Corresponding Manuscript: Conditional Operation of Hole Spin Qubits above 1 K — [Notebook: S4-1_S4-2_Estimate_bare_f_Rabi_J38.7MHZ_v040925_v1.nb]
Uploaded/curated in current form by (blame): M.J. Carballido | ORCID: https://orcid.org/0000-0001-7385-8284
Date: Oct. 4th, 2025

Project mirror: https://github.com/carmig00/Publications-OpenAccess-Code-Conditional2Q

How to run: Evaluate top-to-bottom (Evaluation → Evaluate Notebook).
Outputs: HDF5 (.h5) written to ./exports.

Figure mapping: This notebook reproduces the matrix of Rabi chevrons in the supplementary material SX with the purpose of identifying the adequate bare Rabi frequencies for simulating the 2-tone experiments later on. It also simulates supplementary material SY a which was a single take without random sampling to show how an ideal Rabi chevron could look with no random sampling/varying of the Larmor frequency.

Comment/Note:
-The fast qubit (high f_Rabi), at lower Larmor frequency, is the one located physically on the right and is represented by pink/purple color tones, here qubit 1.
 - The slow qubit (low f_Rabi), at higher Larmor frequency, is the one located physically on the left and is represented by orange/yellow color tones, here qubit 2.

```
(* Parameters *)

(* f_Rabi, MHz *)
(* These are the bare Rabi frequencies of the two ideally isolated spin
 qubits (J=0) which we concluded best match our experimental Rabi Chevron
 data. Below we describe how the matrix was calculated mapping different
 fR1t and fR2t against each other in order to identify the best match.*)
fR1t = 24;
fR2t = 13;

(* NOT USED HERE (because burst time is swept in Chevron):
 Experimentally chosen pulse durations to induce a Pi/2 flip for qubit 1 and qubit 2,
see supplementary material for details on the calibration *)
(* units in \mus *)
Tp1 = 20×10^-3; (* corresponding to 20 ns *)
Tp2 = 30×10^-3; (* corresponding to 30 ns *)

(* Effective exchnage interaction J, MHz *)
Jval = 38.7;

(* Larmor frequencies, 2*Pi*MHz *)
```

```
(*NOTE: these larmor frequencies were chosen by orienting ourselves on the center
  position of the EDSR peaks which are not the most accurate representation of
  the precise Larmor frequency (due to convolution of two Chevrons). Only after,
in the simulation of the 2-tone experiment we could more precisely narrow-
 down and adjust the larmor frequencies as fit parameters. For
  a first estimate this choice was sufficiently accurate.*)
ωz1 = 2 Pi 2094;
ωz2 = 2 Pi 2284;

(*To simulate noise, we averaged over different Larmor
 frequencies sampled from a random, normal distributed set.*)
(*sigma of normal distributuion 2*Pi*MHz *)
σωz1 = 25 × 2 Pi;
σωz2 = 25 × 2 Pi;

(* Resolution of the array of random Larmor frequencies MHz*)
resωz1 = 150;
resωz2 = 150;
(* Arrays of random Larmor freqs. MHz*)
ωz1arr = Abs[RandomVariate[NormalDistribution[ωz1, σωz1], {resωz1}]];
ωz2arr = Abs[RandomVariate[NormalDistribution[ωz2, σωz2], {resωz2}]];

(*NOT USED HERE, this is for varying J in a different notebook/experiment MHz*)
Jmax = 80;
resJ = 40;
Jarr = Table[jval, {jval, 0, Jmax, Jmax / (resJ - 1)}];

(*Resolution of frequency scanned (x-axis) MHz*)
resω = 80;
(*Define the range of scanned frequencies (x-axis) MHz*)
(*initial freq MHz*)
ωin = 2 π 1800;
