Gaussian Orthogonal Ensemble (GOE)

Below we show how to obtain from a GOE
(1) its matrices;
(2) the density of states;
(3) the number of principal components (NPC) of each eigenstate;
(4) the level spacing distribution

- **A matrix from a GOE is obtained as follows:**

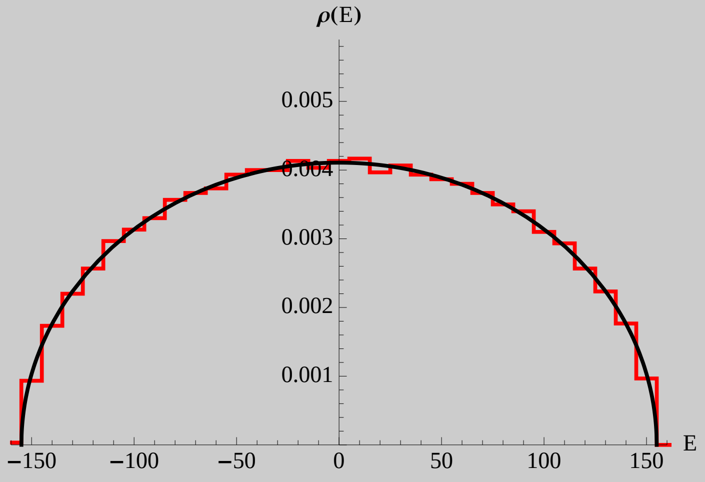
(i) Write a matrix where all elements are random numbers from a Gaussian distribution with mean 0 and variance 1.
(ii) Add this matrix to its transpose to symmetrize it. The result is a matrix from a GOE

- **(1) Code to obtain a matrix from a GOE:**

```
(* matrix from a GOE: matGOE *)
(* dimension of the matrix: dim *)
Clear[dim, rm, matGOE, Egoe, Vecgoe];
dim = 3000;

rm = Table[Table[RandomReal[NormalDistribution[0, 1]], {j, 1, dim}], {k, 1, dim}];
matGOE = rm + Transpose[rm];

Egoe = Eigenvalues[matGOE];
Vecgoe = Eigenvectors[matGOE];
```

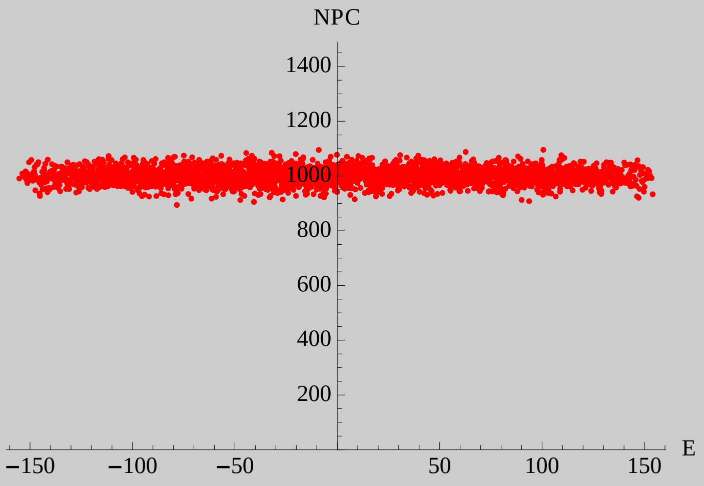
(2) Density of states for a GOE matrix:

```
Clear[bin, Nbin, Eint, NinWindow, hisden, hisdenPlot, semicirc];

bin = 10.;
Nbin = ((160 + bin) - (-160)) / bin;
Eint = Table[(-160 - bin - bin / 2) + bin k, {k, 1, Nbin + 1}];

Do[
  NinWindow[k] = 0;
  , {k, 1, Nbin}];

Do[
  Do[
    If[Eint[[k]] ≤ Egoe[[j]] < Eint[[k + 1]], { NinWindow[k] = NinWindow[k] + 1}];
    , {k, 1, Nbin}];
  , {j, 1, dim}];

(* density of states NORMALIZED: hisden *)
hisden = Flatten[Table[{{Eint[[k]], NinWindow[k] / (bin dim)},
      {Eint[[k + 1]], NinWindow[k] / (bin dim)}}, {k, 1, Nbin}], 1];

hisdenPlot = ListPlot[hisden, Joined → True, PlotRange → All,
   PlotStyle → {Thick, Red}, LabelStyle → Directive[Black, Bold, Medium]];

(* Wigner's semicircular law *)
semicirc = Plot[(2. / (Pi 4. Variance[Egoe])) Sqrt[4. Variance[Egoe] - x^2],
   {x, -Sqrt[4. Variance[Egoe]], Sqrt[4. Variance[Egoe]]},
   PlotStyle → {Thick, Black}, LabelStyle → Directive[Black, Bold, Medium]];

Show[{hisdenPlot, semicirc},
 PlotRange → {{-160, 160}, {0, 0.0059}}, AxesLabel → {"E", "ρ(E)"}]
```



