March 10, 2015

Notes on the 12 qubit PPS

Dawei Lu

Notes about the problems in the 12 qubit PPS preparation, including Matlab codes and Experiments.

DEC 12, 2014

Calculating the state to state GRAPE on Ordi2. In pulsefinder folder. paramsfile is 'twqubit_subS2S.m', and the output file is 'twqubit_7zto12z'.

The GRAPE is to evolve ZZZZZZZIIIII to ZZZZZZZZZZZZ. As the couplings between nearest-neighbored C and H are about 150Hz. I set the GRAPE

```
% Number of timesteps
params.plength = 400;

% Length of each time step
params.timestep = 10e-6;

params.subsystem{1} = [1 2 3 9 10 11];
params.subsystem{2} = [4 5 6 7 8 12];
params.subsys_weight = [6 6];

% Input and goal states for state to state
params.rhoin = mkstate('+1ZZZZZZZIIIII',1);
params.rhogoal = mkstate('+1ZZZZZZZZZZZZ',1);

% Allow Zfreedom or not
params.Zfreedomflag = 1;
```

The fidelity keeps 0 all the time. Guess the reason is 'Zfreedom'. Set 'params.Zfreedomflag = 0;'. However, still 0.

Annie said maybe due to the length. Her SWAP gate requires 8ms, so I changed 'params.plength = 800;'. But for with or without Zfreedom, fidelity is still 0.

Check if some of my GRAPE settings are wrong. try to repeat Annie's SWAP gate calculation.

```
% Number of timesteps
params.plength = 800;

% Length of each time step
params.timestep = 10e-6;

params.subsystem{1} = [1 2 3 9 10 11];
params.subsystem{2} = [4 5 6 7 8 12];
params.subsys_weight = [6 6];

% Input and goal states for state to state
params.rhoin = mkstate('+1IIIIIIIZIIII+1IIIIIIIIZIII+1IIIIIIIIIZII+1IIIIIIIIIIZI
    +1IIIIIIIIIIIZ',1);
params.rhogoal = mkstate('+1IIIIIIIZIIII+1IIIIIIIIZIII+1IIIIIIIIIZII+1
    IIIIIIIIIIZI+1IIIIIIZIIIIZ',1);

% Allow Zfreedom or not
params.Zfreedomflag = 0;
```

The outputfile is 'twqubit_SWAPC7H5'. And the fidelity is already over 98%. Then I changed 'params.Zfreedomflag = 1;', and the fidelity is over 95% after 30 iterations. Much slower than the no Zfreedom case. Maybe due to different initial guesses.

DEC 15, 2014

Generate all $\pi/2$ and π pulses for the 7 Carbons, with the Calibration = 25KHz. $\pi/2$ pulses are 1ms length and 100 steps, and π pulses are 2ms length and 200 steps. Generating Code in 'twqubit_shape.m'

```
for ii = 1:7
loadfile = ['twqubit_C', num2str(ii), '180', '.mat'];
eval(['load ', loadfile]);
filename1 = ['twqubit_C', num2str(ii), '180_C_25000.txt'];
filename2 = ['twqubit_C', num2str(ii), '180_H_25000.txt'];
make_bruker_shape(pulses{1}, 25000, filename1,1);
make_bruker_shape(pulses{1}, 25000, filename2,2);
end
```

The pulses are saved in Ordi2 '\pulsefinder\12 Qubit\' with the names such as 'twqubit_C590_C_25000.txt'.

I checked all the fidelities of the $\pi/2$ pulses in the folder '\pulseexam_12qubit\C_rotations\check_grape.m'. The code is

```
load Para.mat
load twpauliX_full.mat
load twpauliY_full.mat

%% Check all 90 rotations
%% Parameters for the GRAPE pulse
for spin_number = 1:7
Name1 = ['twqubit_C', num2str(spin_number), '90_C_25000.txt'];
Name2 = ['twqubit_C', num2str(spin_number), '90_H_25000.txt'];
Amplitude = 25000;
Time = 1e-3;
Length = 100;
dt = Time/Length;
FirstLine = 19; % the first line which contains the information of power and
    phase

Output1 = 'test1';
Output2 = 'test2';

[power1,phase1]=dataout(Name1,Output1,FirstLine,Length);
[power2,phase2]=dataout(Name2,Output2,FirstLine,Length);
%% Check
X_C = 0; Y_C = 0;
for jj = 1:7
    X_C = X_C + KIx{jj};
    Y_C = Y_C + KIy{jj};
end

X_H = 0; Y_H = 0;
for jj = 8:12
    X_H = X_H + KIx{jj};
    Y_H = Y_H + KIy{jj};
end


U = eye(2^12);
U = U*expm(-i*H*4e-6);
for ii = 1:Length
```

```
    Hext = 2*pi*(Amplitude*power1(ii)/100)*(X_C*cos(phase1(ii)/360*2*pi)-Y_C*sin
        (phase1(ii)/360*2*pi))+2*pi*(Amplitude*power2(ii)/100)*(X_H*cos(phase2(ii
        )/360*2*pi)-Y_H*sin(phase2(ii)/360*2*pi));
    U = expm(-i*(Hext+H)*dt)*U;
end
U = U*expm(-i*H*4e-6);

Utar = expm(-i*KIx{spin_number}*pi/2);

% Fidelity = ['Fidelity_C', num2str(spin_number), '90'];
% eval(['Fidelity_C', num2str(spin_number), '90 = abs(trace(U*Utar'))/2^12']);
Fidelity = abs(trace(U*Utar'))/2^12

savefile = ['twqubit_C', num2str(spin_number), '90_Ufid.mat'];
save (savefile, 'U', 'Fidelity');

end
```