(*final freq*)
ωfin = 2 π 2600;
(*array of freqs. scanned over, MHz*)
ωarr = Table[t, {t, ωin, ωfin, (ωfin - ωin) / (resω - 1)}];


(* Theta is the SO-angle. In an accurate and complete microscopic description,
his accounts for the angle by which the spin is rotated due to SOI,
i.e. the angle in the rotation matrix R. To avoid over-
 fitting this angle is set to 0 as we consider an effective J,
J_eff between the two qubits *)
θso = 0;
(* Phases of the drives. *)
φ1 = 0;
```

```
φ2 = 0;

(* Populations:
 These are the states that are blocked in PSB and which we initialize in. *)
ψ0a = {1, 0, 0, 0};
ψ0b = {0, 0, 0, 1};

(*NOT USED HERE: this block was used for testing and developing.*)
resTg = 100;
(*initial time*)
Tg0 = 0;
(*final time, 66.6ns*)
Tgf = 3 * 1 / 30;
Tgarr = Table[t, {t, Tg0, Tgf, (Tgf - Tg0)/(resTg - 1)}];


(*SO-vector*)
{nx, ny, nz} = {1, 0, 0};

(* Effective exchange interaction*)
(* Relevant component parallel with respect to external magnetic field B.*)
Jpar[J_] = 2 Pi J (nz^2 + (1 - nz^2) Cos[θso]);
(*note: λx00=2 Pi 30; (*Russ/Burkard resonance condition for driving*)*)

(*2-Qubit Hamiltonian*)
HQ[ωz1in_, ωz2in_] = KroneckerProduct[PauliMatrix[3] ωz1in/2, PauliMatrix[0]] +
    KroneckerProduct[PauliMatrix[0], PauliMatrix[3] ωz2in/2];
Rso = {
    {nx^2 + (1 - nx^2) Cos[θso],
     nx ny - nx ny Cos[θso] - nz Sin[θso], nx nz - nx nz Cos[θso] + ny Sin[θso]},
    {nx ny - nx ny Cos[θso] + nz Sin[θso], ny^2 + (1 - ny^2) Cos[θso],
     ny nz - ny nz Cos[θso] - nx Sin[θso]}, {nx nz - nx nz Cos[θso] - ny Sin[θso],
     2 Sin[θso/2] (nx Cos[θso/2] + ny nz Sin[θso/2]), nz^2 + (1 - nz^2) Cos[θso]}
   };

(*Exchange Hamiltonian*)
HJ[J_] =
  J/4 Sum[KroneckerProduct[PauliMatrix[i], (Rso.Array[PauliMatrix, 3])[[i]]], {i, 1, 3}];
```

```
(*turn-on time of pulses*)
dt0 = 0.001;
(*wait time between pulses*)
Tw = 0.001;

(* driving amplitudes, here normalised to 2 Pi and equal to each other. One could choose
 to e.g. drive one qubit harder than the other one. ( this could lead to over-fitting)*)
λx10 = 2 Pi; (* Drive amplitude Q1*)
λx20 = 2 Pi; (* Drive amplitude Q2*)

(*First pulse for a time Tflip1. This amplitude
 is an error function turning on/off the pulse.*)
λx1[t_] = λx10 (1/2 + 1/2 Erf[(Tflip1 - t) / dt0]);
(*second pulse after a time Tflip1+Tw*)
λx2[t_] = λx20 (1/2 + 1/2 Erf[(t - Tflip1 - Tw) / dt0]);

(*Drive Hamiltonian*)
Hλ[t_] = λx1[t] Cos[ωx1 t + φ1]
    (fR1 KroneckerProduct[{nx, ny, nz}.Array[PauliMatrix, 3], PauliMatrix[0]] + fR2
       KroneckerProduct[PauliMatrix[0], {nx, ny, nz}.Array[PauliMatrix, 3]]) + λx2[t] Cos[
     ωx2 t + φ2] (fR1 KroneckerProduct[{nx, ny, nz}.Array[PauliMatrix, 3], PauliMatrix[0]] +
      fR2 KroneckerProduct[PauliMatrix[0], {nx, ny, nz}.Array[PauliMatrix, 3]]);

(*Total Hamiltonian*)
Htot[t_, J_, ωz1in_, ωz2in_] = HQ[ωz1in, ωz2in] + HJ[J] + Hλ[t];
ρSf =
  Function[{ψ0, ψro, Tflip1in, Tflip2in, ωz1in, ωz2in, ωx1in, ωx2in, fR1in, fR2in, Jin}, sub =
    {Tflip1 → Tflip1in, Tflip2 → Tflip2in, ωx1 → ωx1in, ωx2 → ωx2in, fR1 → fR1in, fR2 → fR2in};
   Tfin = Tflip1 + Tw + Tflip2 /. sub;
   ψfin[t_] = {a[t], b[t], c[t], d[t]} /. NDSolve[
       {Htot[t, Jin, ωz1in, ωz2in].{a[t], b[t], c[t], d[t]} ==
          ⅈ D[{a[t], b[t], c[t], d[t]}, t] /. sub,
