Recursive Analysis

Basic Assumptions

The analysis offered below works for two-dimensional fractals that can be drawn using straight lines. Although this code has only been tested for triangles, no assumptions about the number of points in the most basic shape have been made.

The Code

fractalList Data Structure

For the following code, we will use a common language and data structure throughout. A minor refers to the simplest shape of which the fractal is composed. A major is made up of n minors. For example, the Sierpinski gasket is made up of n=3 minor triangles. The fractalList data structure only stores the smallest possible majors in the fractal. The data structure is implemented as a series of nested lists and looks as following:

fractalList = {majors, eigenvalue}majors = {major1, major2, ..., majorm}major = {minors, majorcenter}minors = {minor1, minor2, ..., minorn}minors = {coordinates, eigenvector}coordinates = {x1, x2, ..., xp}
m is the number of majors. The relationship between m, n and the level of subdivision d is \( n^d = m \times n \). Also note that x is a coordinate point and is therefore itself a list.

Finding Eigen Values

Given an eigen value \( z \) at some level of subdivision \( i \), we find the next eigen value \( z_{i+1} \) by solving the equation \( z_{i} = R(z_{i+1}) \) for \( z_{i+1} \).

\[
S(z) = A - z - B(D - z)^{-1}C
\]

\[
R(z) = 1 - \frac{S_{1,1}}{2S_{1,2}}
\]

Also, note that we define the matrix DC as \( -(D - z)^{-1}C \).
Solving for $z_{i+1}$ produces many possible solutions. We wish to pick the smallest value. The code below loops over each potential value and returns the smallest one.

Notes on implementing this function. Mathematica introduces some rounding error in the imaginary plane while running the Solve command. To remove this imaginary part, we first numerically approximate the $z$ value with the N command, then select the real portion with the Re command. Also, the delayed set is used to define dc, s, and r because matrices a through d won’t be defined until the fractal-specific variables are set.

dc := \left(-\frac{4}{25}\right) \text{IdentityMatrix}[Length[d]]; 
s := \text{Inverse}\left(d - (z \text{ IdentityMatrix}[Length[d]])\right); 
r := \text{Simplify}\left[1 - \frac{4}{25}\right];

nextZ[zVal_] := Module[{i, potentialZ, newZ}, 
potentialZ = Solve[zVal == r, z]; 
newZ = \text{Re}\left[zVal \text{ potentialZ}[[i]]\right];

If[newZ > Re[N[Replace[z, potentialZ[[i]]]]] && Re[N[Replace[z, potentialZ[[i]]]]] > 0, 
newZ = Re[N[Replace[z, potentialZ[[i]]]]];
]

Return[\text{Simplify}[newZ]];]

Finding Eigen Vectors

The eigen vector for a given minor triangle is given by multiplying the eigenvector of the major by $T_j$ and evaluating for $z = z_i$ where $j$ is the index of the minor and $i$ is the level of subdivision.

nextEigenVector[t_, zVal_, ev_]:=Module[{nextEV}, 
nextEV = \text{Simplify}[t.ev];
Return[nextEV/. z \rightarrow zVal];]

constructFractal Function

constructFractal is a recursive function that constructs a new major in place of each minor with each call, increasing the level of subdivision by one. Note: in this function, $d$ is the remaining number of subdivisions.

(* scope all variables to local *)
Module[{g, h, i, j, k, m, n, p, ev, newFractalList, majorList, minorList, minorDestCenter, majorCenter, newMinorPts, newMinor, newMajor},

(*basecase : there are no more levels to construct*)


If \(d == 1\),
Return[currentList];

(*ifnotthebasecase, buildmoremajors*)

\[m = \text{Length} [\text{currentList}[[1]]];\] (* m is # existing majors *)
\[n = \text{Length} [\text{currentList}[[1,1,1]]];\] (*nis#minors/major*)
\[p = \text{Length} [\text{currentList}[[1,1,1,1,1]]];\] (* p is # corners in minor *)
evalue = nextZ [currentList[[2]]]; (* evalue is z for level being built *)

