GetRandomCircuitFromChannel

```
SetDirectory @ NotebookDirectory[];
Import["../Link/QuESTlink.m"] // Quiet;
CreateLocalQuESTEnv["../quest_link"];
```

Doc

```
? GetRandomCircuitFromChannel
```

Symbol

GetRandomCircuitFromChannel[channel] returns a pure, random circuit from
the coherent decomposition of the input channel (a circuit including decoherence),
weighted by its probability. The average of the expected values of the circuits
returned by this function approaches the expected value of the noise channel.
See SampleExpecPauliString[] to sample such circuits in order
to efficiently approximate the effect of decoherence on an expectation value.

Correctness

Decomposition

Unitary mixtures

```
Table[GetRandomCircuitFromChannel[Deph_2[0]], 10]
Table[GetRandomCircuitFromChannel[Deph_2[0.125]], 10]
Table[GetRandomCircuitFromChannel[Deph_2[.25]], 10]
Table[GetRandomCircuitFromChannel[Deph_2[1 / 2]], 10]
```

{{}, {}, {}, {}, {}, {}, {}, {}, {}, {}}

{{}, {}, {}, {}, {}, {}, {Z_2}, {}, {}, {Z_2}}

{{Z_2}, {}, {}, {Z_2}, {}, {}, {Z_2}, {}, {}, {Z_2}}

{{Z_2}, {}, {}, {Z_2}, {Z_2}, {}, {}, {Z_2}, {}, {Z_2}}

```
Table[GetRandomCircuitFromChannel[Deph_{5,6}[0]], 10]
Table[GetRandomCircuitFromChannel[Deph_{5,6}[.25]], 10]
Table[GetRandomCircuitFromChannel[Deph_{5,6}[.5]], 10]
Table[GetRandomCircuitFromChannel[Deph_{5,6}[3 / 4]], 10]
```

```
{{}, {}, {}, {}, {}, {}, {}, {}, {}, {}}
{{}, {}, {}, {Z5}, {}, {Z6}, {}, {}, {Z5}, {}}
{{Z5, Z6}, {}, {}, {Z5}, {}, {Z6}, {Z5}, {Z5, Z6}, {Z5}, {}}
{{Z5}, {}, {Z5}, {Z5, Z6}, {Z5}, {Z6}, {}, {}, {}, {}}
```

```
Table[GetRandomCircuitFromChannel[Depol_2[0]], 10]
Table[GetRandomCircuitFromChannel[Depol_2[0.25]], 10]
Table[GetRandomCircuitFromChannel[Depol_2[.5]], 10]
Table[GetRandomCircuitFromChannel[Depol_2[3 / 4]], 10]
```

```
{{}, {}, {}, {}, {}, {}, {}, {}, {}, {}}
{{}, {}, {}, {Y2}, {}, {Z2}, {}, {X2}, {}, {}}
{{Y2}, {X2}, {}, {Y2}, {X2}, {}, {}, {Z2}, {}, {}}
{{Y2}, {Z2}, {Y2}, {}, {Y2}, {X2}, {X2}, {X2}, {}, {X2}}
```

```
Table[GetRandomCircuitFromChannel[Depol_{0,1}[0]], 10]
Table[GetRandomCircuitFromChannel[Depol_{0,1}[.25]], 10]
Table[GetRandomCircuitFromChannel[Depol_{0,1}[.5]], 10]
Table[GetRandomCircuitFromChannel[Depol_{0,1}[15 / 16]], 10]
```

```
{{}, {}, {}, {}, {}, {}, {}, {}, {}, {}}
{{}, {}, {Y0, Y1}, {}, {}, {}, {Y1}, {Z0, Z1}, {}, {}}
{{Z0}, {X0, Z1}, {}, {}, {Z0, X1}, {Z0, X1}, {X0}, {Z0}, {Z0, Y1}, {}}
{{Z1}, {}, {Y0}, {X0, X1}, {Y0}, {X0, X1}, {X0, Z1}, {X0, Z1}, {Y0, Z1}, {Z0, Z1}}
```

Non-unitary mixtures

```
(* probability is fixed 50/50 regardless of parameter *)
GetRandomCircuitFromChannel[Damp_0[0]]
GetRandomCircuitFromChannel[Damp_0[1]]
```

