Fractal Interpolation Functions: Computation and Visualization with Python

November 4, 2024

Overview

Introduction

Fractal interpolation offers a novel way to model irregular curves, making it valuable for data that displays sudden changes, such as time series. This concept was introduced by Barnsley [2], who used an iterated function system (IFS) to create fractal shapes, providing a foundation for applied science and engineering applications.

Computing Fractal Interpolation Functions (FIFs) requires intensive calculations, often relying on iterative methods that are challenging to handle manually. Python simplifies this process, with libraries like NumPy for computations, Matplotlib for visualization, and SciPy for advanced math, making it ideal for efficiently creating and analyzing FIFs.

Introduction

Fractal interpolation offers a novel way to model irregular curves, making it valuable for data that displays sudden changes, such as time series. This concept was introduced by Barnsley [2], who used an iterated function system (IFS) to create fractal shapes, providing a foundation for applied science and engineering applications.

Computing Fractal Interpolation Functions (FIFs) requires intensive calculations, often relying on iterative methods that are challenging to handle manually. Python simplifies this process, with libraries like NumPy for computations, Matplotlib for visualization, and SciPy for advanced math, making it ideal for efficiently creating and analyzing FIFs.

In project, we have used Python to compute and generate visualizations of FIFs and Fractal Function, as presented in research literature [1], and have applied this Python program to find FIFs for Share Market data.

Introduction

Fractal interpolation offers a novel way to model irregular curves, making it valuable for data that displays sudden changes, such as time series. This concept was introduced by Barnsley [2], who used an iterated function system (IFS) to create fractal shapes, providing a foundation for applied science and engineering applications.
Computing Fractal Interpolation Functions (FIFs) requires intensive calculations, often relying on iterative methods that are challenging to handle manually. Python simplifies this process, with libraries like NumPy for computations, Matplotlib for visualization, and SciPy for advanced math, making it ideal for efficiently creating and analyzing FIFs.
In project, we have used Python to compute and generate visualizations of FIFs and Fractal Function, as presented in research literature [1], and have applied this Python program to find FIFs for Share Market data.

IFS and Fractal Interpolation Functions (FIFs)

Consider the affine transformation $w_n \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_n & 0 \\ c_n & d_n \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e_n \\ f_n \end{bmatrix}$.
Using specific conditions and assumptions (like $\alpha_n(x) = d_n$ and $\psi_n(x) = c_n x + f_n$) given in [1], the attractor of the following IFS provides an FIF for data $\{(x_i, F_i) : i = 0, 1, \ldots, N\}$.

$$\{\mathbb{R}^2 : w_n, n = 1, 2, \ldots, N\}$$

where w_n is defined by the five real numbers a_n, c_n, d_n, f_n, and e_n, which satisfy the following four linear equations:

$$\begin{aligned} a_n x_0 + e_n &= x_{n-1}, \\ a_n x_N + e_n &= x_n, \\ c_n x_0 + d_n F_0 + f_n &= F_{n-1}, \\ c_n x_N + d_n F_N + f_n &= F_n. \end{aligned}$$

Algorithm for Fractal Interpolation Function Generation

Input:

(i) Initial data points $x = [x_0, x_1, ..., x_N]$ and $F = [F_0, F_1, ...F_N]$.

(ii) Predefined scaling factors $d = [d_1, d_2, ..., d_N]$ for the fractal interpolation.

Output:

(i) A scatter plot displaying the fractal interpolation function.

Initialize Input Data and Parameters:

(i) Define input arrays x and F.

(ii) Select scaling factors d based on the desired fractal structure. Set variable d_n to indicate the chosen set of scaling factors.

(iii) Calculate $b = x[N] - x[0]$,

Compute Transformation Coefficients:

(i) Initialize arrays a, e, c and ff with zeros to store transformation coefficients for each sub-interval.

(ii) For each interval $n = 1$ to N: Compute transformation coefficients $a[n], e[n], c[n]$ and $ff[n]$

Perform Random Iteration Algorithm:

(i) Set initial point $(x_c ur, y_c ur) = (0, 0)$.

(ii) Define the total number of iterations.

(iii) Repeat the following steps for each iteration:

    1. Randomly select an index k between 1 and N.
    2. Compute the new coordinates: $new_x = a[k] \times x_c ur + e[k]$, $new_y = c[k] \times x_c ur + d[k-1] \times y_c ur + ff[k]$.

(iv) Update $x_c ur, y_c ur$ to new_x and new_y.

