

TROTT'S CORNER

The Sound of an Unusual Drum

Michael Trott

The eigenvalue spectrum of the Laplace operator (and its negative) is of fundamental importance in mathematics and physics. In this column, the lowest eigenvalues of a fractal 2D drum are calculated using a finite element discretization. The corresponding eigenfunctions and their gradients are visualized. The cumulative number of the first 1000 eigenvalues is calculated and compared with Weyl's asymptotic formula.

■ Introduction

For fans and friends of partial differential equations (PDEs), *Mathematica 5* is a great step forward. The solution capabilities for parabolic and hyperbolic equations have been greatly expanded compared with Version 4. Now it is possible to choose various derivative approximation schemes and solve $1 + d$ problems. This means that now we can calculate, for example, Schrödinger equation solutions, Wigner functions, nonlinear reaction diffusion equations, and others in a variety of situations. Elliptic PDEs, on the other hand, cannot (yet) be solved using a direct call to `NDSolve`. But Version 5 has another excellent new toolset—sparse arrays and various linear algebra functions that deal with them. For simple linear elliptic PDEs, this is enough to program a solution “manually.” In this Corner, I will discuss a vibrating drum with an unusual shape—one with many corners.

For physicists, the solutions of the Helmholtz eigenvalue problem, $-\Delta \psi_\lambda + V \psi_\lambda = \lambda^2 \psi_\lambda$, have a magic attraction; seeing this equation makes their eyes start to glow and unavoidably thoughts about “Sturm–Liouville,” “Weyl deficiency indices,” and “spectral density” come to mind. This equation describes many physical phenomena ranging from vibrating drums to quantum mechanics. Depending on the dimension, the boundary shape and boundary conditions, and the form of V , an astonishing number of interesting problems and phenomena arise. Typical investigations deal with the distribution of the eigenvalues, as well as quantitative and qualitative features of the eigenfunctions. One recent area of interest is nontrivial shapes. The eigenvalue problem for fractal Koch squares and Koch triangles was investigated in great detail in [1]. In this Corner, we will use a two-dimensional domain with another unusual boundary, with V set to

zero and simple homogeneous Dirichlet conditions. We will calculate some of the eigenvalues and the eigenfunctions of a highly irregular drum.

■ Construction of the Drum

```
In[1]:= Off[General::spell]; Off[General::spell1];
```