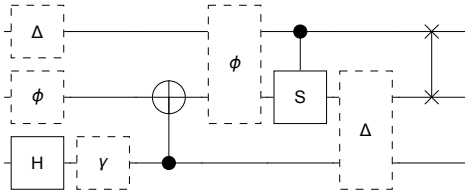
```
{Fac[1.41421], Matr_0[{{1, 0}, {0, 1}}]}
{Fac[1.41421], Matr_0[{{1, 0}, {0, 0}}]}
```

```
k = Table[RandomVariate @ CircularUnitaryMatrixDistribution[2], 3];
GetRandomCircuitFromChannel[Kraus_0[k]]
```

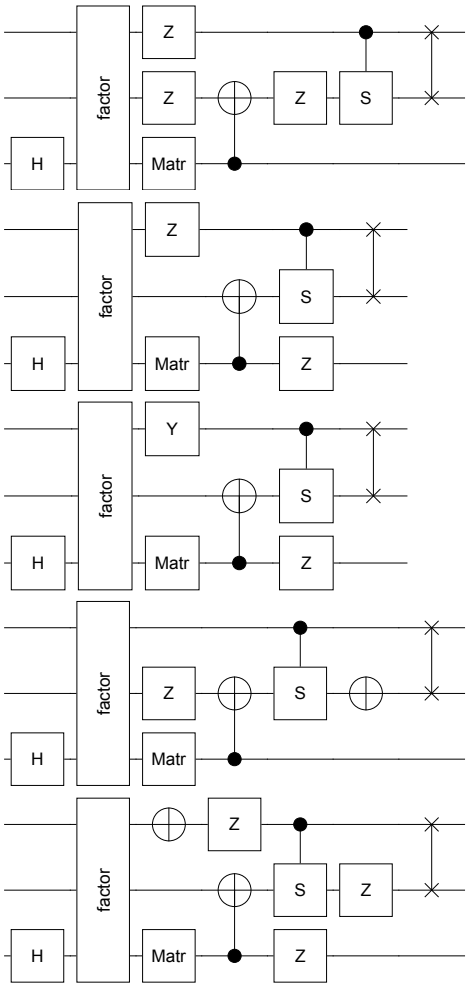
```
{Fac[1.73205], Matr_0[{{0.683258 + 0.454046 i, 0.296674 - 0.488862 i},
   {-0.541799 - 0.182906 i, 0.230265 - 0.787386 i}}]}
```

Circuits

```
u = Circuit[
    H0 Damp0[.1] Deph1[.5] Depol2[.5] C0[X1] Deph1,2[.5] C2[S1] Depol0,1[.7] SWAP1,2];
DrawCircuit[u]
```




```
DrawCircuit /@ Table[GetRandomCircuitFromChannel[u], 5] // Column
```




```
MemberQ[
      GetCircuitsFromChannel[u] /. Fac[_] → Nothing,
      GetRandomCircuitFromChannel[u] /. Fac[_] → Nothing]
```

```
True
```

