PHYSICS 104B, WINTER 2010
ASSIGNMENT FIVE SOLUTIONS

[**1.**] For uniform $m = 1.0$ and springs $g = 2.0$ the eigenvalues have the analytic form,

$$\begin{aligned} \lambda_k &= \frac{2g}{m}(1 - \cos k) \\ k &= k_n = \frac{2\pi n}{N} = \frac{2\pi}{N}\{-\frac{N}{2} + 1, -\frac{N}{2} + 2, \cdots \frac{N}{2}\} \end{aligned}$$

The code jacobi_test yields

```
eigenvalues:
  8.000   0.000   0.536   7.464   7.464   0.536
  2.000   6.000   6.000   2.000   4.000   4.000
```

These are readily seen to agree with the analytic formula The first two values are for $k = \pi$ (i.e. n=N/2) and $k = 0$ (i.e. n=0). The third is $k = \frac{\pi}{6}$ (i.e. n=1) and is degenerate with $k = -\frac{\pi}{6}$ (i.e. n=-1), the sixth eigenvalue.

For eigenvectors, The code jacobi_test yields (only the first two eigenvalues/eigenvectors are listed below):

```
eigenvalues:
  8.000   0.000    ...

eigenvectors:
  0.289   0.289   ...
 -0.289   0.289   ...
  0.289   0.289   ...
 -0.289   0.289   ...
  0.289   0.289   ...
 -0.289   0.289   ...
  0.289   0.289   ...
 -0.289   0.289   ...
  0.289   0.289   ...
 -0.289   0.289   ...
  0.289   0.289   ...
 -0.289   0.289   ...
```

These correctly correspond to $k = \pi$, where the masses beat opposite to each other with eigenvector components $V^l_{k=\pi} = \frac{1}{\sqrt{N}} e^{i\pi l}$, and $k = 0$ where they all move together (at zero energy cost) and $V^l_{k=0} = \frac{1}{\sqrt{N}}$.

Turning to a more macroscopic system, for $N = 128$ the first eight eigenvalues and participation ratios are:

```
eigenvalues:
  8.00000   0.00482   0.04329   7.99518   4.39207   3.80373   1.03620   7.69552
participation ratios:
127.99989  85.33268  85.33334  85.33395  85.33334  85.33335  85.33328  85.33329
```

(I have reformatted the print statement in jacobi_test to give more significant figures for the eigenvalues.) Notice jacobi does not sort the eigenvalues. This is often inconvenient when making plots and analyzing data, so you might want to include a sorting routine to your program. The key point here is that all the participation ratios are close to $N = 128$.

When there is a defect spring, tighter than the rest $g = 4$, the first eight eigenvalues and participation ratios given by jacobi_test are (if you put the defect spring between the first two masses in the chain):

```
eigenvalues:
  10.66667   0.00000   7.99518   0.00482   4.02916   3.83164   7.96896   1.03620
participation ratios:
   2.50001 128.00002  85.33228  85.33340 102.66785  82.47289  84.02576  85.33333
```

Notice there is one participation ratios which is much smaller than all the others. This is the localized mode. Note that since jacobi doesn't do any sorting, in principle you might have to look through the whole list of 128 eigenvalues and participation ratios to find the localized one(s). It happened for this case that it appeared right at the top of the list.

Figure 1 shows a plot of the eigenspectrum. You will notice that one eigenvalue is split off from all the others. This is a characteristic feature of localization. Physically it is reasonable: if an energy level is all alone it is less likely to interact (resonate) with the others, and hence be localized.

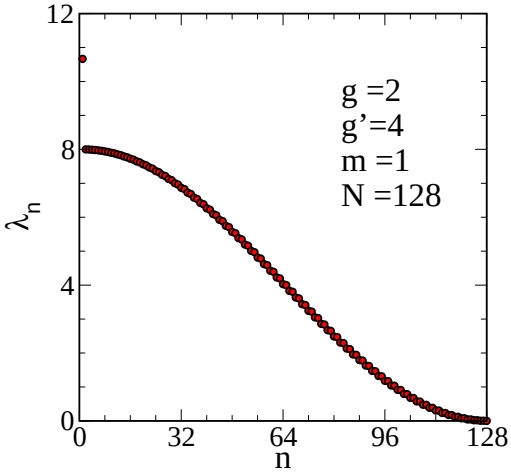

Figure 1: The eigenspectrum for $N = 128, g = 2, m = 1$ and a single defect spring with $g' = 4$. One of the localized modes really jumps out, with an eigenvalue split off much higher than the continuum of delocalized modes. You might also notice that the pair-wise degeneracy of the eigenvalues present in the 'perfect crystal' is lifted. This is a commonly occurring phenomenon you should know about: small perturbations often break degeneracies.

