Superconducting qubits Hub

VQD setup

Set the main directory as the current directory

In[172]:=

```
SetDirectory[NotebookDirectory[]];
```

Load the QuESTLink package
One may also use the off-line questlink.m file, change it to the location of the local file

In[173]:=

```
Import["https://qtechtheory.org/questlink.m"]
```

This will download a binary file **quest_link** if there is no such file found
Otherwise, use a locally-compiled that called **quest_link***

In[174]:=

```
(* Search for existing files that match the pattern "quest_link*" *)
With[{questLinkFiles = Sort@FileNames["quest_link*", {NotebookDirectory[]}]}
 ,
 If[Length[questLinkFiles] > 0,
   (* If one or more matching files are found,
   use the first one alphabetically *)
   Print["Using the existing link file: ", First@questLinkFiles];
   CreateLocalQuESTEnv[First@questLinkFiles];
   ,
   (*If no matching files are found, download the link file*)
   Print["No link file found, download quest_link"];
   CreateDownloadedQuESTEnv[];
  ];
]
```

```
Using the existing link file: /home/cica/VQD/devices/quest_link
```

Load the **VQD** package; must be loaded after QuESTlink is loaded

In[175]:=

```
Get["../vqd.wl"]
```

Set the default configuration of the virtual

superconducting device

frequency unit: ***MHz***
time unit: ***μs***

In[176]:=

```
Options[SuperconductingHub] = {
   (* The number of qubits should match
     all assignments. Qubits are numbered from 0 to N-1 *)
   QubitNum → 6
   ,
   (* The T1 time *)
   T1 → <|0 → 63, 1 → 93, 2 → 109, 3 → 115, 4 → 68, 5 → 125|>
   ,
   (* The T2 time with Hahn echo applied *)
   T2 → <|0 → 113, 1 → 149, 2 → 185, 3 → 161, 4 → 122, 5 → 200|>
   ,
   (* Excited population probability in the initialisation,
   also the thermal state *)
   ExcitedInit → <|0 → 0.032,
     1 → 0.021, 2 → 0.008, 3 → 0.009, 4 → 0.025, 5 → 0.007|>
   ,
   (* Qubit frequency of each qubit  *)
   QubitFreq → <|0 → 4500, 1 → 4900, 2 → 4700, 3 → 5100, 4 → 4900, 5 → 5300|>
   ,
   (* Exchange coupling strength of the resonators on each edge. Use [Esc]o-
    o[Esc] for the edge notation *)
   ExchangeCoupling → <|0 ⟷ 1 → 4, 0 ⟷ 2 → 1.5,
     1 ⟷ 3 → 1.5, 2 ⟷ 3 → 4, 2 ⟷ 4 → 1.5, 3 ⟷ 5 → 1.5, 4 ⟷ 5 → 4|>
   ,
   (* Transmon Anharmonicity *)
   Anharmonicity → <|0 → 296.7,
     1 → 298.6, 2 → 297.4, 3 → 298.3, 4 → 297.2, 5 → 299.1|>
   ,
   (* Fidelity of qubit readout *)
   FidRead → <|0 → 0.9, 1 → 0.92, 2 → 0.96, 3 → 0.97, 4 → 0.93, 5 → 0.97|>
   ,
   (* Measurement duration. It is done without quantum amplifiers *)
   DurMeas → 5
   ,
   (* Duration of the Rx and Ry gates are the
    same regardless the angle. Rz is virtual and perfect. *)
   DurRxRy → 0.05
   ,
```

```
(* Duration of the cross resonance ZX gate that is fixed regardless
 the angle. The error is sourced from the passive noise only. *)
DurZX → 0.5
,
(* Duration of the siZZle gate is fixed regardless the angle that is fixed
 regardless the angle. The error is sourced from the passive noise only. *)
DurZZ → 0.5
,
(* switches to turn on/off standard passive noise, i.e., T1 and T2 decay *)
StdPassiveNoise → True
,
(* switches to turn on/off the cross-talk ZZ-noise *)
ZZPassiveNoise → True
};
```

