Supplementary material
Mathematica codes

Developed by Kira Joel, Davida Kollmar, and Lea F. Santos

Yeshiva University

New York, NY, USA

(If you use these codes, please cite the paper)

Hamiltonian

Description

We write the Hamiltonia matrix of one S_z-subspace.
The Hamiltonian is created using the site-basis.
The eigenvalues and eigenstates of the matrix are computed.

Notation

*) chainsize = number of sites
*) upspins = number of spins pointing up in the z direction
*) dim = dimension of the subspace being studied = $\begin{pmatrix} \text{chainsize} \\ \text{upspins} \end{pmatrix}$
*) Jxy = strength of the flip-flop term between nearest neighbors
*) Jz = strength of the Ising interaction between nearest neighbors
*) basis = all possible configurations of up and down spins in the given subspace. These create the site-basis of the subspace.
They are obtained by permutation of the state where the first sites have spins pointing up and the others have spins pointing down.
*) HH = elements of the Hamiltonian
*) Energy = eigenvalues of the Hamiltonian
*) Vector = eigenstates of the Hamiltonian
*) open = determines whether the chain is open or closed. For closed chain, open=0. For open chain, open=1

Code for eigenvalues and eigenstates

```
(* Parameters of the Hamiltonian *)
Clear[chainsize, upspins, downspins, dim, Jxy, Jz, open];
chainsize = 10;
upspins = chainsize / 2;
downspins = chainsize - upspins;
dim = chainsize! / (upspins! downspins!);
Jxy = 1.0;
Jz = 0.5;
open = 1;
```

```
(* Creating the basis *)
Clear[onebasisvector, basis];
onebasisvector = Flatten[{Table[1, {k, 1, upspins}], Table[0, {k, 1, downspins}]}];
basis = Permutations[onebasisvector];

(* ELEMENTS OF THE HAMILTONIAN *)
(* Initialization *)
Clear[HH];
Do[Do[HH[i, j] = 0., {j, 1, dim}], {i, 1, dim}];

(* Diagonal elements - Ising interaction *)
Do[
  Do[
    HH[i, i] = HH[i, i] + (Jz / 4.) * (-1.)^(basis[[i, k]] + basis[[i, k + 1]]);
    , {k, 1, chainsize - 1}];
  , {i, 1, dim}];
(* Term included in the Ising interaction if the chain is closed *)
If[open == 0, Do[
   HH[i, i] = HH[i, i] + (Jz / 4.) * (-1.)^(basis[[i, chainsize]] + basis[[i, 1]]),
   {i, 1, dim}]];

(* Off-diagonal elements - flip-flop term *)
Clear[howmany, site];
Do[
  Do[
    (* Initialization *)
    howmany = 0;
    Do[site[z] = 0, {z, 1, chainsize}];
    (* Sites where states i and j differ *)
    Do[If[basis[[i, k]] ≠ basis[[j, k]], {howmany = howmany + 1, site[howmany] = k}];,
     {k, 1, chainsize}];
    (* Coupling matrix element - when only two neighbor sites differ *)
    If[howmany == 2,
     If[site[2] - site[1] == 1,
      {HH[i, j] = HH[i, j] + Jxy / 2., HH[j, i] = HH[j, i] + Jxy / 2.}]];
    (* Additional term for closed system *)
    If[open == 0, If[site[2] - site[1] == chainsize - 1,
      {HH[i, j] = HH[i, j] + Jxy / 2., HH[j, i] = HH[j, i] + Jxy / 2.}]];
    , {j, i + 1, dim}];
  , {i, 1, dim - 1}];

(* Hamiltonian *)
Clear[Hamiltonian];
Hamiltonian = Table[Table[HH[i, j], {i, 1, dim}], {j, 1, dim}];
MatrixForm[Hamiltonian];
(* Diagonalization *)
Clear[Energy, Vector];
Energy = Chop[Eigenvalues[Hamiltonian]];
Vector = Chop[Eigenvectors[Hamiltonian]];
```

Histogram

Description

We make histograms for the diagonal elements of the Hamiltonian and for its eigenvalues.
The histograms provide us with information on what to expect for the dynamics of the given system.
The choice of the bin width is arbitrary.