Figure 2 shows the components of the localized eigenvectors. You can see they are big near the defect spring between masses 64 and 65 and fall off. The components alternate sign from mass to mass. This is because the localized mode develops from the eigenvector associalted with $k = \pi$ which has largest eigenvalue.

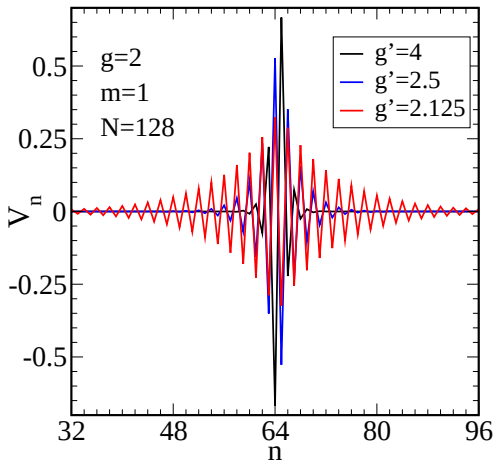

Figure 2: The components of the localized mode for $N = 128, g = 2, m = 1$ and a single defect spring with $g' = 4, 2.5, 2.125$. The defect spring was placed between masses 64 and 65. Notice that as $g' \to g$ the localized mode spreads out more and more. Of course when $g' = g$ there is no defect and no localized mode. The associated eigenenergies of the localized modes are $10.667, 8.333, 8.028$, for $g' = 4, 2.5, 2.125$ respectively. The participation ratios are $2.500, 5.200, 17.058$.

Figure 3 shows an arbitrarily selected second eigenvector, ie. not the localized one. Here I have chosen to show the square of the components. The components are large on many sites: the eigenvecrtors are not localized. It is interesting that some of these vectors have small components near where the localized state lives. I think this is a consequence of the fact that we have a real, symmetric matrix whose eigenvectors are orthogonal.

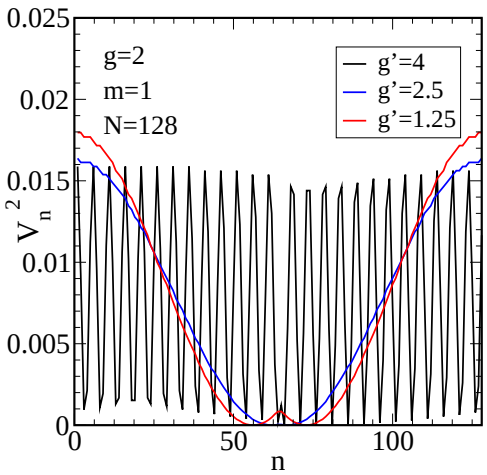

Figure 3: The squares of the components of the first eigenvector (i.e. an arbitrary eigenvector which is not the special localized one) for the same parameters as in Figure 2. This shows that the other eigenvectors indeed do not get localized by the defect: The components are large on many sites.

[**2.**] We fill the diagonal and upper triangular part of a matrix of dimension N randomly with the entries $\pm\frac{1}{2\sqrt{N}}$. The lower triangular part is chosen so the matrix is symmetric. Amazingly, the eigenvalue distribution (in the limit $N \to \infty$) is a perfect semi-circle of radius one!

$$\mathcal{P}(\lambda) = \frac{2}{\pi}\sqrt{1-\lambda^2} \tag{1}$$

Figure 4 shows the result for ten matrices of dimension $N = 1024$. Apparently this N is large enough that the eigenvalue distribution looks quite like the thermodynamic limit $N = \infty$. If you look closely you can see a bit of rounding at $\lambda = \pm 1$ so that there is a small probability of eigenvalues with $|\lambda| > 1$, associated with the fact that N is finite.

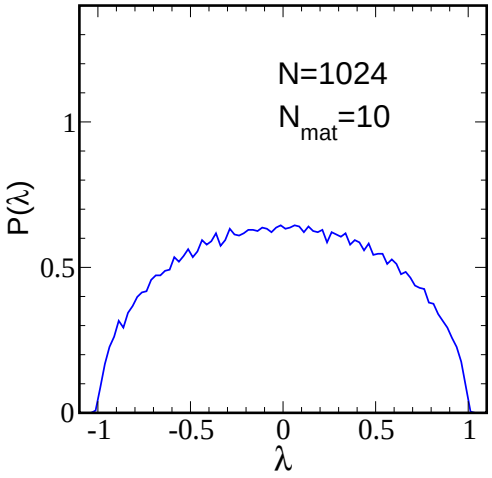

Figure 4: Eigenvalue distribution for a matrix filled randomly with entries $\pm\frac{1}{2\sqrt{N}}$ and also constrained to be symmetric. Although the matrix is random, the distribution of eigenvalues is far from random: it is a semicircle of unit radius.