majorList = {};
For[i = 1, i \leq m, i++, (*major*)
    majorCenter = currentList[[1,1,1,2]};
    For[j = 1, j \leq n, j++, (*minor*)
        minorDestCenter = \[\sum_{k=1}^{p} \frac{1}{p} (\text{currentList}[[1,i,1,j,k]])\];
        minorList = {};
        For[k = 1, k \leq n, k++, (* minor reference *)
            newMinorPts = {};
            For[h = 1, h \leq p, h++, (* each pt in minor reference *)
                pt = \[\psi_{ij} (\text{currentList}[[1,i,1,j,k]]) - \text{majorCenter} + \text{minorDestCenter}\];
                newMinorPts = Append[newMinorPts, pt];
            ];
            (* h *)
            (*calcnewminor'seigenvectorbasedonnewmajor's (a.k.a.oldminor's)eigenvector*)
evector = nextEigenVector \[\{evalue, \text{currentList}[[1,i,1,j,2]]\}\];
            newMinor = {newMinorPts, evector};
            minorList = Append[minorList, newMinor];
        ]; (* end k *)
        newMajor = {minorList, minorDestCenter};
        majorList = Append[majorList, newMajor];
    ]; (* end j *)
]; (* end i *)
newFractalList = {majorList, evalue}; (*finishedconstructingthislevel.recursivelycalltoconstructnext*)
Return[constructFractal[newFractalList, \(d - 1\)]; (* end if *)
]; (* end module *)
Drawing

The draw and draw3D functions accept a fractalList and returns a corresponding Graphics or Graphics3D object respectively. A single function is responsible for both options; the draw name is overloaded to make calling less confusing.

draw[fractalList_] := draw[fractalList, False];
draw3D[fractalList_] := draw[fractalList, True];

draw[fractalList__threeD_] := Module[{m, n, p, pt, g, i, j, k},
  m = Length[fractalList[[1]]];
  n = Length[fractalList[[2,1,1]]];
  p = Length[fractalList[[2,1,1,1]]];
  g = {};
  For[i = 1, i ≤ m, i++, (*major*)
    For[j = 1, j ≤ n, j++, (* minor in major *)
      (*retrieve points *)
      For[k = 1, k ≤ 3, k++,
        pt[k] = fractalList[[i,1,2,3,1,1,1]];
      ];
      g = Append[g, Line[Append[Table[pt[k], {k, 3}], pt[1]]];
    ];
  ];
  (*ifplottingin3D, parseindisplacementvalues*)
  If[threeD,
    For[k = 1, k ≤ 3, k++,
      pt[k] = Append[pt[k], fractalList[[1,1,2,3,2,3,1,1,1]]];
    ];
    g = Append[g, Line[Append[Table[pt[k], {k, 3}], pt[1]]];
  ];
  ];
  If[threeD,
    Return[Graphics3D[g]]
  , (*else : wewant2D*)
    Return[Graphics[g]];
  ];
]

Running the Program

The following global variables must be set prior to executing the program.
  0009- Matrices a, b, c, and d.
  0009- Transform matrices $\psi_1$ through $\psi_n$.
  0009- Matrices $t_1$ through $t_n$. Note that these cannot be set before matrices
a through d are defined.

0009- A full triangleList containing the fractal to at least level 1 subdivisions.
The suggested sequence of commands is:
00091. Set all required global variables.
00092. Call constructTriangles, passing the level 1 triangleList and the final
number of levels desired.
00093. Call drawTriangles, passing the result of constructTriangles.
00094. Call Show, passing the result of drawTriangles and AspectRatio→Automatic
for 2D plots or Boxed→False for 3D plots.
Note: Increasing the magnification when calling the Show command will
increase the resolution of the rendered graphic.

Example 1: Sierpinski Gasket

Note: Prerendered graphics done at 300% magnification.

\[
a = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}; \quad b = \begin{pmatrix} 0 & -\frac{1}{2} & -\frac{1}{4} \\ \frac{-\frac{1}{2}}{2} & 0 & -\frac{1}{2} \\ \frac{-\frac{1}{2}}{2} & 0 & 0 \end{pmatrix};
\]

\[
c = \begin{pmatrix} -\frac{1}{2} & 0 & -\frac{1}{4} \\ -\frac{-\frac{1}{4}}{4} & 0 & -\frac{-\frac{1}{2}}{2} \\ \frac{-\frac{1}{2}}{2} & 0 & 0 \end{pmatrix}; \quad d = \begin{pmatrix} 1 & -\frac{1}{2} & -\frac{1}{4} \\ -\frac{-\frac{1}{2}}{2} & 1 & -\frac{1}{2} \\ \frac{-\frac{1}{2}}{2} & -\frac{-\frac{1}{2}}{2} & 1 \end{pmatrix};
\]