Histogram for diagonal elments: a very small value is a good choice, or instead one can use the analytical expressions provided in the paper.

Histogram for the eigenvalues: have a look at the minimum and maximum values of the eigenvalues before deciding a good value.

NOTE: *Mathematica* has a command to make histograms, but we found it better to create our own procedure.
(The code provided here is used to obtain the top of Figure 1 in the paper).

Notation

*) binsize = width of each bin
*) binedge = the extremes of the bin
*) Nofbins = number of bins
*) Num = how many states contribute to the height of each bin

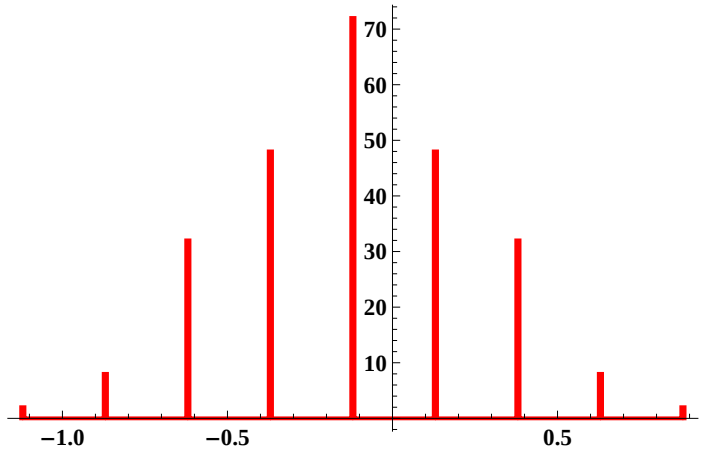
Code for Histogram

Histogram of the diagonal elements of the Hamiltonian

```
(* List of diagonal elements *)
Clear[diagE];
diagE = Sort[Table[HH[i, i], {i, 1, dim}]];

(* Parameters for the histogram *)
Clear[binsize, minDiagE, maxDiagE, Nofbins];
binsize = 0.01;
minDiagE = Min[diagE];
maxDiagE = Max[diagE];
Nofbins = Floor[(maxDiagE - minDiagE) / binsize] + 1;
Clear[binedge];
binedge[1] = minDiagE;
Do[binedge[k] = binedge[k - 1] + binsize, {k, 2, Nofbins + 1}];

(* Number of states in each bin *)
Clear[Num];
Do[Num[k] = 0, {k, 1, Nofbins}];
Do[
  Do[
    If[binedge[k] ≤ HH[j, j] < binedge[k + 1], Num[k] = Num[k] + 1];
    , {k, 1, Nofbins}];
  , {j, 1, dim}];
```

- **Plot of the histogram of the diagonal elements of the Hamiltonian**

```
(* Histogram *)
Clear[histData];
histData = {{binedge[1], 0}};
Do[
  histData = Append[histData, {binedge[k], Num[k]}];
  histData = Append[histData, {binedge[k + 1], Num[k]}];
  , {k, 1, Nofbins}];
histData = Append[histData, {binedge[Nofbins + 1], 0}];
ListPlot[histData, Joined → True, PlotStyle → {Thick, Red},
 LabelStyle → Directive[Black, Bold, Medium], PlotRange → All]
```

