"Pray, Mr Babbage, if you put into the machine wrong figures, will the right answers come out?"

Members of UK Parliament, to Charles Babbage, in 1860

Spinach kernel tools: relaxation theory

Numerical implementation of Redfield theory in Spinach kernel

Ilya Kuprov, University of Southampton

Numerical pathway: the bad old days

Redfield's relaxation theory integral is a serious problem:

$$\hat{\hat{R}} = -\sum_{kmpq}\int_0^{\infty} G_{kmpq}(\tau)\hat{Q}_{km}e^{-i\hat{H}_0\tau}\hat{Q}_{pq}^{\dagger}e^{i\hat{H}_0\tau}d\tau$$

The same applies to rotating frame transformations

$$\mathbf{H}^{(0)} = \frac{\omega_{\mathrm{E}}}{2\pi}\int_0^{2\pi/\omega_{\mathrm{E}}} e^{i\mathbf{H}_0 t}\mathbf{H}_1 e^{-i\mathbf{H}_0 t}dt$$

$$\mathbf{H}^{(1)} = -\frac{i}{2}\frac{\omega_{\mathrm{E}}}{2\pi}\int_0^{2\pi/\omega_{\mathrm{E}}} dt_2\int_0^{t_2} dt_1\left[e^{i\mathbf{H}_0 t_2}\mathbf{H}_1 e^{-i\mathbf{H}_0 t_2}, e^{i\mathbf{H}_0 t_1}\mathbf{H}_1 e^{-i\mathbf{H}_0 t_1}\right]$$

...and to various kinetic processes like CIDNP:

$$Y_{\mathrm{S}}(t) = k_{\mathrm{S}}\int_0^t \langle \mathbf{P}_{\mathrm{S}}|e^{-i\mathbf{L}t'}|\boldsymbol{\rho}_0\rangle dt' \qquad Y_{\mathrm{T}}(t) = k_{\mathrm{T}}\int_0^t \langle \mathbf{P}_{\mathrm{T}}|e^{-i\mathbf{L}t'}|\boldsymbol{\rho}_0\rangle dt'$$

The same type of integral every time!

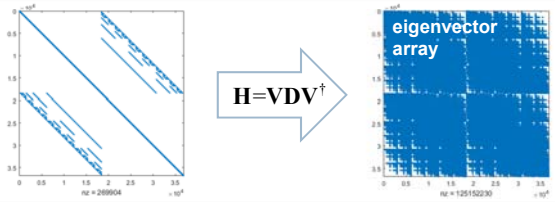
- D.L. Goodwin, I. Kuprov, *J. Chem. Phys.* **143** 084113 (2015).

Numerical pathway: slightly better old days

The usual way is to diagonalise the Hamiltonian and reduce the integrals to FFTs…

…but for any spin system with more than ten spins, *diagonalisation* is a dirty word!

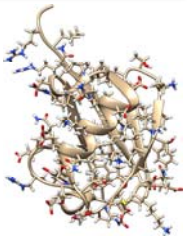
(instant memory overflow and crash boom bang)



Until 2011, Redfield's relaxation theory was effectively limited to about 6 spins.

Diagonalization-free implementation of spin relaxation theory for large spin systems

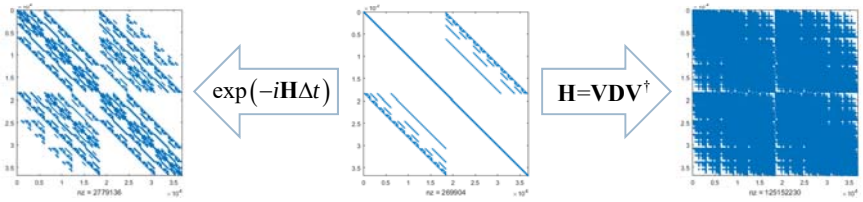
Ilya Kuprov *
Oxford e-Research Centre, University of Oxford, 7 Keble Road, Oxford OX1 3QG, UK



I. Kuprov, *J. Magn. Reson.* **209** (2011) 31-38.

Numerical pathway: exp *vs.* diag

Spin Hamiltonian exponentiation is much cheaper than diagonalisation!



Spin Hamiltonians are *guaranteed* to be sparse in the Pauli basis.

