```
% Exp010_OnePulse.m
% Examples of input parameters for PO.pulse().

clear
close all


rho = PO(1,{'Iz'});% Initial State
rho.dispPOtxt();

% All four cases below are equivalent.
pulse_writing = 'exp4';
switch pulse_writing
    case 'exp1'
        rho = rho.pulse('I','x',1/2*pi);% I90x-pulse
    case 'exp2'
        rho = rho.pulse(1,'x',1/2*pi);  % I90x-pulse
    case 'exp3'
        rho = rho.pulse('I',0,1/2*pi);  % I90x-pulse
    case 'exp4'
        rho = rho.pulse(1,0,1/2*pi);    % I90x-pulse
end
```

```
    Iz
Pulse: I  90x
     - Iy
```

Published with MATLAB® R2020b

```
% Exp015_OnePulse_pmz.m
% One pulse experiment in the lowering/raising operator basis.

clear
close all


rho = PO(2,{'Iz'});% Initial State
rho = xyz2pmz(rho);% Since the default basis of Iz is xyz, it is necessary to convert the basis
to pmz.
rho.dispPOtxt();
rho = rho.pulse('I','y',1/2*pi);% I90x-pulse
```

```
    Iz
Pulse: I  90y
    Ip*1/2 + Im*1/2
```

Published with MATLAB® R2020b

```
% Exp020_CSevolution.m
% Example of chemical shift evolution

clear
close all

% Symbolic constant
syms q

rho = PO(1,{'Iz'});% Initial State
rho.dispPOtxt();
rho = rho.pulse('I','y',1/2*pi);% I90y-pulse
rho = rho.cs('I',q);% CS evolution
```

```
    Iz
Pulse: I  90y
    Ix
CS: I q
    Ix*cos(q) + Iy*sin(q)
```

Published with MATLAB® R2020b

```
% Exp030_JCevolution.m
% Example of J-coupling evolution

clear
close all

PO.create({'I' 'S'});% Preparation of PO objects and symbolic constants
rho = Ix;% Initial State
rho.dispPOtxt();
rho = rho.jc('IS',pi*J12*t);% J-coupling eovlution
```

```
    Ix
JC: IS J12*t*pi
    Ix*cos(J12*t*pi) + 2IySz*sin(J12*t*pi)
```

Published with MATLAB® R2020b

```
% Exp035_FreeEvolution.m
% Comparison of the calculation speeds between UrhoUinv_mt() and UrhoUinv_M()

clear
close all

% spin_label_cell = {'I' 'S'};% Case of two spins
spin_label_cell = {'I' 'S' 'K'};% Case of three spins
PO.create(spin_label_cell);

rho = Ix;
rho.disp = 0;
fprintf(1,'Evolution of Ix under chemical shift and J-coupling\n')
fprintf(1,'Number of Spins: %d\n',length(spin_label_cell));

tic;
obj1 = rho.cs('I',o1*t).jc('IS',pi*JIS*t);% UrhoUinv_mt() is called
et1 = toc;
fprintf(1,'UrhoUinv_mt(): %g s\n',et1);
dispPOtxt(obj1);

H = o1*Iz + pi*JIS*2*Iz*Sz;
tic;
obj2 = UrhoUinv(rho,H*t,1);% UrhoUinv_M() is called
et2 = toc;
fprintf(1,'UrhoUinv_M(): %g s\n',et2);
dispPOtxt(obj2);

% Rewrite obj2.coef
coef_new = simplify(rewrite(obj2.coef,'sincos'));
obj3 = set_coef(obj2,coef_new);
fprintf(1,'Rewrite coef from UrhoUinv_M()\n')
dispPOtxt(obj3);
```

```
Evolution of Ix under chemical shift and J-coupling
Number of Spins: 3
UrhoUinv_mt(): 0.243712 s
     - 2IxSz*sin(o1*t)*sin(JIS*t*pi) + Ix*cos(o1*t)*cos(JIS*t*pi) + 2IySz*cos(o1*t)*sin(JIS*t*pi)
+ Iy*sin(o1*t)*cos(JIS*t*pi)
UrhoUinv_M(): 1.7135 s
    2IxSz*((exp(-o1*t*1i)*exp(-JIS*t*pi*1i))/4 - (exp(-o1*t*1i)*exp(JIS*t*pi*1i))/4 -
(exp(o1*t*1i)*exp(-JIS*t*pi*1i))/4 + (exp(o1*t*1i)*exp(JIS*t*pi*1i))/4) + Ix*((exp(-
o1*t*1i)*exp(-JIS*t*pi*1i))/4 + (exp(-o1*t*1i)*exp(JIS*t*pi*1i))/4 + (exp(o1*t*1i)*exp(-
JIS*t*pi*1i))/4 + (exp(o1*t*1i)*exp(JIS*t*pi*1i))/4) + 2IySz*((exp(-o1*t*1i)*exp(-
JIS*t*pi*1i)*1i)/4 - (exp(-o1*t*1i)*exp(JIS*t*pi*1i)*1i)/4 + (exp(o1*t*1i)*exp(-
JIS*t*pi*1i)*1i)/4 - (exp(o1*t*1i)*exp(JIS*t*pi*1i)*1i)/4) + Iy*((exp(-o1*t*1i)*exp(-
JIS*t*pi*1i)*1i)/4 + (exp(-o1*t*1i)*exp(JIS*t*pi*1i)*1i)/4 - (exp(o1*t*1i)*exp(-
JIS*t*pi*1i)*1i)/4 - (exp(o1*t*1i)*exp(JIS*t*pi*1i)*1i)/4)
Rewrite coef from UrhoUinv_M()
     - 2IxSz*sin(o1*t)*sin(JIS*t*pi) + Ix*cos(o1*t)*cos(JIS*t*pi) + 2IySz*cos(o1*t)*sin(JIS*t*pi)
+ Iy*sin(o1*t)*cos(JIS*t*pi)
```