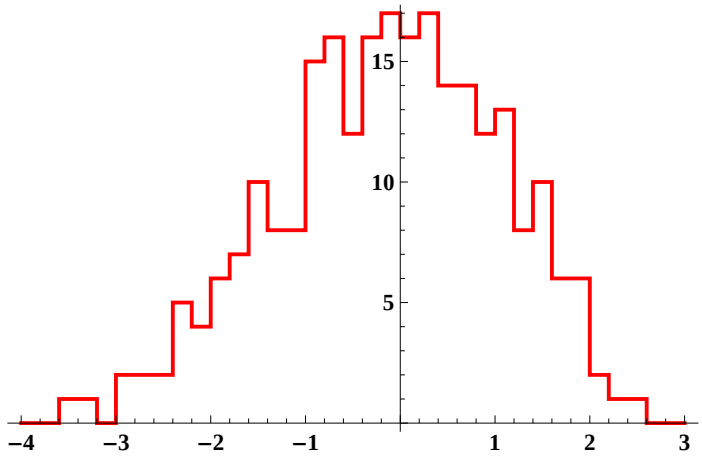


- **Histogram of the eigenvalues**

```
(* Parameters for the histogram *)
Clear[binsize, minE, maxE, Nofbins];
binsize = 0.2;
minE = Floor[Min[Energy]];
maxE = Floor[Max[Energy]] + 1;
Nofbins = IntegerPart[(maxE - minE) / binsize];
Clear[binedge];
binedge[1] = minE;
Do[binedge[k] = binedge[k - 1] + binsize, {k, 2, Nofbins + 1}];

(* Number of states in each bin *)
Clear[Num];
Do[Num[k] = 0, {k, 1, Nofbins}];
Do[
  Do[
    If[binedge[k] ≤ Energy[[j]] < binedge[k + 1], Num[k] = Num[k] + 1];
    , {k, 1, Nofbins}];
  , {j, 1, dim}];
```

- **Plot of the histogram of the eigenvalues**

```
(* Histogram *)
Clear[histData];
histData = {{binedge[1], 0}};
Do[
  histData = Append[histData, {binedge[k], Num[k]}];
  histData = Append[histData, {binedge[k + 1], Num[k]}];
  , {k, 1, Nofbins}];
histData = Append[histData, {binedge[Nofbins + 1], 0}];
ListPlot[histData, Joined → True,
 PlotStyle → {Thick, Red}, LabelStyle → Directive[Black, Bold, Medium]]
```



Average IPR vs Jz

- **Description**

  The Inverse Participation Ratio (IPR) of each eigenstate measures how much spread it is in a particular basis. A high value of IPR means that the eigenstate is spread out in that basis, while a low value of IPR means that the state is more localized.

  Here we choose the site-basis and compute the average value of the IPR's of all eigenstates for each value of Jz. (The code provided here is used to obtain the bottom of Figure 1 in the paper).

- **Notation**

  *) IPR = Inverse Participation Ratio

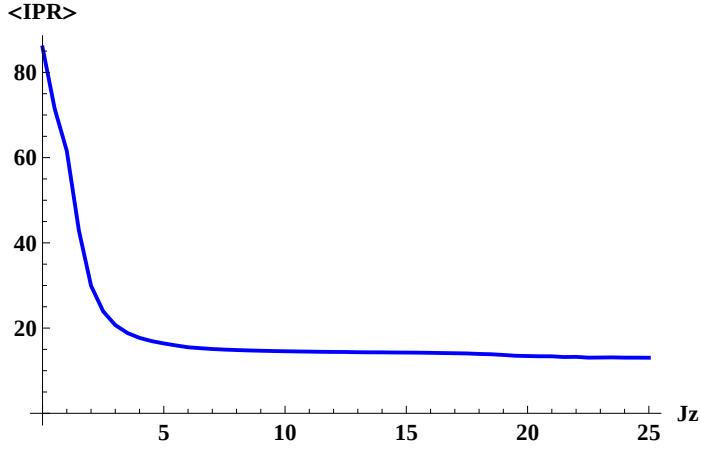
  *) AveIPR = average value of IPR for all eigenstates