Elementary guide

Native gates

Initialisation and readout

$\text{Init}_{0,1,\ldots,n}$, M_q

Single-qubit gates, $\theta \in [-\pi, \pi]$

$\text{Rx}_q[\theta]$, $\text{Ry}_q[\theta]$, $\text{Rz}_q[\theta]$

Two-qubit gates: siZZler and cross-resonant gates

$\text{ZZ}_{\text{q1,q2}}$, $\text{ZX}_{\text{q1,q2}}$

others: doing nothing

Wait_q[duration]

Instantiate the VQD and show connectivity

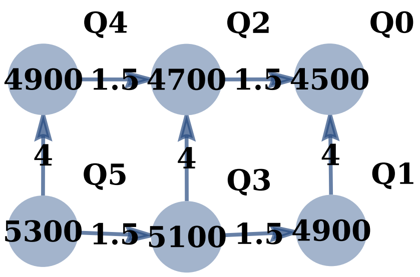
In[177]:=

```
dev = SuperconductingHub[];
```

In[178]:=

```
g = dev[Connectivity]
```

Out[178]=



Passive noise test

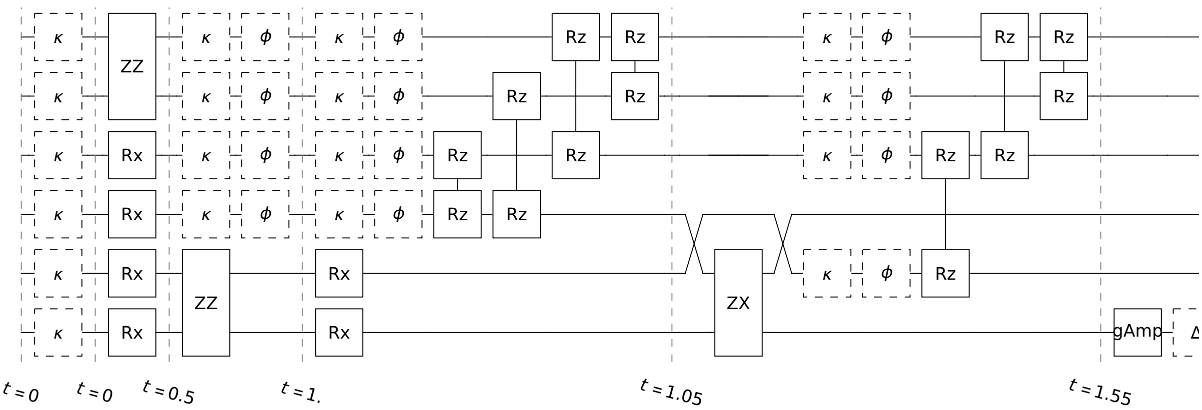
In[179]:=

```
noisycirc =
  InsertCircuitNoise[
   {Init˛ИıИ˒И˒И˓И˓, Rx˛[π], Rxı[π], Rx˒[π/2],
    ZZıИ˛, Rx˛[π/4], Rxı[π], Rx˒[π], ZX˒И˛, ZZ˓И˓, M˛, Wait˛[10]},
   SuperconductingHub[], ReplaceAliases → False];
```

In[180]:=

```
DrawCircuit[noisycirc]
```

Out[180]=



State initialisation to the thermal state

In[181]:=

```
DestroyAllQuregs[];
ρinit = CreateDensityQureg[6];
ρ = CreateDensityQureg[6];
```

The population prepared state should be in the mixture $\rho\_thermal = p|0X0|+(1-p)|1X1|$, where p is specified in **ExcitedInit.**
This is done by applying **Init** operator to each qubit which is done only in the very beginning.

