_{yz} & 0 \end{bmatrix} + \begin{bmatrix} \delta_{xx} & \delta_{xy} & \delta_{xz} \\ \delta_{yx} & \delta_{yy} & \delta_{yz} \\ \delta_{zx} & \delta_{zy} & \delta_{zz} \end{bmatrix}. \tag{11}$$

where

$$D_{iso} = \frac{1}{3}Tr\{\hat{D}\} \qquad \alpha_{xy} = \frac{1}{2}(D_{xy} - D_{yx}) \qquad \delta_{xy} = \frac{1}{2}(D_{xy} + D_{yx} - 2D_{iso}) \tag{12}$$

Note that δ_{xy} is for most spatial tensors is NOT equivalent to the unscaled spatial tensor xy-component (D_{xy} here) *unless there is no isotropic component*. That turns out to be the case for the dipolar interacation as seen from Equation (4). As with any rank 2 spatial tensor, the dipolar spatial tensor can be specified in its principal axis system(PAS), the set of axes in which the irreducible rank 2 component is diagonal[1,2]. The spatial tensor values are experimentally determined in the tensor principal axes. Employing (9) when the irreducible rank 2 component is diagonal,

$$\hat{D}(PAS) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -2 \end{bmatrix} \qquad Tr\{\hat{D}(PAS)\} = 0 \qquad D_{iso} = \frac{1}{3}Tr\{\hat{D}\} = 0$$

1. The principal axis system is set such that $|\delta_{zz}| \geq |\delta_{yy}| \geq |\delta_{xx}|$. The orientation of the x and y axes are inconsequential if η is zero.
2. The dipolar principal axis system for a spin pair has the z-axis pointing along the vector connecting the two spins. The orientation of the x and y axes are inconsequential due to the cylindrical symmetry of the interaction about the dipole vector (PAS z-axis).

Indeed the anti-symmetric components are also all zero since $D_{xy}(PAS)\big|_{x \neq y} = 0$. So we can rewrite out dipolar spatial tensor in the principle axis system as

$$D(\hat{PAS}) = \begin{bmatrix} D_{xx} & D_{xy} & D_{xz} \\ D_{yx} & D_{yy} & D_{yz} \\ D_{zx} & D_{zy} & D_{zz} \end{bmatrix}\Bigg|_{PAS} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -2 \end{bmatrix} = \begin{bmatrix} \delta_{xx} & 0 & 0 \\ 0 & \delta_{yy} & 0 \\ 0 & 0 & \delta_{zz} \end{bmatrix} . \tag{13}$$

where $D_{iso} = 0$ and $\alpha_{xy} = 0$.

3.19.10 Unscaled Spherical Spatial Tensor PAS Components

Rank 2 spatial tensors are also commonly specified in their principal axis system by the three components; the isotropic value A_{iso}, the anisotropy ΔA, and the asymmetry η. These are generally given by

$$A_{iso} = \frac{1}{3}Tr\{A\} , \qquad \Delta A = A_{zz} - \frac{1}{2}(A_{xx} + A_{yy}) = \frac{3}{2}\delta_{zz} \qquad \eta = (\delta_{xx} - \delta_{yy})/\delta_{zz}$$

A set of Euler angles $\{\alpha, \beta, \gamma\}$ is normally also given to relate the spatial tensor principle axes to another coordinate system. For the dipolar spatial tensor we have

$$\hat{D}(PAS) = \overset{0}{D_{iso}} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & \overset{0}{\alpha_{xy}} & \alpha_{xz} \\ -\alpha_{xy} & 0 & \alpha_{yz} \\ -\alpha_{xz} & -\alpha_{yz} & 0 \end{bmatrix} + \delta_{zz} \begin{bmatrix} -\frac{1}{2}(1-\eta) & 0 & 0 \\ 0 & -\frac{1}{2}(1+\eta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{14}$$

We have already seen that both the isotropic and anti-symmetric terms are zero for the dipolar interacation. Again, we note that δ_{zz} is for most spatial tensors is NOT equivalent to the unscaled spatial tensor z-component (D_{zz} here) and that η is usually NOT equivalent to unscaled ratio (here $(D_{xx} - D_{yy})/D_{zz}$) *unless there is no isotropic component*. But we have also seen that there is no isotropic component in the dipolar spatial tensor.

$$D_{iso} = 0, \qquad \Delta D = \frac{3}{2}\delta_{zz} = \frac{3}{2}D_{zz} \qquad \eta = (\delta_{xx} - \delta_{yy})/\delta_{zz} = (D_{xx} - D_{yy})/D_{zz}$$

We can also figure out the asymmetry value from the PAS representation of the tensor. Again from The irreducible spherical elements of the dipolar tensor, $D_{l,m}$, in the principal axis system are, by placement of (14) into (7),

$$D_{0,0}(PAS) = -\sqrt{3}D_{iso} = 0$$

$$D_{1,0}(PAS) = -\frac{i}{\sqrt{2}}[D_{xy} - D_{yx}] = 0 \qquad D_{1,\pm1}(PAS) = -\frac{1}{2}[(D_{zx} - D_{xz}) \pm i(D_{zy} - D_{yz})] = 0$$

$$D_{2,0}(PAS) = \sqrt{3/2}\delta_{zz} = \sqrt{3/2}D_{zz} \qquad D_{2,1}(PAS) = D_{2,-1}(PAS) = 0$$

$$D_{2,2}(PAS) = D_{2,-2}(PAS) = \frac{1}{2}\delta_{zz}\eta = \frac{1}{2}D_{zz}\eta$$

and these values should be equivalent to those given in (8) on page 3-103. Fortunately, there is no isotropic component of the dipolar tensor, nor is there any asymmetry. If we then use these fact we obtain a much simpler result

$$\begin{gathered} D_{0,0}(ij, PAS) = 0 \\ D_{1,0}(ij, PAS) = 0 \qquad D_{1,\pm1}(ij, PAS) = 0 \\ D_{2,0}(ij, PAS) = -\sqrt{6} \qquad D_{2,\pm1}(ij, PAS) = 0 \qquad D_{2,\pm2}(ij, PAS) = 0 \end{gathered} \tag{15}$$

All but one of the spherical components is zero because the dipolar spatial tensor is symmetric and traceless.

3.19.11 Scaled Dipolar Spherical Spatial Tensor PAS Components

Throughout GAMMA we desire all spatial components to be scaled so that they are independent of the particular interaction. To do so, we adjust them to be as similar to normalized spherical harmonics as possible. Thus, we here scale the dipolar spatial tensor such that the $2, 0$ component (the only non-zero one in this instance) will have a magnitude of the $m = 0$ rank two spherical harmonic when the two spherical angles are set to zero. Our "normalization" factor "X" is obtained by

$$A_{2,0}(\theta,\varphi)\big|_{\theta=\varphi=0} = X^D \bullet D_{2,0}(\theta,\varphi)\big|_{\theta=\varphi=0} = Y_{2,0}(\theta,\varphi)\big|_{\theta=\varphi=0} = \sqrt{5/(4\pi)} \tag{16}$$

Using $D_{2,0}(PAS) = \sqrt{3/2}\delta_{zz} = -\sqrt{6}$ we define the GAMMA dipolar spatial tensor such that

$$A_{l,m} = \sqrt{5/(6\pi)}\delta_{zz}^{-1}D_{l,m}^{D} = -\sqrt{5/(24\pi)}D_{l,m}^{D} \tag{17}$$

and the components are given in the next figure.

GAMMA Normalized Dipolar Spatial Tensor PAS Components

$$A_{2,0}(PAS) = \sqrt{\frac{5}{4\pi}} \qquad A_{2,\pm1}(PAS) = 0 \qquad A_{2,\pm2}(PAS) = \sqrt{\frac{5}{24\pi}}\eta = 0$$

Figure 3-5 **Generic irreducible rank 2 spatial tensor components as defined in GAMMA. These are shown in the principle axis system of the tensor and scaled to coincide with normalized spherical harmonics. Since the dipolar interaction is symmetric about the axis connecting the two spins, the interaction assymetry η is zero.**

The scaling factor $\sqrt{5/(6\pi)}\delta_{zz}^{-1} = -\sqrt{5/(24\pi)}$ which was multiplied into the "D" components will be compensated for in the dipolar interaction constant. The dipole-dipole Hamiltonian given in equation (6) becomes

$$\boldsymbol{H}_{i,j}^{D} = \frac{h^2\gamma_i\gamma_j}{r_{ij}^3}\sum_{m}^{\pm2}(-1)^m\hat{D}_{2-m}(ij)\hat{\boldsymbol{T}}_{2m}^{D}(ij) = -\sqrt{\frac{24\pi}{5}}\frac{h^2\gamma_i\gamma_j}{r_{ij}^3}\sum_{m}^{\pm2}(-1)^m A_{lm}(ij)\boldsymbol{T}_{l-m}^{D}(ij) \tag{18}$$

3.19.12 Dipolar Interaction Constant

In GAMMA, since we have defined our generic spatial and spin tensors to be scaled independent of the type of interaction, we use an interaction constant as a scaling factor when formulating Hamiltonians. The dipolar Hamiltonian may be produced from

$$\boldsymbol{H}_{ij}^{D} = \xi_{ij}^{D}\sum_{m}^{\pm2}(-1)^m\hat{A}_{2-m}(ij)\hat{\boldsymbol{T}}_{2m}^{D}(ij) = -\sqrt{\frac{24\pi}{5}}\frac{h^2\gamma_i\gamma_j}{r_{ij}^3}\sum_{m}^{\pm1}(-1)^m A_{2,-m}(ij)\hat{\boldsymbol{T}}_{2m}^{D}(ij) \tag{19}$$

so evidently

$$\xi_{ij}^{D} = -2\sqrt{\frac{6\pi}{5}}\frac{h^2\gamma_i\gamma_j}{r_{ij}^3} = -2\sqrt{\frac{6\pi}{5}}\delta_{zz} = -2\sqrt{\frac{6\pi}{5}}DCC \tag{20}$$

Such interaction constants are not very common in the literature (except with regards to some papers treating relaxation in liquid NMR) and thus not intuitive to many GAMMA users. So, one simply needs to be aware of the relationships between the interaction constant and any commonly used dipolar tensor definitions. Many treatments retain the dipolar tensor in Cartesian components, whereas in GAMMA we (internally) work with the spherical components consistently across the magnetic resonance interaction types. Perhaps the only quantity worthy of mention is the δ_{zz}, the dipolar anisotropy . This is readily related to the typical D tensor Cartesian components.

$$\delta_{zz} = D_{zz} - D_{iso} = D_{zz} - \frac{1}{3}Tr\{\hat{D}\} = D_{zz}$$

3.19.13 Spatial Tensor Rotations

We can express the dipolar spatial tensor components relative to any arbitrary axis system (AAS) by rotating the tensor from the principal axes to the new axes *via* the formula

$$A_{l,\,m}(ij, AAS) = \sum_{m'}^{\pm l} D_{mm'}^{l}(\Omega)A_{l,\,m'}(ij, PAS) \tag{21}$$

where $D_{mm'}^{l}$ are the rank l Wigner rotation matrix elements and Ω the set of three Euler angles which relate the principal axes to the arbitrary axes. As is evident from equations (18-15) - (21), regardless of the coordinate system, *only the rank two components will contribute to the dipolar Hamiltonian*. This is now demonstrated by combining the last two equations.

$$A_{0,\,0}(ij, AAS) = A_{1,\,0}(ij, AAS) = A_{1,\,\pm 1}(ij, AAS) = 0 \tag{22}$$

$$A_{2,\,m}(ij, AAS) = \sum_{m'}^{\pm 2} D_{mm'}^{l}(\Omega)A_{2,\,m'}(ij, PAS) = D_{m0}^{2}(\Omega)A_{2,\,0}(ij, PAS)$$

Using now the Wigner rotation matrix element relationship

$$D_{m0}^{l}(\Omega) = D_{m0}^{l}(\varphi, \theta, \chi) = \sqrt{\frac{4\pi}{2l+1}}Y_{lm}(\theta, \varphi) \tag{23}$$

we have

$$A_{2,\,\pm 1}(ij, AAS) = \sqrt{\frac{4\pi}{5}}Y_{2m}(\theta, \varphi)A_{2,\,0}(ij, PAS) = \sqrt{\frac{4\pi}{5}}Y_{2m}(\theta, \varphi)\sqrt{\frac{5}{4\pi}} = Y_{2m}(\theta, \varphi) \tag{24}$$

3.19.14 Dipolar Hamiltonian

Because only the rank 2 dipolar spatial tensors will be non-zero, it is sufficient to utilize only the irreducible spherical rank 2 components of both the spatial and spin dipolar tensors in construction of the dipolar Hamiltonian. Equation (18-15) can be rewritten as

$$\boldsymbol{H}^D_{i,j} = \xi^D_{ij} \sum_m^{\pm 2} (-1)^m A_{2-m}(ij) \boldsymbol{T}^D_{2m}(ij) \tag{25}$$

where

$$\xi^D_{ij} = -2\sqrt{\frac{6\pi}{5}} \frac{h^2 \gamma_i \gamma_j}{r^3_{ij}} \qquad A_{2,m}(ij, PAS) = \delta_{m0}\sqrt{\frac{5}{4\pi}} \qquad A_{2,m}(ij, AAS) = Y_{2m}(\theta, \varphi) \tag{26}$$

When working with an entire spin system one must sum over all spin pairs with the spatial tensors being placed in the same coordinate system, usually the laboratory system. The dipolar Hamiltonian for a spin system becomes the following.

$$\boldsymbol{H}^D = \sum_i^{spins} \sum_j^{spins} \xi^D_{ij} \sum_{m=-2}^{2} (-1)^m Y^D_{2-m}(\theta_{ij}, \phi_{ij}) \boldsymbol{T}^D_{2m}(ij) \tag{27}$$

Here the angles θ_{ij} and ϕ_{ij} are the polar angles of the dipolar vector between spins i and j when written relative to the coordinate system in which $\boldsymbol{H}^D$ is expressed. $Y_{2m}(\theta, \phi)$ are the normalized rank two spherical harmonics, the superscript D indicating the dipolar interaction is unnecessary but used for consistency with other Hamiltonian definitions.

3.19.15 Dipolar Orientation Angles

A set of Euler angles $\{\alpha, \beta, \gamma\}$ is normally given to relate the spatial tensor principle axes to another coordinate system. In the dipolar Hamiltonian derivation we have instead used $\{\varphi, \theta, \chi\}$ for the Euler angle designations because, due to the dipolar tensor symmetry, we ultimately utilize only the angles $\{\varphi, \theta\}$ which are equivalent to the common angle designations in spherical coordinates. In turn the spherical harmonics are written in these coordinates.

For the dipolar spatial tensor we have

$$A_{iso} = 0 \qquad \Delta^D = \sqrt{15/(8\pi)} \qquad \delta_{zz}^D = \sqrt{5/(6\pi)} \qquad \eta^D = 0$$

This is perhaps more easily seen visually by examination of the matrix breakdown into these components (superscript D has been added so there is an association with the dipolar spatial tensor).

$$\hat{A}(PAS) = -\sqrt{\frac{5}{24\pi}}\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & -2\end{bmatrix} = \sqrt{\frac{5}{6\pi}}\begin{bmatrix}-\frac{1}{2} & 0 & 0\\ 0 & -\frac{1}{2} & 0\\ 0 & 0 & 1\end{bmatrix} = \cancelto{0}{D_{iso}}\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix} + \delta_{zz}^D\begin{bmatrix}-\frac{1}{2}(1-\eta^D) & 0 & 0\\ 0 & -\frac{1}{2}(1+\eta^D) & 0\\ 0 & 0 & 1\end{bmatrix}$$

The following figure contains a grouping of the applicable dipolar Hamiltonian equations.

3.19.16 Summary

The Dipolar Hamiltonian Summary

General System

$$\boldsymbol{H}^D = \sum_i^{spins} \sum_j^{spins} \xi_{ij}^D \sum_{m=-2}^{2} (-1)^m Y_{2-m}^D(\theta_{ij}, \phi_{ij}) \boldsymbol{T}_{2m}^D(ij)$$

$$\xi_{ij}^D = -2\sqrt{\frac{6\pi}{5}} \frac{h^2 \gamma_i \gamma_j}{r_{ij}^3}$$

$$Y_{2,0}(\theta, \phi) = \sqrt{\frac{5}{16\pi}}(3\cos^2\theta - 1) \qquad T_{2,0}^D(ij) = \frac{1}{\sqrt{6}}[3I_{iz}I_{jz} - (\boldsymbol{I}_i \bullet \boldsymbol{I}_j)]$$

$$Y_{2,\pm1}(\theta, \phi) = \mp\sqrt{\frac{15}{8\pi}}\cos\theta\sin\theta e^{\pm i\phi} \qquad T_{2,\pm1}^D(ij) = \mp\frac{1}{2}[I_{i\pm}\ I_{jz} + I_{iz}I_{j\pm}\ ]$$

$$Y_{2,\pm2}(\theta, \phi) = \sqrt{\frac{15}{32\pi}}\sin^2\theta e^{\pm 2i\phi} \qquad T_{2,\pm2}^D(ij) = \frac{1}{2}[I_{i\pm}\ I_{j\pm}\ ]$$

A Single Spin Pair

$$\boldsymbol{H}_{i,j}^D = \xi_{ij}^D \sum_m^{\pm 2} (-1)^m A_{2-m}(ij) \boldsymbol{T}_{2m}^D(ij)$$

$$A_{2,m}(ij, PAS) = \delta_{m0}\sqrt{\frac{5}{4\pi}} \qquad A_{2,m}(ij, AAS) = Y_{2m}(\theta, \varphi)$$

$$A_{l,m}(ij, AAS) = \sum_{m'}^{\pm l} D_{mm'}^l(\varphi, \theta, \chi) A_{l,m'}(ij, PAS)$$

Other Spatial Tensor Elements

$$A_{iso} = 0 \qquad \Delta = \sqrt{15/(8\pi)} \qquad \eta = 0$$

Figure 3-6 **Summary of Dipolar Interaction Treatment in GAMMA.**

3.20 Dipolar Interaction Parameters

This section describes how an ASCII file may be constructed that is self readable by a dipolar interaction. The file can be created with any editor and is read with the dipolar interaction member function "read"[1]. It is important to keep in mind the structure of our interaction as given in Section 3.19.3 on page 97. We will need the **set of {I, S, δ_{zz}, θ, ϕ} specified for each dipolar interaction** we wish to create. The first two, **I, S**, are used to set the spin part of the interaction. The latter three are used to set the overall interaction strength and orientation.

Of course, there are several ways of declaring a dipolar interaction (i.e. the dipolar spatial tensor) other than with direct specification of the five values **{I, S, δ_{zz}, θ, ϕ}**. To accomodate different tensor nomenclature (i.e. spherical *versus* Cartesian, oriented *versus* PAS, etc.), GAMMA dipolar interactions will recognize different sets of parameters! These will be described in the following sections, and are divided into two categories: 1.) Parameters associated with spin pair indices and 2.) Parameters associated with an interaction index. The former are typically used when describing dipolar interactions for spins which are part of a spin system whereas the latter are available when dealing with specific interactions.

3.20.1 Available Spin Pair Parameters

Taking the view that each dipolar interaction is associated with a spin pair, specification of any specific interaction demands two indices (one index for each spin). The spin quantum values are specified with an isotope label, whereas there are multiple parameters to indicate the dipolar coupling strength and interaction orientation.

Dipolar Spin Quantum Numbers: Iso

Specification of the dipolar spin quantum numbers **{I, S}** for a spin pair is accomplished with the parameter ***Iso.*** Iso parameters must designate valid spin isotope typs and each Iso parameter requires a single spin index. Although not recommended, dipolar interactions can be set up without specifying **{I, S}** isotope types wherein they will be assigned default values of I=S=1/2.

Dipolar Interaction Spin Quantum Number Parameters

Parameter	Assumed Units	Examples Parameter (Type=1,3) : Value - Statement
Iso(#)	none	Iso(0) (2) : 131Xe - Spin Isotope Type, I=3/2 for 131Xe Iso(1) (2) : 13C - Spin Isotope Type, I=1/2 for 13C

Fig. 3-7 **Iso parameters require (#) appended to the name, where (#) is the spin index. Dipolar interactions that utilize two spin indices during construction will assosciate the spins with two isotopes.**

1. The interaction read functions (e.g. ask_read) will use the parameters and follow the rules herein.

Dipolar Orientation & Strength: DCC, Dtheta, Dphi

Specification of the dipolar interaction strength and orientation **{**δ_{zz}**,** θ, ϕ**}** can be accomplished either with the three parameters DCC, Dtheta and Dphi or with two coorinate parameters, Coord. When the interaction is associated with a spin pair, the parameters DCC, Dtheta, and Dphi must have two spin indices appended to the parameter names and any Coord parameters must have single spin indices appended.

Dipolar Interaction Strength & Orientation Parameters

Parameter	Assumed Units	Examples Parameter (Type=1,3) : Value - Statement
DCC	KHz	DCC(0,1) (1) : 30.7 - Dipolar Coupling, 1st Interaction
Dtheta	Degrees	Dtheta(0,1) (1) : 90.0 - Dipolar angle down from +z
Dphi	Degrees	Dphi(0,1) (1) : 270.0 - Dipolar angle over from +x
Coord(#)	Angstroms	Coord(0) (3) : (0.0, 0.0, 0.0) - Coordinate 1st Spin Coord(1) (3) : (0.0, 0.0, 2.0) - Coordinate 2nd Spin

Fig. 3-8 **Shown are various parameters that may be used to set dipolar interaction strengths and orientations. Parameter type 1 indicates a double precision number parameter. Parameter type 2 indicates a string parameter. Parameter type 3 indicates a coordinate. Generally, one either uses the set DCC, Dtheta, & Dphi (with two indices for the interaction spins involved) or two Coord parameters (each with a single index for one spin of a spin pair).**

3.20.2 Available Interaction Indexed Parameters

In contrast to the previous parameters, dipolar interaction can be specified using a single interaction index. Rather than using two spin indices, the parameters involve a single index for the interaction.

Dipolar Spin Quantum Numbers: DI, DS, Iso

Specification of the dipolar spin quantum numbers **{I, S}** can be accomplished with the two parameters DI & DS or with two Iso parameters. The values of DI and DS must be a positive integer multiple of 1/2 and may contain an interaction index appended to the names if multiple interactions are to be defined in the same file. Iso parameters must designate a valid spin isotope type and Iso parameters cannot be used unless spin indices are supplied as arguments when the dipolar parameters are read. If both {DI, DS} and two Iso values are set in the same file, the DI and DS values will be preferentialy used to set up the dipolar interaction.

Dipolar Interaction Spin Quantum Number Parameters

Parameter	Assumed Units	Examples Parameter (Type=1,3) : Value - Statement
DI	none	DI (1) : 1.5 - Dipolar 1st Spin Quantum Value
DS	none	DS (1) : 0.5 - Dipolar 2nd Spin Quantum Value

Fig. 3-9 **Shown are 3 possible parameters used to set dipolar spin quantum numbers. Parameter type 1 indicates a double precision number parameter. Parameter type 2 indicates a string parameter. If Iso is used then the function call to read it must supply both spin indices! DI and DS can have (#) appended, the number being an interaction index rather than spin indices.**

Although not recommended, dipolar interactions can be set up without specifying **{I, S}** wherein they will be assigned default values of I=S=1/2 .

Dipolar Orientation & Strength: DCC, Dtheta, Dphi

Specification of the dipolar interaction strength and orientation **{**δ_{zz}**,** θ, ϕ**}** can be accomplished either with the three parameters DCC, Dtheta and Dphi or with two coorinate parameters (Coord) in association with spin isotope types (Iso)., Coord. If the former three values are used they can have an interaction indexThe values of DI and DS must be a positive integer multiple of 1/2 and may contain an interaction index appended to the names if multiple interactions are to be defined in the same file. Iso parameters must designate a valid spin isotope type and Iso parameters cannot be used unless spin indices are supplied as arguments when the dipolar parameters are read. If both {DI, DS} and two Iso values are set in the same file, the DI and DS values will be preferentialy used to set up the dipolar interaction.

Dipolar Interaction Spin Quantum Number Parameters

Parameter	Assumed Units	Examples Parameter (Type=1,3) : Value - Statement
DCC	KHz	DCC(0) (1) : 30.7 - Dipolar Coupling, 1st Interaction
Dtheta	Degrees	Dtheta(1) (1) : 90.0 - Dipolar angle down from +z
Dphi	Degrees	Dphi(1) (1) : 270.0 - Dipolar angle over from +x

Fig. 3-10 **Shown are various parameters that may be used to set dipolar interaction strengths and orientations. Parameter type 1 indicates a double precision number parameter. Parameter type 2 indicates a string parameter. Parameter type 3 indicates a coordinate. Generally, one either uses the set DCC, Dtheta, & Dphi (with a single index for the interaction) or the set Iso & Coord (with a single index for one spin of a spin pair). If coordinates and isotope types are used then is the function call to read them must supply both spin indices!**

Note that there are variations in the parameter names/units associated with a coordinate parameter (see Class coord). Users may input coordinates in Cartesian, Spherical, or Cylindrical terms. Coordinate parameters allow for lengths input in either Angstroms, nanometers, or meters and allow for angles specified in either degrees for radians. Similarly, the dipolar coupling constant parameter DCC can be input in Hz (as opposed to KHz) by using the name DCCHz rather than DCC.

3.20.3 Dipole Interaction Parameter Set 1

The simplest way to designate a dipolar interaction is to provide spin coordinates and isotope types for the two spins involved. These four quantities supply all the information for setting **{I, S**, δ_{zz}**,** θ, ϕ**}** for the interaction. The asymmetry in this case is set to 0 (typical for most dipolar treaments).

Dipolar Interaction Isotope & Coordinate Parameters

Parameter	Units	Examples Parameter (Type) : Value - Statement
Iso	none	Iso(0) (2) : 13C - Spin Isotope Type
Coord	A	Coord(0) (3) : (0.0, 0.0, 1.0) - Coordinate in Angstroms
CoordSph	A, deg	CoordSph(0) (3) : (1.0, 90.0, 9.0) - R=1 Angstrom, θ=ϕ=90 degrees
CoordCyl	A, deg	CoordCyl(0) (3) : (1.0, 0.0, 0.0) - R is 1 Angstrom, θ=90 degrees

Fig. 3-11 **Generic ASCII parameters to declare a GAMMA dipolar interactions using spin isotopes and spin coordinates. Parameter type=1 is a floating point value whereas parameter type=3 is a coordinate. Here the parameter name has (#) appended where # is the spin index.**

Note that the coordinates may be input relative to Cartesian, Spherical, or Cylindrical axes. Additional parameters are also available which allow distances to be expressed in nanometes and meters and allow angles to be expressed in radians. By including these types of parameter statements (right column in previous table) in an ASCII file a GAMMA, dipolar interaction can be set with the ***read*** function. For example, the code below reads “file.asc”.

ASCII File Read With { Iso, Coord }

file.asc

```
Iso(0)        (2) : 2H      - Spin 0 set to deuteriumr
Iso(1)        (2) : 15N     - Spin 1 set to nitrogen 15
Iso(2)        (2) : 13C     - Spin 2 set to carbon 13
Coord(0)      (3) : (0.0, 0.0, 0.0)   - Spin 0 coordinates (x,y,z A)
Coord(1)      (3) : (1.2, 0.2, 1.0)   - Spin 1 coordinates (x,y,z A)
CoordSph(2) (3) : (2.0, 0.0, 90.0)  - Spin 2 coordinates (r,θ,ϕ A,deg)
```

code.cc

```
IntDip D01, D02, D12;
D01.read(“file.asc”, 0, 1);
ParameterAVLSet pset;
pset.read(“file.asc”);
D02.read(pset, 0, 2);
D12.read(pset, 1, 2);
```

Figure 3-12 **Specifying dipolar interactions using an external ASCII file. In this case the interactions are defined using spin isotope types in combination with spin coordinates.**

Remember, these ASCII files are read as GAMMA parameter set files so that they may contain additional lines of information and additional parameters. Things such as column spacing is not important - read about GAMMA parameters sets for full details.

3.20.4 Dipole Interaction Parameter Set 2

Another simple way to specify a dipolar interaction is to provide the dipolar coupling constant, the spin quantum numbers involved, and a tensor orientation in the laboratory frame relative to the principal axis system. This would be the set of parameters {I, S, DCC, θ, ϕ}.

Dipolar Interaction I, S, DCC and Angle Parameters

Parameter	Units	Examples Parameter (Type) : Value - Statement
DI	none	DI __ (1) : 1.5 - Spin Quantum Number
DS	none	DS __ (1) : 0.5 - Spin Quantum Number
DCC	KHz	DCC (1) : 370.3 - Dipolar Coupling (KHz)
Dtheta	degrees	Dtheta (1) : 127.2 - Dip. Orientation from PAS z (deg)
Dphi	degrees	Dphi (1) : 270.9 - Dip. Orientation from PAS x(deg)

Fig. 3-13 **Generic ASCII parameters to declare a GAMMA dipolar interactions using spin quantum numbers, dipolar couplings, and dipolar orientations. Parameter type=1 is a floating point value. Here the parameter names may have (#) appended where # is the dipolar interaction index.**

By including these parameter statements (right column) in an ASCII file a GAMMA dipolar interaction can be set with the ***read*** function. For example, the code below reads “file.asc”.

ASCII File Read With { DI, DS, DCC, Dtheta, Dphi }

file2.asc

```
DI       (1) : 1.5       - Spin Quantum Number
DS       (1) : 0.5       - Spin Quantum Number
DCC      (1) : 70.3      - Dipolar Coupling (KHz)
Dtheta   (1) : 127.2     - Dipolar Orientation from PAS z (deg)
Dphi     (1) : 270.9     - Dipolar Orientation from PAS x(deg)
DI(2)    (1) : 0.5       - Spin Quantum Number
DS(2)    (1) : 0.5       - Spin Quantum Number
DCC(2)   (1) : 10.3      - Dipolar Coupling (KHz)
Dtheta(2) (1) : 0.0      - Dipolar Orient. from PAS z (deg)
Dphi(2)  (1) : 270.9     - Dipolar Orient. from PAS x(deg)
```

code2.cc

```
.............
.............
IntDip D;
D.read("file.asc");
IntDip D2;
D.read("file2.asc", 2);
.............
.
```

Figure 3-14 **Specifying a dipolar interaction using an external ASCII file. In this case the interactions are defined using spin quantum numbers in combination with dipolar coupling constants and dipolar orientation angles.**

These ASCII files areGAMMA parameter set files that may contain additional lines of information and additional parameters. Things such as column spacing are not important - read about GAMMA parameters sets for full details. The (#) appended to the parameter names is used to allow for the definition of multiple interactions in the same ASCII file.

3.20.5 Dipole Interaction Parameter Set 3

The previous two example parameter sets illustrated two distinct ways of specifying dipolar interactions in external ASCII files.

The first was based on associating a dipolar interaction with a ***spin pair***. The read statement in the source code used two ***spin indices*** as arguments and the parameters in the ASCII file had ***mandatory (#)***'s appended to their names with # indicating the ***spin index***. For each dipolar interaction, i.e. for each spin pair, two spin isotopes and two spin coordinates were the parameters of choice.

The second example was based on a more direct declaration of a dipole interaction. It does not associate the interaction with any specific spins. An optional single ***interaction index*** was used in the read statement and the parameters in the ASCII file had an ***optional (#)*** appended to their names with # indicating the ***interaction index***. For each dipolar interaction five parameters (all beginning with D) were required: DI, DS, DCC, Dtheta, & Dphi for two spin quantum numbers, a dipolar coupling strenthg, and two interaction orientation angles respectively.

Unfortunately, neither of these methods may be deemed "intuitive" by every GAMMA user. Although we can't accomodate everyone's view, we can allow for some variation. The alternative in this section takes the spin pair approach except that it allows users to assign a distance and angles for the spin pair (rather than use coordinates per spin).

Therein was nwhere the # was the Another simple way to specify a dipolar interaction is to provide the dipolar coupling constant, the spin quantum numbers involved, and a tensor orientation in the laboratory frame relative to the principal axis system. This would be the set of parameters {I, S, DCC, θ, ϕ}.

Dipolar Interaction Isotopes & Specific Orientation

Parameter	Units	Examples Parameter (Type) : Value - Statement
Iso	none	Iso(0) (2) : 13C - Spin Isotope Type
DOrient	A, Deg	DOrient(0,1) (3) : (2.0, 0.0, 90.0) - r,q,f Coordinate in Angstroms

Fig. 3-15 **Generic ASCII parameters to declare a GAMMA dipolar interactions using spin isotopes and spin coordinates. Parameter type=1 is a floating point value whereas parameter type=3 is a coordinate. Here the parameter name has (#) appended where # is the spin index.**

The spin quantum numbers can be read from isotope declarations rather than with use of parameters DI and DS. Another variation would be to use spin indices in the function reading these parameters rather than an interaction (or no) index. Here are the possibilities.

3.21 Dipolar Interaction Examples

3.21.1 Reading Dipolar Interaction Parameters

To keep GAMMA programs using GARP sequences versatile, users will want to keep all GARP specifications undetermined in the

3 Shift Anisotropy Interactions

3.1 Overview

The class IntCSA contains a fully functional chemical shift anisotropy interaction defined for a single spin. The class allows for the definition and manipulation of such interactions, in particular it allows for the construction of oriented shift anisotropy Hamiltonians. Note that this calss

3.2 Available Shift Anisotropy Interaction Functions

Cartesian Spin Tensor Functions

Auxiliary Functions

Hamiltonian Functions

I/O Functions

3.3 INTCSATheory

3.4 Figures

3.5 Literature Comparisons

3.6 Interaction Examples

3.17.1 Zero Field Transitions, First Order Spectra page 3-166

3.7 Example Programs

3.8 Constructors

3.8.1 IntCSA

Usage:

```
void IntCSA::IntCSA()
void IntCSA::IntCSA(const IntCSA& SA1)
void IntCSA::IntCSA(String& IsoI, double delzz, double theta=0.0, double phi=0.0, double eta=0.0)
void IntCSA::IntCSA(double qn, double delzz, double theta=0.0, double phi=0.0, double eta=0.0)
void IntCSA::IntCSA(ParameterAVLSet& pset, int idx=-1, int warn=2)
```

Description:

The function ***IntCSA*** is used to create a new shift anisotropy interaction.

1. IntCSA() - Called without arguments the function creates a NULL shift anisotropy interaction.
2. IntCSA(const IntCSA& SA1) - Called with another shift anisotropy interaction, a new shift anisotropy interaction is constructed which is identical to the input interaction.
3. IntCSA(String& IsoI, double delzz, double theta=0.0, double phi=0.0, double eta=0.0) - This will construct a new shift anisotropy interaction for a spin of type ***IsoI*** (i.e. 1H, 131Xe, 13C....). The strength of the interaction is set by the argument ***delzz*** assumed to be in units of ***PPM***. The value of ***delzz*** is related to the shift anisotropy ($(3/2)\Delta\sigma = \delta_{zz}$). Two interaction orientation is set by the two angles, ***theta*** (down from +z) and ***phi*** (over from +x), can be optionally specified in ***degrees***. The interaction asymmetry, ***eta,*** can be optionally input and is restricted to be within the range ***[0, 1]***.
4. IntCSA(double qn, double delzz, double theta=0.0, double phi=0.0, double eta=0.0) - New shift anisotropy interaction for a spin having quantum number ***qn***. Other arguments mirror the previous constructor.
5. IntCSAParameterAVLSet& pset, int idx=-1, int warn=2) - This will construct a new shift anisotropy interaction from parameters in the input parameter set ***pset***. If the optional index ***idx*** has been set ***>=0*** the shift anisotropy parameters scanned in ***pset*** will be assumed to have a ***(idx)*** appended to their names. The optional integer ***warn*** specifies how to handle failures in finding appropriate parameters.

Return Value:

Void. It is used strictly to create a shift anisotropy interaction.

Examples:

```
#include <gamma.h>
main()
  {
  IntCSA SA();                              // Make an empty shift anisotropy interaction.
  IntCSA SA1("13C", 150, 19.2, 10.0, 0.5);// New int., Δσ = 150 PPM, θ=19.2 deg, φ=10 degrees, η=0.5
  IntCSA SA5(SA2);                          // New interaction, SA5, identical to SA2.
  }
```

See Also: =, read, ask_read

3.8.2 =

Usage:

void IntCSA::operator= (const IntCSA& SA1)

Description:

The operator = is assign one shift anisotropy interaction to another.

Return Value:

Void.

Example:

```
#include <gamma.h>
main()
  {
  IntCSA SA1("13C", 100, 90.0);        // New int., Δσ = 66.6 PPM, θ=90 deg, φ=0 degrees, η=0
  IntCSA SA2 = SA1;                    // New interaction, SA2, identical to SA1.
  }
```

See Also: constructor, read, ask_read

3.9 Basic Functions

3.9.1 delzz

Usage:

```
#include <IntCSA.h>
double IntRank2::delzz () const
double IntRank2::delz () const
double IntRank2::delzz (double dz) const
double IntRank2::delz (double dz) const
```

Description:

The function ***delzz*** is used to either obtain or set the shift anisotropy interaction's δ_{zz} value in **PPM**. With no arguments the function returns the value. If an argument, ***dz***, is specified then the constant for the interaction is set. The function is overloaded with the name ***delz*** for convenience. Note that setting of ***delzz*** will alter the (equivalent) value of the shift anisotropy. Users must bear in mind that δ_{zz} is not the chemical shift anisotropy (CSA), $\Delta\sigma$, although the two values are related. The following relationship is useful in discerning a proper δ_{zz} value to be used in GAMMA[1].

$$\Delta\sigma = \sigma_{zz} - \frac{1}{2}(\sigma_{xx} + \sigma_{yy}) = \frac{3}{2}\delta_{zz} = \sigma_{\parallel} - \sigma_{\perp}$$

The last relationship used the nomenclature applicable in describing a symmetric CSA tensor with $\eta = 0$, where $\sigma_{\parallel} = \sigma_{zz}$ and $\sigma_{\perp} = \sigma_{xx} = \sigma_{yy}$.

Return Value:

Either void or a floating point number, double precision.

Examples:

```
#include <gamma.h>
main()
  {
  IntCSA SA1("13C", 100, 90.0);        // New int., Δσ = 66.6 PPM, θ =90 deg, φ =0 degrees, η =0
  SA1.delzz(150);                      // Set Δσ = 100 PPM
  cout << "\n\tdelzz value is "        // Output the delzz value to screen
       << SA1.delzz() << " PPM";
  }
```

See Also: CSA

1. Note that this class has no knowledge of field strengths nor gyromagnetic ratios. Since δ_{zz} and $\Delta\sigma$ values are maintained in PPM units a conversion must be performed to obtain frequency values. That involves the Larmor frequency of the spin assoicated with the interaction.

3.9.2 CSA

Usage:

```
#include <IntCSA.h>
double IntCSA::CSA() const
double IntCSA::CSA(double delsig) const
```

Description:

The function ***CSA*** is used to either obtain or set the interaction's shift anisotropy in PPM. With no arguments the function returns the shift anisotropy. If an argument, ***delsig***, is specified then the CSA for the interaction is set[1]. This function is related to function ***delzz*** and value δ_{zz}.

$$\Delta\sigma = \sigma_{zz} - \frac{1}{2}(\sigma_{xx} + \sigma_{yy}) = \frac{3}{2}\delta_{zz} = \sigma_{\|} - \sigma_{\perp}$$

The last relationship used the nomenclature applicable in describing a symmetric CSA tensor with $\eta = 0$, where $\sigma_{\|} = \sigma_{zz}$ and $\sigma_{\perp} = \sigma_{xx} = \sigma_{yy}$.

Return Value:

Either void or a floating point number, double precision.

Examples:

```
#include <gamma.h>
main()
  {
  IntCSA SA1("13C", 100, 90.0);        // New int., Δσ = 66.6 PPM, θ =90 deg, φ =0 degrees, η =0
  SA1.CSA(150);                        // Set Δσ = 150 PPM
  cout << "\n\tdelzz value is "        // Output the CSA value to screen
       << SA1.CSA() << " PPM";
  }
```

See Also: delz, delzz

1. Note that this class has no knowledge of field strengths nor gyromagnetic ratios. Since δ_{zz} and $\Delta\sigma$ values are maintained in PPM units a conversion must be performed to obtain frequency values. That involves the Larmor frequency of the spin assoicated with the interaction.

3.9.3 eta

Usage:

```
#include <IntCSA.h>
double IntRank2A::eta () const
double IntRank2A::eta (double SAeta) const
```

Description:

The function ***eta*** is used to either obtain or set the shift anisotropy interaction asymmetry. With no arguments the function returns the asymmetry (unitless). If an argument, ***SAeta***, is specified then the asymmetry for the interaction is set.The input value is **restricted to the range [0,1]** and is related to the shift anisotropy spatial tensor Cartesian components according to

$$\eta = (A_{xx} - A_{yy}) / A_{zz} \qquad |A_{zz}| \geq |A_{yy}| \geq |A_{xx}|$$

Note that setting ***eta*** will alter the 5 internal irreducible spherical spatial tensor components of the interaction.

Return Value:

Either void or a floating point number, double precision.

Examples:

```
#include <gamma.h>
main()
  {
  IntCSA SA1("13C", 100, 90.0);      // New int., Δσ = 66.6 PPM, θ =90 deg, φ =0 degrees, η =0
  cout << SA1;                       // Have a look at our interaction
  SA1.eta(0.97);                     // Set η = 0.97
  cout << "\n\tEta value is "        // Write Xi value to screen for 13C in 500 MHz (1H) spectrometer
       << SA1.eta();
  cout << SA1;                       // Now our interaction should be a bit different.
  }
```

See Also: delz, CSA, theta, phi

3.9.4 xi

Usage:

double IntCSA::xi(const String& IsoI, double Om) const
double IntCSA::xi(double Om) const

Description:

The function ***xi*** is used to obtain the GAMMA defined shift anisotropy interaction constant in **radians/sec**. The constant is used to scale the interaction such that both its spatial and spin tensors are "independent" of the interaction type.

$$\xi^{SA} = \sqrt{\frac{6\pi}{5}} h\gamma B_o \delta_{zz} = \sqrt{\frac{6\pi}{5}} \frac{\gamma}{\gamma_H} \Omega \delta_{zz} = \sqrt{\frac{6\pi}{5}} \frac{\gamma}{\gamma_H} 2\pi\Omega_\nu \delta_{zz} = \sqrt{\frac{8\pi}{15}} \frac{\gamma}{\gamma_H} 2\pi\Omega_\nu \Delta\sigma$$

This will be used in the formulation of shift anisotropy Hamiltonians according to.

$$\boldsymbol{H}^{SA}(\theta, \varphi) = \xi^{SA} \sum_{m}^{\pm 2} (-1)^m A_{2,-m}(\theta, \varphi) \bullet \boldsymbol{T}^{SA}_{2,m}$$

For a proton in a 500 MHz spectrometer with a δ_{zz} value of 1 Part Per Million (PPM) we have

$$\xi^{SA}_{^1H}\Big|_{\substack{500MHz \\ 1PPM}} = \sqrt{\frac{6\pi}{5}} [2\pi(500 \times 10^6 Hz)](1/10^6) = 6.1 \times 10^3 \text{sec}^{-1}$$

A more typical value would result from an anisotropy, $\Delta\sigma$, of 150 PPM for a carbon nucleus[1].

$$\xi^{CSA}_{^{13}C}\Big|_{\substack{500MHz \\ 150PPM}} = 2\pi \sqrt{\frac{6\pi}{5}} \left(\frac{\gamma_{^{13}C}}{\gamma_{^1H}}\right)(500 \times 10^4 Hz) = 2\pi\sqrt{37.7}125.7Hz = 1.534 \times 10^5 \text{sec}^{-1}$$

Return Value:

A floating point number, double precision.

Example:

```
#include <gamma.h>
main()
  {
  IntCSA SA1("13C", 100, 90.0);        // New int., Δσ = 66.6 PPM, θ =90 deg, φ =0 degrees, η =0
  SA1.CSA(150);                        // Set Δσ = 150 PPM
  cout << "\n\tXi value is "           // Write Xi value to screen for 13C in 500 MHz (1H) spectrometer
       << SA1.xi("13C", 500e6) << " /s";
  }
```

1. Note that $\Delta\sigma = 150$ implies that $\delta_{zz} = 100$ as discussed in the paragraph following the figure.

3.10 Spherical Spatial Tensor Functions

3.10.1 A0, A20

3.10.2 A1, A21

3.10.3 Am1, A2m1

3.10.4 A2, A22

3.10.5 Am2, A2m2

Usage:

```
#include <IntCSA.h>
complex IntRank2A::A0() const
complex IntRank2A::A20() const
complex IntRank2A::A0(double theta, double phi) const
complex IntRank2A::A20(double theta, double phi) const
complex IntRank2A::A1() const
complex IntRank2A::A21() const
complex IntRank2A::A1(double theta, double phi) const
complex IntRank2A::A21(double theta, double phi) const
complex IntRank2A::Am1() const
complex IntRank2A::A2m1() const
complex IntRank2A::Am1(double theta, double phi) const
complex IntRank2A::Am21(double theta, double phi) const
complex IntRank2A::A2() const
complex IntRank2A::A22() const
complex IntRank2A::A2(double theta, double phi) const
complex IntRank2A::A22(double theta, double phi) const
complex IntRank2A::Am2() const
complex IntRank2A::A2m2() const
complex IntRank2A::Am2(double theta, double phi) const
complex IntRank2A::A2m2(double theta, double phi) const
```

Description:

The functions ***AM*** and ***A2M*** are used to obtain the shift anisotropy interaction spatial tensor components $A_{2,m}$. I Here, the names are mapped to the spherical tensor components as follows:

$$\{A0, A20\} \rightarrow A_{2,\,0}$$

$$\{A1, A21\} \rightarrow A_{21} \qquad \{Am1, A2m1\} \rightarrow A_{2,\,-1}$$

$$\{A2, A22\} \rightarrow A_{2,\,2} \qquad \{Am2, A2m2\} \rightarrow A_{2,\,-2}$$

If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments ***theta*** and ***phi*** are given the returned tensor component is for the orientation at ***theta*** degrees down from the interactions PAS z-axis and ***phi*** degrees over from the interactions PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***degrees***.

$$A_{2,\,0}(\theta,\varphi) = \sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]$$

$$A_{2,\,1}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}\sin\theta[3\cos\theta - \eta(\cos\theta\cos 2\varphi - i\sin 2\varphi)] = -A_{2,\,1}{}^*(\theta,\varphi)$$

$$A_{2,\,2}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}\frac{1}{2}[3\sin^2\theta + \eta[\cos 2\varphi(1 + \cos^2\theta) - i2\sin 2\varphi\cos\theta]] = A_{2,\,-2}{}^*(\theta,\varphi)$$

Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type[1]. This component can also be related to the Cartesian tensor components for any arbitrary orientation.

$$A_{2,\,0} = \sqrt{6}[3A_{zz} - Tr\{A\}]$$

$$A_{21} = -\frac{1}{2}[A_{xz} + A_{zx} + i(A_{yz} + A_{zy})] \qquad A_{2,\,-1} = \frac{1}{2}[A_{xz} + A_{zx} + i(A_{yz} - A_{zy})]$$

$$A_{2,\,2} = \frac{1}{2}[A_{xx} - A_{yy} + i(A_{xy} + A_{yx})] \qquad A_{2,\,-2} = \frac{1}{2}[A_{xx} + (-A_{yy}) - i(A_{xy} + A_{yx})]$$

Return Value:

A complex number.

Example:

```
#include <gamma.h>
main()
  {
  IntDip D(0.5,1.5, 3.e4, 45.0, 45.0);      // Make a Dipolar interaction (I=1/2, S=3/2, DCC=30kHz).
  complex A20 = D.A20();                    // This is at theta=phi=45 degrees
  cout << D.A20(15.6, 99.3);                // This is at theta=15.6 and phi=99.3 degrees.
  }
```

See Also: Auv where u,v = {x,y,z}

---

1. Because the GAMMA platform accommodates different interaction types, the scaling on all spatial tensors is chosen to be independent of the interaction. Rather, the spatial tensors are related directly to the familiar rank two spherical harmonics $A_{2,\,m}(\theta,\varphi)\big|_{\eta\,=\,0} = Y_m^2(\theta,\varphi)$. Also, the sign on the term(s) involving η will have opposite sign if the common alternative definition of the PAS orientation ($|A_{zz}| \geq |A_{xx}| \geq |A_{yy}|$) is used rather that the definition used in GAMMA ($|A_{zz}| \geq |A_{yy}| \geq |A_{xx}|$)

3.11 Cartesian Spatial Tensor Functions

3.11.1 Axx

3.11.2 Ayy

3.11.3 Azz

3.11.4 Axy, Ayx

3.11.5 Axz, Azx

3.11.6 Ayz, Azy

Usage:

```
#include <IntCSA.h>
complex IntRank2::Axx() const
complex IntRank2::Axx(double theta, double phi) const
complex IntRank2::Ayy() const
complex IntRank2::Ayy(double theta, double phi) const
complex IntRank2::Azz() const
complex IntRank2::Azz(double theta, double phi) const
complex IntRank2::Axy() const
complex IntRank2::Axy(double theta, double phi) const
complex IntRank2::Ayx() const
complex IntRank2::Ayx(double theta, double phi) const
complex IntRank2::Axz() const
complex IntRank2::Axz(double theta, double phi) const
complex IntRank2::Azx() const
complex IntRank2::Azx(double theta, double phi) const
complex IntRank2::Ayz() const
complex IntRank2::Ayz(double theta, double phi) const
complex IntRank2::Azy() const
complex IntRank2::Azy(double theta, double phi) const
```

Description:

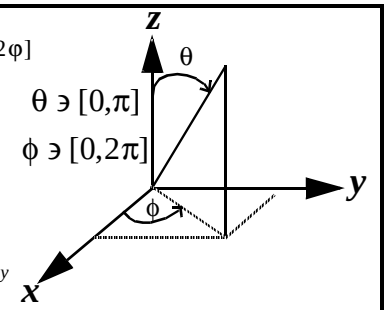
The functions ***Auv*** are used to obtain the dipolar interaction spatial tensor Cartesian components A_{uv}. If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments ***theta*** and ***phi*** are given the returned tensor component is for the orientation at ***theta*** degrees down from the interactions PAS z-axis and ***phi*** degrees over from the interactions PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***degrees***. The formulae for these components are given below (see Figure 19-7, page 2-53).

$$A_{xx}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}[3\sin^2\theta - 1 + \eta\cos 2\varphi\cos^2\theta] \qquad A_{yy}(\theta,\varphi) = -\sqrt{\frac{5}{24\pi}}[1 + \eta\cos 2\varphi]$$

$$A_{zz}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}[3\cos^2\theta - 1 + \eta\sin^2\theta\cos 2\varphi]$$

$$A_{xz}(\theta,\varphi) = -\sqrt{\frac{5}{24\pi}}\sin\theta\cos\theta[3 - \eta\cos 2\varphi] = A_{zx}$$

$$A_{xy}(\theta,\varphi) = -\sqrt{\frac{5}{24\pi}}\eta\sin 2\varphi\cos\theta = A_{yx} \qquad A_{yz}(\theta,\varphi) = -\sqrt{\frac{5}{24\pi}}\eta\sin\theta\sin 2\varphi = A_{zy}$$



Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type.

Return Value:

A complex number.

Example:

```
#include <gamma.h>
main()
  {
  IntCSA SA(1.5, 3.e5, 0.2, 45.0, 45.0);    // Make a shift anisotropy interaction.
  complex A20 = SA.A20();                   // This is at theta=phi=45 degrees
  cout << SA.A20(15.6, 99.3);               // This is at theta=15.6 and phi=99.3 degrees.
  }
```

See Also: Ayy, Azz, Axy, Axz, Ayx, Ayz, Azx, Azy

3.12 Spin Tensor Functions

3.12.1 Tcomp

Usage:

```
#include <IntCSA.h>
matrix IntRank2T::Tcomp(int comp)
```

Description:

The function ***Tcomp*** is used to obtain a shift anisotropy interaction spin tensor component. The component desired is specified by the argument ***comp*** which relates to the m value as follows:

comp:	0	1	2	3	4
$T^{SA}_{2,m}$:	$T^{SA}_{2,0}$	$T^{SA}_{2,1}$	$T^{SA}_{2,-1}$	$T^{SA}_{2,2}$	$T^{SA}_{2,-2}$

The spin components are given

$$T^{SA}_{2,0} = \frac{1}{\sqrt{6}}[3I_z^2 - \vec{I}^2] = \frac{1}{\sqrt{6}}[3I_z^2 - I(I+1)]$$

$$T^{SA}_{2,\pm 1} = \mp\frac{1}{2}[I_\pm\ I_z + I_z I_\pm\ ] \qquad T^{SA}_{2,\pm 2} = \frac{1}{2}I_\pm^2$$

and will be returned as matrices of dimension ***2I+1*** where ***I*** is the spin quantum number associated with the interaction.

Return Value:

A matrix.

Example:

```
#include <gamma.h>
main()
  {
  IntCSA SA(1.5, 3.e5, 0.2, 45.0, 45.0);    // Make a shift anisotropy interaction.
  matrix T20 = SA.Tcomp(0);                 // This is the T20 spin tensor component
  cout << T20);                             // Have a look on screen.
  }
```

3.13 Auxiliary Functions

3.13.1 setPAS

Usage:

```
#include <IntCSA.h>
void IntCSA::setPAS()
```

Description:

The functions ***setPAS*** is used to orient the shift anisotropy interaction into it's principal axis system. All 5 spatial tensor components will be set to PAS values and the internal orientation angles set to zero.

Return Value:

None.

Example:

```
IntCSA SA(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a shift anisotropy interaction.
SA.setPAS();                          // As if we used SA(1.5,3.e5,0.2,0,0)
```

See Also: theta, phi, orient

3.13.2 symmetric

Usage:

```
#include <IntCSA.h>
int IntCSA::symmetric() const
```

Description:

The functions ***symmetric*** is used to check if the shift anisotropy interaction has any asymmetry. The function will return true if the interaction is symmetric and false if there is some asymmetry (non-zero eta value).

Return Value:

An integer

Example:

```
IntCSA SA(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a shift anisotropy interaction.
if(SA.symmetric()) cout << "Yep";    // We should get No for SA because eta=0.2)
else                   << "Nope";
```

See Also: eta

3.13.3 PAS

Usage:

```
int IntCSA::PAS) const
```

Description:

The function ***PAS*** is used to check if the shift anisotropy interaction is oriented in its PAS or not. The function will return true if the interaction is PAS aligned and false if not).

Return Value:

An integer

Example:

```
IntCSA SA(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a shift anisotropy interaction.
if(SA.PAS()) cout << "Yep";          // We should get No for SA because neither θ or ϕ is 0)
else                      << "Nope";
```

See Also: eta

3.13.4 qn

Usage:

```
#include <IntCSA.h>
double IntCSA::qn() const
```

Description:

The functions ***qn*** is used to obtain the shift anisotropy interaction spin quantum number. The function will return a double which will be an integer multiple of 0.5 which is not less than 1 (1.0, 1.5, 2.5, 3.0,.....).

Return Value:

A double

Example:

```
IntCSA SA(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a shift anisotropy interaction.
double HS = 2.*SA.qn()+1.;           // The spin Hilbert space of SA
```

See Also: none

3.14 I/O Functions

3.14.1 read

Usage:

```
void IntCSA::read(const String& filename, const spin_sys) const
void IntCSA::read(const String& filename, const spin_sys) const
void IntCSA::read(const String& filename, const spin_sys) const
void IntCSA::read(const String& filename, const spin_sys) const
```

Description:

The function ***delzz*** is used to either obtain or set the interaction shift anisotropy coupling constant. With no arguments the function returns the coupling in Hz. If an argument, ***dz***, is specified then the coupling constant for the interaction is set. It is assumed that the input value of ***dz*** is in units of ***Hz***. The function is overloaded with the name ***delz*** for convenience. Note that setting of ***delzz*** will alter the (equivalent) value of the shift anisotropy coupling ***QCC***/***NQCC*** as well as the shift anisotropy frequency.

Return Value:

Either void or a floating point number, double precision.

Example(s):

```
#include <IntCSA.h>
IntCSA SA();                    // Empty shift anisotropy interaction.
SA.delzz(100000.0);             // Set QCC to 100 KHz.
cout << SA.delz ();             // Write coupling constant to std output.
```

See Also: QCC, NQCC, wQ

3.14.2 ask

Usage:

```
#include <IntCSA.h>
double IntCSA:: () const
double IntCSA::delz () const
double IntCSA::delzz (double dz) const
double IntCSA::delz (double dz) const
```

Description:

The function ***delzz*** is used to either obtain or set the interaction shift anisotropy coupling constant. With no arguments the function returns the coupling in Hz. If an argument, ***dz***, is specified then the coupling constant for the interaction is set. It is assumed that the input value of ***dz*** is in units of ***Hz***. The function is overloaded with the name ***delz*** for convenience. Note that setting of ***delzz*** will alter the (equivalent) value of the shift anisotropy coupling ***QCC***/***NQCC*** as well as the shift anisotropy frequency.

Return Value:

Either void or a floating point number, double precision.

Example(s):

```
#include <IntCSA.h>
IntCSA SA();                    // Empty shift anisotropy interaction.
SA.delzz(100000.0);             // Set QCC to 100 KHz.
cout << SA.delz ();             // Write coupling constant to std output.
```

See Also: QCC, NQCC, wQ

3.14.3 askset

Usage:

```
#include <IntCSA.h>
double IntCSA:: () const
double IntCSA::delz () const
double IntCSA::delzz (double dz) const
double IntCSA::delz (double dz) const
```

Description:

The function ***delzz*** is used to either obtain or set the interaction shift anisotropy coupling constant. With no arguments the function returns the coupling in Hz. If an argument, ***dz***, is specified then the coupling constant for the interaction is set. It is assumed that the input value of ***dz*** is in units of ***Hz***. The function is overloaded with the name ***delz*** for convenience. Note that setting of ***delzz*** will alter the (equivalent) value of the shift anisotropy coupling ***QCC***/***NQCC*** as well as the shift anisotropy frequency.

Return Value:

Either void or a floating point number, double precision.

Example(s):

```
#include <IntCSA.h>
IntCSA SA();                    // Empty shift anisotropy interaction.
SA.delzz(100000.0);             // Set QCC to 100 KHz.
cout << SA.delz ();             // Write coupling constant to std output.
```

See Also: QCC, NQCC, wQ

3.14.4 print

Usage:

```
#include <IntCSA.h>
ostream& IntCSA::print (ostream& ostr, int fflag=-1)
```

Description:

The function ***print*** is used to write the interaction shift anisotropy coupling constant to an output stream ***ostr***. An additional flag ***fflag*** is set to allow some control over how much information is output. The default (***fflag !=0***) prints all information concerning the interaction. If ***fflag*** is set to zero only the basis parameters are printed.

Return Value:

The ostream is returned.

Example:

```
#include <IntCSA.h>
IntCSA SA(2.5, 2.e6, 0.2, 45.7, 15.0); // Make a shift anisotropy interaction.
cout << SA;                            // Write the interaction to standard output.
```

See Also: <<

3.14.5 <<

Usage:

```
#include <IntCSA.h>
friend ostream& operator << (ostream& out, IntCSA& SA)
```

Description:

The operator << defines standard output for the interaction shift anisotropy coupling constant.

Return Value:

The ostream is returned.

Example:

```
#include <IntCSA.h>
IntCSA SA(1.5, 3.e5, 0.2);             // Make a shift anisotropy interaction.
cout << SA;                            // Write the interaction to standard output.
```

See Also: print

3.14.6 printSpherical

Usage:

```
#include <IntCSA.h>
ostream& IntCSA::print (ostream& ostr, int fflag=-1)
```

Description:

The function ***print*** is used to write the interaction shift anisotropy coupling constant to an output stream ***ostr***. An additional flag ***fflag*** is set to allow some control over how much information is output. The default (***fflag !=0***) prints all information concerning the interaction. If ***fflag*** is set to zero only the basis parameters are printed.

Return Value:

The ostream is returned.

Example:

```
#include <IntCSA.h>
IntCSA SA(2.5, 2.e6, 0.2, 45.7, 15.0); // Make a shift anisotropy interaction.
cout << SA;                            // Write the interaction to standard output.
```

See Also: <<

3.14.7 printCartesian

Usage:

```
#include <IntCSA.h>
ostream& IntCSA::print (ostream& ostr, int fflag=-1)
```

Description:

The function ***print*** is used to write the interaction shift anisotropy coupling constant to an output stream ***ostr***. An additional flag ***fflag*** is set to allow some control over how much information is output. The default (***fflag !=0***) prints all information concerning the interaction. If ***fflag*** is set to zero only the basis parameters are printed.

Return Value:

The ostream is returned.

Example:

```
#include <IntCSA.h>
IntCSA SA(2.5, 2.e6, 0.2, 45.7, 15.0); // Make a shift anisotropy interaction.
cout << SA;                            // Write the interaction to standard output.
```

See Also: <<

3.15 Description

3.15.1 Overview

A chemical shift is the observed effect from the electron cloud surrounding a nucleus responding to an applied magnetic field. The spin itself experiences not only the applied field but also a field from the perturbed electron cloud, the latter field generally opposing the applied field or “shielding” the nucleus. Not only can the shielding contribution be quite large, it is usually orientationally dependent because the surrounding electron cloud is no spherical (due to chemical bonds). In the following discussion we will not be concerned with the isotropic and anti-symmetric parts of the shielding. The former produces measureable chemical shifts whereas the latter is rarely seen. Rather the focus will be on the symmetric rank 2 contribution, that which produces relaxation effects in liquid NMR and orientationally dependent shifts in solids.

3.15.2 Coordinate Systems

We will shortly concern ourselves with the mathematical representation of chemical shift interactions, in particular their description in terms of spatial and spin tensors. The spatial tensors will be cast in both Cartesian and spherical coordinates and we will switch between the two when convenient. The figure below relates the orientation angles theta and phi to the standard right handed coordinate system in all GAMMA treatments.

Cartesian and Spherical Coordinate Systems

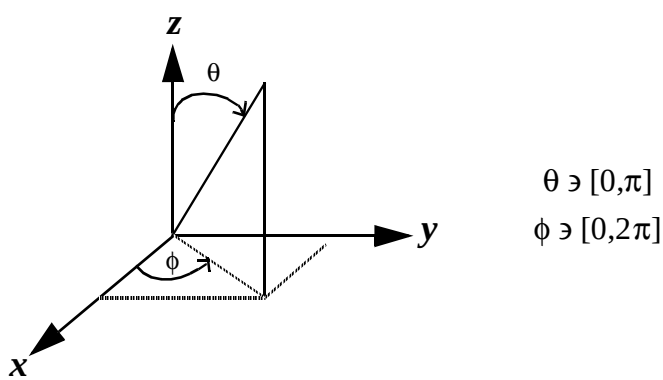


Figure 19-9 **The right handed Cartesian axes with the spherical angles and radius.**

3.15.3 Internal Structure

The internal structure of class ***IntSA*** contains the quantities listed in the following table (names shown are also internal).

Table 3-1: Internal Structure of Class IsoSA

Name	Description	Type	Name	Description	Type
I	Spin Quantum Number	double	THETA	Orientation Angle	double

Table 3-1: Internal Structure of Class IsoSA

Name	Description	Type	Name	Description	Type
DELZZ	Spatial Tensor δ_{zz}	double	Asph	Spatial Tensor Values	complex*
ETA	Spatial Tensor η	double	Tsph	Spin Tensor Values	matrix*
PHI	Orientation Angle	double			

The values of ***I*** is the spin quantum number of the shift anisotropy nucleus. It dictates how many energy levels (and transitions) are associated with the shift anisotropy interaction. It is intrinsically tied into the values and dimensions of the matrices in the vector ***Tsph***. Note that ***I*** will be an integer multiple of 1/2 and that only nuclei with ***I***>1/2 will have a quadrupole moment.

The two values ***DELZZ*** and ***ETA*** are all that is required to specify the shift anisotropy interaction strength and may be used to represent the shift anisotropy spatial tensor. However, in GAMMA the value of ***DELZZ*** is factored out of the spatial tensor such that all rank two interactions (such as the shift anisotropy interaction) have the same spatial tensor scaling.

The two angles ***THETA*** and ***PHI*** indicate how the shift anisotropy interaction is aligned relative to the interaction principal axes (PAS). These are one in the same as the angles shown in Figure 19-9 when the Cartesian axes are those of the PAS with the origin vaguely being the center of the nucleus. These are intrinsically tied into the values in the array ***Asph***.

There are five values in the complex vector ***Asph*** and these are irreducible spherical components of the shift anisotropy spatial tensor oriented at angle ***THETA*** down from the PAS z-axis and over angle ***PHI*** from the PAS x-axis. Note that these 5 values are not only orientation dependent, they are also ***ETA*** dependent. If either of the three the interaction values {***ETA***, ***THETA, PHI***} are altered these components will all be reconstructed. The values in ***Asph*** will be scaled such that they are consistent with other rank 2 spatial tensors in GAMMA which are independent of the interaction type.

Structure of a Variable of Class IntCSA

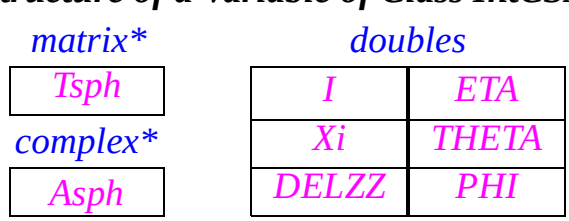


Figure 19-10 **Depiction of class IntCSA contents, i.e. what each GAMMA defined shift anisotropy interaction contains. The values of both Xi and DELZZ are maintained for convenience (one being deduced from the other via I). Tsph will contain 5 matrices which dimension will be 2*I+1 and Asph will contain 5 complex numbers.**

The vector of matrices relates to the sperical spin tensor components according to:

Tsph:	[0]	[1]	[2]	[3]	[4]

$T^{SA}_{2,m}$:	$T^{SA}_{2,0}$	$T^{SA}_{2,1}$	$T^{SA}_{2,-1}$	$T^{SA}_{2,2}$	$T^{SA}_{2,-2}$

and the vector of complex numbers relate to the spherical spatial tensor components via

Asph:	[0]	[1]	[2]	[3]	[4]
$A_{2,m}$:	$A_{2,0}$	$A_{2,1}$	$A_{2,-1}$	$A_{2,2}$	$A_{2,-2}$

3.15.4 Classical Chemical Shielding Treatment

A chemical shift is the observed effect from the electron cloud surrounding a nucleus responding to an applied magnetic field. The spin itself experiences not only the applied field but also a field from the perturbed electron cloud, the latter field generally opposing the applied field or "shielding" the nucleus. We can write this latter "induced" field in terms of the applied field, $\vec{\boldsymbol{B}}_o$, as

$$\vec{\boldsymbol{B}}_{induced} = -\ddot{\sigma} \bullet \vec{\boldsymbol{B}}_o$$

where σ is the chemical shielding tensor, a 3x3 array in Cartesian space, and the $\vec{\boldsymbol{B}}$'s vectors in Cartesian space. In matrix form this is simply[1]

$$\begin{bmatrix} B_{ind,\,x} \\ B_{ind,\,y} \\ B_{ind,\,z} \end{bmatrix} = -\begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{bmatrix}_i \bullet \begin{bmatrix} B_{0x} \\ B_{0y} \\ B_{0z} \end{bmatrix},$$

the induced field depends on the applied field strength, the applied field orientation, and the surrounding electron cloud. Note that $\vec{\boldsymbol{B}}_{induced}$ will not necessarily be co-linear with the applied field. Of course, every nuclear spin will have its own associated chemical shielding tensor. The classical interaction energy between this induced field and a nuclear spin is

$$E_i^{CS} = -\vec{\mu}_i \bullet \vec{\boldsymbol{B}}_{induced} = \vec{\mu}_i \bullet \ddot{\sigma}_i \bullet \vec{\boldsymbol{B}}_o$$

where $\vec{\mu}$ is the magnetic moment, i the spin index, E the energy, and subscript CS used to denote chemical shielding.

3.15.5 Quantum Mechanical Formulation

The associated Hamiltonian is obtained from substitution of $h\gamma\vec{\boldsymbol{I}}_i$ for $\vec{\mu}_i$ in the energy equation.

$$\boldsymbol{H}_i^{CS} = h\gamma_i \vec{\boldsymbol{I}}_i \bullet \ddot{\sigma}_i \bullet \vec{\boldsymbol{B}}_o \tag{0-19}$$

In matrix form this equation looks like

$$\boldsymbol{H}_i^{CS} = h\gamma_i \begin{bmatrix} \boldsymbol{I}_{ix} & \boldsymbol{I}_{iy} & \boldsymbol{I}_{iz} \end{bmatrix} \bullet \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{bmatrix}_i \bullet \begin{bmatrix} B_{0x} \\ B_{0y} \\ B_{0z} \end{bmatrix}. \tag{0-20}$$

Taking the magnitude of the applied field out, equation (0-19) is simply

1. Note that the effect of the chemical shielding is to alter the field which the spin experiences. This is clearly seen from the product $\hat{\sigma}_i \bullet \vec{\boldsymbol{B}}_o$ which produces an effective field vector for the spin.

$$\boldsymbol{H}^{CS} = h\gamma B_o \sum_{u}^{axes} \sum_{v}^{axes} \langle 1|\vec{\boldsymbol{I}}|u\rangle\langle u|\hat{\sigma}|v\rangle\langle v|\vec{\boldsymbol{B}}_n|1\rangle \tag{0-21}$$

with $u, v \in \{x, y, z\}$ and $\vec{\boldsymbol{B}}_n$ a normalized magnetic field vector in the direction of the applied field.

3.15.6 Cartesian Tensor Formulation

Equation (0-20) can also be rearranged to produce an equation involving two rank 2 tensors by taking the dyadic product of the vectors $\vec{\boldsymbol{I}}$ and $\vec{\boldsymbol{B}}_n$.

$$\boldsymbol{H}^{CS} = h\gamma B_o \sum_{u}^{axes} \sum_{v}^{axes} \langle u|\hat{\sigma}|v\rangle\langle v|\vec{\boldsymbol{B}}_n|1\rangle\langle 1|\vec{\boldsymbol{I}}|u\rangle = h\gamma B_o \sum_{u}^{axes} \sum_{v}^{axes} \langle u|\sigma|v\rangle\langle v|\vec{\boldsymbol{B}}_n\vec{\boldsymbol{I}}|u\rangle$$

The dyadic product to produce $\vec{\boldsymbol{B}}_n\vec{\boldsymbol{I}}$ is explicitly done *via*

$$\begin{bmatrix} B_{nx} \\ B_{ny} \\ B_{nz} \end{bmatrix} \bullet \begin{bmatrix} \boldsymbol{I}_x & \boldsymbol{I}_y & \boldsymbol{I}_z \end{bmatrix} = \begin{bmatrix} B_{nx}\boldsymbol{I}_x & B_{nx}\boldsymbol{I}_y & B_{nx}\boldsymbol{I}_z \\ B_{ny}\boldsymbol{I}_x & B_{ny}\boldsymbol{I}_y & B_{ny}\boldsymbol{I}_z \\ B_{nz}\boldsymbol{I}_x & B_{nz}\boldsymbol{I}_y & B_{nz}\boldsymbol{I}_z \end{bmatrix}.$$

The chemical shielding Hamiltonian can thus be formulated as a scalar product of two rank 2 tensors. Letting $\hat{\boldsymbol{T}}^{CS} = \vec{\boldsymbol{B}}_n\vec{\boldsymbol{I}}$, we have

$$\boldsymbol{H}^{CS} = h\gamma B_o \hat{\sigma} \bullet \hat{\boldsymbol{T}}^{CS}.$$

or equivalently (0-22)

$$\boldsymbol{H}^{CS} = h\gamma B_o \sum_{u}^{axes} \sum_{v}^{axes} \langle u|\sigma|v\rangle\langle v|\hat{\boldsymbol{T}}^{CS}|u\rangle$$

3.15.7 Spherical Tensor Formulation

The previous equation, (0-22), can also be rewritten in term of irreducible spherical components rather than in terms of the Cartesian components using the substitution

$$\sum_{l=0}^{2} \sum_{m}^{\pm l} (-1)^m A_{l-m}^{CS} \hat{\boldsymbol{T}}_{lm}^{CS} = \sum_{u}^{axes} \sum_{v}^{axes} \langle u|\sigma|v\rangle\langle v|\hat{\boldsymbol{T}}^{CS}|u\rangle \tag{1}$$

The result is

$$\boldsymbol{H}^{CS} = h\gamma B_o \sum_{l=0}^{2} \sum_{m}^{\pm l} (-1)^m A_{l-m}^{CS} \bullet \hat{\boldsymbol{T}}_{lm}^{CS} \tag{1-1}$$

3.15.8 Shift Anisotropy Spherical Tensor Spin Components

We can obtain the 9 irreducible spherical components of the CS rank 2 "spin" tensor[1] directly from the Cartesian components, $\langle v|\hat{\boldsymbol{T}}^{CS}|u\rangle$, as indicated in GAMMA Class Documentation on Spin Tensor. These are

$$\boldsymbol{T}^{CS}_{l,\,m},$$

where CS signifies the chemical shielding interaction. The tensor index l spans the rank: $l \in [0, 2]$ while the tensor index m spans l: $m \in [-l, l]$ The nine formulas for these quantities a listed in the following figure where the field components are those of the normalized field vector $\vec{\boldsymbol{B}}_n$.[2]

Shielding Rank 2 Irreducible Spherical Spin-Space Tensor Components

$$T_{0,\,0} = \frac{-1}{\sqrt{3}}\left[\boldsymbol{I}_z B_z + \frac{1}{2}(\boldsymbol{I}_+ B_- + \boldsymbol{I}_- B_+)\right] = \frac{-1}{\sqrt{3}}\vec{\boldsymbol{I}} \bullet \vec{\boldsymbol{B}}_n$$

$$T_{1,\,0}(i) = \frac{-1}{2\sqrt{2}}[\boldsymbol{I}_+ B_- - \boldsymbol{I}_- B_+] \qquad T_{1,\,\pm 1}(i) = \frac{-1}{2}[\boldsymbol{I}_\pm\ B_z - \boldsymbol{I}_z B_\pm\ ]$$

$$T_{2,\,0} = \frac{1}{\sqrt{6}}[3\boldsymbol{I}_z B_z - (\vec{\boldsymbol{I}} \bullet \vec{\boldsymbol{B}}_n)]$$

$$T_{2,\,\pm 1} = \mp\frac{1}{2}[\boldsymbol{I}_{\pm\ +} B_z + \boldsymbol{I}_z B_\pm\ ] \qquad T_{2,\,\pm 2} = \frac{1}{2}[\boldsymbol{I}_\pm\ B_\pm\ ]$$

For a spin 1/2 particle and $\vec{B}_0 = B_0\vec{B}_n$, the matrix form of these tensor components are shown in the following figure in the single spin Hilbert space. The spin index has been omitted, the field components are those of the normalized vector $\vec{B}_n$.

The matrix representation of these nine tensor components will depend upon the matrix representations of the individual spin operators from which they are constructed[3]. These in turn depend upon the quantum number of the spin involved. For a treatment of a spin 1/2 particle the shielding tensor components are expressed in their matrix form in the default product basis of GAMMA as

1. Due to the nature of the CS interaction, the rank 2 tensor treatment produces a "spin" tensor $T^{CS}_{l,\,m}$ which contains spatial components, namely the magnetic field vector. As a result, care must be used when performing spatial rotations on shielding tensors. Any spatial rotations must involve rotations of both σ and $\boldsymbol{T}$
2. For these formulae, it is important to note that it is the second component in the composite spin/space tensor which is set to the normalized magnetic field vector $\vec{B}_n$, although we might just as well have used the first vector instead. The difference is that the $l = 1$ equations would then appear of opposite sign from those given here. Our field vector has be set to point along the positive z-axis in the laboratory frame.
3. Note that the spin tensors are invariably constructed in the laboratory coordinate system. Here the z-axis corresponds to the direction of the spectrometer static magnetic field and the coordinate system is right-handed.

follows. In this case the spin index is implicit.

General I=1/2 Spin-Space Tensor Components Matrix Representations

$$T^{(2)}_{0,0} = \frac{-1}{2\sqrt{3}}\begin{bmatrix} B_z & B_- \\ B_+ & -B_z \end{bmatrix} \qquad T^{(2)}_{1,0} = \frac{-1}{2\sqrt{2}}\begin{bmatrix} 0 & B_- \\ -B_+ & 0 \end{bmatrix} \qquad T^{(2)}_{1,-1} = \frac{-1}{2}\begin{bmatrix} -B_-/2 & 0 \\ B_z & B_-/2 \end{bmatrix} \qquad T^{(2)}_{1,1} = \frac{-1}{2}\begin{bmatrix} -B_+/2 & B_z \\ 0 & B_+/2 \end{bmatrix}$$

$$T^{(2)}_{2,0} = \frac{1}{2\sqrt{6}}\begin{bmatrix} 2B_z & -B_- \\ -B_+ & -2B_z \end{bmatrix} \quad T^{(2)}_{2,-1} = \frac{1}{2}\begin{bmatrix} B_-/2 & B_z \\ 0 & -B_-/2 \end{bmatrix} \quad T^{(2)}_{2,1} = \frac{-1}{2}\begin{bmatrix} B_+/2 & 0 \\ B_z & -B_+/2 \end{bmatrix} \quad T^{(2)}_{2,-2} = \frac{1}{2}\begin{bmatrix} 0 & 0 \\ B_- & 0 \end{bmatrix} \quad T^{(2)}_{2,2} = \frac{1}{2}\begin{bmatrix} 0 & B_+ \\ 0 & 0 \end{bmatrix}$$

The raising an lowering components of the field vector are defined in the standard fashion, namely $\vec{B}_\pm = B_x \pm iB_y$. The simplest situation occurs when magnetic field points along the positive z-axis, $\vec{\boldsymbol{B}}_n = \vec{\boldsymbol{k}}$, *i.e.* these spin-space tensors are written in the laboratory frame. Then, the (normalized) field vector simplifies, $\vec{B}_z = 1$ and $B_x = B_y = \vec{B}_\pm = 0$. The applicable equations for the shielding space-spin tensors are then as follows.

Chemical Shielding Spin-Space Tensor Components, Bo Along z-Axis

$$T^{CS2}_{0,0}(ij) = \frac{-1}{\sqrt{3}}\boldsymbol{I}_{iz}$$

$$T^{CS2}_{1,0}(ij) = 0 \qquad T^{CS2}_{1,\pm1}(ij) = \frac{-1}{2}\boldsymbol{I}_{i\pm}$$

$$T^{CS2}_{2,0}(ij) = \frac{2}{\sqrt{6}}\boldsymbol{I}_{iz}$$

$$T^{CS2}_{2,\pm1}(ij) = \mp\frac{1}{2}\boldsymbol{I}_{i\pm} \qquad T^{CS2}_{2,\pm2}(ij) = 0$$

For a spin 1/2 particle and $\vec{B}_0 = B_0\vec{B}_n$ along the positive z-axis, the matrix form of these tensor components are shown in the following figure[1] (in the single spin Hilbert space).

I=1/2 Spin-Space Tensor Components Matrix Representations, Bo on z-Axis

$$T^{CS2}_{0,0} = \frac{-1}{2\sqrt{3}}\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad T^{CS2}_{1,0} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad T^{CS2}_{1,-1} = \frac{-1}{2}\begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \qquad T^{CS2}_{1,1} = \frac{-1}{2}\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

$$T^{CS2}_{2,0} = \frac{1}{\sqrt{6}}\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad T^{CS2}_{2,1} = \frac{-1}{2}\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \quad T^{CS2}_{2,-1} = \frac{1}{2}\begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \quad T^{CS2}_{2,-2} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad T^{CS2}_{2,2} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

We must very careful in using these single spin rank 2 shielding tensors of this type because they

1. The GAMMA program which produced these matrix representations can be found at the end of this Chapter, Rank2SS_SpinT.cc.

contain both spatial and spin components. If we desire to express the shielding Hamiltonian relative to a particular set of axes we must insure that both the spatial tensor and the “spin” tensor are expressed in the proper coordinates. The spatial tensor alone cannot be rotated as it rotates only part of the spatial components[1]. It is improper to rotate this tensor in spin space because it also rotates spatial variables.Furthermore, note that **these rank 2 components are not the same as the rank 1 tensor components**.

1. See the discussion in Mehring

3.15.9 Shielding Spherical Spatial Tensor General Components

The 9 irreducible spherical components of a rank two spatial tensor, $A_{lm}^{(2)}$, are related to its Cartesian components by the following formulas (See GAMMA Class Documentation on Spatial Tensor).

$$\begin{gathered} A_{0,0} = \frac{-1}{\sqrt{3}}[A_{xx} + A_{yy} + A_{zz}] = \frac{-1}{\sqrt{3}}Tr\{A\} \\ A_{1,0} = \frac{-i}{\sqrt{2}}[A_{xy} - A_{yx}] \qquad A_{1,\pm1} = \frac{-1}{2}[A_{zx} - A_{xz} \pm i(A_{zy} - A_{yz})] \\ A_{2,0} = \sqrt{6}[3A_{zz} - (A_{xx} + A_{yy} + A_{zz})] = \sqrt{6}[3A_{zz} - Tr\{A\}] \\ A_{2,\pm1} = \mp\frac{1}{2}[A_{xz} + A_{zx} \pm i(A_{yz} + A_{zy})] \qquad A_{2,\pm2} = \frac{1}{2}[A_{xx} - A_{yy} \pm i(A_{xy} + A_{yx})] \end{gathered} \tag{1-2}$$

Again the subscript l spans the rank as l = [0, 2], and the subscript m spans +/- l, m = [-l, l].

In this chemical shielding treatment we then have the components $A_{l,m}^{CS}$ as indicated in equation (1-1). Thus, the irreducible spherical tensor components can be obtained by substituting the Cartesian elements of σ into equations (1-2).

A general rank two Cartesian tensor can be rewritten in terms of a sum over tensors of ranks 0 through 2 as follows,

$$\hat{\sigma} = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{bmatrix} = \underbrace{\sigma_{iso}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Rank 0}} + \underbrace{\begin{bmatrix} 0 & \alpha_{xy} & \alpha_{xz} \\ -\alpha_{xy} & 0 & \alpha_{yz} \\ -\alpha_{xz} & -\alpha_{yz} & 0 \end{bmatrix}}_{\text{Rank 1}} + \underbrace{\begin{bmatrix} \delta_{xx} & \delta_{xy} & \delta_{xz} \\ \delta_{yx} & \delta_{yy} & \delta_{yz} \\ \delta_{zx} & \delta_{zy} & \delta_{zz} \end{bmatrix}}_{\text{Rank 2}}$$

where

$$\sigma_{iso} = \frac{1}{3}Tr\{\hat{\sigma}\} \qquad \alpha_{xy} = \frac{1}{2}(\sigma_{xy} - \sigma_{yx}) \qquad \delta_{xy} = \frac{1}{2}(\sigma_{xy} + \sigma_{yx} - 2\sigma_{iso})$$

The rank 0 part is isotropic (scalar), the rank 1 part is antisymmetric and traceless, and the rank 2 part traceless and symmetric.

3.15.10 Unscaled Shielding Spherical Spatial Tensor PAS Components

As with any spatial tensor, the chemical shielding spatial tensor can be specified in its principal axis system, the set of axes in which the irreducible rank 2 component is diagonal[1]. The shielding tensor values are experimentally determined in the tensor principal axes.

$$\hat{\sigma}(PAS) = \sigma_{iso}\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix} + \begin{bmatrix}0&\alpha_{xy}&\alpha_{xz}\\-\alpha_{xy}&0&\alpha_{yz}\\-\alpha_{xz}&-\alpha_{yz}&0\end{bmatrix} + \begin{bmatrix}\delta_{xx}&0&0\\0&\delta_{yy}&0\\0&0&\delta_{zz}\end{bmatrix}$$

Rank 2 spatial tensors are commonly specified in their principal axis system by the three components; the isotropic value A_{iso}, the anisotropy ΔA, and the asymmetry η. These are generally given by

$$A_{iso} = \frac{1}{3}Tr\{A\}\ , \qquad \Delta A = A_{zz} - \frac{1}{2}(A_{xx} + A_{yy}) = \frac{3}{2}\delta_{zz} \qquad \eta = (\delta_{xx} - \delta_{yy})/\delta_{zz}$$

A set of Euler angles $\{\alpha, \beta, \gamma\}$ is normally also given to relate the spatial tensor principle axes to another coordinate system. For the shielding spatial tensor we have

$$\hat{\sigma}(PAS) = \sigma_{iso}\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix} + \begin{bmatrix}0&\alpha_{xy}&\alpha_{xz}\\-\alpha_{xy}&0&\alpha_{yz}\\-\alpha_{xz}&-\alpha_{yz}&0\end{bmatrix} + \delta_{zz}\begin{bmatrix}-\frac{1}{2}(1-\eta)&0&0\\0&-\frac{1}{2}(1+\eta)&0\\0&0&1\end{bmatrix} \tag{1-3}$$

The irreducible spherical elements of the shielding tensor, $A^{CS}_{2,m}$, in the principal axis system are, by placement of (4-35) into (1-2),

$$\sigma_{0,0}(PAS) = -\sqrt{3}\sigma_{iso}$$

$$\sigma_{1,0}(PAS) = -\frac{i}{\sqrt{2}}[\sigma_{xy} - \sigma_{yx}] \qquad \sigma_{1,\pm1}(PAS) = -\frac{1}{2}[(\sigma_{zx} - \sigma_{xz}) \pm i(\sigma_{zy} - \sigma_{yz})]$$

$$\sigma_{2,0}(PAS) = \sqrt{3/2}\delta_{zz} \qquad \sigma_{2,1}(PAS) = V_{2,-1}(PAS) = 0$$

$$\sigma_{2,2}(PAS) = V_{2,-2}(PAS) = \frac{1}{2}\delta_{zz}\eta$$

1. The principal axis system is set such that $|\delta_{zz}| \geq |\delta_{yy}| \geq |\delta_{xx}|$. The orientation of the x and y axes are inconsequential if η is zero.

3.15.11 Scaled Shielding Spherical Spatial Tensor PAS Components

Throughout GAMMA, we desire all irreducible spherical rank 2 spatial components to be scaled so as they are independent of the particular interaction. To do so we adjust them to be as similar to normalized spherical harmonics as possible. Thus, we here scale the shielding irreducible ranke 2 spatial tensor so that the $2, 0$ component will have the same magnitude as the $m = 0$ rank two spherical harmonic when the two spherical angles are set to zero. Our "normalization" factor "X" is obtained by

$$A_{2,0}(\theta,\varphi)\big|_{\theta=\varphi=0} = X^{SA} \bullet \sigma(\theta,\varphi)\big|_{\theta=\varphi=0} = Y_{2,0}(\theta,\varphi)\big|_{\theta=\varphi=0} = \sqrt{5/(4\pi)}$$

We thus define the GAMMA shift anisotropy spatial tensor scaled such that

$$A^{SA}_{l,m} = \sqrt{5/(6\pi)}\delta^{-1}_{zz}\sigma^{SA}_{l,m} \tag{2}$$

and the components are given in the next figure.

GAMMA Scaled Shift Anisotropy Spatial Tensor PAS Components

$$A^{SA}_{2,0}(PAS) = \sqrt{\frac{5}{4\pi}} \qquad A^{SA}_{2,\pm1}(PAS) = 0 \qquad A^{SA}_{2,\pm2}(PAS) = \sqrt{\frac{5}{24\pi}}\eta$$

The scaling factor $\sqrt{5/(6\pi)}\delta^{-1}_{zz}$ which was multiplied into the "σ" components will be compensated for in the shielding anisotropy interaction constant. The shielding anisotropy Hamiltonian given in equation (23) becomes

$$\boldsymbol{H}^{SA} = h\gamma B_o\sqrt{\frac{6\pi}{5}}\sum_{m}^{\pm1}(-1)^m A^{SA}_{2,-m} \bullet \boldsymbol{T}^{SA}_{2,m} \tag{3}$$

3.15.12 Shielding Anisotropy Interaction Constant

In GAMMA, since we have defined our spatial and spin tensors to be scaled independent of the type of interaction, we use an interaction constant as a scaling when formulating Hamiltonians. Shielding anisotropy Hamiltonians may be produced from

$$\boldsymbol{H}^{SA} = \xi^{SA}\sum_{m}^{\pm1}(-1)^m A^{SA}_{2,-m}\boldsymbol{T}^{SA}_{2,m} = \sqrt{\frac{6\pi}{5}}h\gamma B_o\delta_{zz}\sum_{m}^{\pm1}(-1)^m A^{SA}_{2,-m}\boldsymbol{T}^{SA}_{2,m} \tag{4}$$

so evidently

$$\xi^{SA} = \sqrt{\frac{6\pi}{5}}h\gamma B_o\delta_{zz} \tag{5}$$

Such interaction constants are not very common in the literature (except with regards to some papers treating relaxation in liquids) and thus not intuitive to many GAMMA users. So, one simply needs to be aware of the relationships between the interaction constant and any commonly used shift anisotropy definitions. One common quantity is the CSA, $\Delta\sigma$, the chemical shift (or shielding) anisotropy[1].

$$QCC = e^2qQ = \frac{2I(2I_i-1)\omega^Q}{3} \qquad \omega^Q = \frac{3e^2qQ}{2I(2I-1)} = \frac{3QCC}{2I(2I-1)} = \sqrt{\frac{15}{2\pi}}\omega^Q$$

The former is often labeled as $NQCC$, an anacronym for Nuclear shift anisotropy Coupling Constant. There are many definitions in the literature for the latter. In GAMMA we chose the definition so that this frequency will be the distance between transitions when the shift anisotropy Hamiltonian is a small perturbation to the Zeeman Hamiltonian (i.e. when a spin's Larmor frequency is much higher than its shift anisotropy coupling constant).