Unitaries and Fidelities of the pulses will both be saved in 'twqubit_C590_Ufid.mat', so they can be called for further calculations in the PPS simulation. Wait for the results.

DEC 16, 2014

Combine pulses in the PPS preparation into big shape files, which should be easy for calibrations and pulsefixing.

The code is in the SVN server for Matlab named '\Twqubit\pulse_combine.m'.

First read all the powers and phases for the $\pi/2$ and π rotations.

```
for spin_number = 1:7
       Name1 = ['twqubit_C', num2str(spin_number), '90_C_25000.txt'];
       Name2 = ['twqubit_C', num2str(spin_number), '90_H_25000.txt'];
       [power1,phase1]=dataout(Name1,Output1,FirstLine,Length_90);
       [power2,phase2]=dataout(Name2,Output2,FirstLine,Length_90);
       eval(['power_C', num2str(spin_number), '90_C = power1;']); eval(['phase_C
          ', num2str(spin_number), '90_C = phase1;']);
       eval(['power_H', num2str(spin_number), '90_H = power2;']); eval(['phase_H
          ', num2str(spin_number), '90_H = phase2;']);
end

for spin_number = 1:7
       Name1 = ['twqubit_C', num2str(spin_number), '180_C_25000.txt'];
       Name2 = ['twqubit_C', num2str(spin_number), '180_H_25000.txt'];
       [power1,phase1]=dataout(Name1,Output1,FirstLine,Length_180);
       [power2,phase2]=dataout(Name2,Output2,FirstLine,Length_180);
       eval(['power_C', num2str(spin_number), '180_C = power1;']); eval(['
          phase_C', num2str(spin_number), '180_C = phase1;']);
       eval(['power_H', num2str(spin_number), '180_H = power2;']); eval(['
          phase_H', num2str(spin_number), '180_H = phase2;']);
end
```