In[184]:=

```
Values@OptionValue[SuperconductingHub, ExcitedInit]
```

Out[184]=

```
{0.032, 0.021, 0.008, 0.009, 0.025, 0.007}
```

In[185]:=

```
(* the init operator in terms of noise *)
noisycirc =
 InsertCircuitNoise[{Init_{0,1,2,3,4,5}}, SuperconductingHub[], ReplaceAliases → True]
```

Out[185]=

```
{{0, {Kraus_0[{{{0.98387, 0.}, {0., 0.}}, {{0., 0.98387}, {0., 0.}},
      {{0., 0.}, {0., 0.178885}}, {{0., 0.}, {0.178885, 0.}}}],
    Kraus_1[{{{0.989444, 0.}, {0., 0.}}, {{0., 0.989444}, {0., 0.}},
      {{0., 0.}, {0., 0.144914}}, {{0., 0.}, {0.144914, 0.}}}],
    Kraus_2[{{{0.995992, 0.}, {0., 0.}}, {{0., 0.995992}, {0., 0.}},
      {{0., 0.}, {0., 0.0894427}}, {{0., 0.}, {0.0894427, 0.}}}],
    Kraus_3[{{{0.99549, 0.}, {0., 0.}}, {{0., 0.99549}, {0., 0.}},
      {{0., 0.}, {0., 0.0948683}}, {{0., 0.}, {0.0948683, 0.}}}],
    Kraus_4[{{{0.987421, 0.}, {0., 0.}}, {{0., 0.987421}, {0., 0.}},
      {{0., 0.}, {0., 0.158114}}, {{0., 0.}, {0.158114, 0.}}}],
    Kraus_5[{{{0.996494, 0.}, {0., 0.}}, {{0., 0.996494}, {0., 0.}},
      {{0., 0.}, {0., 0.083666}}, {{0., 0.}, {0.083666, 0.}}}]}, {}}, {0, {}, {}}}
```

In[186]:=

```
(* initialise matrix as a random mix state of 6 qubits,
then apply the initialisation command *)
SetQuregMatrix[ρinit, RandomMixState[6]];
```

In[187]:=

```
(* apply the noisy circuit, then check the diagonal of the density matrix *)
ApplyCircuit[ρinit, ExtractCircuit @ noisycirc];
Diagonal @ Chop@ Re @ GetQuregMatrix[ρinit]
```

Out[188]=

```
{0.901981, 0.0298175, 0.0193479, 0.0006396, 0.00727404, 0.000240464, 0.000156031,
 5.15806 × 10^-6, 0.00819155, 0.000270795, 0.000175713, 5.80868 × 10^-6,
 0.0000660609, 2.18383 × 10^-6, 1.41704 × 10^-6, 4.68442 × 10^-8, 0.0231277,
 0.000764552, 0.0004961, 0.0000164, 0.000186514, 6.16575 × 10^-6, 4.00081 × 10^-6,
 1.32258 × 10^-7, 0.00021004, 6.94346 × 10^-6, 4.50545 × 10^-6, 1.4894 × 10^-7,
 1.69387 × 10^-6, 5.59957 × 10^-8, 3.63343 × 10^-8, 1.20113 × 10^-9, 0.00635837,
 0.000210194, 0.00013639, 4.50876 × 10^-6, 0.0000512772, 1.69511 × 10^-6,
 1.09992 × 10^-6, 3.6361 × 10^-8, 0.0000577451, 1.90893 × 10^-6, 1.23866 × 10^-6,
 4.09474 × 10^-8, 4.65686 × 10^-7, 1.53946 × 10^-8, 9.98918 × 10^-9, 3.30221 × 10^-10,
 0.000163035, 5.38959 × 10^-6, 3.49718 × 10^-6, 1.15609 × 10^-7, 1.3148 × 10^-6,
 4.34645 × 10^-8, 2.82031 × 10^-8, 9.32333 × 10^-10, 1.48064 × 10^-6, 4.89469 × 10^-8,
 3.17605 × 10^-8, 1.04993 × 10^-9, 1.19407 × 10^-8, 3.94733 × 10^-10, 2.56133 × 10^-10, 0}
```

In[189]:=

```
(* Sanity check: the diagonals should be as follows *)
Reverse @ Chop @ Diagonal[KroneckerProduct @@ ({{#, 0}, {0, 1 - #}} & /@
    (Reverse @ Values @ OptionValue[SuperconductingHub, ExcitedInit]))]
```