Exponentiation preserves matrix sparsity, and the two paths through quantum mechanics are formally equivalent:

$$|\psi(t)\rangle = \exp\left[-i\hat{H}t\right]|\psi(0)\rangle \quad \Leftarrow \quad \frac{\partial}{\partial t}|\psi\rangle = -i\hat{H}|\psi\rangle \quad \Rightarrow \quad \begin{matrix}\hat{H}|\varphi_k\rangle = \omega_k|\varphi_k\rangle \\ |\psi(t)\rangle = \sum_k \alpha_k e^{-i\omega_k t}|\varphi_k\rangle\end{matrix}$$

In fact, exponentiation is perferred because most experiments are time-domain!

I. Kuprov, *J. Magn. Reson.* **209** (2011) 31-38.

Numerical pathway: matrix exponential integrals

In 1978, Charlie van Loan published the following "auxiliary matrix" relation:

$$\exp\left[\begin{pmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{0} & \mathbf{C}\end{pmatrix}t\right]=\begin{pmatrix}e^{\mathbf{A}t} & e^{\mathbf{A}t}\int_0^t e^{-\mathbf{A}t_1}\mathbf{B}e^{\mathbf{C}t_1}dt_1\\ \mathbf{0} & e^{\mathbf{C}t}\end{pmatrix}$$

Notice the similarity to Redfield and CIDNP integrals!

$$\mathbf{R}=-\sum_{kmpq}\mathbf{Q}_{km}\int_0^\infty G_{kmpq}(\tau)e^{-i\mathbf{H}_0\tau}\mathbf{Q}_{pq}^{\dagger}e^{i\mathbf{H}_0\tau}d\tau$$

$$Y_S(t)=\int_0^t \mathrm{Tr}\left[e^{-i\mathbf{H}t'}\boldsymbol{\rho}_0 e^{i\mathbf{H}t'}\mathbf{P}_S\right]f(t')dt'$$

In both cases, the functions $G(\tau)$ and $f(t)$ are known to be combinations of decaying exponentials:

$$G(t)=\sum_n a_n e^{-\lambda_n t},\qquad f(t)=\sum_n k_n e^{-k_n t}$$

...and so we are getting, in the CIDNP case:

$$\int_0^t e^{-i\mathbf{H}t'}\boldsymbol{\rho}_0 e^{i(\mathbf{H}+ik\mathbf{1})t}dt=\mathbf{A}^{\dagger}\mathbf{B},\qquad \begin{pmatrix}\mathbf{A} & \mathbf{B}\\ 0 & \mathbf{C}\end{pmatrix}=\exp\left[\begin{pmatrix}i\mathbf{H} & \boldsymbol{\rho}_0\\ 0 & i\mathbf{H}-k\mathbf{1}\end{pmatrix}t\right]$$

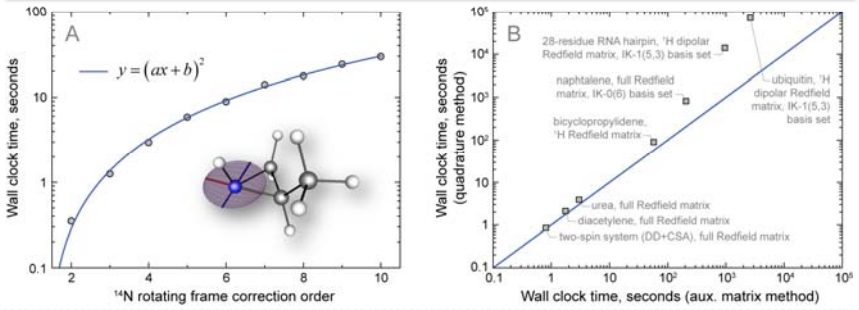
D.L. Goodwin, I. Kuprov, *J. Chem. Phys.* **143** 084113 (2015).

Numerical pathway: performance

The auxiliary matrix method has lead to a massive reduction in Redfield module resource requirements in *Spinach* (1 TB down to 128 GB of RAM for ubiquitin).

$$\mathbf{R}=-\sum_{kmpq}\mathbf{Q}_{km}\int_0^\infty G_{kmpq}(\tau)e^{-i\mathbf{H}_0\tau}\mathbf{Q}_{pq}^{\dagger}e^{i\mathbf{H}_0\tau}d\tau \qquad \exp\left[\begin{pmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{0} & \mathbf{C}\end{pmatrix}t\right]=\begin{pmatrix}e^{\mathbf{A}t} & e^{\mathbf{A}t}\int_0^t e^{-\mathbf{A}t_1}\mathbf{B}e^{\mathbf{C}t_1}dt_1\\ \mathbf{0} & e^{\mathbf{C}t}\end{pmatrix}$$

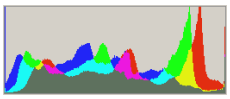
Very large spin systems and high rotating frame correction orders are now possible:



D.L. Goodwin, I. Kuprov, *J. Chem. Phys.* **143** 084113 (2015).