As for the shift anisotropy interaction constant we have

$$\xi^Q = \sqrt{\frac{6\pi}{5}}\frac{e^2qQ}{2I(2I-1)} = \sqrt{\frac{6\pi}{5}}\frac{QCC}{2I(2I-1)} = \sqrt{\frac{2\pi}{15}}\omega^Q \tag{6}$$

We can express the spatial tensor components $A^{CS}_{l,m}$ relative to any arbitrary axis system (AAS) by a rotation from the principal axes to the new axes *via* the formula

$$A^{CS2}_{l,m}(i, AAS) = \sum_{m'}^{\pm l} D^l_{mm'}(\Omega)A^{CS2}_{l,m'}(i, PAS) \tag{6-1}$$

where $D^l_{mm'}$ are the rank l Wigner rotation matrix elements and Ω the set of three Euler angles which relate the principal axes of the chemical shielding tensor to the arbitrary axes[2]. Unlike the dipolar Hamiltonian treatment which only had a rank 2 component, components of ranks l = 0, 1, and 2 may contribute to the shielding Hamiltonian. Since these ranks behave differently under rotations we shall write the overall shielding Hamiltonian to reflect this. Beginning with equation (1-1)

$$\boldsymbol{H}^{CS}_i = h\gamma_i B_o \sum_{l=0}^{2}\sum_{m}^{\pm l}(-1)^m A^{CS2}_{l,-m}(i) \bullet \boldsymbol{T}^{CS2}_{l,m}(i)$$

1. In angular frequency units this is $QCC = e^2qQ/h$ where Q is the quadrupole moment. Note that, although the definition of QCC is standardized, there seems to be some variation in the literature as to what the shift anisotropy splitting frequency ω^Q is.

2. In this instance we must be careful to express the elements $\boldsymbol{T}^{CS2}_{l,-m}$ in the same axis system as $A^{CS2}_{l,m}$. When A^{CS2} is rotated in space, so must be $\boldsymbol{T}^{CS2}$. Essentially the field vector changes relative to any new coordinate system when constructing $\boldsymbol{T}^{CS2}$.

we define a chemical shielding interaction constant as

$$\xi_i^{CS} = h\gamma_i B_o \tag{6-2}$$

and expand the summation over the different ranks.

$$\boldsymbol{H}_i^{CS} = \xi_i^{CS}\left[A_{0,0}^{CS2}(i)\boldsymbol{T}_{0,0}^{CS2}(i) + \sum_m^{\pm 1}(-1)^m A_{1,-m}^{CS2}(i)\boldsymbol{T}_{1,m}^{CS2}(i) + \sum_m^{\pm 2}(-1)^m A_{2,-m}^{CS2}(i)\boldsymbol{T}_{2,m}^{CS2}(i)\right]$$

In other words we now have

$$\boldsymbol{H}_i^{CS} = \boldsymbol{H}_i^{CSI} + \boldsymbol{H}_i^{CSU} + \boldsymbol{H}_i^{CSA}. \tag{6-3}$$

There is good reason to separate these terms. The rank 0 component of the shielding Hamiltonian is rotationally invariant and called the isotropic chemical shielding Hamiltonian. In high-resolution NMR it is normally included in the static Hamiltonian $\boldsymbol{H}_0$. The rank 2 part is call the chemical shielding anisotropy Hamiltonian. In liquid systems this Hamiltonian averages to zero and thus not affect observed shielding values. It will contribute to relaxation of the system. On the other hand, in solid systems this component does not average away and will partially determine peak shapes in powder averages. The rank 1 component is the antisymmetric part of the shielding Hamiltonian. Since the antisymmetric part of the shielding tensor is difficult to measure, this part of the shielding Hamiltonian is usually assumed small and neglected.

The isotropic component ($l = 0$) of the chemical shielding Hamiltonian is thus written

$$\boldsymbol{H}_i^{CSI}(AAS) = \xi_i^{CS} A_{0,0}^{CS2}(i, AAS) \bullet \boldsymbol{T}_{l,-m}^{CS2}(i, AAS) \quad , \tag{6-4}$$

the antisymmetric component ($l = 1$) of the chemical shielding Hamiltonian is

$$\boldsymbol{H}_i^{CSU}(AAS) = \xi_i^{CS}\sum_m^{\pm 1}(-1)^m A_{1,-m}^{CS2}(i, AAS) \bullet \boldsymbol{T}_{1,m}^{CS2}(i, AAS) \quad , \tag{6-5}$$

and the anisotropic component ($l = 2$) of the chemical shielding Hamiltonian is

$$\boldsymbol{H}_i^{CSA}(AAS) = \xi_i^{CS}\sum_m^{\pm 2}(-1)^m A_{2,m}^{CS2}(i, AAS) \bullet \boldsymbol{T}_{2,-m}^{CS2}(i, AAS) \tag{6-6}$$

Throughout GAMMA, we desire all rank 2 spatial tensor irreducible spherical components to be similar to rank 2 normalized spherical harmonics if possible. Thus, we here scale the shielding spatial tensor such that the components $A_{2,m}^{CS2}$ will become normalized rank two spherical harmonics when the asymmetry term is zero, $\eta^{CS} = 0$. Thus our aim is to use the following spherical tensor

to define the spatial chemical shielding tensor.

$$[A_{2,m}^{CSA} = KA_{2,m}^{CS2}]_{\eta=0} = Y_{2,m} \tag{6-7}$$

Now application of equation (6-7) on the $l = 2$, $m = 2$ component reveals the value of the constant K

$$A_{2,0}^{CSA}\Big|_{\eta=0} = KA_{2,0}^{CS2} = K\sqrt{3/2}\delta_{zz} = Y_{2,0}\Big|_{\substack{\theta=0\\ \varphi=0}} = \sqrt{\frac{5}{4\pi}}$$

$$K\sqrt{3/2}\delta_{zz} = \sqrt{\frac{5}{4\pi}} \qquad K = \sqrt{\frac{5}{6\pi}\delta_{zz}^{-1}}$$

and our scaled chemical shielding spatial tensor is then

$$A_{2,m}^{SA} = \sqrt{\frac{5}{6\pi}}\delta_{zz}^{-1}A_{l,m}^{CS2}$$

The (rank 2) components $A_{l,m}^{CSA}$ in the principal axis system are

$$A_{2,0}^{CSA}(PAS) = \sqrt{\frac{5}{4\pi}} \qquad A_{2,\pm1}^{CSA}(PAS) = 0 \qquad A_{2,\pm2}^{CSA}(PAS) = \sqrt{\frac{5}{24\pi}}\eta$$

The anisotropic component ($l = 2$) of the chemical shielding Hamiltonian, equation (6-6), is then equivalently expressed by

$$\boldsymbol{H}_i^{CSA}(AAS) = \xi_i^{CSA}\sum_m^{\pm2}(-1)^m A_{2,m}^{CSA}(i, AAS) \bullet \boldsymbol{T}_{2,-m}^{CS2}(i, AAS) \tag{6-8}$$

where we have define the chemical shielding anisotropy interaction constant to take into account that we have scaled the spatial tensor components by the factor K^{-1}.

$$\xi^{SA} = \sqrt{\frac{6\pi}{5}}h\gamma_i B_o\delta_{zz} \tag{6-9}$$

3.15.13 Shielding Anisotropy Interaction Constant

In GAMMA, since we have defined our spatial and spin tensors to be scaled independent of the type of interaction, we use an interaction constant as a scaling when formulating Hamiltonians. Shielding anisotropy Hamiltonians may be produced from

$$\boldsymbol{H}^{SA} = \xi^{SA}\sum_m^{\pm1}(-1)^m A_{2,-m}^{SA}\boldsymbol{T}_{2,m}^{SA} \tag{7}$$

as we have already seen

$$\boldsymbol{H}^{SA} = \sqrt{\frac{6\pi}{5}} h\gamma B_o \delta_{zz} \sum_{m}^{\pm 1} (-1)^m A^{SA}_{2,-m} \boldsymbol{T}^{SA}_{2,m} \tag{8}$$

so evidently

$$\xi^{SA} = \sqrt{\frac{6\pi}{5}} h\gamma B_o \delta_{zz} \tag{9}$$

Such interaction constants are not very common in the literature (except with regards to some papers treating relaxation in liquids) and thus not intuitive to many GAMMA users. So, one simply needs to be aware of the relationships between the interaction constant and commonly used shift anisotropy definitions. Two common quantities are the chemical shift anisotropy[1] QCC and the shift anisotropy frequency ω^Q.

$$QCC = e^2qQ = \frac{2I(2I_i - 1)\omega^Q}{3} \qquad \omega^Q = \frac{3e^2qQ}{2I(2I-1)} = \frac{3QCC}{2I(2I-1)} = \sqrt{\frac{15}{2\pi}}\omega^Q$$

The former is often labeled as $NQCC$, an anacronym for Nuclear shift anisotropy Coupling Constant. There are many definitions in the literature for the latter. In GAMMA we chose the definition so that this frequency will be the distance between transitions when the shift anisotropy Hamiltonian is a small perturbation to the Zeeman Hamiltonian (i.e. when a spin's Larmor frequency is much higher than its shift anisotropy coupling constant).

As for the shift anisotropy interaction constant we have

$$\xi^Q = \sqrt{\frac{6\pi}{5}} \frac{e^2qQ}{2I(2I-1)} = \sqrt{\frac{6\pi}{5}} \frac{QCC}{2I(2I-1)} = \sqrt{\frac{2\pi}{15}}\omega^Q \tag{10}$$

and the chemical shielding Hamiltonian for a single spin becomes

$$\boldsymbol{H}^{CS}_i = \boldsymbol{H}^{CSI}_i + \boldsymbol{H}^{CSU}_i + \boldsymbol{H}^{CSA}_i$$

$$\boldsymbol{H}^{CS}_i = \xi^{CS}_i \left[A^{CS2}_{00}(i)\boldsymbol{T}^{CS2}_{00}(i) + \sum_{m}^{\pm 1} (-1)^m A^{CS2}_{1-m}(i)\boldsymbol{T}^{CS2}_{1m}(i) \right] + \xi^{CSA}_i \sum_{m}^{\pm 2} (-1)^m A^{CSA}_{2-m}(i)\boldsymbol{T}^{CS2}_{2m}(i)$$

When working with an entire spin system one must sum over all spins with the tensors being in the same coordinate system, for our purposes the laboratory system. The chemical shielding Hamilto-

1. In angular frequency units this is $QCC = e^2qQ/h$ where Q is the quadrupole moment. Note that, although the definition of QCC is standardized, there seems to be some variation in the literature as to what the shift anisotropy splitting frequency ω^Q is.

nian for a spin system becomes the following.

$$\boldsymbol{H}^{CS} = \sum_{i}^{spins} \boldsymbol{H}_i^{CS} = \sum_{i}^{spins} \xi_i^{CS} \sum_{l=0}^{2} \sum_{m}^{\pm l} (-1)^m A_{l,-m}^{CS2}(i, AAS) \bullet \boldsymbol{T}_{l,m}^{CS2}(i, AAS) \tag{10-1}$$

The following figures summarize the rank 2 treatment of the shielding Hamiltonian.

3.15.14 Shielding Anisotropy Hamiltonian

Chemical shielding will affect the observed resonance frequency. The isotropic (rank 0) contribution to the shielding is normally included with the Zeeman Hamiltonian to form the isotropic chemical shift Hamiltonian. The anti-symmetric (rank 1) contribution to shielding is rarely observed. The symmetric rank 2 contribution to the chemical shielding interaciton, that which we are concerned with in class IntSA, produces the following Hamiltonian[1].

$$\boldsymbol{H}^{SA} = \sqrt{\frac{6\pi}{5}} h\gamma B_o \delta_{zz} \sum_m^{\pm 2} (-1)^m A^{SA}_{2,-m} \bullet \boldsymbol{T}^{SA}_{2,m} = \xi^{SA} \sum_m^{\pm 2} (-1)^m A^{SA}_{2,-m} \bullet \boldsymbol{T}^{SA}_{2,m}$$

We have simplified (and standardized) our nomenclature by defining a shielding anisotropy interaction constant as

$$\xi^{SA} = \sqrt{\frac{6\pi}{5}} h\gamma B_o \delta_{zz} \tag{11}$$

Note that in the principal axis system (PAS) when the field is oriented along the +z axis, the shift anisotropy Hamiltonian is given by a relatively simple formula because both the $A^{SA}_{2,\pm 1}$ and the $T^{SA}_{2,\pm 2}$ terms are zero.

$$\begin{aligned} \boldsymbol{H}^{SA}(PAS) &= \xi^{SA} \sum_m^{\pm 2} (-1)^m A^{SA}_{2,-m} \bullet \boldsymbol{T}^{SA}_{2,m} \\ &= \xi^{SA} A^{SA}_{2,0}(PAS) \boldsymbol{T}^{SA}_{2,0} = \left(\sqrt{\frac{6\pi}{5}} h\gamma B_o \delta_{zz}\right)\left(\sqrt{\frac{5}{4\pi}}\right)\left(\frac{2}{\sqrt{6}} \boldsymbol{I}_z\right) = h\gamma B_o \delta_{zz} \boldsymbol{I}_z \end{aligned} \tag{12}$$

When the shift anisotropy interaction is oriented relative to its principal axes the Hamiltonian equation becomes much more complicated than the one above.

$$\begin{aligned} \boldsymbol{H}^{SA}(\theta,\varphi) &= \xi^{SA} \sum_m^{\pm 2} (-1)^m A^{Q}_{2,-m}(\theta,\varphi) \bullet \boldsymbol{T}^{Q}_{2,m} \\ &= \xi^{SA}[A^{SA}_{2,0}(\theta,\varphi)\boldsymbol{T}^{SA}_{2,0} + A^{SA}_{2,1}(\theta,\varphi)\boldsymbol{T}^{SA}_{2,-1} + A^{SA}_{2,-1}(\theta,\varphi)\boldsymbol{T}^{SA}_{2,1}] \\ &= \xi^{SA}[A^{SA}_{2,0}(\theta,\varphi)\boldsymbol{T}^{SA}_{2,0} + A^{SA}_{2,1}(\theta,\varphi)\boldsymbol{T}^{SA}_{2,-1} - A^{SA}_{2,1}{}^{*}(\theta,\varphi)\boldsymbol{T}^{SA}_{2,1}] \\ &= \xi^{SA}\{A^{SA}_{2,0}(\theta,\varphi)\boldsymbol{T}^{SA}_{2,0} + Re[A^{SA}_{2,1}(\theta,\varphi)](\boldsymbol{T}^{SA}_{2,-1} - \boldsymbol{T}^{SA}_{2,1}) + iIm[A^{SA}_{2,1}(\theta,\varphi)](\boldsymbol{T}^{SA}_{2,-1} + \boldsymbol{T}^{SA}_{2,1})\} \end{aligned}$$

At this point we will substitute in the spin operatiors (assuming Bo is along +z)

$$T^{SA}_{2,0} = \frac{2}{\sqrt{6}} \boldsymbol{I}_z \qquad T^{SA}_{2,\pm 1} = \mp\frac{1}{2} \boldsymbol{I}_\pm$$

This produces

1. Keep in mind that this Hamiltonian is for a single spin of quantum number I. In a multi-spin system one will have to sum such Hamiltonians for all spins.

$$\boldsymbol{H}^{SA}(\theta,\varphi) = \xi^{SA}\{A^{SA}_{2,0}(\theta,\varphi)\boldsymbol{T}^{SA}_{2,0} + Re[A^{SA}_{2,1}(\theta,\varphi)](\boldsymbol{T}^{SA}_{2,-1} - \boldsymbol{T}^{SA}_{2,1}) + iIm[A^{SA}_{2,1}(\theta,\varphi)](\boldsymbol{T}^{SA}_{2,-1} + \boldsymbol{T}^{SA}_{2,1})\}$$

$$= \xi^{SA}\left\{A^{SA}_{2,0}(\theta,\varphi)\left[\frac{2}{\sqrt{6}}\boldsymbol{I}_z\right] + Re[A^{SA}_{2,1}(\theta,\varphi)]\frac{1}{2}[(I_- + I_+)] + iIm[A^{SA}_{2,1}(\theta,\varphi)]\frac{1}{2}[(I_- - I_+)]\right.$$

We can use the identities $I_x = \frac{1}{2}(I_- + I_+)$ $\quad I_y = \frac{i}{2}(I_- - I_+)$ to obtain

$$\boldsymbol{H}^{SA}(\theta,\varphi) = \xi^{SA}\left\{A^{SA}_{2,0}(\theta,\varphi)\left[\frac{2}{\sqrt{6}}\boldsymbol{I}_z\right] + Re[A^{SA}_{2,1}(\theta,\varphi)]I_x + Im[A^{SA}_{2,1}(\theta,\varphi)]I_y\right\}$$

Upon substitution of the oriented spatial components we obtain

$$\boldsymbol{H}^{SA}(\theta,\varphi) = \xi^{SA}\left\{\sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]\left[\frac{2}{\sqrt{6}}\boldsymbol{I}_z\right]\right.$$

$$\left. + \left[\sqrt{\frac{5}{24\pi}}\sin\theta[3\cos\theta - \eta(\cos\theta\cos 2\varphi)]\right]I_x + \left[\sqrt{\frac{5}{24\pi}}\sin\theta\eta\sin 2\varphi\right]I_y\right\}$$

$$\boldsymbol{H}^{SA}(\theta,\varphi) = \frac{h\gamma B_o\delta_{zz}}{2}\{[3\cos^2\theta - 1 + \eta\sin^2\theta\cos 2\varphi]\boldsymbol{I}_z$$

$$+ \sin\theta[\cos\theta(3 - \eta\cos 2\varphi)\boldsymbol{I}_x + \eta\sin 2\varphi\boldsymbol{I}_y]\}$$

The Rank 2 Chemical Shielding Hamiltonian Summary

$$\boldsymbol{H}^{CS}(AAS) = \sum_{i}^{spins} \boldsymbol{H}_i^{CS}(AAS) = \sum_{i}^{spins} \xi_i^{CS} \sum_{l=0}^{2} \sum_{m}^{\pm l} (-1)^m A_{l-m}^{CS2}(i, AAS) \bullet \boldsymbol{T}_{lm}^{CS2}(i, AAS)$$

$$\boldsymbol{H}_i^{CS}(AAS) = \xi_i^{CS} \sum_{l=0}^{2} \sum_{m}^{\pm 2} (-1)^m A_{l-m}^{CS2}(i, AAS) \boldsymbol{T}_{lm}^{CS2}(i, AAS)$$

$$\xi_i^{CS} = h\gamma_i B_o$$

$$A_{l,m}^{CS2}(i, AAS) = \sum_{m'}^{\pm l} D_{mm'}^{l}(\varphi, \theta, \chi) A_{l,m'}^{CS2}(i, PAS)$$

$$A_{0,0}^{CS2}(i, PAS) = -\sqrt{3}\sigma_{iso}(i)$$

$$A_{1,0}^{CS2}(i, PAS) = -\frac{i}{\sqrt{2}}[\sigma_{xy}(i, PAS) - \sigma_{yx}(i, PAS)]$$

$$A_{1,\pm 1}^{CS2}(i, PAS) = -\frac{1}{2}[(\sigma_{zx}(i, PAS) - \sigma_{xz}(i, PAS)) \pm i(\sigma_{zy}(i, PAS) - \sigma_{yz}(i, PAS))]$$

$$A_{2,0}^{CS2}(i, PAS) = \sqrt{3/2}\delta_{zz}(i) \qquad A_{2,\pm 1}^{CS2}(i, PAS) = 0 \qquad A_{2,\pm 2}^{CS2}(i, PAS) = \frac{1}{2}\delta_{zz}(i)\eta(i)$$

$$T_{0,0}^{CS2}(i, AAS) = \frac{-1}{\sqrt{3}}\left[\boldsymbol{I}_{iz}B_z + \frac{1}{2}(\boldsymbol{I}_{i+}B_- + \boldsymbol{I}_{i-}B_+)\right] = \frac{-1}{\sqrt{3}}\vec{\boldsymbol{I}}_i \bullet \vec{\boldsymbol{B}}_n$$

$$T_{1,0}^{CS2}(i, AAS) = \frac{-1}{2\sqrt{2}}[\boldsymbol{I}_{i+}B_- - \boldsymbol{I}_{i-}B_+] \qquad T_{1,\pm 1}^{CS2}(i, AAS) = \frac{-1}{2}[\boldsymbol{I}_{i\pm}\ B_z - \boldsymbol{I}_{iz}B_{\pm}\ ]$$

$$T_{2,0}^{CS2}(i, AAS) = \frac{1}{\sqrt{6}}[3\boldsymbol{I}_{iz}B_z - (\vec{\boldsymbol{I}}_i \bullet \vec{\boldsymbol{B}}_n)]$$

$$T_{2,\pm 1}^{CS2}(i, AAS) = \mp\frac{1}{2}[\boldsymbol{I}_{i\pm}\ B_z + \boldsymbol{I}_{iz}B_{\pm}\ ] \qquad T_{2,\pm 2}^{CS2}(i, AAS) = \frac{1}{2}[\boldsymbol{I}_{i\pm}\ B_{\pm}\ ]$$

Although these equations are generally applicable, it is convenient to express the shielding Hamiltonian with clear separation between the different ranks (the components with differing values of l). The isotropic component $\boldsymbol{H}^{CSI}$ in the treatment of liquid samples will normally be placed into an overall isotropic Hamiltonian, H_0 because it does not disappear upon rotational averaging. The

asymmetric component, $\boldsymbol{H}^{CSU}$, is usually zero, the shielding tensor taken as essentially symmetric.

The Chemical Shielding Anisotropy Hamiltonian
Rank 2 Treatment

Arbitrary Axis System

$$\boldsymbol{H}^{SA}(AAS) = \xi_i^{SA}\sum_m^{\pm 2}(-1)^m A_{2-m}^{SA}(AAS)\boldsymbol{T}_{2m}^{SA}(AAS)$$

$$T_{2,0}^{SA}(SA) = \frac{1}{\sqrt{6}}[3\boldsymbol{I}_z B_z - (\vec{\boldsymbol{I}} \bullet \vec{\boldsymbol{B}}_n)]$$

$$T_{2,\pm 1}^{SA}(SA) = \mp\frac{1}{2}[\boldsymbol{I}_{i\pm}\ B_z + \boldsymbol{I}_{iz}B_{\pm}\ ] \qquad T_{2,\pm 2}^{SA}(SA) = \frac{1}{2}[\boldsymbol{I}_{i\pm}\ B_{\pm}\ ]$$

$$\xi_i^{CSA} = \sqrt{\frac{6\pi}{5}}\hbar\gamma_i B_o \delta_{zz}(i)$$

$$A_{2,0}^{CSA}(PAS) = \sqrt{\frac{5}{4\pi}} \qquad A_{2,\pm 1}^{CSA}(PAS) = 0 \qquad A_{2,\pm 2}^{CSA}(PAS) = \sqrt{\frac{5}{24\pi}}\eta$$

$$A_{2,m}^{CSA}(i, AAS) = \sum_{m'}^{\pm 2} D_{mm'}^{2}(\varphi, \theta, \chi)A_{2,m'}^{CSA}(i, PAS)$$

Laboratory Frame

$$\boldsymbol{H}_i^{CSA}(LAB) = \xi_i^{CSA}\sum_m^{\pm 1}(-1)^m A_{2-m}^{CSA}(i, LAB)\boldsymbol{T}_{2m}^{CS2}(i, LAB)$$

$$T_{2,0}^{CS2}(i, LAB) = \frac{2}{\sqrt{6}}\boldsymbol{I}_{iz} \qquad T_{2,\pm 1}^{CS2}(i, LAB) = \mp\frac{1}{2}\boldsymbol{I}_{i\pm} \qquad T_{2,\pm 2}^{CS2}(i, LAB) = 0$$

$$\xi_i^{CSA} = \sqrt{\frac{6\pi}{5}}\hbar\gamma_i B_o \delta_{zz}(i)$$

$$A_{2,0}^{CSA}(PAS) = \sqrt{\frac{5}{4\pi}} \qquad A_{2,\pm 1}^{CSA}(PAS) = 0 \qquad A_{2,\pm 2}^{CSA}(PAS) = \sqrt{\frac{5}{24\pi}}\eta$$

$$A_{2,m}^{CSA}(i, LAB) = \sum_{m'}^{\pm 2} D_{mm'}^{2}(\varphi_{PAS\to LAB}, \theta_{PAS\to LAB}, \chi_{PAS\to LAB})A_{2,m'}^{CSA}(i, PAS)$$

$$A_{2,m}^{CSA}(i, LAB)\Big|_{\eta = 0} = Y_{2,m}(\theta, \varphi)$$

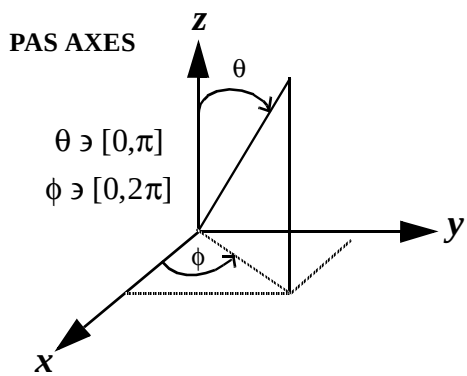
$$T_{2,0}^{CS2}(LAB) = \frac{-2}{\sqrt{6}}\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix} \qquad T_{2,1}^{CS2}(LAB) = \frac{-1}{2}\begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix} \qquad T_{2,-1}^{CS2}(LAB) = \frac{1}{2}\begin{bmatrix}0 & 0\\ 1 & 0\end{bmatrix} \qquad T_{2,\pm 2}^{CS2}(LAB) = \begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix}$$

3.15.15 shift anisotropy PAS Equations

When the shift anisotropy interaction has alignment along its principal axes system virtually all of the shift anisotropy equations simplify. The following figure collects all of these for convenience.

shift anisotropy Equations Involving the PAS

PAS AXES



$$\boldsymbol{H}^Q(PAS) = \xi^Q\sqrt{\frac{5}{24\pi}}\left[3I_z^2 - \vec{\boldsymbol{I}}^2 + \frac{1}{2}\eta(I_-^2 + I_+^2)\right]$$

$$= \frac{e^2qQ}{4I(2I-1)}\left[3I_z^2 - \vec{\boldsymbol{I}}^2 + \frac{1}{2}\eta(I_-^2 + I_+^2)\right]$$

$$= \frac{\omega^Q}{6}\left[3I_z^2 - \vec{\boldsymbol{I}}^2 + \frac{1}{2}\eta(I_-^2 + I_+^2)\right]$$

$$\xi^Q = \frac{e^2qQ}{2I(2I-1)}\sqrt{\frac{6\pi}{5}} = \frac{QCC}{2I(2I-1)}\sqrt{\frac{6\pi}{5}} = \sqrt{\frac{2\pi}{15}}\omega^Q$$

$$\eta = (A_{xx} - A_{yy})/A_{zz}$$

$$|A_{zz}| \geq |A_{yy}| \geq |A_{xx}|$$

$$A^Q_{2,0}(PAS) = \sqrt{6}[3A_{zz} - Tr\{A\}]\big|_{PAS} = \sqrt{\frac{5}{4\pi}}$$

$$A^Q_{2,\pm1}(PAS) = \mp\frac{1}{2}[A_{xz} + A_{zx} \pm i(A_{yz} + A_{zy})]\Big|_{PAS} = 0$$

$$A^Q_{2,\pm2}(PAS) = \frac{1}{2}[A_{xx} - A_{yy} \pm i(A_{xy} + A_{yx})]\big|_{PAS} = \sqrt{\frac{5}{24\pi}}\eta$$

$$A^Q_{xx}(PAS) = \sqrt{\frac{5}{24\pi}}[\eta - 1] \qquad A^Q_{yy}(PAS) = -\sqrt{\frac{5}{24\pi}}[1 + \eta] \qquad A^Q_{zz}(\theta, \varphi) = \sqrt{\frac{5}{6\pi}}$$

$$A^Q_{xz}(PAS) = 0 = A_{zx}(PAS) = A^Q_{xy}(PAS) = A_{yx}(PAS) = A_{yz}(PAS) = A_{zy}(PAS)$$

Figure 19-11 **Equations relevant to the shift anisotropy interaction in its principal axis orientation (PAS). GAMMA uses a spatial tensor which is scaled[1] so that rotations by angles θ & ϕ produce spherical harmonics for a symmetric interaction (η = 0).**

Included are the general relationships between the (GAMMA scaled) Cartesian tensor components to the irreducible spherical components. They are valid when η is defined accordingly! If η is defined by the other common convention ($|A_{zz}| \geq |A_{xx}| \geq |A_{yy}|$) then the sign on the $A^Q_{2,\pm2}$ will change as will the sign on the Hamiltonian term multiplied by η.

1. The scaling on both $\{A_{2m}\}$ and $T_{2m}\}$ are arbitrary, GAMMA uses an (uncommon) scaling which independent of the interaction type. What is NOT arbitrary is the scaling within either of the two sets of components. In addition, the combined scaling of the two sets is critical to the proper formation of shift anisotropy Hamiltonians. For that, GAMMA uses an interaction constant.

3.15.16 shift anisotropy Equations At Any Orientation

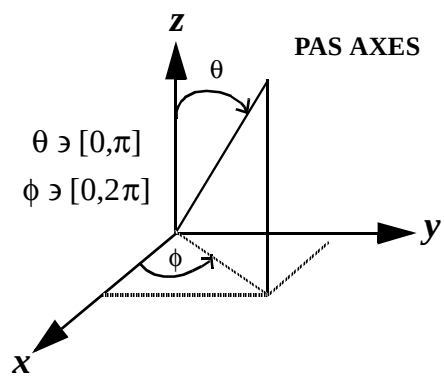
When the shift anisotropy interaction has a arbitrary alignment (relative to its principal axes system) the shift anisotropy equations become complicated. The figure below depicts them for convenience.

shift anisotropy Equations When Oriented At Angles {θ,ϕ}[1]

$$\boldsymbol{H}^{SA}(\theta,\varphi) = \xi^{SA}\sum_{m}^{\pm 2}(-1)^m A_{2,-m}(\theta,\varphi)\bullet \boldsymbol{T}^{Q}_{2,m}$$

$$T_{2,0} = \frac{1}{\sqrt{6}}[3I_z^2 - \check{\boldsymbol{I}}^2] = \frac{1}{\sqrt{6}}[3I_z^2 - I(I+1)]$$

$$T_{2,\pm 1} = \mp\frac{1}{2}[I_\pm\ I_z + I_z I_\pm\ ] \qquad T^{Q}_{2,\pm 2} = \frac{1}{2}I_\pm^2$$



$$A_{2,0}(\theta,\varphi) = \sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]$$

$$A_{2,1}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}\sin\theta[3\cos\theta - \eta(\cos\theta\cos 2\varphi - i\sin 2\varphi)] = -A_{2,-1}{}^*(\theta,\varphi)$$

$$A_{2,2}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}\frac{1}{2}[3\sin^2\theta + \eta[\cos 2\varphi(1+\cos^2\theta) - i2\sin 2\varphi\cos\theta]] = A_{2,-2}{}^*(\theta,\varphi)$$

$$\eta = (A_{xx} - A_{yy})/A_{zz} \qquad |A_{zz}| \geq |A_{yy}| \geq |A_{xx}| \qquad \xi^Q = \frac{e^2qQ}{2I(2I-1)}\sqrt{\frac{6\pi}{5}} = \sqrt{\frac{2\pi}{15}}\omega^Q$$

$$\boldsymbol{H}^Q(\theta,\varphi) = \frac{\omega^Q}{4}[\{3\cos^2\theta - 1 + \eta\sin^2\theta\cos 2\varphi\}(3I_z^2 - \check{\boldsymbol{I}}^2)$$
$$+ \frac{1}{2}[3\sin^2\theta + \eta\cos 2\varphi(1+\cos^2\theta)](I_+^2 + I_-^2) + i\eta\sin 2\varphi\cos\theta(I_+^2 - I_-^2)\Big]$$

Figure 19-12 **Equations relevant to the shift anisotropy Hamiltonian when oriented at angles θ & ϕ from the principal axis orientation (PAS). GAMMA uses a spatial tensor which is scaled[2] so that rotations by angles θ & ϕ produce spherical harmonics for a symmetric interaction (η = 0).**

1. The shift anisotropy interaction constant, as well as the relative scalings on the sets of spatial and spin tensors, can be adjusted as desired. However all components of the space or spin tensor must be adjusted by the same scaling. The GAMMA scaling is oriented to liquids where so that all spatial components are related to the spherical harmonics in the spatial tensor PAS.
2. The scaling on both $\{A_{2m}\}$ and $T_{2m}\}$ are arbitrary, GAMMA uses a scaling which independent of the interaction type. What is NOT arbitrary is the scaling within either of the two sets of components. In addition, the combined scaling of the two sets is also crucial. For that, GAMMA uses an interaction constant.

3.16 Shielding Anisotropy Interaction Parameters

This section describes how an ASCII file may be constructed that is self readable by a shielding anisotropy interaction. The file can be created with any editor and is read with the shielding anisotropy interaction member function “read” or some variant thereof. It is important to keep in mind the structure of a CSA interaction. Each will need the **set of** $\{\mathbf{I}, \delta_{zz}, \eta, \theta, \phi\}$ **specified**. Only the first value, **I**, is used to set the spin part of the interaction. The latter four are used to set the overall interaction strength and orientation.

3.16.1 Available Parameters

Of course, there are several ways of declaring a CSA interaction (e.g the CSA spatial tensor) other than with direct specification of the five values $\{\mathbf{I}, \delta_{zz}, \eta, \theta, \phi\}$. To accomodate different tensor nomenclature (i.e. spherical *versus* Cartesian, oriented *versus* PAS, etc.), GAMMA CSA interactions will recognize different sets of parameters! These are described in the following.

CSA Spin Quantum Number: CI, Iso

Specification of the CSA spin quantum number **I** can be accomplished with the parameter I or an Iso parameter. The value of I must be a positive integer multiple of 1/2 and may contain an interaction index appended to the name if multiple interactions are to be defined in the same file. Iso parameters must designate a valid spin isotope type. If both I & Iso values are set in the same file, I values will be preferentialy used to set up the CSA interaction.

CSA Interaction Spin Quantum Number Parameters

Parameter	Assumed Units	Examples Parameter (Type=1,3) : Value - Statement
CI	none	CI (1) : 1.5 - CSA 1st Spin Quantum Value
Iso(#)	none	Iso(0) (2) : 131Xe - Spin Isotope Type, I=3/2 for 131Xe Iso(1) (2) : 13C - Spin Isotope Type, I=1/2 for 13C

Figure 19-13 Shown are 2 possible parameters used to set CSA spin quantum numbers. Parameter type 1 indicates a double precision number parameter. Parameter type 2 indicates a string parameter. CI can have (#) appended, the number being an interaction index rather than spin index.

Although not recommended, CSA interactions can be set up without specifying **I** wherein they will be assigned default values of I=1/2 .

CSA Strength & Asymmetry: CSA, Ceta

Specification of the CSA interaction strength and asymmetry $\{\delta_{zz}, \eta\}$ can be accomplished either with the two parameters CSA and Ceta. The more commonly used CSA value ($\Delta\sigma$) is taken over the spatial tensor value δ_{zz} since the two related as $\Delta\sigma = (3/2)\delta_{zz}$. According to the standards used in GAMMA to define spatial tensors, the asymmetry value must be with the bounds [0, 1].

CSA Interaction Strength & AsymmetryParameters

Parameter	Assumed Units	Examples Parameter (Type=1,3) : Value - Statement
CSA	ppm	CSA (1) : 156.3 - Shift anisotropy (ppm)
Ceta	none	Ceta (1) : 0.33 - CSA asymmetry [0,1]

Figure 19-14 Shown are the parameters that may be used to set up CSA interaction strength and asymmetry. Parameter type 1 indicates a double precision number parameter.

There asymmetry will be assumed zero if not specified.

CSA Orientation: Ctheta, Cphi

Specification of the CSA interaction orientation $\{\theta, \phi\}$ can be accomplished either with the two parameters Dtheta and Dphi. These are specified in degrees.

CSA Interaction Spin Quantum Number Parameters

Parameter	Assumed Units	Examples Parameter (Type=1,3) : Value - Statement
Ctheta	Degrees	Ctheta(1) (1) : 90.0 - CSA angle down from +z (deg)
Cphi	Degrees	Cphi(1) (1) : 270.0 - CSA angle over from +x (deg)

Figure 19-15 Shown are the parameters that may be used to set CSA orientation. Parameter type 1 indicates a double precision number parameter.

If there is no angle designation specified for a particular interacion it will be taken to be zero. Note that the phi orientation if of no consequence if the asymmetry is zero.

3.16.2 Dipole Interaction Parameter Set 1

The simplest way to designate a CSA interaction is to direclty provide its I value, the shift anisotropy, the asymmetry, and the orientation - the 5 quantities that set**{I,** δ_{zz}**,** η**,** θ, ϕ**}** for the interaction.

CSA Interaction Parameter Set 1

Parameter	Units	Examples Parameter (Type) : Value - Statement
CI	none	CI (1) : 1.5 _ - Spin I Value
CSA	ppm	CSA (1) : 70.3 - Shift anisotropy (PPM)
Ceta	none	Ceta (1) : 0.33 - Shift asymmetry [0,1]
Ctheta	degrees	Ctheta (1) : 127.2 - Shift orientation from PAS z (deg)
Cphi	degrees	Cphi (1) : 270.9 - Shift orientation from PAS x(deg)

Figure 19-16 **Generic ASCII parameters to declare a GAMMA CSA interaction using a spin I value, shift anisotropy and asymmetry, and orientation angles. Parameter type=1 is a floating point value.**

By including these parameter statements (right column) in an ASCII file a GAMMA CSA interaction can be set with the ***read*** function. For example, the code below reads “file.asc”.

ASCII File Read With { CI, CSA, Ceta, Ctheta, Cphi }

file.asc

```
CI       (1) : 1.5       - Spin Quantum Number
CSA      (1) : 70.3      - CS anisotropy (PPM)
Ceta     (1) : 0.2       - CS asymmetry
Ctheta   (1) : 127.2     - CS Orientation from PAS z (deg)
Cphi     (1) : 270.9     - CSA Orientation from PAS x(deg)
CI(2)    (1) : 0.5       - Spin Quantum Number
CSA(2)   (1) : 10.3      - CS anisotropy (PPM)
Dtheta(2)  (1) : 0.0     - CS Orient. from PAS z (deg)
Dphi(2)  (1) : 270.9     - CS Orient. from PAS x(deg)
```

code.cc

```
.............
.............
IntCSA C;
C.read(“file.asc”);
IntCSA C2;
C2.read(“file.asc”, 2);
.............
.
```

Figure 19-17 **Specifying a CSA interaction using an external ASCII file. In this case the interactions are defined using spin quantum numbers in combination with CSA asymmetry and anisotropy values and CSA orientation angles.**

The above ASCII file is a GAMMA parameter set file that may contain additional lines of information and additional parameters. Things such as column spacing are not important - read about GAMMA parameters sets for full details. The (#) appended to the parameter names is used to allow for the definition of multiple interactions in the same ASCII file.

3.16.3 Dipole Interaction Parameter Set 2

Another way to designate a CSA interaction is to provide an isotope type, the shift anisotropy, the asymmetry, and the orientation - the 5 quantities that set**{I,** δ_{zz}**,** η**,** θ, ϕ**}** for the interaction. In this instance all parameter names should be indexed with an appended "(#)" where # is the interaction index.

CSA Interaction Parameter Set 2

Parameter	Units	Examples Parameter (Type) : Value - Statement
Iso	none	Iso(3) (2) : 1H _ - Spin Isotope Type
CSA	ppm	CSA(3) (1) : 90.3 - Shift anisotropy (PPM)
Ceta	none	Ceta(3) (1) : 1.03 - Shift asymmetry [0,1]
Ctheta	degrees	Ctheta(3) (1) : 127.2 - Shift orientation from PAS z (deg)
Cphi	degrees	Cphi(3) (1) : 270.9 - Shift orientation from PAS x(deg)

Figure 19-18 **Generic ASCII parameters to declare a GAMMA CSA interaction using a spin type, shift anisotropy and asymmetry, and orientation angles. Parameter type=1 is a floating point value and type=2 implies a string value.**

By including these parameter statements (right column) in an ASCII file a GAMMA CSA interaction can be set with the ***read*** function. For example, the code below reads "file1.asc".

ASCII File Read With { Iso, CSA, Ceta, Ctheta, Cphi }

file1.asc

```
Iso(3)     (2) : 19F      - Spin isotope type
CSA(3)     (1) : 70.3     - CS anisotropy (PPM)
Ceta(3)    (1) : 0.2      - CS asymmetry
Ctheta(3)  (1) : 127.2    - CS Orientation from PAS z (deg)
Cphi(3)    (1) : 270.9    - CSA Orientation from PAS x(deg)
Iso(2)     (2) : 131Xe    - Spin isotope type
CSA(2)     (1) : 10.3     - CS anisotropy (PPM)
Dtheta(2)  (1) : 0.0      - CS Orient. from PAS z (deg)
Dphi(2)    (1) : 270.9    - CS Orient. from PAS x(deg)
```

code1.cc

```
.............
.............
IntCSA C2;
C2.read("file1.asc", 2);
IntCSA C3;
C3.read("file1.asc", 3);
.............
.
```

Figure 19-19 **Specifying a CSA interaction using an external ASCII file. In this case the interactions are defined using spin isotope types in combination with CSA asymmetry and anisotropy values and CSA orientation angles.**

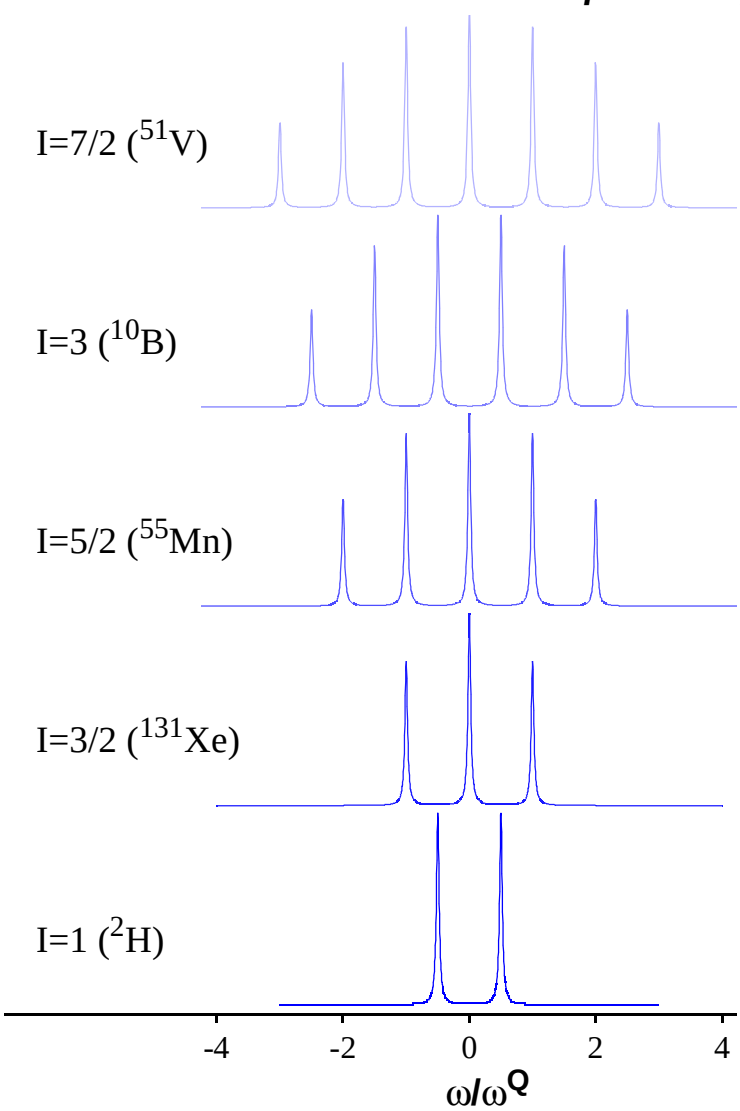
The above ASCII file is a GAMMA parameter set file that may contain additional lines of information and additional parameters. Things such as line ordering is not important - read about GAMMA parameters sets for full details. The (#) appended to the parameter names is used to allow for the definition of multiple interactions in the same ASCII file.

3.17 Examples

3.17.1 Zero Field Transitions, First Order Spectra

As a first example we'll look into some of the shift anisotropy Hamiltonians provided by class IntCSA in the interaction PAS (principal axes). Our results for both the transitions at zero field and NMR spectra to first order should agree with A. J. Vega's article[1] figures 1 & 2.

First Order shift anisotropy Spectra



I=7/2				
ω	$\pm 3\omega^Q$	$\pm 2\omega^Q$	$\pm \omega^Q$	0
I	3.5	6.0	7.5	8.0

I=3			
ω	$\pm 2.5\omega$	$\pm 1.5\omega$	$\pm 0.5\omega^Q$
I	3.0	5.0	6.0

I=5/2			
ω	$\pm 2\omega^Q$	$\pm \omega^Q$	0
I	2.5	4.0	4.5

I=3/2		
ω	$\pm \omega^Q$	0
I	1.5	2.0

I=1	
ω	$\pm 0.5\omega^Q$
I	1.0

Figure 19-20 **Spectra produced by program IntQu_LC6.cc, page -151. The shift anisotropy frequency was set to 300 kHz. The interaction was in its PAS and the asymmetry set to zero. Zero field transtions & relative intensities are shown in the ta-**

1. "shift anisotropy Nuclei in Solids", Alexander J. Vega, Encyclopedia of Nuclear Magnetic Resonance, Editors-in-Chief D.M.Grant and R.K. Harris, Vol. 6, Ped-Rel, pgs 3869-3889.

bles.

3.18 References

[3] D.M.Grant and R.K. Harris, Eds. in Chief, (1996), *Encyclopedia of Nuclear Magnetic Resonance*, John Wiley & Sons, New York.

[4] Brink, D.M. and Satchler, G.R. (1962), Angular Momentum, Clarendon Press, Oxford.

0.3 Programs and Input Files

IntQu_LC0.cc

```
/* IntQu_LC0.cc ************************************************************-*-c++-*-
**                                                                            **
**                        Test Program for the GAMMA Library                  **
**                 shift anisotropy Interaction Literature Comparison 0       **
**                                                                            **
**  This program checks the shift anisotropy interaction class IntCSA in      **
**  GAMMA.  In particular it looks to see how well the class parallels        **
**  the articles by Pascal P. Man                                             **
**                                                                            **
**  "shift anisotropy Interactions", Encyclopedia of Magnetic Resonance,      **
**  by Grant and Harris, Vol 6, Ped-Rel, page 3838-3869.                      **
**                                                                            **
**  and Alexander Vega                                                        **
**                                                                            **
**  "shift anisotropy Nuclei in Solids", Encyclopedia of Magnetic Resonance,  **
**  by Grant and Harris, Vol 6, Ped-Rel, page 3869-3889.                      **
**                                                                            **
**  In particular, their PAS shift anisotropy Hamiltonians are generated and  **
**  compared with the shift anisotropy interaction class Hamiltonians.        **
**                                                                            **
**  Man's Hamiltonians will be generated from equations in (5) on page        **
**  3839 of the his article.  Vega's Hamiltonians will be made from           **
**  equations (28), (32) and (33) of his article.  Note that his (32)         **
**  is missing a factor of 1/3 on the <1|H|3> and <3|H|1> components.         **
**                                                                            **
** Author:      S.A. Smith                                                   **
** Date:       10/11/96                                                      **
** Update:      10/11/96                                                     **
** Version:     3.6                                                          **
** Copyright:   S. Smith.  You can modify this program for personal use, but **
**                                 you must leave it intact if you re-distribute it.  **
**                                                                            **
******************************************************************************/

#include <gamma.h>                          // Include GAMMA

main (int argc, char* argv[])
{
//                      Set Up The shift anisotropy Interaction

  int qn=1;
  double I;
  query_parameter(argc, argv, qn++,                 // Read in the coupling
                  "\n\tSpin Quantum Number? ", I);
  double W;                                         // shift anisotropy frequency
  query_parameter(argc, argv, qn++,                 // Read in the coupling
    "\n\tshift anisotropy Frequency(kHz)? ", W);
  W *= 1.e3;                                        // Put this in Hz
  double Qeta;
  query_parameter(argc, argv, qn++,                 // Read in the coupling
    "\n\tshift anisotropy Asymmetry [0, 1]? ", Qeta);
//                  Construct GAMMA shift anisotropy Interaction

  IntCSA Q(I,wQ2QCC(W,I),Qeta,0.0,0.0);
//              Here are the Operators To Build Man's Hamiltonians

  int Ival = int(2.*I + 1);             // For 1 spin SOp functions
  matrix IE = Ie(Ival);                 // The operator 1
  matrix IM = Im(Ival);                 // The operator I-
  matrix IP = Ip(Ival);                 // The operator I+
  matrix IZ = Iz(Ival);                 // The operator Iz
  matrix IX = Ix(Ival);                 // The operator Ix
  matrix IY = Iy(Ival);                 // The operator Iy
//                  Here's The H Accoring To Man's Equation (5a)
//                    (Note That His W is Half Of Our Definition)
  matrix HMa = 3.0*IZ*IZ - (I*(I+1))*IE + Qeta*((IX*IX)-(IY*IY));
  HMa *= (W/6.0);
//                  Here's The H Accoring To Man'c Equation (5c)
//                    (Note That His W is Half Of Our Definition)
  matrix HMb = 3.0*IZ*IZ - (I*(I+1))*IE + (Qeta/2.)*((IP*IP)+(IM*IM));
  HMb *= (W/6.0);
//                        Here's The H According To GAMMA
  matrix HG = Q.H();
//                      Here's The H Also According To GAMMA
  matrix HGB = Q.H(0.0, 0.0);
//            Here Are Vegas V's According To Equations (22-27, 31)
//          (Switches eta Sign To Account For Opposite PAS Definition)
  double Eta = -Q.eta();
  double Vxx = 0.5*(- 1. - Eta);
  double Vyy = 0.5*(-1. + Eta);
  double Vzz = 1.0;
  double Vxy = 0.0;
  double Vxz = 0.0;
  double Vyz = 0.0;
  complex V1(-Vxz, -Vyz);
  complex Vm1(Vxz, -Vyz);
  complex V2(0.5*(Vxx-Vyy), Vxy);
  complex Vm2(0.5*(Vxx-Vyy), -Vxy);
//        Generate H According To Vega's Equations (32) Or (33)
  matrix HVega;
  if(I == 1)
    {
```

```
  HVega = matrix(3,3);
  HVega.put(Vzz/6.0, 0, 0);
  HVega.put(Vm1/sqrt(2.0), 0, 1);
  HVega.put(Vm2/3., 0, 2);                    // Added 1/3 Factor!
  HVega.put(-V1/sqrt(2.0), 1, 0);
  HVega.put(-Vzz/3.0, 1, 1);
  HVega.put(-Vm1/sqrt(2.0), 1, 2);
  HVega.put(V2/3., 2, 0);                     // Added 1/3 Factor!
  HVega.put(V1/sqrt(2.0), 2, 1);
  HVega.put(Vzz/6.0, 2, 2);
  HVega *= Q.wQ();
  }
else if(I == 1.5)
  {
  HVega = matrix(4,4);
  HVega.put(Vzz/2.0, 0, 0);
  HVega.put(Vm1/sqrt(3.0), 0, 1);
  HVega.put(Vm2/sqrt(3.0), 0, 2);
  HVega.put(0.0, 0, 3);
  HVega.put(-V1/sqrt(3.0), 1, 0);
  HVega.put(-Vzz/2.0, 1, 1);
  HVega.put(0.0, 1, 2);
  HVega.put(Vm2/sqrt(3.0), 1, 3);
  HVega.put(V2/sqrt(3.0), 2, 0);
  HVega.put(0.0, 2, 1);
  HVega.put(-Vzz/2.0, 2, 2);
  HVega.put(-Vm1/sqrt(3.0), 2, 3);
  HVega.put(0.0, 3, 0);
  HVega.put(V2/sqrt(3.0), 3, 1);
  HVega.put(V1/sqrt(3.0), 3, 2);
  HVega.put(Vzz/2.0, 3, 3);
  HVega *= Q.wQ();
  }
//                    Generate H According To Vega's Equation (28)
 matrix HV = Vzz*(3.*IZ*IZ-(I*(I+1.))*IE);
 HV += (Vxx-Vyy)*(IX*IX-IY*IY);
 HV += 2*Vxy*(IX*IY-IY*IX);
 HV += 2*Vxz*(IX*IZ-IZ*IX);
 HV += 2*Vyz*(IY*IZ-IZ*IY);
 HV *= Q.wQ()/6.0;
//                    Output the Results for Visual Comparison
 cout << "\n\t\t\t\tGAMMA's shift anisotropy H:\t" << HG;
 cout << "\n\t\t\t\tGAMMA's Other shift anisotropy H:\t" << HGB;
 cout << "\n\t\t\t\tMan's shift anisotropy H(a):\n\t" << HMa;
 cout << "\n\t\t\t\tMan's shift anisotropy H(b):\n\t" << HMb;
 if(I == 1.0 || I == 1.5)
   cout << "\n\t\t\t\tVega's shift anisotropy H:\n\t" << HVega;
 cout << "\n\t\t\t\tVega's Generic Quad H:\n\t" << HV;
 }
```

IntQu_LC1.cc

```
/* IntQu_LC1.cc ************************************************************-*-c++-*-
**                                                                            **
**                    Test Program for the GAMMA Library                      **
**              shift anisotropy Interaction Literature Comparison 1          **
**                                                                            **
** This program checks the shift anisotropy interaction class IntCSA in       **
** GAMMA.  In particular it looks to see how well the class parallels         **
** the article by Alexander Vega -                                            **
**                                                                            **
** "shift anisotropy Nuclei in Solids", Encyclopedia of Magnetic Resonance,   **
** by Grant and Harris, Vol 6, Ped-Rel, page 3869-3889.                       **
**                                                                            **
** Specifcally, herein we generate the spatial tensor components of           **
** an oriented shift anisotropy interaction and and compare the results to    **
** A.Vega's equations (22-27) and 31 on pages 3884-3885.                      **
**                                                                            **
** Author:     S.A. Smith                                                     **
** Date:       10/11/96                                                       **
** Update:     10/11/96                                                       **
** Version:    3.6                                                            **
** Copyright:  S. Smith.  You can modify this program as you see fit          **
**             for personal use, but you must leave the program intact        **
**             if you re-distribute it.                                       **
**                                                                            **
*****************************************************************************/

#include <gamma.h>                          // Include GAMMA

main (int argc, char* argv[])

 {
//                    Construct A shift anisotropy Interaction
 int qn=1;
 double W;                                  // shift anisotropy frequency
 query_parameter(argc, argv, qn++,          // Read in the coupling
        "\n\tshift anisotropy Frequency(kHz)? ", W);
 W *= 1.e3;                                 // Put this in Hz
 double Qeta;
 query_parameter(argc, argv, qn++,          // Read in the coupling
   "\n\tshift anisotropy Asymmetry [0, 1]? ", Qeta);
 double Qtheta, Qphi;
 query_parameter(argc, argv, qn++,          // Read in the angle
   "\n\tAngle down from z [0, 180]? ", Qtheta);
 query_parameter(argc, argv, qn++,          // Read in the angle
    "\n\tAngle over from x [0, 360]? ", Qphi);
 double I=1.0;                              // Use I=1, but this doesn't
 double QCC = wQ2QCC(W, I);                 // Heres quad. coupling
 IntCSA Q(I,QCC,Qeta,Qtheta,Qphi);          // matter for spatial parts
```

```
//          Here Are Vegas V's According To Equations (22-27, 31)
//   Note We Change Sign On ETA As He Using A Different PAS Definition
double Theta = Q.theta()*DEG2RAD;
double Phi = Q.phi()*DEG2RAD;
double Eta = -Q.eta();
double Stheta = sin(Theta);
double Ctheta = cos(Theta);
double C2phi = cos(2.*Phi);
double S2phi = sin(2.*Phi);
double Vxx = 0.5*(3.*Stheta*Stheta - 1. - Eta*Ctheta*Ctheta*C2phi);
double Vxy = 0.5*Eta*Ctheta*S2phi;
double Vxz = -0.5*(Stheta*Ctheta*(3.0 + Eta*C2phi));
double Vyx = Vxy;
double Vyy = 0.5*(-1. + Eta*C2phi);
double Vyz = 0.5*Eta*Stheta*S2phi;
double Vzx = Vxz;
double Vzy = Vyz;
double Vzz = 0.5*(3.*Ctheta*Ctheta - 1. - Eta*Stheta*Stheta*C2phi);
complex V0(sqrt(1.5)*Vzz);
complex V1(-Vxz, -Vyz);
complex Vm1(Vxz, -Vyz);
complex V2(0.5*(Vxx-Vyy), Vxy);
complex Vm2(0.5*(Vxx-Vyy), -Vxy);
//      Here Are The A's According To GAMMA shift anisotropy Interaction
//        Need To Scale Our A's By (1/2)/sqrt[5/(24*PI)] To Get Vega's V's
double X = 0.5/RT5O24PI;
double Thetad = Q.theta();
double Phid = Q.phi();
double AGxx = X*Q.Axx(Thetad, Phid);
double AGxy = X*Q.Axy(Thetad, Phid);
double AGxz = X*Q.Axz(Thetad, Phid);
double AGyy = X*Q.Ayy(Thetad, Phid);
double AGyx = X*Q.Ayx(Thetad, Phid);
double AGyz = X*Q.Ayz(Thetad, Phid);
double AGzz = X*Q.Azz(Thetad, Phid);
double AGzx = X*Q.Azx(Thetad, Phid);
double AGzy = X*Q.Azy(Thetad, Phid);
//      Here Are The A's According To GAMMA shift anisotropy Interaction
//        Need To Scale Our A's By (1/2)/sqrt[5/(24*PI)] To Get Vega's V's
double AG1xx = X*Q.Axx();
double AG1xy = X*Q.Axy();
double AG1xz = X*Q.Axz();
double AG1yy = X*Q.Ayy();
double AG1yx = X*Q.Ayx();
double AG1yz = X*Q.Ayz();
double AG1zz = X*Q.Azz();
double AG1zx = X*Q.Azx();
double AG1zy = X*Q.Azy();
//      Here Are The A's According To GAMMA shift anisotropy Interaction
//      (Note That space_T Uses Azz>=Ayy>=Axx So ETA Opposite Vega's)
space_T Agen = A2(0.0, 1.0, Qeta);
Agen = Agen.rotate(Phid, Thetad, 0.0);
Cartesian(Agen);
//                    Output Everyone For A Visual Comparison
cout << "\n   " << "      Vega" << "          IntCSAA"
            << "     IntCSAB" << "     space_T";
cout << "\nVxx    " << form("%8.3f", Vxx) << "        " << form("%8.3f", AGxx)
     << "     " << form("%8.3f", AG1xx) << "     " << form("%8.3f", Agen.Ccomponent(0,0));
cout << "\nVxy    " << form("%8.3f", Vxy) << "        " << form("%8.3f", AGxy)
     << "     " << form("%8.3f", AG1xy)  << "     " << form("%8.3f", Agen.Ccomponent(0,1));
cout << "\nVxz    " << form("%8.3f", Vxz) << "        " << form("%8.3f", AGxz)
     << "     " << form("%8.3f", AG1xz) << "     " << form("%8.3f", Agen.Ccomponent(0,2));
cout << "\nVyy    " << form("%8.3f", Vyy) << "        " << form("%8.3f", AGyy)
     << "     " << form("%8.3f", AG1yy) << "     " << form("%8.3f", Agen.Ccomponent(1,1));
cout << "\nVyx    " << form("%8.3f", Vyx) << "        " << form("%8.3f", AGyx)
     << "     " << form("%8.3f", AG1yx) << "     " << form("%8.3f", Agen.Ccomponent(1,0));
cout << "\nVyz    " << form("%8.3f", Vyz) << "        " << form("%8.3f", AGyz)
     << "     " << form("%8.3f", AG1yz) << "     " << form("%8.3f", Agen.Ccomponent(1,2));
cout << "\nVzz    " << form("%8.3f", Vzz) << "        " << form("%8.3f", AGzz)
     << "     " << form("%8.3f", AG1zz) << "     " << form("%8.3f", Agen.Ccomponent(2,2));
cout << "\nVzx    " << form("%8.3f", Vzx) << "        " << form("%8.3f", AGzx)
     << "     " << form("%8.3f", AG1zx) << "     " << form("%8.3f", Agen.Ccomponent(2,0));
cout << "\nVzy    " << form("%8.3f", Vzy) << "        " << form("%8.3f", AGzy)
     << "     " << form("%8.3f", AG1zy) << "     " << form("%8.3f", Agen.Ccomponent(2,1));
cout << "\nV0 " << V0 << "  " << X*Q.A0(Thetad, Phid)
     << "  " << X*Q.A0() << "  " << Agen.component(2,0);
cout << "\nV1 " << V1 << "  " << X*Q.A1(Thetad, Phid)
     << "  " << X*Q.A1() << "  " << Agen.component(2,1);
cout << "\nV-1" << Vm1 << "  " << X*Q.Am1(Thetad, Phid)
     << "  " << X*Q.Am1() << "  " << Agen.component(2,-1);
cout << "\nV2 " << V2 << "  " << X*Q.A2(Thetad, Phid)
     << "  " << X*Q.A2() << "  " << Agen.component(2,2);
cout << "\nV-2" << Vm2 << "  " << X*Q.Am2(Thetad, Phid)
     << "  " << X*Q.Am2() << "  " << Agen.component(2,-2);
cout << "\n\n\n";
}
```

/IntQu_PCT0.cc

```
/* IntQu_PCT0.cc ************************************************************-*-c++-*-
**                                                                              **
**                        Example Program for the GAMMA Library                 **
**                                                                              **
**  This program calculates a powder average for a single spin which            **
**  is associated with a shift anisotropy interaction.  The high field          **
**  approximation is invoked in that the shift anisotropy Hamiltonian is        **
**  treated as a perturbation to the Zeeman Hamiltonian and taken to            **
**  second order.  Only shift anisotropy Hamiltonian terms which are            **
**  rotationally invariant about the field axis (z) are maintained.             **
**  Furthermore, only the central transtion will be considered.                 **
**                                                                              **
**  This will program is similar to IntQu_Pow2.cc but restricts the             **
**  computation to only the central transition. In turn, that means             **
**  only spins with I=m*1/2 where m is odd and larger than 1 are valid.         **
**  Analog formula will be used to construct the spectrum.                      **
**                                                                              **
**  Later version of GAMMA will have the functions "scale" and "sum"            **
**  in the library itself, so you will need to remove them from this            **
**  program in that event.                                                      **
**                                                                              **
** Author:      S.A. Smith                                                      **
** Date:        10/15/96                                                        **
** Update:      10/15/96                                                        **
** Version:     3.6                                                             **
** Copyright:   S. Smith.  You can modify this program as you see fit           **
**              for personal use, but you must leave the program intact         **
**              if you re-distribute it.                                        **
**                                                                              **
*********************************************************************************/

#include <gamma.h>                                  // Include GAMMA

void addW(row_vector& vx, double Fst, double Ffi, double F, double I)

    // Input      vx    : A row vector
    //            Fst   : Frequency of 1st point of vx (Hz)
    //            Ffi   : Frequency of last point of vx (Hz)
    //            F     : Transition frequency (Hz)
    //            I     : Transition intensity
    // Output     void  : The transition is added to the
    //                    row vector (as a Dirac delta).
    //Note               : To start one should zero vx

 {
 if(F<Fst || F>Ffi) return;                         // Insure its in range
 double Nm1 = double(vx.size()-1);                  // Freq. -> point conversion
 double m = Nm1/(Ffi-Fst);                          // Slope Freq. -> pt
 double dpt = m*(F-Fst);                            // Point index of F
 int pt = int(dpt);                                 // Main point for F
 double drem = dpt - pt;                            // Part which isn't
 if(!drem) vx.put(vx.get(pt)+I, pt);                // Add if on a point
 else if(drem > 0)                                  // If in between points
  {                                                 // then just split it up
  vx.put(vx.get(pt)+(1.0-drem)*I, pt);              // between the two
  vx.put(vx.get(pt+1)+drem*I, pt+1);
  }
 else
  {
  vx.put(vx.get(pt)+(1.0+drem)*I, pt);
  vx.put(vx.get(pt-1)-drem*I, pt-1);
  }
 return;
 }


main (int argc, char* argv[])

 {
 cout << "\n\t\tshift anisotropy Central Transition Powder Pattern";
 cout << "\n\t\t    (131Xe:3/2, 55Mn:5/2, 51V:7/2, ...)\n";
//                        First Make A shift anisotropy Interaction
 String Iso;                                        // Isotope of spin
 int qn=1;                                          // Query index
 query_parameter(argc, argv, qn++,                  // Get the isotope type
  "\n\tIsotope Type [131Xe, 55Mn, 51V, ....]? ", Iso);
 double wQ;                                         // Set Quad. frequency
 query_parameter(argc, argv, qn++,                  // Get the shift anisotropy coupling
     "\n\tshift anisotropy Frequency (kHz)? ", wQ);
 wQ *= 1.e3;                                        // Switch to Hz
 double eta;                                        // Set Quad. frequency
 query_parameter(argc, argv, qn++,                  // Get the shift anisotropy coupling
     "\n\tshift anisotropy eta Value [0, 1]? ", eta);
 double Om;
 query_parameter(argc, argv, qn++,                  // Get the field strength
          "\n\tLarmor Frequency (MHz)? ", Om);
 Om *= 1.e6;                                        // Switch to MHz
 Isotope S(Iso);                                    // Make a spin isotope
 double I = S.qn();                                 // This is isotope I value
 IntCSA Q(I,wQ2QCC(wQ, I), eta);                    // Set a Quad interaction
 if(!int(2*I)%2)
  {
  cout << "\n\n\tSorry, I Must Be m*1/2, m Odd!\n\n";
  exit(-1);
  }
//                        Set Things Up For The Powder Average
```

```
 int npts = 4096;                                   // Block size
 int Ntheta, Nphi=0;                                // Angle increment counts
 query_parameter(argc, argv, qn++,                  // Get the theta increments
   "\n\t# Theta (z down) Increments Spanning [0, 180]? ", Ntheta);
 if(eta)
   query_parameter(argc, argv, qn++,                // Get the phi increments
     "\n\t# Phi (x over) Increments Spanning [0, 360)? ", Nphi);
 matrix ABC = Q.wQcentral(Ntheta, Nphi);            // Prep. for 2nd order shifts
//                          Powder Averaging
//        Angle theta Is Down From The +z Axis, Angle phi Over From +x
// Note that since the 2nd order shift Wcentral(theta, phi) is symmetric with
// respect to both angles we need only average over parts of both angle ranges.
// For theta this means we sum the results from angles [0, 90) + half the result
// at 90.  Twice that sum would produce the total theta average over [0, 180].
// For phi we usually average [0, 360) so this is reduced to a sum over 3/4 the
// result at 0 + the results from angles (0, 90) + 1/2 the result at 90.  Four
// times that sum would produce the total phi average over [0, 360).

 double dthe, Nm1o2 = double(Ntheta-1.0)/2.0;      // For powder average
 double dphi, Nm2o4 = double(Nphi)/4.0;             // For powder average
 int theta, phi;                                    // Orientation angles
 double W, WQ = Q.wQ();                             // Base Quad. frequency
 double Ifact = I*(I+1) - 0.75;                     // Part of the prefactor
 double prefact = -WQ*WQ*Ifact/Om;                  // Majority of the prefact
 double Aaxis = (-1.0/9.0)*prefact;                 // For plot scaling
 double Ctheta, Stheta, Cthetasq, Ctheta4;          // We'll need these
 double Fst = -2.5*Aaxis;                           // Starting plot limit
 double Ffi = 1.5*Aaxis;                            // End plot limit
 row_vector data(npts, complex0);                   // Array for spectrum
 for(theta=0; theta<Ntheta; theta++)                // Loop over theta angles
  {
  dthe = double(theta);
  if(dthe <= Nm1o2)                                 // Only look upper half
   {                                                // of the sphere
   Ctheta = ABC.getRe(0,theta);                     // Scale factor cos(theta)
   Stheta = ABC.getRe(1,theta);                     // Scale factor sin(theta)
   Cthetasq = Ctheta*Ctheta;                        // cosine(theta)^2
   if(dthe == Nm1o2) Stheta *= 0.5;                 // Half scale if theta=90
   if(!eta)                                         // Without eta, no phi
    {                                               // averaging is needed
    W=(prefact/16.)*(1.-Cthetasq)*(9.*Cthetasq-1.); // Here is W adjustment
    addW(data, Fst, Ffi, W, Stheta);                // Add transition to spectrum
    }
   else
    {
    cout.flush();
    Ctheta4 = Cthetasq*Cthetasq;                    // cosine(theta)^4
    for(phi=0; phi<Nphi; phi++)                     // Loop over phi angles
     {
     dphi = double(phi);                            // Phi index as double
     if(dphi <= Nm2o4)                              // Only sum 1st quarter
      {
      if(!phi) Stheta *= 0.75;                      // 3/4 scale if phi=0
      else if(dphi == Nm2o4) Stheta *= 0.5;         // 1/2 scale if phi=90
      W = ABC.getRe(2,phi)*Ctheta4;                 // A part of W
      W += ABC.getRe(3,phi)*Cthetasq;               // B part of W
      W += ABC.getRe(4,phi);                        // C part of W
      W *= (prefact/6.);                            // Scale
      addW(data, Fst, Ffi, W, Stheta);              // Add transition to spectrum
      }
     }
    }
   }
  }
 }
 double lb = 40.0;                                  // Set a line broading factor
 cout << "\n\n\tDone With Discrete Powder Average. Processing...";
 cout.flush();
 data = IFFT(data);                                 // Put into time domain
 exponential_multiply(data,-lb);                    // Apodize the "FID"
 data = FFT(data);                                  // Put back into frequency domain
 GP_1D("spec.asc", data, 0, -2.5, 1.5);             // Output the points in ASCII
 GP_1Dplot("spec.gnu", "spec.asc");                 // Call Gnuplot and make plot now
}
```

4 Quadrupolar Interactions

4.1 Overview

The class IntQuad contains a fully functional quadrupolar interaction defined for a single spin having a spin angular moment value of I>1/2. The class embodies the interaction definition and allows for interaction manipulation. The class facilitates construction of oriented Quadrupolar Hamiltonians. Note that solid state spin systems are used to simultaneously treat multiple spins (with multiple interactions.)

4.2 Available Functions

Constructors

IntQuad	- Quadrupolar interaction constructor	page 4-202
=	- Assignment operator	page 4-202

Basic Functions

delzz	- Get or set the quadrupolar spatial tensor delzz value	page 4-204
QCC, NQCC	- Get or set the quadrupolar coupling constant	page 4-204
eta	- Get or set the quadrupolar spatial tensor asymmetry	page 4-205
wQ	- Get or set the quadrupolar frequency	page 4-206
wQ0, wQoriented	- Get 1st order quadrupolar frequency (oriented)	page 4-206
wQcentral	- Get 2nd order central trans. freq. shift (I odd 1/2 mult.)	page 4-207
wQ1	- Get 2nd order transition frequency shifts	page 4-209
xi	- Get the quadrupolar interaction constant	page 4-210

Spherical Spatial Tensor Functions (Inherited[1])

A0, A20	- Get quadrupolar m=0 spherical tensor component)	page 4-212
A1, A21	- Get quadrupolar m=1 spherical tensor component	page 4-212
Am1, A2m1	- Get quadrupolar m=-1 spherical tensor component	page 4-212
A2, A22,	- Get quadrupolar m=2 spherical tensor component	page 4-212
Am2, A2m2	- Get quadrupolar m=-2 spherical tensor component	page 4-212

Cartesian Spatial Tensor Functions (Inherited[2])

Axx, Ayy, Azz	- Get the xx, yy, or zz Cartesian tensor component	page 4-214
Axy, Ayx, Axz,	- Get the xy, yx, or xz Cartesian tensor component	page 4-214
Azx, Ayz, Azy	- Get the zx, yz, or zy Cartesian tensor component	page 4-214

Spherical Spatial Tensor Functions For Averaging

1. These functions are inherited from the base class IntRank2.
2. These functions are inherited from the base class IntRank2.

4.3 Theory Sections

4.4 Chapter Figures

4.5 Literature Comparisons

4.6 Examples

4.7 Example Programs

4.8 Constructors

4.8.1 IntQuad

Usage:

void IntQuad::IntQuad()
void IntQuad::IntQuad(const IntQuad& Q1)
void IntQuad::IntQuad(double qn, double delzz, double eta=0.0, double theta=0.0, double phi=0.0)
void IntQuad::IntQuad(double qn, ParameterAVLSet& pset, int idx=-1)

Description:

The function ***IntQuad*** is used to create a new quadrupolar interaction.

1. IntQuad() - Called without arguments the function creates a NULL quadrupolar interaction.
2. IntQuad(const IntQuad& Q1) - Called with another quadrupolar interaction, a new quadrupolar interaction is constructed which is identical to the input interaction.
3. IntQuad(double qn, double delzz, double eta=0.0, double theta=0.0, double phi=0.0) - This will construct a new quadrupolar interaction for a spin having the quantum number ***qn***. The value of ***qn*** must be an integer multiple of 0.5 and be greater than 0.5 (i.e. 1, 1.5, 2.5, 3.0....). The strength of the interaction is set by the argument ***delzz*** assumed to be in units of ***Hz***. The value of ***delzz*** is equivalent to the (nuclear) quadrupolar coupling constant and related to the quadrupolar frequency. The value of ***eta*** can be optionally input and this set the asymmetry of the interaction. It is restricted to be within the range ***[0, 1]***. Two angles, ***theta*** and ***phi***, can be optionally specified in ***Hz***. This will set the orientation of the quadrupolar interaction relative to its PAS.
4. IntQuad(double qn, ParameterAVLSet& pset, int idx=-1) - This will construct a new quadrupolar interaction for a spin having the quantum number ***qn*** from parameters found in the parameter set ***pset***. If the optional index ***idx*** has been set ***>=0*** the quadrupolar parameters scanned in ***pset*** will be assumed to have a ***(idx)*** appended to their names.

Return Value:

Void. It is used strictly to create a quadrupolar interaction.

Examples:

```
#include <gamma.h>
main()
  {
  IntQuad Q;                          // An empty quadrupolar interaction.
  IntQuad Q1(1.5, 3.e5,.2, 45., 30.); // Q. Int. for I=3/2, QCC=300kHz, η=.2, θ=45, ϕ=30
  IntQuad Q2(Q1);                     // Another Quad. Interaction, here equal to Q1
  Q = Q2;                             // Now Q is the same as Q1 and Q2
  }
```

See Also: =, read, ask_read

4.8.2 =

Usage:

void IntQuad::operator= (const IntQuad& Q1)

Description:

The operator = is assign one quadrupolar interaction to another.

Return Value:

Void.

Example:

```
#include <gamma.h>
main()
  {
  IntQuad Q;                          // An empty quadrupolar interaction.
  IntQuad Q1(1.5, 3.e5,.2, 45., 30.); // Q. Int. for I=3/2, QCC=300kHz, η=.2, θ=45, ϕ=30
  Q = Q1;                             // Now Q is the same as Q1
  }
```

See Also: constructor, read, ask_read

4.9 Basic Functions

4.9.1 delzz

Usage:

```
#include <IntQuad.h>
double IntQuad::delzz () const
double IntQuad::delz () const
double IntQuad::delzz (double dz) const
double IntQuad::delz (double dz) const
```

Description:

The function ***delzz*** is used to either obtain or set the interaction quadrupolar coupling constant. With no arguments the function returns the coupling in Hz. If an argument, ***dz***, is specified then the coupling constant for the interaction is set. It is assumed that the input value of ***dz*** is in units of ***Hz***. The function is overloaded with the name ***delz*** for convenience. Note that setting of ***delzz*** will alter the (equivalent) value of the quadrupolar coupling ***QCC*** (or ***NQCC***) as well as the quadrupolar frequency.

$$\delta_{zz}^{Q} = e^2qQ = QCC = NQCC$$

Return Value:

Either void or a floating point number, double precision.

Example(s):

```
#include <gamma.h>
main()
  {
  IntQuad Q();                    // Empty quadrupolar interaction.
  Q.delzz(100000.0);              // Set QCC to 100 KHz.
  cout << Q.delz ();              // Write coupling constant to std output.
  }
```

See Also: QCC, NQCC, wQ

4.9.2 QCC, NQCC

Usage:

```
#include <IntQuad.h>
double IntQuad::QCC () const
double IntQuad::NQCC () const
double IntQuad::QCC (double dz) const
double IntQuad::NQCC (double dz) const
```

Description:

The function ***QCC*** is used to either obtain or set the interaction quadrupolar coupling constant. With no arguments the function returns the coupling in Hz. If an argument, ***dz***, is specified then the coupling constant for the interaction is set. It is assumed that the input value of ***dz*** is in units of ***Hz***. The function is overloaded with the name ***NQCC*** for convenience. Note that setting of ***QCC*** will alter the (equivalent) value of the quadrupolar spatial tensor ***delzz*** value as well as the quadrupolar frequency. This function has identical functionality as ***delzz*** and ***delz***.

$$QCC = NQCC = e^2qQ = \delta_{zz}^Q = \frac{2I(2I-1)\omega^Q}{3}$$

Return Value:

Either void or a floating point number, double precision.

Examples:

```
#include <gamma.h>
main()
  {
  IntQuad Q();                    // Empty quadrupolar interaction.
  Q.NQCC(100000.0);               // Set QCC to 100 KHz.
  cout << Q.QCC ();               // Write coupling constant to std output.
  }
```

See Also: delz, delzz, wQ

4.9.3 eta

Usage:

#include <IntQuad.h>

double IntQuad::eta () const

double IntQuad::eta (double Qeta) const

Description:

The function ***eta*** is used to either obtain or set the quadrupolar interaction asymmetry. With no arguments the function returns the asymmetry (unitless). If an argument, ***Qeta***, is specified then the asymmetry for the interaction is set.The input value is **restricted to the range [0,1]** and is related to the quadrupolar spatial tensor Cartesian components according to

$$\eta = (A_{xx} - A_{yy})/A_{zz} \qquad |A_{zz}| \geq |A_{yy}| \geq |A_{xx}|$$

Note that setting ***eta*** will alter the 5 internal irreducible spherical spatial tensor components of the interaction.

Return Value:

Either void or a floating point number, double precision.

Examples:

```
#include <gamma.h>
main()
  {
  IntQuad Q();                    // Empty quadrupolar interaction.
  Q.eta(0.75);                    // Set eta to 0.75.
  double Qeta = Q.eta();          // Set Qeta to current eta value
  }
```

See Also: delz, delzz, wQ

4.9.4 wQ

Usage:

```
#include <IntQuad.h>
double IntQuad::wQ () const
double IntQuad::wQ (double W) const
```

Description:

The function ***wQ*** is used to either obtain or set the interaction quadrupolar frequency. With no arguments the function returns the frequency in Hz. If an argument, ***W***, is specified then the frequency for the interaction is set. It is assumed that the input value of ***W*** is in units of ***Hz***. In GAMMA the quadrupolar frequency[1] is defined to be.

$$\omega^Q = \frac{3e^2qQ}{2I(2I-1)} = \frac{3QCC}{2I(2I-1)} = \sqrt{\frac{15}{2\pi}}\xi^Q$$

Return Value:

Either void or a floating point number, double precision.

Examples:

```
#include <gamma.h>
main()
  {
  IntQuad Q();                    // Empty quadrupolar interaction.
  Q.wQ(1.4e5);                    // Set quad. frequency to 140 KHz.
  cout << Q.wQ();                 // Write frequency to std output.
  }
```

See Also: delz, delzz, QCC, NQCC, xi

4.9.5 wQ0, wQoriented

Usage:

```
double IntQuad::wQoriented() const
double IntQuad::wQ0() const
double IntQuad::wQoriented(double theta, double phi) const
double IntQuad::wQ0(double theta, double phi) const
matrix IntQuad::wQoriented(int Ntheta, int Nphi) const
matrix IntQuad::wQ0(int Ntheta, int Nphi) const
```

Description:

The function ***wQ0*** (or its equivalent ***wQoriented***) is used to obtain or generate the 1st order quadrupolar frequency for a chosen orientation in ***Hz***. If the arguments, ***theta*** and ***phi***, are specified then the frequency will be returned at that orientation from the PAS rather that the internal orientation. It is assumed that the input angle values are in

1. There are variations in the literature as to what the quadrupolar frequency is. The definition in GAMMA is set such that the quadrupolar interaction will split the observed NMR transitions by ω^Q when the Zeeman interaction is strong (i.e. high field, first-order quadrupolar interaction). This definition is analogous to that of a scalar coupling.

units of ***degrees***. In GAMMA the oriented quadrupolar frequency[1] is defined to be

$$\omega^Q(\theta, \varphi) = \omega^Q(PAS) \cdot \sqrt{\frac{4\pi}{5}} A^Q_{2,0}(\theta, \varphi) = \frac{\omega^Q(PAS)}{2}[3\cos^2\theta - 1 + \eta \sin^2\theta \cos 2\varphi]$$

Alternatively, one may obtain an array of the mathematical precursors needed to generate the 1st order frequency over evenly spaced angle increments on the unit sphere. In this case the function is called with the integers ***Ntheta*** and ***Nphi***, the number of increments down and over respectively. The matrix returned will have 4 rows whose elements are given by

$$\langle 0|mx|j\rangle = \frac{1}{2}(3\cos^2\theta_j - 1) \qquad \langle 1|mx|j\rangle = \frac{1}{2}\eta \sin^2\theta_j \qquad \langle 2|mx|j\rangle = \sin\theta_j$$

$$\langle 3|mx|j\rangle = \cos 2\varphi_j \qquad \theta_j = \frac{180 \cdot j}{\text{Ntheta} - 1} \qquad \varphi_j = \frac{360 \cdot j}{\text{Nphi}}$$

and the 1st order shifts reconstructed from

$$\omega^Q(\theta_i, \varphi_j) = \omega^Q(\langle 0|mx|i\rangle + \langle 1|mx|i\rangle\langle 3|mx|j\rangle)$$

Remember, these frequencies are the splittings between transitions to first order (high field approximation) for particular orientations. They are valid only when the Zeeman interaction is much stronger than the quadrupolar interaction. One should use the second order frequency corrections when the Larmor frequency is only somewhat stronger than the quadrupolar frequency. One should use the full treatment when the quadrupolar interaction dominates.

Return Value:

Either void or a floating point number, double precision.

Examples:

```
#include <gamma.h>
main()
  {
  IntQuad Q();                    // Empty quadrupolar interaction.
  Q.wQ(1.4e5);                    // Set quad. frequency to 140 KHz.
  cout << Q.wQ();                 // Write frequency to std output.
  }
```

See Also: delz, delzz, QCC, NQCC, xi

4.9.6 wQcentral

Usage:

double IntQuad::wQcentral(double Om) const

double IntQuad::wQcentral(double Om, double theta, double phi) const

matrix IntQuad::wQcentral(int Ntheta, int Nphi) const

1. There are variations in the literature as to what the quadrupolar frequency is. The definition in GAMMA is set such that the quadrupolar interaction will split the observed NMR transitions by ω^Q when the Zeeman interaction is strong (i.e. high field, first-order quadrupolar interaction). This definition is analogous to that of a scalar coupling.

Description:

The function ***wQcentral*** is used to obtain the interaction quadrupolar frequency. The argument ***Om*** is used to indicate the Larmor frequency in ***Hz*** of the spin associated with the interaction. With no other arguments the shift will be that of the central transition at the interaction's internal orientation. With the additional arguments ***theta*** and ***phi*** the frequency will be the central transition second order shift at that orientation from the PAS rather that the internal orientation. It is assumed that the input angle values are in units of ***degrees***.

In GAMMA the 2nd order shifts to the central transition are given by

$$\omega^{Q,(2)}_{-\frac{1}{2},\frac{1}{2}}(\eta,\theta,\varphi) = \frac{-(\omega^Q)^2}{6\Omega}\left[I(I+1)-\frac{3}{4}\right][A(\eta,\varphi)\cos^4\theta + B(\eta,\varphi)\cos^2\theta + C(\eta,\varphi)]$$

where

$$A(\eta,\varphi) = \frac{-27}{8} + \frac{9}{4}\eta\cos(2\varphi) - \frac{3}{8}\eta^2\cos^2(2\varphi)$$

$$B(\eta,\varphi) = \frac{30}{8} - \frac{1}{2}\eta^2 - 2\eta\cos(2\varphi) + \frac{3}{4}\eta^2\cos^2(2\varphi)$$

$$C(\eta,\varphi) = \frac{-3}{8} + \frac{1}{3}\eta^2 - \frac{1}{4}\eta\cos(2\varphi) - \frac{3}{8}\eta^2\cos^2(2\varphi)$$

Alternatively, one may obtain an array of the mathematical precursors needed to generate the 2nd order shifts over evenly spaced angle increments on the unit sphere. In this case the function is called with the integers ***Ntheta*** and ***Nphi***, the number of increments down and over respectively. The matrix returned will have 5 rows whose elements are given by

$$\langle 0|mx|j\rangle = \cos\theta_j \qquad \langle 1|mx|j\rangle = \sin\theta_j$$

$$\langle 2|mx|j\rangle = A(\eta,\varphi_j) \qquad \langle 3|mx|j\rangle = B(\eta,\varphi_j) \qquad \langle 4|mx|j\rangle = C(\eta,\varphi_j)$$

$$\theta_j = \frac{180\cdot j}{\text{Ntheta}-1} \qquad \varphi_j = \frac{360\cdot j}{\text{Nphi}}$$

and the shifts reconstructed from the previous equations.

Note that since second order effects are field dependent, the larger the field the smaller the returned shift(s). Also, the method of obtains such shifts in this function assumes that the quadrupolar interaction is a perturbation to the Zeeman Hamiltonian. The will not be applicable when the quadrupolar splitting is on the same scale as or larger than the Larmor frequency. Finally, if I is not half integer all values returned will be zero.

Return Value:

Either void or a floating point number, double precision.

Examples:

```
#include <gamma.h>
main()
  {
  IntQuad Q();                    // Empty quadrupolar interaction.
  Q.wQ(1.4e5);                    // Set quad. frequency to 140 KHz.
  cout << Q.wQ();                 // Write frequency to std output.
  }
```

See Also: delz, delzz, QCC, NQCC, xi

4.9.7 wQ1

Usage:

double IntQuad::wQ1(double Om, double m) const

double IntQuad::wQ1(double Om, double m, double theta, double phi) const

matrix IntQuad::wQ1(int Ntheta, int Nphi) const

Description:

The function ***wQ1*** is used to obtain the second order frequency shift of a quadrupolar transition. The argument ***Om*** is used to indicate the Larmor frequency in ***Hz*** of the spin associated with the interaction. The value of ***m*** is the spin angular momentum z quantum number and should span [I, I-1,I-2,.....-I+1]. The returned shift will be for the transition between levels ***m*** and ***m-1***. With no additional arguments the shift will be for the specified transition at the interaction's internal orientation. With the additional arguments ***theta*** and ***phi*** the frequency will be the indicated transitions second order shift at that orientation from the PAS rather that the internal orientation. It is assumed that the input angle values are in units of ***degrees***.