Then combine them with the free evolutions. Here I set the time step dt = 10us.

```
%% From Z7 to Z24567
step_27 = round(1/(4*Para(2,7))/dt);
step_67_27 = round((1/(4*Para(6,7))-1/(4*Para(2,7)))/dt);
step_47_67 = round((1/(4*Para(4,7))-1/(4*Para(6,7)))/dt);
step_57_47 = round((1/(4*Para(5,7))-1/(4*Para(4,7)))/dt);
step_57 = round((1/(4*Para(5,7)))/dt);

power_encoding1_C = [power_C790_C; zeros(step_27,1);power_C2180_C; zeros(
   step_67_27,1); power_C6180_C; zeros(step_47_67,1);power_C4180_C; zeros(
   step_57_47,1);...
                                     power_C5180_C; power_C7180_C; zeros(step_57,1)
                                        ;power_C790_C]*Calibration/Calibration_old;
phase_encoding1_C = [phase_C790_C; zeros(step_27,1);phase_C2180_C; zeros(
   step_67_27,1); phase_C6180_C; zeros(step_47_67,1);phase_C4180_C; zeros(
   step_57_47,1);...
                                     phase_C5180_C; phase_C7180_C; zeros(step_57,1)
                                        ;mod((phase_C790_C+90),360)];
power_encoding1_H = [power_H790_H; zeros(step_27,1);power_H2180_H; zeros(
   step_67_27,1); power_H6180_H; zeros(step_47_67,1);power_H4180_H; zeros(
   step_57_47,1);...
                                     power_H5180_H; power_H7180_H; zeros(step_57,1)
                                        ;power_H790_H]*Calibration/Calibration_old;
phase_encoding1_H = [phase_H790_H; zeros(step_27,1);phase_H2180_H; zeros(
   step_67_27,1); phase_H6180_H; zeros(step_47_67,1);phase_H4180_H; zeros(
   step_57_47,1);...
```

```
                                   phase_H5180_H; phase_H7180_H; zeros(step_57,1)
                                      ;mod((phase_H790_H+90),360)];

total_time_encoding1 = length(power_encoding1_C)*dt;

outputfile = 'twqubit_encoding1_C';
shpfile = fopen(outputfile,'w');
    fprintf(shpfile,'##TITLE= %s\n',outputfile);
    fprintf(shpfile,'##JCAMP-DX= 5.00 Bruker JCAMP library\n');
    fprintf(shpfile,'##DATA TYPE= Shape Data\n');
    fprintf(shpfile,'##ORIGIN= Dawei''s GRAPE Pulses \n');
    fprintf(shpfile,'##OWNER= Dawei\n');
    fprintf(shpfile,'##DATE= %s\n',date);
    time = clock;
    fprintf(shpfile,'##TIME= %d:%d\n',fix(time(4)),fix(time(5)));
    fprintf(shpfile,'##MINX= %7.6e\n',min(power_encoding1_C));
    fprintf(shpfile,'##MAXX= %7.6e\n',max(power_encoding1_C));
    fprintf(shpfile,'##MINY= %7.6e\n',min(phase_encoding1_C));
    fprintf(shpfile,'##MAXY= %7.6e\n',max(phase_encoding1_C));
    fprintf(shpfile,'##$SHAPE_EXMODE= None\n');
    fprintf(shpfile,'##$SHAPE_TOTROT= %7.6e\n',90);
    fprintf(shpfile,'##$SHAPE_BWFAC= %7.6e\n',1);
    fprintf(shpfile,'##$SHAPE_INTEGFAC= %7.6e\n',1);
    fprintf(shpfile,'##$SHAPE_MODE= 1\n');
    fprintf(shpfile, '##PULSE_WIDTH= %d\n',total_time_encoding1);
    fprintf(shpfile, '##Calibration_Power= %d\n',Calibration);
    fprintf(shpfile,'##NPOINTS= %d\n',length(power_encoding1_C));
    fprintf(shpfile,'##XYPOINTS= (XY..XY)\n');

for ii = 1:length(power_encoding1_C)
    fprintf(shpfile,'  %7.6e,  %7.6e\n',power_encoding1_C(ii),phase_encoding1_C(
       ii));
end

    fprintf(shpfile,'##END=\n');

outputfile = 'twqubit_encoding1_H';
shpfile = fopen(outputfile,'w');
    fprintf(shpfile,'##TITLE= %s\n',outputfile);
    fprintf(shpfile,'##JCAMP-DX= 5.00 Bruker JCAMP library\n');
    fprintf(shpfile,'##DATA TYPE= Shape Data\n');
    fprintf(shpfile,'##ORIGIN= Dawei''s GRAPE Pulses \n');
    fprintf(shpfile,'##OWNER= Dawei\n');
    fprintf(shpfile,'##DATE= %s\n',date);
    time = clock;
    fprintf(shpfile,'##TIME= %d:%d\n',fix(time(4)),fix(time(5)));
    fprintf(shpfile,'##MINX= %7.6e\n',min(power_encoding1_H));
    fprintf(shpfile,'##MAXX= %7.6e\n',max(power_encoding1_H));
    fprintf(shpfile,'##MINY= %7.6e\n',min(phase_encoding1_H));
    fprintf(shpfile,'##MAXY= %7.6e\n',max(phase_encoding1_H));
    fprintf(shpfile,'##$SHAPE_EXMODE= None\n');
    fprintf(shpfile,'##$SHAPE_TOTROT= %7.6e\n',90);
    fprintf(shpfile,'##$SHAPE_BWFAC= %7.6e\n',1);
    fprintf(shpfile,'##$SHAPE_INTEGFAC= %7.6e\n',1);
    fprintf(shpfile,'##$SHAPE_MODE= 1\n');
    fprintf(shpfile, '##PULSE_WIDTH= %d\n',total_time_encoding1);
```

```
    fprintf(shpfile, '##Calibration_Power= %d\n',Calibration);
    fprintf(shpfile,'##NPOINTS= %d\n',length(power_encoding1_H));
    fprintf(shpfile,'##XYPOINTS= (XY..XY)\n');

for ii = 1:length(power_encoding1_H)
    fprintf(shpfile,'  %7.6e,  %7.6e\n',power_encoding1_H(ii),phase_encoding1_H(
       ii));
end

    fprintf(shpfile,'##END=\n');
```

The two output files are 'twqubit_encoding1_C' and 'twqubit_encoding1_H'. The calibrations are 25000Hz.

DEC 17, 2014

All fidelities of $\pi/2$ pulses are done! The folder is '\pulseexam_12qubit\C_rotations\'. Use 'check_power.m' to check the maximal powers for C and H channel.