[3.] I find that the fraction of bankrupt players is f =0.743, and the average person leaves with B =\$4.16 of his/her original \$40. These answers of course have some error bars. Different sets of 10000 people would have a somewhat altered bankruptcy fraction. Running my code with several seeds gives the following for (f, B):
$(0.728, \$4.52)$ $(0.739, \$4.14)$ $(0.736, \$4.23)$.
So you can see the error bars on f are around 0.005.

A very good rule to know (valid for our next topic of Monte Carlo as well) is that when we do a simulation involving N_{players} independent events, the error bars fall as $1\sqrt{N_{\text{players}}}$. If I run with 10^6 players instead of 10^4, I get values $f = 0.7387, 0.7382, 0.7380, 0.7375, 0.7383$. using different random number seeds. The error bar now is clearly less that 0.001.

[**4.**] The eigenvalues are now complex, since the matrix is not symmetric. It turns out they are uniformly distributed on the unit disk in the complex plane. Figures 5,6,7 give the distribution for matrices of dimension $N = 32, 128, 512$ respectively. I generated $400, 100, 25$ matrices for the three N values so that the total number of eigenvalues (12800) is the same for all N. NOTE: I don't quite like the density of real eigenvalues, which seems too high.

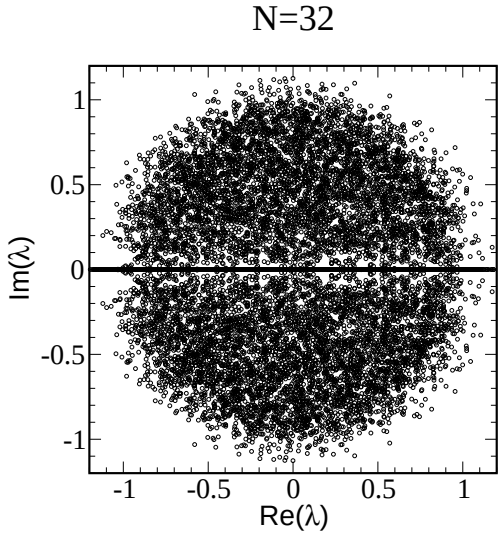

Figure 5: Eigenvalue distribution for 400 matrices of dimension $N = 128$ filled randomly with entries $\pm\frac{1}{\sqrt{N}}$ and allowed to be non-symmetric. Although the matrix is random, the distribution of eigenvalues is far from random: they fill the unit disk in the complex plane uniformly.

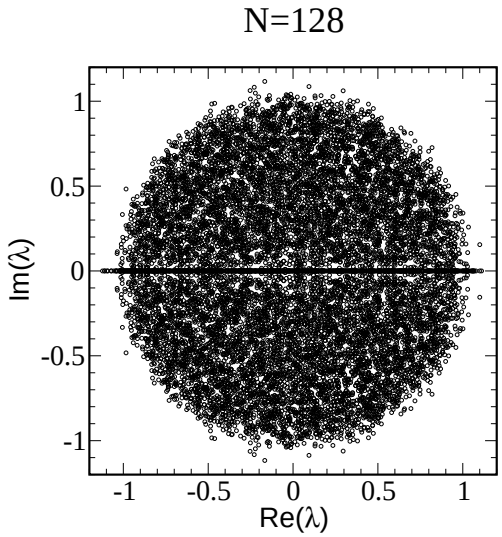

Figure 6: Same as Figure 5 except for 100 matrices of dimension $N = 128$.

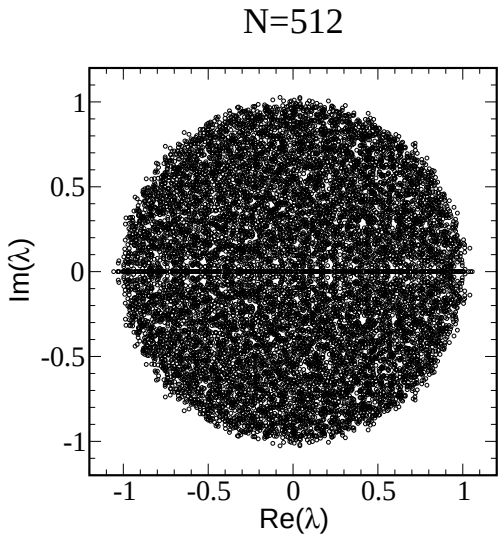

Figure 7: Same as Figure 5 except for 25 matrices of dimension $N = 512$.