Out[189]=

```
{0.901981, 0.0298175, 0.0193479, 0.0006396, 0.00727404, 0.000240464, 0.000156031,
 5.15806 × 10^-6, 0.00819155, 0.000270795, 0.000175713, 5.80868 × 10^-6,
 0.0000660609, 2.18383 × 10^-6, 1.41704 × 10^-6, 4.68442 × 10^-8, 0.0231277,
 0.000764552, 0.0004961, 0.0000164, 0.000186514, 6.16575 × 10^-6, 4.00081 × 10^-6,
 1.32258 × 10^-7, 0.00021004, 6.94346 × 10^-6, 4.50545 × 10^-6, 1.4894 × 10^-7,
 1.69387 × 10^-6, 5.59957 × 10^-8, 3.63343 × 10^-8, 1.20113 × 10^-9, 0.00635837,
 0.000210194, 0.00013639, 4.50876 × 10^-6, 0.0000512772, 1.69511 × 10^-6,
 1.09992 × 10^-6, 3.6361 × 10^-8, 0.0000577451, 1.90893 × 10^-6, 1.23866 × 10^-6,
 4.09474 × 10^-8, 4.65686 × 10^-7, 1.53946 × 10^-8, 9.98918 × 10^-9, 3.30221 × 10^-10,
 0.000163035, 5.38959 × 10^-6, 3.49718 × 10^-6, 1.15609 × 10^-7, 1.3148 × 10^-6,
 4.34645 × 10^-8, 2.82031 × 10^-8, 9.32333 × 10^-10, 1.48064 × 10^-6, 4.89469 × 10^-8,
 3.17605 × 10^-8, 1.04993 × 10^-9, 1.19407 × 10^-8, 3.94733 × 10^-10, 2.56133 × 10^-10, 0}
```

Thermal state (ρinit) can be prepared by waiting

In[190]:=

```
noisycirc = InsertCircuitNoise[{ Wait#[1] & /@ Range[0, 5]},
  SuperconductingHub[], ReplaceAliases → True]
```

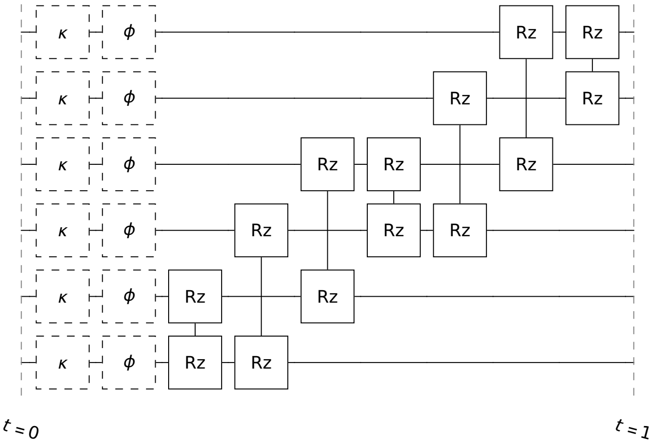
Out[190]=

```
{{0, {Kraus0[{{{0.98387, 0.}, {0., 0.976092}}, {{0., 0.123466}, {0., 0.}},
     {{0.177471, 0.}, {0., 0.178885}}, {{0., 0.}, {0.0224483, 0.}}}], Deph0[0.00440526],
   Kraus1[{{{0.989444, 0.}, {0., 0.984139}}, {{0., 0.102325}, {0., 0.}},
     {{0.144137, 0.}, {0., 0.144914}}, {{0., 0.}, {0.0149866, 0.}}}],
   Deph1[0.00334447], R[0.131923, Z0 Z1],
   Kraus2[{{{0.995992, 0.}, {0., 0.991434}}, {{0., 0.0951803}, {0., 0.}},
     {{0.0890334, 0.}, {0., 0.0894427}}, {{0., 0.}, {0.00854745, 0.}}}],
   Deph2[0.00269541], R[-0.0277977, Z0 Z2],
   Kraus3[{{{0.99549, 0.}, {0., 0.991171}}, {{0., 0.0926285}, {0., 0.}},
     {{0.0944568, 0.}, {0., 0.0948683}}, {{0., 0.}, {0.00882732, 0.}}}],
   Deph3[0.00309597], R[-0.0273321, Z1 Z3], R[0.133003, Z2 Z3],
   Kraus4[{{{0.987421, 0.}, {0., 0.980187}}, {{0., 0.119303}, {0., 0.}},
     {{0.156956, 0.}, {0., 0.158114}}, {{0., 0.}, {0.0191038, 0.}}}],
   Deph4[0.00408161], R[-0.0276241, Z2 Z4],
   Kraus5[{{{0.996494, 0.}, {0., 0.992516}}, {{0., 0.0889512}, {0., 0.}},
     {{0.083332, 0.}, {0., 0.083666}}, {{0., 0.}, {0.00746837, 0.}}}],
   Deph5[0.00249376], R[-0.0274045, Z3 Z5], R[0.132693, Z4 Z5]}, {}}, {1, {}, {}}}
```