We start with a graphic of the drum, the union of a set of more than 300 squares in the plane.

```
In[2]:= squareCenters =
  Union[{{1, 1}, {1, 3}, {1, 7}, {1, 11}, {1, 13}, {1, 17}, {1, 21}, {1, 23}, {1, 29},
    {1, 33}, {1, 39}, {1, 45}, {1, 47}, {3, 3}, {3, 7}, {3, 11}, {3, 17}, {3, 21},
    {3, 27}, {3, 29}, {3, 33}, {3, 35}, {3, 37}, {3, 39}, {3, 41}, {3, 47}, {5, 3},
    {5, 5}, {5, 7}, {5, 9}, {5, 11}, {5, 13}, {5, 17}, {5, 19}, {5, 21}, {5, 23}, {5, 25},
    {5, 27}, {5, 35}, {5, 41}, {5, 43}, {5, 45}, {5, 47}, {7, 1}, {7, 3}, {7, 9}, {7, 19},
    {7, 23}, {7, 27}, {7, 31}, {7, 35}, {7, 37}, {7, 41}, {7, 45}, {9, 7}, {9, 9},
    {9, 11}, {9, 13}, {9, 15}, {9, 19}, {9, 27}, {9, 31}, {9, 33}, {9, 35}, {11, 1},
    {11, 3}, {11, 11}, {11, 15}, {11, 17}, {11, 19}, {11, 23}, {11, 31}, {11, 35},
    {11, 37}, {11, 39}, {11, 43}, {11, 45}, {11, 47}, {13, 3}, {13, 7}, {13, 11},
    {13, 15}, {13, 19}, {13, 23}, {13, 27}, {13, 31}, {13, 35}, {13, 39}, {13, 41},
    {13, 43}, {13, 47}, {15, 3}, {15, 5}, {15, 7}, {15, 9}, {15, 11}, {15, 23}, {15, 25},
    {15, 27}, {15, 29}, {15, 31}, {17, 1}, {17, 3}, {17, 9}, {17, 15}, {17, 17}, {17, 21},
    {17, 23}, {17, 31}, {17, 35}, {17, 37}, {17, 41}, {17, 43}, {17, 47}, {19, 7},
    {19, 9}, {19, 11}, {19, 13}, {19, 15}, {19, 27}, {19, 29}, {19, 31}, {19, 33},
    {19, 35}, {19, 43}, {19, 45}, {19, 47}, {21, 1}, {21, 3}, {21, 15}, {21, 17},
    {21, 19}, {21, 21}, {21, 23}, {21, 31}, {21, 35}, {21, 37}, {21, 39}, {21, 41},
    {21, 43}, {21, 47}, {23, 3}, {23, 5}, {23, 7}, {23, 9}, {23, 13}, {23, 15}, {23, 21},
    {23, 27}, {23, 31}, {23, 41}, {25, 5}, {25, 19}, {25, 21}, {25, 23}, {25, 27},
    {25, 29}, {25, 31}, {25, 33}, {25, 37}, {25, 39}, {25, 41}, {25, 43}, {25, 45},
    {25, 47}, {27, 1}, {27, 5}, {27, 7}, {27, 11}, {27, 15}, {27, 19}, {27, 29},
    {27, 33}, {27, 39}, {29, 1}, {29, 3}, {29, 5}, {29, 11}, {29, 13}, {29, 15},
    {29, 19}, {29, 21}, {29, 23}, {29, 25}, {29, 29}, {29, 33}, {29, 37}, {29, 39},
    {29, 41}, {29, 43}, {29, 45}, {31, 5}, {31, 7}, {31, 9}, {31, 11}, {31, 15},
    {31, 17}, {31, 19}, {31, 25}, {31, 27}, {31, 29}, {31, 45}, {31, 47}, {33, 1},
    {33, 3}, {33, 5}, {33, 11}, {33, 15}, {33, 19}, {33, 23}, {33, 25}, {33, 29},
    {33, 33}, {33, 35}, {33, 37}, {33, 41}, {33, 43}, {33, 45}, {35, 9}, {35, 11},
    {35, 25}, {35, 35}, {35, 45}, {35, 47}, {37, 1}, {37, 5}, {37, 9}, {37, 15},
    {37, 17}, {37, 21}, {37, 23}, {37, 25}, {37, 27}, {37, 31}, {37, 33}, {37, 35},
    {37, 37}, {37, 39}, {37, 47}, {39, 1}, {39, 3}, {39, 5}, {39, 9}, {39, 11}, {39, 13},
    {39, 15}, {39, 27}, {39, 39}, {39, 43}, {41, 5}, {41, 7}, {41, 9}, {41, 15},
    {41, 17}, {41, 19}, {41, 23}, {41, 27}, {41, 29}, {41, 31}, {41, 33}, {41, 37},
    {41, 39}, {41, 41}, {41, 43}, {41, 45}, {41, 47}, {43, 1}, {43, 3}, {43, 5},
    {43, 13}, {43, 15}, {43, 19}, {43, 21}, {43, 23}, {43, 31}, {43, 39}, {43, 43},
    {45, 5}, {45, 9}, {45, 15}, {45, 23}, {45, 25}, {45, 31}, {45, 33}, {45, 35},
    {45, 37}, {45, 39}, {45, 47}, {47, 1}, {47, 3}, {47, 5}, {47, 7}, {47, 9}, {47, 13},
    {47, 15}, {47, 17}, {47, 19}, {47, 25}, {47, 27}, {47, 35}, {47, 43}, {47, 47},
    {49, 1}, {49, 7}, {49, 13}, {49, 19}, {49, 21}, {49, 27}, {49, 29}, {49, 33},
    {49, 35}, {49, 37}, {49, 39}, {49, 41}, {49, 43}, {49, 45}, {49, 47}]] / 2;
```

```
Length[squareCenters]
```

```
Out[3]= 317
```

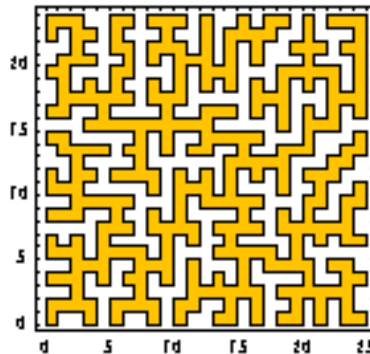
```
In[4]:= drumSquares = Polygon[{-# + {-1, -1} / 2, # + {1, -1} / 2,
    # + {1, 1} / 2, # + {-1, 1} / 2}] & /@squareCenters;
```

The edges of the drum are the edges of unit squares that appear exactly once.

```
In[5]:= drumEdges = Line[First[#]] & /@Select[Split[Sort[Sort /@
    Flatten[Partition[Append[#[[1]], #[[1, 1]], 2, 1] & /@
    drumSquares, 1]]], Length[#] == 1 &];
```

```
In[6]:= Show[Graphics[
    {Hue[0.12], #} & /@drumSquares, {Thickness[0.01], drumEdges}],
    AspectRatio -> Automatic, Frame -> True]
```

From In[6]:=



■ Solving the Eigenvalue Problem

To solve the Helmholtz equation for the drum, we will use a finite difference approximation of the Laplacian. To obtain a sufficient discretization, we subdivide the individual drum squares into rL points in each direction.

```
In[7]:= rL = 10; L = 1 / rL;
```

