Random Matrix Theory

(Sethna, "Entropy, Order Parameters, and Complexity", ex. 1.6, developed with Piet Brouwer)

This is a *Mathematica* notebook. The computer lab in Rockefeller B3 has mathematica installed on the machines; it is also available at the Campus Store for students at a reasonable price.

This hints file is unusually detailed, but only covers the computational parts (a & f) of the exercise; don't neglect the analytical portions! If you are familiar with *Mathematica*, feel free to skip parts you don't need, or start your own notebooks.

One of the most active and unusual applications of ensembles is random matrix theory, used to describe phenomena in nuclear physics, mesoscopic quantum mechanics, and wave phenomena. Random matrix theory was invented in a bold attempt to describe the statistics of energy level spectra in nuclei. In many cases, the statistical behavior of systems exhibiting complex wave phenomena -- almost any correlations involving eigenvalues and eigenstates -- can be quantitatively modeled using ensembles of matrices with completely random, uncorrelated entries!
The most commonly explored ensemble of matrices is the Gauusian orthogonal ensemble (GOE). Generating a member takes two steps:
(1) Generate an NxN matrix whose elements are independent Gaussian distributions of zero mean and standard deviation σ=1.
(2) Add the matrix to its transpose to symmetrize it.

We want to generate an ensemble of NxN symmetric matrices, given as a sum of M and M^T, where M is a random square matrix with independent Gaussian random numbers as entries. Start by finding out how *Mathematica* generates random numbers. Type ?Random for help on random numbers. Try it out. Create a vector of 1000 random numbers using Table. (Follow the table command with a semicolon to suppress output.)

```
? Random
```

```
data = Table[Random[], {i, 1, ...}];
```

Type ?Histogram to find out about plotting the distribution of random numbers. Find out more by clicking on the double-greater-than-signs.
Make a histogram of your data vector, with bins of size {0.1}, normalized so as to be a probability density. Is the distribution Gaussian, uniform, or exponential?

```
? Histogram
```

```
Histogram[data, {0.1}, "ProbabilityDensity"]
```

To generate Gaussian random numbers, we use RandomVariate. It allows us to generate vectors or matrices (or lists of matrices) without using Table. Find out how from ?RandomVariate.

```
? RandomVariate
```

Check that the numbers generated by the NormalDistribution are Gaussian, by doing a histogram.

```
data = RandomVariate[NormalDistribution[], 1000];
Show[Histogram[ ...], Plot[(1/Sqrt[2 π]) Exp[- .../2], {x, -3, 3}]]
```

Generate a 2x2 matrix of Gaussian random numbers (using the NormalDistribution[]), without using Table. Find its transpose.

```
M = RandomVariate[ ..., { ..., ...}]
```

```
? Transpose
```

```
Transpose[M]
```

Generate three 2x2 random matrices as a list of shape {3,2,2}. Take their transposes all at once using Transpose[list,{1,3,2}] (see documentation above). (MatrixForm shows three-index tensors in a funny but easy to view way; delete the semicolons if you want to see the list form of the matrices.)

```
Mats = RandomVariate[ ..., {3, 2, 2}];
MatrixForm[Mats]
MatrixForm[Transpose[ ..., {1, 3, 2}]]
GOEs = Mats + Transpose[ ..., ...];
MatrixForm[GOEs]
```

Write a function GOEEnsemble[num,N] that creates an ensemble of NxN GOE matrices of length num. You can make a multistep routine by separating commands with semicolons inside an overall set of parentheses, for example
F[x_] = (y = x^2; y+2)
should return x^2 + 2.

```
GOEEnsemble[num_, N_] :=
  (Mats = ...]; Mats + Transpose[Mats, ...])
```

Here you may want to generate a larger ensemble, and check if the diagonal elements and the off-diagonal elements have different standard deviations, as you calculated in part (c). (This difference is part of the definition of the GOE ensemble, and is needed to make the ensemble rotation invariant.)

One of the most striking properties that large random matrices share is the distribution of level splittings.