Rotation	Fidelity	File	MaxPower C	MaxPower H
$R_x^1(\pi/2)$	0.9838	twqubit_C190_Ufid.mat	56.0%, 14000Hz	22.3%, 5557Hz
$R_x^2(\pi/2)$	0.9758	twqubit_C290_Ufid.mat	41.7%, 10422Hz	23.5%, 5878Hz
$R_x^3(\pi/2)$	0.9647	twqubit_C390_Ufid.mat	31.9%, 7979.0Hz	22.3%, 5568Hz
$R_x^4(\pi/2)$	0.9801	twqubit_C490_Ufid.mat	31.6%, 7892.0Hz	23.8%, 5954Hz
$R_x^5(\pi/2)$	0.9936	twqubit_C590_Ufid.mat	56.1%, 14033Hz	30.7%, 7678Hz
$R_x^6(\pi/2)$	0.9683	twqubit_C690_Ufid.mat	57.3%, 14333Hz	34.4%, 8595Hz
$R_x^7(\pi/2)$	0.9857	twqubit_C790_Ufid.mat	43.7%, 10925Hz	24.8%, 6207Hz
$R_x^1(\pi)$	0.9699	twqubit_C1180_Ufid.mat	62.6%, 15655Hz	34.9%, 8726Hz
$R_x^2(\pi)$	0.9537	twqubit_C2180_Ufid.mat	51.1%, 12783Hz	32.4%, 8094Hz
$R_x^3(\pi)$	0.9330	twqubit_C3180_Ufid.mat	37.4%, 9350.0Hz	24.0%, 5997Hz
$R_x^4(\pi)$	0.9639	twqubit_C4180_Ufid.mat	45.1%, 11268Hz	20.4%, 5108Hz
$R_x^5(\pi)$	0.9904	twqubit_C5180_Ufid.mat	67.6%, 16895Hz	31.1%, 7782Hz
$R_x^6(\pi)$	0.9393	twqubit_C6180_Ufid.mat	71.8%, 17948Hz	33.6%, 8396Hz
$R_x^7(\pi)$	0.9743	twqubit_C7180_Ufid.mat	51.0%, 12759Hz	32.1%, 8022Hz

For π pulses, the maximal power of C5 exceeds 100% so it cannot be used. Check if we can generate π pulses by combining two $\pi/2$ pulses. A potential problem is when calculating the GRAPE, we have considered the 4us free evolutions in the beginning and in the end. If we combine, we will have an unwanted 8us free evolution in the middle of the new π pulse.

Use 'combine90to180' to check the π pulse fidelity. They are very bad actually. All of them are just 0.75 0.76 in fidelity.

So I run 'check_grape.m' to check the fidelities of the π pulses. Only from C1 to C4, as C5 has exceeds the power limit 25000Hz.

DEC 22, 2014

Got 4 GRAPE pulses for encoding. The folder is '\pulseexam_12qubit\C_rotations\'. The fidelities are in calculation on Ordi2.

Rotation	Fidelity	File	MaxPower C	MaxPower H
$R_x^{5,7}(\pi)$	0.9667	twqubit_C57180_Ufid.mat	32.3%, 8072.5Hz	24.2%, 6049Hz
$R_x^{2,3}(\pi)$	0.8908	twqubit_C23180_Ufid.mat	32.4%, 8101.5Hz	22.8%, 5701Hz
$R_x^{2,3,4,7}(\pi/2)$	0.9156	twqubit_C234790_Ufid.mat	37.4%, 9358.3Hz	28.9%, 7213Hz
$R_x^{1,5,6}(\pi)$	0.9055	twqubit_C156180_Ufid.mat	32.2%, 8039.7Hz	20.3%, 5086Hz

Have to recalculate many π pulses.

DEC 23, 2014

Got π pulse on C6. Combine two $\pi/2$ pulses as the initial guess, with the fidelity 0.75, and then search the optimal π pulse. The convergence speed is very fast, which means initial guess is indeed very important in 12 qubits.

Now C2, C3, C5, C7 π pulses are in calculation, with the initial guess.

FEB 05, 2015

C2, C3, C5, C7 π pulses are all finished, and the update is in the Section Dec 17, 2014.