The two functions `refinePolygon` and `refineEdge` subdivide the squares that form the drum and the outer edges of the drum.

```
In[8]:= refinePolygon[Polygon[{p1_, p2_, p3_, p4_}], rL_] :=
    Module[{d1 = p2 - p1, d2 = p4 - p1},
        Table[Polygon[{p1 + j / rL d1 + k / rL d2, p1 + (j + 1) / rL d1 + k / rL d2,
            p1 + (j + 1) / rL d1 + (k + 1) / rL d2, p1 + j / rL d1 + (k + 1) / rL d2}],
            {j, 0, rL - 1}, {k, 0, rL - 1}] // Flatten];
```

```
In[9]:= refineEdge[Line[{p1_, p2_}], rL_] :=
    Module[{d1 = p2 - p1}, Table[Line[{p1 + j / rL d1, p1 + (j + 1) / rL d1}],
        {j, 0, rL - 1}] // Flatten];
```

Here are the refined squares and the interior and boundary points of the discretized problem.

```
In[10]:= refinedSquares = refinePolygon[#, rL] & /@drumSquares;
```

```

In[11]:= allVertices = Union[Level[refinedSquares, {-2}]];
In[12]:= boundaryVertices =
    Union[Level[refineEdge[#, rL] & /@ drumEdges, {-2}]];
In[13]:= innerVertices = Complement[allVertices, boundaryVertices];

```

We will have to determine the function values for each eigenfunction at 28521 points.

```

In[14]:= {dimD = Length[innerVertices], Length[boundaryVertices]}
Out[14]= {28521, 6360}

```

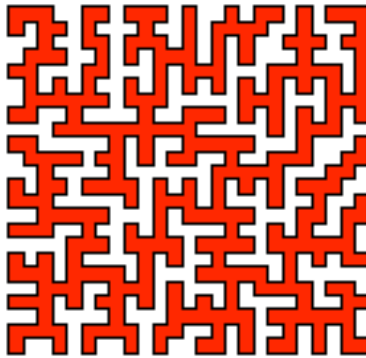
Here is a graphic of the resulting discretized drum.

```

In[15]:= Show[Graphics[{{PointSize[0.001], Hue[0], Point /@ innerVertices},
    {PointSize[0.002], GrayLevel[0], Point /@ boundaryVertices}}],
    PlotRange -> All, AspectRatio -> Automatic]

```

From In[15]:=



The following rules for testing and retrieving the boundary and interior points allow us to build the global finite differences easily and quickly. These rules also make it possible to quickly access the node number of a discretization point in the visualization functions to be defined later.

```

In[16]:= Do[vertexDNumber[innerVertices[[k]]] = k;
    vertexDCoordinates[k] = innerVertices[[k]];
    isInnerDVertex[innerVertices[[k]]] = True, {k, dimD}];

isInnerDVertexQ[vertex_] :=
    isInnerDVertexQ[vertex] = TrueQ[isInnerDVertex[vertex]]

In[18]:= Do[boundaryVertexNumber[boundaryVertices[[k]]] = k;
    boundaryVertexCoordinates[k] = boundaryVertices[[k]];
    isBoundaryVertex[boundaryVertices[[k]]] = True,
    {k, Length[boundaryVertices]}];

isBoundaryVertexQ[vertex_] :=
    isBoundaryVertexQ[vertex] = TrueQ[isBoundaryVertex[vertex]]

```

Next, we must approximate the second derivative at the grid points. The function `stencilWeightList[v][leftPoints, rightPoints]` gives the list of the weights of the v th derivative using the function values at the left and right neighbor points. The function uses the built-in command `NDSolve`FiniteDifference`Derivative`. See Advanced Documentation: `NDSolve` in the Help Browser for more details.

```
In[20]:= stencilWeightList[v_][leftPoints_, rightPoints_] :=
  stencilWeightList[v][leftPoints, rightPoints] =
  Module[{o = leftPoints + 1 + rightPoints, grid},
    grid = Range[o]; Coefficient[#, f /@ grid] & /@
    NDSolve`FiniteDifferenceDerivative[Derivative[v], grid,
    f /@ grid, DifferenceOrder -> o - v]] [[leftPoints + 1]]
```

For instance, these are the weights for a stencil of length seven for the second derivative.

```
In[21]:= stencilWeightList[2][3, 3]
Out[21]= { 1/90, -3/20, 3/2, -49/18, 3/2, -3/20, 1/90 }
```

Now we assemble the system of coupled linear equations representing the approximations of the derivatives. The function `addVertexWeights` adds the contributions of the list of finite difference weights `vws` for each vertex.

```
In[22]:= addVertexWeights[vws_] :=
  If[Length[#] === 1, First[#], #[[1, 1]] -> (Plus @@ (Last /@ #))] & /@
  Split[Sort[vws], #1[[1]] === #2[[1]] &]
```