Make an ensemble with 3 matrices of size 2x2. Find the eigenvalues of each using ‘Map’. (Map will apply a function to all elements of a list or expression, so Map[F,{x,y,z}] = {F[x], F[y], F[z]}. One can also use an abbreviation, F\@{x,y,z}.) Sort them (also using Map), and make a Table whose entries are the difference between them.

```
ensemble = GOEEnsemble[ ...];
eigs = Map[Eigenvalues, ensemble];
sortedEigs = Map[ ..., eigs]
diffs = Table[sortedEigs[[i, 2]] - sortedEigs[[i, 1]], {i, 1, 3}]
```

Now generate a function CenterEigenvalueDifferences[ensemble] that returns the difference between the two eigenvalues closest to the center, for a general ensemble of square matrices. (These should be sortedEigs[[i,N/2+1]] and sortedEigs[[i, N/2]].) Your routine can check the length of the ensemble to find out ‘num’, and the length of it’s first element ensemble[[1]] to find out the size N. (*Mathematica* reserves the variable name N, so store it as ‘size’.) Check it on your length-three ensemble.

```
CenterEigenvalueDifferences[ensemble_] :=
 (num = Length[...]; size = ...;
  eigs = ...;
  sortedEigs = ...;
  diffs = Table[..., {i, 1, num}])
```

```
CenterEigenvalueDifferences[ensemble]
```

Plot a histogram of the probability density for 10000 or more such differences. Notice the smooth distribution, with a linear tail near zero.

```
ensemble = GOEEnsemble[...];
diffs = CenterEigenvalueDifferences[...];
Histogram[...]
```

What is this dip in the eigenvalue splitting probability near zero? It is called level repulsion.

Wigner's surmise (meaning 'guess') was that this distribution was universal, that is, independent of the ensemble of matrices. Clearly another ensemble, with all our matrix elements multiplied by two, will have eigenvalue differences twice as big. Wigner surmised that the eigenvalue differences, divided by the mean of the differences would be universal. Write a routine that plots these normalized centered differences against your analytical prediction for 2x2 matrices (see text), which should be ρ(s) = (π s / 2) exp(-π s^2/4).

```
CompareEnsembleWigner[ensemble_] :=
 (diffs = ...; mean = Mean[diffs];
  Show[Plot[(π s/2) Exp[-π s^2/4], {s, 0, 3}],
   Histogram[diffs/mean, {0.1}, "ProbabilityDensity"], PlotRange → All])
```

```
ensemble = GOEEnsemble[10 000, 2];
CompareEnsembleWigner[ensemble]
```

Try N=4 and N=10; do they look similar?

```
ensemble = ...
  ...
```

```
ensemble = ...
  ...
```

Your 2x2 formula is pretty good, but turns out to be up to 2% off for larger matrices; Wigner was wrong about that. But he was correct that the eigenvalue splitting is universal, if you consider large matrices. We can test this by trying a different ensemble. For example, let's try random symmetric matrices filled with +-1.

In the past, we've created these matrices the hard way. But let's just take the signs of the matrix elements generated by the GOE ensemble!In the past, we've created these matrices the hard way. But let's just take the signs of the matrix elements generated by the GOE ensemble! Define PM1Ensemble[num, N] that returns the Sign of the GOE ensemble, multiplied by 1.0. (The latter is to convert the integers +-1 to floats +-1.0, which otherwise will cause *Mathematica* to try to evaluate the eigenvalues exactly.)

```
Sign[GOEEnsemble[4, 2]] // MatrixForm
PM1Ensemble[num_, N_] := 1.0 Sign[...]
```

Try it for 2x2 matrices.

```
ensemble = PM1Ensemble[...];
CompareEnsembleWigner[...]
```

Is it well approximated by the Wigner surmise? Why not? (Hint: how many different symmetric 2x2 matrices of +-1 are there?)

Try it with 4x4 and 10x10 matrices.

```
ensemble = PM1Ensemble[...]
...
```

10x10 should work better, except for a funny bump at zero.

```
ensemble = PM1Ensemble[...]
...
```

The GOE ensemble has some nice statistical properties (see text).

This is our first example of an emergent symmetry. Many different ensembles of symmetric matrices, as the size N goes to infinity, have eigenvalue and eigenvector distributions that are invariant under orthogo-nal transformations even though the original matrix ensemble did not have this symmetry. Similarly, rotational symmetry emerges in random walks on the square lattice as the number of steps N goes to infinity, and also emerges on long length scales for Ising models at their critical temperatures.