- **Code for IPR vs Jz**

```
(* Parameters of the Hamiltonian *)
Clear[chainsize, upspins, downspins, dim, Jxy, open, total];
chainsize = 10;
upspins = chainsize / 2;
downspins = chainsize - upspins;
dim = chainsize! / (upspins! downspins!);
Jxy = 1.0;
open = 1;
```

```
total = 51;

(* Loop for values of Jz *)
Do[
  Jz = 0.5 (kk - 1);

  (* ELEMENTS OF THE HAMILTONIAN *)
(* Initialization *)
  Clear[HH];
  Do[Do[HH[i, j] = 0., {j, 1, dim}], {i, 1, dim}];

  (* Diagonal elements - Ising interaction *)
  Do[
   Do[
     HH[i, i] = HH[i, i] + (Jz / 4.) * (-1.)^(basis[[i, k]] + basis[[i, k + 1]]);
     , {k, 1, chainsize - 1}];
   , {i, 1, dim}];
(* Term included in the Ising interaction if the chain is closed *)
If[open == 0, Do[
    HH[i, i] = HH[i, i] + (Jz / 4.) * (-1.)^(basis[[i, chainsize]] + basis[[i, 1]]),
    {i, 1, dim}]];

  (* Off-diagonal elements - flip-flop term *)
  Clear[howmany, site];
  Do[
   Do[
     (* Initialization *)
     howmany = 0;
     Do[site[z] = 0, {z, 1, chainsize}];
     (* Sites where states i and j differ *)
     Do[If[basis[[i, k]] ≠ basis[[j, k]], {howmany = howmany + 1, site[howmany] = k}];,
      {k, 1, chainsize}];
     (* Coupling matrix element - when only two neighbor sites differ *)
     If[howmany == 2,
      If[site[2] - site[1] == 1,
       {HH[i, j] = HH[i, j] + Jxy / 2., HH[j, i] = HH[j, i] + Jxy / 2.}]];
    (* Additional term for closed system *)
     If[open == 0, If[site[2] - site[1] == chainsize - 1,
       {HH[i, j] = HH[i, j] + Jxy / 2., HH[j, i] = HH[j, i] + Jxy / 2.}]];
     , {j, i + 1, dim}];
   , {i, 1, dim - 1}];

  (* Hamiltonian *)
  Clear[Hamiltonian];
  Hamiltonian = Table[Table[HH[i, j], {i, 1, dim}], {j, 1, dim}];
  (* Diagonalization *)
  Clear[Energy, Vector];
  Energy = Chop[Eigenvalues[Hamiltonian]];
  Vector = Chop[Eigenvectors[Hamiltonian]];

  (* Inverse Participation Ratio: IPR *)
```

```
  Clear[IPR];
  IPR = 0.0;
  Do[
   Clear[denom];
   denom = Sum[Vector[[i, k]]^4, {k, 1, dim}];
   IPR = IPR + 1 / denom;
   , {i, 1, dim}];

  (* Average value of IPR *)
  AveIPR[kk] = IPR / dim;
  Print[{Jz, AveIPR[kk]}];
  , {kk, 1, total}];

(* Plot: IPR vs Jz *)
Clear[tab];
tab = Table[{0.5 (kk - 1), AveIPR[kk]}, {kk, 1, total}];
ListPlot[tab, Joined → True, PlotRange → All, PlotStyle → {Thick, Blue},
 LabelStyle → Directive[Black, Bold, Medium], AxesLabel → {"Jz", "<IPR>"}]
```



Dynamics

- **Description**

We study the time evolution of the system for a particular initial state corresponding to one of the basis vectors.
We plot the magnetization of each site, with a magnetization of 0.5 corresponding to an upspin, and a magnetization of -0.5 corresponding to a downspin.
We also plot the probability of finding each of the basis vectors is time.
(The code provided here is used to obtain Figure 2 in the paper).