For a given stencil of odd length $2\alpha + 1$, the function `makeNonVanishingDMatrixElements` returns a list of the nonvanishing elements of the discretization of the negative of the 2D Laplace operator. We incorporate the Dirichlet boundary conditions through an antisymmetric continuation of the function across the boundary, using an odd stencil length, to make sure that is always easily possible.

```
In[23]:= makeNonVanishingDMatrixElements[alpha_Integer] :=
  addVertexWeights[Module[
    {centerVertex, jB, swl = -stencilWeightList[2][alpha, alpha]}, Flatten[
    Table[centerVertex = innerVertices[[k]]; {{k, k} -> 2 swl[[alpha + 1]},
    Function[dir, jB = .; Table[nextVertex = centerVertex + j L dir;
    Which[isInnerDVertexQ[nextVertex],
    {k, vertexDNumber[nextVertex]} -> swl[[alpha + 1 + j]],
    isBoundaryVertexQ[nextVertex], jB = j; Sequence @@ {}, True,
    reflectedVertex = centerVertex + jB L dir - (j - jB) L dir;
    {k, vertexDNumber[reflectedVertex]} -> -swl[[alpha + 1 + j]],
    {j, 1, alpha}]] /@ {{+1, 0}, {-1, 0}, {0, 1}, {0, -1}}},
    {k, dimD}]]]]];
```

The resulting matrix approximates the negative Laplacian. It has 332637 nonvanishing entries.

```
In[24]:= MD7 = SparseArray[makeNonVanishingDMatrixElements[3]]
Out[24]= SparseArray[<332637>, {28521, 28521}]
```

The matrix is symmetric by construction.

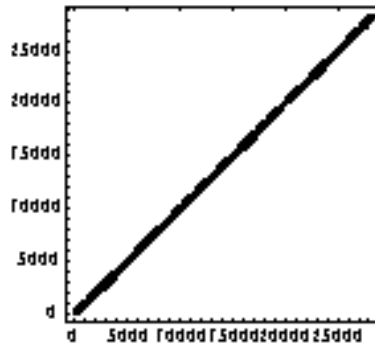
```
In[25]:= Max[MD7 - Transpose[MD7]]
```

```
Out[25]= 0
```

And it is nicely banded.

```
In[26]:= Show[Graphics[{GrayLevel[0], Point[#1]} &@@@
  makeNonVanishingDMatrixElements[3]],
  PlotRange -> All, Frame -> True, AspectRatio -> Automatic]
```

From In[26]:=



Now we are in a position to calculate the eigenvalues and eigenfunctions. Here are the first few eigenvalues, which we reorder and scale.

```
In[27]:= drumEigensystem[n_, opts___] :=
  MapAt[1 / L^2 # &, Reverse /@ Eigensystem[N[MD7], n,
    Method -> {"Arnoldi", opts, Tolerance -> 10^-13,
    BasisSize -> Abs[25 n], MaxIterations -> 10^4}], 1]
```

```
In[28]:= {evals1, evecs1} = drumEigensystem[24, Shift -> 0];
  evals1
```

```
Out[29]= {6.64705, 6.734, 7.02307, 7.13222, 7.14203, 7.14338,
  7.14347, 7.16452, 7.17458, 7.28647, 7.29062, 7.29112,
  7.29209, 7.31277, 7.32897, 7.32952, 7.42389, 7.47142,
  7.51433, 7.52146, 7.53928, 7.70142, 7.71503, 7.74563}
```

■ Visualization of the Eigenfunctions

For the visualization of the eigenfunctions, we define a function `eigen`: `DVibration3DPlot`.

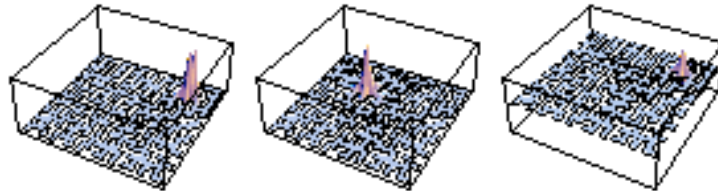
```
In[30]:= drum3DPlot[vertexValueFunction_, opts___] := Graphics3D[{EdgeForm[],
  Map[vertexValueFunction, refinedSquares, {-2}], {GrayLevel[0],
  Thickness[0.006], Map[Append[#, 0] &, drumEdges, {-2}]}}],
  opts, PlotRange -> All, BoxRatios -> {1, 1, 0.4}];
```

```
In[31]:= eigenVibration3DPlot[evvec_, opts___] :=
  With[{ev = evvec / Sign[Plus @@ evvec]},
    drum3DPlot[Function[v, Append[v,
      If[isInnerDVertexQ[v], ev[[vertexDNumber[v]]], 0]]], opts]]
```