In[191]:=

```
DrawCircuit[noisycirc]
```

Out[191]=



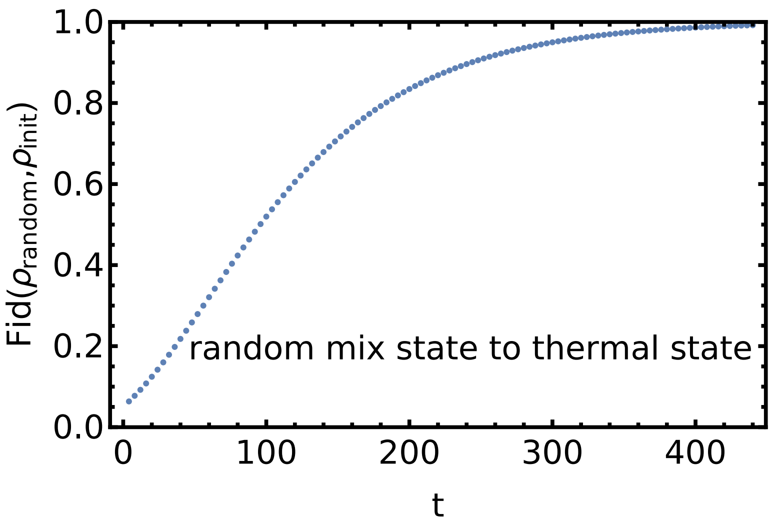
In[192]:=

```
(* wait for t, then check fidelity to the thermal state ρinit *)
δt = 4;
SetQuregMatrix[ρ, RandomMixState[6]];
data = Table[
    ApplyCircuit[ρ, ExtractCircuit @ InsertCircuitNoise[
        Wait#[δt] & /@ Range[0, 5], SuperconductingHub[], ReplaceAliases → True]];
    {t, CalcFidelityDensityMatrices[ρ, ρinit]}
    , {t, δt, 440, δt}];
```

In[195]:=

```
ListPlot[data,
 PlotRange → {Automatic, {0, 1}}, Frame → True, FrameLabel → {"t", "Fid(ρrandom,ρinit)"},
 FrameStyle → Directive[Black, Thick], ImageSize → 400, BaseStyle → {17},
 Epilog → Inset["random mix state to thermal state", Scaled[{0.55, 0.2}]]
]
```