(v) Store $(x_c ur, y_c ur)$ in arrys $x_p lot$ and $y_p lot$ for plotting.

Plotting the Fractal Interpolation Function:

(i) Plot the points stored in x_plot and y_plot as a scatter plot.

(ii) Add a label to the plot showing the scaling factors used.

(iii) Display the plot.

This algorithm systematically explains each step required to generate and visualize the fractal interpolation function.

Example

Input data: x = [0, 0.2, 0.4, 0.6, 0.8, 1.0], F = [9, 11, 15, 8, 12, 10] and d=[-0.3, -0.4, -0.3, 0.4, 0.5].

Scatter plot:

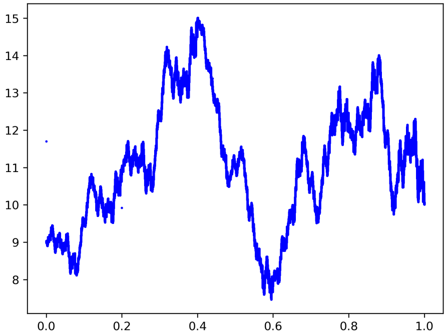


Figure 1. Scaling factors= (-0.3, -0.4, -0.3, 0.4, 0.5)

Fractal Function

The nowhere differentiable Weierstrass function is given by

$$f^{\phi}_{\lambda,l}(x) = \sum_{k=0}^{\infty} \lambda^k \phi(l^k x), \quad x \in R,$$

where $l \geq 2$ be an integer , $1/l < \lambda < 1$ and $\phi : R \to R$ is a $Z-$ -periodic real analytic function. This function displays self-similarity on different scales and it's graph exhibits fractal-like behavior, with intricate and complex structure on all scales([4],[3]).

Algorithm for Plotting the Fractal Function

Input:

(i) Parameters λ, l, n_terms and $\phi(x)$.

(ii) A range of x values in the interval $[0, 1]$.

Output: A plot of the series function over the interval $[0, 1]$.

Define the Series Sum Function:

(i) Create a function $series_sum(x, n_terms)$ that calculates the sum of terms.

(ii) Initialize *result* to an array of zeros, with the same shape as x to store cumulative results.

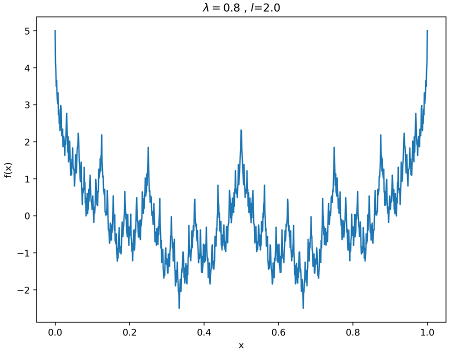
(iii) For each term k from 0 o $n_terms - 1$: Update *result*.

(iv) Return *result* as the computed values of the series sum.

Define the Range of x Values: Generate a set of x values in the interval [0,1] with a resolution points, stored in x_values.
Compute y Values Using the Series Sum Function: Call $series_sum(x_values)$ to calculate the series values, storing the result in y_values.
Plot the Function: Plot x_values on the $x - axis$ and y_values on the $y - axis$.

Example

Input: $l = 2.0$, $\lambda = 0.8$ and $\phi(x) = cos(2\pi x)$.
Plot:



Conclusion

- Fractal Interpolation Functions (FIFs) are useful tools for modeling complex, uneven data. They can create patterns similar to real-world data, like stock prices and natural shapes.
- In this project, we used Python to compute and visualize FIFs. With Python libraries like NumPy, Matplotlib, and SciPy, we were able to perform complex calculations and generate clear visualizations. Python made it easier to implement these mathematical ideas efficiently.
- We also explored fractal functions, such as the Weierstrass function, which helped us understand how fractals create detailed patterns on all scales.

References I

[1] Najmeddine Attia, Taoufik Moulahi, Rim Amami, and Neji Saidi.
Note on fractal interpolation function with variable parameters.
AIMS Mathematics, 9(2):2584–2601, 2024.

[2] Michael F Barnsley.
Fractal functions and interpolation.
Constructive approximation, 2:303–329, 1986.

[3] Tian You Hu and Ka-Sing Lau.
Fractal dimensions and singularities of the weierstrass type functions.
Transactions of the American Mathematical Society, 335(2):649–665, 1993.

[4] Haojie Ren and Weixiao Shen.
A dichotomy for the weierstrass-type functions.
Inventiones mathematicae, 226:1057–1100, 2021.

Thank You