In[1]:=
```
Import["https://quest.qtechtheory.org/QuEST.m"]
CreateDownloadedQuESTEnv[];
```

H_2 Hamiltonian in BK representation, reduced

In[3]:=
```
hamil = .2252 + .3435 Z0 - .4347 Z1 + .5716 Z0 Z1 + .091 Y0 Y1 + .091 X0 X1;

ground = Min @ Eigenvalues @ CalcPauliSumMatrix @ hamil
```

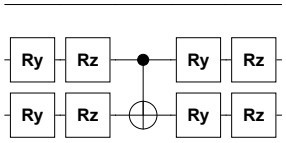
Out[4]=
```
-1.1456
```

A universal ansatz

In[5]:=
```
numQb = 3;
numParams = 8;
ancilla = 2;
ansatz = Circuit[
       Ry1[θ1] Ry0[θ3] Rz1[θ2] Rz0[θ4] C1[X0] Ry1[θ5] Ry0[θ7] Rz1[θ6] Rz0[θ8]];

DrawCircuit[ansatz, numQb]
```

Out[9]=



Li derivative circuits

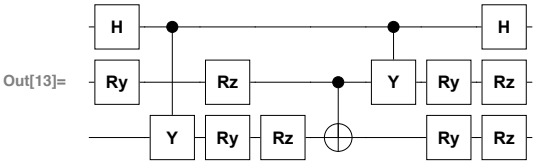
In[10]:=
```
insertGateDeriv[circ_, p_] :=
     circ /.
   {g : Ryqb_[θp] :> {Cancilla[Yqb], g}, g : Rzqb_[θp] :> {Cancilla[Zqb], g}} // Flatten

getCircuitA[p1_, p2_] :=
     Join[{Hancilla}, insertGateDeriv[insertGateDeriv[ansatz, p1], p2], {Hancilla}]

getCircuitC[p_, h_] :=
     Join[
          {Hancilla, Rzancilla[-π/2], Xancilla}, insertGateDeriv[ansatz, p],
          {Xancilla}, If[h == 1, {}, Cancilla /@ (List @@ hamil[[h]])[[2 ;;]]],
          {Hancilla}]
```

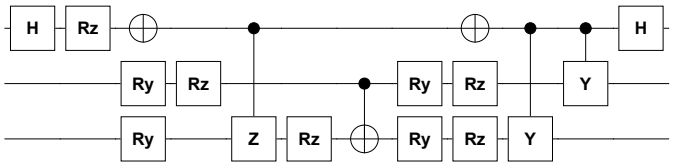
In[13]:=
```
DrawCircuit[getCircuitA[3, 5], numQb]
```

Out[13]=

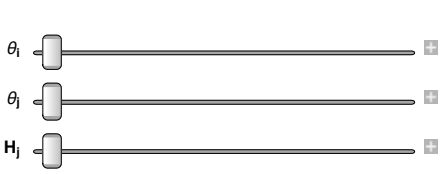


In[14]:=
```
DrawCircuit[getCircuitC[4, 3], numQb]
```

Out[14]=

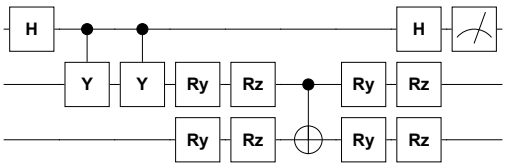


In[15]:=
```
Manipulate[Column@{
    Row[{Style[⟨ "∂ψ"/"∂" θp1 | "∂ψ"/"∂" θp2 ⟩, Blue], ": ",
   DrawCircuit[getCircuitA[p1, p2] ~Append~ Mancilla, numQb]}],
    Row[{Style[Row[{"⟨ψ|H"h, | "∂ψ"/"∂θ"p2 ⟩}]], Red], ": ",
   DrawCircuit[getCircuitC[p2, h] ~Append~ Mancilla, numQb]}]},
   {{p1, 1, θi}, 1, numParams, 1}, {{p2, 1, θj}, 1, numParams, 1},
 {{h, 1, Hj}, 1, Length[hamil], 1}, Paneled → False]
```

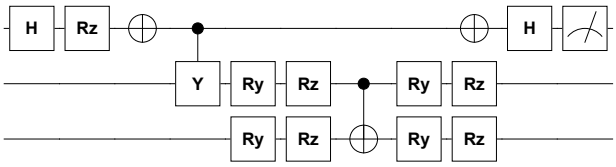


Out[15]=

$\left\langle \frac{\partial\psi}{\partial\theta_1} \middle| \frac{\partial\psi}{\partial\theta_1} \right\rangle$:



$\left\langle \psi \middle| H_1 \middle| \frac{\partial\psi}{\partial\theta_1} \right\rangle$:

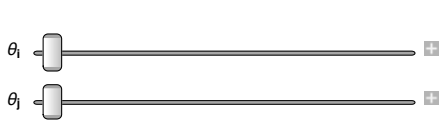


A circuits need not apply gates after entangling the ancilla

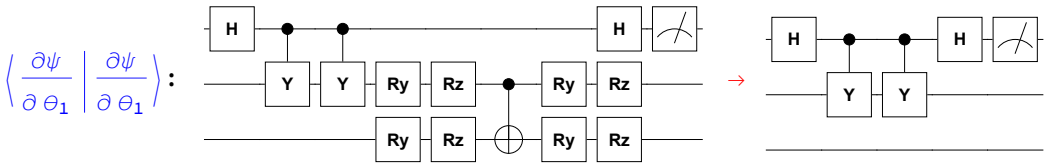
In[16]:=
```
optimizeA[circ_] :=
    ReplaceRepeated[
        circ[[ ;; Position[circ, Cancilla[_]][[-1, 1]]],
        {a___, g1 : Except[C]qb1_[_],
     g2 : (Cancilla[_qb2_] | PatternSequence[_qb2_[_], Cancilla[_qb2_]])} /;
    (qb1 ≠ qb2) :> {a, g2}
     ] ~Append~ Hancilla
```