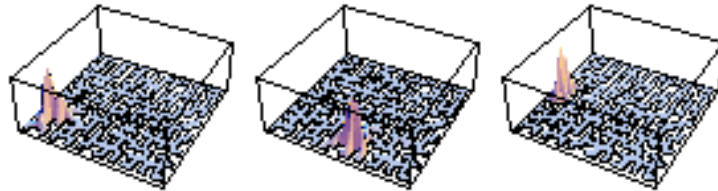
Here we show the first nine eigenfunctions. The two lowest states appear in the two crossings. Such crossings can hold eigenstates by themselves [2].

```
In[32]:= Show[GraphicsArray[#]] & /@
  Partition[eigenVibration3DPlot /@ Take[evvecs1, 9], 3]
```

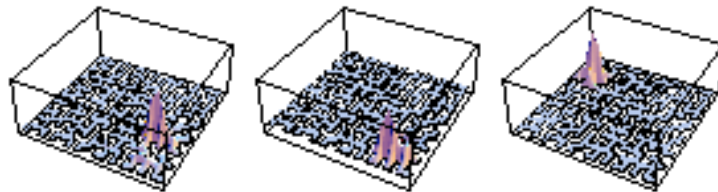
From In[32]:=



From In[32]:=



From In[32]:=



The higher eigenfunctions oscillate more and are better visualized using a contour plot, implemented as the function `eigenVibrationContourPlot`. Here we use the contour values that give homogeneous contour spacing.

```
In[33]:= homogeneousContours[data_, c1_] := With[{λ = Length[data]},
  #[[Round[λ / c1 / 2]]] & /@ Partition[Sort[data], Round[λ / c1]]]
```

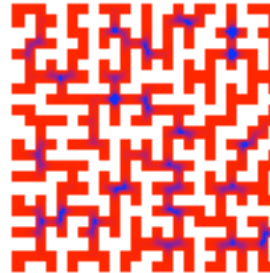
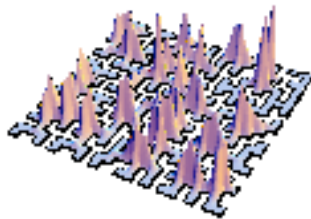
```
In[34]:= {maxX, maxY} = Max /@ Transpose[squareCenters];
squareCentersC = Complement[
  Flatten[Table[{x, y}, {x, 1 / 2, maxX}, {y, 1 / 2, maxY}], 1],
  squareCenters];
coverSquares = With[{a = 0.50001},
  Polygon[{{# + {-a, -a}, # + {a, -a}, # + {a, a}, # + {-a, a}}] & /@
  squareCentersC];
```

```
In[37]:= eigenVibrationContourPlot[vec_, opts___] :=
Module[{ev = Abs[vec], data}, data = Table[
  If[isInnerDVertexQ[{x, y}], ev[[vertexDNumber[{x, y}]]], 0],
  {y, 0, maxY + 1/2, L}, {x, 0, maxX + 1/2, L}];
ListContourPlot[data,
  opts, Contours -> homogeneousContours[ev, 80],
  ColorFunction -> (RGBColor[1 - #, 0, #] &), ContourLines -> False,
  PlotRange -> All, Frame -> False, MeshRange -> {{0, 25}, {0, 24}},
  Epilog -> {GrayLevel[1], coverSquares}]]];
```

The next graphic shows the sum of the squares of the first 24 states. All of the lowest states are localized in the “larger crossing.”

```
In[38]:= With[{evecSum = Plus @@ (evecs1^2)},
  Show[GraphicsArray[Block[{$DisplayFunction = Identity},
    {eigenVibration3DPlot[evecSum, Boxed -> False],
    eigenVibrationContourPlot[evecSum]}]]]]]
```

From In[38]:=



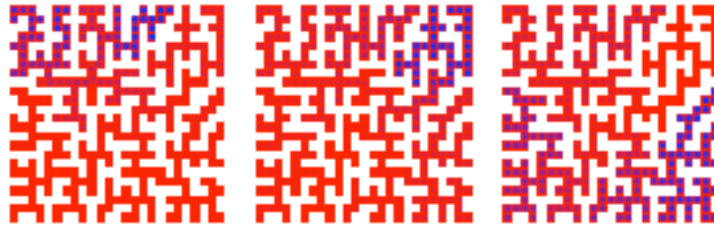
Next, we calculate some of the higher eigenfunctions near the eigenvalue $2\pi^2 \approx 19.7392$. This will be a good test of the method because the function $\psi(x, y) \sim \sin(\pi x) \sin(\pi y)$ is obviously an eigenfunction for the drum and fulfills the boundary conditions trivially.

```
In[39]:= {evals2, evecs2} = drumEigensystem[12, Shift -> L^2 (2 Pi^2 + 0.8)];
evals2
Out[40]:= {19.6675, 19.7075, 19.722, 19.7263, 19.7392, 20.57,
  21.1371, 21.1863, 21.2331, 21.2556, 21.2921, 21.4037}
```