Out[195]=



Free induction decay: T1 experiment

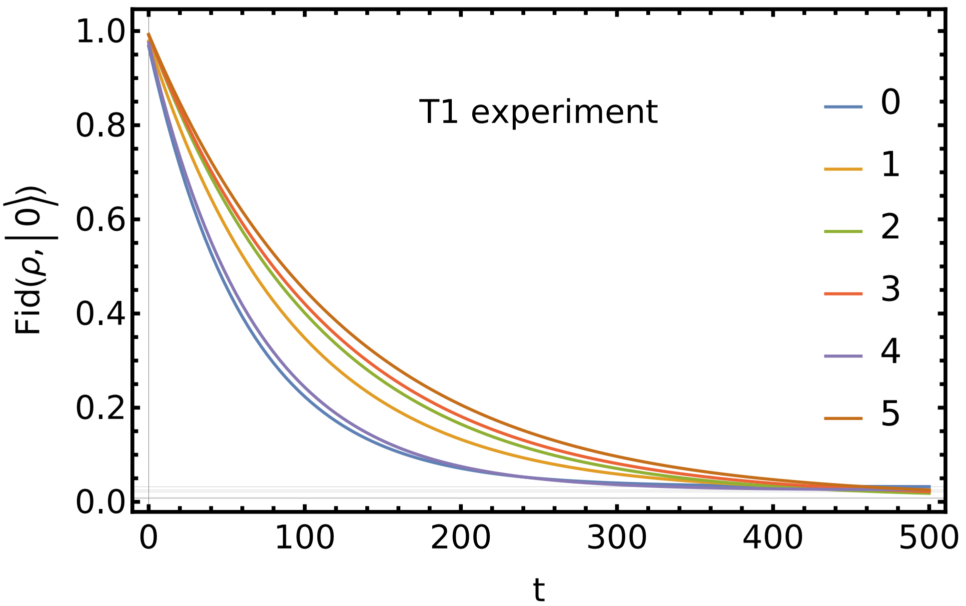
In[196]:=

```
δt = 4;
SetQuregMatrix[ρ, RandomMixState[6]];
dataT1 = Table[
    dev = SuperconductingHub[];
    ApplyCircuit[ρ, ExtractCircuit @
      InsertCircuitNoise[{Init_{0,1,2,3,4,5}}, dev, ReplaceAliases → True]];
    ApplyCircuit[ρ, ExtractCircuit @
      InsertCircuitNoise[Rx_#[π] & /@ Range[0, 5], dev, ReplaceAliases → True]];
    ApplyCircuit[ρ, ExtractCircuit @
      InsertCircuitNoise[Wait_#[t] & /@ Range[0, 5], dev, ReplaceAliases → True]];
    {t, CalcProbOfOutcome[ρ, #, 1]} & /@ Range[0, 5]
    , {t, 0, 500, δt}];
```

In[199]:=

```
(* expected probability at T → ∞, denoted by grey lines *)
expectedprob = Values@OptionValue[SuperconductingHub, ExcitedInit];
ListPlot[Transpose[dataT1],
  Frame → True, PlotLegends → Placed[Range[0, 5], {0.9, 0.5}],
  GridLines → {expectedprob}, Frame → True, FrameLabel → {"t", "Fid(ρ,|0⟩)"},
  FrameStyle → Directive[Black, Thick], ImageSize → 500, BaseStyle → {17},
  Epilog → Inset["T1 experiment", Scaled[{0.5, 0.8}]], Joined → True
]
```