\[x_1 = \{0,0\}; x_2 = \{2,2\sqrt{3}\}; x_3 = \{4,0\};
\]

\[x_4 = \{3,2\sqrt{3}\}; x_5 = \{0,2\}; x_6 = \{1,\sqrt{3}\};
\]

\[\psi_1 = \left( \frac{1}{2}, 0, \frac{1}{2} \right); \psi_2 = \left( \frac{1}{2}, 0, \frac{1}{2} \right); \psi_3 = \left( \frac{1}{2}, 0, \frac{1}{2} \right);
\]

\[t_1 = \{\{1,0,0\}, dc[3], dc[2]\}];
\]

\[t_2 = \{dc[3], \{0,1,0\}, dc[1]\}];
\]

\[t_3 = \{dc[2], dc[1], \{0,0,1\}\}];
\]

\[seedList = \{\{\{x_1, x_6, x_3\}, \{1, \frac{1}{2}, 0\}\}, \{\{x_6, x_2, x_4\}, \{\frac{1}{2}, 0, -\frac{1}{2}\}\}, \{\{x_5, x_4, x_3\}, \{0, -\frac{1}{2}, -\frac{1}{2}\}\}, \{x_2, x_3, \{0, 0, 0\}\}\};
\]

\[seedListFlat = \{\{\{x_1, x_6, x_3\}, \{-1, 1, -1\}\}, \{x_2, x_3, \{-1, 1, 1\}\}, \{x_2, x_3, \{-1, -1, 1\}\} \};
\]

triangleList = constructFractal[seedList, 6];
triangleList = constructFractal[triangleList, 2];
Show[draw[triangleList], AspectRatio → Automatic]

\[\text{analysis 3g1.enganalysis3g1.cps}
\]

− Graphics−

− Graphics−
Show[graph2D6, AspectRatio -> Automatic]

analysis 3,r2.epsanalysis3,r2.cps
- Graphics -
Show[graph2D7, AspectRatio -> Automatic]
graph3D = draw3D[triangleList];
graph3DFlat = draw3D[triangleListFlat];
graph3D6 = drawTriangles3D[triangleList6];
graph3D7 = drawTriangles3D[triangleList7];
Show[graph3D];

analysis 3,r3.epsanalysis3,r3.cps
Show[graph3D, ViewPoint -> {0, -2, .5}]

analysis 3,r4.epsanalysis3,r4.cps
- Graphics3D -
Show[{{graph3D, graph3DFlat}, ViewPoint -> {-1, 0, 2}}];

analysis 3,r5.epsanalysis3,r5.cps
Show[graph3D6, Boxed -> False];

analysis 3,r6.epsanalysis3,r6.cps
Show[graph3D7, Boxed -> False]
Example 2: Level 3 Sierpinski Gasket

\[ a = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad b = \begin{pmatrix} -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{2} & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{2} & -\frac{1}{2} \end{pmatrix}; \]

\[ c = \begin{pmatrix} -\frac{1}{2} & 0 & 0 \\ -\frac{1}{2} & 0 & 0 \\ 0 & -\frac{1}{2} & 0 \\ 0 & 0 & -\frac{1}{2} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad d = \begin{pmatrix} 1 & -\frac{1}{2} & 0 & 0 & -\frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{4} & 1 & -\frac{1}{4} & 0 & 0 & -\frac{1}{4} \\ 0 & -\frac{1}{4} & 1 & -\frac{1}{4} & 0 & -\frac{1}{4} \\ 0 & 0 & -\frac{1}{4} & 1 & -\frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} \end{pmatrix}; \]

\[ x_1 = \{0,0\}; x_2 = \{3,3\sqrt{3}\}; x_3 = \{6,0\}; \]
\[ x_4 = \{2,0\}; x_5 = \{1,\sqrt{3}\}; \]
\[ x_6 = \{2,2\sqrt{3}\}; x_7 = \{4,2\sqrt{3}\}; \]
\[ x_8 = \{5,\sqrt{3}\}; x_9 = \{4,0\}; \]
\[ x_{10} = \{3,\sqrt{3}\}; \]