In GAMMA the 2nd order shifts for the ***m***,***m-1*** transition are given by

$$\omega^{Q,(2)}_{m-1,m}(\eta,\theta,\varphi) = -\frac{\xi^2}{2\Omega_o}\Big\{A_{2,1}(\eta,\theta,\varphi)A_{2,-1}(\eta,\theta,\varphi)[24m(m-1)-4I(I+1)+9] + \frac{1}{2}A_{2,2}(\eta,\theta,\varphi)A_{2,-2}(\eta,\theta,\varphi)[12m(m-1)-4I(I+1)+6]\Big\}$$

Alternatively, one may obtain an array of the mathematical precursors needed to generate the 2nd order shifts over evenly spaced angle increments on the unit sphere. In this case the function is called with the integers ***Ntheta*** and ***Nphi***, the number of increments down and over respectively. The matrix returned will have 6 rows whose elements are given by

$$\langle 0|mx|j\rangle = 3\sqrt{\frac{5}{24\pi}}\sin\theta_j\cos\theta_j \qquad \langle 1|mx|j\rangle = \frac{3}{2}\sqrt{\frac{5}{24\pi}}\sin^2\theta_j$$

$$\langle 2|mx|j\rangle = -\eta\sqrt{\frac{5}{24\pi}}(\cos 2\varphi_j - i\sin 2\varphi_j) \qquad \langle 3|mx|j\rangle = \frac{\eta}{2}\sqrt{\frac{5}{24\pi}}[\cos 2\varphi_j - i2\sin 2\varphi_j]$$

$$\langle 4|mx|j\rangle = \sin\theta_j \qquad \langle 5|mx|j\rangle = \cos\theta_j$$

$$\theta_j = \frac{180\cdot j}{\text{Ntheta}-1} \qquad \varphi_j = \frac{360\cdot j}{\text{Nphi}}$$

Reconstruction of full $A_{2,m}(\theta,\varphi)$ values is based on

$$A_{2,1}(\theta,\varphi) = A_{2,1}(\theta,\varphi)\big|_{\eta=0} + \sin\theta\cos\theta Re(A^Q_{2,1}B(\varphi)) + i\sin\theta Im(A^Q_{2,1}B(\varphi))$$

$$A_{2,2}(\theta,\varphi) = A_{2,2}(\theta,\varphi)\big|_{\eta=0} + (1+\cos^2\theta)Re(A_{2,2}B(\varphi)) + i\cos\theta Im(A_{2,2}B(\varphi))$$

Required $A_{2,m}(\theta_k,\varphi_l)$ components can be reconstructed according to the discrete equations below.

$$A_{2,1}(\theta_k, \varphi_l) = \langle 0|mx|k\rangle + \langle 4|mx|k\rangle[\langle 5|mx|k\rangle Re\langle 2|mx|l\rangle + iIm\langle 2|mx|l\rangle]$$

$$A_{2,2}(\theta_k, \varphi_l) = \langle 1|mx|k\rangle + (1 + \langle 5|mx|k\rangle^2)Re\langle 3|mx|l\rangle + i\langle 5|mx|k\rangle Im\langle 3|mx|l\rangle$$

and the frequencies subsequently generated using

$$A_{2,1}A_{2,-1} = -A_{2,1}A_{2,1}^* \qquad A_{2,2}A_{2,-2} = A_{2,2}A_{2,2}^*$$

Note that since second order effects are field dependent, the larger the field the smaller the returned shift(s). Also, the method of obtains such shifts in this function assumes that the quadrupolar interaction is a perturbation to the Zeeman Hamiltonian. The will not be applicable when the quadrupolar splitting is on the same scale as or larger than the Larmor frequency. Finally, if I is not half integer all values returned will be zero.

Return Value:

Either void or a floating point number, double precision.

Examples:

```
#include <gamma.h>
main()
  {
  IntQuad Q();                    // Empty quadrupolar interaction.
  Q.wQ(1.4e5);                    // Set quad. frequency to 140 KHz.
  cout << Q.wQ();                 // Write frequency to std output.
  }
```

See Also: delz, delzz, QCC, NQCC, xi

4.9.8 xi

Usage:

double IntQuad::xi() const

Description:

The function ***xi*** is used to either obtain the GAMMA defined quadrupolar interaction constant. The constant is used to scale the interaction such that both its spatial and spin tensors are “independent” of the interaction type.

$$\xi^Q = \sqrt{\frac{6\pi}{5}}\frac{e^2qQ}{2I(2I-1)} = \sqrt{\frac{6\pi}{5}}\frac{QCC}{2I(2I-1)} = \sqrt{\frac{2\pi}{15}}\omega^Q$$

This will be used in the formulation of quadrupolar Hamiltonians according to.

$$\boldsymbol{H}^Q(\theta, \varphi) = \xi^Q\sum_{m}^{\pm 2}(-1)^m A^Q_{2,-m}(\theta, \varphi) \bullet \boldsymbol{T}^Q_{2,m}$$

Return Value:

A floating point number, double precision.

Examples:

```
#include <gamma.h>
main()
  {
```

```
IntQuad Q(1.5, 3.e5, 0.2, 45.0, 45.0);    // Make a quadrupolar interaction.
double Xi = Q.xi();                       // Get quad. interaction constant.
}
```

4.10 Spherical Spatial Tensor Functions

4.10.1 A0, A20
4.10.2 A1, A21
4.10.3 Am1, A2m1
4.10.4 A2, A22,
4.10.5 Am2, A2m2

Usage:

```
#include <IntQuad.h>
complex IntRank2::A0() const
complex IntRank2::A20() const
complex IntRank2::A0(double theta, double phi) const
complex IntRank2::A20(double theta, double phi) const
complex IntRank2::A1() const
complex IntRank2::A21() const
complex IntRank2::A1(double theta, double phi) const
complex IntRank2::A21(double theta, double phi) const
complex IntRank2::Am1() const
complex IntRank2::A2m1() const
complex IntRank2::Am1(double theta, double phi) const
complex IntRank2::Am21(double theta, double phi) const
complex IntRank2::A2() const
complex IntRank2::A22() const
complex IntRank2::A2(double theta, double phi) const
complex IntRank2::A22(double theta, double phi) const
complex IntRank2::Am2() const
complex IntRank2::A2m2() const
complex IntRank2::Am2(double theta, double phi) const
complex IntRank2::A2m2(double theta, double phi) const
```

Description:

The functions ***AM*** and ***A2M*** are used to obtain the quadrupolar interaction spatial tensor component $A_{2,m}$. Here, the names are mapped to the spherical tensor components as follows:

$$\{A0, A20\} \rightarrow A_{2,\,0}$$

$$\{A1, A21\} \rightarrow A_{21} \qquad \{Am1, A2m1\} \rightarrow A_{2,\,-1}$$

$$\{A2, A22\} \rightarrow A_{2,\,2} \qquad \{Am2, A2m2\} \rightarrow A_{2,\,-2}$$

If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments ***theta*** and ***phi*** are given the returned tensor component is for the orientation at ***theta*** degrees down from the interactions PAS z-axis and ***phi*** degrees over from the interactions PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***degrees***.

$$A_{2,0}(\theta,\varphi) = \sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]$$

$$A_{2,1}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}\sin\theta[3\cos\theta - \eta(\cos\theta\cos 2\varphi - i\sin 2\varphi)] = -A_{2,1}{}^*(\theta,\varphi)$$

$$A_{2,2}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}\frac{1}{2}[3\sin^2\theta + \eta[\cos 2\varphi(1+\cos^2\theta) - i2\sin 2\varphi\cos\theta]] = A_{2,-2}{}^*(\theta,\varphi)$$

Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type[1]. This component can also be related to the Cartesian tensor components for any arbitrary orientation.

$$A_{2,0} = \sqrt{6}[3A_{zz} - Tr\{A\}]$$

$$A_{21} = -\frac{1}{2}[A_{xz} + A_{zx} + i(A_{yz} + A_{zy})] \qquad A_{2,-1} = \frac{1}{2}[A_{xz} + A_{zx} + i(A_{yz} - A_{zy})]$$

$$A_{2,2} = \frac{1}{2}[A_{xx} - A_{yy} + i(A_{xy} + A_{yx})] \qquad A_{2,-2} = \frac{1}{2}[A_{xx} + (-A_{yy}) - i(A_{xy} + A_{yx})]$$

Return Value:

A complex number.

Example:

```
#include <gamma.h>
main()
  {
  IntQuad Q(1.5, 3.e5, 0.2, 45.0, 45.0); // Make a quadrupolar interaction.
  complex A20 = Q.A20();                 // This is at theta=phi=45 degrees
  cout << Q.A20(15.6, 99.3);             // This is at theta=15.6 and phi=99.3 degrees.
  }
```

See Also: Axx, Axy

1. Because the GAMMA platform accommodates different interaction types, the scaling on all spatial tensors is chosen to be independent of the interaction. Rather, the spatial tensors are related directly to the familiar rank two spherical harmonics $A_{2,m}(\theta,\varphi)\big|_{\eta=0} = Y_m^2(\theta,\varphi)$. Also, the sign on the term(s) involving η will have opposite sign if the common alternative definition of the PAS orientation ($|A_{zz}| \geq |A_{xx}| \geq |A_{yy}|$) is used rather that the definition used in GAMMA ($|A_{zz}| \geq |A_{yy}| \geq |A_{xx}|$)

4.11 Cartesian Spatial Tensor Functions

4.11.1 Axx, Ayy, Azz

4.11.2 Axy, Ayx, Axz,

4.11.3 Azx, Ayz, Azy

Usage:

```
#include <IntQuad.h>
complex IntRank2::Axx() const
complex IntRank2::Axx(double theta, double phi) const
complex IntRank2::Ayy() const
complex IntRank2::Ayy(double theta, double phi) const
complex IntRank2::Azz() const
complex IntRank2::Azz(double theta, double phi) const
complex IntRank2::Axy() const
complex IntRank2::Axy(double theta, double phi) const
complex IntRank2::Ayx() const
complex IntRank2::Ayx(double theta, double phi) const
complex IntRank2::Axz() const
complex IntRank2::Axz(double theta, double phi) const
complex IntRank2::Azx() const
complex IntRank2::Azx(double theta, double phi) const
complex IntRank2::Ayz() const
complex IntRank2::Ayz(double theta, double phi) const
complex IntRank2::Azy() const
complex IntRank2::Azy(double theta, double phi) const
```

Description:

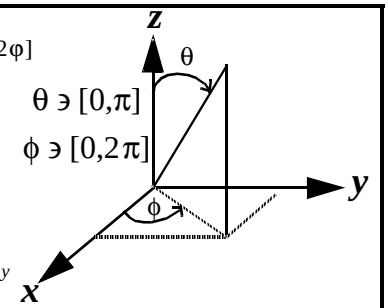
The functions ***Auv*** are used to obtain the quadrupolar interaction spatial tensor Cartesian components A_{uv}. If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments ***theta*** and ***phi*** are given the returned tensor component is for the orientation at ***theta*** degrees down from the interactions PAS z-axis and ***phi*** degrees over from the interactions PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***degrees***. The formulae for these components are given below (see Figure 19-7, page 2-53).

$$A_{xx}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}[3\sin^2\theta - 1 + \eta\cos 2\varphi\cos^2\theta] \qquad A_{yy}(\theta,\varphi) = -\sqrt{\frac{5}{24\pi}}[1 + \eta\cos 2\varphi]$$

$$A_{zz}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}[3\cos^2\theta - 1 + \eta\sin^2\theta\cos 2\varphi]$$

$$A_{xz}(\theta,\varphi) = -\sqrt{\frac{5}{24\pi}}\sin\theta\cos\theta[3 - \eta\cos 2\varphi] = A_{zx}$$

$$A_{xy}(\theta,\varphi) = -\sqrt{\frac{5}{24\pi}}\eta\sin 2\varphi\cos\theta = A_{yx} \qquad A_{yz}(\theta,\varphi) = -\sqrt{\frac{5}{24\pi}}\eta\sin\theta\sin 2\varphi = A_{zy}$$



Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type.

Return Value:

A complex number.

Example:

```
#include <gamma.h>
main()
  {
  IntQuad Q(1.5, 3.e5, 0.2, 45.0, 45.0); // Make a quadrupolar interaction.
  complex AXX = Q.Axx();                 // This is at theta=phi=45 degrees
  cout << Q.Azz0(15.6, 99.3);            // This is at theta=15.6 and phi=99.3 degrees.
  }
```

See Also: A20, A22

4.12 Powder Average Facilitator Functions

4.12.1 A0A, A20A

Usage:

row_vector IntQuad::A0A(int Ntheta)
row_vector IntQuad::A20A(int Ntheta)

Description:

The functions ***A0A*** and ***A20A*** are equivalent. They are used to obtain part of quadrupolar interaction spatial tensor component $A_{2,0}$ for a series of evenly incremented θ values.

$$A_{2,0}A(\theta) = \sqrt{\frac{5}{16\pi}}(3\cos^2\theta - 1) = A_{2,0}(\theta, \varphi)\Big|_{\eta = \varphi = 0}$$

Given a number of angle increments, ***Ntheta***, a row vector of dimension ***Ntheta*** will be returned which contains the η independent terms of $A^Q_{2,0}$ at evenly spaced increments of θ starting at the +z PAS ($\theta = 0$) alignment and finishing at -z PAS ($\theta = 180$) alignment.

$$\langle v|i\rangle = A_{2,0}A(\theta_i) \qquad \theta_i = \frac{180i}{(\text{Ntheta} - 1)}$$

Note that to obtain the full $A_{2,0}$ terms (if they are η dependent) they must be properly combined with the values from the function ***A20B.***

Return Value:

A vector.

Example:

```
#include <gamma.h>
main()
  {
  IntQuad Q(1.5, 3.e5, 0.2, 45.0, 45.0);     // Make a quadrupolar interaction.
  row_vector A20s = Q.A20A(720);             // Get 720 A20A values spanning [0, 180]
  }
```

See Also: A21A, A22A, A20B, A21B, A22B, A2As, A2Bs, A2s

4.12.2 A1A, A21A

Usage:

row_vector IntQuad::A1A(int Ntheta)
row_vector IntQuad::A21A(int Ntheta)

Description:

The functions ***A1A*** and ***A21A*** are equivalent. They are used to obtain part of quadrupolar interaction spatial tensor component $A^Q_{2,1}$ for a series of evenly incremented θ values.

$$A_{2,1}A(\theta) = 3\sqrt{\frac{5}{24\pi}}\sin\theta\cos\theta = A_{2,1}(\theta,\varphi)\big|_{\eta=0}$$

Given a number of angle increments, ***Ntheta***, a row vector of dimension ***Ntheta*** will be returned which contains the η independent terms of $A_{2,1}$ at evenly spaced increments of θ starting at the +z PAS ($\theta = 0$) alignment and finishing at -z PAS ($\theta = 180$) alignment.

$$\langle v|i\rangle = A_{2,1}A(\theta_i) \qquad \theta_i = \frac{180i}{(\mathrm{Ntheta}-1)}$$

Note that to obtain the full $A_{2,1}$ terms (if they are η dependent) they must be properly combined with the values from the function ***A21B.***

Return Value:

A vector.

Example:

```
#include <gamma.h>
main()
  {
  IntQuad Q(1.5, 3.e5, 0.2);              // Make a quadrupolar interaction.
  row_vector A21s = Q.A21A(181);          // Get 181 A20A values spanning [0, 180]
  }
```

See Also: A20A, A22A, A20B, A21B, A22B, A2As, A2Bs, A2s

4.12.3 A2A, A221A

Usage:

row_vector IntQuad::A2A(int Ntheta)
row_vector IntQuad::A22A(int Ntheta)

Description:

The functions ***A2A*** and ***A22A*** are equivalent. They are used to obtain part of quadrupolar interaction spatial tensor component $A_{2,2}$ for a series of evenly incremented θ values.

$$A_{2,2}A(\theta) = \frac{3}{2}\sqrt{\frac{5}{24\pi}}\sin^2\theta = 3\sqrt{\frac{5}{96\pi}}\sin^2\theta = A_{2,2}(\theta,\varphi)\big|_{\eta=0}$$

Given a number of angle increments, ***Ntheta***, a row vector of dimension ***Ntheta*** will be returned which contains the η independent terms of $A_{2,2}$ at evenly spaced increments of θ starting at the +z PAS ($\theta = 0$) alignment and finishing at -z PAS ($\theta = 180$) alignment.

$$\langle v|i\rangle = A_{2,2}A(\theta_i) \qquad \theta_i = \frac{180i}{(\mathrm{Ntheta}-1)}$$

Note that to obtain the full $A_{2,2}$ terms (if they are η dependent) they must be properly combined with the values from the function ***A22B.***

Return Value:

A vector.

Example:

```
#include <gamma.h>
main()
  {
  IntQuad Q(1.5, 3.e5, 0.2);              // Make a quadrupolar interaction.
  row_vector A22s = Q.A22A(181);          // Get 181 A22A values spanning [0, 180]
  }
```

See Also: A20A, A21A, A20B, A21B, A22B, A2As, A2Bs, A2s

4.12.4 A0B, A20B

Usage:

row_vector IntQuad::A0B(int Nphi)
row_vector IntQuad::A20B(int Nphi)

Description:

The functions ***A0B*** and ***A20B*** are equivalent. They are used to obtain part of quadrupolar interaction spatial tensor component $A_{2,0}$ for a series of evenly incremented φ values.

$$A_{2,0}B(\varphi) = \sqrt{\frac{5}{16\pi}}\eta\cos 2\varphi = \frac{1}{\sin^2\theta}[A_{2,0}(\theta,\varphi) - A_{2,0}(\theta,\varphi)\big|_{\eta=0}]$$

Given a number of angle increments, ***Nphi***, a row vector of dimension ***Nphi*** will be returned which contains θ independent terms of $A_{2,0}$ at evenly spaced increments of φ starting at the +x PAS ($\varphi = 0$) alignment and finishing at +x PAS ($\varphi = 360$) alignment.

$$\langle v|i\rangle = A_{2,0}A(\varphi_i) \qquad \varphi_i = \frac{360i}{\text{Nphi}}$$

Note that to obtain the full $A_{2,0}$ terms they must be properly combined with the values from the function ***A20A.***

$$A_{2,0}(\theta,\varphi) = \sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]$$

$$A_{2,1}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}\sin\theta[3\cos\theta - \eta(\cos\theta\cos 2\varphi - i\sin 2\varphi)] = -A_{2,-1}{}^*(\theta,\varphi)$$

$$A_{2,2}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}\frac{1}{2}[3\sin^2\theta + \eta[\cos 2\varphi(1 + \cos^2\theta) - i2\sin 2\varphi\cos\theta]] = A_{2,-2}{}^*(\theta,\varphi)$$

Return Value:

A vector.

Example:

```
#include <gamma.h>
main()
  {
  IntQuad Q(1.5, 3.e5, 0.2);              // Make a quadrupolar interaction.
  row_vector A20s = Q.A20B(120);          // Get 120 A20B values spanning [0, 360)
  }
```

See Also: A20A, A21A, A22A, A21B, A22B, A2As, A2Bs, A2s

4.12.5 A1B, A21B

Usage:

row_vector IntQuad::A1B(int Nphi)
row_vector IntQuad::A21B(int Nphi)

Description:

The functions ***A1B*** and ***A21B*** are equivalent. They are used to obtain part of quadrupolar interaction spatial tensor component $A_{2,1}$ for a series of evenly incremented φ values.

$$A_{2,1}B(\varphi) = -\sqrt{\frac{5}{24\pi}}\eta(\cos 2\varphi - i\sin 2\varphi)$$

where

$$A_{2,1}(\theta,\varphi) = \sin\theta\cos\theta Re(A_{2,1}B(\varphi)) + i\sin\theta Im(A_{2,1}B(\varphi)) + A_{2,1}(\theta,\varphi)\Big|_{\eta=0}$$

Given a number of angle increments, ***Nphi***, a row vector of dimension ***Nphi*** will be returned which contains θ independent terms of $A_{2,1}$ at evenly spaced increments of φ starting at the +x PAS ($\varphi = 0$) alignment and finishing at +x PAS ($\varphi = 360$) alignment.

$$\langle v|i\rangle = A_{2,1}A(\varphi_i) \qquad \varphi_i = \frac{360i}{\mathrm{Nphi}}$$

Note that to obtain the full $A_{2,1}$ terms they must be properly combined with the values from the function ***A21A.***

Return Value:

A vector.

Example:

```
#include <gamma.h>
main()
  {
  IntQuad Q(1.5, 3.e5, 0.2);              // Make a quadrupolar interaction.
  row_vector A21s = Q.A21B(120);          // Get 120 A21B values spanning [0, 360)
  }
```

See Also: A20A, A21A, A22A, A20B, A22B, A2As, A2Bs, A2s

4.12.6 A2B, A22B

Usage:

row_vector IntQuad::A2B(int Nphi)
row_vector IntQuad::A22B(int Nphi)

Description:

The functions ***A1B*** and ***A21B*** are equivalent. They are used to obtain part of quadrupolar interaction spatial tensor component $A_{2,2}$ for a series of evenly incremented φ values.

$$A_{2,2}B(\varphi) = \sqrt{\frac{5}{96\pi}}\eta[\cos 2\varphi - i2\sin 2\varphi]$$

where

$$A_{2,u}(\theta,\varphi) = (1+\cos^2\theta)Re(A_{2,2}B(\varphi)) + i\cos\theta Im(A_{2,2}B(\varphi)) + A_{2,2}(\theta,\varphi)\Big|_{\eta=0}$$

Given a number of angle increments, ***Nphi***, a row vector of dimension ***Nphi*** will be returned which contains θ independent terms of $A_{2,2}$ at evenly spaced increments of φ starting at the +x PAS ($\varphi = 0$) alignment and finishing at +x PAS ($\varphi = 360$) alignment.

$$\langle v|i\rangle = A_{2,2}A(\varphi_i) \qquad \varphi_i = \frac{360i}{\text{Nphi}}$$

Note that to obtain the full $A_{2,2}$ terms they must be properly combined with the values from the function ***A22A.***

Return Value:

A vector.

Example:

```
#include <gamma.h>
main()
  {
  IntQuad Q(1.5, 3.e5, 0.2);          // Make a quadrupolar interaction.
  row_vector A22s = Q.A22B(120);      // Get 120 A22B values spanning [0, 360)
  }
```

See Also: A20A, A21A, A22A, A20B, A21B, A2As, A2Bs, A2s

4.12.7 A2s

Usage:

matrix IntQuad::A2s(int Ntheta, int Nphi)

Description:

The function ***A2s*** is used to construct the quadrupolar interaction spatial tensor components $A^Q_{2,m}$ for a series of

evenly incremented θ and φ values. Given arguments for the number of angle increments, ***Ntheta*** and ***Nphi*** the function will return a matrix of dimension (8 x nc) where nc is the larger of the two input arguments. The matrix columns, indexed by j, will then correspond either to an angle θ or an angle φ where

$$\theta_j = \frac{180j}{(\text{Ntheta} - 1)} \qquad \varphi_j = \frac{360j}{\text{Nphi}}$$

depending upon which row is being accessed. Rows 0-2 of the array will correspond to the η independent terms of $A_{2,\{0,1,2\}}$ at evenly spaced increments of θ starting at the +z PAS ($\theta = 0$) alignment and finishing at -z PAS ($\theta = 180$) alignment. Rows 3-5 of the array will correspond to θ independent parts of the interaction spatial tensor components $A_{2,\{0,1,2\}}$ at evenly spaced increments of φ starting at the +x PAS ($\varphi = 0$) alignment and finishing at +x PAS ($\varphi = 360$) alignment. The final three array columns will contain θ dependent terms that are used to blend with the other rows to form the full $A_{2,m}(\theta, \varphi)$ values. Reconstruction of full $A_{2,m}(\theta, \varphi)$ values is based on

$$A_{2,0}(\theta, \varphi) = A_{2,0}(\theta, \varphi)\big|_{\eta = 0} + \sin^2\theta A_{2,0}B(\varphi)$$
$$A_{2,1}(\theta, \varphi) = A_{2,1}(\theta, \varphi)\big|_{\eta = 0} + \sin\theta\cos\theta Re(A_{2,1}B(\varphi)) + i\sin\theta Im(A_{2,1}B(\varphi))$$
$$A_{2,2}(\theta, \varphi) = A_{2,2}(\theta, \varphi)\big|_{\eta = 0} + (1 + \cos^2\theta)Re(A_{2,2}B(\varphi)) + i\cos\theta Im(A_{2,2}B(\varphi))$$

A particular $A_{2,m}(\theta_k, \varphi_l)$ can be reconstructed according to the analogous discrete equations.

$$A_{2,0}(\theta_k, \varphi_l) = \langle 0|mx|k\rangle + \langle 6|mx|k\rangle^2\langle 3|mx|l\rangle$$
$$A_{2,1}(\theta_k, \varphi_l) = \langle 1|mx|k\rangle + \langle 6|mx|k\rangle[\langle 7|mx|k\rangle Re\langle 4|mx|l\rangle + iIm\langle 4|mx|l\rangle]$$
$$A_{2,2}(\theta_k, \varphi_l) = \langle 2|mx|k\rangle + (1 + \langle 7|mx|k\rangle^2)Re\langle 5|mx|l\rangle + i\langle 7|mx|k\rangle Im\langle 5|mx|l\rangle$$

The components with m negative are obtained from the relationship.

$$A_{2,-m} = (-1)^m A_{2,m}$$

Return Value:

An array.

Example:

```
#include <gamma.h>
main()
  {
  IntQuad Q(1.5, 3.e5, 0.2);          // Make a quadrupolar interaction.
  matrix As = Q.A2x(720, 360);        // Get array for values spanning [0, 180] & [0, 360)
  }
```

See Also: A20A, A21A, A22A, A20B, A21B, A2As, A2Bs, A2s

4.13 Spin Tensor Functions

4.13.1 Tcomp

Usage:

```
#include <IntQuad.h>
matrix IntQuad::Tcomp(int comp)
```

Description:

The function ***Tcomp*** is used to obtain a quadrupolar interaction spin tensor component. The component desired is specified by the argument ***comp*** which relates to the m value as follows:

comp:	0	1	2	3	4
$T^Q_{2,m}$:	$T^Q_{2,0}$	$T^Q_{2,1}$	$T^Q_{2,-1}$	$T^Q_{2,2}$	$T^Q_{2,-2}$

The spin components are given by

$$T^Q_{2,0} = \frac{1}{\sqrt{6}}[3I_z^2 - \mathbf{I}^2] = \frac{1}{\sqrt{6}}[3I_z^2 - I(I+1)] \qquad T^Q_{2,\pm 1} = \mp\frac{1}{2}[I_{i\pm}\ I_{iz} + I_{iz}I_{i\pm}\ ] \qquad T^Q_{2,\pm 2} = \frac{1}{2}I_\pm^2$$

and will be returned as matrices of dimension ***2I+1*** where ***I*** is the spin quantum number associated with the interaction.

Return Value:

A matrix.

Example:

```
#include <gamma.h>
main()
  {
  IntQuad Q(1.5, 3.e5, 0.2, 45.0, 45.0);    // Make a quadrupolar interaction.
  matrix T20 = Q.Tcomp(0);                  // This is the T20 spin tensor component
  cout << T20);                             // Have a look on screen.
  }
```

4.14 Auxiliary Functions

4.14.1 setPAS

Usage:

#include <IntQuad.h>
void IntQuad::setPAS()

Description:

The functions ***setPAS*** is used to orient the quadrupolar interaction into it's principal axis system. All 5 spatial tensor components will be set to PAS values and the internal orientation angles set to zero.

Return Value:

None.

Example:

```
#include <gamma.h>
main()
  {
  IntQuad Q(1.5, 3.e5, 0.2, 45.0, 45.0);     // Make a quadrupolar interaction.
  Q.setPAS();                                // As if we used Q(1.5,3.e5,0.2,0,0)
  }
```

See Also: theta, phi, orient

4.14.2 symmetric

Usage:

#include <IntQuad.h>
int IntQuad::symmetric() const

Description:

The functions ***symmetric*** is used to check if the quadrupolar interaction has any asymmetry. The function will return true if the interaction is symmetric and false if there is some asymmetry (non-zero eta value).

Return Value:

An integer

Example:

```
#include <gamma.h>
main()
  {
  IntQuad Q(1.5, 3.e5, 0.2, 45.0, 45.0);     // Make a quadrupolar interaction.
  if(Q.symmetric()) cout << "Yep";           // We should get No for Q because eta=0.2)
  else                   << "Nope";
  }
```

See Also: eta

4.14.3 PAS

Usage:

int IntQuad::PAS) const

Description:

The function ***PAS*** is used to check if the quadrupolar interaction is oriented in its PAS or not. The function will return true if the interaction is PAS aligned and false if not).

Return Value:

An integer

Example:

```
#include <gamma.h>
main()
  {
  IntQuad Q(1.5, 3.e5, 0.2, 45.0, 45.0);     // Make a quadrupolar interaction.
  if(Q.PAS()) cout << "Yep";                 // We should get No for Q because neither θ or ϕ is 0)
  else                      << "Nope";
  }
```

See Also: eta

4.14.4 qn

Usage:

#include <IntQuad.h>

double IntQuad::qn() const

Description:

The functions ***qn*** is used to obtain the quadrupolar interaction spin quantum number. The function will return a double which will be an integer multiple of 0.5 which is not less than 1 (1.0, 1.5, 2.5, 3.0,.....).

Return Value:

A double

Example:

```
#include <gamma.h>
main()
  {
  IntQuad Q(1.5, 3.e5, 0.2, 45.0, 45.0);     // Make a quadrupolar interaction.
  double HS = 2.*Q.qn()+1.;                  // The spin Hilbert space of Q
  }
```

See Also: none

4.14.5 wQ2QCC

Usage:

#include <IntQuad.h>

friend double wQ2QCC(double wQ, double I)

Description:

The functions ***wQ2QCC*** is used to convert a quadrupolar frequency ***wQ*** for a spin with quantum number ***I*** to a quadrupolar coupling constant. The two are related in GAMMA by

$$QCC = e^2qQ = \frac{2I(2I-1)\omega^Q}{3} = 2I(2I-1)\sqrt{\frac{5}{6\pi}}\xi^Q$$

Return Value:

A double

Example:

```
#include <gamma.h>
main()
  {
  double wQ = 450.e3;                    // Quad. frequency of 450 kHz.
  double NQCC = wQ2QCC(wQ, 1.5);         // Quad. coupling if I=3/2
  }
```

See Also: QCC2wQ

4.14.6 QCC2wQ

Usage:

#include <IntQuad.h>

friend double QCC2wQ(double QCC, double I)

Description:

The functions ***QCC2wQ*** is used to convert a quadrupolar coupling constant to a quadrupolar frequency. The two are related in GAMMA by

$$\omega^Q = \frac{3e^2qQ}{2I(2I-1)} = \frac{3QCC}{2I(2I-1)} = \sqrt{\frac{15}{2\pi}}\xi^Q$$

Return Value:

A double

Example:

```
#include <gamma.h>
main()
  {
  double QCC = 450.e3;                   // Quad. coupling constant of 450 kHz.
  double wQ = QCC2wQ(wQ, 1.5);           // Quad. frequency if I=3/2
  }
```

See Also: wQ2QCC

4.15 Hamiltonian Functions

4.15.1 H0

Usage:

#include <IntQuad.h>are

matrix IntQuad::H0() const

matrix IntQuad::H0(double theta, double phi) const

Description:

The function ***H0*** is used to obtain the quadrupolar Hamiltonian as a first order perturbation to the Zeeman Hamiltonian. As such, the returned matrix is "secular" and commutes with both $\boldsymbol{F_z}$ and $\boldsymbol{R_z}$. It will be valid in a rotating frame about the z-axis. It will not be valid unless the Zeeman Hamiltonian (which it is meant to be added to[1]) is much stronger. The return array will have units of ***Hz***. The dimension of the array will be 2I+1 where I is the spin quantum value associated with the interaction. If the input arguments ***theta*** and ***phi*** are given the returned Hamiltonian is for the orientation at ***theta*** degrees down from the interaction PAS z-axis and ***phi*** degrees over from the interaction PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***degrees.***

In GAMMA the first order quadrupolar Hamiltonian is given by

$$H_Q^{(0)} = \xi^Q A_{0,0}(\eta, \theta, \phi) T_{0,0}^Q = \frac{\omega^Q}{12}[3\cos^2\theta - 1 + \eta\sin^2\theta\cos(2\varphi)][3I_z^2 - I(I+1)]$$

Return Value:

A matrix.

Example:

```
#include <gamma.h>
main()
  {
  IntQuad Q(1.5, 3.e5, 0.2, 45.0, 45.0);     // Make a quadrupolar interaction.
  matrix H = Q.H0();                         // Here's the 1st order Quad. Hamiltonian
  cout << H;                                 // Have a look at the Hamiltonian.
  }
```

See Also: H1, Hsec, H

4.15.2 H1

Usage:

#include <IntQuad.h>

matrix IntQuad::H1() const

matrix IntQuad::H1(double theta, double phi) const

---

1. A spin in a strong magnetic field will evolve under the influence of both the Zeeman Hamiltonian, $H_Z$and the Quadrupolar Hamiltonian H_Q. When the Zeeman interaction is much strong than the Quadrupolar interaction it suffices to use H0 instead of H_Q. This is often nice to use because then the two Hamiltonians commute. In evolving a density operator one may then work in the rotating frame at a spin's Larmor frequency by simply removing the Zeeman Hamiltonian and evolving under only H0.

Description:

The function ***H1*** is used to obtain the second order quadrupolar Hamiltonian as a perturbation to the Zeeman Hamiltonian. As such, the returned matrix is "secular" and commutes with both $\boldsymbol{F_z}$ and $\boldsymbol{R_z}$. It will be valid in a rotating frame about the z-axis. It will not be valid unless the Zeeman Hamiltonian (to which it is meant to be added[1]) is much stronger. The return array will have units of ***Hz***. The dimension of the array will be 2I+1 where I is the spin quantum value associated with the interaction. If the input arguments ***theta*** and ***phi*** are given the returned Hamiltonian is for the orientation at ***theta*** degrees down from the interaction PAS z-axis and ***phi*** degrees over from the interaction PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***degrees***

In GAMMA the second order quadrupolar Hamiltonian is given by

$$H_Q^{(1)} = -\frac{\xi^2}{2\Omega_o} I_z \{A_{0,1}A_{0,-1}[4I(I+1) - 8I_z^2 - 1] + A_{0,2}A_{0,-2}[2I(I+1) - 2I_z^2 - 1]\}$$

Return Value:

A matrix.

Example:

```
#include <gamma.h>
main()
  {
  IntQuad Q(1.5, 3.e5, 0.2, 45.0, 45.0);    // Make a quadrupolar interaction.
  matrix H = Q.H1();                        // Here's the 2nd order Quad. Hamiltonian
  cout << H;                                // Have a look at the Hamiltonian.
  }
```

See Also: QCC, NQCC, wQ

4.15.3 Hsec

Usage:

#include <IntQuad.h>

matrix IntQuad::Hsec() const

Description:

The function ***Hsec*** is used to obtain the sum of the first and second order quadrupolar Hamiltonians as a perturbation to the Zeeman Hamiltonian. As such, the returned matrix is "secular" and commutes with both $\boldsymbol{F_z}$ and $\boldsymbol{R_z}$. It will be valid in a rotating frame about the z-axis. It will not be valid unless the Zeeman Hamiltonian is much stronger. The return array will have units of ***Hz***. The dimension of the array will be 2I+1 where I is the spin quantum value associated with the interaction.

Return Value:

A matrix.

Example:

```
#include <gamma.h>
main()
```

1. In the rotating frame the effective Hamiltonian my have all Zeeman contributions removed. Note that the function does not include the 1st order terms, so should be added to the return from the function H0! The function Hsec will do that automatically.

```
{
IntQuad Q(1.5, 3.e5, 0.2, 45.0, 45.0);    // Make a quadrupolar interaction.
matrix H = Q.H1();                        // Here's the 2nd order Quad. Hamiltonian
cout << H;                                // Have a look at the Hamiltonian.
}
```

See Also: QCC, NQCC, wQ

4.15.4 H

Usage:

```
#include <IntQuad.h>
matrix IntQuad::H() const
```

Description:

The function ***H*** is used to obtain the quadrupolar Hamiltonian. Most likely this will NOT commute with $\boldsymbol{R_z}$. Thus it will be time independent in the laboratory frame (and time dependent in a frame rotating about the z-axis). The return array will have units of ***Hz***. The dimension of the array will be 2I+1 where I is the spin quantum value associated with the interaction.

Return Value:

A matrix.

Example:

```
#include <gamma.h>
main()
  {
  IntQuad Q(1.5, 3.e5, 0.2, 45.0, 45.0);    // Make a quadrupolar interaction.
  matrix H = Q.H1();                        // Here's the 2nd order Quad. Hamiltonian
  cout << H;                                // Have a look at the Hamiltonian.
  }
```

See Also: QCC, NQCC, wQ

4.16 I/O Functions

4.16.1 read

Usage:

void IntQuad::read(const String& filename, const spin_sys) const
void IntQuad::read(const String& filename, const spin_sys) const
void IntQuad::read(const String& filename, const spin_sys) const
void IntQuad::read(const String& filename, const spin_sys) const

Description:

The function ***delzz*** is used to either obtain or set the interaction quadrupolar coupling constant. With no arguments the function returns the coupling in Hz. If an argument, ***dz***, is specified then the coupling constant for the interaction is set. It is assumed that the input value of ***dz*** is in units of ***Hz***. The function is overloaded with the name ***delz*** for convenience. Note that setting of ***delzz*** will alter the (equivalent) value of the quadrupolar coupling ***QCC***/ ***NQCC*** as well as the quadrupolar frequency.

Return Value:

Either void or a floating point number, double precision.

Example(s):

```
#include <gamma.h>
main()
  {
  IntQuad Q();                    // Empty quadrupolar interaction.
  Q.delzz(100000.0);              // Set QCC to 100 KHz.
  cout << Q.delz ();              // Write coupling constant to std output.
  }
```

See Also: QCC, NQCC, wQ

4.16.2 ask

Usage:

#include <IntQuad.h>
double IntQuad:: () const
double IntQuad::delz () const
double IntQuad::delzz (double dz) const
double IntQuad::delz (double dz) const

Description:

The function ***delzz*** is used to either obtain or set the interaction quadrupolar coupling constant. With no arguments the function returns the coupling in Hz. If an argument, ***dz***, is specified then the coupling constant for the interaction is set. It is assumed that the input value of ***dz*** is in units of ***Hz***. The function is overloaded with the name ***delz*** for convenience. Note that setting of ***delzz*** will alter the (equivalent) value of the quadrupolar coupling ***QCC***/ ***NQCC*** as well as the quadrupolar frequency.

Return Value:

Either void or a floating point number, double precision.

Example(s):

```
#include <gamma.h>
main()
  {
  IntQuad Q();                    // Empty quadrupolar interaction.
  Q.delzz(100000.0);              // Set QCC to 100 KHz.
  cout << Q.delz ();              // Write coupling constant to std output.
  }
```

See Also: QCC, NQCC, wQ

4.16.3 askset

Usage:

```
#include <IntQuad.h>
double IntQuad:: () const
double IntQuad::delz () const
double IntQuad::delzz (double dz) const
double IntQuad::delz (double dz) const
```

Description:

The function ***delzz*** is used to either obtain or set the interaction quadrupolar coupling constant. With no arguments the function returns the coupling in Hz. If an argument, ***dz***, is specified then the coupling constant for the interaction is set. It is assumed that the input value of ***dz*** is in units of ***Hz***. The function is overloaded with the name ***delz*** for convenience. Note that setting of ***delzz*** will alter the (equivalent) value of the quadrupolar coupling ***QCC***/ ***NQCC*** as well as the quadrupolar frequency.

Return Value:

Either void or a floating point number, double precision.

Example:

```
#include <gamma.h>
main()
  {
  IntQuad Q();                    // Empty quadrupolar interaction.
  Q.delzz(100000.0);              // Set QCC to 100 KHz.
  cout << Q.delz ();              // Write coupling constant to std output.
  }
```

See Also: QCC, NQCC, wQ

4.16.4 print

Usage:

```
#include <IntQuad.h>
ostream& IntQuad::print (ostream& ostr, int fflag=-1)
```

Description:

The function ***print*** is used to write the interaction quadrupolar coupling constant to an output stream ***ostr***. An additional flag ***fflag*** is set to allow some control over how much information is output. The default (***fflag !=0***) prints all information concerning the interaction. If ***fflag*** is set to zero only the basis parameters are printed.

Return Value:

The ostream is returned.

Example:

```
#include <gamma.h>
main()
  {
  IntQuad Q(2.5, 2.e6, 0.2, 45.7, 15.0);     // Make a quadrupolar interaction.
  cout << Q;                                 // Write the interaction to standard output.
  }
```

See Also: <<

4.16.5 <<

Usage:

#include <IntQuad.h>

friend ostream& operator << (ostream& out, IntQuad& Q)

Description:

The operator << defines standard output for the interaction quadrupolar coupling constant.

Return Value:

The ostream is returned.

Example:

```
#include <gamma.h>
main()
  {
  IntQuad Q(1.5, 3.e5, 0.2);                 // Make a quadrupolar interaction.
  cout << Q;                                 // Write the interaction to standard output.
  }
```

See Also: print

4.16.6 printSpherical

Usage:

#include <IntQuad.h>

ostream& IntQuad::print (ostream& ostr, int fflag=-1)

Description:

The function ***print*** is used to write the interaction quadrupolar coupling constant to an output stream ***ostr***. An additional flag ***fflag*** is set to allow some control over how much information is output. The default (***fflag !=0***) prints all information concerning the interaction. If ***fflag*** is set to zero only the basis parameters are printed.

Return Value:

The ostream is returned.

Example:

```
#include <gamma.h>
main()
```

```
{
IntQuad Q(2.5, 2.e6, 0.2, 45.7, 15.0);    // Make a quadrupolar interaction.
cout << Q;                                // Write the interaction to standard output.
}
```

See Also: <<

4.16.7 printCartesian

Usage:

```
#include <IntQuad.h>
ostream& IntQuad::print (ostream& ostr, int fflag=-1)
```

Description:

The function ***print*** is used to write the interaction quadrupolar coupling constant to an output stream ***ostr***. An additional flag ***fflag*** is set to allow some control over how much information is output. The default (***fflag !=0***) prints all information concerning the interaction. If ***fflag*** is set to zero only the basis parameters are printed.

Return Value:

The ostream is returned.

Example:

```
#include <gamma.h>
main()
  {
  IntQuad Q(2.5, 2.e6, 0.2, 45.7, 15.0);    // Make a quadrupolar interaction.
  cout << Q;                                // Write the interaction to standard output.
  }
```

See Also: <<

4.17 Description

4.17.1 Overview

Nuclei having spin quantum numbers $I > 1/2$ may exhibit electric quadrupole moments, eQ. These quadrupole moments will interact with any electric field gradients, eq, such as those due to the electron cloud surrounding the nucleus. As a consequence, the nuclear spin energy levels are split. Since nuclear quadrupolar interactions are typically large (often measured in MHz) they can produce dramatic effects on NMR experiments in both liquids and solids.

4.17.2 Internal Structure

The internal structure of class ***IntQuad*** contains the quantities listed in the following table (names shown are also internal).

Table 3-1: Internal Structure of Class IntQuad

Name	Description	Type
Inherited From Class IntRank2		
DELZZ	Spatial Tensor δ_{zz}	double
ETA	Spatial Tensor η	double
THETA	Orientation Angle θ	double
PHI	Orientation Angle ϕ	double
Asph	Spatial Tensor Values $A_{2,m}$	complex*

Name	Description	Type
Additional Terms This Interaction		
I	Spin Quantum Number	int
Tsph	Spin Tensor Values $T_{2,m}$	matrix*

Table 3-1 **Depiction of class IntQuad contents, i.e. what each GAMMA defined quadruolar interaction contains. The values in the left column are inherited from the base class IntRank2 whereas the values on the right are specific to quadrupolar interaction. Tsph will contain 5 matrices which dimension will be 2*I+1 and Asph will contain 5 complex numbers. The value of I must be a postive multiple of 1/2 and greater than 1/2.**

The values of ***I*** is the spin quantum number of the quadrupolar nucleus[1]. It dictates how many energy levels (and transitions) are associated with the quadrupolar interaction. It is intrinsically tied into the values and dimensions of the matrices in the vector ***Tsph***. Note that ***I*** will be an integer multiple of 1/2 and that only nuclei with ***I***>1/2 will have a quadrupole moment.

The two values ***DELZZ*** and ***ETA*** are all that is required to specify the quadrupolar interaction strength and may be used to represent the quadrupolar spatial tensor. However, in GAMMA the

1. Class IntQuad has no knowledge of GAMMA's Isotope class. Values of I are NOT associated with any particular spin type. Spin system class(es) which use IntQuad will make such associations. Additionally, isotope labels can be used in parameter sets (external ASCII files) to define a quadrupolar interaction, but the association is not maintained internally in IntQuad.

value of ***DELZZ*** is factored out of the spatial tensor such that all rank two interactions (such as the quadrupolar interaction) have the same spatial tensor scaling.

The two angles ***THETA*** and ***PHI*** indicate how the quadrupolar interaction is aligned relative to the interaction principal axes (PAS). These are one in the same as the angles shown in Figure 19-14 when the Cartesian axes are those of the PAS with the origin vaguely being the center of the nucleus. These are intrinsically tied into the values in the array ***Asph***.

There are five values in the complex vector ***Asph*** and these are irreducible spherical components of the quadrupolar spatial tensor oriented at angle ***THETA*** down from the PAS z-axis and over angle ***PHI*** from the PAS x-axis. Note that these 5 values are not only orientation dependent, they are also ***ETA*** dependent. If either of the three the interaction values {***ETA***, ***THETA, PHI***} are altered these components will all be reconstructed. The values in ***Asph*** will be scaled such that they are consistent with other rank 2 spatial tensors in GAMMA which are independent of the interaction type.

The vector of matrices relates to the spherical spin tensor components according to:

Tsph:	[0]	[1]	[2]	[3]	[4]
$T^Q_{2,m}$:	$T^Q_{2,0}$	$T^Q_{2,1}$	$T^Q_{2,-1}$	$T^Q_{2,2}$	$T^Q_{2,-2}$

and the vector of complex numbers relate to the spherical spatial tensor components via

Asph:	[0]	[1]	[2]	[3]	[4]
$A_{2,m}$:	$A_{2,0}$	$A_{2,1}$	$A_{2,-1}$	$A_{2,2}$	$A_{2,-2}$

4.17.3 Classical Electrostatics

There are many different ways to treat the associated interaction, here we follow the presentations of Slichter[1] and Man[2]. We start with the classical electrostatic interaction between a charge distribution $\rho(\vec{r})$ (in this case the nucleus) and an electrostatic potential $V(\vec{r})$ (set up by the surrounding electron cloud). The associated electrostatic energy is given by

$$E_e = \int_{all\ space} \rho(\vec{r})V(\vec{r})d^3r$$

In this case $V(\vec{r})$ is the potential produced by the electron cloud at the point $\vec{r}$ in the nucleus. Expansion of the electrostatic potential as a Maclaurin's series and it evaluation at $r = 0$ produces

$$E_e = V(0)\int_{\substack{all\\space}} \rho(\vec{r})d^3r + \sum_{u=1}^{3}\frac{\partial V}{\partial u}\int_{\substack{all\\space}} u\rho(\vec{r})d^3r + \frac{1}{2!}\sum_{u=1}^{3}\sum_{v=1}^{3}\frac{\partial^2 V}{\partial u\partial v}\int_{\substack{all\\space}} uv\rho(\vec{r})d^3r + \ldots$$

where $u, v \ni \{x, y, x\}$. The first term is the overall charge distribution q and the second are components of the electric dipole moment of the charge distribution d. The third term is related to the nuclear quadrupole, the one of interest here. We can define components of the electric field gradient tensor (at the nucleus origin) as

$$V_{uv} = \frac{\partial^2 V}{\partial u\partial v}$$

These are the second derivatives of the electric potential for the nucleus. The quadrupolar contribution to the electrostatic energy becomes

$$E_e^Q = \frac{1}{2!}\sum_{u=1}^{3}\sum_{v=1}^{3} V_{uv}\int_{\substack{all\\space}} uv\rho(\vec{r})d^3r$$

The potential must satisfy the Laplace equation $\nabla^2 V = 0$, and this translates in the statement that

$$Tr\{V_{uv}\} = \sum_{u=1}^{3} V_{uu} = 0$$

We shall immediately make use of this trace relationship. First, note that we can add virtually anything to the classical energy equation, as long as we use and Kronecker delta function and the ten-

1. C.P. Slichter, "Principles of Magnetic Resonance", 2nd Revised and Expanded Edition, Springer-Verlag New York, 1978. Specifically, see chapter 9.
2. P.F. Man, "Quadrupolar Interactions", Encyclopedia of Nuclear Magnetic Resonance, Eds. D.M. Grand and R.K. Harris, John Wiley & Sons, Vol. 6, Ped-Rel, 3838- 3868.

sor components.

$$E_e^Q = \frac{1}{2}\sum_{u=1}^{3}\sum_{v=1}^{3}\left[V_{uv}\int_{\substack{all\\space}} uv\rho(\vec{r})d^3r + \delta_{uv}V_{uv}(anything)\right]$$

What we choose to add terms that will place the equation in a form more amenable to tensor nomenclature. This new energy equation is

$$E_e^Q = \frac{1}{2}\sum_{u=1}^{3}\sum_{v=1}^{3}\left[V_{uv}\int\left(uv - \frac{1}{3}\delta_{uv}r^2\right)\rho(\vec{r})d^3r\right] = \frac{1}{6}\sum_{u=1}^{3}\sum_{v=1}^{3}[V_{uv}\int(3uv - \delta_{uv}r^2)\rho(\vec{r})d^3r]$$

where the integrals are taken over all space. We define a new tensor Q where[1]

$$Q_{uv} = \int_{\substack{all\\space}} [3uv - \delta_{uv}r^2]\rho(\vec{r})d^3r$$

and the energy becomes

$$E_e^Q = \frac{1}{6}\sum_{u=1}^{3}\sum_{v=1}^{3} V_{uv}Q_{uv}$$

4.17.4 Quantum Mechanical Formulation

We can now write the quadrupolar Hamiltonian by replacing the charge distribution (in the nucleus) $\rho(\vec{r})$ found in the tensor Q as a quantum mechanical operator.

$$\rho(\vec{r}) = \sum_{\vartheta}^{nucleons} q_\vartheta\delta(\vec{r} - \vec{r}_\vartheta) = \sum_{\vartheta}^{protons} q_\vartheta\delta(\vec{r} - \vec{r}_\vartheta)$$

$$Q_{uv} = \int[3uv - \delta_{uv}r^2]\rho(\vec{r})d^3r = \int[3uv - \delta_{uv}r^2]\left[\sum_{\vartheta}^{protons} q_\vartheta\delta(\vec{r} - \vec{r}_\vartheta)\right]d^3r$$

By placing this into our energy equation we obtain an initial formulation of the Quadrupolar Hamiltonian.

1. As will soon be apparent, the form of Q is that of the irreducible tensor component $T_{2,0}$ and this has distinct advantages.

$$H^Q = \frac{1}{6}\sum_{u=1}^{3}\sum_{v=1}^{3} V_{uv}Q_{uv} \text{ where } Q_{uv} = \int[3uv - \delta_{uv}r^2]\left[\sum_{\vartheta}^{protons} q_\vartheta \delta(\vec{r} - \vec{r}_\vartheta)\right]d^3r$$

However, the quadrupolar Hamiltonian and energy equations remain in terms of quantities we do not wish to work with in NMR, namely nuclear particles. Our primary task in dealing with the quadrupolar Hamiltonian is to somehow re-write the tensor Q in terms of the overall nucleus (spin). We are only concerned with the nuclear ground states and these are characterized by their total angular momentum I, and the associated $2I+1$ angular momentum components. Furthermore we are concerned only with spatial reorientation of nuclei (spin operators remain constant). Thus, it seems evident that we should choose to formulate the elements of Q in terms of the eigenfunctions of angular momentum, that is

$$\langle Im|Q_{uv}|Im'\rangle$$

There is an easy way to relate Cartesian tensor components to angular momentum components, through the Wigner-Ekert theorem. We have

$$\langle Im|Q_{uv}|Im'\rangle = \langle Im|e\sum_{\vartheta}^{protons} 3u_\vartheta v_\vartheta - \delta_{uv}r_\vartheta^2|Im'\rangle = C\langle Im|\left[3\left(\frac{I_uI_v + I_vI_u}{2}\right) - \delta_{uv}I^2\right]|Im'\rangle$$

where C is (as yet) an unknown constant. A particular case would be

$$\langle Im|Q_{zz}|Im\rangle = C\langle Im|\left(\frac{3}{2}(I_zI_z + I_zI_z) - I^2\right)|Im\rangle = C\langle Im|(3I_z^2 - I^2)|Im\rangle$$

$$= C[3I^2 - I(I+1)]\langle Im|Im\rangle = CI(2I-1)$$

The quantity $\langle II|Q_{zz}|II\rangle$ is defined to be the quadrupole moment of the nucleus, so we have

$$eQ = CI(2I-1) \text{ or } C = \frac{eQ}{I(2I-1)}$$

and therefore

$$Q_{uv} = C\langle Im|\left[3\left(\frac{I_uI_v + I_vI_u}{2}\right) - \delta_{uv}I^2\right]|Im'\rangle = \frac{eQ}{I(2I-1)}\left[3\left(\frac{I_uI_v + I_vI_u}{2}\right) - \delta_{uv}I^2\right]$$

4.17.5 Cartesian Tensor Formulation

The Cartesian spin tensor is then given by

$$Q_{uv} = \frac{eQ}{I(2I-1)}\left(\frac{3}{2}(I_u I_v + I_v I_u) - \delta_{uv} I^2\right)$$

and the Hamiltonian given by[1]

$$H^Q = \frac{1}{6}\sum_{u=1}^{3}\sum_{v=1}^{3} V_{uv} Q_{uv} = \frac{1}{6}\sum_{u=1}^{3}\sum_{v=1}^{3} V_{uv}\left[\frac{eQ}{I(2I-1)}\left(\frac{3}{2}(I_u I_v + I_v I_u) - \delta_{uv} I^2\right)\right]$$

$$H^Q = \frac{eQ}{6I(2I-1)}\left[\sum_{u=1}^{3}\sum_{v=1}^{3}\frac{3}{2}V_{uv}(I_u I_v + I_v I_u) - \sum_{u=1}^{3} V_{uv} I^2\right]$$

If we use standard Cartesian coordinate axes and assume that we are working in the principal axis system where

$$V(PAS) = \begin{bmatrix} V_{xx} & V_{xy} & V_{xz} \\ V_{yx} & V_{yy} & V_{yz} \\ V_{zx} & V_{zy} & V_{zz} \end{bmatrix}_{PAS} = \begin{bmatrix} V_{xx}(PAS) & 0 & 0 \\ 0 & V_{yy}(PAS) & 0 \\ 0 & 0 & V_{zz}(PAS) \end{bmatrix}$$

the Quadrupolar Hamiltonian for a single spin is

$$H^Q(PAS) = \frac{eQ}{6I(2I-1)}\sum_{u=1}^{3} V_{uu}\left[\frac{3}{2}(I_u I_u + I_u I_u) - I^2\right] = \frac{eQ}{6I(2I-1)}\sum_{u=1}^{3} V_{uu}[3I_u^2 - I^2]$$

We can use the above formula for some very limited calculations. However the quadrupolar Cartesian tensor Q_{uv} is not close to our desired irreducible spherical tensor format. Hence, we will need a bit more algebraic manipulation.

1. Note that the operator is I^2 constructed from the dot product of the two spin angular momentum vectors $\vec{I} \bullet \vec{I}$. This is often written in the literature as the operator $I(I+1)$ which implies multiplication of the identity operator by the product of the spin angular momentum operator with itself + 1. The results will be identical.

4.17.6 Cartesian Tensor Formulation

We begin with our previous general formula

$$H^Q = \frac{eQ}{6I(2I-1)}\left[\sum_{u=1}^{3}\sum_{v=1}^{3}\frac{3}{2}V_{uv}(I_uI_v + I_vI_u) - I^2\sum_{u=1}^{3}V_{uu}\right]$$

The tensor V is traceless so the second (summed) term in the previous equation is zero. In addition, V is symmetric so we can write

$$H^Q = \frac{eQ}{6I(2I-1)}\left[\sum_{u=1}^{3}\sum_{v=1}^{3}\frac{3}{2}V_{uv}(I_uI_v + I_vI_u)\right] = \frac{eQ}{2I(2I-1)}\sum_{u=1}^{3}\sum_{v=1}^{3}[V_{uv}I_uI_v]$$

This is a desirable form because it can be written as

$$\boldsymbol{H}^Q = \frac{eQ}{2I(2I+1)}\sum_{u}^{axes}\sum_{v}^{axes}\langle 1|\vec{\boldsymbol{I}}|u\rangle\langle u|\hat{\boldsymbol{V}}|v\rangle\langle v|\vec{\boldsymbol{I}}|1\rangle \tag{13}$$

or

$$\boldsymbol{H}^Q = \frac{eQ}{2I(2I+1)}\vec{\boldsymbol{I}}\bullet\hat{V}\bullet\vec{\boldsymbol{I}} = \frac{eQ}{2I(2I+1)}\begin{bmatrix}\boldsymbol{I}_x & \boldsymbol{I}_y & \boldsymbol{I}_z\end{bmatrix}\bullet\begin{bmatrix}V_{xx} & V_{xy} & V_{xz}\\ V_{yx} & V_{yy} & V_{yz}\\ V_{zx} & V_{zy} & V_{zz}\end{bmatrix}\bullet\begin{bmatrix}\boldsymbol{I}_x\\ \boldsymbol{I}_y\\ \boldsymbol{I}_z\end{bmatrix}. \tag{14}$$

which can be rearranged to produce an equation involving two rank 2 Cartesian tensors by taking the dyadic product of the vector $\vec{\boldsymbol{I}}$ with itself.

$$\boldsymbol{H}^Q = \frac{eQ}{2I(2I+1)}\sum_{u}^{axes}\sum_{v}^{axes}\langle u|\hat{\boldsymbol{V}}|v\rangle\langle v|\vec{\boldsymbol{I}}|1\rangle\langle 1|\vec{\boldsymbol{I}}|u\rangle = \frac{eQ_i}{2I(2I+1)}\sum_{u}^{axes}\sum_{v}^{axes}\langle u|\hat{\boldsymbol{V}}|v\rangle\langle v|\vec{\boldsymbol{I}}\vec{\boldsymbol{I}}|u\rangle$$

The dyadic product to produce $\vec{\boldsymbol{I}}\vec{\boldsymbol{I}}$ is explicitly done *via*

$$\begin{bmatrix}\boldsymbol{I}_x\\ \boldsymbol{I}_y\\ \boldsymbol{I}_z\end{bmatrix}\bullet\begin{bmatrix}\boldsymbol{I}_x & \boldsymbol{I}_y & \boldsymbol{I}_z\end{bmatrix} = \begin{bmatrix}\boldsymbol{I}_x\boldsymbol{I}_x & \boldsymbol{I}_x\boldsymbol{I}_y & \boldsymbol{I}_x\boldsymbol{I}_z\\ \boldsymbol{I}_y\boldsymbol{I}_x & \boldsymbol{I}_y\boldsymbol{I}_y & \boldsymbol{I}_y\boldsymbol{I}_z\\ \boldsymbol{I}_z\boldsymbol{I}_x & \boldsymbol{I}_z\boldsymbol{I}_y & \boldsymbol{I}_z\boldsymbol{I}_z\end{bmatrix}.$$

Letting $\hat{T} = \vec{\boldsymbol{I}}\vec{\boldsymbol{I}}$, the quadrupolar Hamiltonian can be expressed as the scalar product of two rank 2 Cartesian tensors.

$$\boldsymbol{H}^Q = \frac{eQ_i}{2I(2I+1)} \sum_{u}^{\text{axes}} \sum_{v}^{\text{axes}} \langle u|\hat{\boldsymbol{V}}|v\rangle\langle v|\hat{\boldsymbol{T}}|u\rangle = \frac{eQ_i}{2I(2I+1)} \hat{\boldsymbol{V}} \bullet \hat{\boldsymbol{T}} \tag{15}$$

4.17.7 Spherical Tensor Formulation

The previous equation, (15), can also be rewritten in term of irreducible spherical components rather than in terms of the Cartesian components using the substitution

$$\sum_{l=0}^{2} \sum_{m}^{\pm l} (-1)^m \boldsymbol{V}^Q_{l-m} \boldsymbol{T}^Q_{lm} = \sum_{u}^{\text{axes}} \sum_{v}^{\text{axes}} \langle u|\hat{\boldsymbol{V}}|v\rangle\langle v|\hat{\boldsymbol{T}}|u\rangle \tag{16}$$

The result is

$$\boldsymbol{H}^Q = \frac{eQ_i}{2I(2I+1)} \sum_{l=0}^{2} \sum_{m}^{\pm l} (-1)^m \boldsymbol{V}^Q_{l-m} \boldsymbol{T}^Q_{lm} \tag{17}$$

At this point the quadrupolar Hamiltonian expressed in terms of a product of the nuclear electric quadrupole moment[1] $e\boldsymbol{Q}$ and the electric field gradient tensor $\boldsymbol{V}$.

1. This is a property of the nucleus itself, not of the nucleus environment, and can be found in tables.

4.17.8 Quadrupolar Spherical Tensor Spin Components

There are, in principle, 9 irreducible spherical components of the (reducible rank 2) quadrupolar spin tensor, $T^Q_{l,m}$. These may be written down directly from the Cartesian components, $\langle v|\hat{\boldsymbol{T}}^Q|u\rangle$, as indicated in GAMMA Class Documentation on Spin Tensors. The component nomenclature is

$$T^Q_{l,m},$$

where the subscript *l* spans the rank (in this case 2) as *l* = [0, 2], and the subscript *m* spans +/- *l*, *m* = [-*l*, *l*]. The nine formulas for these quantities a listed in the following figure.

Quadrupolar Irreducible Spherical Spin Tensor Components

$$T^Q_{0,0}(i) = \frac{-1}{\sqrt{3}}\vec{I}_i^2 = \frac{-1}{\sqrt{3}}[I_{ix}^2 + I_{iy}^2 + I_{iz}^2] = \frac{-1}{\sqrt{3}}\left[I_{iz}^2 + \frac{1}{2}(I_{i+}I_{i-} + I_{i-}I_{i+})\right]$$

$$T^Q_{1,0}(i) = \frac{-1}{2\sqrt{2}}(I_{i+}I_{i-} + I_{i-}I_{i+}) \qquad T^Q_{1,\pm1}(i) = \frac{-1}{2\sqrt{2}}[I_{i\pm}\ I_{iz} + I_{iz}I_{i\pm}\ ]$$

$$T^Q_{2,0}(i) = \frac{1}{\sqrt{6}}[3I_{iz}^2 - \vec{I}_i^2] = \frac{1}{\sqrt{6}}\left[2I_{iz}^2 - \frac{1}{2}(I_{i+}I_{i-} + I_{i-}I_{i+})\right]$$

$$T^Q_{2,\pm1}(i) = \mp\frac{1}{2}[I_{i\pm}\ I_{iz} + I_{iz}I_{i\pm}\ ] \qquad T^Q_{2,\pm2}(i) = \frac{1}{2}I_{i\pm}^2$$

Figure 19-14 **Irreducible spherical rank 2 spin tensor components, quadrupolar interaction.**

The matrix representation of these nine tensor components will depend upon the matrix representations of the individual spin operators from which they are constructed[1]. These in turn depend upon the spin quantum numbers of the two spins involved. For a treatment of a spin 1 particle the quadrupolar tensor components are expressed in their matrix form in the default product basis of GAMMA as follows. In this case the spin index is implicit.

Quad. I=1 Spin Tensor Components Matrix Representations[2]

$$T^Q_{0,0} = \frac{-2}{\sqrt{3}}\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix} \quad T^Q_{1,0} = -1\begin{bmatrix}1&0&0\\0&0&0\\0&0&-1\end{bmatrix} \quad T^Q_{1,1} = \frac{-1}{\sqrt{2}}\begin{bmatrix}0&-1&0\\0&0&-1\\0&0&0\end{bmatrix} \quad T^Q_{1,-1} = \frac{-1}{\sqrt{2}}\begin{bmatrix}0&0&0\\1&0&0\\0&1&0\end{bmatrix}$$

$$T^Q_{2,0} = \frac{1}{\sqrt{6}}\begin{bmatrix}1&0&0\\0&-2&0\\0&0&1\end{bmatrix} \quad T^Q_{2,1} = \frac{1}{\sqrt{2}}\begin{bmatrix}0&-1&0\\0&0&1\\0&0&0\end{bmatrix} \quad T^Q_{2,-1} = \frac{1}{\sqrt{2}}\begin{bmatrix}0&0&0\\1&0&0\\0&-1&0\end{bmatrix} \quad T^Q_{2,2} = \begin{bmatrix}0&0&1\\0&0&0\\0&0&0\end{bmatrix} \quad T^Q_{2,-2} = \begin{bmatrix}0&0&0\\0&0&0\\1&0&0\end{bmatrix}$$

Figure 19-15 **Matrix representations of the irreducible spherical rank 2 spin tensor components appropriate for a quadrupolar interaction where I=1.**

1. Note that the spin tensors are invariably constructed in the laboratory coordinate system. Here the z-axis corresponds to the direction of the spectrometer static magnetic field and the coordinate system is right-handed.
2. The GAMMA program Quad_SpinT.cc on page 84 was used to generate these matrices.The arrays are shown in a Hilbert spin space using a default basis {α, β, γ}.

4.17.9 Quadrupolar Spherical Spatial Tensor General Components

The 9 irreducible spherical components of a spatial tensor (rank 2) are formally specified with the nomenclature

$$A_{l,\,m}$$

They are related to its Cartesian components by the following general formulas[1].

$$\begin{gathered}
A_{0,\,0} = \frac{-1}{\sqrt{3}}[A_{xx} + A_{yy} + A_{zz}] = \frac{-1}{\sqrt{3}}Tr\{A\} \\
A_{1,\,0} = \frac{-i}{\sqrt{2}}[A_{xy} - A_{yx}] \qquad A_{1,\,\pm 1} = \frac{-1}{2}[A_{zx} - A_{xz} \pm i(A_{zy} - A_{yz})] \\
A_{2,\,0} = \sqrt{6}[3A_{zz} - (A_{xx} + A_{yy} + A_{zz})] = \sqrt{6}[3A_{zz} - Tr\{A\}] \\
A_{2,\,\pm 1} = \mp\frac{1}{2}[A_{xz} + A_{zx} \pm i(A_{yz} + A_{zy})] \qquad A_{2,\,\pm 2} = \frac{1}{2}[A_{xx} - A_{yy} \pm i(A_{xy} + A_{yx})]
\end{gathered} \tag{18}$$

Again the subscript l spans the rank as l = [0, 2], and the subscript m spans +/- l, m = [-l, l].

In this quadrupolar treatment, we will have spatial components $V^{Q}_{l,\,m}$ as indicated in equation (16).

A general rank two Cartesian tensor can be written as a sum over tensors of ranks 0 - 2 as follows,

$$\hat{V} = \begin{bmatrix} V_{xx} & V_{xy} & V_{xz} \\ V_{yx} & V_{yy} & V_{yz} \\ V_{zx} & V_{zy} & V_{zz} \end{bmatrix}_i = \underbrace{V_{iso}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Rank 0}} + \underbrace{\begin{bmatrix} 0 & \alpha_{xy} & \alpha_{xz} \\ -\alpha_{xy} & 0 & \alpha_{yz} \\ -\alpha_{xz} & -\alpha_{yz} & 0 \end{bmatrix}}_{\text{Rank 1}} + \underbrace{\begin{bmatrix} \delta_{xx} & \delta_{xy} & \delta_{xz} \\ \delta_{yx} & \delta_{yy} & \delta_{yz} \\ \delta_{zx} & \delta_{zy} & \delta_{zz} \end{bmatrix}}_{\text{Rank 2}}$$

where

$$V_{iso} = \frac{1}{3}Tr\{\hat{V}\} \qquad \alpha_{xy} = \frac{1}{2}(V_{xy} - V_{yx}) \qquad \delta_{xy} = \frac{1}{2}(V_{xy} + V_{yx} - 2V_{iso})$$

As the quadrupolar spatial tensor is symmetric and traceless for all spins we obtain a simple result

$$\begin{aligned} V_{iso} &= 0 \\ \alpha_{xy} &= 0 \\ \delta_{xy} &= V_{xy} \end{aligned} \qquad \hat{V} = \begin{bmatrix} V_{xx} & V_{xy} & V_{xz} \\ V_{yx} & V_{yy} & V_{yz} \\ V_{zx} & V_{zy} & V_{zz} \end{bmatrix} = \underbrace{\begin{bmatrix} \delta_{xx} & \delta_{xy} & \delta_{xz} \\ \delta_{yx} & \delta_{yy} & \delta_{yz} \\ \delta_{zx} & \delta_{zy} & \delta_{zz} \end{bmatrix}}_{\text{Rank2}}$$

1. See GAMMA Class Documentation on Spatial Tensors.

4.17.10 Unscaled Quadrupolar Spherical Spatial Tensor PAS Components

As with any spatial tensor, the quadrupolar spatial tensor can be specified in its principal axis system, the set of axes in which the irreducible rank 2 component is diagonal[1].

$$\hat{V}(PAS) = \begin{bmatrix} \delta_{xx} & 0 & 0 \\ 0 & \delta_{yy} & 0 \\ 0 & 0 & \delta_{zz} \end{bmatrix}$$

Quadrupolar tensors, and their PAS orientation, will normally be different for each quadrupolar spin to which they are associated. Symmetric rank 2 spatial tensors are commonly specified in their principal axis system by the three components: the isotropic value A_{iso}, the anisotropy ΔA, and the asymmetry η. These are given by

$$A_{iso} = \frac{1}{3}Tr\{A\}, \qquad \Delta A = A_{zz} - \frac{1}{2}(A_{xx} + A_{yy}) = \frac{3}{2}\delta_{zz} \qquad \eta = (\delta_{xx} - \delta_{yy})/\delta_{zz}$$

In addition, a set of Euler angles $\{\alpha, \beta, \gamma\}$ is used to relate the interaction at an arbitrary orientation to the spatial tensor principle axes.

The quadrupolar spatial tensor has no anti-symmetric terms and[2] ($V_{iso} = 0$), leaving pure rank 2.

$$\hat{V}(PAS) = \delta_{zz}\begin{bmatrix} -\frac{1}{2}(1-\eta) & 0 & 0 \\ 0 & -\frac{1}{2}(1+\eta) & 0 \\ 0 & 0 & 1 \end{bmatrix} = eq\begin{bmatrix} -\frac{1}{2}(1-\eta) & 0 & 0 \\ 0 & -\frac{1}{2}(1+\eta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{19}$$

The irreducible spherical elements of the quadrupolar tensor, $V^Q_{2,m}$, in the principal axis system are obtained by placement of (19) into (18).

$$V^Q_{0,0}(PAS) = 0 \qquad V^Q_{1,m}(PAS) = 0$$

$$V^Q_{2,0}(PAS) = \sqrt{3/2}\delta_{zz} = \sqrt{3/2}eq \qquad V^Q_{2,1}(PAS) = V^Q_{2,-1}(PAS) = 0$$

$$V^Q_{2,2}(PAS) = V^Q_{2,-2}(PAS) = \frac{1}{2}\delta_{zz}\eta = \frac{1}{2}eq\eta$$

Thus, we need no longer deal with the $l = 0, 1$ spatial components in our treatment.

1. The quadrupolar principal axis system is set such that $|\delta_{zz}| \geq |\delta_{yy}| \geq |\delta_{xx}|$. The orientation of the x and y axes are inconsequential if η is zero.
2. This is unlike the chemical shielding spatial tensor which does indeed have a non-zero isotropic value. This term will result in a frequency shift in NMR spectra which remains even in isotropic media (liquids). For the quadrupolar interaction, no such shift will be seen in liquid NMR spectra. (However, there may be dynamic frequency shifts associated with quadrupolar relaxation effects).

4.17.11 Scaled Quadrupolar Spherical Spatial Tensor PAS Components

Throughout the GAMMA platform we demand all irreducible spherical rank 2 spatial components be scaled so as to be independent of any particular interaction. In fact, we adjust them to be as similar to normalized spherical harmonics as possible. Thus, we here scale the quadrupolar spatial tensor so that the $2, 0$ component will have a magnitude of the $m = 0$ rank two spherical harmonic when the two spherical angles are set to zero. Our "normalization" factor "X" is obtained by

$$V_{2,0}(\theta, \varphi)\big|_{\theta = \varphi = 0} = X \bullet A_{2,0}(\theta, \varphi)\big|_{\theta = \varphi = 0} = Y_{2,0}(\theta, \varphi)\big|_{\theta = \varphi = 0} = \sqrt{5/(4\pi)}$$

We thus define the GAMMA quadrupolar spatial tensor scaled such that

$$A_{l,m} = \sqrt{5/(6\pi)}(eq)^{-1} V^Q_{l,m} = \sqrt{5/(6\pi)}\delta_{zz}^{-1} V^Q_{l,m} \tag{20}$$

and these components are given in the next figure.

GAMMA Scaled Quadrupolar Spatial Tensor PAS Components

$$\boxed{\begin{array}{ccc} A_{0,0}(PAS) = 0 & A_{1,0}(PAS) = 0 & A_{1,\pm1}(PAS) = 0 \\ A_{2,0}(PAS) = \sqrt{\dfrac{5}{4\pi}} & A_{2,\pm1}(PAS) = 0 & A_{2,\pm2}(PAS) = \sqrt{\dfrac{5}{24\pi}}\eta \end{array}} \tag{21}$$

The scaling factor $\sqrt{5/(6\pi)}(eq)^{-1}$ which was multiplied into the "V" components will be compensated for in the full quadrupolar interaction constant. The quadrupolar Hamiltonian given in equation (17) becomes

$$\boldsymbol{H}^Q = \frac{e^2qQ}{2I(2I-1)}\sqrt{\frac{6\pi}{5}}\sum_{m}^{\pm1}(-1)^m A_{2,-m} \bullet \boldsymbol{T}^Q_{2,m} \tag{22}$$

4.17.12 Scaled Quadrupolar Spherical Spatial Tensor Components

We can express the spatial tensor components $A_{2,m}$ relative to any arbitrary axis system (AAS) by a rotation from the principal axes to the new axes *via* the formula[1]

$$A_{2,m}(AAS) = \sum_{m'}^{\pm 2} D^2_{m'm}(\Omega) A_{2,m'}(PAS) \tag{23}$$

where $D_{m'm}$ are the rank 2 Wigner rotation matrix elements and Ω the set of three Euler angles which relate the principal axes of the quadrupolar spatial tensor to the arbitrary axes. For the treatment of quadrupolar relaxation in NMR, these Euler angles are time dependent an averaged. For the treatment of quadrupolar effects in solid state NMR they may be static (powder) or time dependent (MAS). For the latter, often only two angles suffice to orient the interaction relative to its PAS.

We can expand the components of the oriented spatial tensor in terms of the Wigner rotation elements as well as the reduced Wigner rotation elements $d^2_{mm'}$ which are given by

$$\boldsymbol{D}^2_{m,n}(\alpha, \beta, \gamma) = e^{-i\alpha m} d^2_{m,n}(\beta) e^{-i\gamma n} \tag{24}$$

The reduced (rank 2) Wigner rotation matrix elements supplied by this function are given in the following figure[2].

Reduced Rank 2 Wigner Rotation Matrix Elements

$d^2_{m,n}(\beta)$

m \ n	2	1	0	-1	-2
2	$\cos^4(\beta/2)$	$-\frac{1}{2}(1+\cos\beta)\sin\beta$	$\sqrt{3/8}\sin^2\beta$	$\frac{1}{2}(\cos\beta-1)\sin\beta$	$\sin^4(\beta/2)$
1	$\frac{1}{2}\sin\beta(\cos\beta+1)$	$\cos^2\beta-\frac{1}{2}(1-\cos\beta)$	$-\sqrt{3/2}\sin\beta\cos\beta$	$\frac{1}{2}(1+\cos\beta)-\cos^2\beta$	$\frac{1}{2}\sin\beta(\cos\beta-1)$
0	$\sqrt{3/8}\sin^2\beta$	$\sqrt{3/8}\sin(2\beta)$	$\frac{1}{2}(3\cos^2\beta-1)$	$-\sqrt{3/8}\sin(2\beta)$	$\sqrt{3/8}\sin^2\beta$
-1	$-\frac{1}{2}(\cos\beta-1)\sin\beta$	$\frac{1}{2}(1+\cos\beta)-\cos^2\beta$	$\sqrt{3/2}\sin\beta\cos\beta$	$\cos^2\beta-\frac{1}{2}(1-\cos\beta)$	$-\frac{1}{2}(1+\cos\beta)\sin\beta$
-2	$\sin^4(\beta/2)$	$-\frac{1}{2}(\cos\beta-1)\sin\beta$	$\sqrt{3/8}\sin^2\beta$	$\frac{1}{2}(1+\cos\beta)\sin\beta$	$\cos^4(\beta/2)$

Figure 19-16 **The reduced Wigner rotation matrix elements of rank 2. The angle β is the standard Euler angle which is equal to the spherical phi angle.**

1. This rotation takes the PAS into the oriented coordinate axes.
2. See Brink and Satchler, page 24, TABLE 1.

Other useful relationships concerning these elements are

$$d^2_{m,\,n}(\beta) \;=\; (-1)^{m-n} d^2_{n,\,m}(\beta) \;=\; (-1)^{m-n} d^2_{-n,\,-m}(\beta) \tag{25}$$

Putting these into the formula for rotating the spatial tensor

$$\begin{aligned}
A_{2,\,m}(\theta,\varphi) \;&=\; \sum_{m'}^{\pm 2} D^2_{m'm}(0,\theta,\varphi) A^Q_{2,\,m'}(PAS) \\
A_{2,\,m}(\theta,\varphi) \;&=\; A^Q_{2,\,0}(PAS) D^2_{0m}(0,\theta,\varphi) + A^Q_{2,\,2}(PAS)[D^2_{2m}(0,\theta,\varphi) + D^2_{-2m}(0,\theta,\varphi)] \\
A_{2,\,m}(\theta,\varphi) \;&=\; \sqrt{\frac{5}{4\pi}} D^2_{0m}(0,\theta,\varphi) + \sqrt{\frac{5}{24\pi}}\eta[D^2_{2m}(0,\theta,\varphi) + D^2_{-2m}(0,\theta,\varphi)] \\
A_{2,\,m}(\theta,\varphi) \;&=\; \sqrt{\frac{5}{4\pi}} d^2_{0m}(\theta) + \sqrt{\frac{5}{24\pi}}\eta[e^{-i2\varphi} d^2_{2m}(\theta) + e^{i2\varphi} d^2_{-2m}(\theta)]
\end{aligned} \tag{26}$$

Of the five components, two may be generated by symmetry. The remaining three are listed in the next equation.

$$\begin{aligned}
A_{2,\,0}(\theta,\varphi) \;&=\; \sqrt{\frac{5}{4\pi}} d^2_{00}(\theta) + \sqrt{\frac{5}{24\pi}}\eta[e^{-i2\varphi} d^2_{20}(\theta) + e^{i2\varphi} d^2_{-20}(\theta)] \\
A_{2,\,1}(\theta,\varphi) \;&=\; \sqrt{\frac{5}{4\pi}} d^2_{01}(\theta) + \sqrt{\frac{5}{24\pi}}\eta[e^{-i2\varphi} d^2_{21}(\theta) + e^{i2\varphi} d^2_{-21}(\theta)] \;=\; -A_{2,\,-1}{}^*(\theta,\varphi) \\
A_{2,\,2}(\theta,\varphi) \;&=\; \sqrt{\frac{5}{4\pi}} d_{02}(\theta) + \sqrt{\frac{5}{24\pi}}\eta[e^{-i2\varphi} d_{22}(\theta) + e^{i2\varphi} d^2_{-22}(\theta)] \;=\; A_{2,\,-2}{}^*(\theta,\varphi)
\end{aligned} \tag{27}$$

For the m=0 component we can use the relationship $d^2_{-20}(\theta) \;=\; d^2_{20}(\theta)$

$$\begin{aligned}
A_{2,\,0}(\theta,\varphi) \;&=\; \sqrt{\frac{5}{4\pi}} d^2_{00}(\theta) + \sqrt{\frac{5}{24\pi}}\eta[e^{-i2\varphi} d^2_{20}(\theta) + e^{i2\varphi} d^2_{-20}(\theta)] \\
&=\; \sqrt{\frac{5}{4\pi}} d^2_{00}(\theta) + \sqrt{\frac{5}{24\pi}}\eta d^2_{20}(\theta)[e^{-i2\varphi} + e^{i2\varphi}] \\
&=\; \sqrt{\frac{5}{4\pi}} d^2_{00}(\theta) + \sqrt{\frac{5}{24\pi}}\eta d^2_{20}(\theta) 2\cos 2\varphi \\
&=\; \sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \sqrt{\frac{2}{3}}\eta[\sqrt{3/8}\sin^2\theta]\cos 2\varphi\right] \\
&=\; \sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]
\end{aligned} \tag{28}$$

For the m=1 component

$$
\begin{aligned}
A_{2,1}(\theta,\varphi) &= \sqrt{\frac{5}{4\pi}}d^2_{01}(\theta) + \sqrt{\frac{5}{24\pi}}\eta[e^{-i2\varphi}d^2_{21}(\theta) + e^{i2\varphi}d^2_{-21}(\theta)] \\
&= \sqrt{\frac{5}{4\pi}}[\sqrt{3/2}\sin\theta\cos\theta] + \sqrt{\frac{5}{24\pi}}\eta\left[e^{-i2\varphi}\left[-\frac{1}{2}(1+\cos\theta)\sin\theta\right] + e^{i2\varphi}\left[\frac{1}{2}(1-\cos\theta)\sin\theta\right]\right] \\
&= \sqrt{\frac{5}{4\pi}}\left[\sqrt{3/2}\sin\theta\cos\theta + \sqrt{\frac{1}{24}}\eta\sin\theta[e^{-i2\varphi}(-1-\cos\theta) + e^{i2\varphi}(1-\cos\theta)]\right] \\
&= \sqrt{\frac{5}{4\pi}}\left[\sqrt{3/2}\sin\theta\cos\theta + \sqrt{\frac{1}{24}}\eta\sin\theta(2i\sin 2\varphi - 2\cos\theta\cos 2\varphi)\right] \\
&= \sqrt{\frac{5}{24\pi}}\sin\theta[3\cos\theta - \eta(\cos\theta\cos 2\varphi - i\sin 2\varphi)]
\end{aligned}
\tag{29}
$$

For the m=2 component

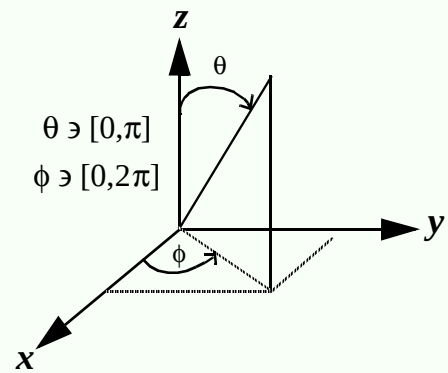
$$
\begin{aligned}
A_{2,2}(\theta,\varphi) &= \sqrt{\frac{5}{4\pi}}d_{02}(\theta) + \sqrt{\frac{5}{24\pi}}\eta[e^{-i2\varphi}d_{22}(\theta) + e^{i2\varphi}d^2_{-22}(\theta)] \\
&= \sqrt{\frac{5}{4\pi}}[\sqrt{3/8}\sin^2\theta] + \sqrt{\frac{5}{24\pi}}\eta\left[e^{-i2\varphi}\left[\frac{1}{4}(1+\cos\theta)^2\right] + e^{i2\varphi}\left[\frac{1}{4}(1-\cos\theta)^2\right]\right] \\
&= \sqrt{\frac{5}{4\pi}}\left[\sqrt{3/8}\sin^2\theta + \sqrt{\frac{1}{6}}\frac{\eta}{4}[e^{-i2\varphi}(1+\cos\theta)^2 + e^{i2\varphi}(1-\cos\theta)^2]\right] \\
&= \sqrt{\frac{5}{4\pi}}\left[\sqrt{3/8}\sin^2\theta + \sqrt{\frac{1}{6}}\frac{\eta}{4}[e^{-i2\varphi}(1+2\cos\theta+\cos^2\theta) + e^{i2\varphi}(1-2\cos\theta+\cos^2\theta)]\right] \\
&= \sqrt{\frac{5}{4\pi}}\left[\sqrt{3/8}\sin^2\theta + \sqrt{\frac{1}{6}}\frac{\eta}{4}[2\cos 2\varphi(1+\cos^2\theta) - i4\sin 2\varphi\cos\theta]\right] \\
&= \sqrt{\frac{5}{24\pi}}\frac{1}{2}[3\sin^2\theta + \eta[\cos 2\varphi(1+\cos^2\theta) - i2\sin 2\varphi\cos\theta]]
\end{aligned}
\tag{30}
$$

GAMMA Scaled Oriented Quad. Spatial Tensor Cmpnts.

$$A_{0,0}(PAS) = 0 \qquad A_{1,0}(PAS) = 0$$

$$A_{1,\pm1}(PAS) = 0 \qquad A_{2,\pm1}(PAS) = 0$$

$$A_{2,0}(PAS) = \sqrt{\frac{5}{4\pi}} \qquad A_{2,\pm2}(PAS) = \sqrt{\frac{5}{24\pi}}\eta$$



$$A_{0,0}(\theta,\varphi) = 0 = A_{\pm1,0}(\theta,\varphi)$$

$$A_{2,0}(\theta,\varphi) = \sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]$$

$$A_{2,1}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}\sin\theta[3\cos\theta - \eta(\cos\theta\cos 2\varphi - i\sin 2\varphi)] = -A_{2,-1}{}^*(\theta,\varphi)$$

$$A_{2,2}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}\frac{1}{2}[3\sin^2\theta + \eta[\cos 2\varphi(1+\cos^2\theta) - i2\sin 2\varphi\cos\theta]] = A_{2,-2}{}^*(\theta,\varphi)$$

Figure 19-17 **Equations relevant to the Quadrupolar Hamiltonian spatial spherical tensor components when oriented at angles θ & ϕ from the PAS[1].**

4.17.13 Scaled Quadrupolar Cartesian Spatial Tensor Components

For the sake of completeness we shall rewrite the Cartesian tensor components for the oriented quadrupolar spatial tensor. These are used in the literature and may form components of equations found therein. We begin with the generalized relations between the Cartesian and irreducible spherical rank 2 tensor components[2].

$$A_{xx} = \frac{1}{2}(A_{2,2} + A_{2,-2}) - \frac{1}{\sqrt{6}}A_{2,0} \qquad A_{xy} = A_{yx} \qquad A_{xz} = A_{zx}$$

$$A_{yx} = -\frac{i}{2}(A_{2,2} - A_{2,-2}) \qquad A_{yy} = \frac{-1}{2}(A_{2,2} + A_{2,-2}) - \frac{1}{\sqrt{6}}A_{2,0} \qquad A_{yz} = A_{zy} \tag{31}$$

$$A_{zx} = -\frac{1}{2}[(A_{2,1} - A_{2,-1})] \qquad A_{zy} = \frac{i}{2}[(A_{2,1} + A_{2,-1})] \qquad A_{zz} = \sqrt{\frac{2}{3}}A_{2,0}$$

Generation of the Cartesian components thus involves the substitution of the previous formulae for the oriented spherical components into the above equations.