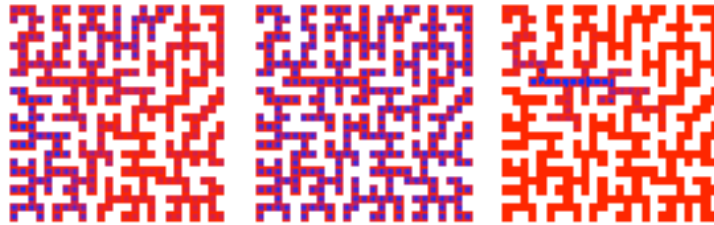
Here we show the corresponding eigenfunctions. The fifth of the selected eigenfunctions is the global product state. Overall we see that the eigenfunctions at this eigenvalue are structured along the drum spine with approximately one maxima per square and only a little structure in the local perpendicular directions.

```
In[41]:= Show[GraphicsArray[#]] & /@ Block[{$DisplayFunction = Identity},
  Partition[eigenVibrationContourPlot /@ Take[evecs2, 9], 3]]
```

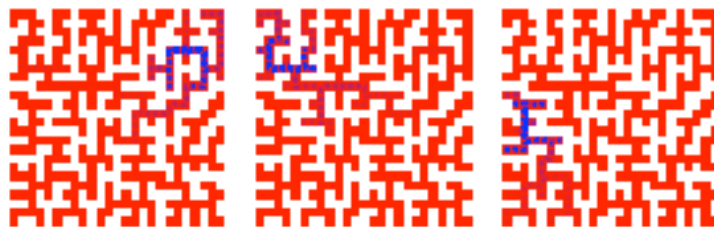
From In[41]:=



From In[41]:=



From In[41]:=



Next, we calculate some eigenvalues and eigenfunctions at still higher eigenvalues.

```
In[42]:= {evals3, evecs3} = drumEigensystem[12, Shift  $\rightarrow$  L2 (8. Pi2)];
evals3
```

```
Out[43]:= {78.7033, 78.7251, 78.7605, 78.7798, 78.8091, 78.892,
78.9135, 78.9487, 78.9614, 79.0539, 79.117, 79.1624}
```

This time we show the nodal curves of the eigenfunctions in addition to the absolute values of the eigenfunctions. The typical state is localized in a small part of the drum. Only the global product state extends over the whole drum.

```

In[44]:= eigenVibrationNodalCurvePlot[evect_, opts___] :=
Module[{ev = Chop[evect, 10^-8], data},
  data = Table[If[isInnerDVertexQ[{x, y}],
    ev[[vertexDNumber[{x, y}]]], 10^-20],
    {y, 0, maxY + 1/2, L}, {x, 0, maxX + 1/2, L}];
  Show[Block[
    {$DisplayFunction = Identity}, {eigenVibrationContourPlot[
      evect, ColorFunction -> (GrayLevel[1 - #] &)],
    ListContourPlot[data, Contours -> {0},
      ContourStyle -> {{Thickness[0.002], Hue[0]}},
      ContourShading -> False, ContourLines -> True, PlotRange -> All,
      Frame -> False, MeshRange -> {{0, 25}, {0, 24}}},
    Graphics[{GrayLevel[1], coverSquares}],
    Graphics[{GrayLevel[0], Thickness[0.005], drumEdges}}]]];

In[45]:= Show[GraphicsArray[#]] & /@ Block[{$DisplayFunction = Identity},
  Partition[eigenVibrationNodalCurvePlot /@ Take[evects3, 12], 3]]

```

From In[45]:=



From In[45]:=



From In[45]:=



From In[45]:=



It is interesting to look at the functions $|\nabla \psi_\lambda(x, y)|$. The function `grad1DValue` again uses a finite difference approximation to calculate an approximate value of $\nabla \psi_\lambda(x, y)$.

```
In[46]:= grad1DValue[vertex_, dir_, maxStencilWidth_, evec_] :=
Module[{jL, jR, j,  $\alpha$  = (maxStencilWidth - 1) / 2, sData, swl, vcs},
If[isBoundaryVertexQ[vertex],
Which[isBoundaryVertexQ[vertex + L dir] ||
isBoundaryVertexQ[vertex - L dir], {jL, jR} = {0, 0},
isInnerDVertexQ[vertex + L dir], {jL, jR} = {0, maxStencilWidth},
isInnerDVertexQ[vertex - L dir], {jL, jR} = {maxStencilWidth, 0}],
{jL, jR} = (j = 1; While[nextVertex = vertex + # j L dir;
j ≤ maxStencilWidth + 1 && isInnerDVertexQ[nextVertex], j++];
j - 1) & /@ {-1, +1};
Which[jL >  $\alpha$  && jR >  $\alpha$ , {jL, jR} = { $\alpha$ ,  $\alpha$ }, jL + jR > maxStencilWidth,
If[jR ≥ jL, jL = Min[ $\alpha$ , jL]; jR = maxStencilWidth - jL,
jR = Min[ $\alpha$ , jR]; jL = maxStencilWidth - jR]];
If[{jL, jR} != {0, 0}, swl = stencilWeightList[1][jL, jR];
vcs = Table[vertexDNumber[vertex + j L dir], {j, -jL, jR}] /.
_vertexDNumber -> 0];
Which[{jL, jR} === {0, 0}, 0, vcs[[+1]] === 0,
Rest[swl].evec[[Rest[vcs]]], vcs[[-1]] === 0,
Most[swl].evec[[Most[vcs]]], True, swl.evec[[vcs]]]

In[47]:= gradValue[vertex_, maxStencilWidth_, evec_] :=
Sqrt[#. #] &@{grad1DValue[vertex, {1, 0}, maxStencilWidth, evec],
grad1DValue[vertex, {0, 1}, maxStencilWidth, evec]}
```