Expectation values

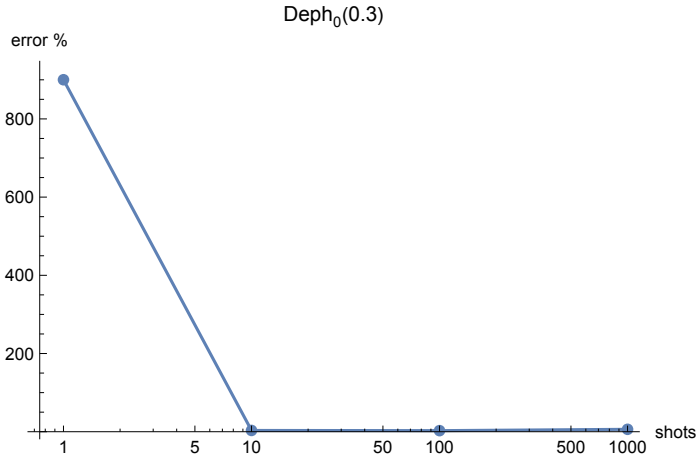
```
n = 3;
{ψ, ψi, ϕ} = CreateQuregs[n, 3];
{ρ, ω} = CreateDensityQuregs[n, 2];

samps = {1, 10, 100, 1000};
```

```
sample[circ_, shots_Integer] := Module[
    {h, u, e1, e2},

    (* randomise initial state and Pauli observable *)
    h = GetRandomPauliString[n];
    SetQuregMatrix[ψi, RandomComplex[{0,1+i}, 2^n]];

    (* compute expec value via channels upon density matrix *)
    InitPureState[ρ, ψi];
    ApplyCircuit[ρ, circ];
    e1 = CalcExpecPauliString[ρ, h, ω];

    (* compute expec value by Monte Carlo sampling statevectors *)
    e2 = Total @ Table[
        u = GetRandomCircuitFromChannel[circ];
        InitPureState[ψ, ψi];
        ApplyCircuit[ψ, u];
        CalcExpecPauliString[ψ, h, ϕ],
        shots] / shots;

    (* return relative error *)
    Abs[e2 - e1]/Abs[e1]
]
```

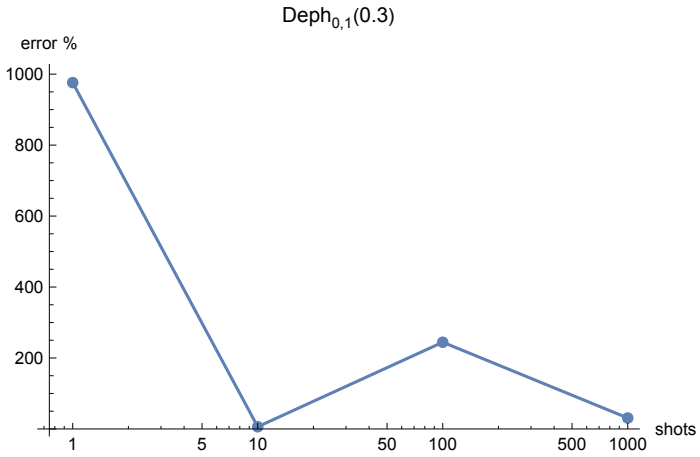
```
plotError[circ_, shots_List, label_:None] :=
    ListLogLinearPlot[
        Table[{s,100 sample[u, s]}, {s, shots}],
        AxesLabel → {"shots", "error %"},
        PlotMarkers → Automatic, Joined→True,
        PlotLabel → If[label===None, circ, label]
    ]
```