Published with MATLAB® R2020b

```
% Exp036_Pulse_PhaseShift.m
% Comparison of the calculation speeds between UrhoUinv_mt() and UrhoUinv_M()

clear
close all

% spin_label_cell = {'I'};
spin_label_cell = {'I' 'S'};

PO.create(spin_label_cell);
fprintf(1,'Evolution of Iz under a pulse with flip angle q and phase f\n')
fprintf(1,'Number of Spins: %d \n',length(spin_label_cell));

rho = Iz;
rho.disp = 0;

tic;
obj1 = rho.pulse_phshift('I',f,q);% UrhoUinv_mt() is called
et1 = toc;
fprintf(1,'UrhoUinv_mt(): %g s\n',et1);
dispPOtxt(obj1);

H = q*(Ix*cos(f) + Iy*sin(f));
tic;
obj2 = UrhoUinv(rho,H,1);% UrhoUinv_M() is called
et2 = toc;
fprintf(1,'UrhoUinv_M(): %g s\n',et2);
dispPOtxt(obj2);
```

```
Evolution of Iz under a pulse with flip angle q and phase f
Number of Spins: 2
UrhoUinv_mt(): 0.28625 s
    Ix*sin(f)*sin(q) - Iy*cos(f)*sin(q) + Iz*cos(q)
UrhoUinv_M(): 0.797315 s
    Ix*sin(f)*sin(q) - Iy*cos(f)*sin(q) + Iz*cos(q)
```

Published with MATLAB® R2020b

```
% Exp040_JCrefocusing.m
% Keeler, J., Understanding NMR Spectroscopy (1st Ed.), Wiley, 2005.
% pp. 168, Fig. 7.14
% I: t/2-   -t/2 => cs is not refocused
% S: t/2-180-t/2 => cs is refocused
%                   jc is refocused

clear
close all

PO.create({'I' 'S'});
rho = Ix + Sx;

% If the constructor PO() is used
% syms J12 t oI oS
% rho = PO(2,{'Ix' 'Sx'});% Initial State

rho.dispPOtxt();
rho = rho.cs('I',oI*t/2);
rho = rho.cs('S',oS*t/2);
rho = rho.jc('IS',pi*J12*t/2);

rho = rho.pulse('S','x',pi);% Refocusing pulse on S
% What if refocusing pulse is also applied to I.
% rho = rho.pulse('I','x',pi);% Refocusing pulse on I

rho = rho.cs('I',oI*t/2);
rho = rho.cs('S',oS*t/2);
rho = rho.jc('IS',pi*J12*t/2);
```

```
    Ix + Sx
CS: I (oI*t)/2
    Ix*cos((oI*t)/2) + Iy*sin((oI*t)/2) + Sx
CS: S (oS*t)/2
    Ix*cos((oI*t)/2) + Iy*sin((oI*t)/2) + Sx*cos((oS*t)/2) + Sy*sin((oS*t)/2)
JC: IS (J12*t*pi)/2
     - 2IxSz*sin((oI*t)/2)*sin((J12*t*pi)/2) + Ix*cos((oI*t)/2)*cos((J12*t*pi)/2) +
2IySz*cos((oI*t)/2)*sin((J12*t*pi)/2) + Iy*sin((oI*t)/2)*cos((J12*t*pi)/2) -
2IzSx*sin((oS*t)/2)*sin((J12*t*pi)/2) + 2IzSy*cos((oS*t)/2)*sin((J12*t*pi)/2) +
Sx*cos((oS*t)/2)*cos((J12*t*pi)/2) + Sy*sin((oS*t)/2)*cos((J12*t*pi)/2)
Pulse: S 180x
    2IxSz*sin((oI*t)/2)*sin((J12*t*pi)/2) + Ix*cos((oI*t)/2)*cos((J12*t*pi)/2) -
2IySz*cos((oI*t)/2)*sin((J12*t*pi)/2) + Iy*sin((oI*t)/2)*cos((J12*t*pi)/2) -
2IzSx*sin((oS*t)/2)*sin((J12*t*pi)/2) - 2IzSy*cos((oS*t)/2)*sin((J12*t*pi)/2) +
Sx*cos((oS*t)/2)*cos((J12*t*pi)/2) - Sy*sin((oS*t)/2)*cos((J12*t*pi)/2)
CS: I (oI*t)/2
    2IxSz*2*cos((oI*t)/2)*sin((oI*t)/2)*sin((J12*t*pi)/2) + Ix*cos(oI*t)*cos((J12*t*pi)/2) -
2IySz*cos(oI*t)*sin((J12*t*pi)/2) + Iy*2*cos((oI*t)/2)*sin((oI*t)/2)*cos((J12*t*pi)/2) -
2IzSx*sin((oS*t)/2)*sin((J12*t*pi)/2) - 2IzSy*cos((oS*t)/2)*sin((J12*t*pi)/2) +
Sx*cos((oS*t)/2)*cos((J12*t*pi)/2) - Sy*sin((oS*t)/2)*cos((J12*t*pi)/2)
CS: S (oS*t)/2
    2IxSz*2*cos((oI*t)/2)*sin((oI*t)/2)*sin((J12*t*pi)/2) + Ix*cos(oI*t)*cos((J12*t*pi)/2) -
```

```
2IySz*cos(oI*t)*sin((J12*t*pi)/2) + Iy*2*cos((oI*t)/2)*sin((oI*t)/2)*cos((J12*t*pi)/2) -
2IzSy*sin((J12*t*pi)/2) + Sx*cos((J12*t*pi)/2)
JC: IS (J12*t*pi)/2
    Ix*cos(oI*t) + Iy*sin(oI*t) + Sx
```