Analytical pathway: pattern matching

statistical analysis



Two complicated 2D luminosity patterns between 300 and 700 nm, hard to say anything definite

pattern-matching

Nic Wagner-Rundell and Peter Hore
(correct answer)

Integration by a machine:

$$\int_{-1}^{1} ae^{-x^4}dx = \lim_{N\to\infty}\frac{1}{2N}\sum_{n=-N}^{N} ae^{-(n/N)^4} = \ldots$$

Integration by a human:

Rule 1: $\int af(x)dx \to a\int f(x)dx$

Rule 2: $\int_{-1}^{1} e^{-x^n}dx \to \left(1+(-1)^n\right)^{-\frac{1}{n}}\Gamma\left(1+\frac{1}{n}\right)$

A typical pattern matching rule:

$$\overline{\mathfrak{D}^{(l)}_{a,b}(t)\mathfrak{D}^{(l)*}_{c,d}(t+\tau)} \to \frac{\delta_{a,c}\delta_{b,d}}{2l+1}e^{-\frac{\tau}{\tau_c}}$$

(an approximation pattern)

Analytical pathway: stage 1

Stage 1: preprocessing the Hamiltonian to expose rotation group properties

$$\hat{\tilde{S}}\cdot\mathbf{A}\cdot\hat{\tilde{L}} = a\hat{S}_X\hat{L}_X + b\hat{S}_Y\hat{L}_Y + c\hat{S}_Z\hat{L}_Z =$$
$$= \frac{a-b}{2}\hat{T}_{2,-2} + \frac{a-b}{2}\hat{T}_{2,2} + \frac{2c-(a+b)}{\sqrt{6}}\hat{T}_{2,0}$$

Transforming the eigenframe expressions to the irreducible spherical tensor notation (equally applicable to linear, bilinear and quadratic spin interactions).

$$\hat{\tilde{R}}(\alpha,\beta,\gamma)\hat{T}_{l,m} = \sum_{m'=-l}^{l}\hat{T}_{l,m'}\mathfrak{D}^{(l)}_{m',m}(\alpha,\beta,\gamma)$$

$$\hat{\tilde{R}}_{pos}\left(\hat{\tilde{S}}\cdot\mathbf{A}\cdot\hat{\tilde{L}}\right) = \sum_{m=-2}^{2}\hat{T}_{2,m}\Phi_m$$

$$\Phi_m = \frac{a-b}{2}\left(\mathfrak{D}^{(2)}_{m,-2} + \mathfrak{D}^{(2)}_{m,2}\right) + \frac{2c-(a+b)}{\sqrt{6}}\mathfrak{D}^{(2)}_{m,0}$$

Setting up internal rotations (to position interactions in a given molecule) and an overall molecular rotation. The angles corresponding to internal rotations may optionally be treated as fixed.

What is happening:

1. Translation of Hamiltonian into operators that have regular rotation properties.
2. Interaction tensor positioning in the molecular frame and overall rotation setup.
3. Representation of static rotations in the most compact and general form.

Analytical pathway: stage 2

Stage 2: automatic matching of correlation function patterns in the BRW expressions

$$\mathfrak{M}^{(l\_)}_{a\_,b\_}(0)\mathfrak{M}^{(k\_)*}_{c\_,d\_}(\tau) \triangleq \frac{\delta_{l,k}}{2l+1} g_{abcd}(\tau)$$

$$\mathfrak{M}^{(l\_)*}_{a\_,b\_}(0)\mathfrak{N}^{(k\_)}_{c\_,d\_}(\tau) \triangleq 0$$

$$\mathfrak{N}^{(l\_)*}_{a\_,b\_}(0)\mathfrak{M}^{(k\_)}_{c\_,d\_}(\tau) \triangleq 0$$

The so-called 'upvalues' are used, which are instructions to the symbolic processing kernel to do the following throughout thecalculations: to keep an eye on the occurrence of Wigner function products that match the left hand side, and on encountering a match to replace it with the right hand side. The variables with trailing underscores may stand for anything and are simply carried over to the right hand side.