Also submitted the last pulse for the Encoding. From the PPS.m file in folder 'Twqubit'

```
%Phase Correction
U7 = R(gop(2,X),90)*R(gop(2,-Z),360*(Para(2,2)-20696)*1/2/148.5)*...
R(gop(3,-Y),90)*R(gop(3,-Z),360*(Para(3,3)-20696)*1/2/148.5)*...
R(gop(4,X),90)*R(gop(4,-Z),360*(Para(4,4)-20696)*1/2/148.5)*...
R(gop(7,X),90)*R(gop(7,-Z),360*(Para(7,7)-20696)*1/2/148.5);
```

And the operator is in 'twqubit_sub_234790_and_phasecorrection.m'

```
params.Uwant = expm(-1i*(90*pi/180)/2*full(mkstate('+1IXIIIIIIIIII-1IIYIIIIIIIII+1IIIXIIIIIIII+1IIIIIIXIIIII',0)))*...
expm(-1i*((8778.95-20696)*1/2/148.5*360*pi/180)/2*full(mkstate('-1IZIIIIIIIIII',0)))*...
expm(-1i*((6245.16675-20696)*1/2/148.5*360*pi/180)/2*full(mkstate('-1IIZIIIIIIIII',0)))*...
expm(-1i*((10333.55-20696)*1/2/148.5*360*pi/180)/2*full(mkstate('-1IIIZIIIIIIII',0)))*...
expm(-1i*((11928.21998-20696)*1/2/148.5*360*pi/180)/2*full(mkstate('-1IIIIIIZIIIII',0)));
```

FEB 06, 2015

The pulse 'twqubit_sub_234790_and_phasecorrection.m' which is the last piece of the encoding was found!

All π pulses are found. The lowest is 0.9330 for C3 and the highest is 0.9904 for C5. Now is calculating the fidelity of the last piece in Encoding 'twqubit_C234790withPC_Ufid.mat'. Will combine all of the pulses in Encoding and check again after this calculation.

The last piece 'twqubit_C234790withPC_Ufid.mat' has been checked. The fidelity is 0.9164.

FEB 12, 2015

When combining all pulses into a large shape file, one has to know how to change a shape for X rotation to Y rotation. It should be a $\pi/2$ phase difference for every segment in the shape. I am going to check it.

The checking uses 4-qubit Crotonic in the folder 'F:\matlab\pulseexam_7qubit\4 qubit pulse check'. The target unitary is $R_x^1(\pi/2)$, and the pulse is 'Croton_90x1.txt' with length 1ms, 500 segments and amplitude 6000Hz.

When compared with $R_x^1(\pi/2)$, the fidelity is 0.9996. Then I changed the target to $R_y^1(\pi/2)$ with the GRAPE pulse unchanged. The fidelity goes to 0.4998 which is reasonable.

In order to produce a $R_y^1(\pi/2)$ from the original X rotation pulse, I added 90 to all phases in all segments. However, the fidelity goes to almost 0. Again all phases are reduced by 90, and this time the fidelity is 0.9996, which is what we want!

Conclusion: If you want to realize a Y rotation based on a X rotation pulse, just change the phase to phase-90 in each segment, and mod by 360 for the spectrometer.

```
phase = phase - 90;
phase = mod(phase, 360);
```

I wrote a program 'grape_phase' to generate the new phase in the folder 'F:\matlab\pulseexam_7qubit\'. Used in this manner 'phase_new = grape_phase(phase, initial_phase, end_phase)', where the initial_phase and end_phase can be X, Y, -X, or -Y.

The way to get the new operator is through the equation $R_z(\theta) = X R_y(\theta) \bar{X}$. If you know the unitary U_x of the X rotation pulse, and when you are realizing Y pulse from that one, the new unitary U_y is thus

$$U_y = R_z(\pi/2) U_x R_z(-\pi/2); \tag{1}$$

Generated the first encoding part, which will evolve Z7 to Z24567. The files 'twqubit_encoding1_C' and 'twqubit_encoding1_H' are in Ordi2 '\pulseexam_12qubit'. The total length is 32.98ms. Next I have to check whether the final state after this pulse will be Z24567 or not. So 'check_encoding.m' is written. The directory is Ordi2 '\pulseexam_12qubit'. This function will load all necessary .mat files to get the unitaries and calculate the final state based on these unitaries from Z7.

The final fidelity is 0.9832 (the same for with or without gradient) for Z24567. Two files 'U_encoding1.mat' and 'rho_encoding1.mat'. Now go on to the second Encoding part.

The fidelity for the second Encoding part is -0.9692 (the same for with or without gradient) for Z1234567. Two files 'twqubit_encoding2_C' and 'twqubit_encoding2_H' are in Ordi2 '\pulseexam_12qubit'. The total length is 21.29ms. Then the last piece in Encoding!

The fidelity for the third Encoding part is -0.9160 (the same for with or without gradient) for Z123456789101112. Two files 'twqubit_encoding3_C' and 'twqubit_encoding3_H' are in Ordi2 '\pulseexam_12qubit'. The total length is 7.36ms.

Update on Mar 04: In the last code, the free evolution time is not the integer times of 10us, but it should be. So the function 'F.m' is modified as

```
function F=F(Hamiltonian, time)
F=diag(exp(diag(-i*Hamiltonian*10e-6*round(time/10e-6))),0);
```

And all the fidelities are recalculated. They are 0.9831 for Encoding 1, -0.7096 for Encoding 2,

FEB 18, 2015

Running the two pulses. One is 'twqubit_all90.m' which is used for phase cycling, and the other one is 'twqubit_all90butC7.m' used for the polarization crush in the beginning. The fidelities are obtained!