- **Parameters explained**

*) initialstate =initial state corresponding to one of the site-basis vectors
*) endtime = how many times the loop for time is repeated (choice depends on the system)
*) increment = dt = discrete interval of time (choice depends on the system)
*) Psi[t] = evolved state
*) Magsite = magnetization of each site

Code for the Dynamics

Hamiltonian, eigenvalues and eigenstates

```
(* Parameters of the Hamiltonian *)
Clear[chainsize, upspins, downspins, dim, Jxy, Jz, open];
chainsize = 6;
upspins = 1;
downspins = chainsize - upspins;
dim = chainsize! / (upspins! downspins!);
Jxy = 1.0;
Jz = 0.5;
open = 1;

(* COPY AND PASTE HERE THE REST OF THE "Code for eigenvalues and eigenstates" *)
```

Choose an initial state

```
Do[
  If[basis[[i, 1]] == 1 && basis[[i, 2]] == 0, initialstate = i];
  , {i, 1, dim}];
```

Dynamics

```
(* DYNAMICS *)
Clear[endtime, PSI, increment, Magsite];
endtime = 81;
increment = .1;
Do[PSI[t] = Sum[Vector[[j, initialstate]]
     Vector[[j]] Exp[-I Energy[[j]] (t - 1) increment], {j, 1, dim}];

  (* Magnetization *)
  Do[
   Magsite[j, t] = 0.5 Sum[Abs[PSI[t][[i]]]^2 (-1.)^(1 + basis[[i, j]]), {i, dim}];
   , {j, 1, chainsize}];

  (* Print[{(t-1)increment,Magsite[1,t]}]; *)
  , {t, 1, endtime}];
```

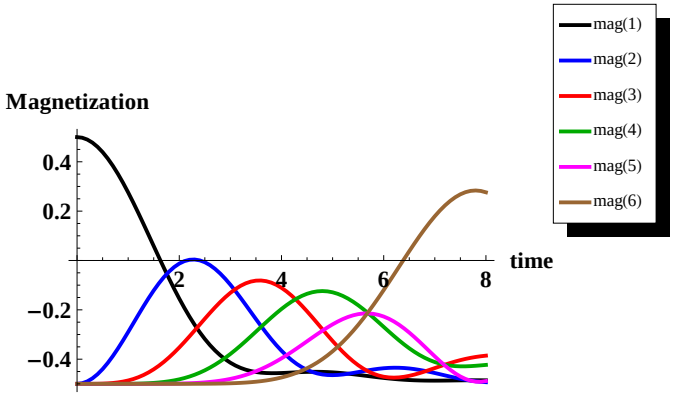
Call Package for Legends

```
<< PlotLegends`;
```

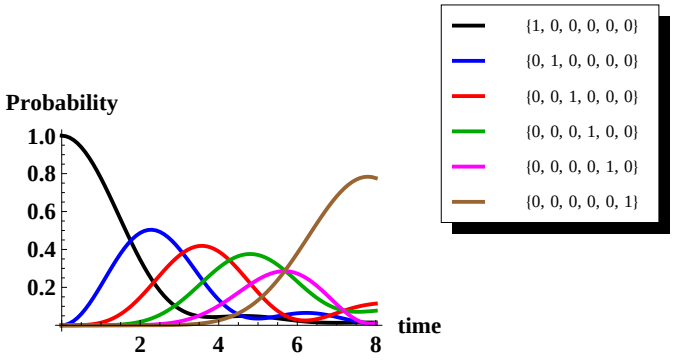
■ Plots

```
Clear[magT, ProbT];
Do[magT[j] = Table[{(t - 1) increment, Magsite[j, t]}, {t, 1, endtime}],
  {j, 1, chainsize}];
Do[ProbT[k] = Table[{(t - 1) increment, Abs[PSI[t][[k]]^2}, {t, 1, endtime}],
  {k, 1, dim}];

Print[];
Print["Magnetization of each site"];
ListPlot[Table[magT[j], {j, 1, chainsize}], PlotRange → All,
 Joined → True, PlotStyle → {{Thick, Black}, {Thick, Blue},
   {Thick, Red}, {Thick, Darker[Green]}, {Thick, Magenta}, {Thick, Brown} },
 LabelStyle → Directive[Black, Bold, Medium], PlotLegend → Table["mag" [x], {x, 1, dim}],
 LegendPosition → {1, 0}, AxesLabel → {"time", "Magnetization"}]

Print[];
Print["Probability of each site-basis"];
ListPlot[Table[ProbT[k], {k, 1, dim}], PlotRange → All, Joined → True,
 PlotStyle → {{Thick, Black}, {Thick, Blue}, {Thick, Red}, {Thick, Darker[Green]},
   {Thick, Magenta}, {Thick, Brown} }, LabelStyle → Directive[Black, Bold, Medium],
 PlotLegend → Table[basis[[k]], {k, 1, dim}], LegendPosition → {1, 0},
 LegendSize → {1, 1}, AxesLabel → {"time", "Probability"}]
```