Programmed in just one line of Mathematica syntax:

```
M /: M[l_, a_, b_, 0] Conjugate[M[k_, c_, d_, τ]] :=
     KroneckerDelta[l,k]KroneckerDelta[a,c]KroneckerDelta[b,d]/(2l+1) G[τ];
```

What is happening:

1. The patterns of Wigner functions given above are matched and replaced.
2. Correlation functions of independent rotations are zeroed.
3. Static (relative) interaction tensor rotations survive, giving correct account of cross-correlations
4. Basic simplification and rearrangement is performed.

Analytical pathway: stage 3

Stage 3: automatic integration of BRW expressions using a dedicated symbolic integrator

$$\Upsilon(a\_ + b\_) := \Upsilon(a) + \Upsilon(b)$$

$$\Upsilon(a\_\, b\_) := a\,\Upsilon(b) \quad \text{if } a \in \mathbb{C}$$

$$\Upsilon\left(e^{i\omega\_\tau} g(\tau)\right) := J(\omega)$$

Due to lack of knowledge of the functional form of the correlation functions, a brute-force symbolic integration would fail. A much improved solution at this stage is not to rely on explicit integration, but rather to create a dedicated integrator which is only aware of a specific set of properties and is therefore very fast.

Implementing linearity, multiplication by a scalar, and a rule for the exponential:

```
BRWIntegrate[A_ + B_] := BRWIntegrate[A] + BRWIntegrate[B];
BRWIntegrate[A_ B_] := A BRWIntegrate[B] /; A∈Complexes∨A∈Parameters
BRWIntegrate[Power[A_, k_] B_] :=
   Power[A, k] BRWIntegrate[B] /; k∈Integers && (A∈Complexes∨A∈Parameters)
BRWIntegrate[e^A-G[τ]] := J[Simplify[-iA/τ]];
BRWIntegrate[G[τ]] := J[0];
BRWIntegrate[0] := 0;
```

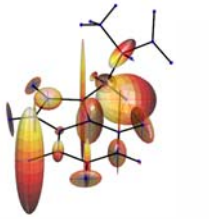
What is happening:

1. The BRW master equation is integrated term by term.
2. Cosmetic simplification is performed.

Example 1: relaxation due to rhombic Zeeman interaction

This is literally it, ten lines of Mathematica code in addition to the above:

```
Hst = ω1 Lz + ω2 Sz + a Lz . Sz;
Hdn[t_] := Rh/2 Sum[T[2,k] M[2,k,-2,t], {k,-2,2}] + Rh/2 Sum[T[2,k] M[2,k,2,t]] +
           Ax/Sqrt[6] Sum[T[2,k] M[2,k,0,t], {k,-2,2}];
Dcomm[ρ_] :=
  - Comm[Hdn[0], Comm[ConjugateTranspose[
      MatrixExp[iHstτ] . Hdn[τ] . MatrixExp[-iHstτ]], ρ]];
Rate[A_, B_] := Scal[Dcomm[A], B] / Sqrt[Scal[B, B] Scal[A, A]] //TrigToExp//ExpandAll//BRWIntegrate//Simplify;
```



Some illustrative output and timings:

```
Rate[Lz,Lz] // Timing
```

$$0.078 \text{ Second}, -\frac{1}{10}(J[-\omega_1]+J[\omega_1])(\Phi L[0]^2 - 2\Phi L[-1]\Phi L[1]+2\Phi L[-2]\Phi L[2])$$

```
Rate[Lz,Lx] // Timing
```

$$0.047 \text{ Second}, -\frac{1}{30}(4J[0]+3J[-\omega_1])(\Phi L[0]^2 - 2\Phi L[-1]\Phi L[1]+2\Phi L[-2]\Phi L[2])$$

50 milliseconds… :)

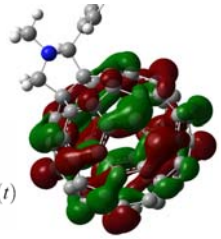
Example 2: relaxation due to ZFS anisotropy