1. The scaling on both $\{A_{2m}\}$ and $T_{2m}\}$ are arbitrary, GAMMA uses a scaling which independent of the interaction type. The $\{A_2m\}$ are scaled so that rotations by angles θ & ϕ produce spherical harmonics for a symmetric interaction (η = 0) What is NOT arbitrary is the scaling within either of the two sets of components. The combined scaling of the two sets is also crucial. For that, GAMMA uses an interaction constant.
2. See the GAMMA documentation on spatial tensors, class space_T.

$$A_{xx} = \frac{1}{2}(A_{2,2} + A_{2,-2}) - \frac{1}{\sqrt{6}}A_{2,0} = \frac{1}{2}(A_{2,2} + A_{2,2}^*) - \frac{1}{\sqrt{6}}A_{2,0} = Re(A_{2,2}) - \frac{1}{\sqrt{6}}A_{2,0}$$

$$= \sqrt{\frac{5}{24\pi}}\frac{1}{2}[3\sin^2\theta + \eta\cos 2\varphi(1 + \cos^2\theta)] - \frac{1}{\sqrt{6}}\sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]$$

$$= \sqrt{\frac{5}{24\pi}}\frac{1}{2}[3\sin^2\theta + \eta\cos 2\varphi(1 + \cos^2\theta) - (3\cos^2\theta - 1) - \eta\sin^2\theta\cos 2\varphi]$$

$$= \sqrt{\frac{5}{24\pi}}\frac{1}{2}[3\sin^2\theta - 3\cos^2\theta + 1 + \eta\cos 2\varphi(1 + \cos^2\theta - \sin^2\theta)]$$

$$= \sqrt{\frac{5}{24\pi}}\frac{1}{2}[6\sin^2\theta - 2 + 2\eta\cos 2\varphi\cos^2\theta] = \sqrt{\frac{5}{24\pi}}[3\sin^2\theta - 1 + \eta\cos 2\varphi\cos^2\theta]$$

$$A_{yy} = \frac{-1}{2}(A_{2,2} + A_{2,-2}) - \frac{1}{\sqrt{6}}A_{2,0} = \frac{-1}{2}(A_{2,2} + A_{2,2}^*) - \frac{1}{\sqrt{6}}A_{2,0} = Re(A_{2,2}) - \frac{1}{\sqrt{6}}A_{2,0}$$

$$= -\sqrt{\frac{5}{24\pi}}\frac{1}{2}[3\sin^2\theta + \eta\cos 2\varphi(1 + \cos^2\theta)] - \frac{1}{\sqrt{6}}\sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]$$

$$= -\sqrt{\frac{5}{24\pi}}\frac{1}{2}[3\sin^2\theta + \eta\cos 2\varphi(1 + \cos^2\theta) + (3\cos^2\theta - 1) + \eta\sin^2\theta\cos 2\varphi]$$

$$= -\sqrt{\frac{5}{24\pi}}\frac{1}{2}[3\sin^2\theta + 3\cos^2\theta - 1 + \eta\cos 2\varphi(1 + \cos^2\theta + \sin^2\theta)] = -\sqrt{\frac{5}{24\pi}}\frac{1}{2}[2 + 2\eta\cos 2\varphi] = -\sqrt{\frac{5}{24\pi}}[1 + \eta\cos 2\varphi]$$

$$A_{zz} = \sqrt{\frac{2}{3}}A_{2,0} = \sqrt{\frac{2}{3}}\sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right] = \sqrt{\frac{5}{24\pi}}[3\cos^2\theta - 1 + \eta\sin^2\theta\cos 2\varphi]$$

$$A_{xy} = -\frac{i}{2}(A_{2,2} - A_{2,-2}) = -\frac{i}{2}(A_{2,2} - A_{2,2}^*) = Im(A_{2,2}) = -\sqrt{\frac{5}{24\pi}}\eta\sin 2\varphi\cos\theta = A_{yx}$$

$$A_{xz} = -\frac{1}{2}[(A_{2,1} - A_{2,-1})] = -\frac{1}{2}(A_{2,1} + A_{2,1}^*) = -Re(A_{2,1}) = -\sqrt{\frac{5}{24\pi}}\sin\theta\cos\theta[3 - \eta\cos 2\varphi] = A_{zx}$$

$$A_{yz} = \frac{i}{2}(A_{2,1} + A_{2,-1}) = \frac{i}{2}(A_{2,1} - A_{2,1}^*) = -Im(A_{2,1}) = -\sqrt{\frac{5}{24\pi}}\sin\theta[-\eta(-\sin 2\varphi)] = -\sqrt{\frac{5}{24\pi}}\eta\sin\theta\sin 2\varphi = A_{zy}$$

GAMMA Scaled Oriented Quad. Cartesian Spatial Tensor Cmpnts.

$$A_{xx}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}[3\sin^2\theta - 1 + \eta\cos 2\varphi\cos^2\theta]$$

$$A_{yy}(\theta,\varphi) = -\sqrt{\frac{5}{24\pi}}[1 + \eta\cos 2\varphi]$$

$$A_{zz}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}[3\cos^2\theta - 1 + \eta\sin^2\theta\cos 2\varphi]$$

$$A_{xz}(\theta,\varphi) = -\sqrt{\frac{5}{24\pi}}\sin\theta\cos\theta[3 - \eta\cos 2\varphi] = A_{zx}$$

$$A_{xy}(\theta,\varphi) = -\sqrt{\frac{5}{24\pi}}\eta\sin 2\varphi\cos\theta = A_{yx} \qquad A_{yz}(\theta,\varphi) = -\sqrt{\frac{5}{24\pi}}\eta\sin\theta\sin 2\varphi = A_{zy}$$

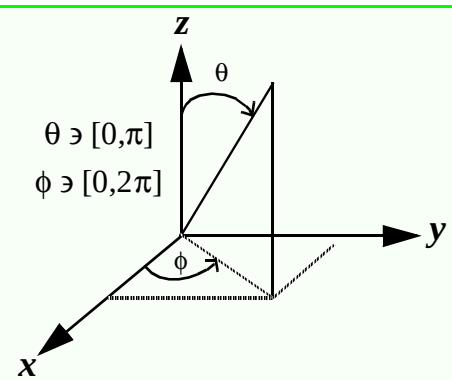


Figure 19-18 **Equations relevant to the Quadrupolar Hamiltonian spatial Cartesian tensor components when oriented at angles θ & ϕ from the PAS. GAMMA's spatial tensor scaling in interaction independent. Rotations by θ & ϕ on the spherical components (not the Cartesian ones) produce spherical harmonics if η = 0.**

4.17.14 Quadrupolar Interaction Constant

In GAMMA, since we have defined our spatial and spin tensors to be scaled independent of the type of interaction, we use an interaction constant as a scaling when formulating Hamiltonians. Hamiltonians may be produced from

$$\boldsymbol{H}^Q = \xi^Q \sum_m^{\pm 1} (-1)^m A_{2,-m} \boldsymbol{T}^Q_{2,m} \tag{32}$$

as we have already seen

$$\boldsymbol{H}^Q = \frac{e^2qQ}{2I(2I-1)} \sqrt{\frac{6\pi}{5}} \sum_m^{\pm 1} (-1)^m A_{2,-m} \boldsymbol{T}^Q_{2,m} \tag{33}$$

so evidently

$$\xi^Q = \frac{e^2qQ}{2I(2I-1)} \sqrt{\frac{6\pi}{5}} \tag{34}$$

Such interaction are not very common in the literature (except with regards to some papers treating quadrupolar relaxation in liquids) and thus not intuitive to many GAMMA users. So, one simply needs to be aware of the relationships between the interaction constant and commonly used quadrupolar definitions. Two common quantities are the quadrupolar coupling[1] constant QCC and the quadrupolar frequency ω^Q.

$$QCC = e^2qQ = \frac{2I(2I_i-1)\omega^Q}{3} \qquad \omega^Q = \frac{3e^2qQ}{2I(2I-1)} = \frac{3QCC}{2I(2I-1)} = \sqrt{\frac{15}{2\pi}}\omega^Q$$

The former is often labeled as $NQCC$, an anacronym for Nuclear Quadrupolar Coupling Constant. There are many definitions in the literature for the latter. In GAMMA we chose a definition so that this frequency will be the distance ***between transitions*** when the quadrupolar Hamiltonian is a small perturbation to the Zeeman Hamiltonian (i.e. when a spin's Larmor frequency is much higher than its quadrupolar coupling constant) and oriented with its principal z-axis aligned with the spectrometer magnetic field. Note that this means that spins with differing I values but the same quadrupolar coupling will exhibit different quadrupolar splittings.

As for the quadrupolar interaction constant we have

$$\xi^Q = \sqrt{\frac{6\pi}{5}} \frac{e^2qQ}{2I(2I-1)} = \sqrt{\frac{6\pi}{5}} \frac{QCC}{2I(2I-1)} = \sqrt{\frac{2\pi}{15}}\omega^Q \tag{35}$$

1. In angular frequency units this is $QCC = e^2qQ/h$ where Q is the quadrupole moment. Note that, although the definition of QCC is standardized, there seems to be some variation in the literature as to what the quadrupolar splitting frequency ω^Q is.

4.17.15 Quadrupolar Hamiltonian

As is eviden from the previous mathematics, regardless of the coordinate system *only the rank two components will contribute to the quadrupolar Hamiltonian*. Since only the rank 2 components exist we have removed the summation over l and may do the same for the quadrupolar Hamiltonian[1].

$$\boldsymbol{H}^Q = \frac{e^2qQ}{2I(2I-1)}\sqrt{\frac{6\pi}{5}}\sum_m^{\pm 2}(-1)^m A_{2,-m} \bullet \boldsymbol{T}^Q_{2,m} = \xi^Q\sum_m^{\pm 2}(-1)^m A_{2,-m} \bullet \boldsymbol{T}^Q_{2,m}$$

We have simplified (and standardized) our nomenclature by defining a quadrupolar interaction constant as

$$\xi^Q = \sqrt{\frac{6\pi}{5}}\frac{e^2qQ}{2I(2I-1)} = \sqrt{\frac{6\pi}{5}}\frac{QCC}{2I(2I-1)} = \sqrt{\frac{2\pi}{15}}\omega^Q \tag{36}$$

In the principal axis system, the quadrupolar Hamiltonian expands into a relatively simple formula.

$$\begin{aligned}
\boldsymbol{H}^Q(PAS) &= \xi^Q\sum_m^{\pm 2}(-1)^m A_{2,-m}(PAS) \bullet \boldsymbol{T}^Q_{2,m} \\
&= \xi^Q[A^Q_{2,0}(PAS)\boldsymbol{T}_{2,0}(i) + A_{2,2}(PAS)\boldsymbol{T}^Q_{2,-2}(i) + A_{2,-2}(PAS)\boldsymbol{T}^Q_{2,2}(i)] \\
&= \xi^Q\left[\sqrt{\frac{5}{4\pi}}\left(\frac{1}{\sqrt{6}}[3I_z^2 - \hat{\boldsymbol{I}}^2]\right) + \sqrt{\frac{5}{24\pi}}\frac{\eta}{2}(I_-^2 + I_+^2)\right] \\
&= \frac{e^2(qQ)}{4I(2I-1)}\left[3I_z^2 - \hat{\boldsymbol{I}}^2 + \frac{\eta}{2}(I_+^2 + I_-^2)\right]
\end{aligned} \tag{37}$$

There are several variations in the way the above equation is presented. Important and/or common substitutions are noted below.

$$\hat{\boldsymbol{I}}^2 = I(I+1) \qquad \frac{1}{2}(I_+^2 + I_-^2) = I_x^2 + I_y^2 \tag{38}$$

When the quadrupolar interaction is oriented relative to its principal axes[2] the Hamiltonian equation becomes much more complicated than the one above.

$$\begin{aligned}
\boldsymbol{H}^Q(\theta,\varphi) &= \xi^Q\sum_m^{\pm 2}(-1)^m A_{2,-m}(\theta,\varphi) \bullet \boldsymbol{T}^Q_{2,m} \\
&= \xi^Q[A_{2,0}(\theta,\varphi)\boldsymbol{T}^Q_{2,0} + A_{2,1}(\theta,\varphi)\boldsymbol{T}^Q_{2,-1} + A_{2,-1}(\theta,\varphi)\boldsymbol{T}^Q_{2,1} + A_{2,2}(\theta,\varphi)\boldsymbol{T}^Q_{2,-2} + A_{2,-2}(\theta,\varphi)\boldsymbol{T}^Q_{2,2}] \\
&= \xi^Q[A_{2,0}(\theta,\varphi)\boldsymbol{T}^Q_{2,0} + A_{2,1}(\theta,\varphi)\boldsymbol{T}^Q_{2,-1} - A_{2,1}{}^*(\theta,\varphi)\boldsymbol{T}^Q_{2,1} + A_{2,2}(\theta,\varphi)\boldsymbol{T}_{2,-2} + A_{2,2}{}^*(\theta,\varphi)\boldsymbol{T}^Q_{2,2}] \\
&= \xi^Q\{A_{2,0}(\theta,\varphi)\boldsymbol{T}^Q_{2,0} + Re[A_{2,1}(\theta,\varphi)](\boldsymbol{T}^Q_{2,-1} - \boldsymbol{T}^Q_{2,1}) + iIm[A_{2,1}(\theta,\varphi)](\boldsymbol{T}^Q_{2,-1} + \boldsymbol{T}^Q_{2,1}) \\
&\quad + Re[A_{2,2}(\theta,\varphi)](\boldsymbol{T}^Q_{2,-2} + \boldsymbol{T}^Q_{2,2}) + iIm[A_{2,2}(\theta,\varphi)](\boldsymbol{T}^Q_{2,-2} - \boldsymbol{T}^Q_{2,2})\}
\end{aligned}$$

1. Keep in mind that this Hamiltonian is for a single spin of quantum number I. In a multi-spin system one will have to sum such Hamiltonians for all spins.
2. That is, the interaction z-axis is not aligned with the spectrometer z-axis.

At this point we will substitute in the spin operations

$$T^Q_{2,0} = \frac{1}{\sqrt{6}}[3I_z^2 - \vec{\boldsymbol{I}}^2] = \frac{1}{\sqrt{6}}[3I_z^2 - I(I+1)]$$

$$T^Q_{2,-1} \pm T^Q_{2,1} = \frac{1}{2}[I_-I_z + I_zI_-] \mp \frac{1}{2}[I_+I_z + I_zI_+] = \frac{1}{2}[(I_- \mp I_+)I_z + I_z(I_- \mp I_+)]$$

$$T^Q_{2,-2} \pm T^Q_{2,2} = \frac{1}{2}I_-^2 \pm \frac{1}{2}I_+^2 = \frac{1}{2}(I_-^2 \pm I_+^2)$$

to produce

$$\begin{aligned}\boldsymbol{H}^Q(\theta,\varphi) = \xi^Q\Big\{ & A_{2,0}(\theta,\varphi)\frac{1}{\sqrt{6}}[3I_z^2 - I(I+1)] \\ & + Re[A_{2,1}(\theta,\varphi)]\frac{1}{2}[(I_- + I_+)I_z + I_z(I_- + I_+)] + iIm[A^Q_{2,1}(\theta,\varphi)]\frac{1}{2}[(I_- - I_+)I_z + I_z(I_- - I_+)] \\ & + Re[(A_{2,2})(\theta,\varphi)]\frac{1}{2}(I_-^2 + I_+^2) + iIm[A_{2,2}(\theta,\varphi)]\frac{1}{2}(I_-^2 - I_+^2)\Big\}\end{aligned}$$

We can then use the identities $I_x = \frac{1}{2}(I_- + I_+)$ $\quad I_y = \frac{i}{2}(I_- - I_+)$

to obtain

$$\begin{aligned}\boldsymbol{H}^Q(\theta,\varphi) = \xi^Q\Big\{ & A_{2,0}(\theta,\varphi)\left[\frac{1}{\sqrt{6}}(3I_z^2 - \vec{\boldsymbol{I}}^2)\right] \\ & + Re[A_{2,1}(\theta,\varphi)](I_xI_z + I_zI_x) + Im[A_{2,1}(\theta,\varphi)](I_yI_z + I_zI_y) \\ & + Re[A_{2,2}(\theta,\varphi)]\left[\frac{1}{2}(I_-^2 + I_+^2)\right] + iIm[A_{2,2}(\theta,\varphi)]\left[\frac{1}{2}(I_-^2 - I_+^2)\right]\Big\}\end{aligned}$$

Upon substitution of the oriented spatial components we obtain

$$
\begin{aligned}
\frac{\boldsymbol{H}^Q(\theta,\varphi)}{\xi^Q} = & \sqrt{\frac{5}{24\pi}}\frac{1}{2}[(3\cos^2\theta - 1) + \eta\sin^2\theta\cos 2\varphi](3I_z^2 - \vec{\boldsymbol{I}}^2) \\
& + \sqrt{\frac{5}{24\pi}}\sin\theta[3\cos\theta - \eta(\cos\theta\cos 2\varphi)](I_xI_z + I_zI_x) \\
& - i\sqrt{\frac{5}{24\pi}}\eta\sin 2\varphi\sin\theta(I_yI_z + I_zI_y) \\
& + \sqrt{\frac{5}{24\pi}}\frac{1}{4}[3\sin^2\theta + \eta\cos 2\varphi(1 + \cos^2\theta)](I_-^2 + I_+^2) \\
& + \sqrt{\frac{5}{24\pi}}\frac{1}{2}\eta\sin 2\varphi\cos\theta(I_-^2 - I_+^2)
\end{aligned}
$$

$$
\begin{aligned}
\frac{\boldsymbol{H}^Q(\theta,\varphi)}{\xi^Q} = & \sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]\left[\frac{1}{\sqrt{6}}(3I_z^2 - \vec{\boldsymbol{I}}^2)\right] \\
& + \sqrt{\frac{5}{24\pi}}\sin\theta[3\cos\theta - \eta(\cos\theta\cos 2\varphi - i\sin 2\varphi)]\left[\frac{1}{2}(I_-I_z + I_zI_-)\right] \\
& - \sqrt{\frac{5}{24\pi}}\sin\theta[3\cos\theta - \eta(\cos\theta\cos 2\varphi + i\sin 2\varphi)]\left[-\frac{1}{2}(I_+I_z + I_zI_+)\right] \\
& + \sqrt{\frac{5}{24\pi}}\frac{1}{2}[3\sin^2\theta + \eta[\cos 2\varphi(1 + \cos^2\theta) - i2\sin 2\varphi\cos\theta]]\left[\frac{1}{2}I_-^2\right] \\
& + \sqrt{\frac{5}{24\pi}}\frac{1}{2}[3\sin^2\theta + \eta[\cos 2\varphi(1 + \cos^2\theta) + i2\sin 2\varphi\cos\theta]]\left[\frac{1}{2}I^2{}_+\right]
\end{aligned}
$$

Now we have some algebraic fun.

$$
\begin{aligned}
\boldsymbol{H}^Q(\theta,\varphi) = & \ \xi^Q\sqrt{\frac{5}{24\pi}}\frac{1}{2} \\
& \times\{(3\cos^2\theta - 1 + \eta\sin^2\theta\cos 2\varphi)(3I_z^2 - \vec{\boldsymbol{I}}^2) \\
& + \sin\theta[3\cos\theta - \eta(\cos\theta\cos 2\varphi - i\sin 2\varphi)](I_-I_z + I_zI_-) \\
& + \sin\theta[3\cos\theta - \eta(\cos\theta\cos 2\varphi + i\sin 2\varphi)](I_+I_z + I_zI_+) \\
& + [3\sin^2\theta + \eta[\cos 2\varphi(1 + \cos^2\theta) - i2\sin 2\varphi\cos\theta]]\left[\frac{1}{2}I_-^2\right] \\
& + [3\sin^2\theta + \eta[\cos 2\varphi(1 + \cos^2\theta) + i2\sin 2\varphi\cos\theta]]\left[\frac{1}{2}I^2{}_+\right]\Bigg\}
\end{aligned}
$$

$$
\begin{aligned}
\boldsymbol{H}^Q(\theta, \varphi) \;=\; & \frac{\omega^Q}{12} \\
& \times \{(3\cos^2\theta - 1 + \eta\sin^2\theta\cos 2\varphi)(3I_z^2 - \vec{\boldsymbol{I}}^2) \\
& + \sin\theta[3\cos\theta - \eta(\cos\theta\cos 2\varphi)][(I_+ + I_-)I_z + I_z(I_+ + I_-)] \\
& - i\eta\sin\theta\sin 2\varphi[(I_+ - I_-)I_z + I_z(I_+ - I_-)] \\
& + \frac{1}{2}[3\sin^2\theta + \eta\cos 2\varphi(1 + \cos^2\theta)][I_+^2 + I_-^2] \\
& + i\eta\sin 2\varphi\cos\theta[I_+^2 - I_-^2]\}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\boldsymbol{H}^Q(\theta, \varphi)}{\xi^Q} \;=\; & \frac{\omega^Q}{12} \\
& \times \{(3\cos^2\theta - 1 + \eta\sin^2\theta\cos 2\varphi)(3I_z^2 - \vec{\boldsymbol{I}}^2) \\
& + \sin\theta[3\cos\theta - \eta(\cos\theta\cos 2\varphi)][(I_+ + I_-)I_z + I_z(I_+ + I_-)] \\
& - i\eta\sin\theta\sin 2\varphi[(I_+ - I_-)I_z + I_z(I_+ - I_-)] \\
& + \frac{1}{2}[3\sin^2\theta + \eta\cos 2\varphi(1 + \cos^2\theta)][I_+^2 + I_-^2] \\
& + i\eta\sin 2\varphi\cos\theta[I_+^2 - I_-^2]\}
\end{aligned}
$$

$$
\begin{aligned}
\boldsymbol{H}^Q(\theta, \varphi) \;=\; & \frac{\omega^Q}{12} \\
& \times \{(3\cos^2\theta - 1 + \eta\sin^2\theta\cos 2\varphi)(3I_z^2 - \vec{\boldsymbol{I}}^2) \\
& + \sin\theta[3\cos\theta - \eta(\cos\theta\cos 2\varphi)](I_-I_z + I_zI_- + I_+I_z + I_zI_+) \\
& - i\eta\sin\theta\sin 2\varphi(I_+I_z + I_zI_+ - I_-I_z - I_zI_-) \\
& + \frac{1}{2}[3\sin^2\theta + \eta\cos 2\varphi(1 + \cos^2\theta)][I_+^2 + I_-^2] \\
& + i\eta\sin 2\varphi\cos\theta[I_+^2 - I_-^2]\}
\end{aligned}
$$

$$\frac{\boldsymbol{H}^Q(\theta,\varphi)}{\xi^Q} = \left[A^Q_{2,0}(\theta,\varphi)\left[\frac{1}{\sqrt{6}}(3I_z^2-\vec{\boldsymbol{I}}^2)\right] + A^Q_{2,2}(\theta,\varphi)\left[\frac{1}{2}I_-^2\right] + A^Q_{2,2}{}^*(\theta,\varphi)\left[\frac{1}{2}I^2{}_+\right]\right]$$

$$= \sqrt{\frac{5}{4\pi}}\sqrt{\frac{1}{24}}[3\cos^2\theta - 1 + \eta\sin^2\theta\cos 2\varphi](3I_z^2-\vec{\boldsymbol{I}}^2)$$

$$+ \sqrt{\frac{5}{24\pi}}\frac{1}{4}[3\sin^2\theta + \eta\cos 2\varphi(1+\cos^2\theta)](I_+^2+I_-^2)$$

$$+ \sqrt{\frac{5}{24\pi}}\frac{i}{2}\eta\sin 2\varphi\cos\theta(I_+^2-I_-^2)$$

$$\frac{\boldsymbol{H}^Q(\theta,\varphi)}{\xi^Q} = \sqrt{\frac{5}{24\pi}}\frac{1}{2}[\{3\cos^2\theta - 1 + \eta\sin^2\theta\cos 2\varphi\}(3I_z^2-\vec{\boldsymbol{I}}^2)$$

$$+ \frac{1}{2}[3\sin^2\theta + \eta\cos 2\varphi(1+\cos^2\theta)](I_+^2+I_-^2) + i\eta\sin 2\varphi\cos\theta(I_+^2-I_-^2)\Big]$$

$$\boldsymbol{H}^Q(\theta,\varphi) = \frac{\omega^Q}{12(4)}[\{3\cos^2\theta - 1 + \eta\sin^2\theta\cos 2\varphi\}(3I_z^2-\vec{\boldsymbol{I}}^2)$$

$$+ \frac{1}{2}[3\sin^2\theta + \eta\cos 2\varphi(1+\cos^2\theta)](I_+^2+I_-^2) + i\eta\sin 2\varphi\cos\theta(I_+^2-I_-^2)\Big]$$

Finally, note that when working with an entire spin system one must sum over all spins with the tensors being in the same coordinate system, for our purposes the laboratory system. The quadrupolar Hamiltonian for a spin system becomes the following.

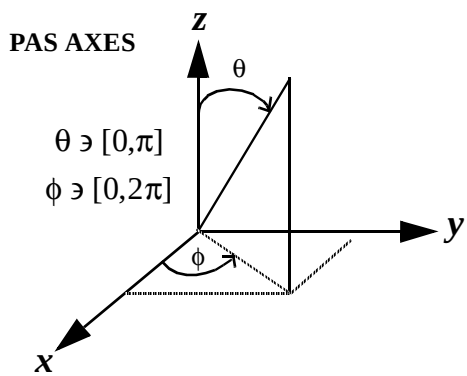
$$\boldsymbol{H}^Q = \sum_i^{spins} \boldsymbol{H}_i^Q = \sum_i^{spins} \xi_i^Q \sum_m^{\pm 2} (-1)^m A_{2,-m}(i) \bullet \boldsymbol{T}^Q_{2,m}(i) \tag{39}$$

4.17.16 Quadrupolar PAS Equations

When the quadrupolar interaction has alignment along its principal axes system virtually all of the quadrupolar equations simplify. The following figure collects all of these for convenience.

Quadrupolar Equations Involving the PAS

PAS AXES



$$\boldsymbol{H}^Q(PAS) = \xi^Q\sqrt{\frac{5}{24\pi}}\left[3I_z^2 - \check{\boldsymbol{I}}^2 + \frac{1}{2}\eta(I_-^2 + I_+^2)\right]$$

$$= \frac{e^2qQ}{4I(2I-1)}\left[3I_z^2 - \check{\boldsymbol{I}}^2 + \frac{1}{2}\eta(I_-^2 + I_+^2)\right]$$

$$= \frac{\omega^Q}{6}\left[3I_z^2 - \check{\boldsymbol{I}}^2 + \frac{1}{2}\eta(I_-^2 + I_+^2)\right]$$

$$\xi^Q = \frac{e^2qQ}{2I(2I-1)}\sqrt{\frac{6\pi}{5}}$$

$$= \frac{QCC}{2I(2I-1)}\sqrt{\frac{6\pi}{5}} = \sqrt{\frac{2\pi}{15}}\omega^Q$$

$$\eta = (A_{xx} - A_{yy})/A_{zz}$$

$$|A_{zz}| \geq |A_{yy}| \geq |A_{xx}|$$

$$A_{2,0}(PAS) = \sqrt{6}[3A_{zz} - Tr\{A\}]\big|_{PAS} = \sqrt{\frac{5}{4\pi}}$$

$$A_{2,\pm1}(PAS) = \mp\frac{1}{2}[A_{xz} + A_{zx} \pm i(A_{yz} + A_{zy})]\Big|_{PAS} = 0$$

$$A_{2,\pm2}(PAS) = \frac{1}{2}[A_{xx} - A_{yy} \pm i(A_{xy} + A_{yx})]\big|_{PAS} = \sqrt{\frac{5}{24\pi}}\eta$$

$$A_{xx}(PAS) = \sqrt{\frac{5}{24\pi}}[\eta - 1] \qquad A_{yy}(PAS) = -\sqrt{\frac{5}{24\pi}}[1 + \eta] \qquad A_{zz}(\theta, \varphi) = \sqrt{\frac{5}{6\pi}}$$

$$A_{xz}(PAS) = 0 = A_{zx}(PAS) = A_{xy}(PAS) = A_{yx}(PAS) = A_{yz}(PAS) = A_{zy}(PAS)$$

Figure 19-19 **Equations relevant to the Quadrupolar interaction in its principal axis orientation (PAS). GAMMA uses a spatial tensor which is scaled[1] so that rotations by angles θ & ϕ produce spherical harmonics for a symmetric interaction (η = 0).**

Included are the general relationships between the (GAMMA scaled) Cartesian tensor components to the irreducible spherical components. They are valid when η is defined accordingly! If η is defined by the other common convention ($|A_{zz}| \geq |A_{xx}| \geq |A_{yy}|$) then the sign on the $A_{2,\pm2}$ will change as will the sign on the Hamiltonian term multiplied by η.

1. The scaling on both $\{A_{2m}\}$ and $T_{2m}\}$ are arbitrary, GAMMA uses an (uncommon) scaling which independent of the interaction type. What is NOT arbitrary is the scaling within either of the two sets of components, e.g. one cannot scale A_{20} differently than A_{21}. In addition, the combined scaling of the two sets is critical to the proper formation of quadrupolar Hamiltonians. For that, GAMMA uses an interaction constant.

4.17.17 Quadrupolar Equations At Any Orientation

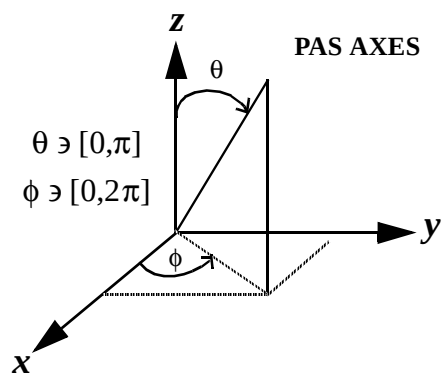
When the quadrupolar interaction has a arbitrary alignment (relative to its principal axes system) the quadrupolar equations become complicated. The figure below depicts them for convenience.

Quadrupolar Equations When Oriented At Angles {θ,ϕ}[1]

$$\boldsymbol{H}^Q(\theta,\varphi) = \xi^Q \sum_m^{\pm 2} (-1)^m A^Q_{2,-m}(\theta,\varphi) \bullet \boldsymbol{T}^Q_{2,m}$$

$$T^Q_{2,0} = \frac{1}{\sqrt{6}}[3I_z^2 - \vec{\boldsymbol{I}}^2] = \frac{1}{\sqrt{6}}[3I_z^2 - I(I+1)]$$

$$T^Q_{2,\pm 1} = \mp\frac{1}{2}[I_\pm\ I_z + I_z I_\pm\ ] \qquad T^Q_{2,\pm 2} = \frac{1}{2}I_\pm^2$$



$$A_{2,0}(\theta,\varphi) = \sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]$$

$$A_{2,1}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}\sin\theta[3\cos\theta - \eta(\cos\theta\cos 2\varphi - i\sin 2\varphi)] = -A^*_{2,-1}(\theta,\varphi)$$

$$A_{2,2}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}\frac{1}{2}[3\sin^2\theta + \eta[\cos 2\varphi(1+\cos^2\theta) - i2\sin 2\varphi\cos\theta]] = A^*_{2,-2}(\theta,\varphi)$$

$$\eta = (A_{xx} - A_{yy})/A_{zz} \qquad |A_{zz}| \geq |A_{yy}| \geq |A_{xx}| \qquad \xi^Q = \frac{e^2qQ}{2I(2I-1)}\sqrt{\frac{6\pi}{5}} = \sqrt{\frac{2\pi}{15}}\omega^Q$$

$$\boldsymbol{H}^Q(\theta,\varphi) = \frac{\omega^Q}{4}[\{3\cos^2\theta - 1 + \eta\sin^2\theta\cos 2\varphi\}(3I_z^2 - \vec{\boldsymbol{I}}^2)$$
$$+ \frac{1}{2}[3\sin^2\theta + \eta\cos 2\varphi(1+\cos^2\theta)](I_+^2 + I_-^2) + i\eta\sin 2\varphi\cos\theta(I_+^2 - I_-^2)\Big]$$

Figure 19-20 **Equations relevant to the Quadrupolar Hamiltonian when oriented at angles θ & ϕ from the principal axis orientation (PAS). GAMMA uses a spatial tensor which is scaled[2] so that rotations by angles θ & ϕ produce spherical harmonics for a symmetric interaction (η = 0).**

1. The quadrupolar interaction constant, as well as the relative scalings on the sets of spatial and spin tensors, can be adjusted as desired. However all components of the space or spin tensor must be adjusted by the same scaling. The GAMMA scaling is oriented to liquids where so that all spatial components are related to the spherical harmonics in the spatial tensor PAS.
2. The scaling on both $\{A_{2m}\}$ and $T_{2m}\}$ are arbitrary, GAMMA uses a scaling which independent of the interaction type. What is NOT arbitrary is the scaling within either of the two sets of components. In addition, the combined scaling of the two sets is also crucial. For that, GAMMA uses an interaction constant.

4.18 Parameters

This section describes how an ASCII file may be constructed that is self readable by a quadrupolar interaction. The file can be created with any editor and is read with the quadrupolar interaction member function “read”. It is important to keep in mind the structure of our interaction as given in Section 4.17.2 on page 233. We will need the set of **{**I, δ_{zz}**,** η, θ, ϕ**}** specified for each quadrupolar interaction we wish to create. Of course, there are several other ways of declaring a quadrupolar tensor than with the use of these five values. To accomodate different tensor nomenclature (i.e. spherical *versus* Cartesian, oriented *versus* PAS, etc.), GAMMA quadrupolar interactions will recognize different sets of parameters! Thus, the following sections detail “parameter sets” which can be used for this purpose.

4.18.1 Spherical Tensor Parameter Set 1

Perhaps the simplest way to designate a quadrupolar interaction is to provide the quadrupolar coupling constant, an asymmetry parameter, and a tensor orientation relative to the principal axis system. This would be the set of parameters {I, QCC, η, θ, ϕ**}**.

Table 4: Quadrupolar Interaction Parameters - Set 1

Parameter	Units	Examples Parameter (Type) : Value - Statement
QI	none	QI __ (1) : 1.5 - Spin Quantum Number
QCC	KHz	QCC (1) : 370.3 - Quadrupolar Coupling (KHz)
Qeta	none	Qeta (1) : 0.33 - Quadrupolar Asymmetry
Qtheta	degrees	Qtheta (1) : 127.2 - Quad. Orientation from PAS z (deg)
Qphi	degrees	Qphi (1) : 270.9 - Quad. Orientation from PAS x(deg)

Table 4-0 **Generic ASCII parameters to declare a GAMMA quadrupolar interaction.**

By including these parameter statements (right column) in an ASCII file a GAMMA quadrupolar interaction can be set with the read function. For example, the code below reads “file.asc”.

Simple Read of Quadrupolar Interaction From An ASCII File

file.asc

```
QI      (1) : 1.5      - Spin Quantum Number
QCC     (1) : 370.3    - Quadrupolar Coupling (KHz)
Qeta    (1) : 0.33     - Quadrupolar Asymmetry
Qtheta  (1) : 127.2    - Quad. Orientation from PAS z (deg)
Qphi    (1) : 270.9    - Quad. Orientation from PAS x(deg)
```

code.cc

```
.............
.............
IntQuad Q;
Q.read("file.asc");
.............
```

Figure 19-21 **Specifying a quadrupolar interaction using an external ASCII file.**

Three things to remember; 1.) These ASCII files are read as GAMMA parameter set files so that they may contain additional likes of information and additional parameters. Things such as column spacing is not important - read about GAMMA parameters sets for full details. 2.) All parameters can have a (#) appended to the name and parameters with that number will be selected if the number is given as an argument in the read function. 3.) There are name variations allowed even within these five parameters. For example, users may specify a quadrupolar frequency rather than a quadrupolar coupling constant. The spin quantum number can be read from an isotope declaration rather than with use of parameter QI. Another variation would be to put an interaction “index” on these parameters when multiple interactions must be designated within the same ASCII file. Here are the possibilities.

Quadrupolar Spin Quantum Number: QI, Iso

The quadrupolar spin quantum number must be specified. This can be accomplished with parameters using either of the two names below or these names with a (#) added as a suffix. The value of QI must be a positive multiple of 1/2 and greater than 1/2. The value of Iso must designate a spin isotope having a spin quantum number greater than 1/2. If both QI and Iso are set in the same file, the QI value will be used to set up the quadrupolar interaction..

Table 5: Quadrupolar Spin Quantum Value[a]

Parameter	Assumed Units	Examples Parameter (Type=1,3) : Value - Statement
QI	none	QI (1) : 1.5 - Quad. Spin Quantum Value
Iso	none	Iso (2) : 131Xe - Quad. Spin Type

a. Shown are two possible parameters used to set the quadrupolar spin quantum number. Parameter type 1 indicates a double precision number parameter. Parameter type 2 indicates a string parameter.

Quadrupolar Frequency: WQ, WQkHz, WQKHz, WQHz, WQMHz

The quadrupolar frequency can be specified. This can be accomplished with parameters using any of the names below or these names with a (#) added as a suffix. The default units for WQ are KHz other names can be used to set the value in particular units. Note that this parameter is related to the quadrupolar coupling constant which is specified with “(N)QCC” parameters. If both QCC and WQ are set in the same file, the quadrupolar frequency will be used to set up the quadrupolar interaction.

Table 6: Quadrupolar Frequency[a]

Parameter	Assumed Units	Examples Parameter (Type=1) : Value - Statement
WQ	KHz	WQ (1) : 320.13 - Quad. Frequency in kHz
WQMHz	MHz	WQMHz (1) : 1.27 - Quad. Frequency in MHz
WQHz	Hz	WQHz(2) (1) : 1320.7 - Quad. Frequency in Hz

a. Shown are three possible parameters used to set the quadrupolar frequency. The others mentioned above can also be used to specify it. Specification of a quadrupolar coupling constant will also set the interaction's quadrupolar frequency. Parameter type 1 indicates a double precision number parameter

Quadrupolar Coupling Constant: QCC, QCCkHz, QCCKHz, QCCHz, QCCMHz

The quadrupolar coupling constant can be specified. This can be accomplished with parameters using any of the names above, these same names with an "N" as a prefix, and/or these names with a (#) added as a suffix. The default units for QCC are KHz other names can be used to set the value in particular units. Note that this parameter is related to the quadrupolar frequency which is specified with "WQ" parameters. If both QCC and WQ are set in the same file, the quadrupolar frequency will be used to set up the quadrupolar interaction.

Table 7: Quadrupolar Coupling Constant[a]

Parameter	Assumed Units	Examples Parameter (Type=1) : Value - Statement
QCC	KHz	QCC (1) : 320.13 - Quad. Coupling in kHz
NQCCMHz	MHz	NQCCMHz (1) : 1.27 - Quad. Coupling in MHz
QCCHz	Hz	QCCHz(2) (1) : 1320.7 - Quad. Coupling in Hz

a. Shown are three possible parameters used to set the quadrupolar coupling. The others mentioned above can also be used to specify it. Specification of a quadrupolar frequency will also set the quadrupolar coupling in the interaction. Parameter type 1 indicates a double precision number parameter

Quadrupolar Asymmetry

The asymmetry parameter must be within the range of [0, 1]. This parameter does not need to be set for a quadrupolar interaction definition, it will be assumed 0 if unspecified.

Table 8: Quadrupolar Asymmetry[a]

Parameter	Assumed Units	Examples Parameter (Type=1) : Value - Statement
Qeta	none	Qeta (1) : 0.4 - Quadrupolar Asymmetry

a. Parameter type 1 indicates an integer parameter.

Quadrupolar Theta Orientation

The angle theta which relates the quadrupolar interactions orientation down from the z-axis of its PAS may be set. This is not essential and will be taken as zero in left unspecified.

Table 9: Theta Orientation[a]

Parameter	Assumed Units	Examples Parameter (Type=1) : Value - Statement
Qtheta	degrees	Qtheta (1) : 45.7 - Quadrupolar Orientation from PAS z

a. Parameter type 1 indicates an integer parameter.

Quadrupolar Phi Orientation

The angle phi which relates the quadrupolar interactions orientation over from the x-axis of its PAS may be set. This is not essential and will be taken as zero in left unspecified.

Table 10: Theta Orientation[a]

Parameter	Assumed Units	Examples Parameter (Type=1) : Value - Statement
Qphi	degrees	Qphi (1) : 134.6 - Quadrupolar Orientation from PAS x

a. Parameter type 1 indicates an integer parameter.

4.18.2 Spherical Tensor Parameter Set 2

Perhaps the simplest way to designate a quadrupolar interaction is to provide the quadrupolar coupling constant, an asymmetry parameter, and a tensor orientation relative to the principal axis system. This would be the set of parameters {I, QCC, η, θ, ϕ}.

Table 11: Quadrupolar Interaction Parameters - Set 1

Parameter	Units	Examples Parameter (Type) : Value - Statement
QI	none	QI __ (1) : 1.5 - Spin Quantum Number
QCC	KHz	QCC (1) : 370.3 - Quadrupolar Coupling (KHz)
Qeta	none	Qeta (1) : 0.33 - Quadrupolar Asymmetry
Qtheta	degrees	Qtheta (1) : 127.2 - Quad. Orientation from PAS z (deg)
Qphi	degrees	Qphi (1) : 270.9 - Quad. Orientation from PAS x(deg)

Table 11-0 **Generic ASCII parameters to declare a GAMMA quadrupolar interaction.**

By including these parameter statements (right column) in an ASCII file a GAMMA quadrupolar interaction can be set with the read function. For example, the code below reads “file.asc”.

Simple Read of Quadrupolar Interaction From An ASCII File

file.asc

```
QI      (1) : 1.5      - Spin Quantum Number
QCC     (1) : 370.3    - Quadrupolar Coupling (KHz)
Qeta    (1) : 0.33     - Quadrupolar Asymmetry
Qtheta  (1) : 127.2    - Quad. Orientation from PAS z (deg)
Qphi    (1) : 270.9    - Quad. Orientation from PAS x(deg)
```

code.cc

```
.............
.............
IntQuad Q;
Q.read("file.asc");
.............
```

Figure 19-22 **Specifying a quadrupolar interaction using an external ASCII file.**

4.18.3 Spherical Tensor Parameter Set 2

4.19 Literature Comparisons

4.19.1 P.P. Man's "Quadrupolar Interactions"

The following figure lists some of the equations found in Pascal P. Man's article[1] along with the corresponding GAMMA equations. Aside from difference in scaling factors, GAMMA is in full agreement with these equations.

Comparison of Γ & P.P. Man's Quadrupolar Equations

Γ's Equations

$$\boldsymbol{H}^Q = \xi^Q \sum_m^{\pm 2} (-1)^m A^Q_{2,-m} \bullet \boldsymbol{T}^Q_{2,m}$$

$$T^Q_{2,0} = \frac{1}{\sqrt{6}}[3I_z^2 - I(I+1)] \qquad T^Q_{2,\pm 1} = \mp\frac{1}{2}[I_\pm\ I_z + I_z I_\pm\ ] \qquad T^Q_{2,\pm 2} = \frac{1}{2}I_\pm^2$$

$$A^Q_{2,0}(PAS) = \sqrt{\frac{5}{4\pi}} \qquad A^Q_{2,\pm 1}(PAS) = 0 \qquad A^Q_{2,\pm 2}(PAS) = \sqrt{\frac{5}{24\pi}}\eta$$

$$\xi^Q = \frac{e^2qQ}{2I(2I-1)}\sqrt{\frac{6\pi}{5}} = \frac{QCC}{2I(2I-1)}\sqrt{\frac{6\pi}{5}} = \sqrt{\frac{2\pi}{15}}\omega^Q$$

$$\boldsymbol{H}^Q(PAS) = \frac{e^2qQ}{4I(2I-1)}\left[3I_z^2 - I(I+1) + \frac{\eta}{2}(I_+^2 + I_-^2)\right]$$

P.P. Man's Equations

$$\hat{\boldsymbol{H}}_Q = \frac{eq}{4I(2I-1)} \sum_{q=-2}^{2} (-1)^q V^{(2,q)} \bullet T^{(2,-q)}$$

$$T^{(2,0)} = \frac{\sqrt{6}}{3}[3I_z^2 - I(I+1)] \qquad T^{(2,\pm 1)} = \mp[I_\pm\ I_z + I_z I_\pm\ ] \qquad T^{(2,\pm 2)} = I_\pm^2$$

$$V^{(2,0)}(PAS) = \sqrt{\frac{3}{2}}eq \qquad v^{(2,\pm 1)}(PAS) = 0 \qquad V^{(2,\pm 2)}(PAS) = \frac{1}{2}\eta eq$$

$$\hat{\boldsymbol{H}}_Q(PAS) = \frac{e^2qQ}{4I(2I-1)}\left[3I_z^2 - I(I+1) + \frac{\eta}{2}(I_+^2 + I_-^2)\right]$$

Equations Shared By Γ & P.P. Man

$$\eta = (A_{xx} - A_{yy})/A_{zz} \qquad |A_{zz}| \geq |A_{yy}| \geq |A_{xx}|$$

Note that Man puts an hbar in front of his Hamiltonians to indicate that they are expressed in angular frequency units (I've left them out). In GAMMA, the Hamiltonian functions associated with

1. "Quadrupolar Interactions", P.P. Man, Encyclopedia of Nuclear Magnetic Resonance, Editors-in-Chief D.M.Grant and R.K. Harris, Vol. 6, Ped-Rel, pgs. 3838-3948.

class IntQuad will typically be in units of Hz. To be precise, they will have the units that are used during construction of the interaction (e.g. if QCC is set in Hz, the returned Hamiltonian(s) will be in Hz as well).

The following table indicates the variables in Man's equations and the conversion factor required to switch between his nomenclature and GAMMA's.

Table 12: Γ & P.P. Man's Quadrupolar Equation Variables

Mathematical Construct	Γ	Man	Γ = X * Man
Spin Tensor Components	$T^{Q}_{2,m}$	$T^{(2,q)}$	$X = 1/2$
Spatial Tensor Components	$A^{Q}_{2,m}$	$V^{(2,q)}$	$X = (eq)^{-1}\sqrt{\frac{5}{6\pi}}$
(Interaction) Constant	ξ^{Q}	$\frac{eQ}{4I(2I-1)}$	$X = 2eq$
Other	$\xi^{Q}A^{Q}_{2,m}$	$\frac{eQ}{4I(2I-1)}V^{(2,q)}$	$X = 2\sqrt{\frac{5}{6\pi}}$
Quadrupolar Hamiltonian	$\boldsymbol{H}^{Q}$	$\hat{\boldsymbol{H}}_{Q}$	1[a]

a. Man says his Hamiltonians are in angular frequency units, but the Hamiltonian units in GAMMA are relative. So there may indeed be a conversion factor but that depends on what on chooses to express is "eQ" factor in.

Of the equations tested, GAMMA is in perfect agreement with Man's equations. The correspondence benefits from use of the same PAS definition

$$|A_{zz}| \geq |A_{yy}| \geq |A_{xx}|$$

Users who wish to make direct computational comparisons between the two treatments should see the literature comparison programs IntQu_LC0.cc and IntQu_LC2.cc which are found at the end of this chapter.

4.19.2 Alexander Vega's "Quadrupolar Nuclei in Solids"

The following figure lists some of the equations found in Alexander Vega's article[1] along with the corresponding GAMMA equations. Please take careful not of the differences between this article and what GAMMA's IntQuad class contains.

Comparison of GAMMA & A.J. Vega's Quadrupolar Equations[2]

Γ's Equations

$$\boldsymbol{H}^Q = \xi^Q \sum_m^{\pm 2} (-1)^m A^Q_{2,-m} \bullet \boldsymbol{T}^Q_{2,m}$$

$$T^Q_{2,0} = \frac{1}{\sqrt{6}}[3I_z^2 - I(I+1)] \qquad T^Q_{2,\pm 1} = \mp\frac{1}{2}[I_\pm\ I_z + I_z I_\pm\ ] \qquad T^Q_{2,\pm 2} = \frac{1}{2}I_\pm^2$$

$$A^Q_{2,0}(PAS) = \sqrt{\frac{5}{4\pi}} \qquad A^Q_{2,\pm 1}(PAS) = 0 \qquad A^Q_{2,\pm 2}(PAS) = \sqrt{\frac{5}{24\pi}}\eta$$

$$\xi^Q = \frac{e^2qQ}{2I(2I-1)}\sqrt{\frac{6\pi}{5}} = \frac{QCC}{2I(2I-1)}\sqrt{\frac{6\pi}{5}} = \sqrt{\frac{2\pi}{15}}\omega^Q$$

$$\boldsymbol{H}^Q(PAS) = \frac{e^2qQ}{4I(2I-1)}\left[3I_z^2 - I(I+1) + \frac{\eta}{2}(I_+^2 + I_-^2)\right]$$

$$\eta = (A_{xx} - A_{yy})/A_{zz} \qquad |A_{zz}| \geq |A_{yy}| \geq |A_{xx}|$$

P.P. Man's Equations

$$\hat{\boldsymbol{H}}_Q = \frac{1}{2}\sum_{k=-2}^{2} (-1)^k \hat{Q}_k \bullet V_{-k} \qquad C = \frac{eQ}{2I(2I-1)}$$

$$\hat{Q}_0 = \frac{2}{\sqrt{6}}C[3I_z^2 - I(I+1)] \qquad \hat{Q}_{\pm 1} = \mp C[I_\pm\ I_z + I_z I_\pm\ ] \qquad \hat{Q}_{\pm 2} = CI_\pm^2$$

$$V_0(PAS) = \sqrt{\frac{3}{2}}eq \qquad V_{\pm 1}(PAS) = 0 \qquad V_{\pm 2}(PAS) = -\frac{1}{2}\eta eq$$

$$\hat{\boldsymbol{H}}_Q(PAS) = \frac{e^2qQ}{4I(2I-1)}\left[3I_z^2 - I(I+1) - \frac{\eta}{2}(I_+^2 + I_-^2)\right]$$

$$\eta = (V_{yy} - V_{xx})/V_{zz} \qquad |V_{zz}| \geq |V_{xx}| \geq |V_{yy}|$$

Note that Vega puts an hbar in front of his Hamiltonians to indicate that they are expressed in an-

1. "Quadrupolar Nuclei in Solids", Alexander J. Vega, Encyclopedia of Nuclear Magnetic Resonance, Editors-in-Chief D.M.Grant and R.K. Harris, Vol. 6, Ped-Rel, pgs 3869-3889.
2. I've taken the liberty to correct Vega's $\hat{Q}_0$ equation herein which could not (should not) be correct as printed. Note that Vega's PAS definition does NOT coincide with GAMMA's, thus the sign on η must be changed on all equations containing it if one is to do comparisons.

gular frequency units (I've left them out). In GAMMA, the Hamiltonian functions associated with class IntQuad will typically be in units of Hz. To be precise, they will have the unit that are used during construction of the interaction (e.g. if QCC is set in Hz, the returned Hamiltonian(s) will be in Hz as well).

The following table indicates the variables in Vega's equations and the conversion factor required to switch between his nomenclature and GAMMA's.

Table 13: Γ & A.J. Vega's Quadrupolar Equation Variables

Mathematical Construct	Γ	Vega	Γ = X * Vega
Spin Tensor Components	$T^{Q}_{2,m}$	$\hat{Q}_k$	$X = \frac{I(2I-1)}{eQ}$
Spatial Tensor Components	$A_{2,m}$	V_k	$X = (eq)^{-1}\sqrt{\frac{5}{6\pi}}$ [a]
(Interaction) Constant	ξ^{Q}	$1/2$	$X = \frac{e^2qQ}{I(2I-1)}$
Quadrupolar Hamiltonian	$\boldsymbol{H}^{Q}$	$\hat{\boldsymbol{H}}_Q$	1[b]

a. Because Vega uses a different convention for defining his PAS than GAMMA, a sign change is requited to relate
b. Vega says his Hamiltonians are in angular frequency units, but the Hamiltonian units in GAMMA are relative. So there may indeed be a conversion factor but that depends on what on chooses to express is "eQ" factor in.

Of the equations tested, GAMMA is NOT in very good agreement with Vega's equations. A small part of the problem is that Vega has used a different convention for his PAS definition that does GAMMA.

$$\text{GAMMA: } |A_{zz}| \geq |A_{yy}| \geq |A_{xx}| \qquad \text{VEGA: } |A_{zz}| \geq |A_{xx}| \geq |A_{yy}|$$

This simply causes all of his equations to have the sign changed on η . Of a more serious nature is the (misprint?) definition of the m=0 spin tensor component. The one in the article is NOT a rank 2 irreducible tensor component. Although the scaling factor on all components can be (simultaneously) changed, the relationship between the components must be strictly adhered to or else there will be trouble when the quadrupolar Hamiltonian is rotated in space.

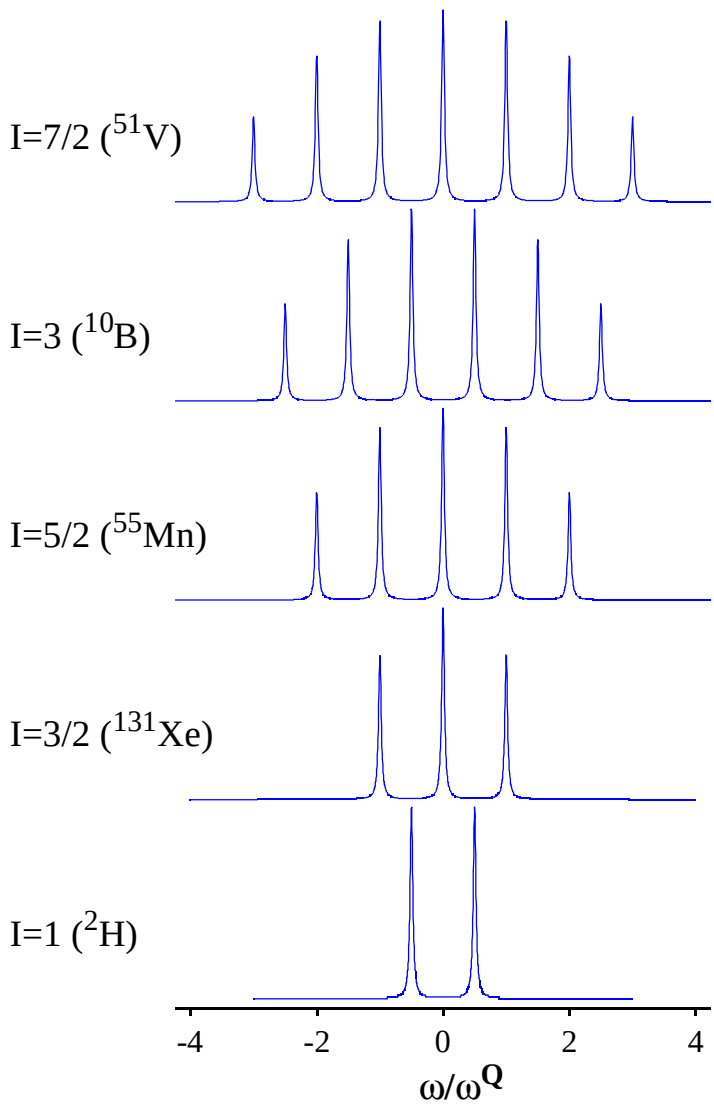
In addition, it seems that Vega's Hamiltonian for an I=1 case, his Eq. (32) is incorrect and does NOT correspond to (28), although the I=3/2 case in Eq. (33) appears correct. Users who wish to make direct computational comparisons between the two treatments should see the literature comparison programs IntQu_LC0.cc - IntQu_LC3.cc which are found at the end of this chapter.

4.20 Examples

4.20.1 Zero Field Transitions, First Order Spectra

As a first example we'll look into some of the quadrupolar Hamiltonians provided by class IntQuad in the interaction PAS (principal axes). Our results for both the transitions at zero field and NMR spectra to first order should agree with A. J. Vega's article[1] figures 1 & 2.

First Order Quadrupolar Spectra



I=7/2				
ω	$\pm 3\omega^Q$	$\pm 2\omega^Q$	$\pm \omega^Q$	0
I	3.5	6.0	7.5	8.0

I=3			
ω	$\pm 2.5\omega$	$\pm 1.5\omega$	$\pm 0.5\omega^Q$
I	3.0	5.0	6.0

I=5/2			
ω	$\pm 2\omega^Q$	$\pm \omega^Q$	0
I	2.5	4.0	4.5

I=3/2		
ω	$\pm \omega^Q$	0
I	1.5	2.0

I=1	
ω	$\pm 0.5\omega^Q$
I	1.0

Figure 19-23 **Spectra produced by program IntQu_LC6.cc, page -284. The quadrupolar frequency was set to 300 kHz. The interaction was in its PAS and the asymmetry set to zero. Zero field transitions & relative intensities are shown in the tables.**

1. "Quadrupolar Nuclei in Solids", Alexander J. Vega, Encyclopedia of Nuclear Magnetic Resonance, Editors-in-Chief D.M.Grant and R.K. Harris, Vol. 6, Ped-Rel, pgs 3869-3889.

4.20.2 First Order Powder Spectra

For our next example we'll use the first order quadrupolar Hamiltonian provided by class IntQuad to generate some powder patterns. Our results should agree with A. J. Vega's article[1] Figures 3.

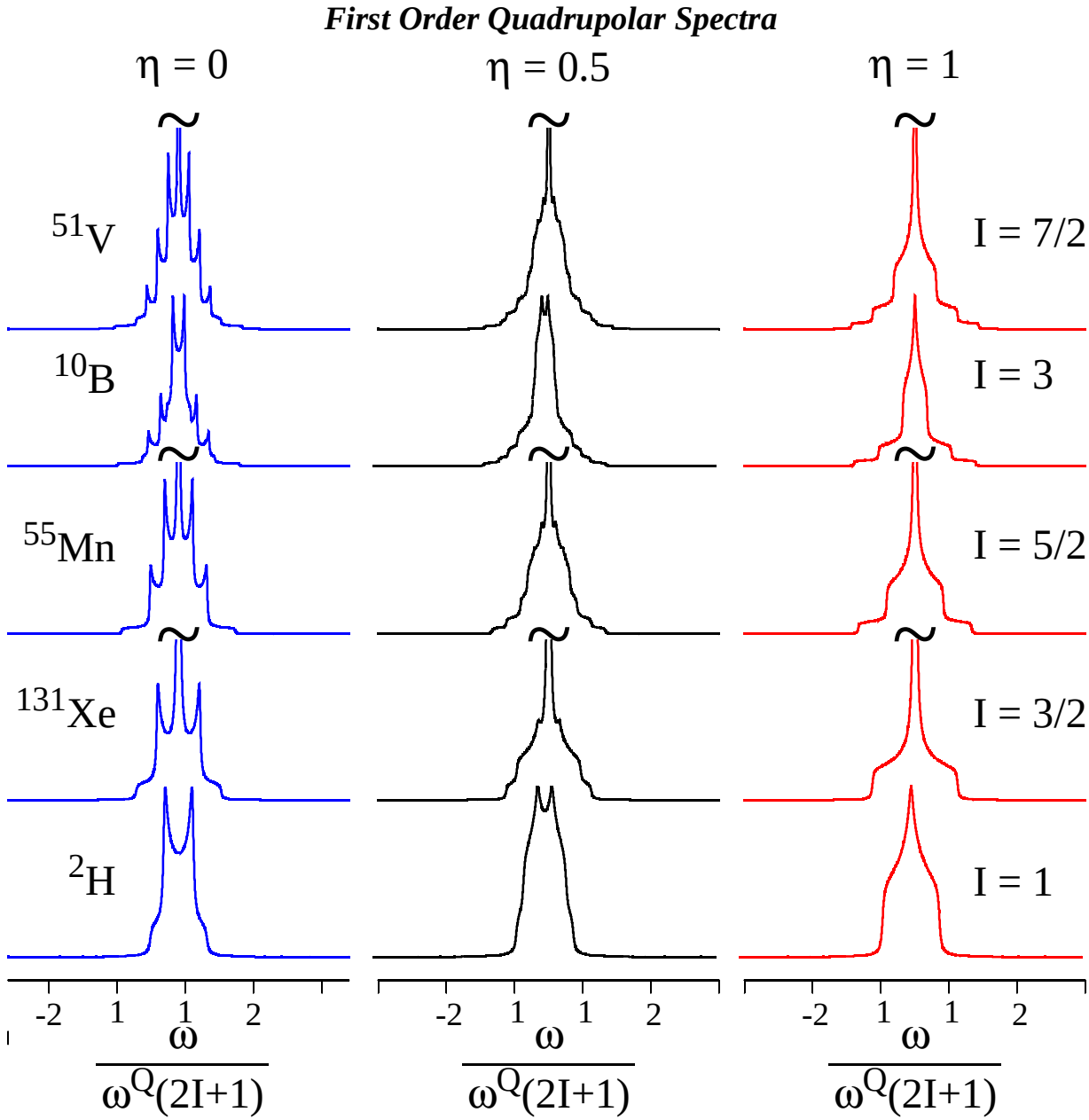


Figure 19-24 **Spectra produced by program IntQu_Pow3.cc, page -286. The quadrupolar frequency was set to 300 kHz. A Cheng value of 19 was used in all cases. A sever line-broadening was use (perhaps a bit to much...).**

1. "Quadrupolar Nuclei in Solids", Alexander J. Vega, Encyclopedia of Nuclear Magnetic Resonance, Editors-in-Chief D.M.Grant and R.K. Harris, Vol. 6, Ped-Rel, pgs 3869-3889.

4.20.3 Second Order Powder Spectra, Central Transition

We'll now use the second order quadrupolar function provided by class IntQuad to generate some powder patterns. Specifically we will reproduce A. J. Vega's article[1] Figure 10. This is the central transition in a quadrupolar spin having an I that is an odd multiple of 1/2.

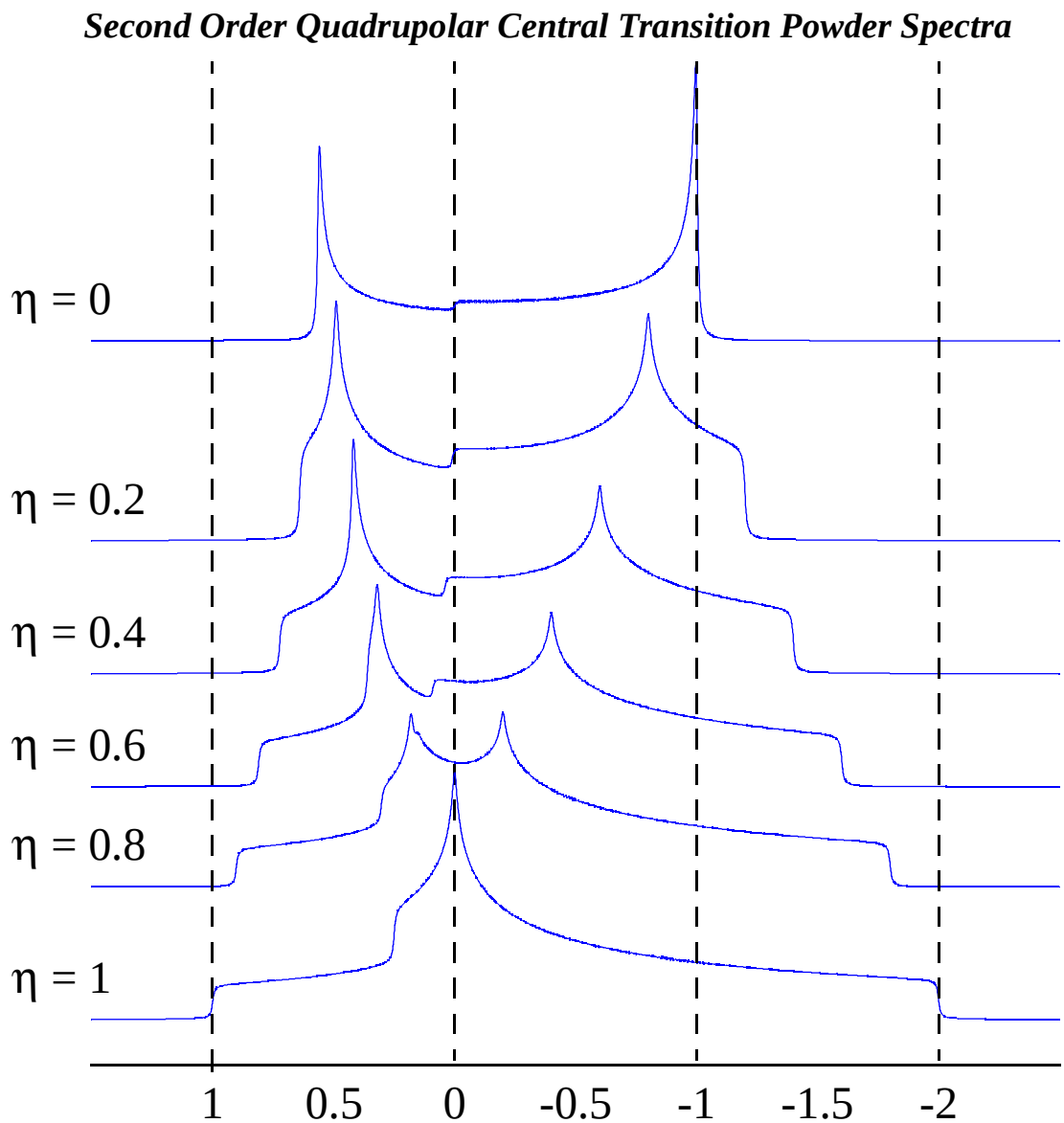


Figure 19-25 **Spectra produced by program IntQu_PCT0.cc, page -288. The quadrupolar frequency was set to 100 kHz and the Larmor frequency to 100 MHz. The input spin was 131Xe. 1000 theta increments and 2000 phi increments were used. The frequencies (x-axis) are expressed in units of** $\frac{1}{9}\left[I(I+1)-\frac{3}{4}\right]\frac{\nu_Q^2}{\Omega_o}$ **.[**

1. "Quadrupolar Nuclei in Solids", Alexander J. Vega, Encyclopedia of Nuclear Magnetic Resonance, Editors-in-Chief D.M.Grant and R.K. Harris, Vol. 6, Ped-Rel, pgs 3869-3889.

4.21 References

[5] D.M.Grant and R.K. Harris, Eds. in Chief, (1996), *Encyclopedia of Nuclear Magnetic Resonance*, John Wiley & Sons, New York.

[6] Brink, D.M. and Satchler, G.R. (1962), Angular Momentum, Clarendon Press, Oxford.

3.22 Programs and Input Files

IntQu_LC0.cc

```
/* IntQu_LC0.cc
*************************************************************-*-c++-*-
**
**
**                    Test Program for the GAMMA Library
**
**              Quadrupolar Interaction Literature Comparison 0
**
**
**
**  This program checks the quadrupolar interaction class IntQuad in
**
**  GAMMA.  In particular it looks to see how well the class parallels
**
**  the articles by Pascal P. Man
**
**
**
**  "Quadrupolar Interactions", Encyclopedia of Magnetic Resonance,
**
**  by Grant and Harris, Vol 6, Ped-Rel, page 3838-3869.
**
**
**
**  and Alexander Vega
**
**
**
**  "Quadrupolar Nuclei in Solids", Encyclopedia of Magnetic Resonance,
**
**  by Grant and Harris, Vol 6, Ped-Rel, page 3869-3889.
**
**
**
**  In particular, their PAS quadrupolar Hamiltonians are generated and
**
**  compared with the quadrupolar interaction class Hamiltonians.
**
**
**
**  Man's Hamiltonians will be generated from equations in (5) on page
**
**  3839 of the his article.  Vega's Hamiltonians will be made from
**
**  equations (28), (32) and (33) of his article.  Note that his (32)
**
**  is missing a factor of 1/3 on the <1|H|3> and <3|H|1> components.
**
**
**
** Author:     S.A. Smith
**
** Date:       10/11/96
**
** Update:     10/11/96
**
** Version:    3.6
**
** Copyright:  S. Smith.  You can modify this program for personal use, but
**
**                  you must leave it intact if you re-distribute it.
**
**
**
*********************************************************************
*******************/

#include <gamma.h>                          // Include GAMMA

main (int argc, char* argv[])
{
//                    Set Up The Quadrupolar Interaction

 int qn=1;
 double I;
 query_parameter(argc, argv, qn++,                  // Read in the coupling
                 "\n\tSpin Quantum Number? ", I);
 double W;                                          // Quadrupolar frequency
 query_parameter(argc, argv, qn++,                  // Read in the coupling
  "\n\tQuadrupolar Frequency(kHz)? ", W);
 W *= 1.e3;                                         // Put this in Hz
 double Qeta;
 query_parameter(argc, argv, qn++,                  // Read in the coupling
  "\n\tQuadrupolar Asymmetry [0, 1]? ", Qeta);
//                  Construct GAMMA Quadrupolar Interaction

 IntQuad Q(I,wQ2QCC(W,I),Qeta,0.0,0.0);
//              Here are the Operators To Build Man's Hamiltonians

 int Ival = int(2.*I + 1);          // For 1 spin SOp functions
 matrix IE = Ie(Ival);              // The operator 1
 matrix IM = Im(Ival);              // The operator I-
 matrix IP = Ip(Ival);              // The operator I+
 matrix IZ = Iz(Ival);              // The operator Iz
 matrix IX = Ix(Ival);              // The operator Ix
 matrix IY = Iy(Ival);              // The operator Iy
//                Here's The H Accoring To Man's Equation (5a)

//                 (Note That His W is Half Of Our Definition)
```

```
 matrix HMa = 3.0*IZ*IZ - (I*(I+1))*IE + Qeta*((IX*IX)-(IY*IY));
 HMa *= (W/6.0);
//                    Here's The H Accoring To Man'c Equation (5c)
//                     (Note That His W is Half Of Our Definition)
 matrix HMb = 3.0*IZ*IZ - (I*(I+1))*IE + (Qeta/2.)*((IP*IP)+(IM*IM));
 HMb *= (W/6.0);
//                         Here's The H According To GAMMA
 matrix HG = Q.H();
//                      Here's The H Also According To GAMMA
 matrix HGB = Q.H(0.0, 0.0);
//                Here Are Vegas V's According To Equations (22-27, 31)
//             (Switches eta Sign To Account For Opposite PAS Definition)
 double Eta = -Q.eta();
 double Vxx = 0.5*(- 1. - Eta);
 double Vyy = 0.5*(-1. + Eta);
 double Vzz = 1.0;
 double Vxy = 0.0;
 double Vxz = 0.0;
 double Vyz = 0.0;
 complex V1(-Vxz, -Vyz);
 complex Vm1(Vxz, -Vyz);
 complex V2(0.5*(Vxx-Vyy), Vxy);
 complex Vm2(0.5*(Vxx-Vyy), -Vxy);
//       Generate H According To Vega's Equations (32) Or (33)
 matrix HVega;
 if(I == 1)
   {
   HVega = matrix(3,3);
   HVega.put(Vzz/6.0, 0, 0);
   HVega.put(Vm1/sqrt(2.0), 0, 1);
   HVega.put(Vm2/3., 0, 2);                    // Added 1/3 Factor!
   HVega.put(-V1/sqrt(2.0), 1, 0);
   HVega.put(-Vzz/3.0, 1, 1);
   HVega.put(-Vm1/sqrt(2.0), 1, 2);
   HVega.put(V2/3., 2, 0);                     // Added 1/3 Factor!
   HVega.put(V1/sqrt(2.0), 2, 1);
   HVega.put(Vzz/6.0, 2, 2);
   HVega *= Q.wQ();
   }
 else if(I == 1.5)
   {
   HVega = matrix(4,4);
   HVega.put(Vzz/2.0, 0, 0);
   HVega.put(Vm1/sqrt(3.0), 0, 1);
   HVega.put(Vm2/sqrt(3.0), 0, 2);
   HVega.put(0.0, 0, 3);
   HVega.put(-V1/sqrt(3.0), 1, 0);
   HVega.put(-Vzz/2.0, 1, 1);
   HVega.put(0.0, 1, 2);
   HVega.put(Vm2/sqrt(3.0), 1, 3);
   HVega.put(V2/sqrt(3.0), 2, 0);
   HVega.put(0.0, 2, 1);
   HVega.put(-Vzz/2.0, 2, 2);
   HVega.put(-Vm1/sqrt(3.0), 2, 3);
   HVega.put(0.0, 3, 0);
   HVega.put(V2/sqrt(3.0), 3, 1);
   HVega.put(V1/sqrt(3.0), 3, 2);
   HVega.put(Vzz/2.0, 3, 3);
   HVega *= Q.wQ();
   }
//                    Generate H According To Vega's Equation (28)
 matrix HV = Vzz*(3.*IZ*IZ-(I*(I+1.))*IE);
 HV += (Vxx-Vyy)*(IX*IX-IY*IY);
 HV += 2*Vxy*(IX*IY-IY*IX);
 HV += 2*Vxz*(IX*IZ-IZ*IX);
 HV += 2*Vyz*(IY*IZ-IZ*IY);
 HV *= Q.wQ()/6.0;
//                      Output the Results for Visual Comparison
 cout << "\n\t\t\t\tGAMMA's Quadrupolar H:\t" << HG;
 cout << "\n\t\t\t\tGAMMA's Other Quadrupolar H:\t" << HGB;
 cout << "\n\t\t\t\tMan's Quadrupolar H(a):\n\t" << HMa;
 cout << "\n\t\t\t\tMan's Quadrupolar H(b):\n\t" << HMb;
 if(I == 1.0 || I == 1.5)
   cout << "\n\t\t\t\tVega's Quadrupolar H:\n\t" << HVega;
 cout << "\n\t\t\t\tVega's Generic Quad H:\n\t" << HV;
 }
```

IntQu_LC1.cc

```
/* IntQu_LC1.cc
*****************************************************************-*-c++-*-
**
**
**                    Test Program for the GAMMA Library
**
**              Quadrupolar Interaction Literature Comparison 1
**
**
**
**  This program checks the quadrupolar interaction class IntQuad in
**
**  GAMMA.  In particular it looks to see how well the class parallels
**
**  the article by Alexander Vega -
**
**
**
```

```
**  "Quadrupolar Nuclei in Solids", Encyclopedia of Magnetic Resonance,
**
**  by Grant and Harris, Vol 6, Ped-Rel, page 3869-3889.
**
**
**
**  Specifcally, herein we generate the spatial tensor components of
**
**  an oriented Quadrupolar interaction and and compare the results to
**
**  A.Vega's equations (22-27) and 31 on pages 3884-3885.
**
**
**
** Author:     S.A. Smith
**
** Date:       10/11/96
**
** Update:     10/11/96
**
** Version:    3.6
**
** Copyright:  S. Smith.  You can modify this program as you see fit
**
**            for personal use, but you must leave the program intact
**
**            if you re-distribute it.
**
**
**
*************************************************************************
*******************/

#include <gamma.h>                                  // Include GAMMA

main (int argc, char* argv[])

  {
//                          Construct A Quadrupolar Interaction

  int qn=1;
  double W;                                         // Quadrupolar frequency
  query_parameter(argc, argv, qn++,                 // Read in the coupling
          "\n\tQuadrupolar Frequency(kHz)? ", W);
  W *= 1.e3;                                        // Put this in Hz
  double Qeta;
  query_parameter(argc, argv, qn++,                 // Read in the coupling
   "\n\tQuadrupolar Asymmetry [0, 1]? ", Qeta);
  double Qtheta, Qphi;
  query_parameter(argc, argv, qn++,                 // Read in the angle
   "\n\tAngle down from z [0, 180]? ", Qtheta);
  query_parameter(argc, argv, qn++,                 // Read in the angle
    "\n\tAngle over from x [0, 360]? ", Qphi);
  double I=1.0;                                     // Use I=1, but this doesn't
  double QCC = wQ2QCC(W, I);                        // Heres quad. coupling
  IntQuad Q(I,QCC,Qeta,Qtheta,Qphi);                // matter for spatial parts
//                  Here Are Vegas V's According To Equations (22-27, 31)
//        Note We Change Sign On ETA As He Using A Different PAS Definition
  double Theta = Q.theta()*DEG2RAD;
  double Phi = Q.phi()*DEG2RAD;
  double Eta = -Q.eta();
  double Stheta = sin(Theta);
  double Ctheta = cos(Theta);
  double C2phi = cos(2.*Phi);
  double S2phi = sin(2.*Phi);
  double Vxx = 0.5*(3.*Stheta*Stheta - 1. - Eta*Ctheta*Ctheta*C2phi);
  double Vxy = 0.5*Eta*Ctheta*S2phi;
  double Vxz = -0.5*(Stheta*Ctheta*(3.0 + Eta*C2phi));
  double Vyx = Vxy;
  double Vyy = 0.5*(-1. + Eta*C2phi);
  double Vyz = 0.5*Eta*Stheta*S2phi;
  double Vzx = Vxz;
  double Vzy = Vyz;
  double Vzz = 0.5*(3.*Ctheta*Ctheta - 1. - Eta*Stheta*Stheta*C2phi);
  complex V0(sqrt(1.5)*Vzz);
  complex V1(-Vxz, -Vyz);
  complex Vm1(Vxz, -Vyz);
  complex V2(0.5*(Vxx-Vyy), Vxy);
  complex Vm2(0.5*(Vxx-Vyy), -Vxy);
//          Here Are The A's According To GAMMA Quadrupolar Interaction
//          Need To Scale Our A's By (1/2)/sqrt[5/(24*PI)] To Get Vega's V's
  double X = 0.5/RT5O24PI;
  double Thetad = Q.theta();
  double Phid = Q.phi();
  double AGxx = X*Q.Axx(Thetad, Phid);
  double AGxy = X*Q.Axy(Thetad, Phid);
  double AGxz = X*Q.Axz(Thetad, Phid);
  double AGyy = X*Q.Ayy(Thetad, Phid);
  double AGyx = X*Q.Ayx(Thetad, Phid);
  double AGyz = X*Q.Ayz(Thetad, Phid);
  double AGzz = X*Q.Azz(Thetad, Phid);
  double AGzx = X*Q.Azx(Thetad, Phid);
  double AGzy = X*Q.Azy(Thetad, Phid);
//          Here Are The A's According To GAMMA Quadrupolar Interaction
//          Need To Scale Our A's By (1/2)/sqrt[5/(24*PI)] To Get Vega's V's
  double AG1xx = X*Q.Axx();
  double AG1xy = X*Q.Axy();
  double AG1xz = X*Q.Axz();
  double AG1yy = X*Q.Ayy();
  double AG1yx = X*Q.Ayx();
```

```
 double AG1yz = X*Q.Ayz();
 double AG1zz = X*Q.Azz();
 double AG1zx = X*Q.Azx();
 double AG1zy = X*Q.Azy();
//          Here Are The A's According To GAMMA Quadrupolar Interaction
//          (Note That space_T Uses Azz>=Ayy>=Axx So ETA Opposite Vega's)
 space_T Agen = A2(0.0, 1.0, Qeta);
 Agen = Agen.rotate(Phid, Thetad, 0.0);
 Cartesian(Agen);
//                    Output Everyone For A Visual Comparison
 cout << "\n   " << "      Vega" << "          IntQuadA"
                << "     IntQuadB" << "     space_T";
 cout << "\nVxx    " << form("%8.3f", Vxx) << "        " << form("%8.3f", AGxx)
      << "      " << form("%8.3f", AG1xx) << "      " << form("%8.3f", Agen.Ccomponent(0,0));
 cout << "\nVxy    " << form("%8.3f", Vxy) << "        " << form("%8.3f", AGxy)
      << "      " << form("%8.3f", AG1xy)  << "      " << form("%8.3f", Agen.Ccomponent(0,1));
 cout << "\nVxz    " << form("%8.3f", Vxz) << "        " << form("%8.3f", AGxz)
      << "      " << form("%8.3f", AG1xz) << "      " << form("%8.3f", Agen.Ccomponent(0,2));
 cout << "\nVyy    " << form("%8.3f", Vyy) << "        " << form("%8.3f", AGyy)
      << "      " << form("%8.3f", AG1yy) << "      " << form("%8.3f", Agen.Ccomponent(1,1));
 cout << "\nVyx    " << form("%8.3f", Vyx) << "        " << form("%8.3f", AGyx)
      << "      " << form("%8.3f", AG1yx) << "      " << form("%8.3f", Agen.Ccomponent(1,0));
 cout << "\nVyz    " << form("%8.3f", Vyz) << "        " << form("%8.3f", AGyz)
      << "      " << form("%8.3f", AG1yz) << "      " << form("%8.3f", Agen.Ccomponent(1,2));
 cout << "\nVzz    " << form("%8.3f", Vzz) << "        " << form("%8.3f", AGzz)
      << "      " << form("%8.3f", AG1zz) << "      " << form("%8.3f", Agen.Ccomponent(2,2));
 cout << "\nVzx    " << form("%8.3f", Vzx) << "        " << form("%8.3f", AGzx)
      << "      " << form("%8.3f", AG1zx) << "      " << form("%8.3f", Agen.Ccomponent(2,0));
 cout << "\nVzy    " << form("%8.3f", Vzy) << "        " << form("%8.3f", AGzy)
      << "      " << form("%8.3f", AG1zy) << "      " << form("%8.3f", Agen.Ccomponent(2,1));
 cout << "\nV0 " << V0 << "  " << X*Q.A0(Thetad, Phid)
      << "  " << X*Q.A0() << "  " << Agen.component(2,0);
 cout << "\nV1 " << V1 << "  " << X*Q.A1(Thetad, Phid)
      << "  " << X*Q.A1() << "  " << Agen.component(2,1);
 cout << "\nV-1" << Vm1 << "  " << X*Q.Am1(Thetad, Phid)
      << "  " << X*Q.Am1() << "  " << Agen.component(2,-1);
 cout << "\nV2 " << V2 << "  " << X*Q.A2(Thetad, Phid)
      << "  " << X*Q.A2() << "  " << Agen.component(2,2);
 cout << "\nV-2" << Vm2 << "  " << X*Q.Am2(Thetad, Phid)
      << "  " << X*Q.Am2() << "  " << Agen.component(2,-2);
 cout << "\n\n\n";
 }
```

IntQu_LC2.cc

```
/* IntQu_LC2.cc
*****************************************************************-*-c++-*-
/**
**
**              Test Program for the GAMMA Library
**
**        Quadrupolar Interaction Literature Comparison 2
**
**
**
**  This program checks the quadrupolar interaction class IntQuad in
**
**  GAMMA.  In particular it looks to see how well the class parallels
**
**  some articles in the literature.
**
**
**
**  1. By Alexander Vega -
**
**  "Quadrupolar Nuclei in Solids", Encyclopedia of Magnetic Resonance,
**
**  by Grant and Harris, Vol 6, Ped-Rel, page 3869-3889, Eq. (30).
**
**
**
**  2. By Pascal P. Man -
**
**  "Quadrupolar Interactions", Encyclopedia of Magnetic Resonance,
**
**  by Grant and Harris, Vol 6, Ped-Rel, page 3839, Eq. (10).
**
**
**
**
**
** Author:     S.A. Smith
**
** Date:       10/11/96
**
** Update:     10/11/96
**
** Version:    3.6
**
** Copyright:   S. Smith.  You can modify this program as you see fit
**
**            for personal use, but you must leave the program intact
**
```

```
**          if you re-distribute it.
**
**
**
*************************************************************************
********************/

#include <gamma.h>                          // Include GAMMA
#
main (int argc, char* argv[])

  {
//                    Construct A Quadrupolar Interaction

  int qn=1;
  double I;                                 // Quadrupolar frequency
  query_parameter(argc, argv, qn++,         // Read in the coupling
          "\n\tSpin Quantum Number (1, 1.5, ...)? ", I);
  IntQuad Q(I,100.0);                       // Make an interaction
//                  Here Are Vegas T's Accoring To Equation (30)
//                      (His T20 Component, Qo, Is Wrong!)
int Ival = int(2.*I + 1);                   // For 1 spin SOp functions
  matrix IE = Ie(Ival);                     // The operator 1
  matrix IM = Im(Ival);                     // The operator I-
  matrix IP = Ip(Ival);                     // The operator I+
  matrix IZ = Iz(Ival);                     // The operator Iz
  matrix TVsph[5];
  TVsph[0]= (3.*IZ*IZ-(I*(I+1))*IE)*2./sqrt(6.);   // T20  = [2IzIz -I(I+1)]*2/sqrt(6)
//  TVsph[0]= (IZ*IZ-(I*(I+1))*IE)/sqrt(6.);        // This is Vega's Incorrect One
  TVsph[1]= -(IP*IZ + IZ*IP);               // T2m1 = -(I+Iz + IzI+)
  TVsph[2]= IM*IZ + IZ*IM;                  // T2m1 =  (I-Iz + IzI-)
  TVsph[3]= IP*IP;                          // T22  =  (I+I+)
  TVsph[4]= IM*IM;                          // T2m2 =  (I-I-)
//                  Here Are Man's Accoring To Equation (9)
  matrix TMsph[5];
  TMsph[0]= (3.*IZ*IZ-(I*(I+1))*IE)*sqrt(6.)/3.;   // T20  = [3*IzIz -I(I+1)]*sqrt(6)/3
  TMsph[1]= -(IP*IZ + IZ*IP);               // T2m1 = -(I+Iz + IzI+)
  TMsph[2]= IM*IZ + IZ*IM;                  // T2m1 =  (I-Iz + IzI-)
  TMsph[3]= IP*IP;                          // T22  =  (I+I+)
  TMsph[4]= IM*IM;                          // T2m2 =  (I-I-)
//                  Here Are GAMMA's Generic Quad Tensors
  spin_sys sys(1);                          // A 1 spin system
  if(I == 1) sys.isotope(0, "2H");          // Set spin to 2H w/ I=1
  else if(I == 1.5) sys.isotope(0, "131Xe"); // Set spin to Xe w/ I=3/2
  else if(I == 2.5) sys.isotope(0, "17O");  // Set spin to 0 w/ I=5/2
  else if(I == 3.0) sys.isotope(0, "10B");  // Set spin to Xe w/ I=3
  else if(I == 3.5) sys.isotope(0, "45Sc");  // Set spin to Xe w/ I=3/2
  else if(I == 4.5) sys.isotope(0, "93Nb");  // Set spin to Xe w/ I=3/2
spin_T TQ = T_Q(sys,0);                     // Get Quadrupolar spin tensor

int mlabs[5] = { 0, 1, -1, 2, -2 };
for(int i=0; i<5; i++)
  {
  cout << "\n\n";
  cout << "\n\tT2" << mlabs[i] << " 1/2*Vega: " << 0.5*TVsph[i];
  cout << "\n\tT2" << mlabs[i] << " 1/2*Man:  " << 0.5*TMsph[i];
  cout << "\n\tT2" << mlabs[i] << " Generic:  " << TQ.component(2, mlabs[i]);
  cout << "\n\tT2" << mlabs[i] << " GAMMA:    " << Q.Tcomp(mlabs[i]);
  }
}
```

