Corresponding Manuscript: Conditional Operation of Hole Spin Qubits above 1 K — [Notebook: F2-c-d_gray_Single_tone_EDSR_J38.7MHZ_v040925_v3.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. Human readable output available as PDF.

Figure mapping: This notebook reproduces the gray EDSR trace in the lower two panels of Figure 2 c and d (SIM EDSR F2 c, d)

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.

```
(* Single-tone spectroscopy, Parameters *)

(* f_Rabi, MHz *)

(* We assume this to be the bare Rabi frequency of the two ideally isolated
 spin qubits. Sometimes also referred to as bare Rabi frequency, for J=0 *)
fR1t = 24;
fR2t = 13;
(* Effective exchnage interaction J, MHz *)
Jval = 38.7;
(* Larmor frequencies, 2*Pi*MHz *)
ωz1 = 2 Pi 2075;
ωz2 = 2 Pi 2270;

(* duration of the MW burst in \mus *)
Tburst = 0.020; (*corresponding to 20 ns*)

(*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}]];

(*Resolution of frequency scanned (x-axis) MHz*)
resω = 160;
(*Define the range of scanned frequencies (x-axis) MHz*)
(*initial freq MHz*)
ωin = 2 π 1800;
(*final freq MHz*)
ωfin = 2 π 2600;
(*array of freqs. scanned over, MHz*)
ωarr = Table[t, {t, ωin, ωfin, (ωfin - ωin) / (resω - 1)}];



(* Main calculation core *)

(* 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};


(* 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]);

(* 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*)

(* Generic case has two tones. Further below we only use one. *)
```

```
(* 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]]}]];
```

```
(* Single-tone experiment *)

(* We read out in the 0,1,0,0 and *)
(* 0,0,1,0 states (anti-parallel) which are not blocked in PSB *)
(*NOTE: Below, we only apply Tburst to one of the two drive
 variables since this simulation of EDSR is a single tone experiment. *)

resF = ParallelTable[{j, ωarr[[i]] / (2 π 10^3),
      (ρSf[ψ0a, {{0, 1, 0, 0}, {0, 0, 1, 0}}, 0, Tburst, ωz1arr[[j]], ωz2arr[[j]], 0,
          ωarr[[i]], fR1t, fR2t, Jpar[Jval]] + ρSf[ψ0b, {{0, 1, 0, 0}, {0, 0, 1, 0}}, 0,
          Tburst, ωz1arr[[j]], ωz2arr[[j]], 0, ωarr[[i]], fR1t, fR2t, Jpar[Jval]]) / 2},
     {i, 1, resω}, {j, 1, resωz1}]; // AbsoluteTiming
Export["C:\\Users\\exports\\single_tone_average_Larmor_sigma" <>
   TextString[σωz1] <> "MHz_J" <> TextString[Jval] <> "MHz_vDDMMYY_vX.h5",
   {resF, ωz1arr/(2 Pi), ωz2arr/(2 Pi), ωarr/(2 Pi)}, {"Datasets", {"resF", "fR1arr", "fR2arr", "farr"}}];
```

Out[•]=

```
{50.7474, Null}
```