Published with MATLAB® R2020b

```
% Exp050_OnePulse_PhaseCycling.m
% Example of writing a pulse sequence with phase cycling

clear
close all

% Phase tables
phid = 1:4;
ph1tab = [1,2,3,0];     % Phase for 90-pulse
phRtab = [0,1,2,3];     % Receiver phase

rho_ini = PO(1,{'Iz'});% Initial State

% Initialization
a0_M = [];
rho_M = [];
rho_total = 0;

% Pulse sequence with phase cycling
for ii = phid
    fprintf(1,'\nii: %2d\n',ii);
    ph1 = PO.phmod(ph1tab,ii);
    phR = PO.phmod(phRtab,ii);

    rho = rho_ini;
    rho.dispPOtxt();% Display Initial state
    rho = rho.pulse(1,ph1,pi/2);% 90-pulse

    rho_detect = receiver(rho,phR);
    rho_total = rho_detect + rho_total;

    [a0_V,rho_V] = rho.SigAmp({'I'},phR);% Detection
    a0_M = cat(1,a0_M,a0_V);
    rho_M = cat(1,rho_M,rho_V);
end
rho_final = observable(rho_total,{'I'});
```

Published with MATLAB® R2020b

```
% Exp050_OnePulse_PhaseCycling_PS.m
% Example of writing a pulse sequence with phase cycling

% Para begin %
phid = 1:4;
ph_cell{1} = [1,2,3,0];     % Phase for 90-pulse
phRtab = [0,1,2,3];         % Receiver phase
spin_label_cell = {'I'};
coef_cell = {};             % Special sym coefs
rho_ini = Iz;
obs_cell = {'I'};
% Para end %

% PS begin %
rho = rho.pulse(1,ph1,pi/2);% 90-pulse
% PS end %
```

Published with MATLAB® R2020b

```
% Exp060_SpinEcho.m
% Spin-echo (Hahn-echo) experiment with phase cycling.
% Effect of the miscalibration of 180 pulse can be checked.

clear
close all

% Phase tables
phid = 1:16;
ph1tab = [1,2,3,0];                         % Phase for 90-pulse
ph2tab = [0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3]; % Phase for 180-pulse
phRtab = [0,3,2,1,2,1,0,3];                 % Receiver phase

% Symbolic constants
syms t oI d

% Initial State
rho_ini = PO(1,{'Iz'});% Initial State

% Initialization
a0_M = [];
rho_M = [];
rho_total = 0;

% Pulse sequence with phase cycling
for ii = phid
    fprintf(1,'\nii: %2d\n',ii);
    ph1 = PO.phmod(ph1tab,ii);
    ph2 = PO.phmod(ph2tab,ii);
    phR = PO.phmod(phRtab,ii);

    rho = rho_ini;
    rho.dispPOtxt();
    rho = rho.pulse(1,ph1,1/2*pi);% 90-pulse

    rho = rho.cs(1,oI*t);% Chemical shift evolution
    % rho = rho.pulse(1,ph2,pi);% 180-pulse
    rho = rho.pulse(1,ph2,pi+d);% 180+d-pulse, where d indicates the miscalibration of 180-pulse
    rho = rho.cs(1,oI*t);% Chemical shift evolution

    rho_detect = receiver(rho,phR);
    rho_total = rho_detect + rho_total;

    [a0_V,rho_V] = rho.SigAmp({'I'},phR);% Detection
    a0_M = cat(1,a0_M,a0_V);
    rho_M = cat(1,rho_M,rho_V);
end
rho_final = observable(rho_total,{'I'});
```