IntQu_LC3.cc

```
/* IntQu_LC3.cc
*****************************************************************-*-c++-*-
**
**
**                    Test Program for the GAMMA Library
**
**              Quadrupolar Interaction Literature Comparison 3
**
**
**
**  This program checks the quadrupolar interaction class IntQuad in
**
**  GAMMA.  In particular it looks to see how well the class parallels
**
**  the articles by Alexander Vega
**
**
**
**  "Quadrupolar Nuclei in Solids", Encyclopedia of Magnetic Resonance,
**
**  by Grant and Harris, Vol 6, Ped-Rel, page 3869-3889.
**
**
**
**  In particular, his oriented quadrupolar Hamiltonian is generated
**
**  and compared with the quadrupolar interaction class Hamiltonians.
**
**
**
**  Vega's Hamiltonians will be made from equations (28), (32), and
**
**  (33) of his article.  Note that his (32) is missing a factor of
**
**  1/3 in many of his Hamiltonian components.
**
**
**
** Author:     S.A. Smith
**
** Date:       10/11/96
**
** Update:     10/11/96
**
** Version:    3.6
**
** Copyright:  S. Smith.  You can modify this program as you see fit for
**
**                         personal use, leave intact if re-distributed.
**
**
**
*****************************************************************
*******************/

#include <gamma.h>                          // Include GAMMA

main (int argc, char* argv[])
 {
//                    Set Up The Quadrupolar Interaction

 int qn=1;
 double I;
 query_parameter(argc, argv, qn++,          // Read in the coupling
   "\n\tSpin Quantum Number? ", I);
 double W;                                  // Quadrupolar frequency
 query_parameter(argc, argv, qn++,          // Read in the coupling
          "\n\tQuadrupolar Frequency(kHz)? ", W);
 W *= 1.e3;                                 // Put this in Hz
 double QCC = wQ2QCC(W,I);                  // Quad coupling constant
 double Qeta;
 query_parameter(argc, argv, qn++,          // Read in the coupling
         "\n\tQuadrupolar Asymmetry [0, 1]? ", Qeta);
 double Qtheta, Qphi;
 query_parameter(argc, argv, qn++,      // Read in the angle
    "\n\tAngle down from z [0, 180]? ", Qtheta);
 query_parameter(argc, argv, qn++,      // Read in the angle
    "\n\tAngle over from x [0, 360]? ", Qphi);
//                  Construct GAMMA Quadrupolar Interaction
//                   Generate H Using Both IntQuad Functions

 IntQuad Q(I,QCC,Qeta,Qtheta,Qphi);
 matrix HG = Q.H();
 matrix HGB = Q.H(Qtheta, Qphi);
//              Here are the Operators To Build Vega's Hamiltonians

 int Ival = int(2.*I + 1);                  // For 1 spin SOp functions
 matrix IE = Ie(Ival);                      // The operator 1
 matrix IM = Im(Ival);                      // The operator I-
 matrix IP = Ip(Ival);                      // The operator I+
 matrix IZ = Iz(Ival);                      // The operator Iz
 matrix IX = Ix(Ival);                      // The operator Ix
 matrix IY = Iy(Ival);                      // The operator Iy
//            Here Are Vegas V's According To Equations (22-27, 31)
//      Note We Change Sign On ETA As He Using A Different PAS Definition

 double Theta = Q.theta()*DEG2RAD;
 double Phi = Q.phi()*DEG2RAD;
 double Eta = -Q.eta();
 double Stheta = sin(Theta);
 double Ctheta = cos(Theta);
 double C2phi = cos(2.*Phi);
 double S2phi = sin(2.*Phi);
```

```
double Vxx = 0.5*(3.*Stheta*Stheta - 1. - Eta*Ctheta*Ctheta*C2phi);
double Vxy = 0.5*Eta*Ctheta*S2phi;
double Vxz = -0.5*(Stheta*Ctheta*(3.0 + Eta*C2phi));
double Vyy = 0.5*(-1. + Eta*C2phi);
double Vyz = 0.5*Eta*Stheta*S2phi;
double Vzz = 0.5*(3.*Ctheta*Ctheta - 1. - Eta*Stheta*Stheta*C2phi);
complex V1(-Vxz, -Vyz);
complex Vm1(Vxz, -Vyz);
complex V2(0.5*(Vxx-Vyy), Vxy);
complex Vm2(0.5*(Vxx-Vyy), -Vxy);
//                    Generate H According To Vega's Equations (32) Or (33)
matrix HVega;
if(I == 1)
  {
  HVega = matrix(3,3);
  HVega.put(Vzz/6.0, 0, 0);
  HVega.put(Vm1/(3.0*sqrt(2.0)), 0, 1);          // Added 1/3 Factor!
  HVega.put(Vm2/3., 0, 2);                       // Added 1/3 Factor!
  HVega.put(-V1/(3.*sqrt(2.0)), 1, 0);           // Added 1/3 Factor!
  HVega.put(-Vzz/3.0, 1, 1);
  HVega.put(-Vm1/(3.*sqrt(2.0)), 1, 2);          // Added 1/3 Factor!
  HVega.put(V2/3., 2, 0);                        // Added 1/3 Factor!
  HVega.put(V1/(3*sqrt(2.0)), 2, 1);             // Added 1/3 Factor!
  HVega.put(Vzz/6.0, 2, 2);
  HVega *= Q.wQ();
  }
else if(I == 1.5)
  {
  HVega = matrix(4,4);
  HVega.put(Vzz/2.0, 0, 0);
  HVega.put(Vm1/sqrt(3.0), 0, 1);
  HVega.put(Vm2/sqrt(3.0), 0, 2);
  HVega.put(0.0, 0, 3);
  HVega.put(-V1/sqrt(3.0), 1, 0);
  HVega.put(-Vzz/2.0, 1, 1);
  HVega.put(0.0, 1, 2);
  HVega.put(Vm2/sqrt(3.0), 1, 3);
  HVega.put(V2/sqrt(3.0), 2, 0);
  HVega.put(0.0, 2, 1);
  HVega.put(-Vzz/2.0, 2, 2);
  HVega.put(-Vm1/sqrt(3.0), 2, 3);
  HVega.put(0.0, 3, 0);
  HVega.put(V2/sqrt(3.0), 3, 1);
  HVega.put(V1/sqrt(3.0), 3, 2);
  HVega.put(Vzz/2.0, 3, 3);
  HVega *= Q.wQ();
  }
//                    Generate H According To Vega's Equation (28)
matrix HV = Vzz*(3.*IZ*IZ-(I*(I+1.))*IE);
HV += (Vxx-Vyy)*(IX*IX-IY*IY);
HV += 2*Vxy*(IX*IY+IY*IX);
HV += 2*Vxz*(IX*IZ+IZ*IX);
HV += 2*Vyz*(IY*IZ+IZ*IY);
HV *= Q.wQ()/6.0;
//                    Generate H Using GAMMA's Generic Tensors
//          The A's Scaling Will Here Be Equal Vegas But the T' Are 1/2 Of His
//       Note Also That The Spatial Tensor Rotation Direction is Defined Reversed
space_T Agen = A2(0.0, 1.0, Qeta);                 // Rank 2 space tensor
Agen = Agen.rotate(Q.phi(), Q.theta(), 0.0);       // Rotate to theta,phi
spin_sys sys(1);                                   // A 1 spin system
if(I == 1) sys.isotope(0, "2H");                   // Set spin to 2H w/ I=1
else if(I == 1.5) sys.isotope(0, "131Xe");         // Set spin to Xe w/ I=3/2
else if(I == 2.5) sys.isotope(0, "17O");           // Set spin to 0 w/ I=5/2
else if(I == 3.0) sys.isotope(0, "10B");           // Set spin to Xe w/ I=3
else if(I == 3.5) sys.isotope(0, "45Sc");          // Set spin to Xe w/ I=3/2
else if(I == 4.5) sys.isotope(0, "93Nb");          // Set spin to Xe w/ I=3/2
spin_T TQ = T_Q(sys,0);                            // Generic Quad spin tensor
gen_op HTen = T_prod(TQ, Agen, 2);
HTen *= Q.wQ()/3.0;
//                    Output the Results for Visual Comparison
cout << "\n\tGAMMA's Quadrupolar H:\n\t" << HG;
cout << "\n\tGAMMA's Other Quadrupolar H:\n\t" << HGB;
if(I == 1.0 || I == 1.5)
  cout << "\n\tVega's Quadrupolar H:\n\t" << HVega;
cout << "\n\tVega's Generic Quad H:\n\t" << HV;
cout << "\n\tGAMMA's Generic Tensor H:\n\t" << HTen.get_mx();
//                    Do As A True/False Test Now
int TF = 1;
if((HG-HGB).is_zero())
  cout << "\n\n\tGAMMA IntQuad Method 1 = IntQuad Method 2:    PASS";
else
  {
  cout << "\n\n\tGAMMA IntQuad Method 1 = IntQuad Method 2:    FAIL!!!!";
  TF *= 0;
  }
if(I==1.0)
  if((HG-HVega).is_zero())
    cout << "\n\tGAMMA IntQuad Method 1 = Vega Equation 32:    PASS";
  else
    {
    cout << "\n\tGAMMA IntQuad Method 1 = Vega Equation 32:    FAIL!!!";
    TF *= 0;
    }
else if(I==1.5)
  if((HG-HVega).is_zero())
    cout << "\n\tGAMMA IntQuad Method 1 = Vega Equation 33:    PASS";
```

```
  else
    {
    cout << "\n\tGAMMA IntQuad Method 1 = Vega Equation 33:     FAIL!!!";
    TF *= 0;
    }
if((HG-HV).is_zero())
  cout << "\n\tGAMMA IntQuad Method 1 = Vega Equation 28:     PASS";
else
  {
  cout << "\n\tGAMMA IntQuad Method 1 = Vega Equation 28:     FAIL!!!";
  TF *= 0;
  }
if((HG-HTen.get_mx()).is_zero())
  cout << "\n\tGAMMA IntQuad Method 1 = GAMMA Generic Tensor: PASS";
else
  {
  cout << "\n\tGAMMA IntQuad Method 1 = GAMMA Generic Tensor: FAIL!!!";
  TF *= 0;
  }
if(TF) cout << "\n\n\t\t\tTEST PASS";
else   cout << "\n\n\t\t\tTEST FAIL!!!";
cout << "\n\n";
}
```

IntQu_LC4.cc

```
//* IntQu_LC4.cc
*************************************************************-*-c++-*-
**
**
**                    Test Program for the GAMMA Library
**
**                Quadrupolar Interaction Literature Comparison 4
**
**
**
**  This program checks the quadrupolar interaction GAMMA class IntQuad .
**
** This see's how well the class parallels the article by Pascal P. Man
**
**
**
**  "Quadrupolar Interactions", Encyclopedia of Magnetic Resonance,
**
**  by Grant and Harris, Vol 6, Ped-Rel, page 3838-3869.
**
**
**
**  In particular, his first order quadrupolar Hamiltonian is generated
**
**  and compared with the quadrupolar interaction class Hamiltonian.
**
**
**
**  Man's Hamiltonian will be made from his equation (18) coupled with
**
**  the rotated spatial tensor component of his equation (28).
**
**
**
** Author:     S.A. Smith
**
** Date:       10/14/96
**
** Update:     10/14/96
**
** Version:    3.6
**
** Copyright:  S. Smith.  You can modify this program as you see fit
**
**          for personal use, but you must leave the program intact
**
**          if you re-distribute it.
**
**
**
*********************************************************************
*******************/
```

```
#include <gamma.h>                                  // Include GAMMA
main (int argc, char* argv[])

 {
//                          Set Up The Quadrupolar Interaction

 int qn=1;
 double I;
 query_parameter(argc, argv, qn++,                  // Read in the coupling
   "\n\tSpin Quantum Number? ", I);
 double W;                                          // Quadrupolar frequency
 query_parameter(argc, argv, qn++,                  // Read in the coupling
   "\n\tQuadrupolar Frequency(kHz)? ", W);
 W *= 1.e3;                                         // Put this in Hz
 double QCC = wQ2QCC(W,I);                          // Quad coupling constant
 double Qeta;
 query_parameter(argc, argv, qn++,                  // Read in the coupling
   "\n\tQuadrupolar Asymmetry [0, 1]? ", Qeta);
 double Qtheta, Qphi;
 query_parameter(argc, argv, qn++,                  // Read in the angle
    "\n\tAngle down from z [0, 180]? ", Qtheta);
 query_parameter(argc, argv, qn++,                  // Read in the angle
    "\n\tAngle over from x [0, 360]? ", Qphi);

// ---------------------------------------------------------------------
//                      Construct GAMMA Quadrupolar Interaction
//                       Generate H Using Both IntQuad Functions

 IntQuad Q(I,QCC,Qeta,Qtheta,Qphi);
 matrix HGA = Q.H0();
 matrix HGB = Q.H0(Qtheta, Qphi);

// ---------------------------------------------------------------------
//              Here are the Spin Ops For The Build Of Man's Hamiltonian

 int Ival = int(2.*I + 1);                          // For 1 spin SOp functions
 matrix IE = Ie(Ival);                              // The operator 1
 matrix IZ = Iz(Ival);                              // The operator Iz
//     Here is The Spatial Tensor Part For Man's Hamiltonian
//   (The eq Factor Has Been Taken Out & Placed in Pre-Factor)

 double beta = Qtheta*DEG2RAD;
 double alpha = Qphi*DEG2RAD;
 double Vo = sqrt(1.5)*(0.5*(3.*cos(beta)*cos(beta)-1.)
        + 0.5*Qeta*sin(beta)*sin(beta)*cos(2.*alpha));
//                      Generate H0 According To Man's Equation (18)
//                            2
//          [1]      (0)     eq Q      sqrt(6.0)        2
//        H     = H   =  -------- * --------- [3*I - I(I+1)]*V
```

```
//          Q       Q    4I(2I-1)          3            z                0

  matrix HMan = (QCC/(4.*I*(2*I-1)))*(sqrt(6.0)/3.0)
            * (3.0*IZ*IZ - (I*(I+1))*IE)*Vo;

// ---------------------------------------------------------------------------
//                    Generate H0 According To A Simplified Direct Equation
//             [1]      (0)     wQ                2
//           H    = H   =  --------- * [3*I - I(I+1)] *V
//              Q      Q    3*sqrt(6)         z                0

  matrix Hx = (W/(3.*sqrt(6.0)))*(3.0*IZ*IZ - (I*(I+1))*IE)*Vo;

// ---------------------------------------------------------------------------
//                              Output the Results for Visual Comparison
  cout << "\n\tGAMMA's IntQuad First Order Quadrupolar H A:\n\t" << HGA;
  cout << "\n\tGAMMA's IntQuad First Order Quadrupolar H B:\n\t" << HGB;
  cout << "\n\tMan's First Order Quadrupolar H:\n\t" << HMan;
  cout << "\n\tSimple Eq. First Order Quadrupolar H:\n\t" << Hx;
  cout << "\n\nFrom HQ0 Routine:\n" << HQ0(I, W, Qeta, Qtheta, Qphi);
  }
```

IntQu_LC5.cc

```
/* IntQu_LC5.cc
*************************************************************-*-c++-*-
**
**
**                    Test Program for the GAMMA Library
**
**                 Quadrupolar Interaction Literature Comparison 5
**
**
**
**  This program checks the quadrupolar interaction class IntQuad in
**
**  GAMMA.  In particular it looks to see how well the class parallels
**
**  the articles by Pascal P. Man
**
**
**
**  "Quadrupolar Interactions", Encyclopedia of Magnetic Resonance,
**
**  by Grant and Harris, Vol 6, Ped-Rel, page 3838-3869.
**
**
**
**  In particular, his 2nd order quadrupolar Hamiltonian is generated
**
**  and compared with the quadrupolar interaction class Hamiltonian.
**
**
**
**  Man's Hamiltonian will be made from his equation (19) coupled with
**
**  the rotated spatial tensor component of his equation (32).
**
**
**
** Author:     S.A. Smith
**
** Date:       10/14/96
**
** Update:     10/14/96
**
** Version:    3.6
**
** Copyright:  S. Smith.  You can modify this program as you see fit
**
**           for personal use, but you must leave the program intact
**
**           if you re-distribute it.
**
**
**
*************************************************************************
*******************/

#include <gamma.h>                                  // Include GAMMA
main (int argc, char* argv[])

 {
//                          Set Up The Quadrupolar Interaction

 int qn=1;
 double I;
 query_parameter(argc, argv, qn++,                  // Read in the coupling
   "\n\tSpin Quantum Number? ", I);
 double W;                                          // Quadrupolar frequency
 query_parameter(argc, argv, qn++,                  // Read in the coupling
   "\n\tQuadrupolar Frequency(kHz)? ", W);
 W *= 1.e3;                                         // Put this in Hz
 double QCC = wQ2QCC(W,I);                          // Quad coupling constant
 double Qeta;
 query_parameter(argc, argv, qn++,                  // Read in the coupling
   "\n\tQuadrupolar Asymmetry [0, 1]? ", Qeta);
 double Qtheta, Qphi;
 query_parameter(argc, argv, qn++,                   // Read in the angle
    "\n\tAngle down from z [0, 180]? ", Qtheta);
 query_parameter(argc, argv, qn++,                  // Read in the angle
    "\n\tAngle over from x [0, 360]? ", Qphi);
 double Om;
 query_parameter(argc, argv, qn++,                  // Read in the field
        "\n\tField Strength (MHz)? ", Om);
 spin_system sys(1);                                // A single spin system
 if(I == 1) sys.isotope(0, "2H");                   // Set spin to 2H w/ I=1
 else if(I == 1.5) sys.isotope(0, "131Xe");         // Set spin to Xe w/ I=3/2
 else if(I == 2.5) sys.isotope(0, "17O");           // Set spin to 0 w/ I=5/2
 else if(I == 3.0) sys.isotope(0, "10B");           // Set spin to Xe w/ I=3
 else if(I == 3.5) sys.isotope(0, "45Sc");          // Set spin to Xe w/ I=3/2
 else if(I == 4.5) sys.isotope(0, "93Nb");          // Set spin to Xe w/ I=3/2
 sys.Omega(Om);                                     // Set the field strength
 double OMHz = sys.Omega(0)*1.e6;                   // Larmor in MHz

// ---------------------------------------------------------------------------
//                          Construct GAMMA Quadrupolar Interaction
//                           Generate H Using Both IntQuad Functions

 IntQuad Q(I,QCC,Qeta,Qtheta,Qphi);
 matrix HGA = Q.H1(OMHz);
 matrix HGB = Q.H1(OMHz, Qtheta, Qphi);

// ---------------------------------------------------------------------------
//                 Here are the Spin Ops For The Build Of Man's Hamiltonian

 int Ival = int(2.*I + 1);                          // For 1 spin SOp functions
```

```
  matrix IE = Ie(Ival);                                   // The operator 1
  matrix IZ = Iz(Ival);                                   // The operator Iz                    }
  matrix T11 = IZ*((4.0*I*(I+1))*IE - 8.*IZ*IZ - IE);
  matrix T22 = IZ*((2.0*I*(I+1))*IE - 2.*IZ*IZ - IE);
//                    Here is The Spatial Tensor Part For Man's Hamiltonian
//                 (The eq Factor Has Been Taken Out & Placed in Pre-Factor)
  double beta = Qtheta*DEG2RAD;
  double alpha = Qphi*DEG2RAD;
  double cos2alpha = cos(2.*alpha);
  double cosbeta = cos(beta);
  double etasq = Qeta*Qeta;
  double C2alphasq = cos2alpha*cos2alpha;
  double Cbetasq = cosbeta*cosbeta;

  double V1Vm11 = ((-1.0/3.0)*etasq*C2alphasq
            + 2.0*Qeta*cos2alpha - 3.0)*Cbetasq*Cbetasq;
  double V1Vm12 = ((2.0/3.0)*etasq*C2alphasq
            - 2.0*Qeta*cos2alpha - etasq/3.0 + 3.0)*Cbetasq;
  double V1Vm13 = (etasq/3.0)*(1.0-C2alphasq);
  double twoV1Vm1 = -1.5*(V1Vm11 + V1Vm12 + V1Vm13);

  double V2Vm21 = ((1.0/24.0)*etasq*C2alphasq
            - 0.25*Qeta*cos2alpha + 3.0/8.0)*Cbetasq*Cbetasq;
  double V2Vm22 = ((-1.0/12.0)*etasq*C2alphasq
            + etasq/6.0 - 0.75)*Cbetasq;
  double V2Vm23 = (etasq/24.0)*C2alphasq
            + 0.25*Qeta*cos2alpha + 3.0/8.0;
  double twoV2Vm2 = 3.0*(V2Vm21 + V2Vm22 + V2Vm23);
//            Generate H1 According To Man's Equation (19)
//                                2           2
//   [2]     (1)   -1      [   e qQ     ]     [                                  2
// H   = H    = -- * | -------- |  * | (2*V  *V )*[4I(I+1)- 8I - 1]
//    Q     Q     Om   [ 4I(2I-1)  ]     [          -1    1                     z
//
//                                                                  2       ]
//                                + (2*V  *V )*[2I(I+1) - 2I  - 1] |
//                                         -2  2                    z       ]

  matrix HMan =  twoV1Vm1*T11 + twoV2Vm2*T22;
  HMan *= (-W*W/(36.0*OMHz));

// ---------------------------------------------------------------------------
//                              Output the Results for Visual Comparison
  cout << "\n\tGAMMA's IntQuad First Order Quadrupolar H A:\n\t" << HGA;
  cout << "\n\tGAMMA's IntQuad First Order Quadrupolar H B:\n\t" << HGB;
  cout << "\n\tMan's Second Order Quadrupolar H:\n\t" << HMan;
  cout << "\n\nFrom IntQuad HQ1 Routine:\n" << HQ1(OMHz,I,W,Qeta,Qtheta,Qphi);
  cout << "\n\nFrom H_Quad HQ1 Routine:\n" << HQ1x(sys,W,Qeta,Qtheta,Qphi,0);
```

IntQu_LC6.cc

```
/* IntQu_LC6.cc
*************************************************************-*-c++-*-
**
**
**                    Example Program for the GAMMA Library
**
**                 Quadrupolar Interaction Literature Comparison 6
**
**
**
**  This program checks the quadrupolar interaction class IntQuad in
**
**  GAMMA.  In particular it looks to see how well the class parallels
**
**  the articles by Alexander Vega
**
**
**
**  "Quadrupolar Nuclei in Solids", Encyclopedia of Magnetic Resonance,
**
**  by Grant and Harris, Vol 6, Ped-Rel, page 3869-3889.
**
**
**
**  In particular, this program looks at transitions associated with
**
**  the Quadrupolar Hamiltonian associated with a single spin (such as
**
**  2H, 131Xe, ...).  Note that it looks at the transitions predicted
**
**  by the 1st order quadrupolar interactions under a symmetric (eta=0)
**
**  tensor.  The results of this program should mimic the transitions
**
**  displayed on pages 3870-3871.  A plot is made of an NMR spectrum
**
**  following a perfect 90 pulse with the 1st order correction applied.
**
**  The plot is made interactively using gnuplot and should coincide
**
**  with Vega's Figure 2 (page 3871).
**
**
**
**  Note: The transitons are split by wQ, the input frequency here.
**
**
**
** Author:     S.A. Smith
**
** Date:       10/12/96
**
** Update:     10/12/96
**
** Version:    3.6
**
** Copyright:  S. Smith.  You can modify this program as you see fit
**
**          for personal use, but you must leave the program intact
**
**          if you re-distribute it.
**
**
**
*********************************************************************
*******************/

#include <gamma.h>                                  // Include GAMMA

main (int argc, char* argv[])

 {
 cout << "\n\t\tSymmetric Quadrupolar Interaction Transitions";
 cout << "\n\t\t (2H:1, 131Xe:3/2, 55Mn:5/2, 10B:3, 51V:7/2)\n";
 int qn=1;                                          // Query index
 String Iso;                                        // Isotope of spin
 spin_system sys(1);                                // A 1 spin system
 query_parameter(argc, argv, qn++,                  // Get an isotope type
                 "\n\tIsotope Type [2H, 131Xe, ....]? ", Iso);
 sys.isotope(0, Iso);                               // Set spin to Iso
 double wQ;                                         // Set Quad. frequency
 query_parameter(argc, argv, qn++,                  // Get the quadrupolar coupling
                 "\n\tQuadrupolar Frequency (kHz)? ", wQ);
 wQ *= 1.e3;                                        // Switch to Hz
 double DCC = wQ2QCC(wQ, sys.qn(0));                // Quad coupling constant
 IntQuad Q(sys.qn(0), DCC);                         // Make a quad interaction
//                    First We Look At Zero Field
//                 We Should Get 2*I Transitions Separated By wQ
//          The Hamiltonian (Not Printed) Will Have Vega's Figure 1 Levels
 gen_op H0(Q.H());                                  // 1st order Quad. Hamiltonian
 int npts = 2048;                                   // Block size
 gen_op D = Fm(sys);                                // Detector to F-
 gen_op sigma0 = sigma_eq(sys);                     // Dens. Op. at equilibrium
 gen_op sigma1 = Iypuls(sys, sigma0, 90.0);         // Apply first (PI/2) pulse
 acquire1D ACQ(D, H0);                              // Prepare for acquisitions
 matrix mx = ACQ.table(sigma1);                     // Table of transitions
 table(cout, mx);                                   // Output transitions table
//       Next We Take The Quad. As Weak Relative To The Zeeman Interaction
//                 We'll Use The First Order Perturbation Hamiltonian
 gen_op H1(Q.H0(O));                                // 1st order Quad. Hamiltonian
```

```
acquire1D ACQ1(D, H1);                    // Prepare for acquisitions
mx = ACQ1.table(sigma1);                  // Table of transitions
table(cout, mx);                          // Output transitions table
double lb = wQ/20.0;                      // Set a line broading factor
offset(mx, 0.0, lb, 1);                   // Apply the line broadening
double I = sys.qn(0);                     // This is the spin I value
double SW = wQ*(2.0*I+1);                 // This will be a plotting range
row_vector vx = ACQ1.F(mx, npts, -SW, SW); // Make a nice plot
GP_1D("spec.asc", vx, 0, -SW/wQ, SW/wQ);  // Output the points in ASCII
GP_1Dplot("spec.gnu", "spec.asc");        // Call Gnuplot and make plot now
}
```

IntQu_Pow3.cc

```
/* IntQu_Pow3.cc
*************************************************************-*-c++-*-
**
**
**            Example Program for the GAMMA Library
**
**
**
**  This program calculates a powder average for a single spin which
**
**  is associated with a quadrupolar interaction.  The high field
**
**  approximation is invoked in that the quadrupolar Hamiltonian is
**
**  treated as a perturbation to the Zeeman Hamiltonian and taken to
**
**  first order.  Only quadrupolar Hamiltonian terms which are
**
**  rotationally invariant about the field axis (z) are maintained.
**
**
**
**  This will perform the same as IntQu_Pow2.cc but uses the powder
**
**  average as suggested by Cheng et. al.
**
**
**
** "Investigations of a nonrandom numerical method for multidimensional
**
**  integration", Vera B. Cheng, Henry H. Suzukawa Jr. and
**
**  Max Wolfsberg, J.Chem.Phys 59 3992-9 (1973)
**
**
**
** Author:     S.A. Smith
**
** Date:       10/15/96
**
** Update:     10/15/96
**
** Version:    3.6
**
** Copyright:  S. Smith.  You can modify this program as you see fit
**
**            for personal use, but you must leave the program intact
**
**            if you re-distribute it.
**
**
**
*******************************************************************
*******************/

#include <gamma.h>                                  // Include GAMMA

main (int argc, char* argv[])

 {
 cout << "\n\t\t   High Field Quadrupolar Powder Pattern";
 cout << "\n\t\t(2H:1, 131Xe:3/2, 55Mn:5/2, 10B:3, 51V:7/2)\n";
//                    First Make A Quadrupolar Interaction
 String Iso;                                        // Isotope of spin
 int qn=1;                                          // Query index
 query_parameter(argc, argv, qn++,                  // Get the isotope type
      "\n\tIsotope Type [2H, 131Xe, ....]? ", Iso);
 double wQ;                                         // Set Quad. frequency
 query_parameter(argc, argv, qn++,                  // Get the quadrupolar coupling
     "\n\tQuadrupolar Frequency (kHz)? ", wQ);
 wQ *= 1.e3;                                        // Switch to Hz
 double eta;                                        // Set Quad. frequency
 query_parameter(argc, argv, qn++,                  // Get the quadrupolar coupling
     "\n\tQuadrupolar eta Value [0, 1]? ", eta);
//                    Make A Spin System For Convenience
 spin_system sys(1);                                // A 1 spin system
 sys.isotope(0, Iso);                               // Set spin to Iso
 double I = sys.qn(0);                              // This is the spin I value
 IntQuad Q(I,wQ2QCC(wQ, I), eta);                   // Set a Quad interaction
//                    Set Things Up For The Powder Average
 int npts = 4096;                                   // Block size
 gen_op D = Fm(sys);                                // Detector to F-
 gen_op sigma0 = sigma_eq(sys);                     // Dens. Op. at equilibrium
 gen_op sigma1 = Iypuls(sys, sigma0, 90.0);         // Apply first (PI/2) pulse
//        Set Up Constants for the Powder Average in Accordance with the Article
//       "Investigations of a nonrandom numerical method for multidimensional integration",
//          V.B. Cheng, H.H. Suzukawa Jr. & M. Wolfsberg, J.Chem.Phys 59 3992-9 (1973)
 int cheng1[] = {    2,    3,    5,     8,    13,
                 21,   34,   55,    89,   144,
                 233,  377,  616,   987,  1597,
                 2584, 4181, 6765, 10946, 17711};
 int cheng2[] = {    1,    1,    2,     3,     5,
                 8,   13,   21,    34,    55,
                 89,  144,  233,   377,   616,
                 987, 1597, 2584,  4181,  6765};

 int cheng;                                         // Powder averaging amount
 query_parameter(argc, argv, qn++,                  // Get the cheng steps
```

```
  "\n\tCheng Value (Sets # Powder Steps) [1, 20]? ", cheng);
 if(cheng<1 || cheng>20)                          // Insure reasonable average
  {
  cout << "\n\n\tNumber of Cheng Steps Must "
      << "Reside in Range [1, 20]!\n\n";
  exit(-1);
  }
//          Set Up A20(theta,phi) Values According To The Cheng Averaging
 int nsteps = cheng1[cheng];                      // Number of steps in average
 double A20s[nsteps];                             // For A20 storage
 double sfacts[nsteps];                           // For theta scaling factors
 double thetainc = 180.0/double(nsteps);          // Theta increment (degrees)
 double phiinc0 = 360.0/double(nsteps);           // Phi "increment" (degrees)
 double theta, phi;                               // Orientation angles (deg)
 int step;
 for(step=1; step<nsteps; step++)                 // Loop over cheng steps
  {
  theta = step*thetainc;                          // Compute the theta angle
  phi = phiinc0 * double((cheng2[cheng]*step)%nsteps);// Compute the phi angle
  A20s[step] = zRe(Q.A20(theta, phi));            // Store A20 this step
  sfacts[step] = sin(theta*DEG2RAD);              // Store sin(theta) this step
  }

//                    Perform Cheng Powder Averaging
//          Angle theta Is Down From The +z Axis, Angle phi Over From +x
 matrix spec0, spec, specsum;                     // Arrays for discrete spectra
 double lb = wQ/20.0;                             // Set a line broading factor
 double lbrad = lb*PI/180.0;                      // Set a line broading factor
 double SW = wQ*(2.0*I+1);                        // This will be a plotting range
 double Xi = Q.xi();                              // Interaction constant
 gen_op H = Xi*Q.Tcomp(0);                        // Base (unscaled) 1st order H
 acquire1D ACQ(D, H);                             // Prepare for acquisitions
 spec0 = ACQ.table(sigma1);                       // Table of base transitions
 for(step=1; step<nsteps; step++)                 // Loop over cheng steps
  {
  spec = spec0;                                   // Copy base spectrum
  scale(spec, A20s[step], sfacts[step]);          // Apply the scaling
  specsum = sum(specsum, spec, lbrad/5);          // Add to previous spectra
  }
//                    Output The Frequency Domain Spectrum
 cout << "\n\n\tDone With Powder Average. Line Broadening & Making Spectrum!";
 cout.flush();
 offset(specsum, 0.0, lb, 1);                     // Apply the line broadening
 row_vector data = ACQ.F(specsum,npts,-SW, SW);   // Spectrum of powder average
 GP_1D("spec.asc", data, 0, -2.0*I-1, 2.0*I+1);   // Output the points in ASCII
 GP_1Dplot("spec.gnu", "spec.asc");               // Call Gnuplot and make plot now
 FM_1D("spec.mif", data);                         // Output plot in FrameMaker too
 }
```

IntQu_PCT0.cc

```
/* IntQu_PCT0.cc ************************************************************-*-c++-*-
**                                                                            **
**                    Example Program for the GAMMA Library                   **
**                                                                            **
** This program calculates a powder average for a single spin which           **
** is associated with a quadrupolar interaction.  The high field              **
** approximation is invoked in that the quadrupolar Hamiltonian is            **
** treated as a perturbation to the Zeeman Hamiltonian and taken to           **
** second order.  Only quadrupolar Hamiltonian terms which are                **
** rotationally invariant about the field axis (z) are maintained.            **
** Furthermore, only the central transtion will be considered.                **
**                                                                            **
** This will program is similar to IntQu_Pow2.cc but restricts the            **
** computation to only the central transition. In turn, that means            **
** only spins with I=m*1/2 where m is odd and larger than 1 are valid.        **
** Analog formula will be used to construct the spectrum.                     **
**                                                                            **
** Later version of GAMMA will have the functions “scale” and “sum”           **
** in the library itself, so you will need to remove them from this           **
** program in that event.                                                     **
**                                                                            **
** Author:     S.A. Smith                                                     **
** Date:       10/15/96                                                       **
** Update:     10/15/96                                                       **
** Version:    3.6                                                            **
** Copyright:  S. Smith.  You can modify this program as you see fit          **
**             for personal use, but you must leave the program intact        **
**             if you re-distribute it.                                       **
**                                                                            **
*****************************************************************************/

#include <gamma.h>                          // Include GAMMA

void addW(row_vector& vx, double Fst, double Ffi, double F, double I)

    // Input    vx    : A row vector
    //          Fst   : Frequency of 1st point of vx (Hz)
    //          Ffi   : Frequency of last point of vx (Hz)
    //          F     : Transition frequency (Hz)
    //          I     : Transition intensity
    // Output   void  : The transition is added to the
    //                  row vector (as a Dirac delta).
    //Note            : To start one should zero vx

 {
 if(F<Fst || F>Ffi) return;                 // Insure its in range
 double Nm1 = double(vx.size()-1);          // Freq. -> point conversion
 double m = Nm1/(Ffi-Fst);                  // Slope Freq. -> pt
 double dpt = m*(F-Fst);                    // Point index of F
 int pt = int(dpt);                         // Main point for F
 double drem = dpt - pt;                    // Part which isn’t
 if(!drem) vx.put(vx.get(pt)+I, pt);        // Add if on a point
 else if(drem > 0)                          // If in between points
   {                                        // then just split it up
   vx.put(vx.get(pt)+(1.0-drem)*I, pt);     // between the two
   vx.put(vx.get(pt+1)+drem*I, pt+1);
   }
 else
   {
   vx.put(vx.get(pt)+(1.0+drem)*I, pt);
   vx.put(vx.get(pt-1)-drem*I, pt-1);
   }
 return;
 }
```

```
main (int argc, char* argv[])

 {
 cout << "\n\t\tQuadrupolar Central Transition Powder Pattern";
 cout << "\n\t\t    (131Xe:3/2, 55Mn:5/2, 51V:7/2, ...)\n";
//                    First Make A Quadrupolar Interaction
 String Iso;                                        // Isotope of spin
 int qn=1;                                          // Query index
 query_parameter(argc, argv, qn++,                  // Get the isotope type
  "\n\tIsotope Type [131Xe, 55Mn, 51V, ....]? ", Iso);
 double wQ;                                         // Set Quad. frequency
 query_parameter(argc, argv, qn++,                  // Get the quadrupolar coupling
     "\n\tQuadrupolar Frequency (kHz)? ", wQ);
 wQ *= 1.e3;                                        // Switch to Hz
 double eta;                                        // Set Quad. frequency
 query_parameter(argc, argv, qn++,                  // Get the quadrupolar coupling
     "\n\tQuadrupolar eta Value [0, 1]? ", eta);
 double Om;
 query_parameter(argc, argv, qn++,                  // Get the field strength
            "\n\tLarmor Frequency (MHz)? ", Om);
 Om *= 1.e6;                                        // Switch to MHz
 Isotope S(Iso);                                    // Make a spin isotope
 double I = S.qn();                                 // This is isotope I value
 IntQuad Q(I,wQ2QCC(wQ, I), eta);                   // Set a Quad interaction
 if(!int(2*I)%2)
   {
   cout << "\n\n\tSorry, I Must Be m*1/2, m Odd!\n\n";
   exit(-1);
   }
//                    Set Things Up For The Powder Average
 int npts = 4096;                                   // Block size
 int Ntheta, Nphi=0;                                // Angle increment counts
 query_parameter(argc, argv, qn++,                  // Get the theta increments
   "\n\t# Theta (z down) Increments Spanning [0, 180]? ", Ntheta);
 if(eta)
   query_parameter(argc, argv, qn++,                // Get the phi increments
     "\n\t# Phi (x over) Increments Spanning [0, 360)? ", Nphi);
 matrix ABC = Q.wQcentral(Ntheta, Nphi);            // Prep. for 2nd order shifts
//                    Powder Averaging
//          Angle theta Is Down From The +z Axis, Angle phi Over From +x
// Note that since the 2nd order shift Wcentral(theta, phi) is symmetric with
// respect to both angles we need only average over parts of both angle ranges.
// For theta this means we sum the results from angles [0, 90) + half the result
// at 90.  Twice that sum would produce the total theta average over [0, 180].
// For phi we usually average [0, 360) so this is reduced to a sum over 3/4 the
// result at 0 + the results from angles (0, 90) + 1/2 the result at 90.  Four
// times that sum would produce the total phi average over [0, 360).

 double dthe, Nm1o2 = double(Ntheta-1.0)/2.0;      // For powder average
 double dphi, Nm2o4 = double(Nphi)/4.0;             // For powder average
 int theta, phi;                                    // Orientation angles
 double W, WQ = Q.wQ();                             // Base Quad. frequency
 double Ifact = I*(I+1) - 0.75;                     // Part of the prefactor
 double prefact = -WQ*WQ*Ifact/Om;                  // Majority of the prefact
 double Aaxis = (-1.0/9.0)*prefact;                 // For plot scaling
 double Ctheta, Stheta, Cthetasq, Ctheta4;          // We'll need these
 double Fst = -2.5*Aaxis;                           // Starting plot limit
 double Ffi = 1.5*Aaxis;                            // End plot limit
 row_vector data(npts, complex0);                   // Array for spectrum
 for(theta=0; theta<Ntheta; theta++)                // Loop over theta angles
   {
   dthe = double(theta);
   if(dthe <= Nm1o2)                                // Only look upper half
     {                                              // of the sphere
```

```
    Ctheta = ABC.getRe(0,theta);                    // Scale factor cos(theta)
    Stheta = ABC.getRe(1,theta);                    // Scale factor sin(theta)
    Cthetasq = Ctheta*Ctheta;                       // cosine(theta)^2
    if(dthe == Nm1o2) Stheta *= 0.5;                // Half scale if theta=90
    if(!eta)                                        // Without eta, no phi
      {                                             // averaging is needed
      W=(prefact/16.)*(1.-Cthetasq)*(9.*Cthetasq-1.);  // Here is W adjustment
      addW(data, Fst, Ffi, W, Stheta);              // Add transition to spectrum
      }
    else
      {
      cout.flush();
      Ctheta4 = Cthetasq*Cthetasq;                  // cosine(theta)^4
      for(phi=0; phi<Nphi; phi++)                   // Loop over phi angles
        {
        dphi = double(phi);                         // Phi index as double
        if(dphi <= Nm2o4)                           // Only sum 1st quarter
          {
          if(!phi) Stheta *= 0.75;                  // 3/4 scale if phi=0
          else if(dphi == Nm2o4) Stheta *= 0.5;     // 1/2 scale if phi=90
          W = ABC.getRe(2,phi)*Ctheta4;             // A part of W
          W += ABC.getRe(3,phi)*Cthetasq;           // B part of W
          W += ABC.getRe(4,phi);                    // C part of W
          W *= (prefact/6.);                        // Scale
          addW(data, Fst, Ffi, W, Stheta);          // Add transition to spectrum
          }
        }
      }
    }
  }
double lb = 40.0;                                   // Set a line broading factor
cout << "\n\n\tDone With Discrete Powder Average. Processing...";
cout.flush();
data = IFFT(data);                                  // Put into time domain
exponential_multiply(data,-lb);                     // Apodize the "FID"
data = FFT(data);                                   // Put back into frequency domain
GP_1D("spec.asc", data, 0, -2.5, 1.5);              // Output the points in ASCII
GP_1Dplot("spec.gnu", "spec.asc");                  // Call Gnuplot and make plot now
}
```

5 Electron G Interactions

5.1 Overview

The class IntG contains a fully functional G interaction defined for a single electron. The class allows for the definition and manipulation of such interactions, in particular it allows for the construction of oriented G Hamiltonians. Note that this class does keep track of the isotropic G component.

5.2 Available G Interaction Functions

5.3 G Interaction Theory

5.4 G Interaction Figures

5.5 Literature Comparisons

5.6 G Interaction Examples

5.7 G Interaction Example Programs

5.8 G Interaction Constructors

5.8.1 IntG

Usage:

void IntG::IntG()
void IntG::IntG(const IntG& G1)
void IntG::IntG(double giso, double delzz, double eta=0.0, double theta=0.0, double phi=0.0)
void IntG::IntG(const coord& GC, double theta=0.0, double phi=0.0)
void IntG::IntG(ParameterAVLSet& pset, int idx=-1)

Description:

The function ***IntG*** is used to create a new G interaction.

1. IntG() - Called without arguments the function creates a NULL G interaction.
2. IntG(const IntG& G1) - Called with another G interaction, a new G interaction is constructed which is identical to the input interaction.
3. IntG(double giso, double delzz, double eta=0.0, double theta=0.0, double phi=0.0) - This will construct a new electron G interaction. The strength of the interaction is set by the arguments ***giso*** and ***delzz,*** both unitless quantities. The value of ***giso*** lis related to the trace of Gtensor. The value of ***delzz***, the G tensor anisotropy, . The value of ***eta*** can be optionally input and this set the asymmetry of the interaction. It is restricted to be within the range ***[0, 1]***. Two angles, ***theta*** and ***phi***, can be optionally specified in ***degrees***. This will set the orientation of the G interaction relative to its PAS in a standard spherical coordinate system.
4. IntG(const coord& GC, double theta=0, double phi=0) - This will construct a new electron G interaction from its three PAS Cartesian components which are contained in coordinate GC. The GC contains gxx, gyy, and gzz of the G tensor. Two angles, ***theta*** and ***phi***, can be optionally specified in ***degrees***. This will set the orientation of the G interaction relative to its PAS in a standard spherical coordinate system.
5. IntG(ParameterAVLSet& pset, int idx=-1) - This will construct a new G interaction for a spin having the quantum number ***qn*** from parameters found in the parameter set ***pset***. If the optional index ***idx*** has been set ***>=0*** the G parameters scanned in ***pset*** will be assumed to have a ***(idx)*** appended to their names.

Return Value:

Void. It is used strictly to create a G interaction.

Examples:

```
IntG G;                          // An empty G interaction.
IntG G1(1.5, 3.e5,.2, 45., 30.); // G. Int. for I=3/2, GCC=300kHz, η=.2, θ=45, ϕ=30
IntG G2(G1);                     // Another Guad. Interaction, here equal to G1
G = G2;                          // Now G is the same as G1 and G2
```

See Also: =, read, ask_read

5.8.2 =

Usage:

void IntG::operator= (const IntG& G1)

Description:

The operator = is assign one G interaction to another.

Return Value:

Void.

Example:

```
IntG G;                          // An empty G interaction.
IntG G1(1.5, 3.e5,.2, 45., 30.); // G. Int. for I=3/2, GCC=300kHz, η=.2, θ=45, ϕ=30
G = G1;                          // Now G is the same as G1
```

See Also: constructor, read, ask_read

5.9 Basic Functions

5.9.1 iso

Usage:

```
#include <IntG.h>
double IntG::iso () const
double IntG::iso (double giso) const
```

Description:

The function ***iso*** is used to either obtain or set the G interaction tensor isotropic value. With no arguments the function returns the value (unitless). If an argument, ***giso***, is specified then the isotropic value for the interaction is set. It is assumed that the input value of ***giso*** is unitless and is related to 1/3 the trace of the G tensor. Note that setting of ***giso*** will alter the (equivalent) value of the G tensor.

$$giso = e^2qQ = QCC = NQCC$$

Return Value:

Either void or a floating point number, double precision.

Example(s):

```
#include <IntG.h>
IntG G();                    // Empty G interaction.
G.delzz(100000.0);           // Set GCC to 100 KHz.
cout << G.delz ();           // Write coupling constant to std output.
```

See Also: GCC, NGCC, wG

5.9.2 delzz

Usage:

```
#include <IntG.h>
double IntG::delzz () const
double IntG::delz () const
double IntG::delzz (double dz) const
double IntG::delz (double dz) const
```

Description:

The function ***delzz*** is used to either obtain or set the interaction G coupling constant. With no arguments the function returns the coupling in Hz. If an argument, ***dz***, is specified then the coupling constant for the interaction is set. It is assumed that the input value of ***dz*** is in units of ***Hz***. The function is overloaded with the name ***delz*** for convenience. Note that setting of ***delzz*** will alter the (equivalent) value of the G coupling ***GCC*** (or ***NGCC***) as well as the G frequency.

$$\delta_{zz}^{G} = e^2qQ = QCC = NQCC$$

Return Value:

Either void or a floating point number, double precision.

Example(s):

```
#include <IntG.h>
IntG G();                    // Empty G interaction.
G.delzz(100000.0);           // Set GCC to 100 KHz.
cout << G.delz ();           // Write coupling constant to std output.
```

See Also: GCC, NGCC, wG

5.9.3 GCC, NGCC

Usage:

```
#include <IntG.h>
double IntG::GCC () const
double IntG::NGCC () const
double IntG::GCC (double dz) const
double IntG::NGCC (double dz) const
```

Description:

The function ***GCC*** is used to either obtain or set the interaction G coupling constant. With no arguments the function returns the coupling in Hz. If an argument, ***dz***, is specified then the coupling constant for the interaction is set. It is assumed that the input value of ***dz*** is in units of ***Hz***. The function is overloaded with the name ***NGCC*** for convenience. Note that setting of ***GCC*** will alter the (equivalent) value of the G spatial tensor ***delzz*** value as well as the G frequency. This function has identical functionality as ***delzz*** and ***delz***.

$$QCC = NQCC = e^2qQ = \delta_{zz}^{Q} = \frac{2I(2I-1)\omega^{Q}}{3}$$

Return Value:

Either void or a floating point number, double precision.

Examples:

```
#include <IntG.h>
IntG G();                    // Empty G interaction.
G.NGCC(100000.0);            // Set GCC to 100 KHz.
cout << G.GCC ();            // Write coupling constant to std output.
```

See Also: delz, delzz, wG

5.9.4 eta

Usage:

```
#include <IntG.h>
double IntG::eta () const
double IntG::eta (double Geta) const
```

Description:

The function ***eta*** is used to either obtain or set the G interaction asymmetry. With no arguments the function returns the asymmetry (unitless). If an argument, ***Geta***, is specified then the asymmetry for the interaction is set.The input value is **restricted to the range [0,1]** and is related to the G spatial tensor Cartesian components according to

$$\eta = (A_{xx} - A_{yy}) / A_{zz} \qquad |A_{zz}| \geq |A_{yy}| \geq |A_{xx}|$$

Note that setting ***eta*** will alter the 5 internal irreducible spherical spatial tensor components of the interaction.

Return Value:

Either void or a floating point number, double precision.

Examples:

```
#include <IntG.h>
IntG G();                    // Empty G interaction.
G.eta(0.75);                 // Set eta to 0.75.
double Geta = G.eta();       // Set Geta to current eta value
```

See Also: delz, delzz, wG

5.9.5 wG

Usage:

```
#include <IntG.h>
double IntG::wG () const
double IntG::wG (double W) const
```

Description:

The function ***wG*** is used to either obtain or set the interaction G frequency. With no arguments the function returns the frequency in Hz. If an argument, ***W***, is specified then the frequency for the interaction is set. It is assumed that the input value of ***W*** is in units of ***Hz***. In GAMMA the G frequency[1] is defined to be.

1. There are variations in the literature as to what the G frequency is. The definition in GAMMA is set such that the G interaction will split the observed NMR transitions by ω^Q when the Zeeman interaction is strong (i.e. high field, first-order G interaction). This definition is analogous to that of a scalar coupling.

$$\omega^Q = \frac{3e^2qQ}{2I(2I-1)} = \frac{3QCC}{2I(2I-1)} = \sqrt{\frac{15}{2\pi}}\xi^Q$$

Return Value:

Either void or a floating point number, double precision.

Examples:

```
#include <IntG.h>
IntG G();                    // Empty G interaction.
G.wG(1.4e5);                 // Set quad. frequency to 140 KHz.
cout << G.wG();              // Write frequency to std output.
```

See Also: delz, delzz, GCC, NGCC, xi

5.9.6 wG0, wGoriented

Usage:

```
double IntG::wGoriented() const
double IntG::wG0() const
double IntG::wGoriented(double theta, double phi) const
double IntG::wG0(double theta, double phi) const
matrix IntG::wGoriented(int Ntheta, int Nphi) const
matrix IntG::wG0(int Ntheta, int Nphi) const
```

Description:

The function ***wG0*** (or its equivalent ***wGoriented***) is used to obtain or generate the 1st order G frequency for a chosen orientation in ***Hz***. If the arguments, ***theta*** and ***phi***, are specified then the frequency will be returned at that orientation from the PAS rather that the internal orientation. It is assumed that the input angle values are in units of ***degrees***. In GAMMA the oriented G frequency[1] is defined to be

$$\omega^Q(\theta,\varphi) = \omega^Q(PAS)\cdot\sqrt{\frac{4\pi}{5}}A^Q_{2,0}(\theta,\varphi) = \frac{\omega^Q(PAS)}{2}[3\cos^2\theta - 1 + \eta\sin^2\theta\cos 2\varphi]$$

Alternatively, one may obtain an array of the mathematical precursors needed to generate the 1st order frequency over evenly spaced angle increments on the unit sphere. In this case the function is called with the integers ***Ntheta*** and ***Nphi***, the number of increments down and over respectively. The matrix returned will have 4 rows whose elements are given by

1. There are variations in the literature as to what the G frequency is. The definition in GAMMA is set such that the G interaction will split the observed NMR transitions by ω^Q when the Zeeman interaction is strong (i.e. high field, first-order G interaction). This definition is analogous to that of a scalar coupling.

$$\langle 0|mx|j\rangle = \frac{1}{2}(3\cos^2\theta_j - 1) \qquad \langle 1|mx|j\rangle = \frac{1}{2}\eta\sin^2\theta_j \qquad \langle 2|mx|j\rangle = \sin\theta_j$$

$$\langle 3|mx|j\rangle = \cos 2\varphi_j \qquad \theta_j = \frac{180\cdot j}{\text{Ntheta} - 1} \qquad \varphi_j = \frac{360\cdot j}{\text{Nphi}}$$

and the 1st order shifts reconstructed from

$$\omega^Q(\theta_i, \varphi_j) = \omega^Q(\langle 0|mx|i\rangle + \langle 1|mx|i\rangle\langle 3|mx|j\rangle)$$

Remember, these frequencies are the splittings between transitions to first order (high field approximation) for particular orientations. They are valid only when the Zeeman interaction is much stronger than the G interaction. One should use the second order frequency corrections when the Larmor frequency is only somewhat stronger than the G frequency. One should use the full treatment when the G interaction dominates.

Return Value:

Either void or a floating point number, double precision.

Examples:

```
IntG G();                    // Empty G interaction.
G.wG(1.4e5);                 // Set quad. frequency to 140 KHz.
cout << G.wG();              // Write frequency to std output.
```

See Also: delz, delzz, GCC, NGCC, xi

5.9.7 wGcentral

Usage:

double IntG::wGcentral(double Om) const
double IntG::wGcentral(double Om, double theta, double phi) const
matrix IntG::wGcentral(int Ntheta, int Nphi) const

Description:

The function ***wGcentral*** is used to obtain the interaction G frequency. The argument ***Om*** is used to indicate the Larmor frequency in ***Hz*** of the spin associated with the interaction. With no other arguments the shift will be that of the central transition at the interaction's internal orientation. With the additional arguments ***theta*** and ***phi*** the frequency will be the central transition second order shift at that orientation from the PAS rather that the internal orientation. It is assumed that the input angle values are in units of ***degrees***.

In GAMMA the 2nd order shifts to the central transition are given by

$$\omega^{Q,(2)}_{-\frac{1}{2},\frac{1}{2}}(\eta, \theta, \varphi) = \frac{-(\omega^Q)^2}{6\Omega}\left[I(I+1) - \frac{3}{4}\right][A(\eta, \varphi)\cos^4\theta + B(\eta, \varphi)\cos^2\theta + C(\eta, \varphi)]$$

where

$$A(\eta, \varphi) = \frac{-27}{8} + \frac{9}{4}\eta\cos(2\varphi) - \frac{3}{8}\eta^2\cos^2(2\varphi)$$

$$B(\eta, \varphi) = \frac{30}{8} - \frac{1}{2}\eta^2 - 2\eta\cos(2\varphi) + \frac{3}{4}\eta^2\cos^2(2\varphi)$$

$$C(\eta, \varphi) = \frac{-3}{8} + \frac{1}{3}\eta^2 - \frac{1}{4}\eta\cos(2\varphi) - \frac{3}{8}\eta^2\cos^2(2\varphi)$$

Alternatively, one may obtain an array of the mathematical precursors needed to generate the 2nd order shifts over evenly spaced angle increments on the unit sphere. In this case the function is called with the integers ***Ntheta*** and ***Nphi***, the number of increments down and over respectively. The matrix returned will have 5 rows whose elements are given by

$$\langle 0|mx|j\rangle = \cos\theta_j \qquad \langle 1|mx|j\rangle = \sin\theta_j$$

$$\langle 2|mx|j\rangle = A(\eta, \varphi_j) \qquad \langle 3|mx|j\rangle = B(\eta, \varphi_j) \qquad \langle 4|mx|j\rangle = C(\eta, \varphi_j)$$

$$\theta_j = \frac{180 \cdot j}{\text{Ntheta} - 1} \qquad \varphi_j = \frac{360 \cdot j}{\text{Nphi}}$$

and the shifts reconstructed from the previous equations.

Note that since second order effects are field dependent, the larger the field the smaller the returned shift(s). Also, the method of obtains such shifts in this function assumes that the G interaction is a perturbation to the Zeeman Hamiltonian. The will not be applicible when the G splitting is on the same scale as or larger than the Larmor frequency. Finally, if I is not half integer all values returned will be zero.

Return Value:

Either void or a floating point number, double precision.

Examples:

```
IntG G();                    // Empty G interaction.
G.wG(1.4e5);                 // Set quad. frequency to 140 KHz.
cout << G.wG();              // Write frequency to std output.
```

See Also: delz, delzz, GCC, NGCC, xi

5.9.8 wG1

Usage:

double IntG::wG1(double Om, double m) const
double IntG::wG1(double Om, double m, double theta, double phi) const
matrix IntG::wG1(int Ntheta, int Nphi) const

Description:

The function ***wG1*** is used to obtain the second order frequency shift of a G transition. The argument ***Om*** is used to indicate the Larmor frequency in ***Hz*** of the spin associated with the interaction. The value of ***m*** is the spin anglular momentum z quantum number and should span [I, I-1,I-2,.....-I+1]. The returned shift will be for the transtion between levels ***m*** and ***m-1***. With no addtional arguments the shift will be for the specified

transition at the interaction's internal orientation. With the additional arguments ***theta*** and ***phi*** the frequency will be the indicated transitions second order shift at that orientation from the PAS rather that the internal orientation. It is assumed that the input angle values are in units of ***degrees***.

In GAMMA the 2nd order shifts for the ***m***,***m-1*** transition are given by

$$\omega^{Q,(2)}_{m-1,m}(\eta,\theta,\varphi) = -\frac{\xi^2}{2\Omega_o}\Big\{A^Q_{2,1}(\eta,\theta,\varphi)A^Q_{2,-1}(\eta,\theta,\varphi)[24m(m-1)-4I(I+1)+9]$$

$$+\frac{1}{2}A^Q_{2,2}(\eta,\theta,\varphi)A^Q_{2,-2}(\eta,\theta,\varphi)[12m(m-1)-4I(I+1)+6]\Big\}$$

Alternatively, one may obtain an array of the mathematical precursors needed to generate the 2nd order shifts over evenly spaced angle increments on the unit sphere. In this case the function is called with the integers ***Ntheta*** and ***Nphi***, the number of increments down and over respectively. The matrix returned will have 6 rows whose elements are given by

$$\langle 0|mx|j\rangle = 3\sqrt{\frac{5}{24\pi}}\sin\theta_j\cos\theta_j \qquad \langle 1|mx|j\rangle = \frac{3}{2}\sqrt{\frac{5}{24\pi}}\sin^2\theta_j$$

$$\langle 2|mx|j\rangle = -\eta\sqrt{\frac{5}{24\pi}}(\cos 2\varphi_j - i\sin 2\varphi_j) \qquad \langle 3|mx|j\rangle = \frac{\eta}{2}\sqrt{\frac{5}{24\pi}}[\cos 2\varphi_j - i2\sin 2\varphi_j]$$

$$\langle 4|mx|j\rangle = \sin\theta_j \qquad \langle 5|mx|j\rangle = \cos\theta_j$$

$$\theta_j = \frac{180\cdot j}{\text{Ntheta}-1} \qquad \varphi_j = \frac{360\cdot j}{\text{Nphi}}$$

Reconstruction of full $A^Q_{2,m}(\theta,\varphi)$ values is based on

$$A^Q_{2,1}(\theta,\varphi) = A^Q_{2,1}(\theta,\varphi)\Big|_{\eta=0} + \sin\theta\cos\theta Re(A^Q_{2,1}B(\varphi)) + i\sin\theta Im(A^Q_{2,1}B(\varphi))$$

$$A^Q_{2,2}(\theta,\varphi) = A^Q_{2,2}(\theta,\varphi)\Big|_{\eta=0} + (1+\cos^2\theta)Re(A^Q_{2,2}B(\varphi)) + i\cos\theta Im(A^Q_{2,2}B(\varphi))$$

Required $A^Q_{2,m}(\theta_k,\varphi_l)$ components can be reconstructed according to the discrete equations below.

$$A^Q_{2,1}(\theta_k,\varphi_l) = \langle 0|mx|k\rangle + \langle 4|mx|k\rangle[\langle 5|mx|k\rangle Re\langle 2|mx|l\rangle + iIm\langle 2|mx|l\rangle]$$

$$A^Q_{2,2}(\theta_k,\varphi_l) = \langle 1|mx|k\rangle + (1+\langle 5|mx|k\rangle^2)Re\langle 3|mx|l\rangle + i\langle 5|mx|k\rangle Im\langle 3|mx|l\rangle$$

and the frequencies subsequently generated using

$$A^Q_{2,1}A^Q_{2,-1} = -A^Q_{2,1}A^{Q\,*}_{2,1} \qquad A^Q_{2,2}A^Q_{2,-2} = A^Q_{2,2}A^{Q\,*}_{2,2}$$

Note that since second order effects are field dependent, the larger the field the smaller the returned shift(s). Also, the method of obtains such shifts in this function assumes that the G interaction is a perturbation to the Zeeman Hamiltonian. The will not be applicible when the G splitting is on the same scale as or larger than

the Larmor frequency. Finally, if I is not half integer all values returned will be zero.

Return Value:

Either void or a floating point number, double precision.

Examples:

```
IntG G();                    // Empty G interaction.
G.wG(1.4e5);                 // Set quad. frequency to 140 KHz.
cout << G.wG();              // Write frequency to std output.
```

See Also: delz, delzz, GCC, NGCC, xi

5.9.9 xi

Usage:

double IntG::xi() const

Description:

The function ***xi*** is used to either obtain the GAMMA defined G interaction constant. The constant is used to scale the interaction such that both its spatial and spin tensors are “independent” of the interaction type.

$$\xi^Q = \sqrt{\frac{6\pi}{5}}\frac{e^2qQ}{2I(2I-1)} = \sqrt{\frac{6\pi}{5}}\frac{QCC}{2I(2I-1)} = \sqrt{\frac{2\pi}{15}}\omega^Q$$

This will be used in the formulation of G Hamiltonians according to.

$$\boldsymbol{H}^Q(\theta,\varphi) = \xi^Q\sum_{m}^{\pm 2}(-1)^m A^Q_{2,-m}(\theta,\varphi)\bullet\boldsymbol{T}^Q_{2,m}$$

Return Value:

A floating point number, double precision.

Examples:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
double Xi = G.xi();          // Get quad. interaction constant.
```

5.10 Spherical Spatial Tensor Functions

5.10.1 A0, A20

Usage:

```
#include <IntG.h>
complex IntG::A0() const
complex IntG::A20() const
complex IntG::A0(double theta, double phi) const
complex IntG::A20(double theta, double phi) const
```

Description:

The functions ***A0*** and ***A20*** are used to obtain the G interaction spatial tensor component $A_{2,0}$. If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments ***theta*** and ***phi*** are given the returned tensor component is for the orientation at ***theta*** degrees down from the interactions PAS z-axis and ***phi*** degrees over from the interactions PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***Hz***.

$$A^{Q}_{2,\,0}(\theta,\,\varphi) \;=\; \sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]$$

Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type[1]. This component can also be related to the Cartesian tensor components for any arbitrary orientation.

$$A_{2,\,0} \;=\; \sqrt{6}[3A_{zz} - Tr\{A\}]$$

Return Value:

A complex number.

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
complex A20 = G.A20();              // This is at theta=phi=45 degrees
cout << G.A20(15.6, 99.3);          // This is at theta=15.6 and phi=99.3 degrees.
```

1. Because the GAMMA platform accommodates different interaction types, the scaling on all spatial tensors is chosen to be independent of the interaction. Rather, the spatial tensors are related directly to the familiar rank two spherical harmonics $A^{Q}_{2,\,m}(\theta,\,\varphi)\Big|_{\eta\,=\,0} \;=\; Y^{2}_{m}(\theta,\,\varphi)$. Also, the sign on the term(s) involving η will have opposite sign if the common alternative definition of the PAS orientation ($|A_{zz}| \geq |A_{xx}| \geq |A_{yy}|$) is used rather that the definition used in GAMMA ($|A_{zz}| \geq |A_{yy}| \geq |A_{xx}|$)

See Also: A1, A21, Am1, A2m1, A2, A22, Am2, A2m2

5.10.2 A1, A21

Usage:

```
#include <IntG.h>
complex IntG::A1() const
complex IntG::A21() const
complex IntG::A1(double theta, double phi) const
complex IntG::A21(double theta, double phi) const
```

Description:

The functions ***A1*** and ***A21*** are used to obtain the G interaction spatial tensor component $A_{2,1}$. If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments ***theta*** and ***phi*** are given the returned tensor component is for the orientation at ***theta*** degrees down from the interactions PAS z-axis and ***phi*** degrees over from the interactions PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***Hz***.

$$A^{Q}_{2,1}(\theta, \varphi) = \sqrt{\frac{5}{24\pi}}\sin\theta[3\cos\theta - \eta(\cos\theta\cos 2\varphi - i\sin 2\varphi)]$$

Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type[1]. This component can also be related to the Cartesian tensor components for any arbitrary orientation.

$$A_{21} = -\frac{1}{2}[A_{xz} + A_{zx} + i(A_{yz} + A_{zy})]$$

Return Value:

A complex number.

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
complex A20 = G.A20();              // This is at theta=phi=45 degrees
cout << G.A20(15.6, 99.3);          // This is at theta=15.6 and phi=99.3 degrees.
```

1. Because the GAMMA platform accommodates different interaction types, the scaling on all spatial tensors is chosen to be independent of the interaction. Rather, the spatial tensors are related directly to the familiar rank two spherical harmonics $A^{Q}_{2,m}(\theta, \varphi)\Big|_{\eta = 0} = Y^{2}_{m}(\theta, \varphi)$. Also, the sign on the term(s) involving η will have opposite sign if the common alternative definition of the PAS orientation ($|A_{zz}| \geq |A_{xx}| \geq |A_{yy}|$) is used rather that the definition used in GAMMA ($|A_{zz}| \geq |A_{yy}| \geq |A_{xx}|$)

See Also: A1, A21, Am1, A2m1, A2, A22, Am2, A2m2

5.10.3 Am1, A2m1

Usage:

```
#include <IntG.h>
complex IntG::Am1() const
complex IntG::A2m1() const
complex IntG::Am1(double theta, double phi) const
complex IntG::Am21(double theta, double phi) const
```

Description:

The functions ***Am1*** and ***A2m1*** are used to obtain the G interaction spatial tensor component $A_{2,1}$. If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments ***theta*** and ***phi*** are given the returned tensor component is for the orientation at ***theta*** degrees down from the interactions PAS z-axis and ***phi*** degrees over from the interactions PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***Hz***.

$$A^Q_{2,-1}(\theta,\varphi) = -\sqrt{\frac{5}{24\pi}}\sin\theta[3\cos\theta - \eta(\cos\theta\cos 2\varphi + i\sin 2\varphi)] = -A^{Q\,*}_{2,1}(\theta,\varphi)$$

Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type[1]. This component can also be related to the Cartesian tensor components for any arbitrary orientation.

$$A_{2,-1} = \frac{1}{2}[A_{xz} + A_{zx} + i(A_{yz} - A_{zy})]$$

Return Value:

A complex number.

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
complex A20 = G.A20();              // This is at theta=phi=45 degrees
cout << G.A20(15.6, 99.3);          // This is at theta=15.6 and phi=99.3 degrees.
```

1. Because the GAMMA platform accommodates different interaction types, the scaling on all spatial tensors is chosen to be independent of the interaction. Rather, the spatial tensors are related directly to the familiar rank two spherical harmonics $A^Q_{2,m}(\theta,\varphi)\big|_{\eta=0} = Y^2_m(\theta,\varphi)$. Also, the sign on the term(s) involving η will have opposite sign if the common alternative definition of the PAS orientation ($|A_{zz}| \geq |A_{xx}| \geq |A_{yy}|$) is used rather that the definition used in GAMMA ($|A_{zz}| \geq |A_{yy}| \geq |A_{xx}|$)

See Also: A1, A21, Am1, A2m1, A2, A22, Am2, A2m2

5.10.4 A2, A22

Usage:

```
#include <IntG.h>
complex IntG::A2() const
complex IntG::A22() const
complex IntG::A2(double theta, double phi) const
complex IntG::A22(double theta, double phi) const
```

Description:

The functions ***A2*** and ***A22*** are used to obtain the G interaction spatial tensor component $A_{2,1}$. If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments ***theta*** and ***phi*** are given the returned tensor component is for the orientation at ***theta*** degrees down from the interactions PAS z-axis and ***phi*** degrees over from the interactions PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***Hz***.

$$A^{Q}_{2,\,2}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}\frac{1}{2}[3\sin^2\theta + \eta[\cos 2\varphi(1+\cos^2\theta) - i2\sin 2\varphi\cos\theta]]$$

Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type[1]. This component can also be related to the Cartesian tensor components for any arbitrary orientation.

$$A_{2,\,2} = \frac{1}{2}[A_{xx} - A_{yy} + i(A_{xy} + A_{yx})]$$

Return Value:

A complex number.

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
complex A20 = G.A20();              // This is at theta=phi=45 degrees
cout << G.A20(15.6, 99.3);          // This is at theta=15.6 and phi=99.3 degrees.
```

1. Because the GAMMA platform accommodates different interaction types, the scaling on all spatial tensors is chosen to be independent of the interaction. Rather, the spatial tensors are related directly to the familiar rank two spherical harmonics $A^{Q}_{2,\,m}(\theta,\varphi)\Big|_{\eta=0} = Y^2_m(\theta,\varphi)$. Also, the sign on the term(s) involving η will have opposite sign if the common alternative definition of the PAS orientation ($|A_{zz}| \geq |A_{xx}| \geq |A_{yy}|$) is used rather that the definition used in GAMMA ($|A_{zz}| \geq |A_{yy}| \geq |A_{xx}|$)

See Also: A1, A21, Am1, A2m1, A2, A22, Am2, A2m2

5.10.5 Am2, A2m2

Usage:

```
#include <IntG.h>
complex IntG::Am2() const
complex IntG::A2m2() const
complex IntG::Am2(double theta, double phi) const
complex IntG::A2m2(double theta, double phi) const
```

Description:

The functions ***Am2*** and ***A2m2*** are used to obtain the G interaction spatial tensor component $A_{2,-2}$. If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments ***theta*** and ***phi*** are given the returned tensor component is for the orientation at ***theta*** degrees down from the interactions PAS z-axis and ***phi*** degrees over from the interactions PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***degrees***.

$$A^Q_{2,-2}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}\frac{1}{2}[3\sin^2\theta + \eta[\cos 2\varphi(1+\cos^2\theta) + i2\sin 2\varphi\cos\theta]] = A^{Q\,*}_{2,2}(\theta,\varphi)$$

Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type[1]. This component can also be related to the Cartesian tensor components for any arbitrary orientation.

$$A_{2,-2} = \frac{1}{2}[A_{xx} + (-A_{yy}) - i(A_{xy} + A_{yx})]$$

Return Value:

A complex number.

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
complex A20 = G.A20();              // This is at theta=phi=45 degrees
cout << G.A20(15.6, 99.3);          // This is at theta=15.6 and phi=99.3 degrees.
```

See Also: A1, A21, Am1, A2m1, A2, A22, Am2, A2m2

1. Because the GAMMA platform accommodates different interaction types, the scaling on all spatial tensors is chosen to be independent of the interaction. Rather, the spatial tensors are related directly to the familiar rank two spherical harmonics $A^Q_{2,m}(\theta,\varphi)\big|_{\eta=0} = Y^2_m(\theta,\varphi)$. Also, the sign on the term(s) involving η will have opposite sign if the common alternative definition of the PAS orientation ($|A_{zz}| \geq |A_{xx}| \geq |A_{yy}|$) is used rather that the definition used in GAMMA ($|A_{zz}| \geq |A_{yy}| \geq |A_{xx}|$)

5.11 Cartesian Spatial Tensor Functions

5.11.1 Axx

Usage:

```
#include <IntG.h>
complex IntG::Axx() const
complex IntG::Axx(double theta, double phi) const
```

Description:

The functions ***Axx*** is used to obtain the G interaction spatial tensor component A_{xx}. If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments ***theta*** and ***phi*** are given the returned tensor component is for the orientation at ***theta*** degrees down from the interactions PAS z-axis and ***phi*** degrees over from the interactions PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***Hz***.

$$A_{xx}(\theta, \varphi) = \sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]$$

Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type. This component can also be related to the spherical tensor components for any arbitrary orientation.

$$A_{xx} = \frac{1}{2}(A_{2,2} + A_{2,-2}) - \frac{1}{\sqrt{6}}A_{2,0}$$

Return Value:

A complex number.

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
complex A20 = G.A20();            // This is at theta=phi=45 degrees
cout << G.A20(15.6, 99.3);        // This is at theta=15.6 and phi=99.3 degrees.
```

See Also: Ayy, Azz, Axy, Axz, Ayx, Ayz, Azx, Azy

5.11.2 Ayy

Usage:

```
#include <IntG.h>
complex IntG::Ayy() const
complex IntG::Axx(double theta, double phi) const
```

Description:

The functions ***Ayy*** is used to obtain the G interaction spatial tensor component A_{yy}. If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments

theta and ***phi*** are given the returned tensor component is for the orientation at ***theta*** degrees down from the interactions PAS z-axis and ***phi*** degrees over from the interactions PAS x-a!a

567s. The values of ***theta*** and ***phi*** are assumed in ***Hz***.

$$A_{yy}(\theta, \varphi) = -\sqrt{\frac{5}{24\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]$$

Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type. This component can also be related to the spherical tensor components for any arbitrary orientation.

$$A_{yy} = \frac{-1}{2}(A_{2,2} + A_{2,-2}) - \frac{1}{\sqrt{6}}A_{2,0}$$

Return Value:

A complex number.

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
complex A20 = G.A20();              // This is at theta=phi=45 degrees
cout << G.A20(15.6, 99.3);          // This is at theta=15.6 and phi=99.3 degrees.
```

See Also: Axx, Azz, Axy, Axz, Ayx, Ayz, Azx, Azy

5.11.3 Azz

Usage:

```
#include <IntG.h>
complex IntG::Azz() const
complex IntG::Azz(double theta, double phi) const
```

Description:

The functions ***Azz*** is used to obtain the G interaction spatial tensor component A_{zz}. If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments ***theta*** and ***phi*** are given the returned tensor component is for the orientation at ***theta*** degrees down from the interactions PAS z-axis and ***phi*** degrees over from the interactions PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***Hz***.

$$A_{zz}(\theta, \varphi) = \sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]$$

Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type. This component can also be related to the spherical tensor components for any arbitrary orientation.

$$A_{zz} = \sqrt{\frac{2}{3}}A_{2,0}$$

Return Value:

A complex number.

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
complex A20 = G.A20();          // This is at theta=phi=45 degrees
cout << G.A20(15.6, 99.3);      // This is at theta=15.6 and phi=99.3 degrees.
```

See Also: Axx, Ayy, Axy, A2xz, Ayx, Ayz, Azx, Azy

5.11.4 Axy, Ayx

Usage:

```
#include <IntG.h>
complex IntG::Axy() const
complex IntG::Axy(double theta, double phi) const
complex IntG::Ayx() const
complex IntG::Ayx(double theta, double phi) const
```

Description:

The functions ***Axy*** and ***Ayx*** are used to obtain the G interaction spatial tensor component $A_{xy} = A_{yx}$. If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments ***theta*** and ***phi*** are given the returned tensor component is for the orientation at ***theta*** degrees down from the interactions PAS z-axis and ***phi*** degrees over from the interactions PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***Hz***.

$$A_{xy}(\theta, \varphi) = \sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]$$

Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type. This component can also be related to the spherical tensor components for any arbitrary orientation.

$$A_{xy} = -\frac{i}{2}(A_{2,2} - A_{2,-2}) = A_{yx}$$

Return Value:

A complex number.

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
complex A20 = G.A20();          // This is at theta=phi=45 degrees
cout << G.A20(15.6, 99.3);      // This is at theta=15.6 and phi=99.3 degrees.
```

See Also: Axx, Ayy, Azz, Axz, Ayz, Azx, Azy

5.11.5 Axz, Azx

Usage:

```
#include <IntG.h>
complex IntG::Axz() const
complex IntG::Axz(double theta, double phi) const
complex IntG::Azx() const
complex IntG::Azx(double theta, double phi) const
```

Description:

The functions ***Axz*** and ***Azx*** are used to obtain the G interaction spatial tensor component $A_{xz} = A_{zx}$. If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments ***theta*** and ***phi*** are given the returned tensor component is for the orientation at ***theta*** degrees down from the interactions PAS z-axis and ***phi*** degrees over from the interactions PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***Hz***.

$$A_{xy}(\theta, \varphi) = \sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]$$

Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type. This component can also be related to the spherical tensor components for any arbitrary orientation.

$$A_{xz} = -\frac{1}{2}[(A_{2,1} - A_{2,-1})] = A_{zx}$$

Return Value:

A complex number.

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
complex A20 = G.A20();              // This is at theta=phi=45 degrees
cout << G.A20(15.6, 99.3);          // This is at theta=15.6 and phi=99.3 degrees.
```

See Also: Axx, Ayy, Azz, Axz, Ayz, Azx, Azy

5.11.6 Ayz, Azy

Usage:

```
#include <IntG.h>
complex IntG::Ayz() const
complex IntG::Ayz(double theta, double phi) const
complex IntG::Azy() const
complex IntG::Azy(double theta, double phi) const
```

Description:

The functions ***Ayz*** and ***Azy*** are used to obtain the G interaction spatial tensor component $A_{yz} = A_{zy}$. If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments ***theta*** and ***phi*** are given the returned tensor component is for the orientation at ***theta*** degrees down from the interactions PAS z-axis and ***phi*** degrees over from the interactions PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***Hz***.

$$A_{xy}(\theta, \varphi) = \sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]$$

Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type. This component can also be related to the spherical tensor components for any arbitrary orientation.

$$A_{yz} = \frac{i}{2}[(A_{2,1} + A_{2,-1})] = A_{zy}$$

Return Value:

A complex number.

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
complex A20 = G.A20();              // This is at theta=phi=45 degrees
cout << G.A20(15.6, 99.3);          // This is at theta=15.6 and phi=99.3 degrees.
```

See Also: Axx, Ayy, Azz, Axz, Ayz, Azx, Azy

5.12 Powder Average Facilitator Functions

5.12.1 A0A, A20A

Usage:

row_vector IntG::A0A(int Ntheta)
row_vector IntG::A20A(int Ntheta)

Description:

The functions ***A0A*** and ***A20A*** are equivalent. They are used to obtain part of G interaction spatial tensor component $A_{2,0}$ for a series of evenly incrmented θ values.

$$A_{2,0}A(\theta) = \sqrt{\frac{5}{16\pi}}(3\cos^2\theta - 1) = A^Q_{2,0}(\theta, \varphi)\Big|_{\eta = \varphi = 0}$$

Given a number of angle increments, ***Ntheta***, a row vector of dimension ***Ntheta*** will be returned which contains the η independent terms of $A^Q_{2,0}$ at evenly spaced increments of θ starting at the +z PAS ($\theta = 0$) alignment and finishing at -z PAS ($\theta = 180$) alignment.

$$\langle v|i\rangle = A^Q_{2,0}A(\theta_i) \qquad \theta_i = \frac{180i}{(\text{Ntheta} - 1)}$$

Note that to obtain the full $A^Q_{2,0}$ terms (if they are η dependent) they must be properly combined with the values from the function ***A20B.***

Return Value:

A vector.

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
row_vector A20s = G.A20A(720);    // Get 720 A20A values spanning [0, 180]
```

See Also: A21A, A22A, A20B, A21B, A22B, A2As, A2Bs, A2s

5.12.2 A1A, A21A

Usage:

row_vector IntG::A1A(int Ntheta)
row_vector IntG::A21A(int Ntheta)

Description:

The functions ***A1A*** and ***A21A*** are equivalent. They are used to obtain part of G interaction spatial tensor com-

ponent $A^Q_{2,1}$ for a series of evenly incrmented θ values.

$$A^Q_{2,1}A(\theta) = 3\sqrt{\frac{5}{24\pi}}\sin\theta\cos\theta = A^Q_{2,1}(\theta,\varphi)\Big|_{\eta=0}$$

Given a number of angle increments, ***Ntheta***, a row vector of dimension ***Ntheta*** will be returned which contains the η independent terms of $A^Q_{2,1}$ at evenly spaced increments of θ starting at the +z PAS ($\theta = 0$) alignment and finishing at -z PAS ($\theta = 180$) alignment.

$$\langle v|i\rangle = A^Q_{2,1}A(\theta_i) \qquad \theta_i = \frac{180i}{(\mathrm{Ntheta}-1)}$$

Note that to obtain the full $A^Q_{2,1}$ terms (if they are η dependent) they must be properly combined with the values from the function ***A21B.***

Return Value:

A vector.

Example:

```
IntG G(1.5, 3.e5, 0.2);              // Make a G interaction.
row_vector A21s = G.A21A(181);       // Get 181 A20A values spanning [0, 180]
```

See Also: A20A, A22A, A20B, A21B, A22B, A2As, A2Bs, A2s

5.12.3 A2A, A221A

Usage:

```
row_vector IntG::A2A(int Ntheta)
row_vector IntG::A22A(int Ntheta)
```

Description:

The functions ***A2A*** and ***A22A*** are equivalent. They are used to obtain part of G interaction spatial tensor component $A^Q_{2,2}$ for a series of evenly incremented θ values.

$$A^Q_{2,2}A(\theta) = \frac{3}{2}\sqrt{\frac{5}{24\pi}}\sin^2\theta = 3\sqrt{\frac{5}{96\pi}}\sin^2\theta = A^Q_{2,2}(\theta,\varphi)\Big|_{\eta=0}$$

Given a number of angle increments, ***Ntheta***, a row vector of dimension ***Ntheta*** will be returned which contains the η independent terms of $A^Q_{2,2}$ at evenly spaced increments of θ starting at the +z PAS ($\theta = 0$) alignment and finishing at -z PAS ($\theta = 180$) alignment.

$$\langle v|i\rangle = A^Q_{2,2}A(\theta_i) \qquad \theta_i = \frac{180i}{(\text{Ntheta} - 1)}$$

Note that to obtain the full $A^Q_{2,2}$ terms (if they are η dependent) they must be properly combined with the values from the function ***A22B.***

Return Value:

A vector.

Example:

```
IntG G(1.5, 3.e5, 0.2);              // Make a G interaction.
row_vector A22s = G.A22A(181);       // Get 181 A22A values spanning [0, 180]
```

See Also: A20A, A21A, A20B, A21B, A22B, A2As, A2Bs, A2s

5.12.4 A0B, A20B

Usage:

```
row_vector IntG::A0B(int Nphi)
row_vector IntG::A20B(int Nphi)
```

Description:

The functions ***A0B*** and ***A20B*** are equivalent. They are used to obtain part of G interaction spatial tensor component $A^Q_{2,0}$ for a series of evenly incrmented φ values.

$$A^Q_{2,0}B(\varphi) = \sqrt{\frac{5}{16\pi}}\eta\cos 2\varphi = \frac{1}{\sin^2\theta}\left[A^Q_{2,0}(\theta,\varphi) - A^Q_{2,0}(\theta,\varphi)\Big|_{\eta=0}\right]$$

Given a number of angle increments, ***Nphi***, a row vector of dimension ***Nphi*** will be returned which contains θ independent terms of $A^Q_{2,0}$ at evenly spaced increments of φ starting at the +x PAS ($\varphi = 0$) alignment and finishing at +x PAS ($\varphi = 360$) alignment.

$$\langle v|i\rangle = A^Q_{2,0}A(\varphi_i) \qquad \varphi_i = \frac{360i}{\text{Nphi}}$$

Note that to obtain the full $A^Q_{2,0}$ terms they must be properly combined with the values from the function ***A20A.***

$$A^Q_{2,0}(\theta,\varphi) = \sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]$$

$$A^Q_{2,1}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}\sin\theta[3\cos\theta - \eta(\cos\theta\cos 2\varphi - i\sin 2\varphi)] = -A^{Q\,*}_{2,-1}(\theta,\varphi)$$

$$A^Q_{2,2}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}\frac{1}{2}[3\sin^2\theta + \eta[\cos 2\varphi(1+\cos^2\theta) - i2\sin 2\varphi\cos\theta]] = A^{Q\,*}_{2,-2}(\theta,\varphi)$$

Return Value:

A vector.

Example:

```
IntG G(1.5, 3.e5, 0.2);// Make a G interaction.
row_vector A20s = G.A20B(120);    // Get 120 A20B values spanning [0, 360)
```

See Also: A20A, A21A, A22A, A21B, A22B, A2As, A2Bs, A2s

5.12.5 A1B, A21B

Usage:

```
row_vector IntG::A1B(int Nphi)
row_vector IntG::A21B(int Nphi)
```

Description:

The functions ***A1B*** and ***A21B*** are equivalent. They are used to obtain part of G interaction spatial tensor component $A^Q_{2,1}$ for a series of evenly incrmented φ values.

$$A^Q_{2,1}B(\varphi) = -\sqrt{\frac{5}{24\pi}}\eta(\cos 2\varphi - i\sin 2\varphi)$$

where

$$A^Q_{2,1}(\theta,\varphi) = \sin\theta\cos\theta Re(A^Q_{2,1}B(\varphi)) + i\sin\theta Im(A^Q_{2,1}B(\varphi)) + A^Q_{2,1}(\theta,\varphi)\Big|_{\eta=0}$$

Given a number of angle increments, ***Nphi***, a row vector of dimension ***Nphi*** will be returned which contains θ independent terms of $A^Q_{2,1}$ at evenly spaced increments of φ starting at the +x PAS ($\varphi = 0$) alignment and finishing at +x PAS ($\varphi = 360$) alignment.

$$\langle v|i\rangle = A^Q_{2,1}A(\varphi_i) \qquad \varphi_i = \frac{360i}{\text{Nphi}}$$

Note that to obtain the full $A^Q_{2,1}$ terms they must be properly combined with the values from the function

A21A.

Return Value:

A vector.

Example:

```
IntG G(1.5, 3.e5, 0.2);// Make a G interaction.
row_vector A21s = G.A21B(120);    // Get 120 A21B values spanning [0, 360)
```

See Also: A20A, A21A, A22A, A20B, A22B, A2As, A2Bs, A2s

5.12.6 A2B, A22B

Usage:

```
row_vector IntG::A2B(int Nphi)
row_vector IntG::A22B(int Nphi)
```

Description:

The functions ***A1B*** and ***A21B*** are equivalent. They are used to obtain part of G interaction spatial tensor component $A^Q_{2,2}$ for a series of evenly incrmented φ values.

$$A^Q_{2,2}B(\varphi) = \sqrt{\frac{5}{96\pi}}\eta[\cos 2\varphi - i2\sin 2\varphi]$$

where

$$A^Q_{2,2}(\theta,\varphi) = (1+\cos^2\theta)Re(A^Q_{2,2}B(\varphi)) + i\cos\theta Im(A^Q_{2,2}B(\varphi)) + A^Q_{2,2}(\theta,\varphi)\Big|_{\eta=0}$$

Given a number of angle increments, ***Nphi***, a row vector of dimension ***Nphi*** will be returned which contains θ independent terms of $A^Q_{2,2}$ at evenly spaced increments of φ starting at the +x PAS ($\varphi = 0$) alignment and finishing at +x PAS ($\varphi = 360$) alignment.

$$\langle v|i\rangle = A^Q_{2,2}A(\varphi_i) \qquad \varphi_i = \frac{360i}{\text{Nphi}}$$

Note that to obtain the full $A^Q_{2,2}$ terms they must be properly combined with the values from the function ***A22A.***

Return Value:

A vector.

Example:

```
IntG G(1.5, 3.e5, 0.2);                // Make a G interaction.
row_vector A22s = G.A22B(120);    // Get 120 A22B values spanning [0, 360)
```

See Also: A20A, A21A, A22A, A20B, A21B, A2As, A2Bs, A2s

5.12.7 A2s

Usage:

matrix IntG::A2s(int Ntheta, int Nphi)

Description:

The function ***A2s*** is used to construct the G interaction spatial tensor components $A^Q_{2,m}$ for a series of evenly incrmented θ and φ values. Given arguments for the number of angle increments, ***Ntheta*** and ***Nphi*** the function will return a matrix of dimenstion (8 x nc) where nc is the larger of the two input arguments. The matrix columns, indexed by j, will then correspond either to an angle θ or an angle φ where

$$\theta_j = \frac{180j}{(\text{Ntheta} - 1)} \qquad \varphi_j = \frac{360j}{\text{Nphi}}$$

depending upon which row is being accessed. Rows 0-2 of the array will correspond to the the η independent terms of $A^Q_{2,\{0,1,2\}}$ at evenly spaced increments of θ starting at the +z PAS ($\theta = 0$) alignment and finishing at -z PAS ($\theta = 180$) alignment. Rows 3-5 of the array will correspond to θ independent parts of the interaction spatial tensor components $A^Q_{2,\{0,1,2\}}$ at evenly spaced increments of φ starting at the +x PAS ($\varphi = 0$) alignment and finishing at +x PAS ($\varphi = 360$) alignment. The final three array columns will contain θ dependent terms that are used to blend with the other rows to form the full $A^Q_{2,m}(\theta, \varphi)$ values. Reconstruction of full $A^Q_{2,m}(\theta, \varphi)$ values is based on

$$A^Q_{2,0}(\theta, \varphi) = A^Q_{2,0}(\theta, \varphi)\Big|_{\eta = 0} + \sin^2\theta A^Q_{2,0} B(\varphi)$$

$$A^Q_{2,1}(\theta, \varphi) = A^Q_{2,1}(\theta, \varphi)\Big|_{\eta = 0} + \sin\theta\cos\theta Re(A^Q_{2,1}B(\varphi)) + i\sin\theta Im(A^Q_{2,1}B(\varphi))$$

$$A^Q_{2,2}(\theta, \varphi) = A^Q_{2,2}(\theta, \varphi)\Big|_{\eta = 0} + (1 + \cos^2\theta) Re(A^Q_{2,2}B(\varphi)) + i\cos\theta Im(A^Q_{2,2}B(\varphi))$$

A particular $A^Q_{2,m}(\theta_k, \varphi_l)$ can be reconstructed according to the analogous discrete equations.

$$A^Q_{2,0}(\theta_k, \varphi_l) = \langle 0|mx|k\rangle + \langle 6|mx|k\rangle^2 \langle 3|mx|l\rangle$$

$$A^Q_{2,1}(\theta_k, \varphi_l) = \langle 1|mx|k\rangle + \langle 6|mx|k\rangle[\langle 7|mx|k\rangle Re\langle 4|mx|l\rangle + iIm\langle 4|mx|l\rangle]$$

$$A^Q_{2,2}(\theta_k, \varphi_l) = \langle 2|mx|k\rangle + (1 + \langle 7|mx|k\rangle^2) Re\langle 5|mx|l\rangle + i\langle 7|mx|k\rangle Im\langle 5|mx|l\rangle$$

The components with m negative are obtained from the relationship .

$$A^{Q}_{2,\,-m} = (-1)^{m} A^{Q}_{2,\,m}$$

Return Value:

An array.

Example:

```
IntG G(1.5, 3.e5, 0.2);          // Make a G interaction.
matrix As = G.A2x(720, 360);     // Get array for values spanning [0, 180] & [0, 360)
```

See Also: A20A, A21A, A22A, A20B, A21B, A2As, A2Bs, A2s

5.13 Spin Tensor Functions

5.13.1 Tcomp

Usage:

```
#include <IntG.h>
matrix IntG::Tcomp(int comp)
```

Description:

The function ***Tcomp*** is used to obtain a G interaction spin tensor component. The component desired is specified by the argument ***comp*** which relates to the m value as follows:

comp:	0	1	2	3	4
$T^G_{2,m}$:	$T^G_{2,0}$	$T^G_{2,1}$	$T^G_{2,-1}$	$T^G_{2,2}$	$T^G_{2,-2}$

The spin components are given

$$T^Q_{2,0} = \frac{1}{\sqrt{6}}[3I_z^2 - \vec{I}^2] = \frac{1}{\sqrt{6}}[3I_z^2 - I(I+1)]$$

$$T^Q_{2,\pm 1} = \mp\frac{1}{2}[I_\pm\ I_z + I_z I_\pm\ ] \qquad T^Q_{2,\pm 2} = \frac{1}{2}I_\pm^2$$

and will be returned as matrices of dimension ***2I+1*** where ***I*** is the spin quantum number associated with the interaction.

Return Value:

A matrix.

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);     // Make a G interaction.
matrix T20 = G.Tcomp(0);                // This is the T20 spin tensor component
cout << T20);                           // Have a look on screen.
```

5.14 Auxiliary Functions

5.14.1 setPAS

Usage:

```
#include <IntG.h>
void IntG::setPAS()
```

Description:

The functions ***setPAS*** is used to orient the G interaction into it's principal axis system. All 5 spatial tensor components will be set to PAS values and the internal orientation angles set to zero.