Rotation	Length	Fidelity	File	MaxPower C	MaxPower H
$R_x^{1-12}(\pi/2)$	1ms	0.8738	twqubit_all90_Ufid.mat	27.8%, 6956.6Hz	30.4%, 7594Hz
$R_x^{1-6,8-12}(\pi/2)$	1ms	0.8849	twqubit_all90butC7_Ufid.mat	24.5%, 6134.9Hz	25.0%, 6239Hz

FEB 23, 2015

Running all 6 pulses for the decoding part. Got 2, 3, 5 and 6! But the first pulse 'twqubit_C2347andH180.m' has to be recalculated.

Then check all the fidelities for pulse 2, 3, 5 and 6 first.

MAR 3, 2015

All 6 pulses in decoding have been checked. To my surprise some fidelities are pretty, such as the fidelity of $R_x^{2,3,4,5,6}(\pi)$ is only 0.8058. Anyway, it is better to check the state fidelity right away.

Rotation	Length	Fidelity	File	MaxPower C	MaxPower H
$R_x^{2,3,4,7-12}(\pi)$	2ms	0.8424	twqubit_C2347andH180_Ufid.mat	61.6%, 15400Hz	52.2%, 13039Hz
$R_x^{1,3,4,6}(\pi/2)R_{-y}^{8-12}(\pi/2)$	1ms	0.9080	twqubit_C134690andH90_Ufid.mat	24.8%, 6203.2Hz	22.1%, 5529Hz
$R_x^{2,3,4,5,6}(\pi)$	2ms	0.8058	twqubit_C23456180_Ufid.mat	37.8%, 9438.2Hz	23.0%, 5746Hz
$R_{-y}^{1,3,4,6}(\pi/2)R_x^2(\pi/2)R_{-z}^1(\frac{1}{2J23} - \frac{1}{2J12})$	1ms	0.8881	twqubit_C1234690withPC_Ufid.mat	21.2%, 5310.9Hz	22.9%, 5724Hz
$R_x^{2,7}(\pi)$	2ms	0.9306	twqubit_C27180_Ufid.mat	29.1%, 7285.3Hz	21.7%, 5414Hz
$R_{-y}^2(\pi/2)R_x^5(\pi/2)$	1ms	0.9728	twqubit_C2Y5X90_Ufid.mat	59.2%, 14795Hz	40.9%, 10232Hz

MAR 4, 2015: RE-CHECK THE PPS CIRCUIT IN SIMULATION

I found the decoding part can be simplified. Check it in Matlab.

All the files are saved in Ordi2 '\twqubit\'. For the encoding which consists of three parts, the fidelities have been checked before.

TABLE I. Encoding in 12-qubit PPS

Target State	State Fidelity	Density Matrix	Unitary Operator
IZIZZZXIIIII	1.0000	rho_encoding1.mat	U_encoding1.mat
ZXZZZZZIIIII	-1.0000	rho_encoding2.mat	U_encoding2.mat
ZZZZZZZZZZZZ	-0.9500	rho_encoding3.mat	U_encoding3.mat

For the phase cycling it has been done too.

TABLE II. Phase Cycling in 12-qubit PPS

Target State	State Fidelity	Density Matrix	Unitary Operator
$\|00...0\rangle\langle 00...0\| + \|11...1\rangle\langle 11...1\|$	0.9511	rho_phasecycling.mat	NAN

For the decoding part, the simulation is as follows.

TABLE III. Decoding in 12-qubit PPS

Target State	State Fidelity	Density Matrix	Unitary Operator
$A_1A_5A_6\|+---++-00000\rangle+$ $A'_1A'_5A'_6\|-+++--+00000\rangle$	0.8717	rho_decoding1.mat	U_decoding1.mat
$A_1A_5A_6A_7\|0+00-0-00000\rangle+$ $A'_1A'_5A'_6A'_7\|0-00+0+00000\rangle$	0.8570	rho_decoding2.mat	U_decoding2.mat
$A_1A_5A_6A_7A_5^{new}\|000000-00000\rangle+$ $A'_1A'_5A'_6A'_7A_5^{new'}\|000000+00000\rangle$	0.8570	rho_decoding3.mat	U_decoding3.mat

