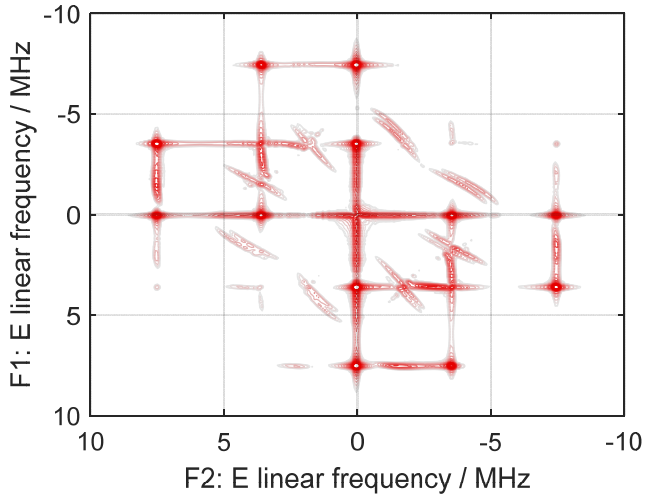
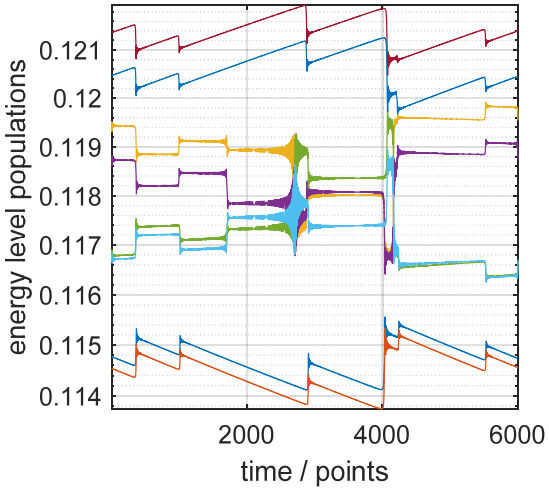
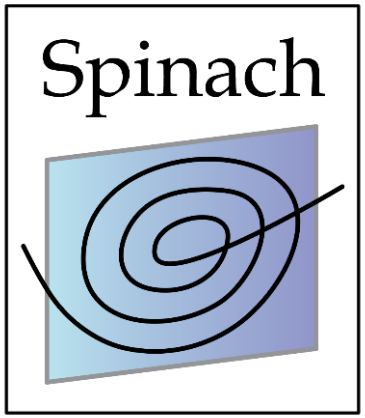






Getting started with pulse sequence modelling

Ilya Kuprov
University of Southampton

sys.*	General system specification (magnet, isotopes, labels, tolerances, algorithm options, output files, *etc.*)
inter.*	Interactions, chemical kinetics, relaxation theory options, spin temperature, order matrix, *etc.*
bas.*	Simulation formalism, approximation options, permutation symmetry, conservation law specification, *etc.*

```
% Magnet field
sys.magnet=0.33898;

% Isotope list
sys.isotopes={'E','14N','19F','1H','1H','1H','1H'};

% Basis set
bas.formalism='sphten-liouv';
bas.approximation='none';
bas.sym_group={'S2','S2'};
bas.sym_spins={[4 5],[6 7]};

% Zeeman interactions
inter.zeeman.eigs=cell(7,1);
inter.zeeman.euler=cell(7,1);
inter.zeeman.eigs{1}=[2.0032 2.0012 2.0097];
```

```
% Relaxation superoperator
inter.relaxation='redfield';
inter.rlx_keep='secular';
inter.tau_c={80e-12};
```

```
% Spin-spin couplings
inter.coupling.eigs=cell(7,7);
inter.coupling.euler=cell(7,7);
inter.coupling.eigs{1,2}=(40.40+[24 -12 -12])*1e6;
inter.coupling.eigs{1,3}=(22.51+[34.9 -19.8 -15])*1e6;
inter.coupling.eigs{1,4}=[9.69 9.69 9.69]*1e6;
inter.coupling.eigs{1,5}=[9.69 9.69 9.69]*1e6;
inter.coupling.eigs{1,6}=[3.16 3.16 3.16]*1e6;
inter.coupling.eigs{1,7}=[3.16 3.16 3.16]*1e6;

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

create()	Processing of spin system structure and interactions, input checking. Creates the *spin_system* data structure.
basis()	Processing of basis set options, approximations, symmetry and conservation laws. Updates *spin_system* data structure.
assume()	Case-specific simulation assumptions (*'labframe', 'nmr', 'endor', etc.*) that have an effect on the Hamiltonian.

hamiltonian()	Returns the isotropic Hamiltonian and the irreducible components of the anisotropic part.
relaxation()	Returns the relaxation superoperator using the theory specified by the user.
equilibrium()	Returns the thermal equilibrium density matrix or state vector at the specified temperature.
operator()	Returns user-specified operators or superoperators (including sided product superoperators)
state()	Returns user specified density matrices (Hilbert space) or state vectors (Liouville space)
average()	Performs average Hamiltonian theory treatment (Waugh, Krylov-Bogolyubov, matrix log)
evolution()	Runs all types of spin system evolution and detection, uses reduced state spaces under the bonnet.

Et cetera… the whole of magnetic resonance is covered.

Basis set selection

The bigger, the better – same variational principle as quantum chemistry.

