Supporting Information for

"Fractal Geometry of Aerosol Particles and Its Impact on Atmospheric Optical Properties: Development and Validation of the Fractal Aerosol Cluster Model in a Heavy Haze Event"

Liu Zhenxin[1†], Li Weimin[1], Zhang Bihui[2], Li Xiaolan[3], Mao Yuhao[1], Zhang Kuan[1], Ma Xinye[1], Liao Hong[1†]

[1] Jiangsu Key Laboratory of Intelligent Atmospheric Environment Monitoring and Carbon–Pollution Co-control /Jiangsu Collaborative Innovation Center of Atmospheric Environment and Equipment Technology, School of Environmental Science and Engineering, Nanjing University of Information Science and Technology (NUIST), Nanjing 210044, China.

[2] National Meteorological Centre, China Meteorological Administration, Beijing, 100081, China.

[3] Institute of Atmospheric Environment, China Meteorological Administration, Shenyang, Liaoning, 110166, China/Shenyang Academy of Agricultural and Ecological Meteorology, Chinese Academy of Meteorological Sciences, Shenyang 110166, P. R. China

Corresponding author: Liu Zhenxin (liuzhenxin@nuist.edu.cn) and Liao Hong(hongliao@nuist.edu.cn)

Contents of this file

In the Supplementary Information, we provide the Diffusion-Limited Aggregation (DLA) algorithm along with MATLAB scripts for model visualization (Text S1); Meanwhile we have provided time series plots of visibility simulations and observations at different stations (Figure S1).

AlgorithmS1.

The DLA algorithm employed in this study consists of three main components. The first part is the main program body, the function fractal_growth_rad, which simulates the complex and irregular shapes of aerosol clusters as described in section 2.1 of the paper. Additionally, the algorithm calculates the fractal dimension and invokes other functions for the purpose of visualization.

The main part of the DLA algorithm

```
function structure = fractal_growth_rad(N)
    % The size of three-dimensional space
    gridSize = 100;
    % Initialize the three-dimensional space
    structure = zeros(gridSize, gridSize, gridSize);
    origin = [ceil(gridSize/2), ceil(gridSize/2),
ceil(gridSize/2)];

    % Place a fixed particle at the origin at the
beginning
    structure(origin(1), origin(2), origin(3)) = 1;

    % Define the neighborhood: Use a spherical
neighborhood with a radius of 1.5
    neighborOffsets = create_spherical_neighbors(1.5);

    % Initialize the plot
    figure;
    axis equal;
    hold on;
    view(3);
    xlim([0 gridSize]);
    ylim([0 gridSize]);
    zlim([0 gridSize]);
    title(['Dynamic Fractal Aerosol Cluster Growth.
N=',num2str(N)]);

    % Add global light sources
    light('Position', [1 1 1], 'Style', 'infinite');
    light('Position', [-1 -1 -1], 'Style', 'infinite');

    draw_sphere(origin(1), origin(2), origin(3), 1);
```

```
    %  Random walk and build the structure
    for i = 1:N
        %  Randomly generate a particle initially¤„
        particlePos = origin + randi([15, gridSize/2], 1,
3).*sign(rand(1, 3) - 0.5);

        while true
            % Randomly walk to an adjacent grid point
            step =
neighborOffsets(randi(size(neighborOffsets, 1)), :);
            particlePos = particlePos + step;

            % If it goes out of bounds, randomly generate
the particle again
            if any(particlePos <= 1) || any(particlePos >=
gridSize)
                particlePos = origin + randi([15,
gridSize/2], 1, 3).*sign(rand(1, 3) - 0.5);
                continue;
            end

            % Check if it is near an existing structure
            for j = 1:size(neighborOffsets, 1)
                neighborPos = particlePos +
neighborOffsets(j, :);
                if structure(neighborPos(1),
neighborPos(2), neighborPos(3)) == 1
                    % Calculate the distance to the
origin¦»
                    distToOrigin = norm(particlePos -
origin);

                    % Calculate a fixed probability based
on the distance
                    fixationProb = 0.01 + 3.99 *
(distToOrigin / (gridSize/2));

                    % particle fixed?
                    if rand < fixationProb
                        structure(particlePos(1),
particlePos(2), particlePos(3)) = 1;
                        % Dynamically draw the newly grown
particle
                        draw_sphere(particlePos(1),
particlePos(2), particlePos(3), 1.0);
                        drawnow;
                        pause(0.001);
```

```
                        break;
                    end
                end
            end
            if structure(particlePos(1), particlePos(2),
particlePos(3)) == 1
                break; % The particle is fixed, stop this
random walk
            end
        end
    end

    % Calculate the fractal dimension - Box Counting
Method
    boxCounts = [];
    boxSizes = 2.^(0:5);
    for boxSize = boxSizes
        numBoxes = ceil(gridSize / boxSize);
        boxCount = 0;
        for x = 1:numBoxes
            for y = 1:numBoxes
                for z = 1:numBoxes
                    xRange = (x-
1)*boxSize+1:min(x*boxSize, gridSize);
                    yRange = (y-
1)*boxSize+1:min(y*boxSize, gridSize);
                    zRange = (z-
1)*boxSize+1:min(z*boxSize, gridSize);
                    if any(any(any(structure(xRange,
yRange, zRange))))
                        boxCount = boxCount + 1;
                    end
                end
            end
        end
        boxCounts = [boxCounts, boxCount];
    end

    % Fit the relationship between box size and box count
to calculate the fractal dimension
    logBoxSizes = log( 1./boxSizes);
    logBoxCounts = log(boxCounts);
    fractalDim = polyfit(logBoxSizes, logBoxCounts, 1);

    % Display the fractal dimension
    fprintf('Fractal Dimension: %f\n', fractalDim(1));
    disp(['Fractal Dimension=', num2str(fractalDim(1))])
```

```
    % Plot the box counting graph
    figure;
    scatter(logBoxSizes, logBoxCounts, 'filled');
    hold on;

    % Add the fitted line
    fittedLine = polyval(fractalDim, logBoxSizes);
    plot(logBoxSizes, fittedLine, 'r--', 'LineWidth', 2);

    xlabel('log(1/Box Size)');
    ylabel('log(Box Count)');
    title('Box Counting Method: Fractal Dimension');
    hold off;
end
```