A_1, A_5 and A_6 are phases produced by the chemical shift evolutions of C1, C5 and C6 during $1/2\mathrm{J}_{\mathrm{C7H5}}$. A_7 is the phase by the chemical shift evolution of C7 (**Note: C7 is x-iy so the phase is the conjugate**) during $1/2\mathrm{J}_{\mathrm{C2C3}}$, and the coupling evolutions between C7 and all protons. A_5^{new} is the phase of C5 again in $1/2\mathrm{J}_{27}$, including the chemical shift evolution and coupling evolutions (J25 and J57 are switched off). So the phases are

$$\begin{aligned}
A_1 &= cos(2\pi(\omega_1 - O_1)/2\mathrm{J}_{\mathrm{C7H5}}) - isin(2\pi(\omega_1 - O_1)/2\mathrm{J}_{\mathrm{C7H5}}), \\
A_5 &= cos(2\pi(\omega_5 - O_1)/2\mathrm{J}_{\mathrm{C7H5}}) - isin(2\pi(\omega_5 - O_1)/2\mathrm{J}_{\mathrm{C7H5}}), \\
A_6 &= cos(2\pi(\omega_6 - O_1)/2\mathrm{J}_{\mathrm{C7H5}}) - isin(2\pi(\omega_6 - O_1)/2\mathrm{J}_{\mathrm{C7H5}}), \\
A_7 &= cos(2\pi(\omega_7 - O_1)/2\mathrm{J}_{23}) + isin(2\pi(\omega_7 - O_1)/2\mathrm{J}_{23}) * \\
&\prod_{k=8}^{12}(cos(\pi\mathrm{J}_{7k}/\mathrm{J}_{23}) + isin(\pi\mathrm{J}_{7k}/\mathrm{J}_{23})), \\
A_5^{new} &= cos(2\pi(\omega_5 - O_1)/2\mathrm{J}_{27}) + isin(2\pi(\omega_5 - O_1)/2\mathrm{J}_{27}) * \\
&\prod_{k\neq 2,5,7}(cos(\pi\mathrm{J}_{5k}/\mathrm{J}_{27}) + isin(\pi\mathrm{J}_{5k}/\mathrm{J}_{27}))
\end{aligned} \tag{2}$$

When calculating the evolutions in the Decoding part, using the following equations (the evolution is 1/2J with two π pulses inserted in the middle. **Note: without π in the end**)

$$\begin{aligned}
(X+iY)\otimes(X+iZ) &\longrightarrow (X-iY)\otimes(I+Z), \\
(X-iY)\otimes(X+iZ) &\longrightarrow (X+iY)\otimes(I-Z), \\
(X+iY)\otimes(X-iZ) &\longrightarrow (X+iY)\otimes(I-Z), \\
(X-iY)\otimes(X-iZ) &\longrightarrow (X+iY)\otimes(I+Z).
\end{aligned} \tag{3}$$

MAR 10, 2015: CHECK THE SIMULATED SPECTRA FOR THE 12 QUBIT PPS CIRCUIT

The final state 'rho_decoding3.mat' includes a phase on C7. 'rho_decoding3(1,33) = 0.1979 + 0.5728i', and I used a Z rotation $R_z^7(\theta)$ to rotate it into X. The rotating angle θ is 1.2381, and the element becomes 0.6060 after the rotation. Note 0.6060/0.7071 (the ideal value) = 0.8570 which is exactly the fidelity for the final state 'rho_decoding3'. The new state is named 'rho_12pps_circuit', and is saved in Ordi2 '\twqubit\rho_12pps_circuit.mat'.

The simulation is implemented in Ordi2 '\NMR\Experiments\twqubit'. The main file is 'sim_twqubit.m'.

Some settings: o1_C = 20696, o1_H = 2894, td = 281684, swh = 30030.

ALL PULSES FOR 12 QUBITS

The saving folder is '\pulseexam_12qubit\C_rotations\'.

$\pi/2$ and π rotations on every single spin.