```
% Basis set
bas.formalism='sphten-liouv';
bas.approximation='IK-1';
bas.connectivity='scalar_couplings';
bas.level=4; bas.space_level=3;
```

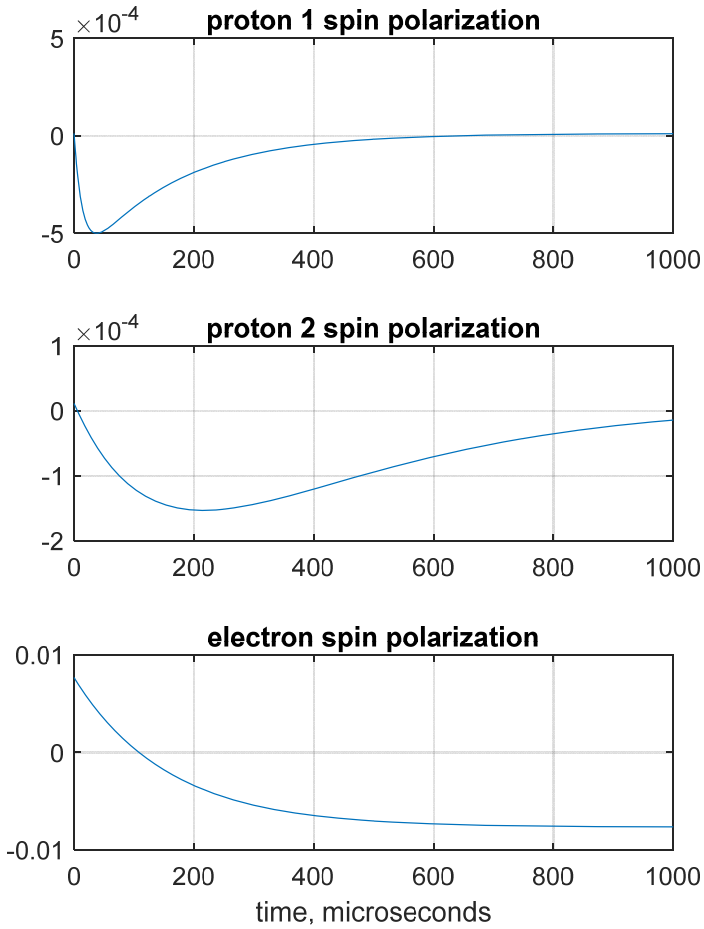
```
% Chemical kinetics
inter.chem.parts={[1 2],[3 4]};
inter.chem.rates=[-5e2  2e3;
                   5e2 -2e3];
```

Basis set	Description
IK-0(n)	All spin correlations up to, and including, order n, irrespective of proximity on J-coupling or dipolar coupling graphs. Generated with a combinatorial procedure.
IK-1(n,k)	All spin correlations up to order n between directly J-coupled spins (with couplings above a user-specified threshold) and up to order k between spatially proximate spins (with distances below the user-specified threshold). Generated by coupling graph analysis.
IK-2(n)	For each spin, all of its correlations with directly J-coupled spins, and correlations up to order n with spatially proximate spins (below the user-specified distance threshold). Generated by coupling graph analysis.

N.B. Many further optimisations available – see the basis preparation section of the manual.

Manual step-by-step simulation

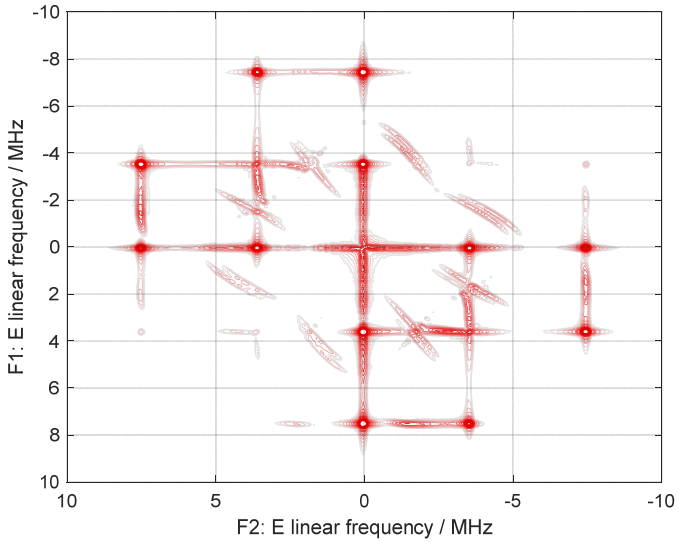
```
% Spinach housekeeping
spin_system=create(sys,inter);
spin_system=basis(spin_system,bas);
spin_system=assume(spin_system,'esr');