Unitary mixtures

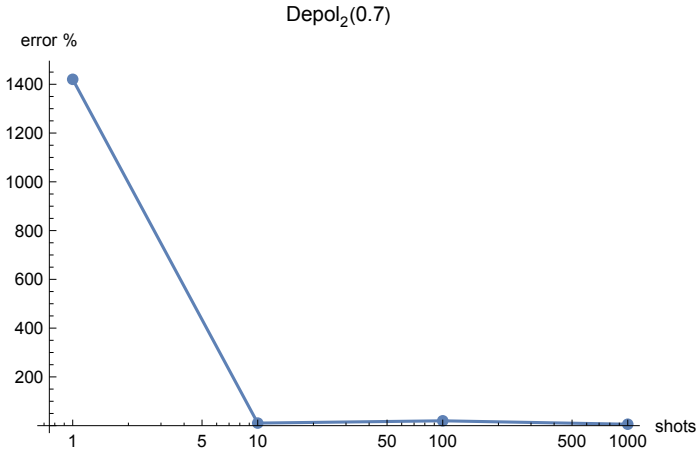
```
plotError[Deph0[.3], samps]
```



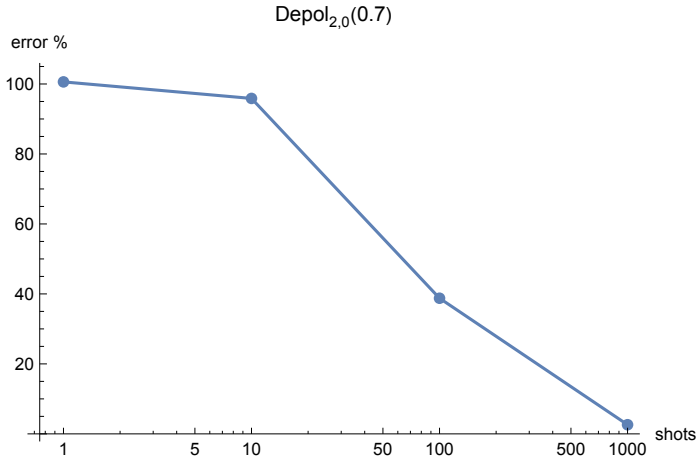

```
plotError[Deph0,1[.3], samps]
```




```
plotError[Depol2[.7], samps]
```

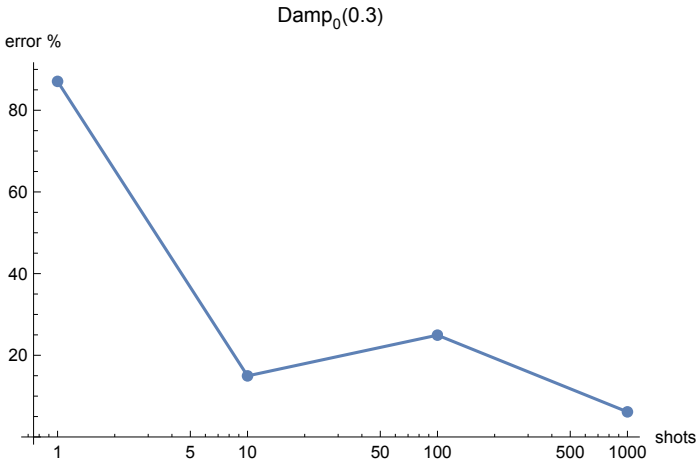


```
plotError[Depol2,0[.7], samps]
```