\[ \psi_1 = \begin{pmatrix} \frac{1}{3} & 0 \\ 0 & \frac{1}{3} \end{pmatrix}; \psi_2 = \begin{pmatrix} \frac{1}{3} & 0 \\ 0 & \frac{1}{3} \end{pmatrix}; \psi_3 = \begin{pmatrix} \frac{1}{3} & 0 \\ 0 & \frac{1}{3} \end{pmatrix}; \]
\[ \psi_4 = \begin{pmatrix} \frac{1}{3} & 0 \\ 0 & \frac{1}{3} \end{pmatrix}; \psi_5 = \begin{pmatrix} \frac{1}{3} & 0 \\ 0 & \frac{1}{3} \end{pmatrix}; \psi_6 = \begin{pmatrix} \frac{1}{3} & 0 \\ 0 & \frac{1}{3} \end{pmatrix}; \]

\[ t_1 = \{\{0,0,0\}, dc[2], dc[1]\}; \]
\[ t_2 = \{dc[3], \{0,1,0\}, dc[4]\}; \]
\[ t_3 = \{dc[6], dc[5], \{0,0,1\}\}; \]
\[ t_5 = \{dc[7], dc[4], dc[5]\}; \]
\[ t_6 = \{dc[1], dc[7], dc[6]\}; \]
\[ t_4 = \{dc[2], dc[3], dc[7]\}; \]

<< LinearAlgebraMatrixManipulation

\[ m = \text{BlockMatrix}[\{(a, b), \{c, d\}\}]; \]

Eigensystem[m]

\[ \{(\frac{1}{2}, \frac{3}{2}, \frac{3}{2}, \frac{1}{2}, \frac{1}{2}, \frac{3}{2}, \frac{1}{2}, \frac{3}{2}, \frac{1}{2}, \frac{1}{2}, 1, \frac{1}{2}, (3 - \sqrt{2}), 1, \frac{1}{2}, (3 - \sqrt{2}), 1, \frac{1}{2}, (3 - \sqrt{2}), 0\}, \]
\[ \{2, 1, 0, -1, -1, 0, -1, 0, 0, 0, 1\}, \{1, 0, -1, -1, 0, 0, 0, 1\}, \{0, 1, -1, 0, 0, 0, 1\}, \{0, 1, -1, 0, 0, 0, 1\}, \{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\}, \]
\[ \{1, -1, 0, 0, -1, 1, 0, 0, 0, 0, 0, 0, 0, 0\}, \{2, 6, 0, 6, 2, 6, 2, 6, 2, 6, 2, 6, 2, 6, 2\}, \{3, +\sqrt{2}, 3, +\sqrt{2}, 3, +\sqrt{2}, 3, +\sqrt{2}, 3, +\sqrt{2}, 3, +\sqrt{2}, 3, +\sqrt{2}, 3, +\sqrt{2}, 3, +\sqrt{2}, 3, +\sqrt{2}\}, \{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\}, \]
\[ \{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1\}, \{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1\}, \]
\[ \{4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2\}, \{3 +\sqrt{2}, 3 +\sqrt{2}, 3 +\sqrt{2}, 3 +\sqrt{2}, 3 +\sqrt{2}, 3 +\sqrt{2}, 3 +\sqrt{2}, 3 +\sqrt{2}, 3 +\sqrt{2}, 3 +\sqrt{2}, 3 +\sqrt{2}, 3 +\sqrt{2}, 3 +\sqrt{2}, 3 +\sqrt{2}, 3 +\sqrt{2}, 3 +\sqrt{2}, 3 +\sqrt{2}, 3 +\sqrt{2}\}, \{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\}, \]
\[ \{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1\}, \{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1\}, \]
\[ y_i = ylist[[i]]; \]

seedList =

\[ \{\{\{x_1, x_5, x_4\}, \{y_1, y_5, y_4\}\}, \{\{x_6, x_2, x_7\}, \{y_6, y_2, y_7\}\}, \{\{x_9, x_8, x_3\}, \{y_9, y_8, y_3\}\}, \{\{x_5, x_6, x_10\}, \{y_5, y_6, y_10\}\}, \{\{x_{10}, x_7, x_8\}, \{y_{10}, y_7, y_8\}\}, \{\{x_4, x_10, x_9\}, \{y_4, y_10, y_9\}\}\}; \]

7
triangleList = constructFractal[seedList, 4];
graph = draw3D[triangleList];
Show[graph]