% Initial state - thermal equilibrium
rho=equilibrium(spin_system);

% Detection state - Lz on all spins
coil=[state(spin_system,{'Lz'},{1})...
      state(spin_system,{'Lz'},{2})...
      state(spin_system,{'Lz'},{3})];

% Hamiltonian and relaxation superoperators
H=hamiltonian(spin_system);
R=relaxation(spin_system);

% Electron control operator
Lp=operator(spin_system,'L+','E'); Lx=(Lp+Lp')/2;

% Pi pulse on electron
rho=step(spin_system,Lx,rho,pi);

% Evolution (10000 steps, 0.1 microseconds each)
answer=evolution(spin_system,H+1i*R,coil,rho,1e-7,10000,'multichannel');
```



N.B.: most *Spinach* functions are parallelised – it will use as many cores as it finds.

Manual step-by-step simulation

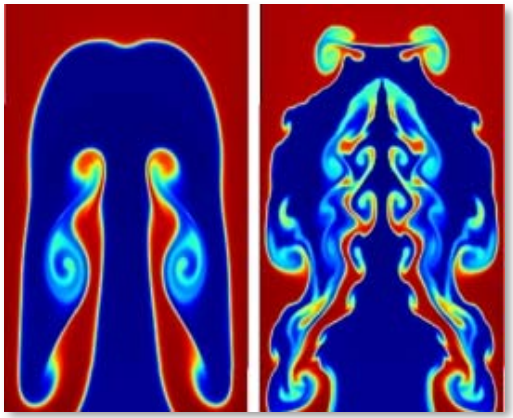
```
[…]

% Apply the first pulse
rho=step(spin_system,Lx,parameters.rho0,pi/2);

% Run the tau evolution
rho=evolution(spin_system,L,[],rho,parameters.tau,1,'final');

% Apply the second pulse
rho=step(spin_system,Lx,rho,pi/2);

% Apply coherence filter
rho=coherence(spin_system,rho,{{'E',0}});

% Run the indirect dimension evolution
rho_stack=evolution(spin_system,L,[],rho,1/parameters.sweep,...
                    parameters.nsteps(1)-1,'trajectory');

% Apply the third pulse
rho_stack=step(spin_system,Lx,rho_stack,pi);

% Propagate coil state backwards in time
coil=evolution(spin_system,L,[],parameters.coil,-parameters.tau,1,'final');