Out[200]=



Free induction decay: T2 experiment

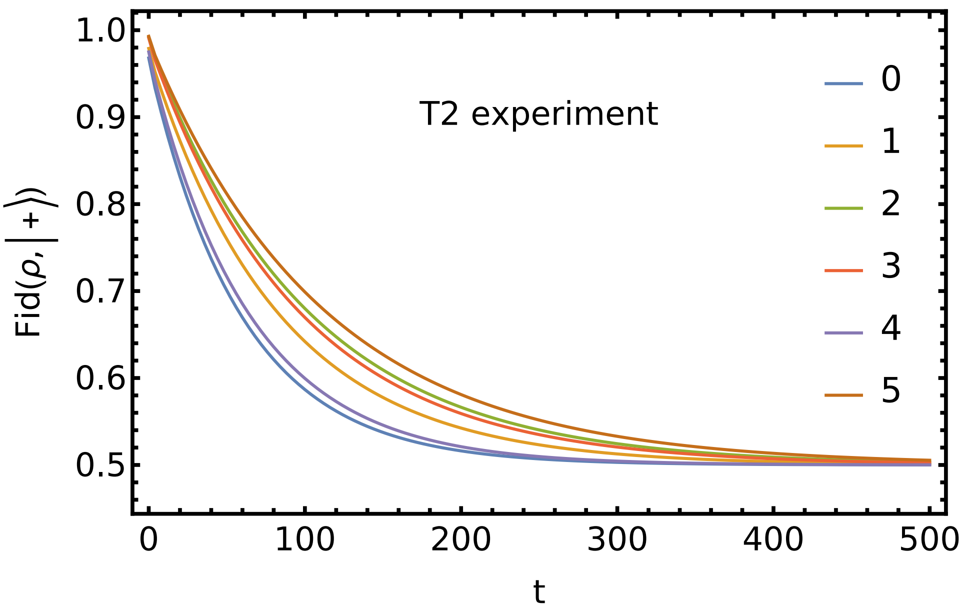
In[201]:=

```
δt = 4;
SetQuregMatrix[ρ, RandomMixState[6]];
dataT2 = Table[
    dev = SuperconductingHub[];
    ApplyCircuit[ρ, ExtractCircuit @
      InsertCircuitNoise[{Init0,1,2,3,4,5}, dev, ReplaceAliases → True]];
    ApplyCircuit[ρ, ExtractCircuit @ InsertCircuitNoise[
       Flatten[Ry#[π / 2] & /@ Range[0, 5]], dev, ReplaceAliases → True]];
    ApplyCircuit[ρ, ExtractCircuit @
      InsertCircuitNoise[Wait#[t] & /@ Range[0, 5], dev, ReplaceAliases → True]];
    ApplyCircuit[ρ, ExtractCircuit @ InsertCircuitNoise[
       Flatten[Ry#[-π / 2] & /@ Range[0, 5]], dev, ReplaceAliases → True]];
    {t, CalcProbOfOutcome[ρ, #, 0]} & /@ Range[0, 5]
    , {t, 0, 500, δt}];
```

In[204]:=

```
ListPlot[Transpose[dataT2], PlotLegends → Placed[Range[0, 5], {0.9, 0.55}],
 PlotRange → All, Frame → True, FrameStyle → Directive[Black, Thick],
 ImageSize → 500, FrameLabel → {"t", "Fid(ρ,|+⟩)"}, BaseStyle → {17},
 Epilog → Inset["T2 experiment", Scaled[{0.5, 0.8}]], Joined → True]
```

Out[204]=



Paper supplement: VQE of H_2 on the superconducting

qubit

Modules

In[205]:=

```
(* The virtual device works only for θ ϵ [-π, π] *)
angleToMinusPiToPi[angle_] := Mod[angle + π, 2 π] - π
```

Load the pre-run data

In[206]:=

```
(* exact ground state energies *)
gsH2 << "supplement/VQEonSuperconductingHub/gsH2.mx";
(* noiseless *)
vqeH20 << "supplement/VQEonSuperconductingHub/run1/vqeH20.mx";
 (*realistic noise *)
vqeH21 << "supplement/VQEonSuperconductingHub/run1/vqeH21.mx";
(* static noise only *)
vqeH22 << "supplement/VQEonSuperconductingHub/run1/vqeH22.mx";
```

In[210]:=

```
data = Join[
   {Values@gsH2[[All, {"distance", "groundstate"}]]},
    Values @ #[[All, {"distance", "cost"}]] & /@ {vqeH20, vqeH21, vqeH22}
  ];
```