Magnetization of each site



Probability of each site-basis



Symmetries - Table

- **Description**

  We study the parity and rotation symmetry of each eigenstate and use it to determine which ones can take part in the evolution of particular initial states.
  (The code provided here is used to find the values for Table II in the paper).

- **Parameters explained**

  *) VectorSort=eigenstates sorted out from lowest to highest energy
  *) mirror = finds pairs of site-basis vectors where one is the reflection of the other
  *) parity = eigenvalue of the parity operator for each eigenstate
  *) rot = finds pairs of site-basis vectors where one is the π rotation around x of the other
  *) rotationX = eigenvalue of the global rotation operator around x for each eigenstate
  *) psi0 = chosen initial state (it has well defined parity, rotation X, or both)
  *) psi0EigSt = initial state written in the basis of eigenstates of the Hamiltonian

- **Code to determine the eigenvalues of parity and of a global π-rotation around x**

- **Hamiltonian**

```
(* Parameters of the Hamiltonian *)
Clear[chainsize, upspins, downspins, dim, Jxy, Jz, open];
chainsize = 6;
upspins = 3;
downspins = chainsize - upspins;
dim = chainsize! / (upspins! downspins!);
Jxy = 1.0;
Jz = 0.4;
open = 1;

(* COPY AND PASTE HERE THE REST OF THE "Code for eigenvalues and eigenstates" *)
```

- **Sort the eigenstates from lowest to highest energy**

```
Clear[tab, EnergySort, VectorSort];
tab = Sort[Table[{Energy[[k]], k}, {k, 1, dim}]];
EnergySort = Table[tab[[k, 1]], {k, 1, dim}];
VectorSort = Table[Vector[[tab[[k, 2]]]], {k, 1, dim}];
```

■ Eigenvalues of Parity

```
(* First we find all pairs of basis vectors where one is the
 reflection of the other. Example: state 1100 pairs with state 0011 *)
Clear[pair];
Do[
  Do[Clear[mirror];
    mirror = 0;
    Do[mirror = mirror + Mod[basis[[i, k]] + basis[[j, chainsize + 1 - k]], 2];
     , {k, 1, chainsize}];
    If[mirror == 0, pair[i] = j];
    , {j, 1, dim}];
  , {i, 1, dim}];

(* Now we separate the eigenstates according to their parity, even=+1 or odd=-1 *)
Clear[parity];
Do[
  Clear[project];
  project = 0;
  Do[
   project = project + (1 / 4.) (VectorSort[[i]][[j]] + VectorSort[[i]][[pair[j]]])^2;
   , {j, 1, dim}];
  (* project will be very close to 1 if the parity is even *)
  (* project will be very close to 0 if the parity is odd *)
  If[0.9 < project < 1.1, parity[i] = +1, parity[i] = -1];
  , {i, 1, dim}];
```