% Apply a backwards pulse on the coil
coil=step(spin_system,-Lx,coil,pi/2);

% Detect on new coil state in the direct dimension
fid=evolution(spin_system,L,coil,rho_stack,1/parameters.sweep,...
              parameters.nsteps(2)-1,'observable');

[…]
```



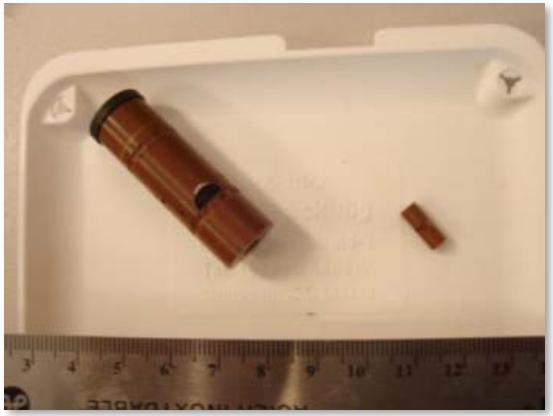
In simulations, you can be rather liberal with the direction of time!

Operators, states, elementary operations, and powder grids are all handled by Spinach.

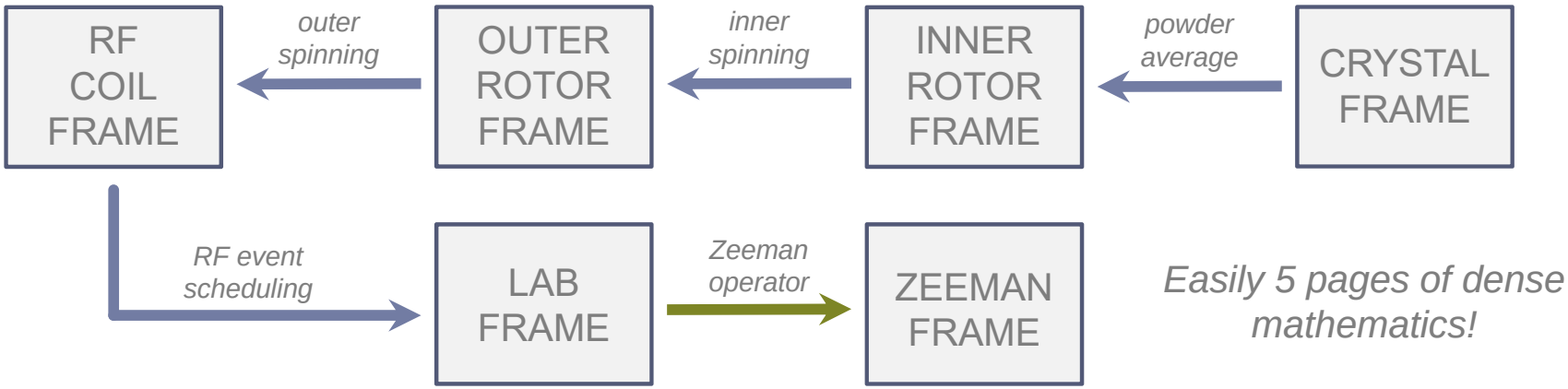
The need for context functions



Rayleigh-Taylor instability in fluid flow (simulated with HERACLES package)



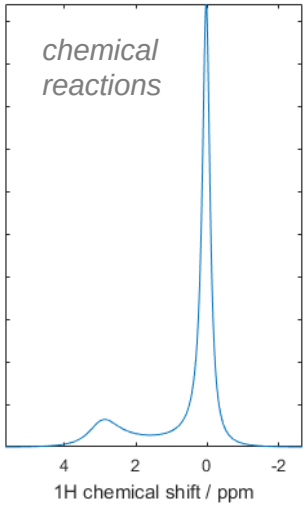
DOR probe built by the group of David Bryce (Ottawa University). Typical spinning rate is 1.5 kHz for the outer rotor and 7 kHz for the inner rotor.

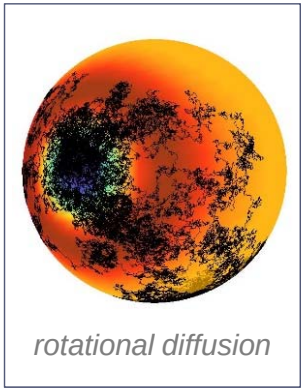


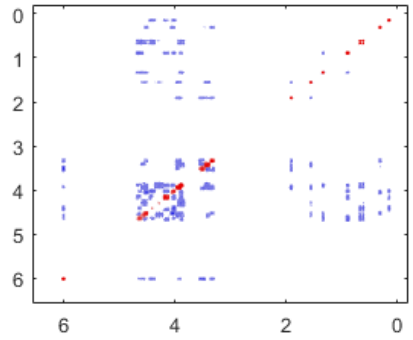
Easily 5 pages of dense mathematics!

See the review by Paul Hodgkinson (PNMRS, 36(3), 2000, 201-239) for further information.

The need for context functions







spin diffusion and cross-relaxation (can be fully incoherent or effectively incoherent)

CORRELATION FUNCTIONS

HAMILTONIAN EXPANSION

RELAXATION THEORY

KINETICS SUPEROP.

GROUND STATE

SYSTEM DYNAMICS

Time to implement from scratch and debug:

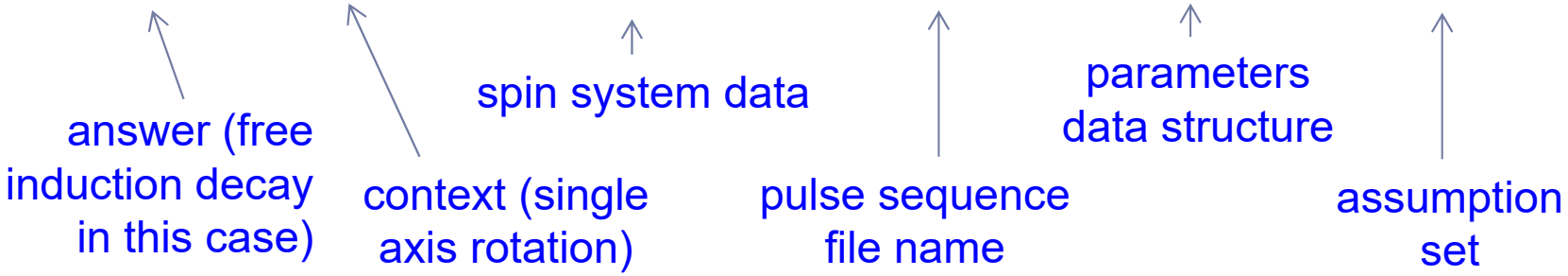
can easily be years

See the review by Maurice Goldman (JMR, 149 (2001) 160-187) for further information.

Spinach context functions

A *context* is a function that sits between Spinach kernel (that only knows mathematics) and the experiment (that only knows physics):