Published with MATLAB® R2020b

```
% Exp060_SpinEcho_PS.m
% Spin-echo (Hahn-echo) experiment with phase cycling.
% Effect of the miscalibration of 180 pulse can be checked.

% Para begin %
phid = 1:16;
ph_cell{1} = [1,2,3,0];                          % Phase for 90-pulse
ph_cell{2} = [0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3]; % Phase for 180-pulse
phRtab = [0,3,2,1,2,1,0,3];    % phR
spin_label_cell = {'I'};
coef_cell = {}; % Special sym coefs
rho_ini = Iz;
obs_cell = {'I'};
% Para end %

% PS begin %
rho = rho.pulse(1,ph1,1/2*pi);% 90-pulse
rho = rho.cs(1,oI*t);% Chemical shift evolution
% rho = rho.pulse(1,ph2,pi);% 180-pulse
rho = rho.pulse(1,ph2,pi+d);% 180+d-pulse, where d indicates the miscalibration of 180-pulse
rho = rho.cs(1,oI*t);% Chemical shift evolution
% PS end %
```

Published with MATLAB® R2020b

```
% Exp080_refocusedINEPT_InS.m
% Intensity calculation of refocused INEPT in InS system (n = 1,2 or 3)
% Levitt, M. H., Spin Dynamics (2nd Ed.), pp. 440 - 442, pp.488 - 491.

clear
close all

InS = 'I3S';
switch InS
    case 'IS'
        % IS system
        PO.create({'I1' 'S2'})
        rho = I1z*B + S2z;
        jc_cell = {'I1S2'};

    case 'I2S'
        % I2S system
        PO.create({'I1' 'I2' 'S3'});
        rho = I1z*B + I2z*B + S3z;
        jc_cell = {'I1S3' 'I2S3'};

    case 'I3S'
        % I3S system
        PO.create({'I1' 'I2' 'I3' 'S4'});
        rho = I1z*B + I2z*B + I3z*B + S4z;
        jc_cell = {'I1S4' 'I2S4' 'I3S4'};

end
q1 = 1/2*pi;
q1_cell = PO.v2cell(q1,jc_cell);

q2 = pi*J*t;
q2_cell = PO.v2cell(q2,jc_cell);

dispPOtxt(rho);
rho = simpulse(rho,{'I*' 'S*'},{'x' 'x'},{3/2*pi pi});
rho = simjc(rho,jc_cell,q1_cell);

rho = simpulse(rho,{'I*' 'S*'},{'y' 'y'},{1/2*pi 1/2*pi});
rho = simpulse(rho,{'I*' 'S*'},{'x' 'x'},{pi pi});
rho = simjc(rho,jc_cell,q2_cell);

rho_detect = receiver(rho,'x');
rho_final = observable(rho_detect,{'S*'});
dispPO(rho_final);
[a0_V,rho_V] = rho.SigAmp({'S*'},'x');
```

