```
% Performs rotations of Cartesian representations of spin interaction
% tensors by the user-supplied Euler angles.
%
% i.kuprov@soton.ac.uk

function inter_tensor=rotate_interaction(inter_tensor,alpha,beta,gamma)

% Gamma rotation in XY plane
R_gamma=[cos(gamma) -sin(gamma) 0;
         sin(gamma)  cos(gamma) 0;
         0           0          1];

% Beta rotation in YZ plane
R_beta=[1      0           0;
        0      cos(beta)  -sin(beta);
        0      sin(beta)   cos(beta)];

% Alpha rotation in XY plane
R_alpha=[cos(alpha) -sin(alpha) 0;
         sin(alpha)  cos(alpha) 0;
         0           0          1];

% The total rotation
R=R_alpha*R_beta*R_gamma;

% Apply rotation to the interaction
inter_tensor=R*inter_tensor*R';

end
```

```
% Basic 14N axial quadrupolar powder pattern.
%
% i.kuprov@soton.ac.uk

function hello_world()

% Pauli matrices for spin-1 particle
sigma_x=[0  1  0; 1  0  1; 0  1  0]/sqrt(2);
sigma_y=[0  -1i  0; 1i  0  -1i; 0  1i  0]/sqrt(2);
sigma_z=[1 0 0; 0 0 0; 0 0 -1];

% Isotropic Zeeman interaction
omega_zeeman=2*pi*1000;

% Quadrupolar tensor (must be axial - phi averaging not done)
quad_tensor=2*pi*[-5e2  0    0
                  0   -5e2  0
                  0    0    1e3];

% Isotropic Hamiltonian
H=omega_zeeman*sigma_z;

% Timing and detection parameters
time_step=1e-4; coil=sigma_x+1i*sigma_y;

% Preallocate the fid
fid=zeros(1,500);

% Loop over theta only
for theta=linspace(0,pi,500);

    % Compute grid weight
    weight=sin(theta)/500;

    % Rotate quadrupolar tensor into current orientation
    current_quad=rotate_interaction(quad_tensor,0,theta,0);

    % Create current orientation Hamiltonian
    H_current=H+(current_quad(3,3)-0.5*(current_quad(1,1)+current_quad(2,2)))*...
                (2/3)*(sigma_z^2-0.5*(sigma_x^2+sigma_y^2));

    % Initial condition
    rho=sigma_x;

    % Compute the propagator
    P=expm(-1i*H_current*time_step);

    % Run time evolution
    for n=1:500
        fid(n)=fid(n)+weight*trace(coil'*rho);
        rho=P'*rho*P;
    end

end
```

```
% Apodization
fid=fid.*exp(-10*linspace(0,1,numel(fid)));

% Plotting
plot(real(fftshift(fft(fid))));

end
```

```
% Basic nitroxide radical EPR powder pattern.
%
% i.kuprov@soton.ac.uk

function hello_again_world()

% Pauli matrices for spin-1 particle
sigma2_x=[0  1  0; 1  0  1;  0  1  0]/sqrt(2);
sigma2_y=[0 -1i 0; 1i 0 -1i; 0  1i 0]/sqrt(2);
sigma2_z=[1 0 0; 0 0 0; 0 0 -1];

% Pauli matrices for spin-1/2 particle
sigma1_x=[0 1/2; 1/2  0];
sigma1_y=[0 -1i/2; 1i/2  0];
sigma1_z=[1/2 0; 0 -1/2];

% Interactions (must be axial and collinear - phi integration not done)
zeeman_tensor=2*pi*[-1e6 0      0
                     0    -1e6  0
                     0     0    2e6];
hfc_tensor=   2*pi*[-2e6   0     0
                     0    -2e6   0
                     0     0     4e6] + 2*pi*eye(3)*1e6;

% Isotropic Hamiltonian
H=(trace(hfc_tensor)/3)*(kron(sigma1_x,sigma2_x)+...
                         kron(sigma1_y,sigma2_y)+...
                         kron(sigma1_z,sigma2_z));

% Destroy the isotropic part of the hfc tensor (no longer needed)
hfc_tensor=hfc_tensor-eye(3)*trace(hfc_tensor)/3;

% Timing and detection parameters
time_step=6e-8;
coil=kron(sigma1_x,eye(3))+1i*kron(sigma1_y,eye(3));

% Preallocate the fid
fid=zeros(1,500);

% Loop over theta
for theta=linspace(0,pi,500);

    % Compute grid weight
    weight=sin(theta)/500;

    % Rotate interaction tensors into current orientation
    current_hfc=rotate_interaction(hfc_tensor,0,theta,0);
    current_zeeman=rotate_interaction(zeeman_tensor,0,theta,0);

    % Create current orientation Hamiltonian
    H_current=H+current_zeeman(3,3)*kron(sigma1_z,eye(3))+...
                (2/3)*(current_hfc(3,3)-0.5*(current_hfc(1,1)+current_hfc(2,2)))*...
                (kron(sigma1_z,sigma2_z)-0.5*(kron(sigma1_x,sigma2_x)+kron(sigma1_y,↙
sigma2_y)));
```

```
    % Initial condition
    rho=kron(sigma1_x,eye(3));

    % Compute the propagator
    P=expm(-1i*H_current*time_step);

    % Run time evolution
    for n=1:500
        fid(n)=fid(n)+weight*trace(coil'*rho);
        rho=P'*rho*P;
    end

end

% Apodization
fid=fid.*exp(-10*linspace(0,1,numel(fid)));

% Plotting
plot(real(fftshift(fft(fid))));

end
```