Rotation	Length	Fidelity	File	MaxPower C	MaxPower H
$R_x^1(\pi/2)$	1ms	0.9838	twqubit_C190_Ufid.mat	56.0%, 14000Hz	22.3%, 5557Hz
$R_x^2(\pi/2)$	1ms	0.9758	twqubit_C290_Ufid.mat	41.7%, 10422Hz	23.5%, 5878Hz
$R_x^3(\pi/2)$	1ms	0.9647	twqubit_C390_Ufid.mat	31.9%, 7979.0Hz	22.3%, 5568Hz
$R_x^4(\pi/2)$	1ms	0.9801	twqubit_C490_Ufid.mat	31.6%, 7892.0Hz	23.8%, 5954Hz
$R_x^5(\pi/2)$	1ms	0.9936	twqubit_C590_Ufid.mat	56.1%, 14033Hz	30.7%, 7678Hz
$R_x^6(\pi/2)$	1ms	0.9683	twqubit_C690_Ufid.mat	57.3%, 14333Hz	34.4%, 8595Hz
$R_x^7(\pi/2)$	1ms	0.9857	twqubit_C790_Ufid.mat	43.7%, 10925Hz	24.8%, 6207Hz
$R_x^1(\pi)$	2ms	0.9699	twqubit_C1180_Ufid.mat	62.6%, 15655Hz	34.9%, 8726Hz
$R_x^2(\pi)$	2ms	0.9537	twqubit_C2180_Ufid.mat	51.1%, 12783Hz	32.4%, 8094Hz
$R_x^3(\pi)$	2ms	0.9330	twqubit_C3180_Ufid.mat	37.4%, 9350.0Hz	24.0%, 5997Hz
$R_x^4(\pi)$	2ms	0.9639	twqubit_C4180_Ufid.mat	45.1%, 11268Hz	20.4%, 5108Hz
$R_x^5(\pi)$	2ms	0.9904	twqubit_C5180_Ufid.mat	67.6%, 16895Hz	31.1%, 7782Hz
$R_x^6(\pi)$	2ms	0.9393	twqubit_C6180_Ufid.mat	71.8%, 17948Hz	33.6%, 8396Hz
$R_x^7(\pi)$	2ms	0.9743	twqubit_C7180_Ufid.mat	51.0%, 12759Hz	32.1%, 8022Hz

Pulses for the encoding part of PPS preparation.

Rotation	Length	Fidelity	File	MaxPower C	MaxPower H
$R_x^{5,7}(\pi)$	2ms	0.9667	twqubit_C57180_Ufid.mat	32.3%, 8072.5Hz	24.2%, 6049Hz
$R_x^{2,3}(\pi)$	2ms	0.8908	twqubit_C23180_Ufid.mat	32.4%, 8101.5Hz	22.8%, 5701Hz
$R_x^{2,3,4,7}(\pi/2)$	1ms	0.9156	twqubit_C234790_Ufid.mat	37.4%, 9358.3Hz	28.9%, 7213Hz
$R_x^{1,5,6}(\pi)$	2ms	0.9055	twqubit_C156180_Ufid.mat	32.2%, 8039.7Hz	20.3%, 5086Hz
$R_x^{2,4,7}(\pi/2)R_{-y}^{3}(\pi/2)R_{-z}^{i=2,3,4,7}((w_i - O_1) * 1/2J_{CH})$	1ms	0.9164	twqubit_C234790withPC_Ufid.mat	47.4%, 11841Hz	25.3%, 6336Hz

All 3 Encoding pulses in the saving folder '\pulseexam_12qubit\'. The fidelities in the following table are state fidelities

Files	Length	Target State	State Fidelity
twqubit_encoding1_C, H	32.98ms	IZIZZZZIIIII	0.9832
twqubit_encoding2_C, H	21.29ms	ZZZZZZZIIIII	-0.9692
twqubit_encoding3_C, H	7.36ms	ZZZZZZZZZZZZ	-0.9160

Pulses for polarization crush and phase cycling.

Rotation	Length	Fidelity	File	MaxPower C	MaxPower H
$R_x^{1-12}(\pi/2)$	1ms	0.8738	twqubit_all90_Ufid.mat	27.8%, 6956.6Hz	30.4%, 7594Hz
$R_x^{1-6,8-12}(\pi/2)$	1ms	0.8849	twqubit_all90butC7_Ufid.mat	24.5%, 6134.9Hz	25.0%, 6239Hz

Pulses for the decoding part of PPS preparation.

Rotation	Length	Fidelity	File	MaxPower C	MaxPower H
$R_x^{2,3,4,7-12}(\pi)$	2ms	0.8424	twqubit_C2347andH180_Ufid.mat	61.6%, 15400Hz	52.2%, 13039Hz
$R_x^{1,3,4,6}(\pi/2)R_{-y}^{8-12}(\pi/2)$	1ms	0.9080	twqubit_C134690andH90_Ufid.mat	24.8%, 6203.2Hz	22.1%, 5529Hz
$R_x^{2,3,4,5,6}(\pi)$	2ms	0.8058	twqubit_C23456180_Ufid.mat	37.8%, 9438.2Hz	23.0%, 5746Hz
$R_{-y}^{4,6}R_y^{1,3}(\pi/2)R_x^2(\pi/2)R_{-z}^1(\frac{1}{2J23}-\frac{1}{2J12})$	1ms	0.8884	twqubit_C1234690withPC_Ufid.mat	26.4%, 6596.0Hz	24.4%, 6110Hz
$R_x^{2,7}(\pi)$	2ms	0.9306	twqubit_C27180_Ufid.mat	29.1%, 7285.3Hz	21.7%, 5414Hz
$R_y^2(\pi/2)R_x^5(\pi/2)$	1ms	0.9715	twqubit_C2Y5X90_Ufid.mat	28.9%, 7233.9Hz	29.2%, 7292Hz