Return Value:

None.

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
G.setPAS();                        // As if we used G(1.5,3.e5,0.2,0,0)
```

See Also: theta, phi, orient

5.14.2 symmetric

Usage:

```
#include <IntG.h>
int IntG::symmetric() const
```

Description:

The functions ***symmetric*** is used to check if the G interaction has any asymmetry. The function will return true if the interaction is symmetric and false if there is some asymmetry (non-zero eta value).

Return Value:

An integer

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
if(G.symmetric()) cout << "Yep";     // We should get No for G because eta=0.2)
else                  << "Nope";
```

See Also: eta

5.14.3 PAS

Usage:

```
int IntG::PAS) const
```

Description:

The function ***PAS*** is used to check if the G interaction is oriented in its PAS or not. The function will return true if the interaction is PAS aligned and false if not).

Return Value:

An integer

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
if(G.PAS()) cout << "Yep";          // We should get No for G because neither θ or ϕ is 0)
else                     << "Nope";
```

See Also: eta

5.14.4 wG2GCC

Usage:

```
#include <IntG.h>
friend double wG2GCC(double wG, double I)
```

Description:

The functions ***wG2GCC*** is used to convert a G frequency ***wG*** for a spin with quantum number ***I*** to a G coupling constant. The two are related in GAMMA by

$$QCC = e^2qQ = \frac{2I(2I-1)\omega^Q}{3} = 2I(2I-1)\sqrt{\frac{5}{6\pi}}\xi^Q$$

Return Value:

A double

Example:

```
double wG = 450.e3;                 // Guad. frequency of 450 kHz.
double NGCC = wG2GCC(wG, 1.5);// Guad. coupling if I=3/2
```

See Also: GCC2wG

5.14.5 GCC2wG

Usage:

```
#include <IntG.h>
friend double GCC2wG(double GCC, double I)
```

Description:

The functions ***GCC2wG*** is used to convert a G coupling constant to a G frequency. The two are related in GAMMA by

$$\omega^Q = \frac{3e^2qQ}{2I(2I-1)} = \frac{3QCC}{2I(2I-1)} = \sqrt{\frac{15}{2\pi}}\xi^Q$$

Return Value:

A double

Example:

```
double GCC = 450.e3;              // Guad. coupling constant of 450 kHz.
double wG = GCC2wG(wG, 1.5);      // Guad. frequency if I=3/2
```

See Also: wG2GCC

5.15 Hamiltonian Functions

5.15.1 H0

Usage:

```
#include <IntG.h>are
matrix IntG::H0() const
matrix IntG::H0(double theta, double phi) const
```

Description:

The function ***H0*** is used to obtain the G Hamiltonian as a first order perturbation to the Zeeman Hamiltonian. As such, the returned matrix is "secular" and commutes with both $\boldsymbol{F_z}$ and $\boldsymbol{R_z}$. It will be valid in a rotating frame about the z-axis. It will not be valid unless the Zeeman Hamiltonian (which it is meant to be added to[1]) is much stronger. The return array will have units of ***Hz***. The dimension of the array will be 2I+1 where I is the spin quantum value associated with the interaction. If the input arguments ***heta*** and ***phi*** are given the returned Hamiltonian is for the orientation at ***theta*** degrees down from the interaction PAS z-axis and ***phi*** degrees over from the interaction PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***degrees.***

In GAMMA the first order G Hamiltonian is given by

$$H_Q^{(0)} = \xi^Q A_{0,0}^Q(\eta, \theta, \phi) T_{0,0}^Q = \frac{\omega^Q}{12}[3\cos^2\theta - 1 + \eta\sin^2\theta\cos(2\varphi)][3I_z^2 - I(I+1)]$$

Return Value:

A matrix.

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
matrix H = G.H0();                 // Here's the 1st order Guad. Hamiltonian
cout << H;                         // Have a look at the Hamiltonian.
```

See Also: H1, Hsec, H

5.15.2 H1

Usage:

```
#include <IntG.h>
matrix IntG::H1() const
matrix IntG::H1(double theta, double phi) const
```

1. A spin in a strong magnetic field will evolve under the influence of both the Zeeman Hamiltonian, $H_Z$and the G Hamiltonian H_G. When the Zeeman interaction is much strong than the G interaction it suffices to use H0 instead of H_G. This is often nice to use because then the two Hamiltonians commute. In evolving a density operator one may then work in the rotating frame at a spin's Larmor frequency by simply removing the Zeeman Hamiltonian and evolving under only H0.

Description:

The function ***H1*** is used to obtain the second order G Hamiltonian as a perturbation to the Zeeman Hamiltonian. As such, the returned matrix is "secular" and commutes with both $\boldsymbol{F_z}$ and $\boldsymbol{R_z}$. It will be valid in a rotating frame about the z-axis. It will not be valid unless the Zeeman Hamiltonian (to which it is meant to be added[1]) is much stronger. The return array will have units of ***Hz***. The dimension of the array will be 2I+1 where I is the spin quantum value associated with the interaction. If the input arguments ***heta*** and ***phi*** are given the returned Hamiltonian is for the orientation at ***theta*** degrees down from the interaction PAS z-axis and ***phi*** degrees over from the interaction PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***degrees***

In GAMMA the second order G Hamiltonian is given by

$$H_Q^{(1)} = -\frac{\xi^2}{2\Omega_o} I_z \{ A_{0,1}^Q A_{0,-1}^Q [4I(I+1) - 8I_z^2 - 1] + A_{0,2}^Q A_{0,-2}^Q [2I(I+1) - 2I_z^2 - 1] \}$$

Return Value:

A matrix.

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
matrix H = G.H1();                 // Here's the 2nd order Guad. Hamiltonian
cout << H;                         // Have a look at the Hamiltonian.
```

See Also: GCC, NGCC, wG

5.15.3 Hsec

Usage:

```
#include <IntG.h>
matrix IntG::Hsec() const
```

Description:

The function ***Hsec*** is used to obtain the sum of the first and second order G Hamiltonians as a perturbation to the Zeeman Hamiltonian. As such, the returned matrix is "secular" and commutes with both $\boldsymbol{F_z}$ and $\boldsymbol{R_z}$. It will be valid in a rotating frame about the z-axis. It will not be valid unless the Zeeman Hamiltonian is much stronger. The return array will have units of ***Hz***. The dimension of the array will be 2I+1 where I is the spin quantum value associated with the interaction.

Return Value:

A matrix.

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
```

1. In the rotating frame the effective Hamiltonian my have all Zeeman contributions removed. Note that the function does not include the 1st order terms, so should be added to the return from the function H0! The function Hsec will do that automatically.

```
matrix H = G.H1();          // Here's the 2nd order Guad. Hamiltonian
cout << H;                  // Have a look at the Hamiltonian.
```

See Also: GCC, NGCC, wG

5.15.4 H

Usage:

```
#include <IntG.h>
matrix IntG::H() const
```

Description:

The function ***H*** is used to obtain the G Hamiltonian. Most likely this will NOT commute with $\boldsymbol{R_z}$. Thus it will be time independent in the laboratory frame (and time dependent in a frame rotating about the z-axis). The return array will have units of ***Hz***. The dimension of the array will be 2I+1 where I is the spin quantum value associated with the interaction.

Return Value:

A matrix.

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
matrix H = G.H1();          // Here's the 2nd order Guad. Hamiltonian
cout << H;                  // Have a look at the Hamiltonian.
```

See Also: GCC, NGCC, wG

5.16 I/O Functions

5.16.1 read

Usage:

void IntG::read(const String& filename, const spin_sys) const
void IntG::read(const String& filename, const spin_sys) const
void IntG::read(const String& filename, const spin_sys) const
void IntG::read(const String& filename, const spin_sys) const

Description:

The function ***delzz*** is used to either obtain or set the interaction G coupling constant. With no arguments the function returns the coupling in Hz. If an argument, ***dz***, is specified then the coupling constant for the interaction is set. It is assumed that the input value of ***dz*** is in units of ***Hz***. The function is overloaded with the name ***delz*** for convenience. Note that setting of ***delzz*** will alter the (equivalent) value of the G coupling ***GCC***/***NGCC*** as well as the G frequency.

Return Value:

Either void or a floating point number, double precision.

Example(s):

```
#include <IntG.h>
IntG G();                    // Empty G interaction.
G.delzz(100000.0);           // Set GCC to 100 KHz.
cout << G.delz ();           // Write coupling constant to std output.
```

See Also: GCC, NGCC, wG

5.16.2 ask

Usage:

#include <IntG.h>
double IntG:: () const
double IntG::delz () const
double IntG::delzz (double dz) const
double IntG::delz (double dz) const

Description:

The function ***delzz*** is used to either obtain or set the interaction G coupling constant. With no arguments the function returns the coupling in Hz. If an argument, ***dz***, is specified then the coupling constant for the interaction is set. It is assumed that the input value of ***dz*** is in units of ***Hz***. The function is overloaded with the name ***delz*** for convenience. Note that setting of ***delzz*** will alter the (equivalent) value of the G coupling ***GCC***/***NGCC*** as well as the G frequency.

Return Value:

Either void or a floating point number, double precision.

Example(s):

```
#include <IntG.h>
IntG G();                    // Empty G interaction.
G.delzz(100000.0);           // Set GCC to 100 KHz.
cout << G.delz ();           // Write coupling constant to std output.
```

See Also: GCC, NGCC, wG

5.16.3 askset

Usage:

```
#include <IntG.h>
double IntG:: () const
double IntG::delz () const
double IntG::delzz (double dz) const
double IntG::delz (double dz) const
```

Description:

The function ***delzz*** is used to either obtain or set the interaction G coupling constant. With no arguments the function returns the coupling in Hz. If an argument, ***dz***, is specified then the coupling constant for the interaction is set. It is assumed that the input value of ***dz*** is in units of ***Hz***. The function is overloaded with the name ***delz*** for convenience. Note that setting of ***delzz*** will alter the (equivalent) value of the G coupling ***GCC***/***NGCC*** as well as the G frequency.

Return Value:

Either void or a floating point number, double precision.

Example(s):

```
#include <IntG.h>
IntG G();                    // Empty G interaction.
G.delzz(100000.0);           // Set GCC to 100 KHz.
cout << G.delz ();           // Write coupling constant to std output.
```

See Also: GCC, NGCC, wG

5.16.4 print

Usage:

```
#include <IntG.h>
ostream& IntG::print (ostream& ostr, int fflag=-1)
```

Description:

The function ***print*** is used to write the interaction G coupling constant to an output stream ***ostr***. An additional flag ***fflag*** is set to allow some control over how much information is output. The default (***fflag !=0***) prints all information concerning the interaction. If ***fflag*** is set to zero only the basis parameters are printed.

Return Value:

The ostream is returned.

Example:

```
#include <IntG.h>
IntG G(2.5, 2.e6, 0.2, 45.7, 15.0);     // Make a G interaction.
cout << G;                              // Write the interaction to standard output.
```

See Also: <<

5.16.5 <<

Usage:

```
#include <IntG.h>
friend ostream& operator << (ostream& out, IntG& G)
```

Description:

The operator << defines standard output for the interaction G coupling constant.

Return Value:

The ostream is returned.

Example:

```
#include <IntG.h>
IntG G(1.5, 3.e5, 0.2);                 // Make a G interaction.
cout << G;                              // Write the interaction to standard output.
```

See Also: print

5.16.6 printSpherical

Usage:

```
#include <IntG.h>
ostream& IntG::print (ostream& ostr, int fflag=-1)
```

Description:

The function ***print*** is used to write the interaction G coupling constant to an output stream ***ostr***. An additional flag ***fflag*** is set to allow some control over how much information is output. The default (***fflag !=0***) prints all information concerning the interaction. If ***fflag*** is set to zero only the basis parameters are printed.

Return Value:

The ostream is returned.

Example:

```
#include <IntG.h>
IntG G(2.5, 2.e6, 0.2, 45.7, 15.0);     // Make a G interaction.
cout << G;                              // Write the interaction to standard output.
```

See Also: <<

5.16.7 printCartesian

Usage:

```
#include <IntG.h>
ostream& IntG::print (ostream& ostr, int fflag=-1)
```

Description:

The function ***print*** is used to write the interaction G coupling constant to an output stream ***ostr***. An additional flag ***fflag*** is set to allow some control over how much information is output. The default (***fflag !=0***) prints all information concerning the interaction. If ***fflag*** is set to zero only the basis parameters are printed.

Return Value:

The ostream is returned.

Example:

```
#include <IntG.h>
IntG G(2.5, 2.e6, 0.2, 45.7, 15.0);     // Make a G interaction.
cout << G;                              // Write the interaction to standard output.
```

See Also: <<

5.17 Description

5.17.1 Overview

A G interaction is the observed effect from the electron cloud surrounding a nucleus responding to an applied magnetic field. The spin itself experiences not only the applied field but also a field from the perturbed electron cloud, the latter field generally opposing the applied field or “shielding” the nucleus. Not only can the shielding contribution be quite large, it is usually orientationally dependent because the surrounding electron cloud is no spherical (due to chemical bonds). In the following discussion we will not be concerned with the isotropic and anti-symmetric parts of the shielding. The former produces measureable chemical shifts whereas the latter is rarely seen. Rather the focus will be on the symmetric rank 2 contribution, that which produces relaxation effects in liquid NMR and orientationally dependent shifts in solids.

5.17.2 Coordinate Systems

We will shortly concern ourselves with the mathematical representation of G interactions, in particular their description in terms of spatial and spin tensors. The spatial tensors will be cast in both Cartesian and spherical coordinates and we will switch between the two when convenient. The figure below relates the orientation angles theta and phi to the standard right handed coordinate system in all GAMMA treatments.

Cartesian and Spherical Coordinate Systems

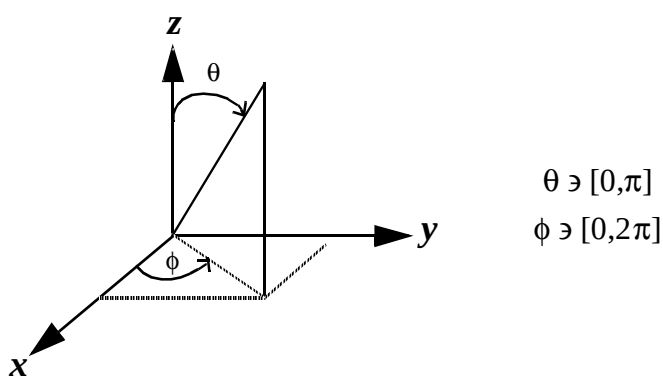


Figure 19-26 **The right handed Cartesian axes with the spherical angles and radius.**

5.17.3 Internal Structure

The internal structure of class **IntG** contains the quantities listed in the following table (names shown are also internal).

Table 3-1: Internal Structure of Class IsoG

Name	Description	Type	Name	Description	Type
AISO	Isotropic G Value	double	THETA	Orientation Angle	double
DELZZ	Spatial Tensor δ_{zz}	double	Asph	Spatial Tensor Values	complex*
ETA	Spatial Tensor η	double	Tsph	Spin Tensor Values	matrix*
PHI	Orientation Angle	double			

Note that since the spin angular momentum of an electron is ***I***=1/2, the spin tensor components will reside in a spin Hilbert space of dimension 2.

The three values ***AISO***, ***DELZZ,*** and ***ETA*** are all that is required to specify the G interaction strength and may be used to represent the G spatial tensor. However, in GAMMA the values of ***AISO*** and ***DELZZ*** are factored out of the spatial tensor such that all rank two interactions (such as the G interaction) have the same spatial tensor scaling.

The two angles ***THETA*** and ***PHI*** indicate how the G interaction is aligned relative to the interaction principal axes (PAS). These are one in the same as the angles shown in Figure 19-26 when the Cartesian axes are those of the PAS with the origin vaguely being the center of the nucleus. These are intrinsically tied into the values in the array ***Asph***.

There are five values in the complex vector ***Asph*** and these are irreducible spherical components of the G spatial tensor oriented at angle ***THETA*** down from the PAS z-axis and over angle ***PHI*** from the PAS x-axis. Note that these 5 values are not only orientation dependent, they are also ***ETA*** dependent. If either of the three the interaction values {***ETA***, ***THETA, PHI***} are altered these components will all be reconstructed. The values in ***Asph*** will be scaled such that they are consistent with other rank 2 spatial tensors in GAMMA which are independent of the interaction type.

Structure of a Variable of Class IntG

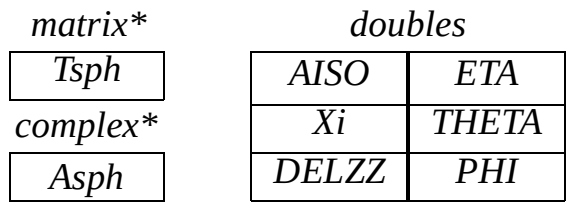


Figure 19-27 **Depiction of class IntG contents, i.e. what each GAMMA defined G interaction contains. The values of both Xi and DELZZ are maintained for convenience (one being deduced from the other if the field is specified). Tsph will contain 5 matrices which dimension will be 2*I+1 and Asph will contain 5 complex numbers.**

The vector of matrices relates to the sperical spin tensor components according to:

Tsph:	[0]	[1]	[2]	[3]	[4]
$T^G_{2,m}$:	$T^G_{2,0}$	$T^G_{2,1}$	$T^G_{2,-1}$	$T^G_{2,2}$	$T^G_{2,-2}$

and the vector of complex numbers relate to the GAMMA normalized spherical spatial tensor components via

Asph:	[0]	[1]	[2]	[3]	[4]
$A_{2,m}$:	$A_{2,0}$	$A_{2,1}$	$A_{2,-1}$	$A_{2,2}$	$A_{2,-2}$

5.17.4 Classical G Treatment

A chemical shift is the observed effect from the electron cloud surrounding a nucleus responding to an applied magnetic field. The spin itself experiences not only the applied field but also a field from the perturbed electron cloud, the latter field generally opposing the applied field or "shielding" the nucleus. We can write this latter "induced" field in terms of the applied field, $\vec{\boldsymbol{B}}_o$, as

$$\vec{\boldsymbol{B}}_{induced} = -\hat{\sigma} \bullet \vec{\boldsymbol{B}}_o$$

where $\hat{\sigma}$ is the chemical g tensor, a 3x3 array in Cartesian space, and the $\vec{\boldsymbol{B}}$'s vectors in Cartesian space. In matrix form this is simply[1]

$$\begin{bmatrix} B_{ind,\,x} \\ B_{ind,\,y} \\ B_{ind,\,z} \end{bmatrix} = -\begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{bmatrix}_i \bullet \begin{bmatrix} B_{0x} \\ B_{0y} \\ B_{0z} \end{bmatrix},$$

the induced field depends on the applied field strength, the applied field orientation, and the surrounding electron cloud. Note that $\vec{\boldsymbol{B}}_{induced}$ will not necessarily be co-linear with the applied field. Of course, every nuclear spin will have its own associated chemical g tensor. The classical interaction energy between this induced field and a nuclear spin is

$$E^G = -\vec{\mu}_e \bullet \vec{\boldsymbol{H}}_{effective} = -\vec{\mu}_e \bullet \frac{\hat{\boldsymbol{G}}}{g_e} \bullet \vec{\boldsymbol{H}}$$

where $\vec{\mu}_e$ is the electron magnetic moment, E the energy, and superscript G used to denote an electron G interaction.

5.17.5 Quantum Mechanical Formulation

The associated G interaction Hamiltonian is obtained from substitution of $-\beta\vec{\boldsymbol{S}} = \frac{h\gamma_e}{g_e}\vec{\boldsymbol{S}}$ for $\frac{\vec{\mu}}{g_e}$.

$$\boldsymbol{H}^G = \beta\vec{\boldsymbol{S}} \bullet \hat{\boldsymbol{G}} \bullet \vec{\boldsymbol{H}} = \frac{-h\gamma_e\vec{\boldsymbol{S}}}{g_e} \bullet \hat{\boldsymbol{G}} \bullet \vec{\boldsymbol{H}}\,; \qquad (39\text{-}1)$$

In matrix form this equation looks like

1. Note that the effect of the G tensor is to alter the overall external field which the electron experiences. This is clearly seen from the product $\hat{\boldsymbol{G}} \bullet \vec{\boldsymbol{H}}$ which produces an effective field vector for the electron.

$$\boldsymbol{H}^G = \beta\begin{bmatrix}\boldsymbol{S}_x & \boldsymbol{S}_y & \boldsymbol{S}_z\end{bmatrix} \bullet \begin{bmatrix} g_{xx} & g_{xy} & g_{xz} \\ g_{yx} & g_{yy} & g_{yz} \\ g_{zx} & g_{zy} & g_{zz}\end{bmatrix} \bullet \begin{bmatrix}\boldsymbol{H}_x \\ \boldsymbol{H}_y \\ \boldsymbol{H}_z\end{bmatrix}. \tag{39-2}$$

Taking the magnitude of the applied field out, equation (39-1) is simply

$$\boldsymbol{H}^G = \beta H \sum_u^{axes} \sum_v^{axes} \langle 1|\vec{\boldsymbol{S}}|u\rangle\langle u|\hat{\boldsymbol{G}}|v\rangle\langle v|\vec{\boldsymbol{H}}_n|1\rangle \tag{39-3}$$

with $u, v \in \{x, y, z\}$ and $\vec{\boldsymbol{H}}_n$ a normalized magnetic field vector in the direction of the applied field.

5.17.6 Cartesian Tensor Formulation

Equation (39-2) can also be rearranged to produce an equation involving two rank 2 tensors by taking the dyadic product of the vectors $\vec{\boldsymbol{S}}$ and $\vec{\boldsymbol{H}}_n$.

$$\boldsymbol{H}^G = \beta H \sum_u^{axes} \sum_v^{axes} \langle u|\hat{\boldsymbol{G}}|v\rangle\langle v|\vec{\boldsymbol{H}}_n|1\rangle\langle 1|\vec{\boldsymbol{S}}|u\rangle = \beta H \sum_u^{axes} \sum_v^{axes} \langle u|\hat{\boldsymbol{G}}|v\rangle\langle v|\vec{\boldsymbol{H}}_n\vec{\boldsymbol{S}}|u\rangle$$

The dyadic product to produce $\vec{\boldsymbol{H}}_n\vec{\boldsymbol{S}}$ is explicitly done *via*

$$\begin{bmatrix}H_{nx} \\ H_{ny} \\ H_{nz}\end{bmatrix} \bullet \begin{bmatrix}\boldsymbol{S}_x & \boldsymbol{S}_y & \boldsymbol{S}_z\end{bmatrix} = \begin{bmatrix} H_{nx}\boldsymbol{S}_x & H_{nx}\boldsymbol{S}_y & H_{nx}\boldsymbol{S}_z \\ H_{ny}\boldsymbol{S}_x & H_{ny}\boldsymbol{S}_y & H_{ny}\boldsymbol{S}_z \\ H_{nz}\boldsymbol{S}_x & H_{nz}\boldsymbol{S}_y & H_{nz}\boldsymbol{S}_z\end{bmatrix}.$$

The G interaction Hamiltonian can thus be formulated as a scalar product of two rank 2 tensors. Letting $\hat{\boldsymbol{T}}^G = \vec{\boldsymbol{H}}_n\vec{\boldsymbol{S}}$, we have

$$\boldsymbol{H}^G = \beta H \hat{\boldsymbol{G}} \bullet \hat{\boldsymbol{T}}^G = \beta H \sum_u^{axes} \sum_v^{axes} \langle u|\hat{\boldsymbol{G}}|v\rangle\langle v|\hat{\boldsymbol{T}}^G|u\rangle$$

5.17.7 Spherical Tensor Formulation

The previous equation, , can also be rewritten in term of irreducible spherical components rather than in terms of the Cartesian components using the substitution

$$\sum_{l=0}^{2} \sum_m^{\pm l} (-1)^m g_{l-m} \hat{\boldsymbol{T}}^G_{lm} = \sum_u^{axes} \sum_v^{axes} \langle u|\hat{\boldsymbol{G}}|v\rangle\langle v|\hat{\boldsymbol{T}}^G|u\rangle \tag{39-4}$$

where g_{l-m} are spherical components of the tensor $\hat{\boldsymbol{G}}$. The result is

$$\boldsymbol{H}^G = \beta H \sum_{l=0}^{2} \sum_{m}^{\pm l} (-1)^m g_{l-m} \bullet \hat{\boldsymbol{T}}^G_{lm} \tag{39-5}$$

and we can expand the summation over the different ranks.

$$\boldsymbol{H}^G = \beta H \left| g_{0,0} \boldsymbol{T}^G_{0,0} + \sum_{m}^{\pm 1} (-1)^m g_{1,-m} \boldsymbol{T}^G_{1,m} + \sum_{m}^{\pm 2} (-1)^m g_{2,-m} \boldsymbol{T}^G_{2,m} \right|$$

In other words we now have

$$\boldsymbol{H}^G = \boldsymbol{H}^{GI} + \boldsymbol{H}^{GU} + \boldsymbol{H}^{GA}. \tag{39-6}$$

There is good reason to separate these terms. The rank 0 component of the G Hamiltonian is rotationally invariant and called the isotropic G Hamiltonian. In liquid EPR it will dictate where the electron resonance occurs. The rank 2 part is call the chemical G Anisotropy Hamiltonian. In liquid systems this Hamiltonian averages to zero and thus not affect observed g values. It will contribute to relaxation of the system. On the other hand, in solid systems this component does not average away and will partially determine peak shapes in powder averages. The rank 1 component is the antisymmetric part of the G Hamiltonian. Since the antisymmetric part of the G tensor is difficult to measure, this part of the G Hamiltonian is usually assumed small and neglected.

The isotropic component ($l = 0$) of the G Hamiltonian is thus written

$$\boldsymbol{H}^{GI}(AAS) = \beta H g_{0,0} \boldsymbol{T}^G_{0,0}, \tag{39-7}$$

the antisymmetric component ($l = 1$) of the G Hamiltonian is

$$\boldsymbol{H}^{GU}(AAS) = \beta H \sum_{m}^{\pm 1} (-1)^m A^G_{1,-m}(i, AAS) \bullet \boldsymbol{T}^G_{1,m}(i, AAS) \quad , \tag{39-8}$$

and the anisotropic component ($l = 2$) of the G Hamiltonian is

$$\boldsymbol{H}^{GA}_i(AAS) = \beta H \sum_{m}^{\pm 2} (-1)^m g_{2,m}(AAS) \bullet \boldsymbol{T}^G_{2,-m}(AAS) \tag{39-9}$$

5.17.8 G Interaction Spherical Tensor Spin Components

We can obtain the 9 irreducible spherical components of the G rank 2 "spin" tensor[1] directly from the Cartesian components, $\langle v|\hat{\boldsymbol{T}}|u\rangle$, as indicated in GAMMA Class Documentation on Spin Tensors. These are

$$\boldsymbol{T}^{G}_{l,\,m},$$

where G signifies the electron G interaction. The tensor index l spans the rank: $l \in [0, 2]$ while the tensor index m spans l: $m \in [-l, l]$ The nine formulas for these quantities a listed in the following figure where the field components are those of the normalized field vector $\vec{\boldsymbol{H}}_n$.[2]

G Rank 2 Irreducible Spherical Spin-Space Tensor Components

$$T^{G}_{0,\,0} = \frac{-1}{\sqrt{3}}\left[\boldsymbol{S}_z H_z + \frac{1}{2}(\boldsymbol{S}_+ H_- + \boldsymbol{S}_- H_+)\right] = \frac{-1}{\sqrt{3}}\vec{\boldsymbol{S}} \bullet \vec{\boldsymbol{H}}_n$$

$$T^{G}_{1,\,0} = \frac{-1}{2\sqrt{2}}[\boldsymbol{S}_+ H_- - \boldsymbol{S}_- H_+] \qquad T^{G}_{1,\,\pm 1} = \frac{-1}{2}[\boldsymbol{S}_\pm\, H_z - \boldsymbol{S}_z H_\pm\,]$$

$$T^{G}_{2,\,0} = \frac{1}{\sqrt{6}}[3\boldsymbol{S}_z H_z - (\vec{\boldsymbol{S}} \bullet \vec{\boldsymbol{H}}_n)]$$

$$T^{G}_{2,\,\pm 1} = \mp\frac{1}{2}[\boldsymbol{S}_{\pm\,+} H_z + \boldsymbol{S}_z H_\pm\,] \qquad T^{G}_{2,\,\pm 2} = \frac{1}{2}[\boldsymbol{S}_\pm\, H_\pm\,]$$

Figure 19-28 **The rank 2 spin-space tensor components for the electron G interaction.**

For $\vec{\boldsymbol{H}} = H\vec{\boldsymbol{H}}_n$, the matrix form of these tensor components are shown in the following figure in the single electron spin Hilbert space. The spin index has been omitted, the field components are those of the normalized vector $\vec{\boldsymbol{H}}_n$.

1. Due to the nature of the G interaction, the rank 2 tensor treatment produces a "spin" tensor $T^{G}_{l,\,m}$ which contains spatial components, namely the magnetic field vector. As a result, care must be used when performing spatial rotations on G tensors. Any spatial rotations must involve rotations of both G and $\boldsymbol{T}$
2. For these formulae, it is important to note that it is the second component in the composite spin/space tensor which is set to the normalized magnetic field vector $\vec{\boldsymbol{H}}_n$, although we might just as well have used the first vector instead. The difference is that the $l = 1$ equations would then appear of opposite sign from those given here. Our field vector has be set to point along the positive z-axis in the laboratory frame.

General G Spin-Space Tensor Components Matrix Representations

$$T^{(2)}_{0,0} = \frac{-1}{2\sqrt{3}}\begin{bmatrix} H_z & H_- \\ B_+ & -B_z \end{bmatrix} \qquad T^{(2)}_{1,0} = \frac{-1}{2\sqrt{2}}\begin{bmatrix} 0 & H_- \\ -H_+ & 0 \end{bmatrix} \qquad T^{(2)}_{1,-1} = \frac{-1}{2}\begin{bmatrix} -H_-/2 & 0 \\ H_z & H_-/2 \end{bmatrix} \qquad T^{(2)}_{1,1} = \frac{-1}{2}\begin{bmatrix} -H_+/2 & H_z \\ 0 & H_+/2 \end{bmatrix}$$

$$T^{(2)}_{2,0} = \frac{1}{2\sqrt{6}}\begin{bmatrix} 2H_z & -H_- \\ -H_+ & -2H_z \end{bmatrix} \quad T^{(2)}_{2,-1} = \frac{1}{2}\begin{bmatrix} H_-/2 & H_z \\ 0 & -H_-/2 \end{bmatrix} \quad T^{(2)}_{2,1} = \frac{-1}{2}\begin{bmatrix} H_+/2 & 0 \\ H_z & -H_+/2 \end{bmatrix} \quad T^{(2)}_{2,-2} = \frac{1}{2}\begin{bmatrix} 0 & 0 \\ H_- & 0 \end{bmatrix} \quad T^{(2)}_{2,2} = \frac{1}{2}\begin{bmatrix} 0 & H_+ \\ 0 & 0 \end{bmatrix}$$

Figure 19-29 **A general matrix representation of the rank 2 spin-space tensor components for the electron G interaction. The spin Hilbert space dimension if 2 due to the election having spin angular momentum of 1/2. The direction of the applied field is arbitrary, however the field vector is normalized in this formulation.**

The matrix representation of these nine tensor components will depend upon the matrix representations of the individual spin operators from which they are constructed[1]. These in turn depend upon the fact that electrons are spin 1/2 particles. Their G tensor components are, in the previous figure, expressed in matrix form in the default product basis of GAMMA. In this case the spin index is implicit.

The raising an lowering components of the field vector are defined in the standard fashion, namely $H_\pm = H_x \pm iH_y$. The simplest situation occurs when magnetic field points along the positive z-axis, $\vec{\boldsymbol{H}}_n = \hat{\boldsymbol{k}}$, *i.e.* these spin-space tensors are written in the laboratory frame. Then, the (normalized) field vector simplifies, $H_z = 1$ and $H_x = H_y = H_\pm = 0$. The applicable equations for the shielding space-spin tensors are then as follows.

G Spin-Space Tensor Components, H Along z-Axis

$$T^G_{0,0}(i) = \frac{-1}{\sqrt{3}}\boldsymbol{S}_{iz} \qquad T^G_{1,0}(i) = 0 \qquad T^G_{1,\pm1}(i) = \frac{-1}{2}\boldsymbol{S}_{i\pm}$$

$$T^G_{2,\pm1}(i) = \mp\frac{1}{2}\boldsymbol{S}_{i\pm} \qquad T^G_{2,0}(i) = \frac{2}{\sqrt{6}}\boldsymbol{S}_{iz} \qquad T^G_{2,\pm2}(i) = 0$$

Figure 19-30 **The rank 2 spin-space tensor components for the electron G interaction when the field vector is oriented along the +z axis in the laboratory frame.**

For $\vec{\boldsymbol{H}} = H\vec{\boldsymbol{H}}_n$ along the positive z-axis, the matrix form of these tensor components are shown in the following figure[2] (in the single spin Hilbert space).

G Spin-Space Tensor Components Matrix Representations, H on z-Axis

Figure 19-31 **A general matrix representation of the rank 2 spin-space tensor components for the**

1. Note that the spin tensors are invariably constructed in the laboratory coordinate system. Here the z-axis corresponds to the direction of the spectrometer static magnetic field and the coordinate system is right-handed.
2. The GAMMA program which produced these matrix representations can be found at the end of this Chapter, sosix Rank2SS_SpinT.cc.

$$T^G_{0,0} = \frac{-1}{2\sqrt{3}}\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix} \qquad T^G_{1,0} = \begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix} \qquad T^G_{1,-1} = \frac{-1}{2}\begin{bmatrix}0 & 0\\ 1 & 0\end{bmatrix} \qquad T^G_{1,1} = \frac{-1}{2}\begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}$$

$$T^G_{2,0} = \frac{1}{\sqrt{6}}\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix} \quad T^G_{2,1} = \frac{-1}{2}\begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix} \quad T^G_{2,-1} = \frac{1}{2}\begin{bmatrix}0 & 0\\ 1 & 0\end{bmatrix} \quad T^G_{2,-2} = \begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix} \quad T^G_{2,2} = \begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix}$$

electron G interaction when the field vector is oriented along the +z axis in the laboratory frame. The spin Hilbert space dimension if 2 due to the election having spin angular momentum of 1/2.

We must very careful in using these single spin rank 2 G tensors of this type because they contain both spatial and spin components. If we desire to express the G Hamiltonian relative to a particular set of axes we must insure that both the spatial tensor and the “spin” tensor are expressed in the proper coordinates. The spatial tensor alone cannot be rotated as it rotates only part of the spatial components[1]. It is improper to rotate this tensor in spin space because it also rotates spatial variables. Furthermore, note that **these rank 2 components are not the same as the rank 1 tensor components**.

5.17.9 General Rank 2 Spatial Tensor Components

The 9 irreducible spherical components of a rank 2 spatial tensor, $A^{(2)}_{lm}$, are related to its Cartesian components by the following formulas[2].

$$\begin{gathered}
A_{0,0} = \frac{-1}{\sqrt{3}}[A_{xx} + A_{yy} + A_{zz}] = \frac{-1}{\sqrt{3}}Tr\{A\} \\
A_{1,0} = \frac{-i}{\sqrt{2}}[A_{xy} - A_{yx}] \qquad A_{1,\pm 1} = \frac{-1}{2}[A_{zx} - A_{xz} \pm i(A_{zy} - A_{yz})] \\
A_{2,0} = \sqrt{6}[3A_{zz} - (A_{xx} + A_{yy} + A_{zz})] = \sqrt{6}[3A_{zz} - Tr\{A\}] \\
A_{2,\pm 1} = \mp\frac{1}{2}[A_{xz} + A_{zx} \pm i(A_{yz} + A_{zy})] \qquad A_{2,\pm 2} = \frac{1}{2}[A_{xx} - A_{yy} \pm i(A_{xy} + A_{yx})]
\end{gathered} \tag{39-10}$$

Again the subscript l spans the rank as l = [0, 2], and the subscript m spans +/- l, m = [-l, l].

In this G interaction treatment, we then have the components $g_{l,m}$ as indicated in equation (39-5). Thus, the irreducible spherical tensor components can be obtained by substituting the Cartesian elements of the G tensor, $\hat{\boldsymbol{G}}$, into equations (39-10).

1. See the discussion in Mehring
2. See the GAMMA Class Documentaion on Rank 2 Interactions.

$$
\begin{aligned}
g_{0,0} &= \frac{-1}{\sqrt{3}}[g_{xx}+g_{yy}+g_{zz}] = \frac{-1}{\sqrt{3}}Tr\{\hat{\boldsymbol{G}}\} \\
g_{1,0} &= \frac{-i}{\sqrt{2}}[g_{xy}-g_{yx}] \qquad g_{1,\pm1} = \frac{-1}{2}[g_{zx}-g_{xz}\pm i(g_{zy}-g_{yz})] \\
g_{2,0} &= \sqrt{6}[3g_{zz}-(g_{xx}+g_{yy}+g_{zz})] = \sqrt{6}[3g_{zz}-Tr\{\hat{\boldsymbol{G}}\}] \\
g^{G}_{2,\pm1} &= \mp\frac{1}{2}[g_{xz}+g_{zx}\pm i(g_{yz}+g_{zy})] \qquad g_{2,\pm2} = \frac{1}{2}[g_{xx}-g_{yy}\pm i(g_{xy}+g_{yx})]
\end{aligned}
\tag{39-11}
$$

However, it is more convenient to rewrite the general rank two Cartesian tensor in terms of a sum over tensors of ranks 0 through 2 as follows,

$$
\hat{A} = \begin{bmatrix} A_{xx} & A_{xy} & A_{xz} \\ A_{yx} & A_{yy} & A_{yz} \\ A_{zx} & A_{zy} & A_{zz} \end{bmatrix} = \underbrace{A_{iso}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Rank 0}} + \underbrace{\begin{bmatrix} 0 & \alpha_{xy} & \alpha_{xz} \\ -\alpha_{xy} & 0 & \alpha_{yz} \\ -\alpha_{xz} & -\alpha_{yz} & 0 \end{bmatrix}}_{\text{Rank 1}} + \underbrace{\begin{bmatrix} \delta_{xx} & \delta_{xy} & \delta_{xz} \\ \delta_{yx} & \delta_{yy} & \delta_{yz} \\ \delta_{zx} & \delta_{zy} & \delta_{zz} \end{bmatrix}}_{\text{Rank 2}} \tag{39-12}
$$

where

$$
A_{iso} = \frac{1}{3}Tr\{\hat{A}\} \qquad \alpha_{xy} = \frac{1}{2}(A_{xy}-A_{yx}) \qquad \delta_{xy} = \frac{1}{2}(A_{xy}+A_{yx}-2A_{iso}) \tag{39-13}
$$

The rank 0 part is isotropic (scalar), the rank 1 part is antisymmetric and traceless, and the rank 2 part traceless and symmetric. We shall apply this same nomeclature to our G spatial tensor to produce

$$
\hat{G} = \begin{bmatrix} g_{xx} & g_{xy} & g_{xz} \\ g_{yx} & g_{yy} & g_{yz} \\ g_{zx} & g_{zy} & g_{zz} \end{bmatrix} = g_{iso}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & \alpha_{xy} & \alpha_{xz} \\ -\alpha_{xy} & 0 & \alpha_{yz} \\ -\alpha_{xz} & -\alpha_{yz} & 0 \end{bmatrix} + \begin{bmatrix} \delta_{xx} & 0 & 0 \\ 0 & \delta_{yy} & 0 \\ 0 & 0 & \delta_{zz} \end{bmatrix}. \tag{39-14}
$$

where

$$
g_{iso} = \frac{1}{3}Tr\{\hat{G}\} \qquad \alpha_{xy} = \frac{1}{2}(g_{xy}-g_{yx}) \qquad \delta_{xy} = \frac{1}{2}(g_{xy}+g_{yx}-2g_{iso}) \tag{39-15}
$$

5.17.10 Unscaled G Spherical Spatial Tensor PAS Components

As with any rank 2 spatial tensor, the G spatial tensor can be specified in its principal axis system, the set of axes in which the irreducible rank 2 component is diagonal[1]. The G tensor values are experimentally determined in the tensor principal axes. Employing (39-12) in the case where the ir-

1. The principal axis system is set such that $|\delta_{zz}|\geq|\delta_{yy}|\geq|\delta_{xx}|$. The orientation of the x and y axes are inconsequential if η is zero.

reducible rank 2 component is diagonal,

$$\hat{G}(PAS) = g_{iso}\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix} + \begin{bmatrix}0&\alpha_{xy}&\alpha_{xz}\\-\alpha_{xy}&0&\alpha_{yz}\\-\alpha_{xz}&-\alpha_{yz}&0\end{bmatrix} + \begin{bmatrix}\delta_{xx}&0&0\\0&\delta_{yy}&0\\0&0&\delta_{zz}\end{bmatrix}$$

where (39-15) still applies.

Rank 2 spatial tensors are also commonly specified in their principal axis system by the three components; the isotropic value A_{iso}, the anisotropy ΔA, and the asymmetry η. These are generally given by

$$A_{iso} = \frac{1}{3}Tr\{A\}\ , \qquad \Delta A = A_{zz} - \frac{1}{2}(A_{xx} + A_{yy}) = \frac{3}{2}\delta_{zz} \qquad \eta = (\delta_{xx} - \delta_{yy})/\delta_{zz}$$

A set of Euler angles $\{\alpha, \beta, \gamma\}$ is normally also given to relate the spatial tensor principle axes to another coordinate system. For the g-tensor we have

$$\hat{G}(PAS) = g_{iso}\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix} + \begin{bmatrix}0&\alpha_{xy}&\alpha_{xz}\\-\alpha_{xy}&0&\alpha_{yz}\\-\alpha_{xz}&-\alpha_{yz}&0\end{bmatrix} + \delta_{zz}\begin{bmatrix}-\frac{1}{2}(1-\eta)&0&0\\0&-\frac{1}{2}(1+\eta)&0\\0&0&1\end{bmatrix} \tag{39-16}$$

Note that δ_{zz} is NOT equivalent to g_{zz} and that η is NOT equivalent to $(g_{xx} - g_{yy})/g_{zz}$. The irreducible spherical elements of the G tensor, $g_{l,m}$, in the principal axis system are, by placement of (39-16) into (39-10),

$$g_{0,0}(PAS) = -\sqrt{3}g_{iso}$$

$$g_{1,0}(PAS) = -\frac{i}{\sqrt{2}}[g_{xy} - g_{yx}] \qquad g_{1,\pm1}(PAS) = -\frac{1}{2}[(g_{zx} - g_{xz}) \pm i(g_{zy} - g_{yz})]$$

$$g_{2,0}(PAS) = \sqrt{3/2}\delta_{zz} \qquad g_{2,1}(PAS) = g_{2,-1}(PAS) = 0$$

$$g_{2,2}(PAS) = g_{2,-2}(PAS) = \frac{1}{2}\delta_{zz}\eta$$

and these values should be equivalent to those given in (39-11) on page 5-340.

5.17.11 Scaled G Spherical Spatial Tensor PAS Components

Throughout GAMMA, we desire all irreducible spherical rank 2 spatial components to be scaled so as they are independent of the particular interaction. To do so, we adjust them to be as similar to normalized spherical harmonics as possible. Thus, we here scale the G irreducible rank 2 spatial tensor so that the $2, 0$ component will have the same magnitude as the $m = 0$ rank two spherical harmonic when the two spherical angles are set to zero. Our "normalization" factor "X" is obtained by

$$A_{2,0}(\theta,\varphi)\Big|_{\theta=\varphi=0} = X^G \bullet g_{2,0}(\theta,\varphi)\Big|_{\theta=\varphi=0} = Y_{2,0}(\theta,\varphi)\Big|_{\theta=\varphi=0} = \sqrt{5/(4\pi)}$$

Using $g_{2,0}(PAS) = \sqrt{3/2}\delta_{zz}$ we thus define the GAMMA G anisotropy spatial tensor to be scaled such that its normalized spherical components are given by

$$A_{l,m} = \sqrt{5/(6\pi)}\delta_{zz}^{-1} g_{l,m} \tag{39-17}$$

and the irreducible rank 2 components are given in the next figure.

GAMMA Normalized Rank 2 Spatial Tensor PAS Components

$$A_{2,0}(PAS) = \sqrt{\frac{5}{4\pi}} \qquad A_{2,\pm1}(PAS) = 0 \qquad A_{2,\pm2}(PAS) = \sqrt{\frac{5}{24\pi}}\eta$$

Figure 19-32 **Generic irreducible rank 2 spatial tensor components as defined in GAMMA. These are shown in the principle axis system of the tensor and scaled to coincide with normalized spherical harmonics.**

The scaling factor $\sqrt{5/(6\pi)}\delta_{zz}^{-1}$ which was multiplied into the spherical G tensor components will subsequently be compensated for in the G interaction by use of a G interaction constant. The Anisotropic G Hamiltonian given in equation (23) becomes

$$\boldsymbol{H}^{GA} = \beta H \sum_{m}^{\pm1} (-1)^m g_{2-m} \bullet \hat{\boldsymbol{T}}_{2m}^G = \beta H \delta_{zz} \sqrt{\frac{6\pi}{5}} \sum_{m}^{\pm l} (-1)^m A_{2-m} \bullet \hat{\boldsymbol{T}}_{lm}^G \tag{39-18}$$

5.17.12 G Interaction Constant

In GAMMA, since we have defined our generic spatial and spin tensors to be scaled independent of the type of interaction, we use an interaction constant as a scaling factor when formulating Hamiltonians. The G anisotropic Hamiltonian may be produced from

$$\boldsymbol{H}^{GA} = \xi^G \sum_{m}^{\pm1} (-1)^m A_{2,-m} \boldsymbol{T}_{2,m}^G = \beta H \delta_{zz} \sqrt{\frac{6\pi}{5}} \sum_{m}^{\pm1} (-1)^m A_{2,-m} \boldsymbol{T}_{2,m}^G \tag{39-19}$$

so evidently

$$\xi^G = \beta H \delta_{zz} \sqrt{\frac{6\pi}{5}} \tag{39-20}$$

Such interaction constants are not very common in the literature (except with regards to some papers treating relaxation in liquid NMR) and thus not intuitive to many GAMMA users. So, one simply needs to be aware of the relationships between the interaction constant and any commonly used G tensor definitions. Most EPR literature retain the G tensor in Cartesian components, whereas in GAMMA we (internally) work with the spherical components consistently across the magnetic resonance interaction types. Perhaps the only quantity worthy of mention is the δ_{zz}, the G anisotropy . This is readily related to the typical G tensor Cartesian components.

$$\delta_{zz} = g_{zz} - g_{iso} = g_{zz} - \frac{1}{3} Tr\{\hat{G}\}$$

5.17.13 Spatial Tensor Rotations

We can express the spatial tensor components $A_{l,m}$ relative to any arbitrary axis system (AAS) by a rotation from the principal axes to the new axes *via* the formula

$$A_{l,m}(AAS) = \sum_{m'}^{\pm l} D^l_{mm'}(\Omega) A_{l,m'}(PAS) \tag{39-21}$$

where $D^l_{mm'}$ are the rank l Wigner rotation matrix elements and Ω the set of three Euler angles which relate the principal axes of the spatial tensor to the arbitrary axes[1].

5.17.14 G Hamiltonian Rotations

The G Hamiltonian can now be expressed with respect to any arbitrary axes through use of its spherical tensor components and the previous equation. Our Hamiltonian in spherical tensor form is

$$\boldsymbol{H}^G = \boldsymbol{H}^{GI} + \boldsymbol{H}^{GU} + \boldsymbol{H}^{GA} = \beta H g_{0,0} \boldsymbol{T}^G_{0,0} + \boldsymbol{H}^{GU} + \xi^G \sum_{m}^{\pm 2} (-1)^m A_{2,-m} \boldsymbol{T}^G_{2,m}$$

Negletcting the antisymmetric component and recalling that the isotropic component is rotationally

1. In this instance, i.e. the treatment of an electron G interaction, we must be careful to express the elements $\boldsymbol{T}^G_{l,-m}$ in the same axis system as $A_{l,m}$. When A is rotated in space, so must be $\boldsymbol{T}^G$. Essentially, the field vector changes relative to any new coordinate system when constructing $\boldsymbol{T}^G$. In other words, when $A_{l,m}$ is represented in its PAS (normally thought of as $\theta = \varphi = 0$) it does NOT necessarily see the externally applied field point along +z since the latter is defined in the laboratory frame whereas the former is set in an internal (electron cloud fixed) frame.

invarient we obtain, for an arbitrary axis system

$$\boldsymbol{H}^G(AAS) = \beta H g_{0,0}\boldsymbol{T}^G_{0,0} + \xi^G \sum_{m}^{\pm 2} (-1)^m A_{2,-m}(AAS)\boldsymbol{T}^G_{2,m}$$

$$= \beta H g_{iso}\hat{\vec{\boldsymbol{S}}} \bullet \hat{\vec{\boldsymbol{H}}}_n + \xi^G \sum (-1)^m A_{2,-m}(AAS)\boldsymbol{T}^G_{2,m}$$

which becomes, if the system is related to the laboratory frame in which the static external field is pointed along +z,

$$\boldsymbol{H}^G(AAS) = \beta H g_{iso}\boldsymbol{S}_z + \xi^G \sum_{m}^{\pm 2} (-1)^m A_{2,-m}(AAS)\boldsymbol{T}^G_{2,m}$$

5.17.15 G Hamiltonian Units

At this point it is evident that the Hamiltonian has units which are dictated by the factor

$$\beta H$$

This factor occurs in both isotropic and anisotropic terms. The G tensor is taken to be unitless and the units of angular momentum from the spin term are considered included in this factor. The value of the Bohr magneton β is

$$\beta = 9.2741 \times 10^{-21} \text{erg-}G^{-1}$$

and H is typicall specified in units of Gauss. Thus βH as shown will have energy units (ergs). We can readily convert to frequency units using h.

For a free electron where $g_e = 2.0023193 = g_{iso}$, the resonance frequency (the transition between $\boldsymbol{S}_z = \pm\frac{1}{2}$) in a 3000 G field will be given by

$$\omega_e = \frac{g_e \beta H}{h} = \frac{(2.0023193)(9.2741 \times 10^{-21} erg - G^{-1})(3000G)}{6.6262 \times 10^{-27} erg - s - cycle^{-1}} =$$

$$= (2.0023193)(1.3996 \times 10^6 Hz - G^{-1})(3000G) = 8.4074 GHz$$

Typical isotropic g factors are larger than that of a free electron so that a higher frequency will be required at any set field. However, most ESR spectrometers operate in CW mode where the frequency is set and the field is swept. As a result it is better to think that at a specified frequency most electrons resonance at a lower field than does a free electon.

5.17.16 The Anisotropic G Hamiltonian

The **G** tensor orientation will affect the observed electron resonance frequency. Unlike isotropic chemical shifts in NMR, the isotropic (rank 0) contribution to **G** is normally NOT included with the Zeeman Hamiltonian. Furthermore, the anti-symmetric (rank 1) contribution to **G** is rarely treated. The symmetric rank 2 contribution to the **G** interaction, that which we are primarily concerned with in class IntG, produces the following amisotropic Hamiltonian[1].

$$\boldsymbol{H}^{GA} = \xi^G \sum_m^{\pm 2} (-1)^m A_{2,-m} \bullet \boldsymbol{T}^G_{2,m} \qquad \xi^G = \beta H \delta_{zz} \sqrt{\frac{6\pi}{5}}$$

The reader should note normally the spin tensors, $T^G_{2,m}$, are specified in the laboratory frame where the applied magnetic field is along the +z axis. When that is true the $T^G_{2,\pm 2}$ terms are zero and the summation need only be taken over $m = 0, \pm 1$.

$$\boldsymbol{H}^{GA}(LAB) = \xi^G \sum_m^{\pm 1} (-1)^m A_{2,-m}(LAB) \bullet \boldsymbol{T}^G_{2,m}(LAB)$$

Furthermore, if we orient the spatial tensor principal axis system (PAS) to coincide with the laboratory axes, the anisotropic contribution to the G Hamiltonian is given by a relatively simple formula because both the $A^G_{2,\pm 1}$ terms are zero as well.

$$\begin{aligned} \boldsymbol{H}^{GA}(LAB, PAS) &= \xi^G \sum_m^{\pm 1} (-1)^m A_{2,-m}(LAB, PAS) \bullet \boldsymbol{T}^G_{2,m}(LAB) \\ &= \xi^G A^G_{2,0}(LAB, PAS) \boldsymbol{T}^G_{2,0}(LAB) \\ &= \beta H \delta_{zz} \sqrt{\frac{6\pi}{5}} \left(\sqrt{\frac{5}{4\pi}} \right) \left(\frac{2}{\sqrt{6}} \boldsymbol{S}_z \right) = \beta H \delta_{zz} \boldsymbol{S}_z \end{aligned} \tag{39-22}$$

However, when the G interaction principal axes are not oriented to coincide with the laboratory axes the anisotropic Hamiltonian equation becomes much more complicated than the one above.

$$\begin{aligned} \boldsymbol{H}^{GA}(\theta, \varphi) &= \xi^G \sum_m^{\pm 2} (-1)^m A^G_{2,-m}(\theta, \varphi) \bullet \boldsymbol{T}^G_{2,m} \\ &= \xi^G [A^G_{2,0}(\theta, \varphi) \boldsymbol{T}^G_{2,0} + A^G_{2,1}(\theta, \varphi) \boldsymbol{T}^G_{2,-1} + A^G_{2,-1}(\theta, \varphi) \boldsymbol{T}^G_{2,1}] \\ &= \xi^G [A^G_{2,0}(\theta, \varphi) \boldsymbol{T}_{2,0} + A^G_{2,1}(\theta, \varphi) \boldsymbol{T}^G_{2,-1} - A^{G\,*}_{2,1}(\theta, \varphi) \boldsymbol{T}^G_{2,1}] \\ &= \xi^G \{A^G_{2,0}(\theta, \varphi) \boldsymbol{T}_{2,0} + Re[A^G_{2,1}(\theta, \varphi)](\boldsymbol{T}^G_{2,-1} - \boldsymbol{T}^G_{2,1}) + iIm[A^G_{2,1}(\theta, \varphi)](\boldsymbol{T}^G_{2,-1} + \boldsymbol{T}^G_{2,1})\} \end{aligned}$$

1. Keep in mind that this Hamiltonian is for a single electron. In a multi-spin system one will have to sum such Hamiltonians for all electron spins.

Remember, the orientation angles, θ and φ, are spherical angles relative to the laboratory coordinate system. We have thus left off the "LAB" label on all terms. At this point we will substitute in the spin operators (assuming H is along +z)

$$T^G_{2,0} = \frac{2}{\sqrt{6}}\boldsymbol{S}_z \qquad T^G_{2,\pm 1} = \mp\frac{1}{2}\boldsymbol{S}_\pm$$

This produces

$$\begin{aligned}\boldsymbol{H}^{GA}(\theta,\varphi) &= \xi^{SA}\{A^{SA}_{2,0}(\theta,\varphi)\boldsymbol{T}^{SA}_{2,0} + Re[A^{SA}_{2,1}(\theta,\varphi)](\boldsymbol{T}^{SA}_{2,-1} - \boldsymbol{T}^{SA}_{2,1}) + iIm[A^{SA}_{2,1}(\theta,\varphi)](\boldsymbol{T}^{SA}_{2,-1} + \boldsymbol{T}^{SA}_{2,1})\}\\ &= \xi^{SA}\left\{A^{SA}_{2,0}(\theta,\varphi)\left[\frac{2}{\sqrt{6}}\boldsymbol{S}_z\right] + Re[A^{SA}_{2,1}(\theta,\varphi)]\frac{1}{2}[(\boldsymbol{S}_- + \boldsymbol{S}_+)] + iIm[A^{SA}_{2,1}(\theta,\varphi)]\frac{1}{2}[(\boldsymbol{S}_- - \boldsymbol{S}_+)]\right.\end{aligned}$$

We can use the identities $I_x = \frac{1}{2}(I_- + I_+)$ $\quad I_y = \frac{i}{2}(I_- - I_+)$ to obtain

$$\boldsymbol{H}^{GA}(\theta,\varphi) = \xi^G\left\{A^G_{2,0}(\theta,\varphi)\left[\frac{2}{\sqrt{6}}\boldsymbol{S}_z\right] + Re[A^G_{2,1}(\theta,\varphi)]\boldsymbol{S}_x + Im[A^G_{2,1}(\theta,\varphi)]\boldsymbol{S}_y\right\}$$

Upon substitution of the oriented spatial components we obtain

$$\begin{aligned}\boldsymbol{H}^{GA}(\theta,\varphi) = \xi^G\Bigg\{&\sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]\left[\frac{2}{\sqrt{6}}\boldsymbol{S}_z\right]\\ &+ \left[\sqrt{\frac{5}{24\pi}}\sin\theta[3\cos\theta - \eta(\cos\theta\cos 2\varphi)]\right]\boldsymbol{S}_x + \left[\sqrt{\frac{5}{24\pi}}\sin\theta\eta\sin 2\varphi\right]\boldsymbol{S}_y\Bigg\}\end{aligned}$$

and in turn

$$\begin{aligned}\boldsymbol{H}^{GA}(\theta,\varphi) = \xi^G\sqrt{\frac{5}{24\pi}}\{&[3\cos^2\theta - 1 + \eta\sin^2\theta\cos 2\varphi]\boldsymbol{S}_z\\ &+ \sin\theta[\cos\theta(3 - \eta\cos 2\varphi)\boldsymbol{S}_x + \eta\sin 2\varphi\boldsymbol{S}_y]\}\end{aligned} \tag{39-23}$$

which will condense down into the previous result, equation (39-22) on page 345, when the two angles are set to zero. Often it can be assumed that work is being done in a "high field limit" where the contributions to the anisotropy from the $\boldsymbol{S}_x$ and $\boldsymbol{S}_y$ terms is negligible. When such is the case the previous equation becomes (hfl => high field limit)

$$\begin{aligned}\boldsymbol{H}^{GA}_{hfl}(\theta,\varphi) &= \xi^G\sqrt{\frac{5}{24\pi}}\{[3\cos^2\theta - 1 + \eta\sin^2\theta\cos 2\varphi]\boldsymbol{S}_z\\ &= \frac{1}{2}\beta H\delta_{zz}\{[3\cos^2\theta - 1 + \eta\sin^2\theta\cos 2\varphi]\boldsymbol{S}_z\end{aligned} \tag{39-24}$$

5.17.17 The Full G Hamiltonian

By combining the isotropic and anisotropic parts of the G Hamiltonian we obtain the full Hamiltonian. We are still excluding the anti-symmetric (rank 1) component.

$$\begin{aligned} \boldsymbol{H}^G(\theta,\varphi) &= \boldsymbol{H}^{GI} + \boldsymbol{H}^{GA}(\theta,\varphi) \\ &= \beta H g_{iso}\boldsymbol{S}_z + \frac{1}{2}\beta H \delta_{zz}\{[3\cos^2\theta - 1 + \eta\sin^2\theta\cos 2\varphi]\boldsymbol{S}_z \\ &\quad + \sin\theta[\cos\theta(3-\eta\cos 2\varphi)\boldsymbol{S}_x + \eta\sin 2\varphi\boldsymbol{S}_y]\} \end{aligned} \tag{39-25}$$

We will define an isotropic resonance condition as $\Omega_{iso} = \frac{\beta H g_{iso}}{h}$ so that the Hamiltonian can be expressed relative to some base frequency (or field) as

$$\begin{aligned} \boldsymbol{H}^G(\theta,\varphi) &= \Omega_{iso}\boldsymbol{S}_z + \frac{1}{2}\Omega_{iso}\frac{\delta_{zz}}{g_{iso}}\{[3\cos^2\theta - 1 + \eta\sin^2\theta\cos 2\varphi]\boldsymbol{S}_z \\ &\quad + \sin\theta[\cos\theta(3-\eta\cos 2\varphi)\boldsymbol{S}_x + \eta\sin 2\varphi\boldsymbol{S}_y]\} \end{aligned} \tag{39-26}$$

In the high-field limit, we have simply

$$\boldsymbol{H}^G_{hfl}(\theta,\varphi) = \left[1 + \frac{1}{2}\frac{\delta_{zz}}{g_{iso}}\{3\cos^2\theta - 1 + \eta\sin^2\theta\cos 2\varphi\}\right]\Omega_{iso}\boldsymbol{S}_z \tag{39-27}$$

and this explicitly indicates the dominant way in whcih the G Hamiltonian is modulated by the interaciton orientation.

5.17.18 Electron Transition Frequencies

Having determined what the **G** Hamiltonian looks like at any orientation we are now in the position to determine the electron transition frequency. Since the electron is only a spin 1/2 particle, there is only one transition and that is between the $|\alpha\rangle$ and $|\beta\rangle$ states. We shall examime the energy levels of these states using $H|\psi\rangle = \varepsilon|\psi\rangle$, knowing that the transition frequency will be the difference between the two energies. Our working Hamitonian form is

$$\boldsymbol{H}^G(\theta, \varphi) = \boldsymbol{H}^{GI} + \boldsymbol{H}^{GA}(\theta, \varphi) = \beta H g_{iso} \boldsymbol{S}_z + \boldsymbol{H}^{GA}(\theta, \varphi)$$

and we can immediately calculate the isotropic contribution to the transition frequency.

$$\begin{aligned} \boldsymbol{H}^{GI}|\alpha\rangle &= \beta H g_{iso} \boldsymbol{S}_z|\alpha\rangle = \frac{1}{2}\beta H g_{iso}|\alpha\rangle \\ \boldsymbol{H}^{GI}|\beta\rangle &= \beta H g_{iso} \boldsymbol{S}_z|\beta\rangle = \frac{-1}{2}\beta H g_{iso}|\beta\rangle \end{aligned} \qquad \Omega^{GI} = (\varepsilon_\alpha - \varepsilon_\beta)/h = \frac{\beta H g_{iso}}{h} \tag{39-28}$$

The anisotropic contribution at high field is equally trivial. In fact, we can just read it off of equation (39-27) on page 347.

$$\Omega^{GA}_{hfl}(\theta, \varphi) = \frac{1}{2h}\beta H \delta_{zz}\{3\cos^2\theta - 1 + \eta \sin^2\theta \cos 2\varphi\} \tag{39-29}$$

The third term, due to the x & y spin operator components are a bit more tenacious. We have

$$\begin{aligned} \boldsymbol{H}^{GA}_{x,y}(\theta, \varphi)|\alpha\rangle &= \xi^G \sqrt{\frac{5}{24\pi}}\{\sin\theta[\cos\theta(3 - \eta\cos 2\varphi)\boldsymbol{S}_x|\alpha\rangle + \eta \sin 2\varphi \boldsymbol{S}_y|\alpha\rangle]\} \\ \boldsymbol{H}^{GA}_{x,y}(\theta, \varphi)|\beta\rangle &= \xi^G \sqrt{\frac{5}{24\pi}}\{\sin\theta[\cos\theta(3 - \eta\cos 2\varphi)\boldsymbol{S}_x|\beta\rangle + \eta \sin 2\varphi \boldsymbol{S}_y|\beta\rangle]\} \end{aligned}$$

We can use the ladder operators define d earlier to determine the

$$I_x|\alpha\rangle = \frac{1}{2}(I_- + I_+)|\alpha\rangle = \frac{1}{2}|\beta\rangle \qquad I_y|\alpha\rangle = \frac{i}{2}(I_- - I_+)|\alpha\rangle = \left(-\frac{i}{2}\right)|\beta\rangle$$

$$\Omega^G_{hfl}(\theta, \varphi) = \frac{\beta H}{h}\left[g_{iso} + \frac{\delta_{zz}}{2}\{3\cos^2\theta - 1 + \eta \sin^2\theta \cos 2\varphi\}\right] = \frac{\beta H}{h} g_{eff}$$

where

$$g_{eff} = g_{iso} + \frac{\delta_{zz}}{2}\{3\cos^2\theta - 1 + \eta \sin^2\theta \cos 2\varphi\}$$

The Rank 2 G Hamiltonian Summary

$$\boldsymbol{H}^G(AAS) = \sum_i \boldsymbol{H}_i^G(AAS) = \sum_i \xi^G \sum_{l=0}^{2} \sum_m (-1)^m A_{l-m}(i, AAS) \bullet \boldsymbol{T}_{lm}^G(i, AAS)$$

$$\boldsymbol{H}_i^G(AAS) = \xi^G \sum_{l=0}^{2} \sum_{m}^{\pm l} (-1)^m A_{l-m}(i, AAS) \boldsymbol{T}_{lm}^G(i, AAS)$$

$$\xi^G = \beta H \delta_{zz} \sqrt{\frac{6\pi}{5}}$$

$$A_{l,m}^G(i, AAS) = \sum_{m'}^{\pm l} D_{mm'}^l(\varphi, \theta, \chi) A_{l,m'}^G(i, PAS)$$

$$A_{0,0}^G(i, PAS) = -\sqrt{3} g_{iso}(i)$$

$$A_{1,0}^G(i, PAS) = -\frac{i}{\sqrt{2}}[g_{xy}(i, PAS) - g_{yx}(i, PAS)]$$

$$A_{1,\pm1}^G(i, PAS) = -\frac{1}{2}[(g_{zx}(i, PAS) - g_{xz}(i, PAS)) \pm i(g_{zy}(i, PAS) - g_{yz}(i, PAS))]$$

$$A_{2,0}^G(i, PAS) = \sqrt{3/2} g_{zz}(i) \qquad A_{2,\pm1}^G(i, PAS) = 0 \qquad A_{2,\pm2}^G(i, PAS) = \frac{1}{2}\delta_{zz}(i)\eta(i)$$

$$T_{0,0}^G(i, AAS) = \frac{-1}{\sqrt{3}}\left[\boldsymbol{I}_{iz}B_z + \frac{1}{2}(\boldsymbol{I}_{i+}B_- + \boldsymbol{I}_{i-}B_+)\right] = \frac{-1}{\sqrt{3}}\vec{\boldsymbol{I}}_i \bullet \vec{\boldsymbol{B}}_n$$

$$T_{1,0}^G(i, AAS) = \frac{-1}{2\sqrt{2}}[\boldsymbol{I}_{i+}B_- - \boldsymbol{I}_{i-}B_+] \qquad T_{1,\pm1}^G(i, AAS) = \frac{-1}{2}[\boldsymbol{I}_{i\pm}\ B_z - \boldsymbol{I}_{iz}B_\pm\ ]$$

$$T_{2,0}^G(i, AAS) = \frac{1}{\sqrt{6}}[3\boldsymbol{I}_{iz}B_z - (\vec{\boldsymbol{I}}_i \bullet \vec{\boldsymbol{B}}_n)]$$

$$T_{2,\pm1}^G(i, AAS) = \mp\frac{1}{2}[\boldsymbol{I}_{i\pm}\ B_z + \boldsymbol{I}_{iz}B_\pm\ ] \qquad T_{2,\pm2}^G(i, AAS) = \frac{1}{2}[\boldsymbol{I}_{i\pm}\ B_\pm\ ]$$

Although these equations are generally applicable, it is convenient to express the G Hamiltonian with clear separation between the different ranks (the components with differing values of l). The isotropic component $\boldsymbol{H}^{GI}$ in the treatment of liquid samples will normally be placed into an overall isotropic Hamiltonian, H_0 because it does not disappear upon rotational averaging. The asymmetric component, $\boldsymbol{H}^{GU}$, is usually zero, the G tensor taken as essentially symmetric. ***The***

The Electron G Anisotropy Hamiltonian

Arbitrary Axis System

$$\boldsymbol{H}^{GA}(AAS) = \xi^G \sum_{m}^{\pm 2} (-1)^m A_{2-m}(AAS) \boldsymbol{T}^G_{2m}(AAS)$$

$$\xi^G = \beta H \delta_{zz} \sqrt{\frac{6\pi}{5}}$$

$$A_{2,0}(PAS) = \sqrt{\frac{5}{4\pi}} \qquad T^G_{2,0} = \frac{1}{\sqrt{6}}[3\boldsymbol{S}_z H_z - (\vec{\boldsymbol{S}} \bullet \vec{\boldsymbol{H}}_n)]$$

$$A_{2,\pm 1}(PAS) = 0 \qquad T^G_{2,\pm 1} = \mp\frac{1}{2}[\boldsymbol{S}_{\pm}\ {}_{+}H_z + \boldsymbol{S}_z H_{\pm}\ ]$$

$$A_{2,\pm 2}(PAS) = \sqrt{\frac{5}{24\pi}}\eta \qquad T^G_{2,\pm 2} = \frac{1}{2}[\boldsymbol{S}_{\pm}\ H_{\pm}\ ]$$

$$A_{2,m}(AAS) = \sum_{m'}^{\pm 2} D^2_{mm'}(\varphi, \theta, \chi) A_{2,m'}(PAS)$$

Laboratory Frame

$$\boldsymbol{H}^{GA}(LAB) = \xi^G \sum_{m}^{\pm 1} (-1)^m A_{2-m}(LAB) \boldsymbol{T}^G_{2m}(LAB)$$

$$\xi^G = \beta H \delta_{zz} \sqrt{\frac{6\pi}{5}}$$

$$A_{2,0}(PAS) = \sqrt{\frac{5}{4\pi}} \qquad T^G_{2,0}(LAB) = \frac{2}{\sqrt{6}}\boldsymbol{I}_{iz}$$

$$A_{2,\pm 1}(PAS) = 0 \qquad T^G_{2,\pm 1}(LAB) = \mp\frac{1}{2}\boldsymbol{I}_{i\pm}$$

$$A_{2,\pm 2}(PAS) = \sqrt{\frac{5}{24\pi}}\eta \qquad T^G_{2,\pm 2}(LAB) = 0$$

$$A_{2,m}(LAB) = \sum_{m'}^{\pm 2} D^2_{mm'}(\varphi_{PAS \to LAB}, \theta_{PAS \to LAB}, \chi_{PAS \to LAB}) A_{2,m'}(PAS)$$

$$A^G_{2,m}(i, LAB)\Big|_{\eta = 0} = Y_{2,m}(\theta, \varphi)$$

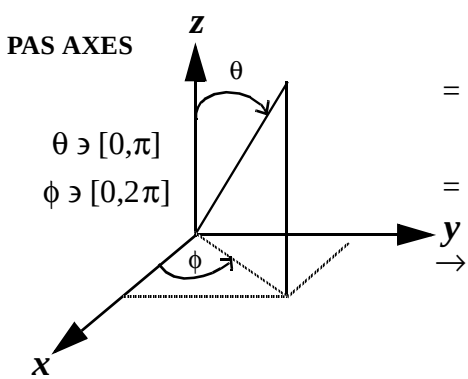
$$T^G_{2,0}(LAB) = \frac{-2}{\sqrt{6}}\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad T^G_{2,1}(LAB) = \frac{-1}{2}\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \qquad T^G_{2,-1}(LAB) = \frac{1}{2}\begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \qquad T^G_{2,\pm 2}(LAB) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

5.17.19 G PAS Equations

When the G interaction has alignment along its principal axes system virtually all of the G spatial tensor equations simplify. However, because the magnetic field components will then be oriented, the space-spin tensor components become complicated. Only when the PAS is aligned with the laboratory z-axis do both space and space-spin simplify. The following figure collects these equations for convenience.

G Equations Involving the PAS

$$\boldsymbol{H}^G(PAS) = \beta H g_{iso} S_z + \xi^G \sum_m (-1)^m A_{2-m}(PAS)\boldsymbol{T}^G_{2m}(PAS)$$



$$= \beta H g_{iso} S_z + \xi^G \sqrt{\frac{5}{24\pi}}[\sqrt{6}T^G_{2,0} + \eta(T^G_{2,2} + T^G_{2,-2})]$$

$$= \beta H\left[g_{iso} S_z + \frac{\delta_{zz}}{2}\left(3\boldsymbol{S}_z H_z - (\vec{\boldsymbol{S}} \bullet \vec{\boldsymbol{H}}_n) + \frac{\eta}{2}(\boldsymbol{S}_- H_- + \boldsymbol{S}_+ H_+)\right)\right]$$

$$\rightarrow \beta H(g_{iso} + \delta_{zz})\boldsymbol{S}_z$$

$$\xi^G = \beta H \delta_{zz}\sqrt{\frac{6\pi}{5}}$$

$$\eta = (A_{xx} - A_{yy})/A_{zz}$$

$$|A_{zz}| \geq |A_{yy}| \geq |A_{xx}|$$

$$A_{2,0}(PAS) = \sqrt{6}[3A_{zz} - Tr\{A\}]\big|_{PAS} = \sqrt{\frac{5}{4\pi}}$$

$$A_{2,\pm1}(PAS) = \mp\frac{1}{2}[A_{xz} + A_{zx} \pm i(A_{yz} + A_{zy})]\Big|_{PAS} = 0$$

$$A_{2,\pm2}(PAS) = \frac{1}{2}[A_{xx} - A_{yy} \pm i(A_{xy} + A_{yx})]\big|_{PAS} = \sqrt{\frac{5}{24\pi}}\eta$$

$$A_{xx}(PAS) = \sqrt{\frac{5}{24\pi}}[\eta - 1] \qquad A_{yy}(PAS) = -\sqrt{\frac{5}{24\pi}}[1 + \eta] \qquad A_{zz}(\theta, \varphi) = \sqrt{\frac{5}{6\pi}}$$

$$A_{xz}(PAS) = 0 = A_{zx}(PAS) = A_{xy}(PAS) = A_{yx}(PAS) = A_{yz}(PAS) = A_{zy}(PAS)$$

Figure 19-33 **Equations relevant to the G interaction in its principal axis orientation (PAS). GAMMA uses a spatial tensor which is scaled[1] so that rotations by angles θ & ϕ produce spherical harmonics for a symmetric interaction (η = 0).**

Included are the general relationships between the (GAMMA scaled) Cartesian tensor components to the irreducible spherical components. They are valid when η is defined accordingly! If η is defined by the other common convention ($|A_{zz}| \geq |A_{xx}| \geq |A_{yy}|$) then the sign on the $A^G_{2,\pm2}$ will change as will the sign on the Hamiltonian terms multiplied by η.

1. The scaling on both $\{A_{2m}\}$ and $T_{2m}\}$ are arbitrary, GAMMA uses an (uncommon) scaling which independent of the interaction type. What is NOT arbitrary is the scaling within either of the two sets of components. In addition, the combined scaling of the two sets is critical to the proper formation of G Hamiltonians. For that, GAMMA uses an interaction constant.

5.17.20 G Equations At Any Orientation

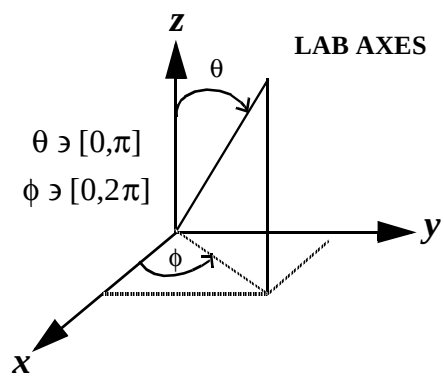
When the G interaction has a arbitrary alignment (relative to the laboratory frame, where the static field sets the z-axis) the G equations become sligltly more complicated. The figure below depicts them for convenience.

G Equations Oriented At Angles {θ,ϕ} From Lab Frame[1]

$$\boldsymbol{H}^G(\theta,\varphi) = \beta H g_{iso} S_z + \xi^G \sum_m^{\pm 2} (-1)^m A_{2,-m}(\theta,\varphi) \bullet \boldsymbol{T}^G_{2,m}$$

$$T^G_{2,0}(LAB) = \frac{2}{\sqrt{6}} \boldsymbol{I}_{iz}$$

$$T^G_{2,\pm 1}(LAB) = \mp \frac{1}{2} \boldsymbol{I}_{i\pm} \qquad T^G_{2,\pm 2}(LAB) = 0$$



$$A_{2,0}(\theta,\varphi) = \sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta \sin^2\theta \cos 2\varphi\right]$$

$$A_{2,1}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}} \sin\theta[3\cos\theta - \eta(\cos\theta\cos 2\varphi - i\sin 2\varphi)] = -A_{2,-1}{}^*(\theta,\varphi)$$

$$A_{2,2}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}\frac{1}{2}[3\sin^2\theta + \eta[\cos 2\varphi(1+\cos^2\theta) - i2\sin 2\varphi\cos\theta]] = A_{2,-2}{}^*(\theta,\varphi)$$

$$\eta = (A_{xx} - A_{yy})/A_{zz} \qquad |A_{zz}| \geq |A_{yy}| \geq |A_{xx}| \qquad \xi^G = \beta H \delta_{zz}\sqrt{\frac{6\pi}{5}}$$

$$\boldsymbol{H}^G(\theta,\varphi) = \beta H g_{iso} S_z + \frac{1}{2}\beta H \delta_{zz}\{[3\cos^2\theta - 1 + \eta\sin^2\theta\cos 2\varphi]\boldsymbol{S}_z + \sin\theta[\cos\theta(3 - \eta\cos 2\varphi)\boldsymbol{S}_x + \eta\sin 2\varphi \boldsymbol{S}_y]\}$$

Figure 19-34 **Equations relevant to the G Hamiltonian when oriented at angles θ & ϕ from the laboratory axis orientation (LAB). GAMMA uses a spatial tensor which is scaled[2] so that rotations by angles θ & ϕ produce spherical harmonics for a symmetric interaction (η = 0).**

1. The G interaction constant, as well as the relative scalings on the sets of spatial and spin tensors, can be adjusted as desired. However all components of the space or spin tensor must be adjusted by the same scaling. The GAMMA scaling is oriented to liquids where so that all spatial components are related to the spherical harmonics in the spatial tensor PAS.
2. The scaling on both $\{A_{2m}\}$ and $T_{2m}\}$ are arbitrary, GAMMA uses a scaling which independent of the interaction type. What is NOT arbitrary is the scaling within either of the two sets of components. In addition, the combined scaling of the two sets is also crucial. For that, GAMMA uses an interaction constant.

5.18 G Interaction Parameters

This section describes how an ASCII file may be constructed that is self readable by a G interaction. The file can be created with any editor and is read with the G interaction member function “read”. An example of one such file is given in its entirety at the end of this section. Keep in mind that parameter ordering in the file is arbitrary. Other parameters are allowed in the file which do not relate to G interactions.

Table 4: G Interaction Parameters

	Parameter	Units	Examples Parameter (Type) : Value - Statement
I.	AG	KHz	AG (1) : 370.3 - G Spatial Tensor (spherical)
II.	g	KHz	g (1) : 370.3 - Isotropic G value (Gauss)
	ga	none	ga (1) : 0.33 - G asymmetry value
	geta	degrees	geta (1) : 127.2 - G anisotropy value
	gtheta	degrees	gtheta (1) : 127.2 - G Orientation from PAS z (deg)
	gphi	degrees	gphi (1) : 270.9 - G Orientation from PAS x(deg)
III.	gxx	KHz	gxx (1) : 370.3 - G Cartesian PAS x-axis (Gauss)
	gyy	none	gyy (1) : 0.33 - G Cartesian PAS y-axis (Gauss)y
	gzz	degrees	gzz (1) : 127.2 - G Cartesian PAS z-axis (Gauss)
	gtheta	degrees	gtheta (1) : 127.2 - G Orientation from PAS z (deg)
	gphi	degrees	gphi (1) : 270.9 - G Orientation from PAS x(deg)

AG: WG, WGkHz, WGKHz, WGHz, WGMHz

The G frequency can be specified. This can be accomplished with parameters using any of the names above or these names with a (#) added as a suffix. The default units for WG are KHz other names can be used to set the value in particular units. Note that this parameter is related to the G coupling constant which is specified with “(N)GCC” parameters. If both GCC and WG are set in the same file, the G frequency will be used to set

up the G interaction.

Table 5: G Frequency[a]

Parameter	Assumed Units	Examples Parameter (Type=1) : Value - Statement
WG	KHz	WG (1) : 320.13 - Guad. Frequency in kHz
WGMHz	MHz	WGMHz (1) : 1.27 - Guad. Frequency in MHz
WGHz	Hz	WGHz(2) (1) : 1320.7 - Guad. Frequency in Hz

a. Shown are three possible parameters used to set the G frequency. The others mentioned above can also be used to specify it. Specification of a G coupling constant will also set the interaction's G frequency. Parameter type 1 indicates a double precision number parameter

G Frequency: WG, WGkHz, WGKHz, WGHz, WGMHz

The G frequency can be specified. This can be accomplished with parameters using any of the names above or these names with a (#) added as a suffix. The default units for WG are KHz other names can be used to set the value in particular units. Note that this parameter is related to the G coupling constant which is specified with "(N)GCC" parameters. If both GCC and WG are set in the same file, the G frequency will be used to set up the G interaction.

Table 6: G Frequency[a]

Parameter	Assumed Units	Examples Parameter (Type=1) : Value - Statement
WG	KHz	WG (1) : 320.13 - Guad. Frequency in kHz
WGMHz	MHz	WGMHz (1) : 1.27 - Guad. Frequency in MHz
WGHz	Hz	WGHz(2) (1) : 1320.7 - Guad. Frequency in Hz

a. Shown are three possible parameters used to set the G frequency. The others mentioned above can also be used to specify it. Specification of a G coupling constant will also set the interaction's G frequency. Parameter type 1 indicates a double precision number parameter

G Coupling Constant: GCC, GCCkHz, GCCKHz, GCCHz, GCCMHz

The G coupling constant can be specified. This can be accomplished with parameters using any of the names above, these same names with an "N" as a prefix, and/or these names with a (#) added as a suffix. The default units for GCC are KHz other names can be used to set the value in particular units. Note that this parameter is related to the G frequency which is specified with "WG" parameters. If both GCC and WG are set in the

same file, the G frequency will be used to set up the G interaction.

Table 7: G Coupling Constant[a]

Parameter	Assumed Units	Examples Parameter (Type=1) : Value - Statement
GCC	KHz	GCC (1) : 320.13 - Guad. Coupling in kHz
NGCCMHz	MHz	NGCCMHz (1) : 1.27 - Guad. Coupling in MHz
GCCHz	Hz	GCCHz(2) (1) : 1320.7 - Guad. Coupling in Hz

a. Shown are three possible parameters used to set the G coupling. The others mentioned above can also be used to specify it. Specification of a G frequency will also set the G coupling in the interaction. Parameter type 1 indicates a double precision number parameter

G Asymmetry

The asymmetry parameter must be within the range of [0, 1]. This parameter does not need to be set for a G interaction definition, it will be assumed 0 if unspecified.

Table 8: G Asymmetry[a]

Parameter	Assumed Units	Examples Parameter (Type=1) : Value - Statement
Geta	none	Geta (1) : 0.4 - G Asymmetry

a. Parameter type 1 indicates an integer parameter.

G Theta Orientation

The angle theta which relates the G interactions orientation down from the z-axis of its PAS may be set. This is not essential and will be taken as zero in left unspecified.

Table 9: Theta Orientation[a]

Parameter	Assumed Units	Examples Parameter (Type=1) : Value - Statement
Gtheta	degrees	Gtheta (1) : 45.7 - G Orientation from PAS z

a. Parameter type 1 indicates an integer parameter.

G Phi Orientation

The angle phi which relates the G interactions orientation over from the x-axis of its PAS may be set. This is

not essential and will be taken as zero in left unspecified.

Table 10: Theta Orientation[a]

Parameter	Assumed Units	Examples Parameter (Type=1) : Value - Statement
Gphi	degrees	Gphi (1) : 134.6 - G Orientation from PAS x

a. Parameter type 1 indicates an integer parameter.

5.19 Literature Comparisons

5.19.1

The fol

Comparison of GAMMA & Equations

5.20 G Interaction Examples

5.20.1 Zero Field Transitions, First Order Spectra

As a first example we'll look into some of the G Hamiltonians provided by class IntG in the interaction PAS (principal axes). Our results for both the transitions at zero field and NMR spectra to first order should agree with A. J. Vega's article[1] figures 1 & 2.

First Order G Spectra

Figure 19-35 **Spectra produced by program IntQu_LC6.cc, page -151. The G frequency was set to 300 kHz. The interaction was in its PAS and the asymmetry set to zero. Zero field transtions & relative intensities are shown in the tables.**

1. "G Nuclei in Solids", Alexander J. Vega, Encyclopedia of Nuclear Magnetic Resonance, Editors-in-Chief D.M.Grant and R.K. Harris, Vol. 6, Ped-Rel, pgs 3869-3889.

5.21 References

[7] J.E. Wertz and J.R. Bolton, *Electron Spin Resonance. Elementary Theory and Practical Applications*, McGraw-Hill Book Co., New York, New York, (1986), Chapman and Hall.

[8] Brink, D.M. and Satchler, G.R. (1962), Angular Momentum, Clarendon Press, Oxford.

0.4 Programs and Input Files

IntGu_LC0.cc

```
/* IntGu_LC0.cc
*************************************************************-*-c++-*-
**
**
**                    Test Program for the GAMMA Library
**
**              G Interaction Literature Comparison 0
**
**
**
** This program checks the G interaction class IntG in
**
** GAMMA.  In particular it looks to see how well the class parallels
**
** the articles by Pascal P. Man
**
**
**
** "G Interactions", Encyclopedia of Magnetic Resonance,
**
** by Grant and Harris, Vol 6, Ped-Rel, page 3838-3869.
**
**
**
** and Alexander Vega
**
**
**
** "G Nuclei in Solids", Encyclopedia of Magnetic Resonance,
**
** by Grant and Harris, Vol 6, Ped-Rel, page 3869-3889.
**
**
**
** In particular, their PAS G Hamiltonians are generated and
**
** compared with the G interaction class Hamiltonians.
**
**
**
** Man's Hamiltonians will be generated from equations in (5) on page
**
** 3839 of the his article.  Vega's Hamiltonians will be made from
**
** equations (28), (32) and (33) of his article.  Note that his (32)
**
** is missing a factor of 1/3 on the <1|H|3> and <3|H|1> components.
**
**
** Author:     S.A. Smith
**
** Date:       10/11/96
**
** Update:     10/11/96
**
** Version:    3.6
**
** Copyright:  S. Smith.  You can modify this program for personal use, but
**
**                    you must leave it intact if you re-distribute it.
**
**
**
*********************************************************************
*******************/

#include <gamma.h>                          // Include GAMMA

main (int argc, char* argv[])
{
//                    Set Up The G Interaction

  int qn=1;
  double I;
  query_parameter(argc, argv, qn++,                  // Read in the coupling
                  "\n\tSpin Guantum Number? ", I);
  double W;                                          // G frequency
  query_parameter(argc, argv, qn++,                  // Read in the coupling
    "\n\tG Frequency(kHz)? ", W);
  W *= 1.e3;                                         // Put this in Hz
  double Geta;
  query_parameter(argc, argv, qn++,                  // Read in the coupling
    "\n\tG Asymmetry [0, 1]? ", Geta);
//                    Construct GAMMA G Interaction

  IntG G(I,wG2GCC(W,I),Geta,0.0,0.0);
//                Here are the Operators To Build Man's Hamiltonians

  int Ival = int(2.*I + 1);          // For 1 spin SOp functions
  matrix IE = Ie(Ival);              // The operator 1
  matrix IM = Im(Ival);              // The operator I-
  matrix IP = Ip(Ival);              // The operator I+
  matrix IZ = Iz(Ival);              // The operator Iz
  matrix IX = Ix(Ival);              // The operator Ix
  matrix IY = Iy(Ival);              // The operator Iy
//                Here's The H Accoring To Man's Equation (5a)

//                (Note That His W is Half Of Our Definition)

  matrix HMa = 3.0*IZ*IZ - (I*(I+1))*IE + Geta*((IX*IX)-(IY*IY));
  HMa *= (W/6.0);
```