Published with MATLAB® R2020b

```
% Exp090_refocusedINEPT_PhaseCycling.m
% refocused INEPT I => S
% Example to check phase cycling.
% Keeler, J., Understanding NMR Spectroscopy (1st Ed.), Wiley, 2005.
% pp. 174 - 175.

clear
close all

% % 2-steps
% phid = 1:2;
% ph1tab = [0,2];% I 90
% ph2tab = [0];  % S INEPT 1st 180
% ph3tab = [0];  % I INEPT 1st 180
% ph4tab = [0];  % S INEPT 2nd 90
% ph5tab = [1];  % I INEPT 2nd 90
% ph6tab = [0];  % S INEPT 3rd 180
% ph7tab = [0];  % I INEPT 3rd 180
% phRtab = [0,2];% Receiver

% 16-steps
phid = 1:16;
ph1tab = [0,0,0,0,0,0,0,0,2,2,2,2,2,2,2,2];% I 90
ph2tab = [0,2,0,2];                       % S INEPT 1st 180
ph3tab = [0,2,0,2];                       % I INEPT 1st 180
ph4tab = [0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3];% S INEPT 2nd 90
ph5tab = [1,1,3,3];                       % I INEPT 2nd 90
ph6tab = [0,2,0,2,1,3,1,3];               % S INEPT 3rd 180
ph7tab = [0,2,0,2];                       % I INEPT 3rd 180
phRtab = [0,0,2,2,1,1,3,3];               % Receiver

%
syms B J t1 t2

% Initial State
rho_ini = PO(2,{'Iz' 'Sz'},{B 1});


% IS system
a0_M = [];
rho_M = [];
rho_total = 0;
for ii = phid
    fprintf(1,'\nii: %2d\n',ii);
    ph1 = PO.phmod(ph1tab,ii);
    ph2 = PO.phmod(ph2tab,ii);
    ph3 = PO.phmod(ph3tab,ii);
    ph4 = PO.phmod(ph4tab,ii);
    ph5 = PO.phmod(ph5tab,ii);
    ph6 = PO.phmod(ph6tab,ii);
    ph7 = PO.phmod(ph7tab,ii);
    phR = PO.phmod(phRtab,ii);
```

```
    % OOP dot-style, CS ommitted, Pulse positions moved.
    rho = rho_ini;                                          % Preparation of the initial rho
    rho.dispPOtxt();
    rho = rho.pulse('I',ph1,1/2*pi);                        % I 90 pulse
    rho = rho.simpulse({'I' 'S'},{ph3 ph2},{pi pi});        % I,S 180 pulses
    rho = rho.jc('IS',pi*J*2*t1);                           % J-coupling evolution
    rho = rho.simpulse({'I' 'S'},{ph5 ph4},{1/2*pi 1/2*pi});% I,S 90 pulses
    rho = rho.simpulse({'I' 'S'},{ph7 ph6},{pi pi});        % I,S 180 pulses
    rho = rho.jc('IS',pi*J*2*t2);                           % J-coupling evolution

    rho_detect = receiver(rho,phR);
    rho_total = rho_detect + rho_total;

    [a0_V, rho_V] = rho.SigAmp({'S'},phR); % Detection
    a0_M = cat(1,a0_M,a0_V);
    rho_M = cat(1,rho_M,rho_V);
end
rho_final = observable(rho_total,{'S'});
```

Published with MATLAB® R2020b

```
% Exp090_refocusedINEPT_PhaseCycling_PS.m
% refocused INEPT I => S
% Example to check phase cycling.
% Keeler, J., Understanding NMR Spectroscopy (1st Ed.), Wiley, 2005.
% pp. 174 - 175.

% Para begin %
phid = 1:2;
ph_cell{1} = [0,2];% ph1
ph_cell{2} = [0];  % ph2
ph_cell{3} = [0];  % ph3
ph_cell{4} = [0];  % ph4
ph_cell{5} = [1];  % ph5
ph_cell{6} = [0];  % ph6
ph_cell{7} = [0];  % ph7
phRtab = [0,2];    % phR
spin_label_cell = {'I' 'S'};
coef_cell = {}; % Special sym coefs
rho_ini = Iz*B + Sz;
obs_cell = {'S'};
% Para end %

% PS begin %
rho = rho.pulse('I',ph1,1/2*pi);
rho = rho.simpulse({'I' 'S'},{ph3 ph2},{pi pi});
rho = rho.jc('IS',pi*J*2*t1);
rho = rho.simpulse({'I' 'S'},{ph5 ph4},{1/2*pi 1/2*pi});
rho = rho.simpulse({'I' 'S'},{ph7 ph6},{pi pi});
rho = rho.jc('IS',pi*J*2*t2);
% PS end %
```

Published with MATLAB® R2020b

```
% Exp100_INADEQUATE.m
% Levitt, M. H., Spin Dynamics(2nd Ed.), p.433.
% 2D-INADEQUATE using -45 deg phase shift

clear
close all

syms oI oS t
rho = PO(2,{'Iz' 'Sz'});
% rho = xyz2pmz(rho);% Check the result in the pmz basis.
rho.dispPOtxt();

States = 'sin';

switch States
    case 'cos'
        phi = 0;

   case 'sin'
        phi = -1/4*pi;

end

rho = rho.simpulse_phshift({'I' 'S'},{phi phi},{3/2*pi 3/2*pi});
rho = rho.jc('IS',pi/2);
rho = rho.simpulse_phshift({'I' 'S'},{phi phi},{1/2*pi 1/2*pi});
rho = rho.simcs({'I' 'S'},{oI*t oS*t});
rho = rho.simpulse({'I' 'S'},{'y' 'y'},{pi/2 pi/2});

phR = 0;
[a0_V,rho_V] = rho.SigAmp({'I' 'S'},phR);
```