In[17]:=
```
Manipulate[
    Row @ {Style[⟨ "∂ψ"/"∂" θp1 | "∂ψ"/"∂" θp2 ⟩, Blue], ": ",

  DrawCircuit[getCircuitA[p1, p2] ~Append~ Mancilla, numQb], Style[" → ", Red],
    DrawCircuit[optimizeA@getCircuitA[p1, p2] ~Append~ Mancilla, numQb]},
  {{p1, 1, θi}, 1, numParams, 1}, {{p2, 1, θj}, 1, numParams, 1}, Paneled → False]
```



Out[17]=



Direct simulation of experimental routine

In[18]:=
```
ψ = CreateQureg[3];
ϕ = CreateQureg[3];

getMatrixA[params_] := 1/4 Table[
        ApplyCircuit[ optimizeA@getCircuitA[r, c] /. params, InitZeroState[ψ]];
        2 CalcProbOfOutcome[ψ, ancilla, 0] - 1,
        {r, numParams}, {c, numParams}]

getVectorC[params_] := 1/2 Table[
        Sum[
            ApplyCircuit[getCircuitC[p, h] /. params, InitZeroState[ψ]];
            If[h == 1, hamil[[h]], hamil[[h, 1]]] ×
              (2 CalcProbOfOutcome[ψ, ancilla, 0] - 1),
            {h, Length[hamil]}],
        {p, numParams}]
```

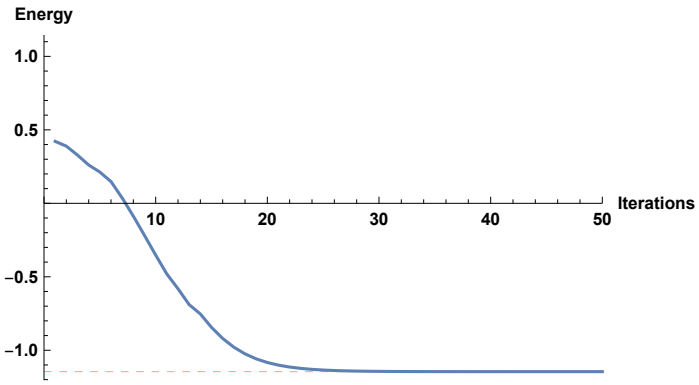
In[22]:=
```
directImagTime[θi_, Δt_, steps_] := Module[
      {params, Δparams, matrA, vecC, energy, energies},
      params = θi;
      energies = {};
      Do[
           matrA = getMatrixA[params];
           vecC = getVectorC[params];
           Δparams = Δt LinearSolve[matrA, vecC, Method → "Multifrontal"];
           params = MapIndexed[θ#2⟦1⟧ → #1 &, params⟦All, 2⟧ + Δparams];

           ApplyCircuit[ansatz /. params, InitZeroState[ψ]];
           energy = CalcExpecPauliSum[ψ, hamil, ϕ];
           AppendTo[energies, energy];

           plotDirect = ListLinePlot[
                 energies,
                 GridLines → {{}, {{ground, Directive[Red, Dashed]}}},
                 PlotRange → {{0, steps}, {1.05 ground, -ground}},
                 AxesLabel → {"Iterations", "Energy"}
             ],
           steps
       ]
 ]
```

In[23]:=
```
Dynamic[plotDirect]
directImagTime[Table[θp → RandomReal[{0, 2 π}], {p, numParams}], .1, 50]
```

Out[23]=



Optimised simulation

In[25]:=
```
dψ = CreateQuregs[numQb, numParams];
hψ = CreateQureg[numQb];
```

In[27]:=
```
optimisedImagTime[θi_, Δt_, steps_] := Module[
      {params, Δparams, matrA, vecC, energy, energies},
      params = θi;
      energies = {};
      Do[
            CalcQuregDerivs[ansatz, InitZeroState[ψ], params, dψ];
            matrA = CalcInnerProducts[dψ] // Re;

            ApplyCircuit[ansatz /. params, InitZeroState[ψ]];
            ApplyPauliSum[ψ, hamil, hψ];
            vecC = -CalcInnerProducts[hψ, dψ] // Re;

            Δparams = Δt LinearSolve[matrA, vecC, Method → "Multifrontal"];
            params = MapIndexed[θ#2⟦1⟧ → #1 &, params⟦All, 2⟧ + Δparams];

            energy = CalcExpecPauliSum[ψ, hamil, ϕ];
            AppendTo[energies, energy],
            steps];
      energies]

optimisedRepeatImagTime[Δt_, steps_, reps_] := Module[
      {evo, evos},
      evos = {};
      Do[
            evo = optimisedImagTime[
     Table[θp → RandomReal[{0, 2 π}], {p, numParams}], Δt, steps];
            AppendTo[evos, evo];
            plotOptimised = ListLinePlot[
                  evos,
                  GridLines → {{}, {{ground, Directive[Red, Dashed]}}},
                  PlotRange → {{0, steps}, {1.05 ground, -ground}},
                  AxesLabel → {"Iterations", "Energy"}
              ],
            reps]]
```

In[29]:=
```
Dynamic[plotOptimised]
optimisedRepeatImagTime[.05, 100, 10]
```

Out[29]=