```
//                    Here's The H Accoring To Man'c Equation (5c)
//                    (Note That His W is Half Of Our Definition)
 matrix HMb = 3.0*IZ*IZ - (I*(I+1))*IE + (Geta/2.)*((IP*IP)+(IM*IM));
 HMb *= (W/6.0);
//                    Here's The H According To GAMMA
 matrix HG = G.H();
//                    Here's The H Also According To GAMMA
 matrix HGB = G.H(0.0, 0.0);
//                    Here Are Vegas V's According To Equations (22-27, 31)
//                    (Switches eta Sign To Account For Opposite PAS Definition)
 double Eta = -G.eta();
 double Vxx = 0.5*(- 1. - Eta);
 double Vyy = 0.5*(-1. + Eta);
 double Vzz = 1.0;
 double Vxy = 0.0;
 double Vxz = 0.0;
 double Vyz = 0.0;
 complex V1(-Vxz, -Vyz);
 complex Vm1(Vxz, -Vyz);
 complex V2(0.5*(Vxx-Vyy), Vxy);
 complex Vm2(0.5*(Vxx-Vyy), -Vxy);
//          Generate H According To Vega's Equations (32) Or (33)
 matrix HVega;
 if(I == 1)
  {
  HVega = matrix(3,3);
  HVega.put(Vzz/6.0, 0, 0);
  HVega.put(Vm1/sqrt(2.0), 0, 1);
  HVega.put(Vm2/3., 0, 2);                    // Added 1/3 Factor!
  HVega.put(-V1/sqrt(2.0), 1, 0);
  HVega.put(-Vzz/3.0, 1, 1);
  HVega.put(-Vm1/sqrt(2.0), 1, 2);
  HVega.put(V2/3., 2, 0);                     // Added 1/3 Factor!
  HVega.put(V1/sqrt(2.0), 2, 1);
  HVega.put(Vzz/6.0, 2, 2);
  HVega *= G.wG();
  }
 else if(I == 1.5)
  {
  HVega = matrix(4,4);
  HVega.put(Vzz/2.0, 0, 0);
  HVega.put(Vm1/sqrt(3.0), 0, 1);
  HVega.put(Vm2/sqrt(3.0), 0, 2);
  HVega.put(0.0, 0, 3);
  HVega.put(-V1/sqrt(3.0), 1, 0);
  HVega.put(-Vzz/2.0, 1, 1);
  HVega.put(0.0, 1, 2);
  HVega.put(Vm2/sqrt(3.0), 1, 3);
  HVega.put(V2/sqrt(3.0), 2, 0);
  HVega.put(0.0, 2, 1);
  HVega.put(-Vzz/2.0, 2, 2);
  HVega.put(-Vm1/sqrt(3.0), 2, 3);
  HVega.put(0.0, 3, 0);
  HVega.put(V2/sqrt(3.0), 3, 1);
  HVega.put(V1/sqrt(3.0), 3, 2);
  HVega.put(Vzz/2.0, 3, 3);
  HVega *= G.wG();
  }
//                    Generate H According To Vega's Equation (28)
 matrix HV = Vzz*(3.*IZ*IZ-(I*(I+1.))*IE);
 HV += (Vxx-Vyy)*(IX*IX-IY*IY);
 HV += 2*Vxy*(IX*IY-IY*IX);
 HV += 2*Vxz*(IX*IZ-IZ*IX);
 HV += 2*Vyz*(IY*IZ-IZ*IY);
 HV *= G.wG()/6.0;
//                    Output the Results for Visual Comparison
 cout << "\n\t\t\t\tGAMMA's G H:\t" << HG;
 cout << "\n\t\t\t\tGAMMA's Other G H:\t" << HGB;
 cout << "\n\t\t\t\tMan's G H(a):\n\t" << HMa;
 cout << "\n\t\t\t\tMan's G H(b):\n\t" << HMb;
 if(I == 1.0 || I == 1.5)
  cout << "\n\t\t\t\tVega's G H:\n\t" << HVega;
 cout << "\n\t\t\t\tVega's Generic Guad H:\n\t" << HV;
 }
```

IntGu_LC1.cc

```
/* IntGu_LC1.cc
*****************************************************************-*-c++-*-
**
**
**                    Test Program for the GAMMA Library
**
**                G Interaction Literature Comparison 1
**
**
**
**  This program checks the G interaction class IntG in
**
**  GAMMA.  In particular it looks to see how well the class parallels
**
**  the article by Alexander Vega -
**
**
**
**  "G Nuclei in Solids", Encyclopedia of Magnetic Resonance,
**
```

```
**  by Grant and Harris, Vol 6, Ped-Rel, page 3869-3889.
**
**
**
**  Specifcally, herein we generate the spatial tensor components of
**
**  an oriented G interaction and and compare the results to
**
**  A.Vega's equations (22-27) and 31 on pages 3884-3885.
**
**
**
** Author:     S.A. Smith
**
** Date:       10/11/96
**
** Update:     10/11/96
**
** Version:    3.6
**
** Copyright:  S. Smith.  You can modify this program as you see fit
**
**            for personal use, but you must leave the program intact
**
**            if you re-distribute it.
**
**
**
*************************************************************************
*******************/

#include <gamma.h>                                  // Include GAMMA

main (int argc, char* argv[])

 {
//                                  Construct A G Interaction

 int qn=1;
 double W;                                          // G frequency
 query_parameter(argc, argv, qn++,                  // Read in the coupling
          "\n\tG Frequency(kHz)? ", W);
 W *= 1.e3;                                         // Put this in Hz
 double Geta;
 query_parameter(argc, argv, qn++,                  // Read in the coupling
  "\n\tG Asymmetry [0, 1]? ", Geta);
 double Gtheta, Gphi;
 query_parameter(argc, argv, qn++,                  // Read in the angle
  "\n\tAngle down from z [0, 180]? ", Gtheta);
 query_parameter(argc, argv, qn++,                  // Read in the angle
    "\n\tAngle over from x [0, 360]? ", Gphi);
 double I=1.0;                                      // Use I=1, but this doesn't
 double GCC = wG2GCC(W, I);                         // Heres quad. coupling
 IntG G(I,GCC,Geta,Gtheta,Gphi);                    // matter for spatial parts
//                  Here Are Vegas V's According To Equations (22-27, 31)
//        Note We Change Sign On ETA As He Using A Different PAS Definition
 double Theta = G.theta()*DEG2RAD;
 double Phi = G.phi()*DEG2RAD;
 double Eta = -G.eta();
 double Stheta = sin(Theta);
 double Ctheta = cos(Theta);
 double C2phi = cos(2.*Phi);
 double S2phi = sin(2.*Phi);
 double Vxx = 0.5*(3.*Stheta*Stheta - 1. - Eta*Ctheta*Ctheta*C2phi);
 double Vxy = 0.5*Eta*Ctheta*S2phi;
 double Vxz = -0.5*(Stheta*Ctheta*(3.0 + Eta*C2phi));
 double Vyx = Vxy;
 double Vyy = 0.5*(-1. + Eta*C2phi);
 double Vyz = 0.5*Eta*Stheta*S2phi;
 double Vzx = Vxz;
 double Vzy = Vyz;
 double Vzz = 0.5*(3.*Ctheta*Ctheta - 1. - Eta*Stheta*Stheta*C2phi);
 complex V0(sqrt(1.5)*Vzz);
 complex V1(-Vxz, -Vyz);
 complex Vm1(Vxz, -Vyz);
 complex V2(0.5*(Vxx-Vyy), Vxy);
 complex Vm2(0.5*(Vxx-Vyy), -Vxy);
//                  Here Are The A's According To GAMMA G Interaction
//            Need To Scale Our A's By (1/2)/sqrt[5/(24*PI)] To Get Vega's V's
 double X = 0.5/RT5O24PI;
 double Thetad = G.theta();
 double Phid = G.phi();
 double AGxx = X*G.Axx(Thetad, Phid);
 double AGxy = X*G.Axy(Thetad, Phid);
 double AGxz = X*G.Axz(Thetad, Phid);
 double AGyy = X*G.Ayy(Thetad, Phid);
 double AGyx = X*G.Ayx(Thetad, Phid);
 double AGyz = X*G.Ayz(Thetad, Phid);
 double AGzz = X*G.Azz(Thetad, Phid);
 double AGzx = X*G.Azx(Thetad, Phid);
 double AGzy = X*G.Azy(Thetad, Phid);
//                  Here Are The A's According To GAMMA G Interaction
//            Need To Scale Our A's By (1/2)/sqrt[5/(24*PI)] To Get Vega's V's
 double AG1xx = X*G.Axx();
 double AG1xy = X*G.Axy();
 double AG1xz = X*G.Axz();
 double AG1yy = X*G.Ayy();
 double AG1yx = X*G.Ayx();
 double AG1yz = X*G.Ayz();
 double AG1zz = X*G.Azz();
```

```
 double AG1zx = X*G.Azx();
 double AG1zy = X*G.Azy();
//                    Here Are The A's According To GAMMA G Interaction
//          (Note That space_T Uses Azz>=Ayy>=Axx So ETA Opposite Vega's)
 space_T Agen = A2(0.0, 1.0, Geta);
 Agen = Agen.rotate(Phid, Thetad, 0.0);
 Cartesian(Agen);
//                          Output Everyone For A Visual Comparison
 cout << "\n   " << "       Vega" << "          IntGA"
                << "     IntGB" << "      space_T";
 cout << "\nVxx    " << form("%8.3f", Vxx) << "        " << form("%8.3f", AGxx)
      << "      " << form("%8.3f", AG1xx) << "     " << form("%8.3f", Agen.Ccomponent(0,0));
 cout << "\nVxy    " << form("%8.3f", Vxy) << "        " << form("%8.3f", AGxy)
      << "      " << form("%8.3f", AG1xy)  << "     " << form("%8.3f", Agen.Ccomponent(0,1));
 cout << "\nVxz    " << form("%8.3f", Vxz) << "        " << form("%8.3f", AGxz)
      << "      " << form("%8.3f", AG1xz) << "     " << form("%8.3f", Agen.Ccomponent(0,2));
 cout << "\nVyy    " << form("%8.3f", Vyy) << "        " << form("%8.3f", AGyy)
      << "      " << form("%8.3f", AG1yy) << "     " << form("%8.3f", Agen.Ccomponent(1,1));
 cout << "\nVyx    " << form("%8.3f", Vyx) << "        " << form("%8.3f", AGyx)
      << "      " << form("%8.3f", AG1yx) << "     " << form("%8.3f", Agen.Ccomponent(1,0));
 cout << "\nVyz    " << form("%8.3f", Vyz) << "        " << form("%8.3f", AGyz)
      << "      " << form("%8.3f", AG1yz) << "     " << form("%8.3f", Agen.Ccomponent(1,2));
 cout << "\nVzz    " << form("%8.3f", Vzz) << "        " << form("%8.3f", AGzz)
      << "      " << form("%8.3f", AG1zz) << "     " << form("%8.3f", Agen.Ccomponent(2,2));
 cout << "\nVzx    " << form("%8.3f", Vzx) << "        " << form("%8.3f", AGzx)
      << "      " << form("%8.3f", AG1zx) << "     " << form("%8.3f", Agen.Ccomponent(2,0));
 cout << "\nVzy    " << form("%8.3f", Vzy) << "        " << form("%8.3f", AGzy)
      << "      " << form("%8.3f", AG1zy) << "     " << form("%8.3f", Agen.Ccomponent(2,1));
 cout << "\nV0 " << V0 << "  " << X*G.A0(Thetad, Phid)
      << "  " << X*G.A0() << "  " << Agen.component(2,0);
 cout << "\nV1 " << V1 << "  " << X*G.A1(Thetad, Phid)
      << "  " << X*G.A1() << "  " << Agen.component(2,1);
 cout << "\nV-1" << Vm1 << "  " << X*G.Am1(Thetad, Phid)
      << "  " << X*G.Am1() << "  " << Agen.component(2,-1);
 cout << "\nV2 " << V2 << "  " << X*G.A2(Thetad, Phid)
      << "  " << X*G.A2() << "  " << Agen.component(2,2);
 cout << "\nV-2" << Vm2 << "  " << X*G.Am2(Thetad, Phid)
      << "  " << X*G.Am2() << "  " << Agen.component(2,-2);
 cout << "\n\n\n";
 }
```

/IntGu_PCT0.cc

```
/* IntGu_PCT0.cc
*************************************************************-*-c++-*-
**
**
**                    Example Program for the GAMMA Library
**
**
**
**  This program calculates a powder average for a single spin which
**
**  is associated with a G interaction.  The high field
**
**  approximation is invoked in that the G Hamiltonian is
**
**  treated as a perturbation to the Zeeman Hamiltonian and taken to
**
**  second order.  Only G Hamiltonian terms which are
**
**  rotationally invariant about the field axis (z) are maintained.
**
**  Furthermore, only the central transtion will be considered.
**
**
**
**  This will program is similar to IntGu_Pow2.cc but restricts the
**
**  computation to only the central transition. In turn, that means
**
**  only spins with I=m*1/2 where m is odd and larger than 1 are valid.
**
**  Analog formula will be used to construct the spectrum.
**
**
**
**  Later version of GAMMA will have the functions “scale” and “sum”
**
**  in the library itself, so you will need to remove them from this
**
**  program in that event.
**
**
**
** Author:     S.A. Smith
**
** Date:       10/15/96
**
** Update:     10/15/96
**
** Version:    3.6
**
** Copyright:   S. Smith.  You can modify this program as you see fit
**
**          for personal use, but you must leave the program intact
**
**          if you re-distribute it.
**
**
**
*********************************************************************
*******************/

#include <gamma.h>                                  // Include GAMMA

void addW(row_vector& vx, double Fst, double Ffi, double F, double I)

    // Input     vx    : A row vector
    //           Fst   : Frequency of 1st point of vx (Hz)
    //           Ffi   : Frequency of last point of vx (Hz)
    //           F     : Transition frequency (Hz)
    //           I     : Transition intensity
    // Output    void  : The transition is added to the
    //                   row vector (as a Dirac delta).
    //Note             : To start one should zero vx

 {
 if(F<Fst || F>Ffi) return;                         // Insure its in range
 double Nm1 = double(vx.size()-1);                  // Freq. -> point conversion
 double m = Nm1/(Ffi-Fst);                          // Slope Freq. -> pt
 double dpt = m*(F-Fst);                            // Point index of F
 int pt = int(dpt);                                 // Main point for F
 double drem = dpt - pt;                            // Part which isn’t
 if(!drem) vx.put(vx.get(pt)+I, pt);                // Add if on a point
 else if(drem > 0)                                  // If in between points
  {                                                 // then just split it up
  vx.put(vx.get(pt)+(1.0-drem)*I, pt);              // between the two
  vx.put(vx.get(pt+1)+drem*I, pt+1);
  }
 else
  {
  vx.put(vx.get(pt)+(1.0+drem)*I, pt);
  vx.put(vx.get(pt-1)-drem*I, pt-1);
  }
 return;
 }


main (int argc, char* argv[])

 {
 cout << “\n\t\tG Central Transition Powder Pattern”;
```

```
 cout << "\n\t\t    (131Xe:3/2, 55Mn:5/2, 51V:7/2, ...)\n";
//                          First Make A G Interaction
 String Iso;                                        // Isotope of spin
 int qn=1;                                          // Guery index
 query_parameter(argc, argv, qn++,                  // Get the isotope type
  "\n\tIsotope Type [131Xe, 55Mn, 51V, ....]? ", Iso);
 double wG;                                         // Set Guad. frequency
 query_parameter(argc, argv, qn++,                  // Get the G coupling
     "\n\tG Frequency (kHz)? ", wG);
 wG *= 1.e3;                                        // Switch to Hz
 double eta;                                        // Set Guad. frequency
 query_parameter(argc, argv, qn++,                  // Get the G coupling
     "\n\tG eta Value [0, 1]? ", eta);
 double Om;
 query_parameter(argc, argv, qn++,                  // Get the field strength
            "\n\tLarmor Frequency (MHz)? ", Om);
 Om *= 1.e6;                                        // Switch to MHz
 Isotope S(Iso);                                    // Make a spin isotope
 double I = S.qn();                                 // This is isotope I value
 IntG G(I,wG2GCC(wG, I), eta);                      // Set a Guad interaction
 if(!int(2*I)%2)
   {
   cout << "\n\n\tSorry, I Must Be m*1/2, m Odd!\n\n";
   exit(-1);
   }
//                      Set Things Up For The Powder Average
 int npts = 4096;                                   // Block size
 int Ntheta, Nphi=0;                                // Angle increment counts
 query_parameter(argc, argv, qn++,                  // Get the theta increments
   "\n\t# Theta (z down) Increments Spanning [0, 180]? ", Ntheta);
 if(eta)
   query_parameter(argc, argv, qn++,                // Get the phi increments
     "\n\t# Phi (x over) Increments Spanning [0, 360)? ", Nphi);
 matrix ABC = G.wGcentral(Ntheta, Nphi);            // Prep. for 2nd order shifts
//                          Powder Averaging
//            Angle theta Is Down From The +z Axis, Angle phi Over From +x
// Note that since the 2nd order shift Wcentral(theta, phi) is symmetric with
// respect to both angles we need only average over parts of both angle ranges.
// For theta this means we sum the results from angles [0, 90) + half the result
// at 90.  Twice that sum would produce the total theta average over [0, 180].
// For phi we usually average [0, 360) so this is reduced to a sum over 3/4 the
// result at 0 + the results from angles (0, 90) + 1/2 the result at 90.  Four
// times that sum would produce the total phi average over [0, 360).

 double dthe, Nm1o2 = double(Ntheta-1.0)/2.0;      // For powder average
 double dphi, Nm2o4 = double(Nphi)/4.0;             // For powder average
 int theta, phi;                                    // Orientation angles
 double W, WG = G.wG();                             // Base Guad. frequency
 double Ifact = I*(I+1) - 0.75;                     // Part of the prefactor
 double prefact = -WG*WG*Ifact/Om;                  // Majority of the prefact
 double Aaxis = (-1.0/9.0)*prefact;                 // For plot scaling
 double Ctheta, Stheta, Cthetasq, Ctheta4;          // We'll need these
 double Fst = -2.5*Aaxis;                           // Starting plot limit
 double Ffi = 1.5*Aaxis;                            // End plot limit
 row_vector data(npts, complex0);                   // Array for spectrum
 for(theta=0; theta<Ntheta; theta++)                // Loop over theta angles
   {
   dthe = double(theta);
   if(dthe <= Nm1o2)                                // Only look upper half
     {                                              // of the sphere
     Ctheta = ABC.getRe(0,theta);                   // Scale factor cos(theta)
     Stheta = ABC.getRe(1,theta);                   // Scale factor sin(theta)
     Cthetasq = Ctheta*Ctheta;                      // cosine(theta)^2
     if(dthe == Nm1o2) Stheta *= 0.5;               // Half scale if theta=90
     if(!eta)                                       // Without eta, no phi
       {                                            // averaging is needed
       W=(prefact/16.)*(1.-Cthetasq)*(9.*Cthetasq-1.);  // Here is W adjustment
       addW(data, Fst, Ffi, W, Stheta);             // Add transition to spectrum
       }
     else
       {
       cout.flush();
       Ctheta4 = Cthetasq*Cthetasq;                 // cosine(theta)^4
       for(phi=0; phi<Nphi; phi++)                  // Loop over phi angles
         {
         dphi = double(phi);                        // Phi index as double
         if(dphi <= Nm2o4)                          // Only sum 1st quarter
           {
           if(!phi) Stheta *= 0.75;                 // 3/4 scale if phi=0
           else if(dphi == Nm2o4) Stheta *= 0.5;    // 1/2 scale if phi=90
           W = ABC.getRe(2,phi)*Ctheta4;            // A part of W
           W += ABC.getRe(3,phi)*Cthetasq;          // B part of W
           W += ABC.getRe(4,phi);                   // C part of W
           W *= (prefact/6.);                       // Scale
           addW(data, Fst, Ffi, W, Stheta);         // Add transition to spectrum
           }
         }
       }
     }
   }
 double lb = 40.0;                                  // Set a line broading factor
 cout << "\n\n\tDone With Discrete Powder Average. Processing...";
 cout.flush();
 data = IFFT(data);                                 // Put into time domain
 exponential_multiply(data,-lb);                    // Apodize the "FID"
 data = FFT(data);                                  // Put back into frequency domain
 GP_1D("spec.asc", data, 0, -2.5, 1.5);             // Output the points in ASCII
 GP_1Dplot("spec.gnu", "spec.asc");                 // Call Gnuplot and make plot now
 }
```

6 Electron G Interactions

6.1 Overview

The class IntG contains a fully functional G interaction defined for a single electron. The class allows for the definition and manipulation of such interactions, in particular it allows for the construction of oriented G Hamiltonians. Note that this class does keep track of the isotropic G component.

6.2 Available G Interaction Functions

6.3 G Interaction Theory

6.4 G Interaction Figures

6.5 Literature Comparisons

6.6 G Interaction Examples

6.20.1 Zero Field Transitions, First Order Spectra page 6-434

6.7 G Interaction Example Programs

6.8 G Interaction Constructors

6.8.1 IntG

Usage:

void IntG::IntG()
void IntG::IntG(const IntG& G1)
void IntG::IntG(double giso, double delzz, double eta=0.0, double theta=0.0, double phi=0.0)
void IntG::IntG(const coord& GC, double theta=0.0, double phi=0.0)
void IntG::IntG(ParameterAVLSet& pset, int idx=-1)

Description:

The function ***IntG*** is used to create a new G interaction.

1. IntG() - Called without arguments the function creates a NULL G interaction.
2. IntG(const IntG& G1) - Called with another G interaction, a new G interaction is constructed which is identical to the input interaction.
3. IntG(double giso, double delzz, double eta=0.0, double theta=0.0, double phi=0.0) - This will construct a new electron G interaction. The strength of the interaction is set by the arguments ***giso*** and ***delzz,*** both unitless quantities. The value of ***giso*** lis related to the trace of Gtensor. The value of ***delzz***, the G tensor anisotropy, . The value of ***eta*** can be optionally input and this set the asymmetry of the interaction. It is restricted to be within the range ***[0, 1]***. Two angles, ***theta*** and ***phi***, can be optionally specified in ***degrees***. This will set the orientation of the G interaction relative to its PAS in a standard spherical coordinate system.
4. IntG(const coord& GC, double theta=0, double phi=0) - This will construct a new electron G interaction from its three PAS Cartesian components which are contained in coordinate GC. The GC contains gxx, gyy, and gzz of the G tensor. Two angles, ***theta*** and ***phi***, can be optionally specified in ***degrees***. This will set the orientation of the G interaction relative to its PAS in a standard spherical coordinate system.
5. IntG(ParameterAVLSet& pset, int idx=-1) - This will construct a new G interaction for a spin having the quantum number ***qn*** from parameters found in the parameter set ***pset***. If the optional index ***idx*** has been set ***>=0*** the G parameters scanned in ***pset*** will be assumed to have a ***(idx)*** appended to their names.

Return Value:

Void. It is used strictly to create a G interaction.

Examples:

```
IntG G;                          // An empty G interaction.
IntG G1(1.5, 3.e5,.2, 45., 30.); // G. Int. for I=3/2, GCC=300kHz, η=.2, θ=45, ϕ=30
IntG G2(G1);                     // Another Guad. Interaction, here equal to G1
G = G2;                          // Now G is the same as G1 and G2
```

See Also: =, read, ask_read

6.8.2 =

Usage:

void IntG::operator= (const IntG& G1)

Description:

The operator = is assign one G interaction to another.

Return Value:

Void.

Example:

```
IntG G;                          // An empty G interaction.
IntG G1(1.5, 3.e5,.2, 45., 30.); // G. Int. for I=3/2, GCC=300kHz, η=.2, θ=45, ϕ=30
G = G1;                          // Now G is the same as G1
```

See Also: constructor, read, ask_read

6.9 Basic Functions

6.9.1 iso

Usage:

```
#include <IntG.h>
double IntG::iso () const
double IntG::iso (double giso) const
```

Description:

The function ***iso*** is used to either obtain or set the G interaction tensor isotropic value. With no arguments the function returns the value (unitless). If an argument, ***giso***, is specified then the isotropic value for the interaction is set. It is assumed that the input value of ***giso*** is unitless and is related to 1/3 the trace of the G tensor. Note that setting of ***giso*** will alter the (equivalent) value of the G tensor.

$$giso = e^2qQ = QCC = NQCC$$

Return Value:

Either void or a floating point number, double precision.

Example(s):

```
#include <IntG.h>
IntG G();                    // Empty G interaction.
G.delzz(100000.0);           // Set GCC to 100 KHz.
cout << G.delz ();           // Write coupling constant to std output.
```

See Also: GCC, NGCC, wG

6.9.2 delzz

Usage:

```
#include <IntG.h>
double IntG::delzz () const
double IntG::delz () const
double IntG::delzz (double dz) const
double IntG::delz (double dz) const
```

Description:

The function ***delzz*** is used to either obtain or set the interaction G coupling constant. With no arguments the function returns the coupling in Hz. If an argument, ***dz***, is specified then the coupling constant for the interaction is set. It is assumed that the input value of ***dz*** is in units of ***Hz***. The function is overloaded with the name ***delz*** for convenience. Note that setting of ***delzz*** will alter the (equivalent) value of the G coupling ***GCC*** (or ***NGCC***) as well as the G frequency.

$$\delta_{zz}^{G} = e^2qQ = QCC = NQCC$$

Return Value:

Either void or a floating point number, double precision.

Example(s):

```
#include <IntG.h>
IntG G();                    // Empty G interaction.
G.delzz(100000.0);           // Set GCC to 100 KHz.
cout << G.delz ();           // Write coupling constant to std output.
```

See Also: GCC, NGCC, wG

6.9.3 GCC, NGCC

Usage:

```
#include <IntG.h>
double IntG::GCC () const
double IntG::NGCC () const
double IntG::GCC (double dz) const
double IntG::NGCC (double dz) const
```

Description:

The function ***GCC*** is used to either obtain or set the interaction G coupling constant. With no arguments the function returns the coupling in Hz. If an argument, ***dz***, is specified then the coupling constant for the interaction is set. It is assumed that the input value of ***dz*** is in units of ***Hz***. The function is overloaded with the name ***NGCC*** for convenience. Note that setting of ***GCC*** will alter the (equivalent) value of the G spatial tensor ***delzz*** value as well as the G frequency. This function has identical functionality as ***delzz*** and ***delz***.

$$QCC = NQCC = e^2qQ = \delta_{zz}^{Q} = \frac{2I(2I-1)\omega^{Q}}{3}$$

Return Value:

Either void or a floating point number, double precision.

Examples:

```
#include <IntG.h>
IntG G();                    // Empty G interaction.
G.NGCC(100000.0);            // Set GCC to 100 KHz.
cout << G.GCC ();            // Write coupling constant to std output.
```

See Also: delz, delzz, wG

6.9.4 eta

Usage:

```
#include <IntG.h>
double IntG::eta () const
double IntG::eta (double Geta) const
```

Description:

The function ***eta*** is used to either obtain or set the G interaction asymmetry. With no arguments the function returns the asymmetry (unitless). If an argument, ***Geta***, is specified then the asymmetry for the interaction is set.The input value is **restricted to the range [0,1]** and is related to the G spatial tensor Cartesian components according to

$$\eta = (A_{xx} - A_{yy}) / A_{zz} \qquad |A_{zz}| \geq |A_{yy}| \geq |A_{xx}|$$

Note that setting ***eta*** will alter the 5 internal irreducible spherical spatial tensor components of the interaction.

Return Value:

Either void or a floating point number, double precision.

Examples:

```
#include <IntG.h>
IntG G();                    // Empty G interaction.
G.eta(0.75);                 // Set eta to 0.75.
double Geta = G.eta();       // Set Geta to current eta value
```

See Also: delz, delzz, wG

6.9.5 wG

Usage:

```
#include <IntG.h>
double IntG::wG () const
double IntG::wG (double W) const
```

Description:

The function ***wG*** is used to either obtain or set the interaction G frequency. With no arguments the function returns the frequency in Hz. If an argument, ***W***, is specified then the frequency for the interaction is set. It is assumed that the input value of ***W*** is in units of ***Hz***. In GAMMA the G frequency[1] is defined to be.

1. There are variations in the literature as to what the G frequency is. The definition in GAMMA is set such that the G interaction will split the observed NMR transitions by ω^Q when the Zeeman interaction is strong (i.e. high field, first-order G interaction). This definition is analogous to that of a scalar coupling.

$$\omega^Q = \frac{3e^2qQ}{2I(2I-1)} = \frac{3QCC}{2I(2I-1)} = \sqrt{\frac{15}{2\pi}}\xi^Q$$

Return Value:

Either void or a floating point number, double precision.

Examples:

```
#include <IntG.h>
IntG G();                    // Empty G interaction.
G.wG(1.4e5);                 // Set quad. frequency to 140 KHz.
cout << G.wG();              // Write frequency to std output.
```

See Also: delz, delzz, GCC, NGCC, xi

6.9.6 wG0, wGoriented

Usage:

```
double IntG::wGoriented() const
double IntG::wG0() const
double IntG::wGoriented(double theta, double phi) const
double IntG::wG0(double theta, double phi) const
matrix IntG::wGoriented(int Ntheta, int Nphi) const
matrix IntG::wG0(int Ntheta, int Nphi) const
```

Description:

The function ***wG0*** (or its equivalent ***wGoriented***) is used to obtain or generate the 1st order G frequency for a chosen orientation in ***Hz***. If the arguments, ***theta*** and ***phi***, are specified then the frequency will be returned at that orientation from the PAS rather that the internal orientation. It is assumed that the input angle values are in units of ***degrees***. In GAMMA the oriented G frequency[1] is defined to be

$$\omega^Q(\theta,\varphi) = \omega^Q(PAS)\cdot\sqrt{\frac{4\pi}{5}}A^Q_{2,0}(\theta,\varphi) = \frac{\omega^Q(PAS)}{2}[3\cos^2\theta - 1 + \eta\sin^2\theta\cos 2\varphi]$$

Alternatively, one may obtain an array of the mathematical precursors needed to generate the 1st order frequency over evenly spaced angle increments on the unit sphere. In this case the function is called with the integers ***Ntheta*** and ***Nphi***, the number of increments down and over respectively. The matrix returned will have 4 rows whose elements are given by

1. There are variations in the literature as to what the G frequency is. The definition in GAMMA is set such that the G interaction will split the observed NMR transitions by ω^Q when the Zeeman interaction is strong (i.e. high field, first-order G interaction). This definition is analogous to that of a scalar coupling.

$$\langle 0|mx|j\rangle = \frac{1}{2}(3\cos^2\theta_j - 1) \qquad \langle 1|mx|j\rangle = \frac{1}{2}\eta\sin^2\theta_j \qquad \langle 2|mx|j\rangle = \sin\theta_j$$

$$\langle 3|mx|j\rangle = \cos 2\varphi_j \qquad \theta_j = \frac{180\cdot j}{\text{Ntheta} - 1} \qquad \varphi_j = \frac{360\cdot j}{\text{Nphi}}$$

and the 1st order shifts reconstructed from

$$\omega^Q(\theta_i, \varphi_j) = \omega^Q(\langle 0|mx|i\rangle + \langle 1|mx|i\rangle\langle 3|mx|j\rangle)$$

Remember, these frequencies are the splittings between transitions to first order (high field approximation) for particular orientations. They are valid only when the Zeeman interaction is much stronger than the G interaction. One should use the second order frequency corrections when the Larmor frequency is only somewhat stronger than the G frequency. One should use the full treatment when the G interaction dominates.

Return Value:

Either void or a floating point number, double precision.

Examples:

```
IntG G();                    // Empty G interaction.
G.wG(1.4e5);                 // Set quad. frequency to 140 KHz.
cout << G.wG();              // Write frequency to std output.
```

See Also: delz, delzz, GCC, NGCC, xi

6.9.7 wGcentral

Usage:

double IntG::wGcentral(double Om) const
double IntG::wGcentral(double Om, double theta, double phi) const
matrix IntG::wGcentral(int Ntheta, int Nphi) const

Description:

The function ***wGcentral*** is used to obtain the interaction G frequency. The argument ***Om*** is used to indicate the Larmor frequency in ***Hz*** of the spin associated with the interaction. With no other arguments the shift will be that of the central transition at the interaction's internal orientation. With the additional arguments ***theta*** and ***phi*** the frequency will be the central transition second order shift at that orientation from the PAS rather that the internal orientation. It is assumed that the input angle values are in units of ***degrees***.

In GAMMA the 2nd order shifts to the central transition are given by

$$\omega^{Q,(2)}_{-\frac{1}{2},\frac{1}{2}}(\eta, \theta, \varphi) = \frac{-(\omega^Q)^2}{6\Omega}\left[I(I+1) - \frac{3}{4}\right][A(\eta, \varphi)\cos^4\theta + B(\eta, \varphi)\cos^2\theta + C(\eta, \varphi)]$$

where

$$A(\eta, \varphi) = \frac{-27}{8} + \frac{9}{4}\eta\cos(2\varphi) - \frac{3}{8}\eta^2\cos^2(2\varphi)$$

$$B(\eta, \varphi) = \frac{30}{8} - \frac{1}{2}\eta^2 - 2\eta\cos(2\varphi) + \frac{3}{4}\eta^2\cos^2(2\varphi)$$

$$C(\eta, \varphi) = \frac{-3}{8} + \frac{1}{3}\eta^2 - \frac{1}{4}\eta\cos(2\varphi) - \frac{3}{8}\eta^2\cos^2(2\varphi)$$

Alternatively, one may obtain an array of the mathematical precursors needed to generate the 2nd order shifts over evenly spaced angle increments on the unit sphere. In this case the function is called with the integers ***Ntheta*** and ***Nphi***, the number of increments down and over respectively. The matrix returned will have 5 rows whose elements are given by

$$\langle 0|mx|j\rangle = \cos\theta_j \qquad \langle 1|mx|j\rangle = \sin\theta_j$$

$$\langle 2|mx|j\rangle = A(\eta, \varphi_j) \qquad \langle 3|mx|j\rangle = B(\eta, \varphi_j) \qquad \langle 4|mx|j\rangle = C(\eta, \varphi_j)$$

$$\theta_j = \frac{180 \cdot j}{\text{Ntheta} - 1} \qquad \varphi_j = \frac{360 \cdot j}{\text{Nphi}}$$

and the shifts reconstructed from the previous equations.

Note that since second order effects are field dependent, the larger the field the smaller the returned shift(s). Also, the method of obtains such shifts in this function assumes that the G interaction is a perturbation to the Zeeman Hamiltonian. The will not be applicible when the G splitting is on the same scale as or larger than the Larmor frequency. Finally, if I is not half integer all values returned will be zero.

Return Value:

Either void or a floating point number, double precision.

Examples:

```
IntG G();                    // Empty G interaction.
G.wG(1.4e5);                 // Set quad. frequency to 140 KHz.
cout << G.wG();              // Write frequency to std output.
```

See Also: delz, delzz, GCC, NGCC, xi

6.9.8 wG1

Usage:

double IntG::wG1(double Om, double m) const
double IntG::wG1(double Om, double m, double theta, double phi) const
matrix IntG::wG1(int Ntheta, int Nphi) const

Description:

The function ***wG1*** is used to obtain the second order frequency shift of a G transition. The argument ***Om*** is used to indicate the Larmor frequency in ***Hz*** of the spin associated with the interaction. The value of ***m*** is the spin anglular momentum z quantum number and should span [I, I-1,I-2,.....-I+1]. The returned shift will be for the transtion between levels ***m*** and ***m-1***. With no addtional arguments the shift will be for the specified

transition at the interaction's internal orientation. With the additional arguments ***theta*** and ***phi*** the frequency will be the indicated transitions second order shift at that orientation from the PAS rather that the internal orientation. It is assumed that the input angle values are in units of ***degrees***.

In GAMMA the 2nd order shifts for the ***m***,***m-1*** transition are given by

$$\omega^{Q,(2)}_{m-1,m}(\eta,\theta,\varphi) = -\frac{\xi^2}{2\Omega_o}\{A^Q_{2,1}(\eta,\theta,\varphi)A^Q_{2,-1}(\eta,\theta,\varphi)[24m(m-1)-4I(I+1)+9]$$

$$+\frac{1}{2}A^Q_{2,2}(\eta,\theta,\varphi)A^Q_{2,-2}(\eta,\theta,\varphi)[12m(m-1)-4I(I+1)+6]\Big\}$$

Alternatively, one may obtain an array of the mathematical precursors needed to generate the 2nd order shifts over evenly spaced angle increments on the unit sphere. In this case the function is called with the integers ***Ntheta*** and ***Nphi***, the number of increments down and over respectively. The matrix returned will have 6 rows whose elements are given by

$$\langle 0|mx|j\rangle = 3\sqrt{\frac{5}{24\pi}}\sin\theta_j\cos\theta_j \qquad \langle 1|mx|j\rangle = \frac{3}{2}\sqrt{\frac{5}{24\pi}}\sin^2\theta_j$$

$$\langle 2|mx|j\rangle = -\eta\sqrt{\frac{5}{24\pi}}(\cos 2\varphi_j - i\sin 2\varphi_j) \qquad \langle 3|mx|j\rangle = \frac{\eta}{2}\sqrt{\frac{5}{24\pi}}[\cos 2\varphi_j - i2\sin 2\varphi_j]$$

$$\langle 4|mx|j\rangle = \sin\theta_j \qquad \langle 5|mx|j\rangle = \cos\theta_j$$

$$\theta_j = \frac{180\cdot j}{\text{Ntheta}-1} \qquad \varphi_j = \frac{360\cdot j}{\text{Nphi}}$$

Reconstruction of full $A^Q_{2,m}(\theta,\varphi)$ values is based on

$$A^Q_{2,1}(\theta,\varphi) = A^Q_{2,1}(\theta,\varphi)\Big|_{\eta=0} + \sin\theta\cos\theta Re(A^Q_{2,1}B(\varphi)) + i\sin\theta Im(A^Q_{2,1}B(\varphi))$$

$$A^Q_{2,2}(\theta,\varphi) = A^Q_{2,2}(\theta,\varphi)\Big|_{\eta=0} + (1+\cos^2\theta)Re(A^Q_{2,2}B(\varphi)) + i\cos\theta Im(A^Q_{2,2}B(\varphi))$$

Required $A^Q_{2,m}(\theta_k,\varphi_l)$ components can be reconstructed according to the discrete equations below.

$$A^Q_{2,1}(\theta_k,\varphi_l) = \langle 0|mx|k\rangle + \langle 4|mx|k\rangle[\langle 5|mx|k\rangle Re\langle 2|mx|l\rangle + iIm\langle 2|mx|l\rangle]$$

$$A^Q_{2,2}(\theta_k,\varphi_l) = \langle 1|mx|k\rangle + (1+\langle 5|mx|k\rangle^2)Re\langle 3|mx|l\rangle + i\langle 5|mx|k\rangle Im\langle 3|mx|l\rangle$$

and the frequencies subsequently generated using

$$A^Q_{2,1}A^Q_{2,-1} = -A^Q_{2,1}A^{Q\,*}_{2,1} \qquad A^Q_{2,2}A^Q_{2,-2} = A^Q_{2,2}A^{Q\,*}_{2,2}$$

Note that since second order effects are field dependent, the larger the field the smaller the returned shift(s). Also, the method of obtains such shifts in this function assumes that the G interaction is a perturbation to the Zeeman Hamiltonian. The will not be applicible when the G splitting is on the same scale as or larger than

the Larmor frequency. Finally, if I is not half integer all values returned will be zero.

Return Value:

Either void or a floating point number, double precision.

Examples:

```
IntG G();                    // Empty G interaction.
G.wG(1.4e5);                 // Set quad. frequency to 140 KHz.
cout << G.wG();              // Write frequency to std output.
```

See Also: delz, delzz, GCC, NGCC, xi

6.9.9 xi

Usage:

double IntG::xi() const

Description:

The function ***xi*** is used to either obtain the GAMMA defined G interaction constant. The constant is used to scale the interaction such that both its spatial and spin tensors are “independent” of the interaction type.

$$\xi^Q = \sqrt{\frac{6\pi}{5}}\frac{e^2qQ}{2I(2I-1)} = \sqrt{\frac{6\pi}{5}}\frac{QCC}{2I(2I-1)} = \sqrt{\frac{2\pi}{15}}\omega^Q$$

This will be used in the formulation of G Hamiltonians according to.

$$\boldsymbol{H}^Q(\theta,\varphi) = \xi^Q\sum_{m}^{\pm 2}(-1)^m A^Q_{2,-m}(\theta,\varphi)\bullet\boldsymbol{T}^Q_{2,m}$$

Return Value:

A floating point number, double precision.

Examples:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
double Xi = G.xi();          // Get quad. interaction constant.
```

6.10 Spherical Spatial Tensor Functions

6.10.1 A0, A20

Usage:

```
#include <IntG.h>
complex IntG::A0() const
complex IntG::A20() const
complex IntG::A0(double theta, double phi) const
complex IntG::A20(double theta, double phi) const
```

Description:

The functions ***A0*** and ***A20*** are used to obtain the G interaction spatial tensor component $A_{2,0}$. If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments ***theta*** and ***phi*** are given the returned tensor component is for the orientation at ***theta*** degrees down from the interactions PAS z-axis and ***phi*** degrees over from the interactions PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***Hz***.

$$A^{Q}_{2,\,0}(\theta,\,\varphi) \;=\; \sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]$$

Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type[1]. This component can also be related to the Cartesian tensor components for any arbitrary orientation.

$$A_{2,\,0} \;=\; \sqrt{6}[3A_{zz} - Tr\{A\}]$$

Return Value:

A complex number.

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
complex A20 = G.A20();              // This is at theta=phi=45 degrees
cout << G.A20(15.6, 99.3);          // This is at theta=15.6 and phi=99.3 degrees.
```

1. Because the GAMMA platform accommodates different interaction types, the scaling on all spatial tensors is chosen to be independent of the interaction. Rather, the spatial tensors are related directly to the familiar rank two spherical harmonics $A^{Q}_{2,\,m}(\theta,\,\varphi)\Big|_{\eta\,=\,0} \;=\; Y^{2}_{m}(\theta,\,\varphi)$. Also, the sign on the term(s) involving η will have opposite sign if the common alternative definition of the PAS orientation ($|A_{zz}| \geq |A_{xx}| \geq |A_{yy}|$) is used rather that the definition used in GAMMA ($|A_{zz}| \geq |A_{yy}| \geq |A_{xx}|$)

See Also: A1, A21, Am1, A2m1, A2, A22, Am2, A2m2

6.10.2 A1, A21

Usage:

```
#include <IntG.h>
complex IntG::A1() const
complex IntG::A21() const
complex IntG::A1(double theta, double phi) const
complex IntG::A21(double theta, double phi) const
```

Description:

The functions ***A1*** and ***A21*** are used to obtain the G interaction spatial tensor component $A_{2,1}$. If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments ***theta*** and ***phi*** are given the returned tensor component is for the orientation at ***theta*** degrees down from the interactions PAS z-axis and ***phi*** degrees over from the interactions PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***Hz***.

$$A^{Q}_{2,1}(\theta, \varphi) = \sqrt{\frac{5}{24\pi}}\sin\theta[3\cos\theta - \eta(\cos\theta\cos 2\varphi - i\sin 2\varphi)]$$

Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type[1]. This component can also be related to the Cartesian tensor components for any arbitrary orientation.

$$A_{21} = -\frac{1}{2}[A_{xz} + A_{zx} + i(A_{yz} + A_{zy})]$$

Return Value:

A complex number.

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
complex A20 = G.A20();              // This is at theta=phi=45 degrees
cout << G.A20(15.6, 99.3);          // This is at theta=15.6 and phi=99.3 degrees.
```

1. Because the GAMMA platform accommodates different interaction types, the scaling on all spatial tensors is chosen to be independent of the interaction. Rather, the spatial tensors are related directly to the familiar rank two spherical harmonics $A^{Q}_{2,m}(\theta, \varphi)\Big|_{\eta = 0} = Y^{2}_{m}(\theta, \varphi)$. Also, the sign on the term(s) involving η will have opposite sign if the common alternative definition of the PAS orientation ($|A_{zz}| \geq |A_{xx}| \geq |A_{yy}|$) is used rather that the definition used in GAMMA ($|A_{zz}| \geq |A_{yy}| \geq |A_{xx}|$)

See Also: A1, A21, Am1, A2m1, A2, A22, Am2, A2m2

6.10.3 Am1, A2m1

Usage:

```
#include <IntG.h>
complex IntG::Am1() const
complex IntG::A2m1() const
complex IntG::Am1(double theta, double phi) const
complex IntG::Am21(double theta, double phi) const
```

Description:

The functions ***Am1*** and ***A2m1*** are used to obtain the G interaction spatial tensor component $A_{2,1}$. If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments ***theta*** and ***phi*** are given the returned tensor component is for the orientation at ***theta*** degrees down from the interactions PAS z-axis and ***phi*** degrees over from the interactions PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***Hz***.

$$A^Q_{2,-1}(\theta,\varphi) = -\sqrt{\frac{5}{24\pi}}\sin\theta[3\cos\theta - \eta(\cos\theta\cos 2\varphi + i\sin 2\varphi)] = -A^{Q\,*}_{2,1}(\theta,\varphi)$$

Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type[1]. This component can also be related to the Cartesian tensor components for any arbitrary orientation.

$$A_{2,-1} = \frac{1}{2}[A_{xz} + A_{zx} + i(A_{yz} - A_{zy})]$$

Return Value:

A complex number.

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
complex A20 = G.A20();              // This is at theta=phi=45 degrees
cout << G.A20(15.6, 99.3);          // This is at theta=15.6 and phi=99.3 degrees.
```

1. Because the GAMMA platform accommodates different interaction types, the scaling on all spatial tensors is chosen to be independent of the interaction. Rather, the spatial tensors are related directly to the familiar rank two spherical harmonics $A^Q_{2,m}(\theta,\varphi)\Big|_{\eta=0} = Y^2_m(\theta,\varphi)$. Also, the sign on the term(s) involving η will have opposite sign if the common alternative definition of the PAS orientation ($|A_{zz}| \geq |A_{xx}| \geq |A_{yy}|$) is used rather that the definition used in GAMMA ($|A_{zz}| \geq |A_{yy}| \geq |A_{xx}|$)

See Also: A1, A21, Am1, A2m1, A2, A22, Am2, A2m2

6.10.4 A2, A22

Usage:

```
#include <IntG.h>
complex IntG::A2() const
complex IntG::A22() const
complex IntG::A2(double theta, double phi) const
complex IntG::A22(double theta, double phi) const
```

Description:

The functions ***A2*** and ***A22*** are used to obtain the G interaction spatial tensor component $A_{2,1}$. If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments ***theta*** and ***phi*** are given the returned tensor component is for the orientation at ***theta*** degrees down from the interactions PAS z-axis and ***phi*** degrees over from the interactions PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***Hz***.

$$A^{Q}_{2,\,2}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}\frac{1}{2}[3\sin^2\theta + \eta[\cos 2\varphi(1+\cos^2\theta) - i2\sin 2\varphi\cos\theta]]$$

Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type[1]. This component can also be related to the Cartesian tensor components for any arbitrary orientation.

$$A_{2,\,2} = \frac{1}{2}[A_{xx} - A_{yy} + i(A_{xy} + A_{yx})]$$

Return Value:

A complex number.

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
complex A20 = G.A20();              // This is at theta=phi=45 degrees
cout << G.A20(15.6, 99.3);          // This is at theta=15.6 and phi=99.3 degrees.
```

1. Because the GAMMA platform accommodates different interaction types, the scaling on all spatial tensors is chosen to be independent of the interaction. Rather, the spatial tensors are related directly to the familiar rank two spherical harmonics $A^{Q}_{2,\,m}(\theta,\varphi)\Big|_{\eta=0} = Y^2_m(\theta,\varphi)$. Also, the sign on the term(s) involving η will have opposite sign if the common alternative definition of the PAS orientation ($|A_{zz}| \geq |A_{xx}| \geq |A_{yy}|$) is used rather that the definition used in GAMMA ($|A_{zz}| \geq |A_{yy}| \geq |A_{xx}|$)

See Also: A1, A21, Am1, A2m1, A2, A22, Am2, A2m2

6.10.5 Am2, A2m2

Usage:

```
#include <IntG.h>
complex IntG::Am2() const
complex IntG::A2m2() const
complex IntG::Am2(double theta, double phi) const
complex IntG::A2m2(double theta, double phi) const
```

Description:

The functions ***Am2*** and ***A2m2*** are used to obtain the G interaction spatial tensor component $A_{2,-2}$. If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments ***theta*** and ***phi*** are given the returned tensor component is for the orientation at ***theta*** degrees down from the interactions PAS z-axis and ***phi*** degrees over from the interactions PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***degrees***.

$$A^Q_{2,-2}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}\frac{1}{2}[3\sin^2\theta + \eta[\cos 2\varphi(1+\cos^2\theta) + i2\sin 2\varphi\cos\theta]] = A^{Q\,*}_{2,2}(\theta,\varphi)$$

Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type[1]. This component can also be related to the Cartesian tensor components for any arbitrary orientation.

$$A_{2,-2} = \frac{1}{2}[A_{xx} + (-A_{yy}) - i(A_{xy} + A_{yx})]$$

Return Value:

A complex number.

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
complex A20 = G.A20();              // This is at theta=phi=45 degrees
cout << G.A20(15.6, 99.3);          // This is at theta=15.6 and phi=99.3 degrees.
```

See Also: A1, A21, Am1, A2m1, A2, A22, Am2, A2m2

1. Because the GAMMA platform accommodates different interaction types, the scaling on all spatial tensors is chosen to be independent of the interaction. Rather, the spatial tensors are related directly to the familiar rank two spherical harmonics $A^Q_{2,m}(\theta,\varphi)\big|_{\eta=0} = Y^2_m(\theta,\varphi)$. Also, the sign on the term(s) involving η will have opposite sign if the common alternative definition of the PAS orientation ($|A_{zz}| \geq |A_{xx}| \geq |A_{yy}|$) is used rather that the definition used in GAMMA ($|A_{zz}| \geq |A_{yy}| \geq |A_{xx}|$)

6.11 Cartesian Spatial Tensor Functions

6.11.1 Axx

Usage:

```
#include <IntG.h>
complex IntG::Axx() const
complex IntG::Axx(double theta, double phi) const
```

Description:

The functions ***Axx*** is used to obtain the G interaction spatial tensor component A_{xx}. If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments ***theta*** and ***phi*** are given the returned tensor component is for the orientation at ***theta*** degrees down from the interactions PAS z-axis and ***phi*** degrees over from the interactions PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***Hz***.

$$A_{xx}(\theta, \varphi) = \sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]$$

Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type. This component can also be related to the spherical tensor components for any arbitrary orientation.

$$A_{xx} = \frac{1}{2}(A_{2,2} + A_{2,-2}) - \frac{1}{\sqrt{6}}A_{2,0}$$

Return Value:

A complex number.

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
complex A20 = G.A20();              // This is at theta=phi=45 degrees
cout << G.A20(15.6, 99.3);          // This is at theta=15.6 and phi=99.3 degrees.
```

See Also: Ayy, Azz, Axy, Axz, Ayx, Ayz, Azx, Azy

6.11.2 Ayy

Usage:

```
#include <IntG.h>
complex IntG::Ayy() const
complex IntG::Axx(double theta, double phi) const
```

Description:

The functions ***Ayy*** is used to obtain the G interaction spatial tensor component A_{yy}. If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments

theta and ***phi*** are given the returned tensor component is for the orientation at ***theta*** degrees down from the interactions PAS z-axis and ***phi*** degrees over from the interactions PAS x-a!a

567s. The values of ***theta*** and ***phi*** are assumed in ***Hz***.

$$A_{yy}(\theta, \varphi) = -\sqrt{\frac{5}{24\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]$$

Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type. This component can also be related to the spherical tensor components for any arbitrary orientation.

$$A_{yy} = \frac{-1}{2}(A_{2,2} + A_{2,-2}) - \frac{1}{\sqrt{6}}A_{2,0}$$

Return Value:

A complex number.

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
complex A20 = G.A20();              // This is at theta=phi=45 degrees
cout << G.A20(15.6, 99.3);          // This is at theta=15.6 and phi=99.3 degrees.
```

See Also: Axx, Azz, Axy, Axz, Ayx, Ayz, Azx, Azy

6.11.3 Azz

Usage:

```
#include <IntG.h>
complex IntG::Azz() const
complex IntG::Azz(double theta, double phi) const
```

Description:

The functions ***Azz*** is used to obtain the G interaction spatial tensor component A_{zz}. If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments ***theta*** and ***phi*** are given the returned tensor component is for the orientation at ***theta*** degrees down from the interactions PAS z-axis and ***phi*** degrees over from the interactions PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***Hz***.

$$A_{zz}(\theta, \varphi) = \sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]$$

Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type. This component can also be related to the spherical tensor components for any arbitrary orientation.

$$A_{zz} = \sqrt{\frac{2}{3}}A_{2,0}$$

Return Value:

A complex number.

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
complex A20 = G.A20();          // This is at theta=phi=45 degrees
cout << G.A20(15.6, 99.3);      // This is at theta=15.6 and phi=99.3 degrees.
```

See Also: Axx, Ayy, Axy, A2xz, Ayx, Ayz, Azx, Azy

6.11.4 Axy, Ayx

Usage:

```
#include <IntG.h>
complex IntG::Axy() const
complex IntG::Axy(double theta, double phi) const
complex IntG::Ayx() const
complex IntG::Ayx(double theta, double phi) const
```

Description:

The functions ***Axy*** and ***Ayx*** are used to obtain the G interaction spatial tensor component $A_{xy} = A_{yx}$. If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments ***theta*** and ***phi*** are given the returned tensor component is for the orientation at ***theta*** degrees down from the interactions PAS z-axis and ***phi*** degrees over from the interactions PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***Hz***.

$$A_{xy}(\theta, \varphi) = \sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]$$

Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type. This component can also be related to the spherical tensor components for any arbitrary orientation.

$$A_{xy} = -\frac{i}{2}(A_{2,2} - A_{2,-2}) = A_{yx}$$

Return Value:

A complex number.

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
complex A20 = G.A20();          // This is at theta=phi=45 degrees
cout << G.A20(15.6, 99.3);      // This is at theta=15.6 and phi=99.3 degrees.
```

See Also: Axx, Ayy, Azz, Axz, Ayz, Azx, Azy

6.11.5 Axz, Azx

Usage:

```
#include <IntG.h>
complex IntG::Axz() const
complex IntG::Axz(double theta, double phi) const
complex IntG::Azx() const
complex IntG::Azx(double theta, double phi) const
```

Description:

The functions ***Axz*** and ***Azx*** are used to obtain the G interaction spatial tensor component $A_{xz} = A_{zx}$. If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments ***theta*** and ***phi*** are given the returned tensor component is for the orientation at ***theta*** degrees down from the interactions PAS z-axis and ***phi*** degrees over from the interactions PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***Hz***.

$$A_{xy}(\theta, \varphi) = \sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]$$

Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type. This component can also be related to the spherical tensor components for any arbitrary orientation.

$$A_{xz} = -\frac{1}{2}[(A_{2,1} - A_{2,-1})] = A_{zx}$$

Return Value:

A complex number.

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
complex A20 = G.A20();              // This is at theta=phi=45 degrees
cout << G.A20(15.6, 99.3);          // This is at theta=15.6 and phi=99.3 degrees.
```

See Also: Axx, Ayy, Azz, Axz, Ayz, Azx, Azy

6.11.6 Ayz, Azy

Usage:

```
#include <IntG.h>
complex IntG::Ayz() const
complex IntG::Ayz(double theta, double phi) const
complex IntG::Azy() const
complex IntG::Azy(double theta, double phi) const
```

Description:

The functions ***Ayz*** and ***Azy*** are used to obtain the G interaction spatial tensor component $A_{yz} = A_{zy}$. If no arguments are given the functions return the value of the tensor component at the current interaction orientation. If the arguments ***theta*** and ***phi*** are given the returned tensor component is for the orientation at ***theta*** degrees down from the interactions PAS z-axis and ***phi*** degrees over from the interactions PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***Hz***.

$$A_{xy}(\theta, \varphi) = \sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]$$

Note that GAMMA uses a scaling on all spatial tensor components which is independent of the interaction type. This component can also be related to the spherical tensor components for any arbitrary orientation.

$$A_{yz} = \frac{i}{2}[(A_{2,1} + A_{2,-1})] = A_{zy}$$

Return Value:

A complex number.

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
complex A20 = G.A20();            // This is at theta=phi=45 degrees
cout << G.A20(15.6, 99.3);        // This is at theta=15.6 and phi=99.3 degrees.
```

See Also: Axx, Ayy, Azz, Axz, Ayz, Azx, Azy

6.12 Powder Average Facilitator Functions

6.12.1 A0A, A20A

Usage:

row_vector IntG::A0A(int Ntheta)
row_vector IntG::A20A(int Ntheta)

Description:

The functions ***A0A*** and ***A20A*** are equivalent. They are used to obtain part of G interaction spatial tensor component $A_{2,0}$ for a series of evenly incrmented θ values.

$$A_{2,0}A(\theta) = \sqrt{\frac{5}{16\pi}}(3\cos^2\theta - 1) = A^Q_{2,0}(\theta, \varphi)\Big|_{\eta = \varphi = 0}$$

Given a number of angle increments, ***Ntheta***, a row vector of dimension ***Ntheta*** will be returned which contains the η independent terms of $A^Q_{2,0}$ at evenly spaced increments of θ starting at the +z PAS ($\theta = 0$) alignment and finishing at -z PAS ($\theta = 180$) alignment.

$$\langle v|i\rangle = A^Q_{2,0}A(\theta_i) \qquad \theta_i = \frac{180i}{(\text{Ntheta} - 1)}$$

Note that to obtain the full $A^Q_{2,0}$ terms (if they are η dependent) they must be properly combined with the values from the function ***A20B.***

Return Value:

A vector.

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
row_vector A20s = G.A20A(720);    // Get 720 A20A values spanning [0, 180]
```

See Also: A21A, A22A, A20B, A21B, A22B, A2As, A2Bs, A2s

6.12.2 A1A, A21A

Usage:

row_vector IntG::A1A(int Ntheta)
row_vector IntG::A21A(int Ntheta)

Description:

The functions ***A1A*** and ***A21A*** are equivalent. They are used to obtain part of G interaction spatial tensor com-

ponent $A^Q_{2,1}$ for a series of evenly incrmented θ values.

$$A^Q_{2,1}A(\theta) = 3\sqrt{\frac{5}{24\pi}}\sin\theta\cos\theta = A^Q_{2,1}(\theta,\varphi)\Big|_{\eta=0}$$

Given a number of angle increments, ***Ntheta***, a row vector of dimension ***Ntheta*** will be returned which contains the η independent terms of $A^Q_{2,1}$ at evenly spaced increments of θ starting at the +z PAS ($\theta = 0$) alignment and finishing at -z PAS ($\theta = 180$) alignment.

$$\langle v|i\rangle = A^Q_{2,1}A(\theta_i) \qquad \theta_i = \frac{180i}{(\mathrm{Ntheta}-1)}$$

Note that to obtain the full $A^Q_{2,1}$ terms (if they are η dependent) they must be properly combined with the values from the function ***A21B.***

Return Value:

A vector.

Example:

```
IntG G(1.5, 3.e5, 0.2);              // Make a G interaction.
row_vector A21s = G.A21A(181);       // Get 181 A20A values spanning [0, 180]
```

See Also: A20A, A22A, A20B, A21B, A22B, A2As, A2Bs, A2s

6.12.3 A2A, A221A

Usage:

```
row_vector IntG::A2A(int Ntheta)
row_vector IntG::A22A(int Ntheta)
```

Description:

The functions ***A2A*** and ***A22A*** are equivalent. They are used to obtain part of G interaction spatial tensor component $A^Q_{2,2}$ for a series of evenly incremented θ values.

$$A^Q_{2,2}A(\theta) = \frac{3}{2}\sqrt{\frac{5}{24\pi}}\sin^2\theta = 3\sqrt{\frac{5}{96\pi}}\sin^2\theta = A^Q_{2,2}(\theta,\varphi)\Big|_{\eta=0}$$

Given a number of angle increments, ***Ntheta***, a row vector of dimension ***Ntheta*** will be returned which contains the η independent terms of $A^Q_{2,2}$ at evenly spaced increments of θ starting at the +z PAS ($\theta = 0$) alignment and finishing at -z PAS ($\theta = 180$) alignment.

$$\langle v|i\rangle = A^Q_{2,2}A(\theta_i) \qquad \theta_i = \frac{180i}{(\text{Ntheta} - 1)}$$

Note that to obtain the full $A^Q_{2,2}$ terms (if they are η dependent) they must be properly combined with the values from the function ***A22B.***

Return Value:

A vector.

Example:

```
IntG G(1.5, 3.e5, 0.2);              // Make a G interaction.
row_vector A22s = G.A22A(181);       // Get 181 A22A values spanning [0, 180]
```

See Also: A20A, A21A, A20B, A21B, A22B, A2As, A2Bs, A2s

6.12.4 A0B, A20B

Usage:

```
row_vector IntG::A0B(int Nphi)
row_vector IntG::A20B(int Nphi)
```

Description:

The functions ***A0B*** and ***A20B*** are equivalent. They are used to obtain part of G interaction spatial tensor component $A^Q_{2,0}$ for a series of evenly incrmented φ values.

$$A^Q_{2,0}B(\varphi) = \sqrt{\frac{5}{16\pi}}\eta\cos 2\varphi = \frac{1}{\sin^2\theta}\left[A^Q_{2,0}(\theta,\varphi) - A^Q_{2,0}(\theta,\varphi)\Big|_{\eta=0}\right]$$

Given a number of angle increments, ***Nphi***, a row vector of dimension ***Nphi*** will be returned which contains θ independent terms of $A^Q_{2,0}$ at evenly spaced increments of φ starting at the +x PAS ($\varphi = 0$) alignment and finishing at +x PAS ($\varphi = 360$) alignment.

$$\langle v|i\rangle = A^Q_{2,0}A(\varphi_i) \qquad \varphi_i = \frac{360i}{\text{Nphi}}$$

Note that to obtain the full $A^Q_{2,0}$ terms they must be properly combined with the values from the function ***A20A.***

$$A^Q_{2,0}(\theta,\varphi) = \sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]$$

$$A^Q_{2,1}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}\sin\theta[3\cos\theta - \eta(\cos\theta\cos 2\varphi - i\sin 2\varphi)] = -A^{Q\,*}_{2,-1}(\theta,\varphi)$$

$$A^Q_{2,2}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}\frac{1}{2}[3\sin^2\theta + \eta[\cos 2\varphi(1+\cos^2\theta) - i2\sin 2\varphi\cos\theta]] = A^{Q\,*}_{2,-2}(\theta,\varphi)$$

Return Value:

A vector.

Example:

```
IntG G(1.5, 3.e5, 0.2);// Make a G interaction.
row_vector A20s = G.A20B(120);    // Get 120 A20B values spanning [0, 360)
```

See Also: A20A, A21A, A22A, A21B, A22B, A2As, A2Bs, A2s

6.12.5 A1B, A21B

Usage:

```
row_vector IntG::A1B(int Nphi)
row_vector IntG::A21B(int Nphi)
```

Description:

The functions ***A1B*** and ***A21B*** are equivalent. They are used to obtain part of G interaction spatial tensor component $A^Q_{2,1}$ for a series of evenly incrmented φ values.

$$A^Q_{2,1}B(\varphi) = -\sqrt{\frac{5}{24\pi}}\eta(\cos 2\varphi - i\sin 2\varphi)$$

where

$$A^Q_{2,1}(\theta,\varphi) = \sin\theta\cos\theta Re(A^Q_{2,1}B(\varphi)) + i\sin\theta Im(A^Q_{2,1}B(\varphi)) + A^Q_{2,1}(\theta,\varphi)\Big|_{\eta=0}$$

Given a number of angle increments, ***Nphi***, a row vector of dimension ***Nphi*** will be returned which contains θ independent terms of $A^Q_{2,1}$ at evenly spaced increments of φ starting at the +x PAS ($\varphi = 0$) alignment and finishing at +x PAS ($\varphi = 360$) alignment.

$$\langle v|i\rangle = A^Q_{2,1}A(\varphi_i) \qquad \varphi_i = \frac{360i}{\text{Nphi}}$$

Note that to obtain the full $A^Q_{2,1}$ terms they must be properly combined with the values from the function

A21A.

Return Value:

A vector.

Example:

```
IntG G(1.5, 3.e5, 0.2);// Make a G interaction.
row_vector A21s = G.A21B(120);    // Get 120 A21B values spanning [0, 360)
```

See Also: A20A, A21A, A22A, A20B, A22B, A2As, A2Bs, A2s

6.12.6 A2B, A22B

Usage:

```
row_vector IntG::A2B(int Nphi)
row_vector IntG::A22B(int Nphi)
```

Description:

The functions ***A1B*** and ***A21B*** are equivalent. They are used to obtain part of G interaction spatial tensor component $A^Q_{2,2}$ for a series of evenly incrmented φ values.

$$A^Q_{2,2}B(\varphi) = \sqrt{\frac{5}{96\pi}}\eta[\cos 2\varphi - i2\sin 2\varphi]$$

where

$$A^Q_{2,2}(\theta,\varphi) = (1+\cos^2\theta)Re(A^Q_{2,2}B(\varphi)) + i\cos\theta Im(A^Q_{2,2}B(\varphi)) + A^Q_{2,2}(\theta,\varphi)\Big|_{\eta=0}$$

Given a number of angle increments, ***Nphi***, a row vector of dimension ***Nphi*** will be returned which contains θ independent terms of $A^Q_{2,2}$ at evenly spaced increments of φ starting at the +x PAS ($\varphi = 0$) alignment and finishing at +x PAS ($\varphi = 360$) alignment.

$$\langle v|i\rangle = A^Q_{2,2}A(\varphi_i) \qquad \varphi_i = \frac{360i}{\text{Nphi}}$$

Note that to obtain the full $A^Q_{2,2}$ terms they must be properly combined with the values from the function ***A22A.***

Return Value:

A vector.

Example:

```
IntG G(1.5, 3.e5, 0.2);                // Make a G interaction.
row_vector A22s = G.A22B(120);    // Get 120 A22B values spanning [0, 360)
```

See Also: A20A, A21A, A22A, A20B, A21B, A2As, A2Bs, A2s

6.12.7 A2s

Usage:

matrix IntG::A2s(int Ntheta, int Nphi)

Description:

The function ***A2s*** is used to construct the G interaction spatial tensor components $A^Q_{2,m}$ for a series of evenly incrmented θ and φ values. Given arguments for the number of angle increments, ***Ntheta*** and ***Nphi*** the function will return a matrix of dimenstion (8 x nc) where nc is the larger of the two input arguments. The matrix columns, indexed by j, will then correspond either to an angle θ or an angle φ where

$$\theta_j = \frac{180j}{(\text{Ntheta} - 1)} \qquad \varphi_j = \frac{360j}{\text{Nphi}}$$

depending upon which row is being accessed. Rows 0-2 of the array will correspond to the the η independent terms of $A^Q_{2,\{0,1,2\}}$ at evenly spaced increments of θ starting at the +z PAS ($\theta = 0$) alignment and finishing at -z PAS ($\theta = 180$) alignment. Rows 3-5 of the array will correspond to θ independent parts of the interaction spatial tensor components $A^Q_{2,\{0,1,2\}}$ at evenly spaced increments of φ starting at the +x PAS ($\varphi = 0$) alignment and finishing at +x PAS ($\varphi = 360$) alignment. The final three array columns will contain θ dependent terms that are used to blend with the other rows to form the full $A^Q_{2,m}(\theta, \varphi)$ values. Reconstruction of full $A^Q_{2,m}(\theta, \varphi)$ values is based on

$$A^Q_{2,0}(\theta, \varphi) = A^Q_{2,0}(\theta, \varphi)\Big|_{\eta = 0} + \sin^2\theta A^Q_{2,0} B(\varphi)$$

$$A^Q_{2,1}(\theta, \varphi) = A^Q_{2,1}(\theta, \varphi)\Big|_{\eta = 0} + \sin\theta\cos\theta Re(A^Q_{2,1}B(\varphi)) + i\sin\theta Im(A^Q_{2,1}B(\varphi))$$

$$A^Q_{2,2}(\theta, \varphi) = A^Q_{2,2}(\theta, \varphi)\Big|_{\eta = 0} + (1 + \cos^2\theta) Re(A^Q_{2,2}B(\varphi)) + i\cos\theta Im(A^Q_{2,2}B(\varphi))$$

A particular $A^Q_{2,m}(\theta_k, \varphi_l)$ can be reconstructed according to the analogous discrete equations.

$$A^Q_{2,0}(\theta_k, \varphi_l) = \langle 0|mx|k\rangle + \langle 6|mx|k\rangle^2 \langle 3|mx|l\rangle$$

$$A^Q_{2,1}(\theta_k, \varphi_l) = \langle 1|mx|k\rangle + \langle 6|mx|k\rangle[\langle 7|mx|k\rangle Re\langle 4|mx|l\rangle + iIm\langle 4|mx|l\rangle]$$

$$A^Q_{2,2}(\theta_k, \varphi_l) = \langle 2|mx|k\rangle + (1 + \langle 7|mx|k\rangle^2) Re\langle 5|mx|l\rangle + i\langle 7|mx|k\rangle Im\langle 5|mx|l\rangle$$

The components with m negative are obtained from the relationship .

$$A^{Q}_{2,\,-m} = (-1)^{m} A^{Q}_{2,\,m}$$

Return Value:

An array.

Example:

```
IntG G(1.5, 3.e5, 0.2);          // Make a G interaction.
matrix As = G.A2x(720, 360);     // Get array for values spanning [0, 180] & [0, 360)
```

See Also: A20A, A21A, A22A, A20B, A21B, A2As, A2Bs, A2s

6.13 Spin Tensor Functions

6.13.1 Tcomp

Usage:

```
#include <IntG.h>
matrix IntG::Tcomp(int comp)
```

Description:

The function ***Tcomp*** is used to obtain a G interaction spin tensor component. The component desired is specified by the argument ***comp*** which relates to the m value as follows:

comp:	0	1	2	3	4
$T^G_{2,m}$:	$T^G_{2,0}$	$T^G_{2,1}$	$T^G_{2,-1}$	$T^G_{2,2}$	$T^G_{2,-2}$

The spin components are given

$$T^Q_{2,0} = \frac{1}{\sqrt{6}}[3I_z^2 - \vec{I}^2] = \frac{1}{\sqrt{6}}[3I_z^2 - I(I+1)]$$

$$T^Q_{2,\pm 1} = \mp\frac{1}{2}[I_\pm\ I_z + I_z I_\pm\ ] \qquad T^Q_{2,\pm 2} = \frac{1}{2}I_\pm^2$$

and will be returned as matrices of dimension ***2I+1*** where ***I*** is the spin quantum number associated with the interaction.

Return Value:

A matrix.

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);     // Make a G interaction.
matrix T20 = G.Tcomp(0);                // This is the T20 spin tensor component
cout << T20);                           // Have a look on screen.
```

6.14 Auxiliary Functions

6.14.1 setPAS

Usage:

```
#include <IntG.h>
void IntG::setPAS()
```

Description:

The functions ***setPAS*** is used to orient the G interaction into it's principal axis system. All 5 spatial tensor components will be set to PAS values and the internal orientation angles set to zero.

Return Value:

None.

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
G.setPAS();                        // As if we used G(1.5,3.e5,0.2,0,0)
```

See Also: theta, phi, orient

6.14.2 symmetric

Usage:

```
#include <IntG.h>
int IntG::symmetric() const
```

Description:

The functions ***symmetric*** is used to check if the G interaction has any asymmetry. The function will return true if the interaction is symmetric and false if there is some asymmetry (non-zero eta value).

Return Value:

An integer

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
if(G.symmetric()) cout << "Yep";     // We should get No for G because eta=0.2)
else                  << "Nope";
```

See Also: eta

6.14.3 PAS

Usage:

```
int IntG::PAS) const
```

Description:

The function ***PAS*** is used to check if the G interaction is oriented in its PAS or not. The function will return true if the interaction is PAS aligned and false if not).

Return Value:

An integer

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
if(G.PAS()) cout << "Yep";          // We should get No for G because neither θ or ϕ is 0)
else                     << "Nope";
```

See Also: eta

6.14.4 wG2GCC

Usage:

```
#include <IntG.h>
friend double wG2GCC(double wG, double I)
```

Description:

The functions ***wG2GCC*** is used to convert a G frequency ***wG*** for a spin with quantum number ***I*** to a G coupling constant. The two are related in GAMMA by

$$QCC = e^2qQ = \frac{2I(2I-1)\omega^Q}{3} = 2I(2I-1)\sqrt{\frac{5}{6\pi}}\xi^Q$$

Return Value:

A double

Example:

```
double wG = 450.e3;                 // Guad. frequency of 450 kHz.
double NGCC = wG2GCC(wG, 1.5);// Guad. coupling if I=3/2
```

See Also: GCC2wG

6.14.5 GCC2wG

Usage:

```
#include <IntG.h>
friend double GCC2wG(double GCC, double I)
```

Description:

The functions ***GCC2wG*** is used to convert a G coupling constant to a G frequency. The two are related in GAMMA by

$$\omega^Q = \frac{3e^2qQ}{2I(2I-1)} = \frac{3QCC}{2I(2I-1)} = \sqrt{\frac{15}{2\pi}}\xi^Q$$

Return Value:

A double

Example:

```
double GCC = 450.e3;              // Guad. coupling constant of 450 kHz.
double wG = GCC2wG(wG, 1.5);      // Guad. frequency if I=3/2
```

See Also: wG2GCC

6.15 Hamiltonian Functions

6.15.1 H0

Usage:

```
#include <IntG.h>are
matrix IntG::H0() const
matrix IntG::H0(double theta, double phi) const
```

Description:

The function ***H0*** is used to obtain the G Hamiltonian as a first order perturbation to the Zeeman Hamiltonian. As such, the returned matrix is "secular" and commutes with both $\boldsymbol{F_z}$ and $\boldsymbol{R_z}$. It will be valid in a rotating frame about the z-axis. It will not be valid unless the Zeeman Hamiltonian (which it is meant to be added to[1]) is much stronger. The return array will have units of ***Hz***. The dimension of the array will be 2I+1 where I is the spin quantum value associated with the interaction. If the input arguments ***heta*** and ***phi*** are given the returned Hamiltonian is for the orientation at ***theta*** degrees down from the interaction PAS z-axis and ***phi*** degrees over from the interaction PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***degrees.***

In GAMMA the first order G Hamiltonian is given by

$$H_Q^{(0)} = \xi^Q A_{0,0}^Q(\eta, \theta, \phi) T_{0,0}^Q = \frac{\omega^Q}{12}[3\cos^2\theta - 1 + \eta\sin^2\theta\cos(2\varphi)][3I_z^2 - I(I+1)]$$

Return Value:

A matrix.

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
matrix H = G.H0();                 // Here's the 1st order Guad. Hamiltonian
cout << H;                         // Have a look at the Hamiltonian.
```

See Also: H1, Hsec, H

6.15.2 H1

Usage:

```
#include <IntG.h>
matrix IntG::H1() const
matrix IntG::H1(double theta, double phi) const
```

1. A spin in a strong magnetic field will evolve under the influence of both the Zeeman Hamiltonian, H_Z and the G Hamiltonian H_G. When the Zeeman interaction is much strong than the G interaction it suffices to use H0 instead of H_G. This is often nice to use because then the two Hamiltonians commute. In evolving a density operator one may then work in the rotating frame at a spin's Larmor frequency by simply removing the Zeeman Hamiltonian and evolving under only H0.

Description:

The function ***H1*** is used to obtain the second order G Hamiltonian as a perturbation to the Zeeman Hamiltonian. As such, the returned matrix is "secular" and commutes with both $\boldsymbol{F_z}$ and $\boldsymbol{R_z}$. It will be valid in a rotating frame about the z-axis. It will not be valid unless the Zeeman Hamiltonian (to which it is meant to be added[1]) is much stronger. The return array will have units of ***Hz***. The dimension of the array will be 2I+1 where I is the spin quantum value associated with the interaction. If the input arguments ***heta*** and ***phi*** are given the returned Hamiltonian is for the orientation at ***theta*** degrees down from the interaction PAS z-axis and ***phi*** degrees over from the interaction PAS x-axis. The values of ***theta*** and ***phi*** are assumed in ***degrees***

In GAMMA the second order G Hamiltonian is given by

$$H_Q^{(1)} = -\frac{\xi^2}{2\Omega_o} I_z \{ A_{0,1}^Q A_{0,-1}^Q [4I(I+1) - 8I_z^2 - 1] + A_{0,2}^Q A_{0,-2}^Q [2I(I+1) - 2I_z^2 - 1] \}$$

Return Value:

A matrix.

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
matrix H = G.H1();                 // Here's the 2nd order Guad. Hamiltonian
cout << H;                         // Have a look at the Hamiltonian.
```

See Also: GCC, NGCC, wG

6.15.3 Hsec

Usage:

```
#include <IntG.h>
matrix IntG::Hsec() const
```

Description:

The function ***Hsec*** is used to obtain the sum of the first and second order G Hamiltonians as a perturbation to the Zeeman Hamiltonian. As such, the returned matrix is "secular" and commutes with both $\boldsymbol{F_z}$ and $\boldsymbol{R_z}$. It will be valid in a rotating frame about the z-axis. It will not be valid unless the Zeeman Hamiltonian is much stronger. The return array will have units of ***Hz***. The dimension of the array will be 2I+1 where I is the spin quantum value associated with the interaction.

Return Value:

A matrix.

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
```

1. In the rotating frame the effective Hamiltonian my have all Zeeman contributions removed. Note that the function does not include the 1st order terms, so should be added to the return from the function H0! The function Hsec will do that automatically.

