December 17, 2014

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%, 79790Hz	22.3%, 5568Hz
$R_x^4(\pi/2)$	0.9801	twqubit_C490_Ufid.mat	31.6%, 78920Hz	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

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.