Published with MATLAB® R2020b

```
% Exp100_INADEQUATE_PS.m
% Levitt, M. H., Spin Dynamics(2nd Ed.), p.433.
% 2D-INADEQUATE using -45 deg phase shift

% Para begin %
phid = 1:1;
phRtab = [0];
% spin_label_cell = {'I1' 'I2'};
% rho_ini = I1z + I2z;
spin_label_cell = {'I' 'S'};
rho_ini = Iz + Sz;
coef_cell = {}; % Special sym coefs
obs_cell = {1 2};
phi_vec = [0 -1/4*pi];
States = 'sin';
phi_id = [contains(States,'cos') contains(States,'sin')];% switch syntax
phi = phi_vec(phi_id ~= 0);% switch syntax
% Para end %

% PS begin %
rho = rho.simpulse_phshift({1 2},{phi phi},{3/2*pi 3/2*pi});
rho = rho.jc([1 2],pi/2);
rho = rho.simpulse_phshift({1 2},{phi phi},{1/2*pi 1/2*pi});
rho = rho.simcs({1 2},{o1*t o2*t});
rho = rho.simpulse({1 2},{'y' 'y'},{pi/2 pi/2});
% PS end %
```

Published with MATLAB® R2020b

```
% Exp110_3QF_COSY.m
% Guntert, P. et al., J. Magn. Reson. Ser. A, 101, 103-105, 1993.
% Guntert, P. Int. J. Quant. Chem., 106, 344-350, 2006.

clear
close all

phid = 1:6;
ph1tab = sym([0:5]*pi/3);% I 90
phRtab = [0 2];% Receiver

% Initial State
rho_ini = PO(3,{'I1z'},{1},{'I1' 'I2' 'I3'});
rho_ini.disp = 1;
PO.symcoef({'I1' 'I2' 'I3'})

a0_M = [];
rho_M = [];
rho_total = 0;

for ii = phid
    fprintf(1,'\nii: %2d\n',ii);
    ph1 = PO.phmod(ph1tab,ii);
    phR = PO.phmod(phRtab,ii);

    rho = rho_ini;
    rho.dispPOtxt();

    rho = rho.simpulse_phshift({'I*'},{ph1},{1/2*pi});
    rho = rho.simcs({'I*'},{o1*t1});
    rho = rho.simjc({'I1I2' 'I1I3'},{pi*J12*t1 pi*J13*t1});
    rho = rho.simpulse_phshift({'I*'},{ph1},{1/2*pi});
    rho = rho.simpulse({'I*'},{0},{1/2*pi});

    rho_detect = receiver(rho,phR);
    rho_total = rho_detect + rho_total;

    [a0_V, rho_V] = rho.SigAmp({'I*'},phR); % Detection
    a0_M = cat(1,a0_M,a0_V);
    rho_M = cat(1,rho_M,rho_V);
end
rho_final = observable(rho_total,{'I*'});
```

Published with MATLAB® R2020b

```
% Exp110_3QF_COSY_PS.m
% Guntert, P. et al., J. Magn. Reson. Ser. A, 101, 103-105, 1993.
% Guntert, P. Int. J. Quant. Chem., 106, 344-350, 2006.

% Para begin %
ph_cell{1} = sym([0:5]*pi/3);% I 90
phRtab = [0 2];% Receiver
spin_label_cell = {'I1' 'I2' 'I3'};
rho_ini = I1z;
% rho_ini = PO(length(spin_label_cell),{'I1z'},{1},spin_label_cell);
obs_cell = {'I*'};
phid = 1:6;
coef_cell = {}; % Special sym coefs
disp_bin = 1;
% Para end %

% PS begin %
rho = rho.simpulse_phshift({'I*'},{ph1},{1/2*pi});
rho = rho.simcs({'I*'},{o1*t1});
rho = rho.simjc({'I1I2' 'I1I3'},{pi*J12*t1 pi*J13*t1});
rho = rho.simpulse_phshift({'I*'},{ph1},{1/2*pi});
rho = rho.simpulse({'I*'},{0},{1/2*pi});
% PS end %
```