```
matrix H = G.H1();          // Here's the 2nd order Guad. Hamiltonian
cout << H;                  // Have a look at the Hamiltonian.
```

See Also: GCC, NGCC, wG

6.15.4 H

Usage:

```
#include <IntG.h>
matrix IntG::H() const
```

Description:

The function ***H*** is used to obtain the G Hamiltonian. Most likely this will NOT commute with $\boldsymbol{R_z}$. Thus it will be time independent in the laboratory frame (and time dependent in a frame rotating about the z-axis). The return array will have units of ***Hz***. The dimension of the array will be 2I+1 where I is the spin quantum value associated with the interaction.

Return Value:

A matrix.

Example:

```
IntG G(1.5, 3.e5, 0.2, 45.0, 45.0);// Make a G interaction.
matrix H = G.H1();          // Here's the 2nd order Guad. Hamiltonian
cout << H;                  // Have a look at the Hamiltonian.
```

See Also: GCC, NGCC, wG

6.16 I/O Functions

6.16.1 read

Usage:

void IntG::read(const String& filename, const spin_sys) const
void IntG::read(const String& filename, const spin_sys) const
void IntG::read(const String& filename, const spin_sys) const
void IntG::read(const String& filename, const spin_sys) const

Description:

The function ***delzz*** is used to either obtain or set the interaction G coupling constant. With no arguments the function returns the coupling in Hz. If an argument, ***dz***, is specified then the coupling constant for the interaction is set. It is assumed that the input value of ***dz*** is in units of ***Hz***. The function is overloaded with the name ***delz*** for convenience. Note that setting of ***delzz*** will alter the (equivalent) value of the G coupling ***GCC***/***NGCC*** as well as the G frequency.

Return Value:

Either void or a floating point number, double precision.

Example(s):

```
#include <IntG.h>
IntG G();                    // Empty G interaction.
G.delzz(100000.0);           // Set GCC to 100 KHz.
cout << G.delz ();           // Write coupling constant to std output.
```

See Also: GCC, NGCC, wG

6.16.2 ask

Usage:

#include <IntG.h>
double IntG:: () const
double IntG::delz () const
double IntG::delzz (double dz) const
double IntG::delz (double dz) const

Description:

The function ***delzz*** is used to either obtain or set the interaction G coupling constant. With no arguments the function returns the coupling in Hz. If an argument, ***dz***, is specified then the coupling constant for the interaction is set. It is assumed that the input value of ***dz*** is in units of ***Hz***. The function is overloaded with the name ***delz*** for convenience. Note that setting of ***delzz*** will alter the (equivalent) value of the G coupling ***GCC***/***NGCC*** as well as the G frequency.

Return Value:

Either void or a floating point number, double precision.

Example(s):

```
#include <IntG.h>
IntG G();                    // Empty G interaction.
G.delzz(100000.0);           // Set GCC to 100 KHz.
cout << G.delz ();           // Write coupling constant to std output.
```

See Also: GCC, NGCC, wG

6.16.3 askset

Usage:

```
#include <IntG.h>
double IntG:: () const
double IntG::delz () const
double IntG::delzz (double dz) const
double IntG::delz (double dz) const
```

Description:

The function ***delzz*** is used to either obtain or set the interaction G coupling constant. With no arguments the function returns the coupling in Hz. If an argument, ***dz***, is specified then the coupling constant for the interaction is set. It is assumed that the input value of ***dz*** is in units of ***Hz***. The function is overloaded with the name ***delz*** for convenience. Note that setting of ***delzz*** will alter the (equivalent) value of the G coupling ***GCC***/***NGCC*** as well as the G frequency.

Return Value:

Either void or a floating point number, double precision.

Example(s):

```
#include <IntG.h>
IntG G();                    // Empty G interaction.
G.delzz(100000.0);           // Set GCC to 100 KHz.
cout << G.delz ();           // Write coupling constant to std output.
```

See Also: GCC, NGCC, wG

6.16.4 print

Usage:

```
#include <IntG.h>
ostream& IntG::print (ostream& ostr, int fflag=-1)
```

Description:

The function ***print*** is used to write the interaction G coupling constant to an output stream ***ostr***. An additional flag ***fflag*** is set to allow some control over how much information is output. The default (***fflag !=0***) prints all information concerning the interaction. If ***fflag*** is set to zero only the basis parameters are printed.

Return Value:

The ostream is returned.

Example:

```
#include <IntG.h>
IntG G(2.5, 2.e6, 0.2, 45.7, 15.0);     // Make a G interaction.
cout << G;                              // Write the interaction to standard output.
```

See Also: <<

6.16.5 <<

Usage:

```
#include <IntG.h>
friend ostream& operator << (ostream& out, IntG& G)
```

Description:

The operator << defines standard output for the interaction G coupling constant.

Return Value:

The ostream is returned.

Example:

```
#include <IntG.h>
IntG G(1.5, 3.e5, 0.2);                 // Make a G interaction.
cout << G;                              // Write the interaction to standard output.
```

See Also: print

6.16.6 printSpherical

Usage:

```
#include <IntG.h>
ostream& IntG::print (ostream& ostr, int fflag=-1)
```

Description:

The function ***print*** is used to write the interaction G coupling constant to an output stream ***ostr***. An additional flag ***fflag*** is set to allow some control over how much information is output. The default (***fflag !=0***) prints all information concerning the interaction. If ***fflag*** is set to zero only the basis parameters are printed.

Return Value:

The ostream is returned.

Example:

```
#include <IntG.h>
IntG G(2.5, 2.e6, 0.2, 45.7, 15.0);     // Make a G interaction.
cout << G;                              // Write the interaction to standard output.
```

See Also: <<

6.16.7 printCartesian

Usage:

```
#include <IntG.h>
ostream& IntG::print (ostream& ostr, int fflag=-1)
```

Description:

The function ***print*** is used to write the interaction G coupling constant to an output stream ***ostr***. An additional flag ***fflag*** is set to allow some control over how much information is output. The default (***fflag !=0***) prints all information concerning the interaction. If ***fflag*** is set to zero only the basis parameters are printed.

Return Value:

The ostream is returned.

Example:

```
#include <IntG.h>
IntG G(2.5, 2.e6, 0.2, 45.7, 15.0);     // Make a G interaction.
cout << G;                              // Write the interaction to standard output.
```

See Also: <<

6.17 Description

6.17.1 Overview

A G interaction is the observed effect from the electron cloud surrounding a nucleus responding to an applied magnetic field. The spin itself experiences not only the applied field but also a field from the perturbed electron cloud, the latter field generally opposing the applied field or “shielding” the nucleus. Not only can the shielding contribution be quite large, it is usually orientationally dependent because the surrounding electron cloud is no spherical (due to chemical bonds). In the following discussion we will not be concerned with the isotropic and anti-symmetric parts of the shielding. The former produces measureable chemical shifts whereas the latter is rarely seen. Rather the focus will be on the symmetric rank 2 contribution, that which produces relaxation effects in liquid NMR and orientationally dependent shifts in solids.

6.17.2 Coordinate Systems

We will shortly concern ourselves with the mathematical representation of G interactions, in particular their description in terms of spatial and spin tensors. The spatial tensors will be cast in both Cartesian and spherical coordinates and we will switch between the two when convenient. The figure below relates the orientation angles theta and phi to the standard right handed coordinate system in all GAMMA treatments.

Cartesian and Spherical Coordinate Systems

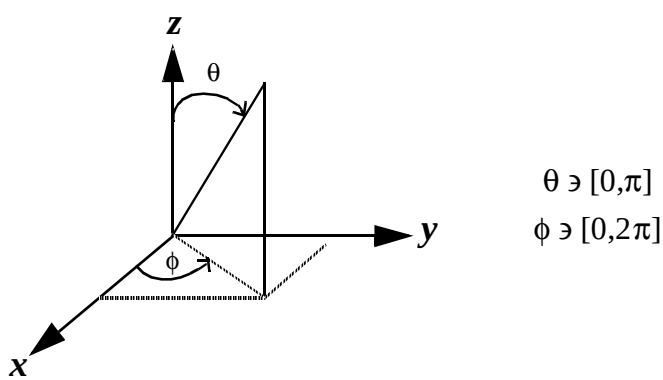


Figure 19-36 **The right handed Cartesian axes with the spherical angles and radius.**

6.17.3 Internal Structure

The internal structure of class **IntG** contains the quantities listed in the following table (names shown are also internal).

Table 3-1: Internal Structure of Class IsoG

Name	Description	Type	Name	Description	Type
AISO	Isotropic G Value	double	THETA	Orientation Angle	double
DELZZ	Spatial Tensor δ_{zz}	double	Asph	Spatial Tensor Values	complex*
ETA	Spatial Tensor η	double	Tsph	Spin Tensor Values	matrix*
PHI	Orientation Angle	double			

Note that since the spin angular momentum of an electron is ***I***=1/2, the spin tensor components will reside in a spin Hilbert space of dimension 2.

The three values ***AISO***, ***DELZZ,*** and ***ETA*** are all that is required to specify the G interaction strength and may be used to represent the G spatial tensor. However, in GAMMA the values of ***AISO*** and ***DELZZ*** are factored out of the spatial tensor such that all rank two interactions (such as the G interaction) have the same spatial tensor scaling.

The two angles ***THETA*** and ***PHI*** indicate how the G interaction is aligned relative to the interaction principal axes (PAS). These are one in the same as the angles shown in Figure 19-36 when the Cartesian axes are those of the PAS with the origin vaguely being the center of the nucleus. These are intrinsically tied into the values in the array ***Asph***.

There are five values in the complex vector ***Asph*** and these are irreducible spherical components of the G spatial tensor oriented at angle ***THETA*** down from the PAS z-axis and over angle ***PHI*** from the PAS x-axis. Note that these 5 values are not only orientation dependent, they are also ***ETA*** dependent. If either of the three the interaction values {***ETA***, ***THETA, PHI***} are altered these components will all be reconstructed. The values in ***Asph*** will be scaled such that they are consistent with other rank 2 spatial tensors in GAMMA which are independent of the interaction type.

Structure of a Variable of Class IntG

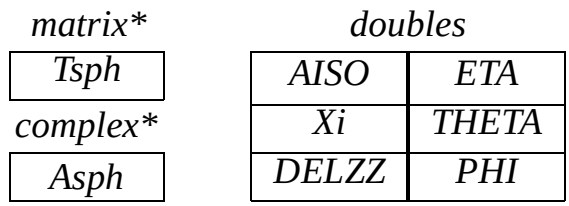


Figure 19-37 **Depiction of class IntG contents, i.e. what each GAMMA defined G interaction contains. The values of both Xi and DELZZ are maintained for convenience (one being deduced from the other if the field is specified). Tsph will contain 5 matrices which dimension will be 2*I+1 and Asph will contain 5 complex numbers.**

The vector of matrices relates to the sperical spin tensor components according to:

Tsph:	[0]	[1]	[2]	[3]	[4]
$T^G_{2,m}$:	$T^G_{2,0}$	$T^G_{2,1}$	$T^G_{2,-1}$	$T^G_{2,2}$	$T^G_{2,-2}$

and the vector of complex numbers relate to the GAMMA normalized spherical spatial tensor components via

Asph:	[0]	[1]	[2]	[3]	[4]
$A_{2,m}$:	$A_{2,0}$	$A_{2,1}$	$A_{2,-1}$	$A_{2,2}$	$A_{2,-2}$

6.17.4 Classical G Treatment

A chemical shift is the observed effect from the electron cloud surrounding a nucleus responding to an applied magnetic field. The spin itself experiences not only the applied field but also a field from the perturbed electron cloud, the latter field generally opposing the applied field or "shielding" the nucleus. We can write this latter "induced" field in terms of the applied field, $\vec{\boldsymbol{B}}_o$, as

$$\vec{\boldsymbol{B}}_{induced} = -\hat{\sigma} \bullet \vec{\boldsymbol{B}}_o$$

where $\hat{\sigma}$ is the chemical g tensor, a 3x3 array in Cartesian space, and the $\vec{\boldsymbol{B}}$'s vectors in Cartesian space. In matrix form this is simply[1]

$$\begin{bmatrix} B_{ind,\,x} \\ B_{ind,\,y} \\ B_{ind,\,z} \end{bmatrix} = -\begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{bmatrix}_i \bullet \begin{bmatrix} B_{0x} \\ B_{0y} \\ B_{0z} \end{bmatrix},$$

the induced field depends on the applied field strength, the applied field orientation, and the surrounding electron cloud. Note that $\vec{\boldsymbol{B}}_{induced}$ will not necessarily be co-linear with the applied field. Of course, every nuclear spin will have its own associated chemical g tensor. The classical interaction energy between this induced field and a nuclear spin is

$$E^G = -\vec{\mu}_e \bullet \vec{\boldsymbol{H}}_{effective} = -\vec{\mu}_e \bullet \frac{\hat{\boldsymbol{G}}}{g_e} \bullet \vec{\boldsymbol{H}}$$

where $\vec{\mu}_e$ is the electron magnetic moment, E the energy, and superscript G used to denote an electron G interaction.

6.17.5 Quantum Mechanical Formulation

The associated G interaction Hamiltonian is obtained from substitution of $-\beta\vec{\boldsymbol{S}} = \frac{h\gamma_e}{g_e}\vec{\boldsymbol{S}}$ for $\frac{\vec{\mu}}{g_e}$.

$$\boldsymbol{H}^G = \beta\vec{\boldsymbol{S}} \bullet \hat{\boldsymbol{G}} \bullet \vec{\boldsymbol{H}} = \frac{-h\gamma_e\vec{\boldsymbol{S}}}{g_e} \bullet \hat{\boldsymbol{G}} \bullet \vec{\boldsymbol{H}}\ ; \qquad (39\text{-}30)$$

1. Note that the effect of the G tensor is to alter the overall external field which the electron experiences. This is clearly seen from the product $\hat{\boldsymbol{G}} \bullet \vec{\boldsymbol{H}}$ which produces an effective field vector for the electron.

In matrix form this equation looks like

$$\boldsymbol{H}^G = \beta\begin{bmatrix}\boldsymbol{S}_x & \boldsymbol{S}_y & \boldsymbol{S}_z\end{bmatrix} \bullet \begin{bmatrix} g_{xx} & g_{xy} & g_{xz} \\ g_{yx} & g_{yy} & g_{yz} \\ g_{zx} & g_{zy} & g_{zz}\end{bmatrix} \bullet \begin{bmatrix}\boldsymbol{H}_x \\ \boldsymbol{H}_y \\ \boldsymbol{H}_z\end{bmatrix}. \tag{39-31}$$

Taking the magnitude of the applied field out, equation (39-30) is simply

$$\boldsymbol{H}^G = \beta H \sum_u^{axes} \sum_v^{axes} \langle 1|\vec{\boldsymbol{S}}|u\rangle\langle u|\hat{\boldsymbol{G}}|v\rangle\langle v|\vec{\boldsymbol{H}}_n|1\rangle \tag{39-32}$$

with $u, v \in \{x, y, z\}$ and $\vec{\boldsymbol{H}}_n$ a normalized magnetic field vector in the direction of the applied field.

6.17.6 Cartesian Tensor Formulation

Equation (39-31) can also be rearranged to produce an equation involving two rank 2 tensors by taking the dyadic product of the vectors $\vec{\boldsymbol{S}}$ and $\vec{\boldsymbol{H}}_n$.

$$\boldsymbol{H}^G = \beta H \sum_u^{axes} \sum_v^{axes} \langle u|\hat{\boldsymbol{G}}|v\rangle\langle v|\vec{\boldsymbol{H}}_n|1\rangle\langle 1|\vec{\boldsymbol{S}}|u\rangle = \beta H \sum_u^{axes} \sum_v^{axes} \langle u|\hat{\boldsymbol{G}}|v\rangle\langle v|\vec{\boldsymbol{H}}_n\vec{\boldsymbol{S}}|u\rangle$$

The dyadic product to produce $\vec{\boldsymbol{H}}_n\vec{\boldsymbol{S}}$ is explicitly done *via*

$$\begin{bmatrix} H_{nx} \\ H_{ny} \\ H_{nz}\end{bmatrix} \bullet \begin{bmatrix}\boldsymbol{S}_x & \boldsymbol{S}_y & \boldsymbol{S}_z\end{bmatrix} = \begin{bmatrix} H_{nx}\boldsymbol{S}_x & H_{nx}\boldsymbol{S}_y & H_{nx}\boldsymbol{S}_z \\ H_{ny}\boldsymbol{S}_x & H_{ny}\boldsymbol{S}_y & H_{ny}\boldsymbol{S}_z \\ H_{nz}\boldsymbol{S}_x & H_{nz}\boldsymbol{S}_y & H_{nz}\boldsymbol{S}_z\end{bmatrix}.$$

The G interaction Hamiltonian can thus be formulated as a scalar product of two rank 2 tensors. Letting $\hat{\boldsymbol{T}}^G = \vec{\boldsymbol{H}}_n\vec{\boldsymbol{S}}$, we have

$$\boldsymbol{H}^G = \beta H \hat{\boldsymbol{G}} \bullet \hat{\boldsymbol{T}}^G = \beta H \sum_u^{axes} \sum_v^{axes} \langle u|\hat{\boldsymbol{G}}|v\rangle\langle v|\hat{\boldsymbol{T}}^G|u\rangle$$

6.17.7 Spherical Tensor Formulation

The previous equation, , can also be rewritten in term of irreducible spherical components rather than in terms of the Cartesian components using the substitution

$$\sum_{l=0}^{2}\sum_{m}^{\pm l}(-1)^m g_{l-m}\hat{\boldsymbol{T}}^G_{lm} = \sum_{u}^{axes}\sum_{v}^{axes}\langle u|\hat{\boldsymbol{G}}|v\rangle\langle v|\hat{\boldsymbol{T}}^G|u\rangle \tag{39-33}$$

where g_{l-m} are spherical components of the tensor $\hat{\boldsymbol{G}}$. The result is

$$\boldsymbol{H}^G = \beta H\sum_{l=0}^{2}\sum_{m}^{\pm l}(-1)^m g_{l-m}\bullet\hat{\boldsymbol{T}}^G_{lm} \tag{39-34}$$

and we can expand the summation over the different ranks.

$$\boldsymbol{H}^G = \beta H\left| g_{0,0}\boldsymbol{T}^G_{0,0} + \sum_{m}^{\pm 1}(-1)^m g_{1,-m}\boldsymbol{T}^G_{1,m} + \sum_{m}^{\pm 2}(-1)^m g_{2,-m}\boldsymbol{T}^G_{2,m}\right|$$

In other words we now have

$$\boldsymbol{H}^G = \boldsymbol{H}^{GI} + \boldsymbol{H}^{GU} + \boldsymbol{H}^{GA}. \tag{39-35}$$

There is good reason to separate these terms. The rank 0 component of the G Hamiltonian is rotationally invariant and called the isotropic G Hamiltonian. In liquid EPR it will dictate where the electron resonance occurs. The rank 2 part is call the chemical G Anisotropy Hamiltonian. In liquid systems this Hamiltonian averages to zero and thus not affect observed g values. It will contribute to relaxation of the system. On the other hand, in solid systems this component does not average away and will partially determine peak shapes in powder averages. The rank 1 component is the antisymmetric part of the G Hamiltonian. Since the antisymmetric part of the G tensor is difficult to measure, this part of the G Hamiltonian is usually assumed small and neglected.

The isotropic component ($l = 0$) of the G Hamiltonian is thus written

$$\boldsymbol{H}^{GI}(AAS) = \beta H g_{0,0}\boldsymbol{T}^G_{0,0}\ , \tag{39-36}$$

the antisymmetric component ($l = 1$) of the G Hamiltonian is

$$\boldsymbol{H}^{GU}(AAS) = \beta H\sum_{m}^{\pm 1}(-1)^m A^G_{1,-m}(i,AAS)\bullet\boldsymbol{T}^G_{1,m}(i,AAS)\quad , \tag{39-37}$$

and the anisotropic component ($l = 2$) of the G Hamiltonian is

$$\boldsymbol{H}^{GA}_i(AAS) = \beta H\sum_{m}^{\pm 2}(-1)^m g_{2,m}(AAS)\bullet\boldsymbol{T}^G_{2,-m}(AAS) \tag{39-38}$$

6.17.8 G Interaction Spherical Tensor Spin Components

We can obtain the 9 irreducible spherical components of the G rank 2 "spin" tensor[1] directly from the Cartesian components, $\langle v|\hat{\boldsymbol{T}}|u\rangle$, as indicated in GAMMA Class Documentation on Spin Tensors. These are

$$\boldsymbol{T}^{G}_{l,\,m},$$

where G signifies the electron G interaction. The tensor index l spans the rank: $l \in [0, 2]$ while the tensor index m spans l: $m \in [-l, l]$ The nine formulas for these quantities a listed in the following figure where the field components are those of the normalized field vector $\vec{\boldsymbol{H}}_n$.[2]

G Rank 2 Irreducible Spherical Spin-Space Tensor Components

$$T^{G}_{0,\,0} = \frac{-1}{\sqrt{3}}\left[\boldsymbol{S}_z H_z + \frac{1}{2}(\boldsymbol{S}_+ H_- + \boldsymbol{S}_- H_+)\right] = \frac{-1}{\sqrt{3}}\vec{\boldsymbol{S}} \bullet \vec{\boldsymbol{H}}_n$$

$$T^{G}_{1,\,0} = \frac{-1}{2\sqrt{2}}[\boldsymbol{S}_+ H_- - \boldsymbol{S}_- H_+] \qquad T^{G}_{1,\,\pm 1} = \frac{-1}{2}[\boldsymbol{S}_\pm\, H_z - \boldsymbol{S}_z H_\pm\,]$$

$$T^{G}_{2,\,0} = \frac{1}{\sqrt{6}}[3\boldsymbol{S}_z H_z - (\vec{\boldsymbol{S}} \bullet \vec{\boldsymbol{H}}_n)]$$

$$T^{G}_{2,\,\pm 1} = \mp\frac{1}{2}[\boldsymbol{S}_{\pm\,+} H_z + \boldsymbol{S}_z H_\pm\,] \qquad T^{G}_{2,\,\pm 2} = \frac{1}{2}[\boldsymbol{S}_\pm\, H_\pm\,]$$

Figure 19-38 **The rank 2 spin-space tensor components for the electron G interaction.**

For $\vec{\boldsymbol{H}} = H\vec{\boldsymbol{H}}_n$, the matrix form of these tensor components are shown in the following figure in the single electron spin Hilbert space. The spin index has been omitted, the field components are those of the normalized vector $\vec{\boldsymbol{H}}_n$.

1. Due to the nature of the G interaction, the rank 2 tensor treatment produces a "spin" tensor $T^{G}_{l,\,m}$ which contains spatial components, namely the magnetic field vector. As a result, care must be used when performing spatial rotations on G tensors. Any spatial rotations must involve rotations of both G and $\boldsymbol{T}$
2. For these formulae, it is important to note that it is the second component in the composite spin/space tensor which is set to the normalized magnetic field vector $\vec{\boldsymbol{H}}_n$, although we might just as well have used the first vector instead. The difference is that the $l = 1$ equations would then appear of opposite sign from those given here. Our field vector has be set to point along the positive z-axis in the laboratory frame.

General G Spin-Space Tensor Components Matrix Representations

$$T^{(2)}_{0,0} = \frac{-1}{2\sqrt{3}}\begin{bmatrix} H_z & H_- \\ B_+ & -B_z \end{bmatrix} \qquad T^{(2)}_{1,0} = \frac{-1}{2\sqrt{2}}\begin{bmatrix} 0 & H_- \\ -H_+ & 0 \end{bmatrix} \qquad T^{(2)}_{1,-1} = \frac{-1}{2}\begin{bmatrix} -H_-/2 & 0 \\ H_z & H_-/2 \end{bmatrix} \qquad T^{(2)}_{1,1} = \frac{-1}{2}\begin{bmatrix} -H_+/2 & H_z \\ 0 & H_+/2 \end{bmatrix}$$

$$T^{(2)}_{2,0} = \frac{1}{2\sqrt{6}}\begin{bmatrix} 2H_z & -H_- \\ -H_+ & -2H_z \end{bmatrix} \quad T^{(2)}_{2,-1} = \frac{1}{2}\begin{bmatrix} H_-/2 & H_z \\ 0 & -H_-/2 \end{bmatrix} \quad T^{(2)}_{2,1} = \frac{-1}{2}\begin{bmatrix} H_+/2 & 0 \\ H_z & -H_+/2 \end{bmatrix} \quad T^{(2)}_{2,-2} = \frac{1}{2}\begin{bmatrix} 0 & 0 \\ H_- & 0 \end{bmatrix} \quad T^{(2)}_{2,2} = \frac{1}{2}\begin{bmatrix} 0 & H_+ \\ 0 & 0 \end{bmatrix}$$

Figure 19-39 **A general matrix representation of the rank 2 spin-space tensor components for the electron G interaction. The spin Hilbert space dimension if 2 due to the election having spin angular momentum of 1/2. The direction of the applied field is arbitrary, however the field vector is normalized in this formulation.**

The matrix representation of these nine tensor components will depend upon the matrix representations of the individual spin operators from which they are constructed[1]. These in turn depend upon the fact that electrons are spin 1/2 particles. Their G tensor components are, in the previous figure, expressed in matrix form in the default product basis of GAMMA. In this case the spin index is implicit.

The raising an lowering components of the field vector are defined in the standard fashion, namely $H_\pm = H_x \pm iH_y$. The simplest situation occurs when magnetic field points along the positive z-axis, $\vec{\boldsymbol{H}}_n = \hat{\boldsymbol{k}}$, *i.e.* these spin-space tensors are written in the laboratory frame. Then, the (normalized) field vector simplifies, $H_z = 1$ and $H_x = H_y = H_\pm = 0$. The applicable equations for the shielding space-spin tensors are then as follows.

G Spin-Space Tensor Components, H Along z-Axis

$$T^G_{0,0}(i) = \frac{-1}{\sqrt{3}}\boldsymbol{S}_{iz} \qquad T^G_{1,0}(i) = 0 \qquad T^G_{1,\pm1}(i) = \frac{-1}{2}\boldsymbol{S}_{i\pm}$$

$$T^G_{2,\pm1}(i) = \mp\frac{1}{2}\boldsymbol{S}_{i\pm} \qquad T^G_{2,0}(i) = \frac{2}{\sqrt{6}}\boldsymbol{S}_{iz} \qquad T^G_{2,\pm2}(i) = 0$$

Figure 19-40 **The rank 2 spin-space tensor components for the electron G interaction when the field vector is oriented along the +z axis in the laboratory frame.**

For $\vec{\boldsymbol{H}} = H\vec{\boldsymbol{H}}_n$ along the positive z-axis, the matrix form of these tensor components are shown in the following figure[2] (in the single spin Hilbert space).

G Spin-Space Tensor Components Matrix Representations, H on z-Axis

Figure 19-41 **A general matrix representation of the rank 2 spin-space tensor components for the**

1. Note that the spin tensors are invariably constructed in the laboratory coordinate system. Here the z-axis corresponds to the direction of the spectrometer static magnetic field and the coordinate system is right-handed.
2. The GAMMA program which produced these matrix representations can be found at the end of this Chapter, sosix Rank2SS_SpinT.cc.

$$T^G_{0,0} = \frac{-1}{2\sqrt{3}}\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix} \qquad T^G_{1,0} = \begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix} \qquad T^G_{1,-1} = \frac{-1}{2}\begin{bmatrix}0 & 0\\ 1 & 0\end{bmatrix} \qquad T^G_{1,1} = \frac{-1}{2}\begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}$$

$$T^G_{2,0} = \frac{1}{\sqrt{6}}\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix} \quad T^G_{2,1} = \frac{-1}{2}\begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix} \quad T^G_{2,-1} = \frac{1}{2}\begin{bmatrix}0 & 0\\ 1 & 0\end{bmatrix} \quad T^G_{2,-2} = \begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix} \quad T^G_{2,2} = \begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix}$$

electron G interaction when the field vector is oriented along the +z axis in the laboratory frame. The spin Hilbert space dimension if 2 due to the election having spin angular momentum of 1/2.

We must very careful in using these single spin rank 2 G tensors of this type because they contain both spatial and spin components. If we desire to express the G Hamiltonian relative to a particular set of axes we must insure that both the spatial tensor and the “spin” tensor are expressed in the proper coordinates. The spatial tensor alone cannot be rotated as it rotates only part of the spatial components[1]. It is improper to rotate this tensor in spin space because it also rotates spatial variables. Furthermore, note that **these rank 2 components are not the same as the rank 1 tensor components**.

6.17.9 General Rank 2 Spatial Tensor Components

The 9 irreducible spherical components of a rank 2 spatial tensor, $A^{(2)}_{lm}$, are related to its Cartesian components by the following formulas (See GAMMA Class Documentation on Spatial Tensors).

$$\begin{gathered}
A_{0,0} = \frac{-1}{\sqrt{3}}[A_{xx} + A_{yy} + A_{zz}] = \frac{-1}{\sqrt{3}}Tr\{A\} \\
A_{1,0} = \frac{-i}{\sqrt{2}}[A_{xy} - A_{yx}] \qquad A_{1,\pm1} = \frac{-1}{2}[A_{zx} - A_{xz} \pm i(A_{zy} - A_{yz})] \\
A_{2,0} = \sqrt{6}[3A_{zz} - (A_{xx} + A_{yy} + A_{zz})] = \sqrt{6}[3A_{zz} - Tr\{A\}] \\
A_{2,\pm1} = \mp\frac{1}{2}[A_{xz} + A_{zx} \pm i(A_{yz} + A_{zy})] \qquad A_{2,\pm2} = \frac{1}{2}[A_{xx} - A_{yy} \pm i(A_{xy} + A_{yx})]
\end{gathered} \tag{39-39}$$

Again the subscript l spans the rank as l = [0, 2], and the subscript m spans +/- l, m = [-l, l].

In this G interaction treatment, we then have the components $g_{l,m}$ as indicated in equation (39-34). Thus, the irreducible spherical tensor components can be obtained by substituting the Cartesian elements of the G tensor, $\hat{\boldsymbol{G}}$, into equations (39-39).

$$\begin{gathered}
g_{0,0} = \frac{-1}{\sqrt{3}}[g_{xx} + g_{yy} + g_{zz}] = \frac{-1}{\sqrt{3}}Tr\{\hat{\boldsymbol{G}}\} \\
g_{1,0} = \frac{-i}{\sqrt{2}}[g_{xy} - g_{yx}] \qquad g_{1,\pm1} = \frac{-1}{2}[g_{zx} - g_{xz} \pm i(g_{zy} - g_{yz})] \\
g_{2,0} = \sqrt{6}[3g_{zz} - (g_{xx} + g_{yy} + g_{zz})] = \sqrt{6}[3g_{zz} - Tr\{\hat{\boldsymbol{G}}\}] \\
g^G_{2,\pm1} = \mp\frac{1}{2}[g_{xz} + g_{zx} \pm i(g_{yz} + g_{zy})] \qquad g_{2,\pm2} = \frac{1}{2}[g_{xx} - g_{yy} \pm i(g_{xy} + g_{yx})]
\end{gathered} \tag{39-40}$$

1. See the discussion in Mehring

However, it is more convenient to rewrite the general rank two Cartesian tensor in terms of a sum over tensors of ranks 0 through 2 as follows,

$$\hat{A} = \begin{bmatrix} A_{xx} & A_{xy} & A_{xz} \\ A_{yx} & A_{yy} & A_{yz} \\ A_{zx} & A_{zy} & A_{zz} \end{bmatrix} = \underbrace{A_{iso}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Rank 0}} + \underbrace{\begin{bmatrix} 0 & \alpha_{xy} & \alpha_{xz} \\ -\alpha_{xy} & 0 & \alpha_{yz} \\ -\alpha_{xz} & -\alpha_{yz} & 0 \end{bmatrix}}_{\text{Rank 1}} + \underbrace{\begin{bmatrix} \delta_{xx} & \delta_{xy} & \delta_{xz} \\ \delta_{yx} & \delta_{yy} & \delta_{yz} \\ \delta_{zx} & \delta_{zy} & \delta_{zz} \end{bmatrix}}_{\text{Rank 2}} \tag{39-41}$$

where

$$A_{iso} = \frac{1}{3}Tr\{\hat{A}\} \qquad \alpha_{xy} = \frac{1}{2}(A_{xy} - A_{yx}) \qquad \delta_{xy} = \frac{1}{2}(A_{xy} + A_{yx} - 2A_{iso}) \tag{39-42}$$

The rank 0 part is isotropic (scalar), the rank 1 part is antisymmetric and traceless, and the rank 2 part traceless and symmetric. We shall apply this same nomeclature to our G spatial tensor to produce

$$\hat{G} = \begin{bmatrix} g_{xx} & g_{xy} & g_{xz} \\ g_{yx} & g_{yy} & g_{yz} \\ g_{zx} & g_{zy} & g_{zz} \end{bmatrix} = g_{iso}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & \alpha_{xy} & \alpha_{xz} \\ -\alpha_{xy} & 0 & \alpha_{yz} \\ -\alpha_{xz} & -\alpha_{yz} & 0 \end{bmatrix} + \begin{bmatrix} \delta_{xx} & 0 & 0 \\ 0 & \delta_{yy} & 0 \\ 0 & 0 & \delta_{zz} \end{bmatrix}. \tag{39-43}$$

where

$$g_{iso} = \frac{1}{3}Tr\{\hat{G}\} \qquad \alpha_{xy} = \frac{1}{2}(g_{xy} - g_{yx}) \qquad \delta_{xy} = \frac{1}{2}(g_{xy} + g_{yx} - 2g_{iso}) \tag{39-44}$$

6.17.10 Unscaled G Spherical Spatial Tensor PAS Components

As with any rank 2 spatial tensor, the G spatial tensor can be specified in its principal axis system, the set of axes in which the irreducible rank 2 component is diagonal[1]. The G tensor values are experimentally determined in the tensor principal axes. Employing (39-41) in the case where the irreducible rank 2 component is diagonal,

$$\hat{G}(PAS) = g_{iso}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & \alpha_{xy} & \alpha_{xz} \\ -\alpha_{xy} & 0 & \alpha_{yz} \\ -\alpha_{xz} & -\alpha_{yz} & 0 \end{bmatrix} + \begin{bmatrix} \delta_{xx} & 0 & 0 \\ 0 & \delta_{yy} & 0 \\ 0 & 0 & \delta_{zz} \end{bmatrix}$$

where (39-44) still applies.

Rank 2 spatial tensors are also commonly specified in their principal axis system by the three com-

1. The principal axis system is set such that $|\delta_{zz}| \geq |\delta_{yy}| \geq |\delta_{xx}|$. The orientation of the x and y axes are inconsequential if η is zero.

ponents; the isotropic value A_{iso}, the anisotropy ΔA, and the asymmetry η. These are generally given by

$$A_{iso} = \frac{1}{3}Tr\{A\}\ , \qquad \Delta A = A_{zz} - \frac{1}{2}(A_{xx} + A_{yy}) = \frac{3}{2}\delta_{zz} \qquad \eta = (\delta_{xx} - \delta_{yy})/\delta_{zz}$$

A set of Euler angles $\{\alpha, \beta, \gamma\}$ is normally also given to relate the spatial tensor principle axes to another coordinate system. For the g-tensor we have

$$\hat{G}(PAS) = g_{iso}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & \alpha_{xy} & \alpha_{xz} \\ -\alpha_{xy} & 0 & \alpha_{yz} \\ -\alpha_{xz} & -\alpha_{yz} & 0 \end{bmatrix} + \delta_{zz}\begin{bmatrix} -\frac{1}{2}(1-\eta) & 0 & 0 \\ 0 & -\frac{1}{2}(1+\eta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{39-45}$$

Note that δ_{zz} is NOT equivalent to g_{zz} and that η is NOT equivalent to $(g_{xx} - g_{yy})/g_{zz}$. The irreducible spherical elements of the G tensor, $g_{l,m}$, in the principal axis system are, by placement of (39-45) into (39-39),

$$g_{0,0}(PAS) = -\sqrt{3}g_{iso}$$

$$g_{1,0}(PAS) = -\frac{i}{\sqrt{2}}[g_{xy} - g_{yx}] \qquad g_{1,\pm 1}(PAS) = -\frac{1}{2}[(g_{zx} - g_{xz}) \pm i(g_{zy} - g_{yz})]$$

$$g_{2,0}(PAS) = \sqrt{3/2}\delta_{zz} \qquad g_{2,1}(PAS) = g_{2,-1}(PAS) = 0$$

$$g_{2,2}(PAS) = g_{2,-2}(PAS) = \frac{1}{2}\delta_{zz}\eta$$

and these values should be equivalent to those given in (39-40) on page 6-416.

6.17.11 Scaled G Spherical Spatial Tensor PAS Components

Throughout GAMMA, we desire all irreducible spherical rank 2 spatial components to be scaled so as they are independent of the particular interaction. To do so, we adjust them to be as similar to normalized spherical harmonics as possible. Thus, we here scale the G irreducible rank 2 spatial tensor so that the $2, 0$ component will have the same magnitude as the $m = 0$ rank two spherical harmonic when the two spherical angles are set to zero. Our "normalization" factor "X" is obtained by

$$A_{2,0}(\theta,\varphi)\Big|_{\theta=\varphi=0} = X^G \bullet g_{2,0}(\theta,\varphi)\Big|_{\theta=\varphi=0} = Y_{2,0}(\theta,\varphi)\Big|_{\theta=\varphi=0} = \sqrt{5/(4\pi)}$$

Using $g_{2,0}(PAS) = \sqrt{3/2}\delta_{zz}$ we thus define the GAMMA G anisotropy spatial tensor to be scaled such that its normalized spherical components are given by

$$A_{l,m} = \sqrt{5/(6\pi)}\delta_{zz}^{-1} g_{l,m} \tag{39-46}$$

and the irreducible rank 2 components are given in the next figure.

GAMMA Normalized Rank 2 Spatial Tensor PAS Components

$$A_{2,0}(PAS) = \sqrt{\frac{5}{4\pi}} \qquad A_{2,\pm1}(PAS) = 0 \qquad A_{2,\pm2}(PAS) = \sqrt{\frac{5}{24\pi}}\eta$$

Figure 19-42 **Generic irreducible rank 2 spatial tensor components as defined in GAMMA. These are shown in the principle axis system of the tensor and scaled to coincide with normalized spherical harmonics.**

The scaling factor $\sqrt{5/(6\pi)}\delta_{zz}^{-1}$ which was multiplied into the spherical G tensor components will subsequently be compensated for in the G interaction by use of a G interaction constant. The Anisotropic G Hamiltonian given in equation (23) becomes

$$\boldsymbol{H}^{GA} = \beta H \sum_{m}^{\pm1} (-1)^m g_{2-m} \bullet \hat{\boldsymbol{T}}_{2m}^G = \beta H \delta_{zz} \sqrt{\frac{6\pi}{5}} \sum_{m}^{\pm l} (-1)^m A_{2-m} \bullet \hat{\boldsymbol{T}}_{lm}^G \tag{39-47}$$

6.17.12 G Interaction Constant

In GAMMA, since we have defined our generic spatial and spin tensors to be scaled independent of the type of interaction, we use an interaction constant as a scaling factor when formulating Hamiltonians. The G anisotropic Hamiltonian may be produced from

$$\boldsymbol{H}^{GA} = \xi^G \sum_{m}^{\pm1} (-1)^m g_{2,-m} \boldsymbol{T}_{2,m}^G = \beta H \delta_{zz} \sqrt{\frac{6\pi}{5}} \sum_{m}^{\pm1} (-1)^m A_{2,-m} \boldsymbol{T}_{2,m}^G \tag{39-48}$$

so evidently

$$\xi^G = \beta H \delta_{zz} \sqrt{\frac{6\pi}{5}} \tag{39-49}$$

Such interaction constants are not very common in the literature (except with regards to some papers treating relaxation in liquid NMR) and thus not intuitive to many GAMMA users. So, one simply needs to be aware of the relationships between the interaction constant and any commonly used G tensor definitions. Most EPR literature retain the G tensor in Cartesian components, whereas in GAMMA we (internally) work with the spherical components consistently across the magnetic resonance interaction types. Perhaps the only quantity worthy of mention is the δ_{zz}, the G anisotropy . This is readily related to the typical G tensor Cartesian components.

$$\delta_{zz} = g_{zz} - g_{iso} = g_{zz} - \frac{1}{3} Tr\{\hat{G}\}$$

6.17.13 Spatial Tensor Rotations

We can express the spatial tensor components $A_{l,m}$ relative to any arbitrary axis system (AAS) by a rotation from the principal axes to the new axes *via* the formula

$$A_{l,m}(AAS) = \sum_{m'}^{\pm l} D^l_{mm'}(\Omega) A_{l,m'}(PAS) \tag{39-50}$$

where $D^l_{mm'}$ are the rank l Wigner rotation matrix elements and Ω the set of three Euler angles which relate the principal axes of the spatial tensor to the arbitrary axes[1].

6.17.14 G Hamiltonian Rotations

The G Hamiltonian can now be expressed with respect to any arbitrary axes through use of its spherical tensor components and the previous equation. Our Hamiltonian in spherical tensor form is

$$\boldsymbol{H}^G = \boldsymbol{H}^{GI} + \boldsymbol{H}^{GU} + \boldsymbol{H}^{GA} = \beta H g_{0,0} \boldsymbol{T}^G_{0,0} + \boldsymbol{H}^{GU} + \xi^G \sum_{m}^{\pm 2} (-1)^m A_{2,-m} \boldsymbol{T}^G_{2,m}$$

Negletcting the antisymmetric component and recalling that the isotropic component is rotationally

1. In this instance, i.e. the treatment of an electron G interaction, we must be careful to express the elements $\boldsymbol{T}^G_{l,-m}$ in the same axis system as $A_{l,m}$. When A is rotated in space, so must be $\boldsymbol{T}^G$. Essentially, the field vector changes relative to any new coordinate system when constructing $\boldsymbol{T}^G$. In other words, when $A_{l,m}$ is represented in its PAS (normally thought of as $\theta = \varphi = 0$) it does NOT necessarily see the externally applied field point along +z since the latter is defined in the laboratory frame whereas the former is set in an internal (electron cloud fixed) frame.

invarient we obtain, for an arbitrary axis system

$$\boldsymbol{H}^G(AAS) = \beta H g_{0,0}\boldsymbol{T}^G_{0,0} + \xi^G \sum_{m}^{\pm 2} (-1)^m A_{2,-m}(AAS)\boldsymbol{T}^G_{2,m}$$

$$= \beta H g_{iso}\vec{\boldsymbol{S}} \bullet \vec{\boldsymbol{H}}_n + \xi^G \sum (-1)^m A_{2,-m}(AAS)\boldsymbol{T}^G_{2,m}$$

which becomes, if the system is related to the laboratory frame in which the static external field is pointed along +z,

$$\boldsymbol{H}^G(AAS) = \beta H g_{iso}\boldsymbol{S}_z + \xi^G \sum_{m}^{\pm 2} (-1)^m A_{2,-m}(AAS)\boldsymbol{T}^G_{2,m}$$

6.17.15 G Hamiltonian Units

At this point it is evident that the Hamiltonian has units which are dictated by the factor

$$\beta H$$

This factor occurs in both isotropic and anisotropic terms. The G tensor is taken to be unitless and the units of angular momentum from the spin term are considered included in this factor. The value of the Bohr magneton β is

$$\beta = 9.2741 \times 10^{-21} \text{erg-}G^{-1}$$

and H is typicall specified in units of Gauss. Thus βH as shown will have energy units (ergs). We can readily convert to frequency units using h.

For a free electron where $g_e = 2.0023193 = g_{iso}$, the resonance frequency (the transition between $\boldsymbol{S}_z = \pm\frac{1}{2}$) in a 3000 G field will be given by

$$\omega_e = \frac{g_e \beta H}{h} = \frac{(2.0023193)(9.2741 \times 10^{-21} erg - G^{-1})(3000G)}{6.6262 \times 10^{-27} erg - s - cycle^{-1}} =$$

$$= (2.0023193)(1.3996 \times 10^6 Hz - G^{-1})(3000G) = 8.4074 GHz$$

Typical isotropic g factors are larger than that of a free electron so that a higher frequency will be required at any set field. However, most ESR spectrometers operate in CW mode where the frequency is set and the field is swept. As a result it is better to think that at a specified frequency most electrons resonance at a lower field than does a free electon.

6.17.16 The Anisotropic G Hamiltonian

The **G** tensor orientation will affect the observed electron resonance frequency. Unlike isotropic chemical shifts in NMR, the isotropic (rank 0) contribution to **G** is normally NOT included with the Zeeman Hamiltonian. Furthermore, the anti-symmetric (rank 1) contribution to **G** is rarely treated. The symmetric rank 2 contribution to the **G** interaction, that which we are primarily concerned with in class IntG, produces the following amisotropic Hamiltonian[1].

$$\boldsymbol{H}^{GA} = \xi^G \sum_m^{\pm 2} (-1)^m A_{2,-m} \bullet \boldsymbol{T}^G_{2,m} \qquad \xi^G = \beta H \delta_{zz} \sqrt{\frac{6\pi}{5}}$$

The reader should note normally the spin tensors, $T^G_{2,m}$, are specified in the laboratory frame where the applied magnetic field is along the +z axis. When that is true the $T^G_{2,\pm 2}$ terms are zero and the summation need only be taken over $m = 0, \pm 1$.

$$\boldsymbol{H}^{GA}(LAB) = \xi^G \sum_m^{\pm 1} (-1)^m A_{2,-m}(LAB) \bullet \boldsymbol{T}^G_{2,m}(LAB)$$

Furthermore, if we orient the spatial tensor principal axis system (PAS) to coincide with the laboratory axes, the anisotropic contribution to the G Hamiltonian is given by a relatively simple formula because both the $A^G_{2,\pm 1}$ terms are zero as well.

$$\begin{aligned} \boldsymbol{H}^{GA}(LAB, PAS) &= \xi^G \sum_m^{\pm 1} (-1)^m A_{2,-m}(LAB, PAS) \bullet \boldsymbol{T}^G_{2,m}(LAB) \\ &= \xi^G A^G_{2,0}(LAB, PAS) \boldsymbol{T}^G_{2,0}(LAB) \\ &= \beta H \delta_{zz} \sqrt{\frac{6\pi}{5}} \left(\sqrt{\frac{5}{4\pi}} \right) \left(\frac{2}{\sqrt{6}} \boldsymbol{S}_z \right) = \beta H \delta_{zz} \boldsymbol{S}_z \end{aligned} \tag{39-51}$$

However, when the G interaction principal axes are not oriented to coincide with the laboratory axes the anisotropic Hamiltonian equation becomes much more complicated than the one above.

$$\begin{aligned} \boldsymbol{H}^{GA}(\theta, \varphi) &= \xi^G \sum_m^{\pm 2} (-1)^m A^G_{2,-m}(\theta, \varphi) \bullet \boldsymbol{T}^G_{2,m} \\ &= \xi^G [A^G_{2,0}(\theta, \varphi) \boldsymbol{T}^G_{2,0} + A^G_{2,1}(\theta, \varphi) \boldsymbol{T}^G_{2,-1} + A^G_{2,-1}(\theta, \varphi) \boldsymbol{T}^G_{2,1}] \\ &= \xi^G [A^G_{2,0}(\theta, \varphi) \boldsymbol{T}_{2,0} + A^G_{2,1}(\theta, \varphi) \boldsymbol{T}^G_{2,-1} - A^{G\,*}_{2,1}(\theta, \varphi) \boldsymbol{T}^G_{2,1}] \\ &= \xi^G \{A^G_{2,0}(\theta, \varphi) \boldsymbol{T}_{2,0} + Re[A^G_{2,1}(\theta, \varphi)](\boldsymbol{T}^G_{2,-1} - \boldsymbol{T}^G_{2,1}) + iIm[A^G_{2,1}(\theta, \varphi)](\boldsymbol{T}^G_{2,-1} + \boldsymbol{T}^G_{2,1})\} \end{aligned}$$

1. Keep in mind that this Hamiltonian is for a single electron. In a multi-spin system one will have to sum such Hamiltonians for all electron spins.

Remember, the orientation angles, θ and φ, are spherical angles relative to the laboratory coordinate system. We have thus left off the "LAB" label on all terms. At this point we will substitute in the spin operators (assuming H is along +z)

$$T^G_{2,0} = \frac{2}{\sqrt{6}}\boldsymbol{S}_z \qquad T^G_{2,\pm 1} = \mp\frac{1}{2}\boldsymbol{S}_\pm$$

This produces

$$\begin{aligned}\boldsymbol{H}^{GA}(\theta,\varphi) &= \xi^{SA}\{A^{SA}_{2,0}(\theta,\varphi)\boldsymbol{T}^{SA}_{2,0} + Re[A^{SA}_{2,1}(\theta,\varphi)](\boldsymbol{T}^{SA}_{2,-1} - \boldsymbol{T}^{SA}_{2,1}) + iIm[A^{SA}_{2,1}(\theta,\varphi)](\boldsymbol{T}^{SA}_{2,-1} + \boldsymbol{T}^{SA}_{2,1})\}\\ &= \xi^{SA}\left\{A^{SA}_{2,0}(\theta,\varphi)\left[\frac{2}{\sqrt{6}}\boldsymbol{S}_z\right] + Re[A^{SA}_{2,1}(\theta,\varphi)]\frac{1}{2}[(\boldsymbol{S}_- + \boldsymbol{S}_+)] + iIm[A^{SA}_{2,1}(\theta,\varphi)]\frac{1}{2}[(\boldsymbol{S}_- - \boldsymbol{S}_+)]\right.\end{aligned}$$

We can use the identities $I_x = \frac{1}{2}(I_- + I_+)$ $\quad I_y = \frac{i}{2}(I_- - I_+)$ to obtain

$$\boldsymbol{H}^{GA}(\theta,\varphi) = \xi^G\left\{A^G_{2,0}(\theta,\varphi)\left[\frac{2}{\sqrt{6}}\boldsymbol{S}_z\right] + Re[A^G_{2,1}(\theta,\varphi)]\boldsymbol{S}_x + Im[A^G_{2,1}(\theta,\varphi)]\boldsymbol{S}_y\right\}$$

Upon substitution of the oriented spatial components we obtain

$$\begin{aligned}\boldsymbol{H}^{GA}(\theta,\varphi) = \xi^G\Bigg\{&\sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta\sin^2\theta\cos 2\varphi\right]\left[\frac{2}{\sqrt{6}}\boldsymbol{S}_z\right]\\ &+ \left[\sqrt{\frac{5}{24\pi}}\sin\theta[3\cos\theta - \eta(\cos\theta\cos 2\varphi)]\right]\boldsymbol{S}_x + \left[\sqrt{\frac{5}{24\pi}}\sin\theta\eta\sin 2\varphi\right]\boldsymbol{S}_y\Bigg\}\end{aligned}$$

and in turn

$$\begin{aligned}\boldsymbol{H}^{GA}(\theta,\varphi) = \xi^G\sqrt{\frac{5}{24\pi}}\{&[3\cos^2\theta - 1 + \eta\sin^2\theta\cos 2\varphi]\boldsymbol{S}_z\\ &+ \sin\theta[\cos\theta(3 - \eta\cos 2\varphi)\boldsymbol{S}_x + \eta\sin 2\varphi\boldsymbol{S}_y]\}\end{aligned} \tag{39-52}$$

which will condense down into the previous result, equation (39-51) on page 422, when the two angles are set to zero. Often it can be assumed that work is being done in a "high field limit" where the contributions to the anisotropy from the $\boldsymbol{S}_x$ and $\boldsymbol{S}_y$ terms is negligible. When such is the case the previous equation becomes (hfl => high field limit)

$$\begin{aligned}\boldsymbol{H}^{GA}_{hfl}(\theta,\varphi) &= \xi^G\sqrt{\frac{5}{24\pi}}\{[3\cos^2\theta - 1 + \eta\sin^2\theta\cos 2\varphi]\boldsymbol{S}_z\\ &= \frac{1}{2}\beta H\delta_{zz}\{[3\cos^2\theta - 1 + \eta\sin^2\theta\cos 2\varphi]\boldsymbol{S}_z\end{aligned} \tag{39-53}$$

6.17.17 The Full G Hamiltonian

By combining the isotropic and anisotropic parts of the G Hamiltonian we obtain the full Hamiltonian. We are still excluding the anti-symmetric (rank 1) component.

$$\begin{aligned} \boldsymbol{H}^G(\theta,\varphi) &= \boldsymbol{H}^{GI} + \boldsymbol{H}^{GA}(\theta,\varphi) \\ &= \beta H g_{iso}\boldsymbol{S}_z + \frac{1}{2}\beta H \delta_{zz}\{[3\cos^2\theta - 1 + \eta\sin^2\theta\cos 2\varphi]\boldsymbol{S}_z \\ &\quad + \sin\theta[\cos\theta(3-\eta\cos 2\varphi)\boldsymbol{S}_x + \eta\sin 2\varphi\boldsymbol{S}_y]\} \end{aligned} \tag{39-54}$$

We will define an isotropic resonance condition as $\Omega_{iso} = \frac{\beta H g_{iso}}{h}$ so that the Hamiltonian can be expressed relative to some base frequency (or field) as

$$\begin{aligned} \boldsymbol{H}^G(\theta,\varphi) &= \Omega_{iso}\boldsymbol{S}_z + \frac{1}{2}\Omega_{iso}\frac{\delta_{zz}}{g_{iso}}\{[3\cos^2\theta - 1 + \eta\sin^2\theta\cos 2\varphi]\boldsymbol{S}_z \\ &\quad + \sin\theta[\cos\theta(3-\eta\cos 2\varphi)\boldsymbol{S}_x + \eta\sin 2\varphi\boldsymbol{S}_y]\} \end{aligned} \tag{39-55}$$

In the high-field limit, we have simply

$$\boldsymbol{H}^G_{hfl}(\theta,\varphi) = \left[1 + \frac{1}{2}\frac{\delta_{zz}}{g_{iso}}\{3\cos^2\theta - 1 + \eta\sin^2\theta\cos 2\varphi\}\right]\Omega_{iso}\boldsymbol{S}_z \tag{39-56}$$

and this explicitly indicates the dominant way in whcih the G Hamiltonian is modulated by the interaciton orientation.

6.17.18 Electron Transition Frequencies

Having determined what the **G** Hamiltonian looks like at any orientation we are now in the position to determine the electron transition frequency. Since the electron is only a spin 1/2 particle, there is only one transition and that is between the $|\alpha\rangle$ and $|\beta\rangle$ states. We shall examime the energy levels of these states using $H|\psi\rangle = \varepsilon|\psi\rangle$, knowing that the transition frequency will be the difference between the two energies. Our working Hamitonian form is

$$\boldsymbol{H}^G(\theta, \varphi) = \boldsymbol{H}^{GI} + \boldsymbol{H}^{GA}(\theta, \varphi) = \beta H g_{iso} \boldsymbol{S}_z + \boldsymbol{H}^{GA}(\theta, \varphi)$$

and we can immediately calculate the isotropic contribution to the transition frequency.

$$\begin{aligned} \boldsymbol{H}^{GI}|\alpha\rangle &= \beta H g_{iso} \boldsymbol{S}_z|\alpha\rangle = \frac{1}{2}\beta H g_{iso}|\alpha\rangle \\ \boldsymbol{H}^{GI}|\beta\rangle &= \beta H g_{iso} \boldsymbol{S}_z|\beta\rangle = \frac{-1}{2}\beta H g_{iso}|\beta\rangle \end{aligned} \qquad \Omega^{GI} = (\varepsilon_\alpha - \varepsilon_\beta)/h = \frac{\beta H g_{iso}}{h} \tag{39-57}$$

The anisotropic contribution at high field is equally trivial. In fact, we can just read it off of equation (39-56) on page 424.

$$\Omega^{GA}_{hfl}(\theta, \varphi) = \frac{1}{2h}\beta H \delta_{zz}\{3\cos^2\theta - 1 + \eta \sin^2\theta \cos 2\varphi\} \tag{39-58}$$

The third term, due to the x & y spin operator components are a bit more tenacious. We have

$$\begin{aligned} \boldsymbol{H}^{GA}_{x,y}(\theta, \varphi)|\alpha\rangle &= \xi^G \sqrt{\frac{5}{24\pi}}\{\sin\theta[\cos\theta(3 - \eta\cos 2\varphi)\boldsymbol{S}_x|\alpha\rangle + \eta \sin 2\varphi \boldsymbol{S}_y|\alpha\rangle]\} \\ \boldsymbol{H}^{GA}_{x,y}(\theta, \varphi)|\beta\rangle &= \xi^G \sqrt{\frac{5}{24\pi}}\{\sin\theta[\cos\theta(3 - \eta\cos 2\varphi)\boldsymbol{S}_x|\beta\rangle + \eta \sin 2\varphi \boldsymbol{S}_y|\beta\rangle]\} \end{aligned}$$

We can use the ladder operators define d earlier to determine the

$$I_x|\alpha\rangle = \frac{1}{2}(I_- + I_+)|\alpha\rangle = \frac{1}{2}|\beta\rangle \qquad I_y|\alpha\rangle = \frac{i}{2}(I_- - I_+)|\alpha\rangle = \left(-\frac{i}{2}\right)|\beta\rangle$$

$$\Omega^G_{hfl}(\theta, \varphi) = \frac{\beta H}{h}\left[g_{iso} + \frac{\delta_{zz}}{2}\{3\cos^2\theta - 1 + \eta \sin^2\theta \cos 2\varphi\}\right] = \frac{\beta H}{h} g_{eff}$$

where

$$g_{eff} = g_{iso} + \frac{\delta_{zz}}{2}\{3\cos^2\theta - 1 + \eta \sin^2\theta \cos 2\varphi\}$$

The Rank 2 G Hamiltonian Summary

$$\boldsymbol{H}^G(AAS) = \sum_i \boldsymbol{H}_i^G(AAS) = \sum_i \xi^G \sum_{l=0}^{2} \sum_{m}^{\pm l} (-1)^m A_{l-m}(i, AAS) \bullet \boldsymbol{T}_{lm}^G(i, AAS)$$

$$\boldsymbol{H}_i^G(AAS) = \xi^G \sum_{l=0}^{2} \sum_{m}^{\pm l} (-1)^m A_{l-m}(i, AAS) \boldsymbol{T}_{lm}^G(i, AAS)$$

$$\xi^G = \beta H \delta_{zz} \sqrt{\frac{6\pi}{5}}$$

$$A_{l,m}^G(i, AAS) = \sum_{m'}^{\pm l} D_{mm'}^l(\varphi, \theta, \chi) A_{l,m'}^G(i, PAS)$$

$$A_{0,0}^G(i, PAS) = -\sqrt{3} g_{iso}(i)$$

$$A_{1,0}^G(i, PAS) = -\frac{i}{\sqrt{2}}[g_{xy}(i, PAS) - g_{yx}(i, PAS)]$$

$$A_{1,\pm1}^G(i, PAS) = -\frac{1}{2}[(g_{zx}(i, PAS) - g_{xz}(i, PAS)) \pm i(g_{zy}(i, PAS) - g_{yz}(i, PAS))]$$

$$A_{2,0}^G(i, PAS) = \sqrt{3/2}\, g_{zz}(i) \qquad A_{2,\pm1}^G(i, PAS) = 0 \qquad A_{2,\pm2}^G(i, PAS) = \frac{1}{2}\delta_{zz}(i)\eta(i)$$

$$T_{0,0}^G(i, AAS) = \frac{-1}{\sqrt{3}}\left[\boldsymbol{I}_{iz}B_z + \frac{1}{2}(\boldsymbol{I}_{i+}B_- + \boldsymbol{I}_{i-}B_+)\right] = \frac{-1}{\sqrt{3}}\vec{\boldsymbol{I}}_i \bullet \vec{\boldsymbol{B}}_n$$

$$T_{1,0}^G(i, AAS) = \frac{-1}{2\sqrt{2}}[\boldsymbol{I}_{i+}B_- - \boldsymbol{I}_{i-}B_+] \qquad T_{1,\pm1}^G(i, AAS) = \frac{-1}{2}[\boldsymbol{I}_{i\pm}\, B_z - \boldsymbol{I}_{iz}B_\pm\,]$$

$$T_{2,0}^G(i, AAS) = \frac{1}{\sqrt{6}}[3\boldsymbol{I}_{iz}B_z - (\vec{\boldsymbol{I}}_i \bullet \vec{\boldsymbol{B}}_n)]$$

$$T_{2,\pm1}^G(i, AAS) = \mp\frac{1}{2}[\boldsymbol{I}_{i\pm}\, B_z + \boldsymbol{I}_{iz}B_\pm\,] \qquad T_{2,\pm2}^G(i, AAS) = \frac{1}{2}[\boldsymbol{I}_{i\pm}\, B_\pm\,]$$

Although these equations are generally applicable, it is convenient to express the G Hamiltonian with clear separation between the different ranks (the components with differing values of l). The isotropic component $\boldsymbol{H}^{GI}$ in the treatment of liquid samples will normally be placed into an overall isotropic Hamiltonian, H_0 because it does not disappear upon rotational averaging. The asymmetric component, $\boldsymbol{H}^{GU}$, is usually zero, the G tensor taken as essentially symmetric. ***The***

The Electron G Anisotropy Hamiltonian

Arbitrary Axis System

$$\boldsymbol{H}^{GA}(AAS) = \xi^G \sum_{m}^{\pm 2} (-1)^m A_{2-m}(AAS) \boldsymbol{T}^G_{2m}(AAS)$$

$$A_{2,0}(PAS) = \sqrt{\frac{5}{4\pi}} \qquad T^G_{2,0} = \frac{1}{\sqrt{6}}[3\boldsymbol{S}_z H_z - (\vec{\boldsymbol{S}} \bullet \vec{\boldsymbol{H}}_n)]$$

$$\xi^G = \beta H \delta_{zz} \sqrt{\frac{6\pi}{5}} \qquad A_{2,\pm 1}(PAS) = 0 \qquad T^G_{2,\pm 1} = \mp\frac{1}{2}[\boldsymbol{S}_{\pm}\ {}_{+}H_z + \boldsymbol{S}_z H_{\pm}\ ]$$

$$A_{2,\pm 2}(PAS) = \sqrt{\frac{5}{24\pi}}\eta \qquad T^G_{2,\pm 2} = \frac{1}{2}[\boldsymbol{S}_{\pm}\ H_{\pm}\ ]$$

$$A_{2,m}(AAS) = \sum_{m'}^{\pm 2} D^2_{mm'}(\varphi, \theta, \chi) A_{2,m'}(PAS)$$

Laboratory Frame

$$\boldsymbol{H}^{GA}(LAB) = \xi^G \sum_{m}^{\pm 1} (-1)^m A_{2-m}(LAB) \boldsymbol{T}^G_{2m}(LAB)$$

$$A_{2,0}(PAS) = \sqrt{\frac{5}{4\pi}} \qquad T^G_{2,0}(LAB) = \frac{2}{\sqrt{6}}\boldsymbol{I}_{iz}$$

$$\xi^G = \beta H \delta_{zz} \sqrt{\frac{6\pi}{5}} \qquad A_{2,\pm 1}(PAS) = 0 \qquad T^G_{2,\pm 1}(LAB) = \mp\frac{1}{2}\boldsymbol{I}_{i\pm}$$

$$A_{2,\pm 2}(PAS) = \sqrt{\frac{5}{24\pi}}\eta \qquad T^G_{2,\pm 2}(LAB) = 0$$

$$A_{2,m}(LAB) = \sum_{m'}^{\pm 2} D^2_{mm'}(\varphi_{PAS \to LAB}, \theta_{PAS \to LAB}, \chi_{PAS \to LAB}) A_{2,m'}(PAS)$$

$$A^G_{2,m}(i, LAB)\Big|_{\eta = 0} = Y_{2,m}(\theta, \varphi)$$

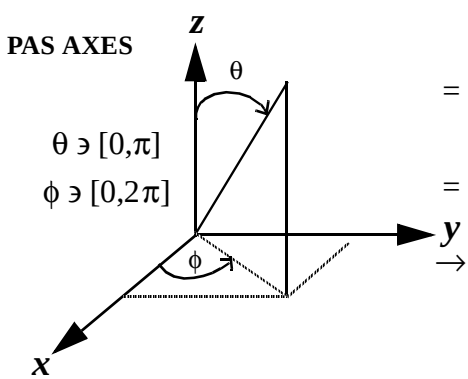
$$T^G_{2,0}(LAB) = \frac{-2}{\sqrt{6}}\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad T^G_{2,1}(LAB) = \frac{-1}{2}\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \qquad T^G_{2,-1}(LAB) = \frac{1}{2}\begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \qquad T^G_{2,\pm 2}(LAB) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

6.17.19 G PAS Equations

When the G interaction has alignment along its principal axes system virtually all of the G spatial tensor equations simplify. However, because the magnetic field components will then be oriented, the space-spin tensor components become complicated. Only when the PAS is aligned with the laboratory z-axis do both space and space-spin simplify. The following figure collects these equations for convenience.

G Equations Involving the PAS

$$\boldsymbol{H}^G(PAS) = \beta H g_{iso} S_z + \xi^G \sum_m (-1)^m A_{2-m}(PAS)\boldsymbol{T}^G_{2m}(PAS)$$



$$= \beta H g_{iso} S_z + \xi^G \sqrt{\frac{5}{24\pi}}[\sqrt{6}T^G_{2,0} + \eta(T^G_{2,2} + T^G_{2,-2})]$$

$$= \beta H\left[g_{iso}S_z + \frac{\delta_{zz}}{2}\left(3\boldsymbol{S}_z H_z - (\vec{\boldsymbol{S}} \bullet \vec{\boldsymbol{H}}_n) + \frac{\eta}{2}(\boldsymbol{S}_- H_- + \boldsymbol{S}_+ H_+)\right)\right]$$

$$\rightarrow \beta H(g_{iso} + \delta_{zz})\boldsymbol{S}_z$$

$$\xi^G = \beta H \delta_{zz}\sqrt{\frac{6\pi}{5}}$$

$$\eta = (A_{xx} - A_{yy})/A_{zz}$$

$$|A_{zz}| \geq |A_{yy}| \geq |A_{xx}|$$

$$A_{2,0}(PAS) = \sqrt{6}[3A_{zz} - Tr\{A\}]\big|_{PAS} = \sqrt{\frac{5}{4\pi}}$$

$$A_{2,\pm1}(PAS) = \mp\frac{1}{2}[A_{xz} + A_{zx} \pm i(A_{yz} + A_{zy})]\Big|_{PAS} = 0$$

$$A_{2,\pm2}(PAS) = \frac{1}{2}[A_{xx} - A_{yy} \pm i(A_{xy} + A_{yx})]\big|_{PAS} = \sqrt{\frac{5}{24\pi}}\eta$$

$$A_{xx}(PAS) = \sqrt{\frac{5}{24\pi}}[\eta - 1] \qquad A_{yy}(PAS) = -\sqrt{\frac{5}{24\pi}}[1 + \eta] \qquad A_{zz}(\theta, \varphi) = \sqrt{\frac{5}{6\pi}}$$

$$A_{xz}(PAS) = 0 = A_{zx}(PAS) = A_{xy}(PAS) = A_{yx}(PAS) = A_{yz}(PAS) = A_{zy}(PAS)$$

Figure 19-43 **Equations relevant to the G interaction in its principal axis orientation (PAS). GAMMA uses a spatial tensor which is scaled[1] so that rotations by angles θ & ϕ produce spherical harmonics for a symmetric interaction (η = 0).**

Included are the general relationships between the (GAMMA scaled) Cartesian tensor components to the irreducible spherical components. They are valid when η is defined accordingly! If η is defined by the other common convention ($|A_{zz}| \geq |A_{xx}| \geq |A_{yy}|$) then the sign on the $A^G_{2,\pm2}$ will change as will the sign on the Hamiltonian terms multiplied by η.

1. The scaling on both $\{A_{2m}\}$ and $T_{2m}\}$ are arbitrary, GAMMA uses an (uncommon) scaling which independent of the interaction type. What is NOT arbitrary is the scaling within either of the two sets of components. In addition, the combined scaling of the two sets is critical to the proper formation of G Hamiltonians. For that, GAMMA uses an interaction constant.

6.17.20 G Equations At Any Orientation

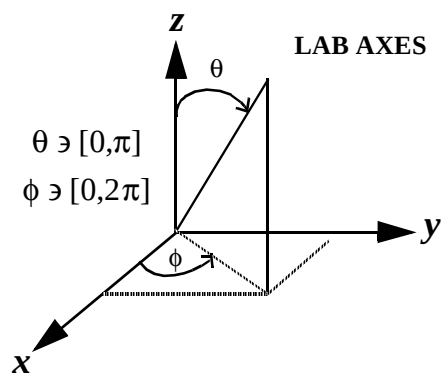
When the G interaction has a arbitrary alignment (relative to the laboratory frame, where the static field sets the z-axis) the G equations become sligltly more complicated. The figure below depicts them for convenience.

G Equations Oriented At Angles {θ,ϕ} From Lab Frame[1]

$$\boldsymbol{H}^G(\theta,\varphi) = \beta H g_{iso} S_z + \xi^G \sum_m^{\pm 2} (-1)^m A_{2,-m}(\theta,\varphi) \bullet \boldsymbol{T}^G_{2,m}$$

$$T^G_{2,0}(LAB) = \frac{2}{\sqrt{6}} \boldsymbol{I}_{iz}$$

$$T^G_{2,\pm 1}(LAB) = \mp \frac{1}{2} \boldsymbol{I}_{i\pm} \qquad T^G_{2,\pm 2}(LAB) = 0$$



$$A_{2,0}(\theta,\varphi) = \sqrt{\frac{5}{4\pi}}\left[\frac{1}{2}(3\cos^2\theta - 1) + \frac{1}{2}\eta \sin^2\theta \cos 2\varphi\right]$$

$$A_{2,1}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}} \sin\theta[3\cos\theta - \eta(\cos\theta\cos 2\varphi - i\sin 2\varphi)] = -A_{2,-1}{}^*(\theta,\varphi)$$

$$A_{2,2}(\theta,\varphi) = \sqrt{\frac{5}{24\pi}}\frac{1}{2}[3\sin^2\theta + \eta[\cos 2\varphi(1+\cos^2\theta) - i2\sin 2\varphi\cos\theta]] = A_{2,-2}{}^*(\theta,\varphi)$$

$$\eta = (A_{xx} - A_{yy})/A_{zz} \qquad |A_{zz}| \geq |A_{yy}| \geq |A_{xx}| \qquad \xi^G = \beta H \delta_{zz}\sqrt{\frac{6\pi}{5}}$$

$$\boldsymbol{H}^G(\theta,\varphi) = \beta H g_{iso} S_z + \frac{1}{2}\beta H \delta_{zz}\{[3\cos^2\theta - 1 + \eta\sin^2\theta\cos 2\varphi]\boldsymbol{S}_z + \sin\theta[\cos\theta(3 - \eta\cos 2\varphi)\boldsymbol{S}_x + \eta\sin 2\varphi \boldsymbol{S}_y]\}$$

Figure 19-44 **Equations relevant to the G Hamiltonian when oriented at angles θ & ϕ from the laboratory axis orientation (LAB). GAMMA uses a spatial tensor which is scaled[2] so that rotations by angles θ & ϕ produce spherical harmonics for a symmetric interaction (η = 0).**

1. The G interaction constant, as well as the relative scalings on the sets of spatial and spin tensors, can be adjusted as desired. However all components of the space or spin tensor must be adjusted by the same scaling. The GAMMA scaling is oriented to liquids where so that all spatial components are related to the spherical harmonics in the spatial tensor PAS.
2. The scaling on both $\{A_{2m}\}$ and $T_{2m}\}$ are arbitrary, GAMMA uses a scaling which independent of the interaction type. What is NOT arbitrary is the scaling within either of the two sets of components. In addition, the combined scaling of the two sets is also crucial. For that, GAMMA uses an interaction constant.

6.18 G Interaction Parameters

This section describes how an ASCII file may be constructed that is self readable by a G interaction. The file can be created with any editor and is read with the G interaction member function “read”. An example of one such file is given in its entirety at the end of this section. Keep in mind that parameter ordering in the file is arbitrary. Other parameters are allowed in the file which do not relate to G interactions.

Table 4: G Interaction Parameters

Parameter	Units	Examples Parameter (Type) : Value - Statement
GCC	KHz	GCC (1) : 370.3 - G Coupling (KHz)
Geta	none	Geta (1) : 0.33 - G Asymmetry
Gtheta	degrees	Gtheta (1) : 127.2 - G Orientation from PAS z (deg)
Gphi	degrees	Gphi (1) : 270.9 - G Orientation from PAS x(deg)

G Frequency: WG, WGkHz, WGKHz, WGHz, WGMHz

The G frequency can be specified. This can be accomplished with parameters using any of the names above or these names with a (#) added as a suffix. The default units for WG are KHz other names can be used to set the value in particular units. Note that this parameter is related to the G coupling constant which is specified with “(N)GCC” parameters. If both GCC and WG are set in the same file, the G frequency will be used to set up the G interaction.

Table 5: G Frequency[a]

Parameter	Assumed Units	Examples Parameter (Type=1) : Value - Statement
WG	KHz	WG (1) : 320.13 - Guad. Frequency in kHz
WGMHz	MHz	WGMHz (1) : 1.27 - Guad. Frequency in MHz
WGHz	Hz	WGHz(2) (1) : 1320.7 - Guad. Frequency in Hz

a. Shown are three possible parameters used to set the G frequency. The others mentioned above can also be used to specify it. Specification of a G coupling constant will also set the interaction’s G frequency. Parameter type 1 indicates a double precision number parameter

G Coupling Constant: GCC, GCCkHz, GCCKHz, GCCHz, GCCMHz

The G coupling constant can be specified. This can be accomplished with parameters using any of the names above, these same names with an “N” as a prefix, and/or these names with a (#) added as a suffix. The default units for GCC are KHz other names can be used to set the value in particular units. Note that this parameter

is related to the G frequency which is specified with “WG” parameters. If both GCC and WG are set in the same file, the G frequency will be used to set up the G interaction.

Table 6: G Coupling Constant[a]

Parameter	Assumed Units	Examples Parameter (Type=1) : Value - Statement
GCC	KHz	GCC (1) : 320.13 - Guad. Coupling in kHz
NGCCMHz	MHz	NGCCMHz (1) : 1.27 - Guad. Coupling in MHz
GCCHz	Hz	GCCHz(2) (1) : 1320.7 - Guad. Coupling in Hz

a. Shown are three possible parameters used to set the G coupling. The others mentioned above can also be used to specify it. Specification of a G frequency will also set the G coupling in the interaction. Parameter type 1 indicates a double precision number parameter

G Asymmetry

The asymmetry parameter must be within the range of [0, 1]. This parameter does not need to be set for a G interaction definition, it will be assumed 0 if unspecified.

Table 7: G Asymmetry[a]

Parameter	Assumed Units	Examples Parameter (Type=1) : Value - Statement
Geta	none	Geta (1) : 0.4 - G Asymmetry

a. Parameter type 1 indicates an integer parameter.

G Theta Orientation

The angle theta which relates the G interactions orientation down from the z-axis of its PAS may be set. This is not essential and will be taken as zero in left unspecified.

Table 8: Theta Orientation[a]

Parameter	Assumed Units	Examples Parameter (Type=1) : Value - Statement
Gtheta	degrees	Gtheta (1) : 45.7 - G Orientation from PAS z

a. Parameter type 1 indicates an integer parameter.

G Phi Orientation

The angle phi which relates the G interactions orientation over from the x-axis of its PAS may be set. This is

not essential and will be taken as zero in left unspecified.

Table 9: Theta Orientation[a]

Parameter	Assumed Units	Examples Parameter (Type=1) : Value - Statement
Gphi	degrees	Gphi (1) : 134.6 - G Orientation from PAS x

a. Parameter type 1 indicates an integer parameter.

6.19 Literature Comparisons

6.19.1

The fol

Comparison of GAMMA & Equations

6.20 G Interaction Examples

6.20.1 Zero Field Transitions, First Order Spectra

As a first example we'll look into some of the G Hamiltonians provided by class IntG in the interaction PAS (principal axes). Our results for both the transitions at zero field and NMR spectra to first order should agree with A. J. Vega's article[1] figures 1 & 2.

First Order G Spectra

Figure 19-45 **Spectra produced by program IntQu_LC6.cc, page -151. The G frequency was set to 300 kHz. The interaction was in its PAS and the asymmetry set to zero. Zero field transtions & relative intensities are shown in the tables.**

1. "G Nuclei in Solids", Alexander J. Vega, Encyclopedia of Nuclear Magnetic Resonance, Editors-in-Chief D.M.Grant and R.K. Harris, Vol. 6, Ped-Rel, pgs 3869-3889.

6.21 References

[9] J.E. Wertz and J.R. Bolton, *Electron Spin Resonance. Elementary Theory and Practical Applications*, McGraw-Hill Book Co., New York, New York, (1986), Chapman and Hall.

[10] Brink, D.M. and Satchler, G.R. (1962), Angular Momentum, Clarendon Press, Oxford.

0.5 Programs and Input Files

IntGu_LC0.cc

```
/* IntGu_LC0.cc
*************************************************************-*-c++-*-
**
**
**                    Test Program for the GAMMA Library
**
**              G Interaction Literature Comparison 0
**
**
**
**  This program checks the G interaction class IntG in
**
**  GAMMA.  In particular it looks to see how well the class parallels
**
**  the articles by Pascal P. Man
**
**
**
**  "G Interactions", Encyclopedia of Magnetic Resonance,
**
**  by Grant and Harris, Vol 6, Ped-Rel, page 3838-3869.
**
**
**
**  and Alexander Vega
**
**
**
**  "G Nuclei in Solids", Encyclopedia of Magnetic Resonance,
**
**  by Grant and Harris, Vol 6, Ped-Rel, page 3869-3889.
**
**
**
**  In particular, their PAS G Hamiltonians are generated and
**
**  compared with the G interaction class Hamiltonians.
**
**
**
**  Man's Hamiltonians will be generated from equations in (5) on page
**
**  3839 of the his article.  Vega's Hamiltonians will be made from
**
**  equations (28), (32) and (33) of his article.  Note that his (32)
**
**  is missing a factor of 1/3 on the <1|H|3> and <3|H|1> components.
**
**
**
** Author:     S.A. Smith
**
** Date:       10/11/96
**
** Update:     10/11/96
**
** Version:    3.6
**
** Copyright:  S. Smith.  You can modify this program for personal use, but
**
**                    you must leave it intact if you re-distribute it.
**
**
**
*********************************************************************
*******************/

#include <gamma.h>                          // Include GAMMA

main (int argc, char* argv[])
{
//                    Set Up The G Interaction

  int qn=1;
  double I;
  query_parameter(argc, argv, qn++,                  // Read in the coupling
                  "\n\tSpin Guantum Number? ", I);
  double W;                                          // G frequency
  query_parameter(argc, argv, qn++,                  // Read in the coupling
    "\n\tG Frequency(kHz)? ", W);
  W *= 1.e3;                                         // Put this in Hz
  double Geta;
  query_parameter(argc, argv, qn++,                  // Read in the coupling
    "\n\tG Asymmetry [0, 1]? ", Geta);
//                    Construct GAMMA G Interaction

  IntG G(I,wG2GCC(W,I),Geta,0.0,0.0);
//                Here are the Operators To Build Man's Hamiltonians

  int Ival = int(2.*I + 1);          // For 1 spin SOp functions
  matrix IE = Ie(Ival);              // The operator 1
  matrix IM = Im(Ival);              // The operator I-
  matrix IP = Ip(Ival);              // The operator I+
  matrix IZ = Iz(Ival);              // The operator Iz
  matrix IX = Ix(Ival);              // The operator Ix
  matrix IY = Iy(Ival);              // The operator Iy
//                Here's The H Accoring To Man's Equation (5a)

//                (Note That His W is Half Of Our Definition)

  matrix HMa = 3.0*IZ*IZ - (I*(I+1))*IE + Geta*((IX*IX)-(IY*IY));
  HMa *= (W/6.0);
```

```
//                    Here's The H Accoring To Man'c Equation (5c)
//                    (Note That His W is Half Of Our Definition)
 matrix HMb = 3.0*IZ*IZ - (I*(I+1))*IE + (Geta/2.)*((IP*IP)+(IM*IM));
 HMb *= (W/6.0);
//                    Here's The H According To GAMMA
 matrix HG = G.H();
//                    Here's The H Also According To GAMMA
 matrix HGB = G.H(0.0, 0.0);
//                    Here Are Vegas V's According To Equations (22-27, 31)
//                    (Switches eta Sign To Account For Opposite PAS Definition)
 double Eta = -G.eta();
 double Vxx = 0.5*(- 1. - Eta);
 double Vyy = 0.5*(-1. + Eta);
 double Vzz = 1.0;
 double Vxy = 0.0;
 double Vxz = 0.0;
 double Vyz = 0.0;
 complex V1(-Vxz, -Vyz);
 complex Vm1(Vxz, -Vyz);
 complex V2(0.5*(Vxx-Vyy), Vxy);
 complex Vm2(0.5*(Vxx-Vyy), -Vxy);
//          Generate H According To Vega's Equations (32) Or (33)
 matrix HVega;
 if(I == 1)
  {
  HVega = matrix(3,3);
  HVega.put(Vzz/6.0, 0, 0);
  HVega.put(Vm1/sqrt(2.0), 0, 1);
  HVega.put(Vm2/3., 0, 2);                    // Added 1/3 Factor!
  HVega.put(-V1/sqrt(2.0), 1, 0);
  HVega.put(-Vzz/3.0, 1, 1);
  HVega.put(-Vm1/sqrt(2.0), 1, 2);
  HVega.put(V2/3., 2, 0);                     // Added 1/3 Factor!
  HVega.put(V1/sqrt(2.0), 2, 1);
  HVega.put(Vzz/6.0, 2, 2);
  HVega *= G.wG();
  }
 else if(I == 1.5)
  {
  HVega = matrix(4,4);
  HVega.put(Vzz/2.0, 0, 0);
  HVega.put(Vm1/sqrt(3.0), 0, 1);
  HVega.put(Vm2/sqrt(3.0), 0, 2);
  HVega.put(0.0, 0, 3);
  HVega.put(-V1/sqrt(3.0), 1, 0);
  HVega.put(-Vzz/2.0, 1, 1);
  HVega.put(0.0, 1, 2);
  HVega.put(Vm2/sqrt(3.0), 1, 3);
  HVega.put(V2/sqrt(3.0), 2, 0);
  HVega.put(0.0, 2, 1);
  HVega.put(-Vzz/2.0, 2, 2);
  HVega.put(-Vm1/sqrt(3.0), 2, 3);
  HVega.put(0.0, 3, 0);
  HVega.put(V2/sqrt(3.0), 3, 1);
  HVega.put(V1/sqrt(3.0), 3, 2);
  HVega.put(Vzz/2.0, 3, 3);
  HVega *= G.wG();
  }
//                    Generate H According To Vega's Equation (28)
 matrix HV = Vzz*(3.*IZ*IZ-(I*(I+1.))*IE);
 HV += (Vxx-Vyy)*(IX*IX-IY*IY);
 HV += 2*Vxy*(IX*IY-IY*IX);
 HV += 2*Vxz*(IX*IZ-IZ*IX);
 HV += 2*Vyz*(IY*IZ-IZ*IY);
 HV *= G.wG()/6.0;
//                    Output the Results for Visual Comparison
 cout << "\n\t\t\t\tGAMMA's G H:\t" << HG;
 cout << "\n\t\t\t\tGAMMA's Other G H:\t" << HGB;
 cout << "\n\t\t\t\tMan's G H(a):\n\t" << HMa;
 cout << "\n\t\t\t\tMan's G H(b):\n\t" << HMb;
 if(I == 1.0 || I == 1.5)
  cout << "\n\t\t\t\tVega's G H:\n\t" << HVega;
 cout << "\n\t\t\t\tVega's Generic Guad H:\n\t" << HV;
 }
```

IntGu_LC1.cc

```
/* IntGu_LC1.cc
*****************************************************************-*-c++-*-
**
**
**                    Test Program for the GAMMA Library
**
**                G Interaction Literature Comparison 1
**
**
**
**  This program checks the G interaction class IntG in
**
**  GAMMA.  In particular it looks to see how well the class parallels
**
**  the article by Alexander Vega -
**
**
**
**  "G Nuclei in Solids", Encyclopedia of Magnetic Resonance,
**
```

```
**  by Grant and Harris, Vol 6, Ped-Rel, page 3869-3889.
**
**
**
**  Specifcally, herein we generate the spatial tensor components of
**
**  an oriented G interaction and and compare the results to
**
**  A.Vega's equations (22-27) and 31 on pages 3884-3885.
**
**
**
** Author:     S.A. Smith
**
** Date:       10/11/96
**
** Update:     10/11/96
**
** Version:    3.6
**
** Copyright:   S. Smith.  You can modify this program as you see fit
**
**            for personal use, but you must leave the program intact
**
**            if you re-distribute it.
**
**
**
*************************************************************************
*******************/

#include <gamma.h>                                  // Include GAMMA

main (int argc, char* argv[])

 {
//                                  Construct A G Interaction

 int qn=1;
 double W;                                          // G frequency
 query_parameter(argc, argv, qn++,                  // Read in the coupling
          "\n\tG Frequency(kHz)? ", W);
 W *= 1.e3;                                         // Put this in Hz
 double Geta;
 query_parameter(argc, argv, qn++,                  // Read in the coupling
  "\n\tG Asymmetry [0, 1]? ", Geta);
 double Gtheta, Gphi;
 query_parameter(argc, argv, qn++,                  // Read in the angle
  "\n\tAngle down from z [0, 180]? ", Gtheta);
 query_parameter(argc, argv, qn++,                  // Read in the angle
    "\n\tAngle over from x [0, 360]? ", Gphi);
 double I=1.0;                                      // Use I=1, but this doesn't
 double GCC = wG2GCC(W, I);                         // Heres quad. coupling
 IntG G(I,GCC,Geta,Gtheta,Gphi);                    // matter for spatial parts
//                  Here Are Vegas V's According To Equations (22-27, 31)
//          Note We Change Sign On ETA As He Using A Different PAS Definition
 double Theta = G.theta()*DEG2RAD;
 double Phi = G.phi()*DEG2RAD;
 double Eta = -G.eta();
 double Stheta = sin(Theta);
 double Ctheta = cos(Theta);
 double C2phi = cos(2.*Phi);
 double S2phi = sin(2.*Phi);
 double Vxx = 0.5*(3.*Stheta*Stheta - 1. - Eta*Ctheta*Ctheta*C2phi);
 double Vxy = 0.5*Eta*Ctheta*S2phi;
 double Vxz = -0.5*(Stheta*Ctheta*(3.0 + Eta*C2phi));
 double Vyx = Vxy;
 double Vyy = 0.5*(-1. + Eta*C2phi);
 double Vyz = 0.5*Eta*Stheta*S2phi;
 double Vzx = Vxz;
 double Vzy = Vyz;
 double Vzz = 0.5*(3.*Ctheta*Ctheta - 1. - Eta*Stheta*Stheta*C2phi);
 complex V0(sqrt(1.5)*Vzz);
 complex V1(-Vxz, -Vyz);
 complex Vm1(Vxz, -Vyz);
 complex V2(0.5*(Vxx-Vyy), Vxy);
 complex Vm2(0.5*(Vxx-Vyy), -Vxy);
//                  Here Are The A's According To GAMMA G Interaction
//            Need To Scale Our A's By (1/2)/sqrt[5/(24*PI)] To Get Vega's V's
 double X = 0.5/RT5O24PI;
 double Thetad = G.theta();
 double Phid = G.phi();
 double AGxx = X*G.Axx(Thetad, Phid);
 double AGxy = X*G.Axy(Thetad, Phid);
 double AGxz = X*G.Axz(Thetad, Phid);
 double AGyy = X*G.Ayy(Thetad, Phid);
 double AGyx = X*G.Ayx(Thetad, Phid);
 double AGyz = X*G.Ayz(Thetad, Phid);
 double AGzz = X*G.Azz(Thetad, Phid);
 double AGzx = X*G.Azx(Thetad, Phid);
 double AGzy = X*G.Azy(Thetad, Phid);
//                  Here Are The A's According To GAMMA G Interaction
//            Need To Scale Our A's By (1/2)/sqrt[5/(24*PI)] To Get Vega's V's
 double AG1xx = X*G.Axx();
 double AG1xy = X*G.Axy();
 double AG1xz = X*G.Axz();
 double AG1yy = X*G.Ayy();
 double AG1yx = X*G.Ayx();
 double AG1yz = X*G.Ayz();
 double AG1zz = X*G.Azz();
```

```
 double AG1zx = X*G.Azx();
 double AG1zy = X*G.Azy();
//                    Here Are The A's According To GAMMA G Interaction
//          (Note That space_T Uses Azz>=Ayy>=Axx So ETA Opposite Vega's)
 space_T Agen = A2(0.0, 1.0, Geta);
 Agen = Agen.rotate(Phid, Thetad, 0.0);
 Cartesian(Agen);
//                          Output Everyone For A Visual Comparison
 cout << "\n   " << "       Vega" << "          IntGA"
                << "     IntGB" << "      space_T";
 cout << "\nVxx    " << form("%8.3f", Vxx) << "        " << form("%8.3f", AGxx)
       << "      " << form("%8.3f", AG1xx) << "     " << form("%8.3f", Agen.Ccomponent(0,0));
 cout << "\nVxy    " << form("%8.3f", Vxy) << "        " << form("%8.3f", AGxy)
       << "      " << form("%8.3f", AG1xy)  << "     " << form("%8.3f", Agen.Ccomponent(0,1));
 cout << "\nVxz    " << form("%8.3f", Vxz) << "        " << form("%8.3f", AGxz)
       << "      " << form("%8.3f", AG1xz) << "     " << form("%8.3f", Agen.Ccomponent(0,2));
 cout << "\nVyy    " << form("%8.3f", Vyy) << "        " << form("%8.3f", AGyy)
       << "      " << form("%8.3f", AG1yy) << "     " << form("%8.3f", Agen.Ccomponent(1,1));
 cout << "\nVyx    " << form("%8.3f", Vyx) << "        " << form("%8.3f", AGyx)
       << "      " << form("%8.3f", AG1yx) << "     " << form("%8.3f", Agen.Ccomponent(1,0));
 cout << "\nVyz    " << form("%8.3f", Vyz) << "        " << form("%8.3f", AGyz)
       << "      " << form("%8.3f", AG1yz) << "     " << form("%8.3f", Agen.Ccomponent(1,2));
 cout << "\nVzz    " << form("%8.3f", Vzz) << "        " << form("%8.3f", AGzz)
       << "      " << form("%8.3f", AG1zz) << "     " << form("%8.3f", Agen.Ccomponent(2,2));
 cout << "\nVzx    " << form("%8.3f", Vzx) << "        " << form("%8.3f", AGzx)
       << "      " << form("%8.3f", AG1zx) << "     " << form("%8.3f", Agen.Ccomponent(2,0));
 cout << "\nVzy    " << form("%8.3f", Vzy) << "        " << form("%8.3f", AGzy)
       << "      " << form("%8.3f", AG1zy) << "     " << form("%8.3f", Agen.Ccomponent(2,1));
 cout << "\nV0 " << V0 << "  " << X*G.A0(Thetad, Phid)
       << "  " << X*G.A0() << "  " << Agen.component(2,0);
 cout << "\nV1 " << V1 << "  " << X*G.A1(Thetad, Phid)
       << "  " << X*G.A1() << "  " << Agen.component(2,1);
 cout << "\nV-1" << Vm1 << "  " << X*G.Am1(Thetad, Phid)
       << "  " << X*G.Am1() << "  " << Agen.component(2,-1);
 cout << "\nV2 " << V2 << "  " << X*G.A2(Thetad, Phid)
       << "  " << X*G.A2() << "  " << Agen.component(2,2);
 cout << "\nV-2" << Vm2 << "  " << X*G.Am2(Thetad, Phid)
       << "  " << X*G.Am2() << "  " << Agen.component(2,-2);
 cout << "\n\n\n";
 }
```

/IntGu_PCT0.cc

```
/* IntGu_PCT0.cc
*************************************************************-*-c++-*-
**
**
**                    Example Program for the GAMMA Library
**
**
**
**  This program calculates a powder average for a single spin which
**
**  is associated with a G interaction.  The high field
**
**  approximation is invoked in that the G Hamiltonian is
**
**  treated as a perturbation to the Zeeman Hamiltonian and taken to
**
**  second order.  Only G Hamiltonian terms which are
**
**  rotationally invariant about the field axis (z) are maintained.
**
**  Furthermore, only the central transtion will be considered.
**
**
**
**  This will program is similar to IntGu_Pow2.cc but restricts the
**
**  computation to only the central transition. In turn, that means
**
**  only spins with I=m*1/2 where m is odd and larger than 1 are valid.
**
**  Analog formula will be used to construct the spectrum.
**
**
**
**  Later version of GAMMA will have the functions “scale” and “sum”
**
**  in the library itself, so you will need to remove them from this
**
**  program in that event.
**
**
**
** Author:     S.A. Smith
**
** Date:       10/15/96
**
** Update:     10/15/96
**
** Version:    3.6
**
** Copyright:   S. Smith.  You can modify this program as you see fit
**
**          for personal use, but you must leave the program intact
**
**          if you re-distribute it.
**
**
**
*********************************************************************
*******************/

#include <gamma.h>                                  // Include GAMMA

void addW(row_vector& vx, double Fst, double Ffi, double F, double I)

    // Input     vx    : A row vector
    //           Fst   : Frequency of 1st point of vx (Hz)
    //           Ffi   : Frequency of last point of vx (Hz)
    //           F     : Transition frequency (Hz)
    //           I     : Transition intensity
    // Output    void  : The transition is added to the
    //                   row vector (as a Dirac delta).
    //Note             : To start one should zero vx

 {
 if(F<Fst || F>Ffi) return;                         // Insure its in range
 double Nm1 = double(vx.size()-1);                  // Freq. -> point conversion
 double m = Nm1/(Ffi-Fst);                          // Slope Freq. -> pt
 double dpt = m*(F-Fst);                            // Point index of F
 int pt = int(dpt);                                 // Main point for F
 double drem = dpt - pt;                            // Part which isn’t
 if(!drem) vx.put(vx.get(pt)+I, pt);                // Add if on a point
 else if(drem > 0)                                  // If in between points
  {                                                 // then just split it up
  vx.put(vx.get(pt)+(1.0-drem)*I, pt);              // between the two
  vx.put(vx.get(pt+1)+drem*I, pt+1);
  }
 else
  {
  vx.put(vx.get(pt)+(1.0+drem)*I, pt);
  vx.put(vx.get(pt-1)-drem*I, pt-1);
  }
 return;
 }


main (int argc, char* argv[])

 {
 cout << “\n\t\tG Central Transition Powder Pattern”;
```

```
 cout << "\n\t\t    (131Xe:3/2, 55Mn:5/2, 51V:7/2, ...)\n";
//                          First Make A G Interaction
 String Iso;                                        // Isotope of spin
 int qn=1;                                          // Guery index
 query_parameter(argc, argv, qn++,                  // Get the isotope type
  "\n\tIsotope Type [131Xe, 55Mn, 51V, ....]? ", Iso);
 double wG;                                         // Set Guad. frequency
 query_parameter(argc, argv, qn++,                  // Get the G coupling
     "\n\tG Frequency (kHz)? ", wG);
 wG *= 1.e3;                                        // Switch to Hz
 double eta;                                        // Set Guad. frequency
 query_parameter(argc, argv, qn++,                  // Get the G coupling
     "\n\tG eta Value [0, 1]? ", eta);
 double Om;
 query_parameter(argc, argv, qn++,                  // Get the field strength
            "\n\tLarmor Frequency (MHz)? ", Om);
 Om *= 1.e6;                                        // Switch to MHz
 Isotope S(Iso);                                    // Make a spin isotope
 double I = S.qn();                                 // This is isotope I value
 IntG G(I,wG2GCC(wG, I), eta);                      // Set a Guad interaction
 if(!int(2*I)%2)
   {
   cout << "\n\n\tSorry, I Must Be m*1/2, m Odd!\n\n";
   exit(-1);
   }
//                      Set Things Up For The Powder Average
 int npts = 4096;                                   // Block size
 int Ntheta, Nphi=0;                                // Angle increment counts
 query_parameter(argc, argv, qn++,                  // Get the theta increments
   "\n\t# Theta (z down) Increments Spanning [0, 180]? ", Ntheta);
 if(eta)
   query_parameter(argc, argv, qn++,                // Get the phi increments
     "\n\t# Phi (x over) Increments Spanning [0, 360)? ", Nphi);
 matrix ABC = G.wGcentral(Ntheta, Nphi);            // Prep. for 2nd order shifts
//                          Powder Averaging
//            Angle theta Is Down From The +z Axis, Angle phi Over From +x
// Note that since the 2nd order shift Wcentral(theta, phi) is symmetric with
// respect to both angles we need only average over parts of both angle ranges.
// For theta this means we sum the results from angles [0, 90) + half the result
// at 90.  Twice that sum would produce the total theta average over [0, 180].
// For phi we usually average [0, 360) so this is reduced to a sum over 3/4 the
// result at 0 + the results from angles (0, 90) + 1/2 the result at 90.  Four
// times that sum would produce the total phi average over [0, 360).

 double dthe, Nm1o2 = double(Ntheta-1.0)/2.0;      // For powder average
 double dphi, Nm2o4 = double(Nphi)/4.0;             // For powder average
 int theta, phi;                                    // Orientation angles
 double W, WG = G.wG();                             // Base Guad. frequency
 double Ifact = I*(I+1) - 0.75;                     // Part of the prefactor
 double prefact = -WG*WG*Ifact/Om;                  // Majority of the prefact
 double Aaxis = (-1.0/9.0)*prefact;                 // For plot scaling
 double Ctheta, Stheta, Cthetasq, Ctheta4;          // We'll need these
 double Fst = -2.5*Aaxis;                           // Starting plot limit
 double Ffi = 1.5*Aaxis;                            // End plot limit
 row_vector data(npts, complex0);                   // Array for spectrum
 for(theta=0; theta<Ntheta; theta++)                // Loop over theta angles
   {
   dthe = double(theta);
   if(dthe <= Nm1o2)                                // Only look upper half
     {                                              // of the sphere
     Ctheta = ABC.getRe(0,theta);                   // Scale factor cos(theta)
     Stheta = ABC.getRe(1,theta);                   // Scale factor sin(theta)
     Cthetasq = Ctheta*Ctheta;                      // cosine(theta)^2
     if(dthe == Nm1o2) Stheta *= 0.5;               // Half scale if theta=90
     if(!eta)                                       // Without eta, no phi
       {                                            // averaging is needed
       W=(prefact/16.)*(1.-Cthetasq)*(9.*Cthetasq-1.);  // Here is W adjustment
       addW(data, Fst, Ffi, W, Stheta);             // Add transition to spectrum
       }
     else
       {
       cout.flush();
       Ctheta4 = Cthetasq*Cthetasq;                 // cosine(theta)^4
       for(phi=0; phi<Nphi; phi++)                  // Loop over phi angles
         {
         dphi = double(phi);                        // Phi index as double
         if(dphi <= Nm2o4)                          // Only sum 1st quarter
           {
           if(!phi) Stheta *= 0.75;                 // 3/4 scale if phi=0
           else if(dphi == Nm2o4) Stheta *= 0.5;    // 1/2 scale if phi=90
           W = ABC.getRe(2,phi)*Ctheta4;            // A part of W
           W += ABC.getRe(3,phi)*Cthetasq;          // B part of W
           W += ABC.getRe(4,phi);                   // C part of W
           W *= (prefact/6.);                       // Scale
           addW(data, Fst, Ffi, W, Stheta);         // Add transition to spectrum
           }
         }
       }
     }
   }
 double lb = 40.0;                                  // Set a line broading factor
 cout << "\n\n\tDone With Discrete Powder Average. Processing...";
 cout.flush();
 data = IFFT(data);                                 // Put into time domain
 exponential_multiply(data,-lb);                    // Apodize the "FID"
 data = FFT(data);                                  // Put back into frequency domain
 GP_1D("spec.asc", data, 0, -2.5, 1.5);             // Output the points in ASCII
 GP_1Dplot("spec.gnu", "spec.asc");                 // Call Gnuplot and make plot now
 }
```