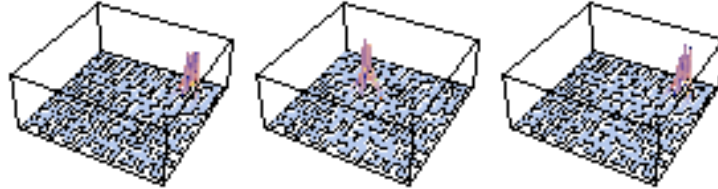
We now define a function `eigenVibrationGrad3DPlot` to visualize the gradients as 3D plots similar to the function `eigenDVibration3DPlot`.

```
In[48]:= eigenVibrationGrad3DPlot[gradInnerVector_,
gradBoundaryVector_, opts___] := drum3DPlot[Function[v, Append[v,
If[isInnerDVertexQ[v], gradInnerVector[[vertexDNumber[v]]],
gradBoundaryVector[[boundaryVertexNumber[v]]]]]]]
```

Here are the norms values of the gradients of the lowest eigenfunctions. As is well known, the gradients of harmonic functions diverge as interior corners [3].

```
In[49]:= Show[GraphicsArray[eigenVibrationGrad3DPlot[
  Table[gradValue[vertexCoordinates[k], 7, #], {k, dimD}],
  Table[gradValue[boundaryVertexCoordinates[k], 7, #],
    {k, Length[boundaryVertices]}]] & /@ Take[evecs1, 3]]]
```

From In[49]:=



■ The Weyl Law

Finally, we calculate some more eigenvalues. Because the Arnoldi method calculates only some of the eigenvalues, we will call the function `drumEigensystem` repeatedly to calculate the first 1000 eigenvalues. To keep the memory usage low, we will calculate 24 eigenvalues each time. By aligning the newly calculated eigenvalues with the last set, we are sure not to miss an eigenvalue.

```
In[50]:= nextEvalCenter[l_] :=
  (Plus@@Take[l, -4] / 4) + 1 / 2 (l[[5]] - l[[4]])

In[51]:= evalNumberMax = 1000;

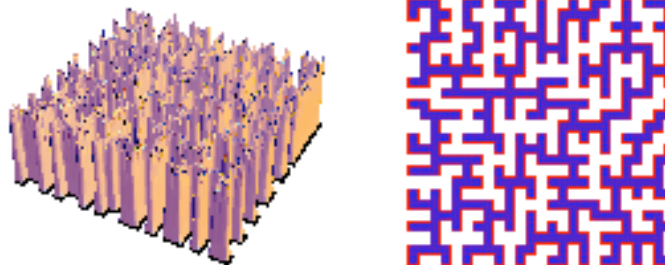
{currentEvals, currentEvecs} = drumEigensystem[24, Shift -> 0];
evalList = currentEvals;
evecSum = Plus@@(currentEvecs^2);
nodalLengthList = nodalLength /@ currentEvecs;

In[56]:= While[
  evDim = 24;
  While[{nextEvals, nextEvecs} =
    drumEigensystem[evDim, Shift -> L^2 nextEvalCenter[evalList]];
  Δs = #.# & /@ ((Take[evalList, {-4, -1}] - #) & /@
    Partition[nextEvals, 4, 1]); Print[{evDim, Δs}];
  minPos = Position[Δs, _?((# < 10^-6) &)];
  newPosis = If[minPos != {} && minPos != {},
    {minPos[[1, 1]] + 4, Length[nextEvals]}, {}];
  newPosis === {}, evDim = Round[3 / 2 evDim]];
evalList = Join[evalList, Take[nextEvals, newPosis]];
evecSum = evecSum + Plus@@(Take[nextEvecs, newPosis]^2);
nodalLengthList =
  Join[nodalLengthList, nodalLength /@ Take[nextEvecs, newPosis]];
Length[evalList] < evalNumberMax, Null]
```