In[211]:=

```
colors = {■, ■, ■, ■};
```

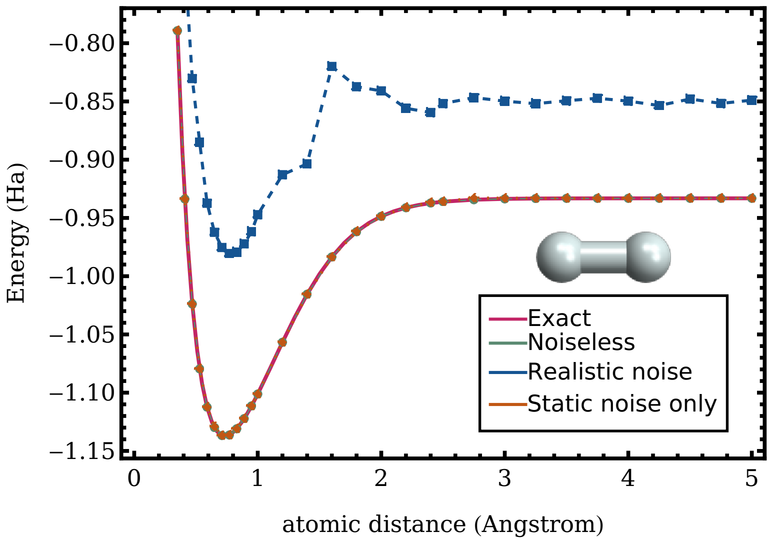
In[216]:=

```
(* molecule image *)
H2 = ImageResize[MoleculePlot3D[
     Molecule[ConstantArray["H", 2], Bond[{#, # + 1}, "Single"] & /@ Range[1],
      AtomCoordinates → ({.8 * #, 0, 0} & /@ Range[0, 1])]], 140];
```

In[213]:=

```
Show[
 ListPlot[Values/@gsH2[[All, {"distance", "groundstate"}]],
  Joined → True, PlotStyle → Directive[colors[[1]], Thickness → Scaled[0.006]],
  BaseStyle → {11, FontFamily → "Serif"}, Epilog → Inset[H2, Scaled[{0.8, 0.92}]]],
 ListPlot[Values@#[[All, {"distance", "cost"}]] &/@{vqeH20, vqeH21, vqeH22},
  PlotMarkers → {Automatic, 5}, PlotStyle → {Directive[colors[[2]]
     , Dashed, Thickness → Scaled[0.002]],
    Directive[colors[[3]], Dashed], Directive[colors[[4]], Dotted]}, Joined → True],
 Frame → True, FrameStyle → Directive[Black, Thick], Background → White
 , Epilog →
  Inset[Column[{H2, LineLegend[colors, {"Exact", "Noiseless", "Realistic noise",
        "Static noise only"}, Spacings → 0., LegendFunction → Framed,
      LegendMargins → 0]}, Alignment → Center], Scaled[{0.75, 0.28}]],
 BaseStyle → {12, FontFamily → "Serif"}, FrameLabel →
  {"atomic distance (Angstrom)", "Energy (Ha)"},
 ImageSize → 400, AspectRatio → 0.7,
 ImagePadding → {{60, 5}, {45, 5}}
]
(*Export["vqeh2.pdf",%]*)
```

Out[213]=



In[214]:=

```
yticks = {{10^-10, "10^-10"}, {10^-8, "10^-8"}, {10^-6, "10^-6"},
   {10^-6, "10^-6"}, {10^-4, "10^-4"}, {0.0015, "chem"}, {0.1, "0.1"}};
ListLogPlot[
 {
  Transpose @
   {vqeH20[[All, "distance"]], vqeH20[[All, "cost"]] - vqeH20[[All, "groundstate"]]},
  Transpose @
   {vqeH21[[All, "distance"]], vqeH21[[All, "cost"]] - vqeH21[[All, "groundstate"]]},
  Transpose @
   {vqeH22[[All, "distance"]], vqeH22[[All, "cost"]] - vqeH21[[All, "groundstate"]]}},
 Frame → True, FrameStyle → Directive[Black, Thick],
 Background → White, PlotRange → All,
 GridLines → {None, {0.0015}}, GridLinesStyle → Directive[Thick, Dashed, Red],
 FrameTicks → {{yticks, Automatic}, {Automatic, Automatic}},
 FrameLabel → {None, "accuracy (Ha)"}, PlotLegends →
  PointLegend[Automatic, {"Noiseless", "Realistic noise", "Static noise only"},
   LegendMargins → 0, LegendMarkerSize → 15],
 ImageSize → 400, AspectRatio → 0.4, LabelStyle → {11, FontFamily → "Serif"},
 ImagePadding → {{60, 5}, {15, 5}}, PlotStyle → colors[[2 ;;]]
]
(*Export["vqeh2err.pdf",%]*)
```

Out[215]=