```
fid=singlerot(spin_system,@crosspol,parameters,'nmr');
```



The data supplied in the *parameters* structure is used by the context function and also passed on to the pulse sequence.

See the manual pages for each pulse sequence to see which parameters the sequence needs.

```
% MAS parameters
parameters.rate=10000;
parameters.axis=[1 1 1];
parameters.max_rank=3;
parameters.irr_powers=[5e4 4e4];
parameters.spins={'1H','15N'};
parameters.irr_opers={Hy Nx};
parameters.exc_opers={Hx Ny};
parameters.coil=state(spin_system,'L+','15N');
parameters.nsteps=100;
parameters.timestep=1e-5;
parameters.grid='repulsion_ab_100_sph';
parameters.verbose=1;
```

- Spinach contexts are *liquid, powder, crystal, roadmap, singlerot, doublerot* and *imaging.*

Pulse sequence programming

```
function fid=noesy(spin_system,parameters,H,R,K)

% Compose Liouvillian
L=H+1i*R+1i*K;

% Coherent evolution timestep
timestep=1/parameters.sweep;

% Detection state
coil=state(spin_system,'L+',parameters.spins{1});

% Pulse operators
Lp=operator(spin_system,'L+',parameters.spins{1});
Lx=(Lp+Lp')/2; Ly=(Lp-Lp')/2i;