(3) NPC of each eigenstate vs energy

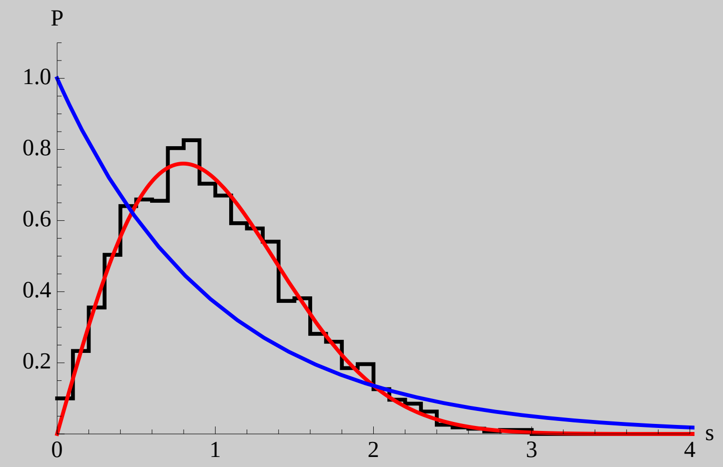
```
Clear[NPCgoe, tabNPC];
Do[
  NPCgoe[j] = 1 / Sum[Abs[Vecgoe[[j, k]] ]^4, {k, 1, dim}];
  , {j, 1, dim}];

tabNPC = Table[{Egoe[[j]], NPCgoe[j]}, {j, 1, dim}];

ListPlot[tabNPC, PlotRange → {0, dim / 2 - 10}, PlotStyle → Red,
 LabelStyle → Directive[Black, Bold, Medium], AxesLabel → {"E", "NPC"}]
```



(4) Level Spacing Distribution

```
(* LEVEL SPACINGS OF THE UNFOLDED SPECTRUM *)
(* Order the eigenvalues from lowest to highest values *)
Clear[Ener];
Ener = Sort[Table[Egoe[[k]], {k, 1, dim}]];

(* Discard ~10% of the eigenvalues located at the borders of the spectrum *)
Clear[percentage, half, spacing];
percentage = 0.1 dim;
half = Floor[percentage / 2.];
Do[
  Clear[average];
  (* Compute the neighboring level spacings
   for the remaining eigenvalues after unfolding them *)
  (* Unfolding here means that the average of each
    group of 10 level spacings = 1 *)
  average = (Ener[[half + 10 j]] - Ener[[half + 10 (j - 1)]]) / 10.;
  Do[spacing[i] = (Ener[[half + i]] - Ener[[(half - 1) + i]]) / average;
   , {i, 1 + 10 (j - 1), 10 j}];
  , {j, 1, Floor[(dim - percentage) / 10]}];

(* HISTOGRAM *)
Clear[spcmin, spcmax, bin, Nofbins];
```

```
spcmin = 0.;
spcmax = 8.;
bin = 0.1;
Nofbins = IntegerPart[(spcmax - spcmin) / bin];

Clear[SPChist, Nhist];
SPChist[1] = spcmin;
Do[SPChist[i + 1] = SPChist[i] + bin, {i, 1, Nofbins}];
Do[Nhist[j] = 0., {j, 1, Nofbins}];

(* Nhist[j] gives how many spacings we have
 in the interval SPChist[j+1] and SPChist[j] *)
Do[
  Do[
    If[SPChist[j] ≤ spacing[k] < SPChist[j + 1], Nhist[j] = Nhist[j] + 1];
    , {j, 1, Nofbins}];
  , {k, 1, 10 Floor[(dim - percentage) / 10]}];

(* Normalization *)
Clear[Norma];
Norma = Sum[bin Nhist[j], {j, 1, Nofbins}];
Do[Nhist[j] = Nhist[j] / Norma, {j, 1, Nofbins}];

(* ListPlot with the obtained data *)
Clear[jj, nl];
jj = 0;
nl = {};
Do[jj += 1;
  nl = Append[nl, {SPChist[jj], Nhist[jj]}];
  nl = Append[nl, {SPChist[jj + 1], Nhist[jj]}];
  , {j, 1, Nofbins - 1}];
DataPlot = ListPlot[nl, Joined → True,
   PlotRange → {{0, 8}, {0, 1}}, PlotStyle → {Black, Thick},
   LabelStyle → Directive[Black, Bold, Medium], AxesLabel → {"s", "P"}];
(* Theoretical curves *)
WignerDyson = Plot[Pi s / 2. Exp[-Pi s^2 / 4.],
   {s, 0, 8}, PlotRange → {0, 1}, PlotStyle → {Red, Thick},
   LabelStyle → Directive[Black, Bold, Medium], AxesLabel → {"s", "P"}];
Poisson = Plot[Exp[-s], {s, 0, 8}, PlotRange → {0, 1}, PlotStyle → {Blue, Thick},
   LabelStyle → Directive[Black, Bold, Medium], AxesLabel → {"s", "P"}];
(* The three curves together *)
Show[{DataPlot, WignerDyson, Poisson}, PlotRange → {{0, 4}, {0, 1.1}}]
```


P
1.0
0.8
0.6
0.4
0.2
0
1
2
3
4
s