■ Eigenvalues of Rx

```
(* First we find all pairs of basis vectors where one is the π rotation
 around x of the other. Example: state 0110 pairs with state 1001 *)
Clear[pair];
Do[
  Do[Clear[rot];
    rot = 0;
    Do[rot = rot + Mod[basis[[i, k]] + basis[[j, k]], 2];
     , {k, 1, chainsize}];
    If[rot == chainsize, pair[i] = j];
    , {j, 1, dim}];
  , {i, 1, dim}];

(* Now we separate the eigenstates according to their eigenvalue of Rx,
Rx=+1 or Rx=-1 *)
Clear[rotationX];
Do[
  Clear[project];
  project = 0;
  Do[
   project = project + (1 / 4.) (VectorSort[[i]][[j]] + VectorSort[[i]][[pair[j]]])^2;
   , {j, 1, dim}];
  (* project will be very close to 1 if Rx=+1 *)
  (* project will be very close to 0 if Rx=-1 *)
  If[0.9 < project < 1.1, rotationX[i] = +1, rotationX[i] = -1];
  , {i, 1, dim}];
```

- **Table with the eigenvalues of parity and Rx of each eigenstate (from lowest to highest energy)**

```
Clear[tab, symbPar, symbRot];
Do[If[parity[k] == -1, symbPar[k] = "-1", symbPar[k] = "+1"], {k, 1, dim}];
Do[If[rotationX[k] == -1, symbRot[k] = "-1", symbRot[k] = "+1"], {k, 1, dim}];
tab = Table[{k, symbPar[k], symbRot[k]}, {k, 1, dim}];
TableForm[ tab, TableHeadings -> {None, { "Eigenstate", "Parity", "Rx(π)"} } ]
```

Eigenstate	Parity	Rx(π)
1	−1	−1
2	+1	+1
3	−1	+1
4	−1	−1
5	+1	+1
6	+1	−1
7	+1	+1
8	−1	+1
9	−1	−1
10	−1	−1
11	+1	+1
12	+1	−1
13	+1	+1
14	−1	−1
15	−1	−1
16	−1	+1
17	+1	−1
18	+1	+1
19	−1	−1
20	+1	+1

- **Probability amplitudes of each eigenstate (sorted from lowest to highest energy) for a chosen initial state.**
  INITIAL STATE A: (011100 + 001110)/sqrt(2)
  Parity=+1, RotationX=0

```
basis[[11]]
basis[[17]]
```

```
{0, 1, 1, 1, 0, 0}
```

```
{0, 0, 1, 1, 1, 0}
```

```
Clear[psi0, psi0EigenStates];
psi0 = (1 / Sqrt[2.]) {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0} +
    (1 / Sqrt[2.]) {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0};

psi0EigenStates = Chop[(Conjugate[VectorSort].psi0) ]
```

```
{0, -0.190202, 0, 0, 0.19095, -0.479483, -0.0541377, 0, 0, 0, -0.501372,
 -0.146691, -0.335202, 0, 0, 0, 0.498575, 0.0367174, 0, -0.243615}
```

- **Probability amplitudes of each eigenstate (sorted from lowest to highest energy) for a chosen initial state.**
  INITIAL STATE D: (011100 - 001110+100011-110001)/2
  Parity=-1, RotationX=+1

```
basis[[11]]
basis[[17]]
basis[[10]]
basis[[4]]
```

```
{0, 1, 1, 1, 0, 0}

{0, 0, 1, 1, 1, 0}

{1, 0, 0, 0, 1, 1}

{1, 1, 0, 0, 0, 1}
```

```
Clear[psi0, psi0EigenStates];
psi0 = (1 / 2.) {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0} -
    (1 / 2.) {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0} +
    (1 / 2.) {0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0} -
    (1 / 2.) {0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

psi0EigenStates = Chop[(Conjugate[VectorSort].psi0) ]
```

```
{0, 0, 0.465234, 0, 0, 0, 0, -0.399734, 0, 0, 0, 0, 0, 0, 0, -0.789791, 0, 0, 0, 0}
```