% First pulse
rho=step(spin_system,Lx,parameters.rho0,pi/2);

% F1 evolution
rho_stack=evolution(spin_system,L,[],rho,timestep,...
                    parameters.npoints(1)-1,'trajectory');

% Second pulse
rho_stack_cos=step(spin_system,Lx,rho_stack,pi/2);
rho_stack_sin=step(spin_system,Ly,rho_stack,pi/2);
```

Hamiltonian, relaxation, and kinetics operators are received from the context function.

The context takes care of powder averages, anisotropic relaxation, equilibrium, *etc.*

Rotating frames, offsets, RDC, and anything that the actual sequence should not worry about, is handled there.

- Infrastructure functions are available for all common pulse sequence events.

Pulse sequence programming

```
% Homospoil
rho_stack_cos=homospoil(spin_system,rho_stack_cos,'destroy');
rho_stack_sin=homospoil(spin_system,rho_stack_sin,'destroy');

% Mixing time
rho_stack_cos=evolution(spin_system,1i*R+1i*K,[],...
                        rho_stack_cos,parameters.tmix,1,'final');
rho_stack_sin=evolution(spin_system,1i*R+1i*K,[],...
                        rho_stack_sin,parameters.tmix,1,'final');

% Homospoil
rho_stack_cos=homospoil(spin_system,rho_stack_cos,'destroy');
rho_stack_sin=homospoil(spin_system,rho_stack_sin,'destroy');

% Third pulse
rho_stack_cos=step(spin_system,Ly,rho_stack_cos,pi/2);
rho_stack_sin=step(spin_system,Ly,rho_stack_sin,pi/2);

% F2 evolution
fid.cos=evolution(spin_system,L,coil,rho_stack_cos,timestep,...
                  parameters.npoints(2)-1,'observable');
fid.sin=evolution(spin_system,L,coil,rho_stack_sin,timestep,...
                  parameters.npoints(2)-1,'observable');

end
```

N.B. Coherence selection in *Spinach* is analytical – no need to model phase cycles explicitly.

Ubiquitin NOESY simulation

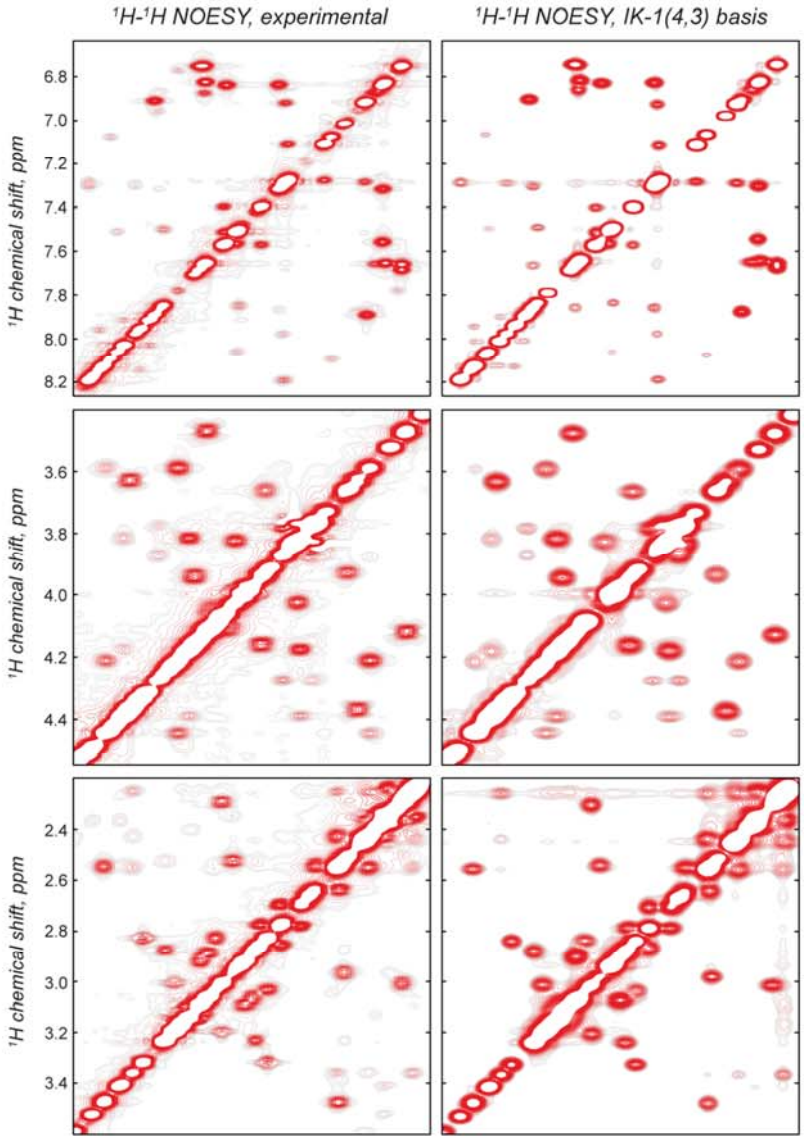
```
% Protein data
[sys,inter]=protein('1D3Z.pdb','1D3Z.bmrb');