ZFS anisotropy is the dominant electron relaxation mechanism in e.g. derivatised endofullerenes. For the spin-3/2 ($N@C_{60}$) case:

$$\hat{H} = \omega_1 \hat{S}_Z + \omega_2 \hat{L}_Z + a\hat{S}_Z\hat{L}_Z + \hat{\vec{S}} \cdot \mathbf{Z} \cdot \hat{\vec{S}}$$

$$\hat{H}_0 = \omega_1 \hat{S}_Z + \omega_2 \hat{L}_Z + a\hat{S}_Z\hat{L}_Z,$$

$$\hat{H}_1(t) = \frac{Rh}{2}\sum_{m'=-2}^{2} \hat{T}_{2,m'}\mathfrak{M}^{(2)}_{m',-2}(t) + \frac{Rh}{2}\sum_{m'=-2}^{2} \hat{T}_{2,m'}\mathfrak{M}^{(2)}_{m',2}(t) + \frac{Ax}{\sqrt{6}}\sum_{m'=-2}^{2} \hat{T}_{2,m'}\mathfrak{M}^{(2)}_{m',0}(t)$$



Resulting relaxation rates and timings:

```
Rate[Sx,Sx] // Timing
```

$$0.328 \text{ Second}, -\frac{Ax^2+3Rh^2}{75}(J[-a-\omega_1]+J[a-\omega_1]+4J[2a-2\omega_1]+4J[-2\omega_1]+J[-\omega_1]+J[\omega_1] + 4J[2\omega_1]+J[-a+\omega_1]+4J[-2a-2\omega_1]+J[a+\omega_1]+4J[2a+2\omega_1]+4J[-2a+2\omega_1]),$$

```
Rate[Sz,Sz] // Timing
```

$$0.297 \text{ Second}, -\frac{Ax^2+3Rh^2}{75}(9J[0]+3J[-a-\omega_1]+3J[a-\omega_1]+2J[2a-2\omega_1]+2J[-2\omega_1] + 3J[-\omega_1]+2J[\omega_1]+2J[-a+\omega_1]+2J[-2a-2\omega_1]+2J[a+\omega_1]),$$

In a simplified form (HFC=0):

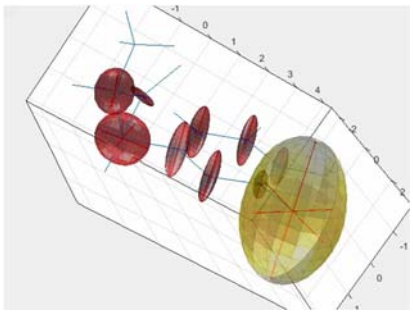
$$-\frac{2}{25}(Ax^2+3Rh^2)(J[\omega_1]+4J[2\omega_1])$$

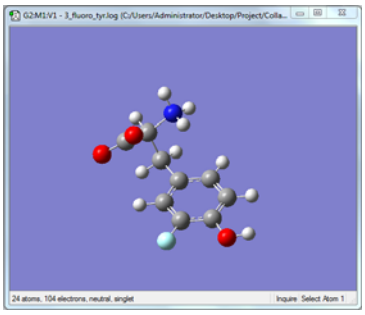
$$-\frac{1}{25}(Ax^2+3Rh^2)(3J[0]+5J[\omega_1]+2J[2\omega_1])$$

Two completely dissimilar methods yield the same numbers in the end – both correct!

Practical example: ^{13}C-^{19}F TROSY in fluorotyrosine

CSA tensors are rarely available from experiments, and so DFT must be run. Many methods benchmarked in the literature, GIAO M06/cc-pVTZ + SMD used here.





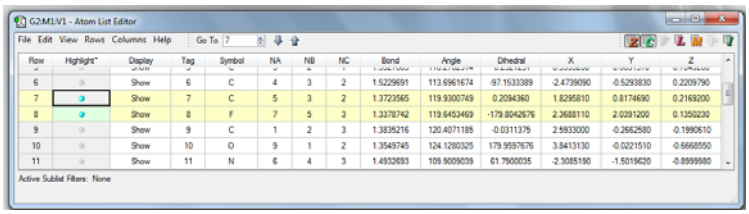
Spinach directly imports *Gaussian* logs:

```
% Read 3-fluorotyrosine DFT calculation
[~,inter_dft]=g2spinach(gparse('3_fluoro_tyr.log'),...
     {{'C','13C'},{'F','19F'}},[186.38 192.97],[]);
```

- Most chemistry departments would have a *Gaussian* license.

