Electron transport in disordered environments: Anderson localization

We calculate the ways electrons move through dirty materials. Our Hamiltonian has hopping of strength -tHop between neighboring sites on a one-dimensional chain of atoms, and random on-site energies picked from a Gaussian normal distribution of width W and mean zero.

```
Hamiltonian[tHop_, W_, Nsize_] := Module[{SiteEnergy}, SeedRandom[6572];
  SiteEnergy = RandomVariate[NormalDistribution[..., ...], ...];
  Table[If[i == j, ..., If[i == j + 1 || j == i + 1, ..., ...]],
   {i, 1, ...}, {j, 1, ...}]]
(* Test with small Hamiltonian *)
HamTest = Hamiltonian[1.0, 0.1, 5]
(* Construct full-sized Hamiltonians for later use *)
HamT01 = Hamiltonian[1.0, 0.1, 100];
HamT1 = Hamiltonian[1.0, 1.0, 100];
```

Define the propagator K(i,j,t) = exp(-i H t). Here we use the function MatrixExp. Define ψ_n(t) = K(1,n,t). Hint: what row or column of K corresponds to i=1?

```
K[t_, Ham_] := MatrixExp[-I ...]
(* Test with small matrix *)
K[1.0, HamTest]
ψ[t_, Ham_] := ...
(* Tests: test sum(ψ* ψ) = 1; plot real and imag *)
ψ0Test = ψ[1.0, HamTest]
.... . ...
ListPlot[{Re[ψ0Test], Im[ψ0Test]}, Joined → True]
```

Animate ψ(t). Capture the plot at t=10.

```
(* HamT01 *)
Animate[
 ListPlot[{Re[...], Im[...]}, Joined → True, PlotRange → {-1, 1}], {t, 0, 10}]
```

Make a routine that calculates the current.

(Hint: RotateLeft could be useful. Make sure the imaginary part is zero, and then modify to return the real part. For perfection, drop the term N -> N+1, since there is no site N+1.)

```
J[ψ_, tHop_] := Drop[Re[ ...], -1]
(* Test with HamTest *)
J[ψ[1.0, HamTest], 1.0]
```

Animate the current, first for W=0.1 and then for W=1.0.

```
(* HamT01 *)
Animate[ListPlot[ ..., Joined → True, PlotRange → {-0.5, 0.5}], {t, 0, 100}]

(* HamT1 *)
Animate[ListPlot[ ..., Joined → True, PlotRange → {-0.5, 0.5}], {t, 0, 100}]
```

Calculate eigenvalues and eigenvectors of the Hamiltonian. Plot the eigenvector with lowest energy, and the one with energy closest to zero.

```
{vals, vecs} = Eigensystem[HamT01];
(* {vals, vecs} = Eigensystem[HamT1]; *)

(* Warning: Eigenvalues and eigenvectors are sorted weirdly to the
  absolute value of the eigenvalues. To get the lowest energy eigenvector,
we need to find which has the minimum energy. Use the function 'Ordering'. *)
valLowest = ...;
(* To get the eigenvector location with eigenvalue closest to zero,
use Ordering on Abs[vals] *)
valNearestZero = Ordering[Abs[vals], 1][[1]];
(* To label plots, use PlotLegends for recent versions of Mathematica,
and PlotLegend (with Needs["PlotLegends`"]) for older versions *)

ListPlot[{..., ...}, Joined → True,
 PlotRange → All, PlotLegends → {"Lowest", "Nearest to Zero"}]
```