        a[0] == ψ0[[1]], b[0] == ψ0[[2]], c[0] == ψ0[[3]], d[0] == ψ0[[4]]},
       {a[t], b[t], c[t], d[t]}, {t, 0, Tfin},
       Method → "PDEDiscretization" → {"MethodOfLines",
          "SpatialDiscretization" → "FiniteElement", MaxCellMeasure → 0.0005 / fR1}][[1]];
   (*readout signal*)
   Sum[Abs[(ψro[[i]].ψfin[Tfin])]^2, {i, 1, Dimensions[ψro][[1]]}]];
```

```
(* Search for the optimal fR1 and fR2 *)

For[fR1t = 30, fR1t > 20, fR1t += -2,
  For[fR2t = 19, fR2t > 9, fR2t += -2,
   (*average Chevrons over Larmor*)
   (*Resolution in MW frequency, MHz*)
   resω = 80; (*160*)
ωin = 2 π 1800; (*initial freq*)
   ωfin = 2 π 2600; (*final freq*)
   ωarr = Table[t, {t, ωin, ωfin, (ωfin - ωin) / (resω - 1)}];
(*Resolution in burst duration*)
   resT = 40; (*80*)
Tin = 0.0; (* corresponding to 0 ns*)
Tfin = 0.08; (* corresponding to 80 ns *)
Tarr = Table[t, {t, Tin, Tfin, (Tfin - Tin) / (resT - 1)}];
   (*w=const cuts of Chevron*)

   (*block below, already defined above. this was used for checks*)
σωz1 = 25 × 2 Pi;
σωz2 = 25 × 2 Pi;
resωz1 = 150; (*150*)
resωz2 = 150; (*150*)
ωz1arr = Abs[RandomVariate[NormalDistribution[ωz1, σωz1], {resωz1}]];
ωz2arr = Abs[RandomVariate[NormalDistribution[ωz2, σωz2], {resωz2}]];

   (*Run matrix of different chevrons*)
   (* ρSf=Function[
    {ψ0,ψro,Tflip1in,Tflip2in,ωz1in,ωz2in,ωx1in,ωx2in,fR1in,fR2in,Jin}, ... *)

   (* We read out in the 0,1,0,0 and *)
   (* 0,0,1,0 states (anti-parallel) which are not blocked in PSB *)


   Chevron =
    ParallelTable[(ρSf[ψ0a, {{0, 1, 0, 0}, {0, 0, 1, 0}}, Tarr⟦i⟧, 0, ωz1arr⟦k⟧, ωz2arr⟦k⟧,
         ωarr⟦j⟧, 0, fR1t, fR2t, Jpar[Jval]] + ρSf[ψ0b, {{0, 1, 0, 0}, {0, 0, 1, 0}},
         Tarr⟦i⟧, 0, ωz1arr⟦k⟧, ωz2arr⟦k⟧, ωarr⟦j⟧, 0, fR1t, fR2t, Jpar[Jval]]) / 2,
     {k, 1, resωz1}, {j, 1, resω}, {i, 1, resT}];
Export["C:\\Users\\exports\\Chevron_average_Larmor_sigma" <> TextString[σωz1] <>
     "MHz_J" <> TextString[Jval] <> "MHz_fR1t" <> TextString[fR1t] <> "MHz_fR2t" <>
     TextString[fR2t] <> "MHz_vDDMMYY_vX.h5", {Chevron, ωarr / (2 Pi), Tarr, fR1t, fR2t},
    {"Datasets", {"testChevron", "warr", "Tarr", "fR1t", "fR2t"}}]]];
```

```
(* One single value with not-random Larmor ωz1=2094,
ωz1=2284 to show for Fig SX Experiment vs Simulation*)
Jval = 38.7;

(*Resolution in burst duration*)
resT = 40;
(*initial time *)
Tin = 0.0; (*corresponding to 0 ns*)
(* emd time*)
Tfin = 0.08; (* corresponding to 80 ns*)
(*array of times (burst times)*)
Tarr = Table[t, {t, Tin, Tfin, (Tfin - Tin) / (resT - 1)}];

(*we just want one shot at the larmor frequencies, with no random sampling averages*)
resωz1 = 2;
ωz1arr = {ωz1, ωz1};
ωz2arr = {ωz2, ωz2};
Chevron =
  ParallelTable[(ρSf[ψ0a, {{0, 1, 0, 0}, {0, 0, 1, 0}}, Tarr[[i]], 0, ωz1arr[[k]], ωz2arr[[k]],
       ωarr[[j]], 0, fR1t, fR2t, Jpar[Jval]] + ρSf[ψ0b, {{0, 1, 0, 0}, {0, 0, 1, 0}},
       Tarr[[i]], 0, ωz1arr[[k]], ωz2arr[[k]], ωarr[[j]], 0, fR1t, fR2t, Jpar[Jval]]) / 2,
   {k, 1, resωz1}, {j, 1, resω}, {i, 1, resT}];
Export["C:\\Users\\export\\Chevron_NOT_average_Larmor_J" <> TextString[Jval] <>
  "MHz_fR1t" <> TextString[fR1t] <> "MHz_fR2t" <> TextString[fR2t] <> "MHz_vDDMMYY_vX.h5",
 {Chevron, ωarr / (2 Pi), Tarr, fR1t, fR2t},
 {"Datasets", {"testChevron", "warr", "Tarr", "fR1t", "fR2t"}}]
```