Practical example: ^{13}C-^{19}F TROSY in fluorotyrosine

We need to find which of the many imported atoms are our C-F pair…




```
>> inter_dft.coordinates

    2.0334   -1.5277   -0.1028
    0.7482   -1.6888    0.3888
   -0.0101   -0.6003    0.7966
   -1.4162   -0.7823    1.2871
    0.5555    0.6692    0.7049
   -2.4739   -0.5294    0.2210
    1.8296    0.8175    0.2169  <<<<
    2.3688    2.0391    0.1350  <<<<
    2.5933   -0.2663   -0.1991
   -2.4178    0.8955   -0.3380
```

Many ways of specifying the system:

1. Import Gaussian log.
2. Look inside the log (it is a text file) and copy necessary data, then use standard input syntax.
3. Copy necessary data out of the GaussView user interface and use the standard input syntax.

- Some experience with *Matlab*, *Gaussian*, and *Spinach* is inevitably required here.

Practical example: ^{13}C-^{19}F TROSY in fluorotyrosine

Get the data into Spinach one way or another…

```
% Extract coordinates and CSAs
sys.isotopes={'19F','13C'};
inter.zeeman.matrix=cell(1,2);
inter.zeeman.matrix{1}=inter_dft.zeeman.matrix{8};
inter.zeeman.matrix{2}=inter_dft.zeeman.matrix{7};
inter.coordinates=cell(2,1);
inter.coordinates{1}=inter_dft.coordinates{8};
inter.coordinates{2}=inter_dft.coordinates{7};
```

Specify other parameters (standard syntax – see manual):

```
% Relaxation theory
inter.relaxation={'redfield'};
inter.rlx_keep='labframe';
inter.equilibrium='zero';
inter.tau_c={25e-9};
```

```
% Basis set
bas.formalism='sphten-liouv';
bas.approximation='none';

% Spinach housekeeping
spin_system=create(sys,inter);
spin_system=basis(spin_system,bas);
```

- Just a matter of following the standard syntax and *Spinach* input grumbles.

Practical example: ^{13}C-^{19}F TROSY in fluorotyrosine

Get the relaxation superoperator (yes, *that* simple):

```
% Relaxation superoperator
R=relaxation(spin_system);
```

Ask *Spinach* for the states of interest and normalise them:

```
% States of interest
Fp=state(spin_system,{'L+'},{1});
Fm=state(spin_system,{'L-'},{1});
Fz=state(spin_system,{'Lz'},{1});
Fy=(Fp-Fm)/2i;
Cp=state(spin_system,{'L+'},{2});
Cm=state(spin_system,{'L-'},{2});
Cz=state(spin_system,{'Lz'},{2});
Cy=(Cp-Cm)/2i;
CzFz=state(spin_system,{'Lz','Lz'},{1,2});
CzFp=state(spin_system,{'L+','Lz'},{1,2});
CzFm=state(spin_system,{'L-','Lz'},{1,2});
CzFy=(CzFp-CzFm)/2i;
```

```
% Normalisation
Fy=Fy/norm(full(Fy),2);
Fz=Fz/norm(full(Fz),2);
Cy=Cy/norm(full(Cy),2);
Cz=Cz/norm(full(Cz),2);
CzFz=CzFz/norm(full(CzFz),2);
CzFy=CzFy/norm(full(CzFy),2);
etc.
```

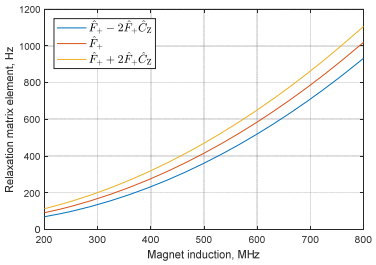
$$a_{nk}=\frac{\langle n|\mathbf{A}|k\rangle}{\sqrt{\langle n|n\rangle}\sqrt{\langle k|k\rangle}}$$

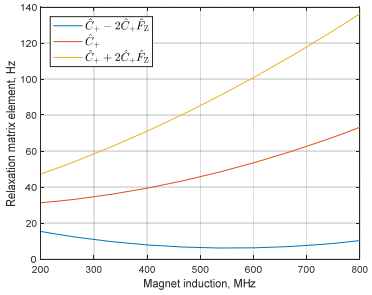
- Relaxation superoperator is huge. We need only a few specific elements…

Practical example: ^{13}C-^{19}F TROSY in fluorotyrosine

Compute the rates and plot the results against the field:

```
% Relaxation rates
disp(['Cz:    ' num2str(Cz'*R*Cz)]);
disp(['Cy:    ' num2str(Cy'*R*Cy)]);
etc.
```





Relaxation superoperator contains information about every relaxation process in the system. You need individual matrix elements.

- To account for secondary processes (remote dipolar, *etc.*) import more atoms.