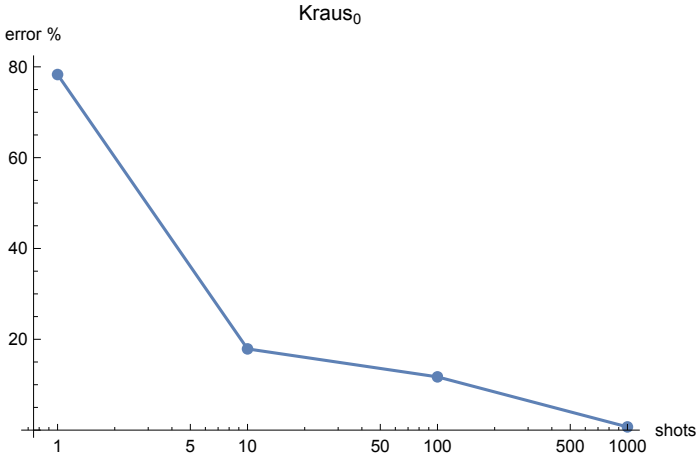


Non-unitary mixtures

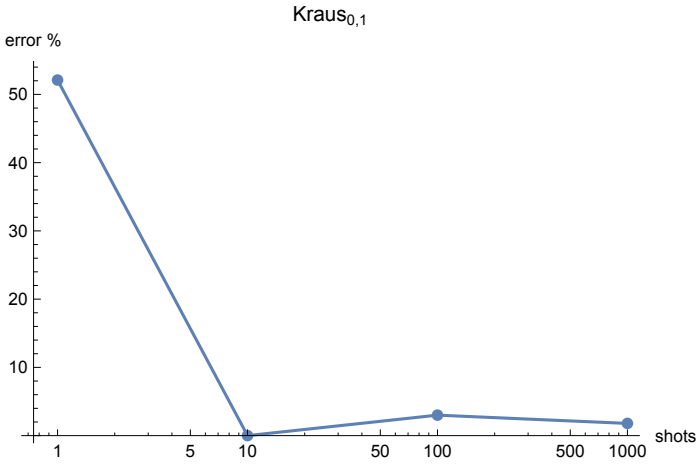
```
plotError[Damp0[.3], samps]
```



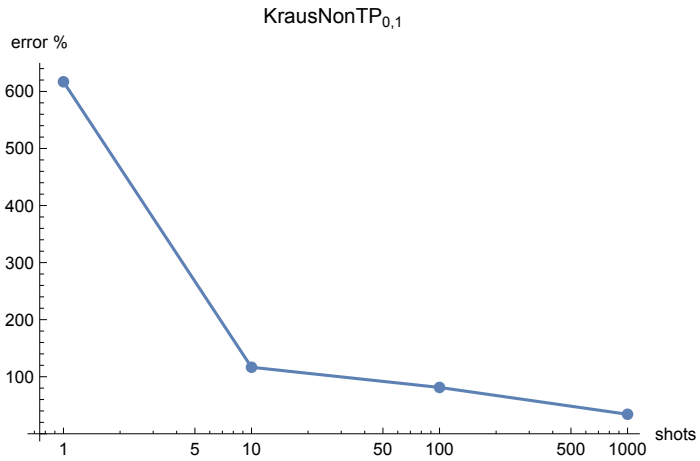

```
k = Table[RandomVariate @ CircularUnitaryMatrixDistribution[2], 3];
plotError[Kraus0[k], samps, Kraus0]
```



```
k = Table[RandomVariate @ CircularUnitaryMatrixDistribution[4], 9];
plotError[Kraus0,1[k], samps, Kraus0,1]
```

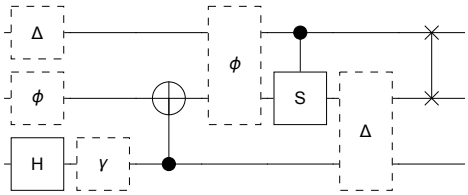



```
k = Table[RandomComplex[{-1 - ⅈ, 1 + ⅈ}, {2^2, 2^2}], 9];
plotError[KrausNonTP0,1[k], samps, KrausNonTP0,1]
```

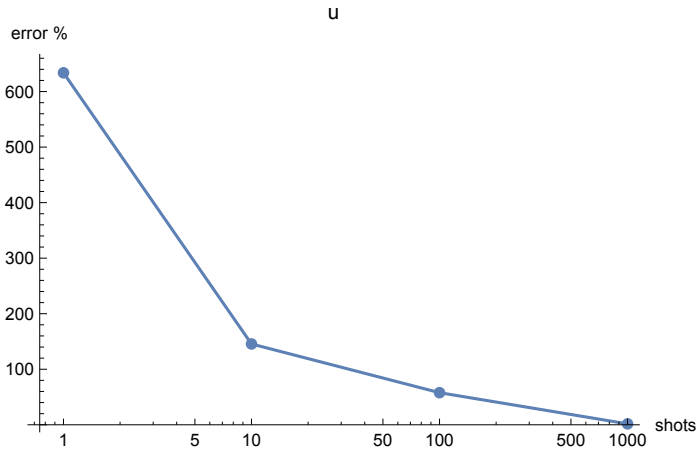


Circuits

```
u = Circuit[
    H0 Damp0[.1] Deph1[.5] Depol2[.5] C0[X1] Deph1,2[.5] C2[S1] Depol0,1[.7] SWAP1,2];
DrawCircuit[u]
```



```
plotError[u, samps, "u"]
```



Errors

```
GetRandomCircuitFromChannel[Damp_0[1.1]]
```

••• GetRandomCircuitFromChannel: A unitary–mixture channel had an invalid probability which was negative or exceeded that causing maximal mixing.

```
$Failed
```

```
GetRandomCircuitFromChannel[Deph_{0,1}[.9]]
```

••• GetRandomCircuitFromChannel: A unitary–mixture channel had an invalid probability which was negative or exceeded that causing maximal mixing.

```
$Failed
```

```
GetRandomCircuitFromChannel[Depol_{0,1}[15 / 16 + .00001]]
```

••• GetRandomCircuitFromChannel: A unitary–mixture channel had an invalid probability which was negative or exceeded that causing maximal mixing.

```
$Failed
```

```
GetRandomCircuitFromChannel[Depol_0[x]]
```

••• GetRandomCircuitFromChannel: The probabilities of a decomposition of a decoherence operator were invalid and/or unnormalised.

```
$Failed
```