The sum of the first 1000 eigenvectors squared yields the following castle-like graphic. We see a smooth constant background density with relatively small oscillations superimposed.

```
In[57]:= Show[GraphicsArray[Block[{$DisplayFunction = Identity},
  {eigenVibration3DPlot[evectSum, Boxed -> False],
   eigenVibrationContourPlot[evectSum]}]]]
```

From In[57]:=



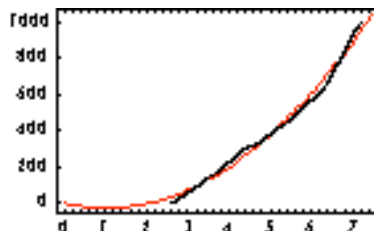
One of the holy grails of $-\Delta \psi_\lambda + V \psi_\lambda = \lambda^2 \psi_\lambda$ is counting the number of eigenvalues λ that are less than Λ . According to the classical Weyl law, the number of eigenvalues $n(\Lambda)$ up to the eigenvalue λ is given by the asymptotic formula $n(\Lambda) = A/(4\pi)\Lambda^2 - l/(4\pi)\Lambda$ [4] for smooth boundaries. Here A is the area of the drum and l the length of its boundary. We see a good overall agreement between the theoretical curve and the calculated cumulative eigenvalue count. At $\Lambda = 2\pi^2$ —where the eigenfunctions start to develop nontrivial lateral structure—we see a clear spectral gap and a change in the slope of the calculated $n(\Lambda)$. For small Λ , the linear contribution from the length of the circumference dominates the quadratic area contribution.

```
In[58]:= {drumBoundaryLength, drumArea} =
  {Length@drumEdges, Length@drumSquares}
```

```
Out[58]:= {636, 317}
```

```
In[59]:= Show[Graphics[{{Hue[0], Thickness[0.01], Line[
  Table[{Λ, drumArea / (4 Pi) Λ^2 - drumBoundaryLength / (4 Pi) Λ},
    {Λ, 0, 7.5, 1 / 10}]}], {GrayLevel[0], PointSize[0.006],
  MapIndexed[Point[{{#1, #2[[1]]}] &, Sqrt[evalList]}]},
  Frame -> True, PlotRange -> All]]]
```

From In[59]:=



While for $\Lambda \lesssim 4.5$ the calculated $n(\Lambda)$ seems to be linear overall, analyzing the detailed behavior of the terms that contribute to $n(\Lambda)$ would lead us outside of the realm of this column. The formula provided earlier was for smooth boundaries. For fractal-like boundaries, we have a correction to the volume term of the

form $c\Lambda^d$ instead [5], where d is the fractal dimension of the boundary. And, indeed, a quadratic function fits the corrections slightly better than a linear one.

```
In[60]:=  $\delta$ IDOS = MapIndexed[{#1, #2[[1]] - drumArea / (4 Pi) #1^2} &,
      Select[Sqrt[evalList], # < 4.5 &]];
fitLinear[x_] = Fit[ $\delta$ IDOS, {1, x}, x];
fitQuadratic[x_] = Fit[ $\delta$ IDOS, {1, x^2}, x];
Function[fit, (Plus@@(#2 - fit[#1])^2 &@@@ $\delta$ IDOS)] /@
  {fitLinear, fitQuadratic}

Out[63]:= {7827.27, 7069.3}
```

In conclusion, we leave it to the interested reader to calculate the distribution of the eigenvalue spacings, the autocorrelation functions of the eigenfunctions, and the localization lengths, and to compare them with the properties of a quantum graph corresponding to the drum spine, to add a magnetic field perpendicular to the drum, and so on; or, to go in a more artistic instead of scientific direction and “play” the drum using Play on the eigenvalues.

■ References

- [1] S. Homolya, C. F. Osborne, and I. D. Svalbe, “Density of States for Vibrations of Fractal Drums,” *Physical Review E*, **67**(2), 2003 pp. 026211-1–026211-13.
- [2] R. L. Schult, D. G. Ravenhall, and H. W. Wyld, “Quantum Bound States in a Classically Unbound System of Crossed Wires,” *Physical Review B*, **39**(8), 1989 pp. 5476–5479.
- [3] S. Russ and B. Sapoval, “Increased Damping of Irregular Resonators,” *Physical Review E*, **65**(3), 2002 pp. 036614-1–036614-10.
- [4] H. P. Baltes and E. R. Hilf, *Spectra of Finite Systems*, Mannheim: BI Wissenschaftsverlag, 1976.
- [5] M. V. Berry, “Distribution of Modes in Fractal Resonators,” *Structural Stability in Physics* (W. Guttlinger and H. Elkheimer, eds.), Berlin: Springer-Verlag, 1979 pp. 51–53.

Michael Trott

Special Functions Developer
 Wolfram Research, Inc.
 mtrott@wolfram.com