% Magnet field
sys.magnet=21.1356;

% Tolerances
sys.tols.prox_cutoff=4.0;
sys.tols.inter_cutoff=2.0;

% Relaxation theory
inter.relaxation='redfield';
inter.rlx_keep='kite';
inter.tau_c={5e-9};

% Basis set
bas.formalism='sphten-liouv';
bas.approximation='IK-1';
bas.connectivity='scalar_couplings';
bas.level=4; bas.space_level=3;
```

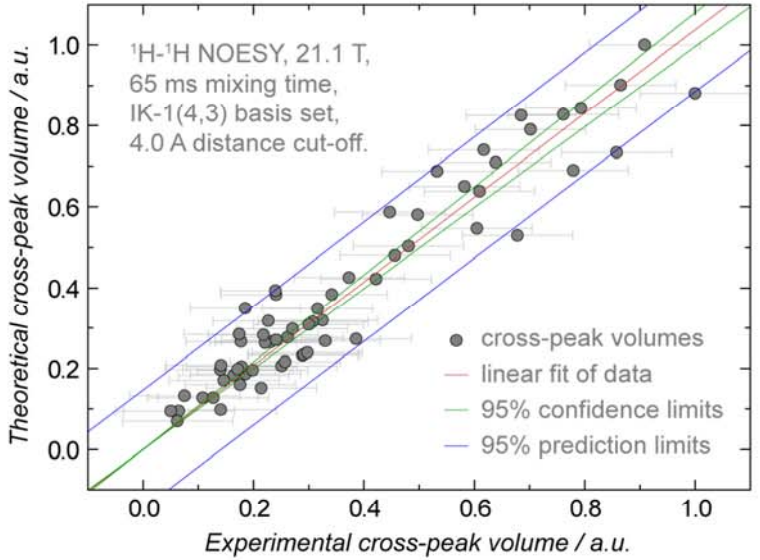
```
% Create the spin system structure
spin_system=create(sys,inter);

% Build the basis
spin_system=basis(spin_system,bas);

% Sequence parameters
parameters.tmix=0.065;
parameters.offset=4250;
parameters.sweep=10750;
parameters.npoints=[512 512];
parameters.zerofill=[2048 2048];
parameters.spins={'1H'};
parameters.axis_units='ppm';

% Simulation
fid=liquid(spin_system,@noesy,parameters,'nmr');
```

L.J. Edwards, D.V. Savostyanov, Z.T. Welderufael, D. Lee, I. Kuprov, *"Quantum mechanical NMR simulation algorithm for protein-size spin systems"*, Journal of Magnetic Resonance, **2014**, 243, 107-113.

N.B. In most cases you would be parsing some data of your own in the second line.

NOESY of ubiquitin





The anisotropic Hamiltonian has between 60,000 and 200,000 terms, depending on the distance cut-off radius. There are about 2,000 *J*-couplings and a few hundred CSA tensors.

Such calculations are supposed to be impossible: matrix dimension goes as 2^N with the number of spins!

Experimental data from Donghan Lee