Published with MATLAB® R2020b

```
% Exp120HomoINEPT.m
% Homonuclear INEPT
% Movellan, T.K., ..., Andreas, L. B.
% J. Am. Chem. Soc. 2020, 142, 2704-2708.

clear
close all

% Homonuclear pulses thus the phases of simpulse() shoud be same
phid = 1:1;
ph1tab = [2 2 0 0];% Converted from (1H, 15N) phases for CP => 15N One pulse phase
ph2tab = [0*ones(1,8) 1*ones(1,8)];
ph3tab = [0*ones(1,16) 1*ones(1,16)];
ph4tab = [0 2];
ph5tab = [1*ones(1,4) 3*ones(1,4)];
phRtab = [1 3 3 1 3 1 1 3 3 1 1 3 1 3 3 1 3 1 1 3 1 3 3 1 1 3 3 1 3 1 1 3];

% Symbolic constants
syms B J t oI oS t1

coef = [];
for ii = phid
    fprintf(1,'%2d\n',ii)
    ph1 = PO.phmod(ph1tab,ii);
    ph2 = PO.phmod(ph2tab,ii);
    ph3 = PO.phmod(ph3tab,ii);
    ph4 = PO.phmod(ph4tab,ii);
    ph5 = PO.phmod(ph5tab,ii);
    phR = PO.phmod(phRtab,ii);

% Short CP: only I spin being close to 1Hs is polarized.
    rho = PO(2,{'Iz'});% Both I and S are 15N
    rho.dispPOtxt();
    rho = pulse(rho,'I',ph1,pi/2);

% Long CP: both I and S spins are excited.
%     rho = PO(2,{'Iz' 'Sz'});% Both I and S are 15N
%     rho.dispPOtxt();
%     rho = simpulse(rho,{'I' 'S'},{ph1 ph1},{pi/2 pi/2});

    % 1st INEPT
    rho = jc(rho,'IS',pi*J*t);
    rho = simpulse(rho,{'I' 'S'},{ph2 ph2},{pi pi});
    rho = jc(rho,'IS',pi*J*t);

    % 90 pulse - t1 - 90 pulse
    rho = simpulse(rho,{'I' 'S'},{'y' 'y'},{pi/2 pi/2});
    rho = simcs(rho,{'I' 'S'},{oI*t1 oS*t1});
    id_vec = findcoef(rho,{sin(oI*t1) sin(oS*t1)});
    rho = delPO(rho,id_vec);% Delete the term with sin(oI*t1) and sin(oS*t1)
    rho = simpulse(rho,{'I' 'S'},{'y' 'y'},{pi/2 pi/2});
```

```
    % 2nd INEPT
    rho = jc(rho,'IS',pi*J*t);
    rho = simpulse(rho,{'I' 'S'},{ph3 ph3},{pi pi});
    rho = jc(rho,'IS',pi*J*t);

    % Z-filter
    rho = simpulse(rho,{'I' 'S'},{ph4 ph4},{pi/2 pi/2});
    rho = simpulse(rho,{'I' 'S'},{'x' 'x'},{pi/2 pi/2});

    rho = delPO(rho,{'IxSz'});% delete 2IxSz term

    % 15N => 1H CP
    % ph5 is y or -y
    % 180 phase shift of ph5 changes the sign of the signal amplitude.
    if ph5 == 1
        ph5sign = 1;
    elseif ph5 == 3
        ph5sign = -1;
    end

    % Receiver
    % phR is y or -y
    % 180 phase shift of phR changes the sign of the signal amplitude.
    if phR == 1
        phRsign = 1;
    elseif phR == 3
        phRsign = -1;
    end

    coefI_tmp = rho.coef(1)*ph5sign*phRsign;

    I_tmp = coeffs(coefI_tmp,cos(oI*t1));
    I_tmp = I_tmp(2);

    S_tmp = coeffs(coefI_tmp,cos(oS*t1));
    S_tmp = S_tmp(2);

    coef = cat(1,coef,simplify([I_tmp S_tmp],100));

end
```