CODES

Problem 1: I just used jacobi_test as available on the course website. The one small addition was the computation of the participation ratios:

```
for( j = 1; j <= N; j++) {
   partic=0.;
   for( i = 1; i <= N; i++) {
      partic+=pow(eigenvectors[i][j],4);
   }
   partic=1./partic;
   printf(format2, partic);
}
```

Problem 2: Again, the main routine here is jacobi_test. You need a small ancillary code to generate the random matrix. The essence of that routine is this:

```
for ( i=0 ; i<N ; i++ ) {
    for ( j=i ; j<N ; j++ ) {
       *your favorite random number generator returning 0<r<1*
       if (r<0.5)
          A[i][j]=-1.;
          else
          A[i][j]= 1.;
          A[i][j]=A[i][j]/(2.*sqrt(N));
          A[j][i]=A[i][j];
     }
}
```

Problem 3: Just for fun, I give this program in fortran.

```
      implicit none
      integer*8 M
      integer*8 a,b,i,seed,Npeople,Ngames,j,k,bankroll,br
      integer*8 busted,avefinalbr
      real*8 r,rM,p

      write (6,*) 'enter seed,Npeople,Ngames,p,bankroll'
      read  (5,*)        seed,Npeople,Ngames,p,bankroll

      b=0
      a=7*7*7*7*7
      M=  2*2*2*2*2*2*2*2*2
      M=M*2*2*2*2*2*2*2*2
      M=M*2*2*2*2*2*2*2*2
      M=M*2*2*2*2*2*2*2-1
      rM=dfloat(M)

      i=seed
      r=dfloat(i)/rM
      busted=0
      avefinalbr=0
      do 200 j=1,Npeople
      br=bankroll
      do 100 k=1,Ngames
           i=mod(a*i+b,M)
           r=dfloat(i)/rM
           if (r.le.p) then
               br=br+1
               else
               br=br-1
           endif
           if (br.eq.0) then
               write (67,*) 'player ',j,' went bankrupt'
               busted=busted+1
               go to 120
           endif
100   continue
120   write (67,*) 'player ',j,' final bankroll ',br
      if (br.ge.0) then
           avefinalbr=avefinalbr+br
      endif
200   continue
      write (67,*) 'fraction of bankrupt players= ',
```

```
     1            float(busted)/float(Npeople)
          write (6,*) 'fraction of bankrupt players= ',
     1            float(busted)/float(Npeople)
          write (67,*) 'average final bankroll= ',
     1            float(avefinalbr)/float(Npeople)
          write (6,*) 'average final bankroll= ',
     1            float(avefinalbr)/float(Npeople)
990       format(i35,f12.8)

          end
```

Problem 4: This one I have to give in fortran because I do not have a diagonalizer for non-symmetric matrices in C:

```
          implicit none
          integer imat,Nmat,iran
          integer i,j
          real*8 ran2,r

          integer N
          parameter (N=1024)
          integer N2,N4
          parameter (N2=2*N,N4=4*N)
          complex*16 EC(N),A(N,N),VL(N,N),VR(N,N),WORK(N4)
          real*8 RWORK(N2)
          integer INFO
          character*1 trans
          real*8 mean
          character*80 arg
          character*80 outname

c         INPUT STUFF

          write (6,*) 'outname'
          read(5,1789)  outname
1789      format(A79)
          write (6,*) ' # matrices, iran'
          read  (5,*) Nmat,iran

          open(unit=67, status='new',file=outname)

c         ECHO BACK TO OUTPUT FILE

          write (67,*) '  '
```

```
        write (67,990) N
990     format(' matrix dimension      N= ',i8)
        write (67,991) Nmat
991     format(' # of realizations  Nmat= ',i8)
        write (67,992) iran
992     format(' random # seed      iran= ',i8)

c       LOOP OVER NUMBER OF DISORDER REALIZATIONS

        do 1000 imat=1,Nmat

        write (67,*) ' matrix number ',imat
        write (6,*) ' matrix number ',imat

c       SET UP THE MATRIX

        do 50 i=1,N
        do 30 j=1,N
             r=ran2(iran)
             if (r.le.0.5d0) then
                 A(i,j)=(1.d0,0.d0)
                 else
                 A(i,j)=(-1.d0,0.d0)
             endif
             A(i,j)=A(i,j)/( dsqrt(dfloat(N)) )
30      continue
50      continue
888     format(2i5,2f10.5)

c       GET THE EIGENVALUES/EIGENVECTORS

        call ZGEEV('n','v',N,A,N,EC,VL,N,VR,N,
     1             WORK,N4,RWORK,INFO)

c       WRITE THEM OUT

        do 100 i=1,N
              write (67,970) i,dreal(EC(i)),dimag(EC(i))
100     continue
970     format(i8,2f12.6)

1000    continue
        end
```