Published with MATLAB® R2020b

```
% Exp150_RefocusingPulse_PFG.m
% Keeler, J., Understanding NMR Spectroscopy, p. 406, 11.12.3
% Gradient G - 180+d pulse - Gradient G
% The selection of p => -p pathway.
% "Cleaning up" the results of an imperfect 180 pulse.

clear
close all

syms G gH d
pfg_switch = 1;

ini_status = 'DQ';
switch ini_status
    case 'SQ'
        spin_label_cell = {'I1'};
        rho = PO(1,{'I1p'},{1},spin_label_cell);% SQ
    case 'DQ'
        spin_label_cell = {'I1' 'I2'};
        rho = PO(2,{'I1pI2p'},{1},spin_label_cell);% DQ
    case 'TQ'
        spin_label_cell = {'I1' 'I2' 'I3'};
        rho = PO(3,{'I1pI2pI3p'},{1},spin_label_cell);% TQ
end
% % Alternative way to create rho from spin_label_cell
% ns = length(spin_label_cell);
% M_in = zeros(2^ns,2^ns);
% M_in(1,end) = 1;% I1pI2p...Inp
% rho = PO.M2pol(M_in,spin_label_cell);% Speed is a bit slower than PO().

dispPOtxt(rho);
gH_cell = PO.v2cell(gH,spin_label_cell);

% PFG
if pfg_switch == 1
    rho = pfg(rho, G, gH_cell);
end

% Imperfect 180 pulse (pi + d)
rho = simpulse(rho,{'I*'},{'x'},{pi + d});

% PFG
if pfg_switch == 1
    rho = pfg(rho, G, gH_cell);
end

dispPO(rho);

rho = dephase(rho);
dispPO(rho);
```

```
    I1pI2p
CS: I1 G*Z*gH
    I1pI2p*exp(-G*Z*gH*1i)
CS: I2 G*Z*gH
    I1pI2p*exp(-G*Z*gH*2i)
Pulse: I1 d + pi x
     - I1zI2p*exp(-G*Z*gH*2i)*sin(d)*1i - I1pI2p*(exp(-G*Z*gH*2i)*(cos(d) - 1))/2 + I1mI2p*(exp(-G*Z*gH*2i)*(cos(d) + 1))/2
Pulse: I2 d + pi x
     - I1zI2z*exp(-G*Z*gH*2i)*sin(d)^2 + I1zI2p*(exp(-G*Z*gH*2i)*(sin(2*d)*1i - sin(d)*2i))/4 - I1zI2m*(exp(-G*Z*gH*2i)*(sin(2*d)*1i + sin(d)*2i))/4 + I1pI2z*(exp(-G*Z*gH*2i)*(sin(2*d)*1i - sin(d)*2i))/4 + I1pI2p*(exp(-G*Z*gH*2i)*(cos(d) - 1)^2)/4 - I1pI2m*(exp(-G*Z*gH*2i)*(cos(d)^2 - 1))/4 - I1mI2z*(exp(-G*Z*gH*2i)*(sin(2*d)*1i + sin(d)*2i))/4 - I1mI2p*(exp(-G*Z*gH*2i)*(cos(d)^2 - 1))/4 + I1mI2m*(exp(-G*Z*gH*2i)*(cos(d) + 1)^2)/4
CS: I1 G*Z*gH
     - I1zI2z*exp(-G*Z*gH*2i)*sin(d)^2 + I1zI2p*(exp(-G*Z*gH*2i)*(sin(2*d)*1i - sin(d)*2i))/4 - I1zI2m*(exp(-G*Z*gH*2i)*(sin(2*d)*1i + sin(d)*2i))/4 + I1pI2z*(exp(-G*Z*gH*3i)*(sin(2*d)*1i - sin(d)*2i))/4 + I1pI2p*(exp(-G*Z*gH*3i)*(cos(d) - 1)^2)/4 - I1pI2m*(exp(-G*Z*gH*3i)*(cos(d)^2 - 1))/4 - I1mI2z*(exp(-G*Z*gH*1i)*(sin(2*d)*1i + sin(d)*2i))/4 - I1mI2p*(exp(-G*Z*gH*1i)*(cos(d)^2 - 1))/4 + I1mI2m*(exp(-G*Z*gH*1i)*(cos(d) + 1)^2)/4
CS: I2 G*Z*gH
     - I1zI2z*exp(-G*Z*gH*2i)*sin(d)^2 + I1zI2p*(exp(-G*Z*gH*3i)*(sin(2*d)*1i - sin(d)*2i))/4 - I1zI2m*(exp(-G*Z*gH*1i)*(sin(2*d)*1i + sin(d)*2i))/4 + I1pI2z*(exp(-G*Z*gH*3i)*(sin(2*d)*1i - sin(d)*2i))/4 + I1pI2p*(exp(-G*Z*gH*4i)*(cos(d) - 1)^2)/4 - I1pI2m*(exp(-G*Z*gH*2i)*(cos(d)^2 - 1))/4 - I1mI2z*(exp(-G*Z*gH*1i)*(sin(2*d)*1i + sin(d)*2i))/4 - I1mI2p*(exp(-G*Z*gH*2i)*(cos(d)^2 - 1))/4 + I1mI2m*((cos(d) + 1)^2/4)

   1     I1zI2z     -exp(-G*Z*gH*2i)*sin(d)^2
   2     I1zI2p     (exp(-G*Z*gH*3i)*(sin(2*d)*1i - sin(d)*2i))/4
   3     I1zI2m     -(exp(-G*Z*gH*1i)*(sin(2*d)*1i + sin(d)*2i))/4
   4     I1pI2z     (exp(-G*Z*gH*3i)*(sin(2*d)*1i - sin(d)*2i))/4
   5     I1pI2p     (exp(-G*Z*gH*4i)*(cos(d) - 1)^2)/4
   6     I1pI2m     -(exp(-G*Z*gH*2i)*(cos(d)^2 - 1))/4
   7     I1mI2z     -(exp(-G*Z*gH*1i)*(sin(2*d)*1i + sin(d)*2i))/4
   8     I1mI2p     -(exp(-G*Z*gH*2i)*(cos(d)^2 - 1))/4
   9     I1mI2m     (cos(d) + 1)^2/4

Dephasing of terms including Z
    I1mI2m*((cos(d) + 1)^2/4)

   1     I1mI2m     (cos(d) + 1)^2/4
```

Published with MATLAB® R2020b