Meanwhile, to ensure the normal operation of the main program, we have also constructed a function function neighborOffsets. Its purpose is to create a spherical neighborhood.

Create a spherical neighborhood.

```
function neighborOffsets =
create_spherical_neighbors(radius)
    [x, y, z] = ndgrid(-ceil(radius):ceil(radius), -
ceil(radius):ceil(radius), -ceil(radius):ceil(radius));
    distances = sqrt(x.^2 + y.^2 + z.^2);

    % Select points within the spherical radius, but not
including the origin itself.
    valid = (distances <= radius) & (distances > 0);
    neighborOffsets = [x(valid), y(valid), z(valid)];
end
```

Furthermore, we use the function draw_sphere to plot the clusters, making them more visually intuitive.

Plotting the spatial structure of fractal clusters.

```
function draw_sphere(x, y, z, r)
    % Draw a sphere with a 3D grayscale gradient effect at
the specified position with radius r
    [sx, sy, sz] = sphere(50);
    sphereX = r * sx + x;
    sphereY = r * sy + y;
    sphereZ = r * sz + z;
    colormap(gray);
    % Generate a grayscale color matrix, using normalized
values of the 3D coordinates for the grayscale gradient
    C = sqrt(sx.^2 + sy.^2 + sz.^2); % Generate grayscale
using normalized distance from the center of the sphere on
```

```
the surface
    % Plot the sphere with a grayscale gradient
    surf(sphereX, sphereY, sphereZ, C, 'FaceColor',
'interp', 'EdgeColor', 'none', 'FaceLighting', 'gouraud');
end
```

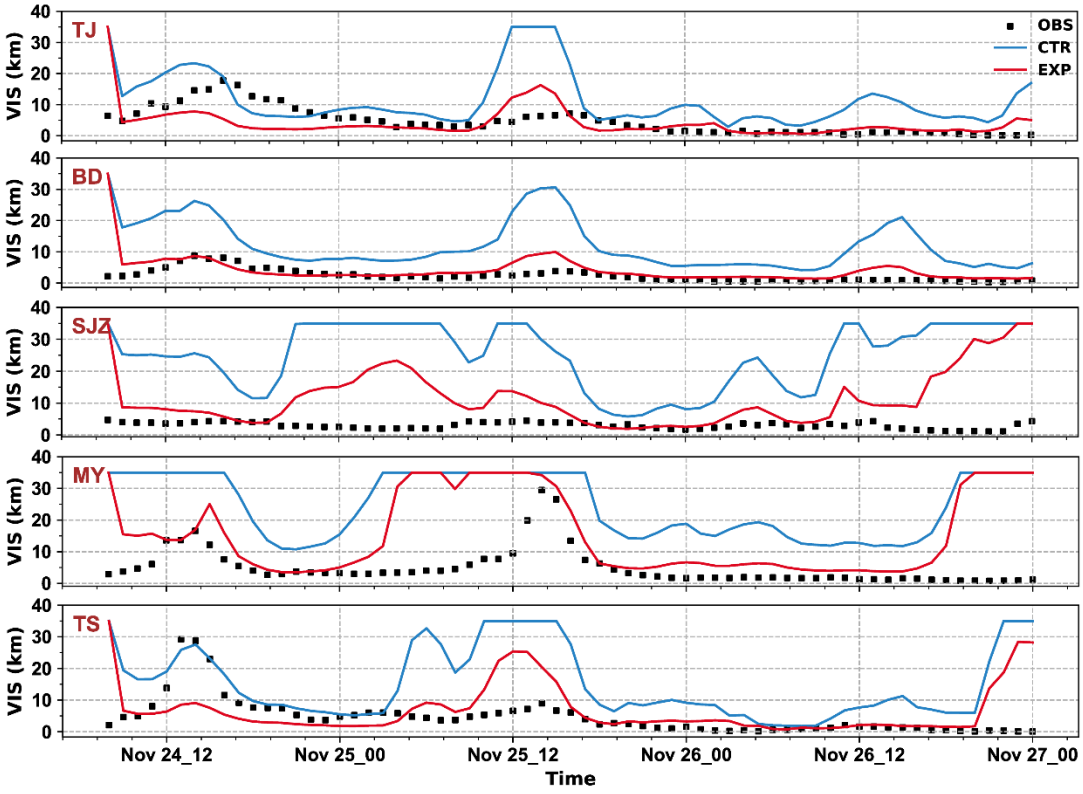

Figure S1. The AV timeseries at station TJ, BD, SJZ, MY and TS: the black square markers (OBS) represent the observation; the blue line (CTR) stands for other by CTR and the red line (EXP) by EXP simulation.

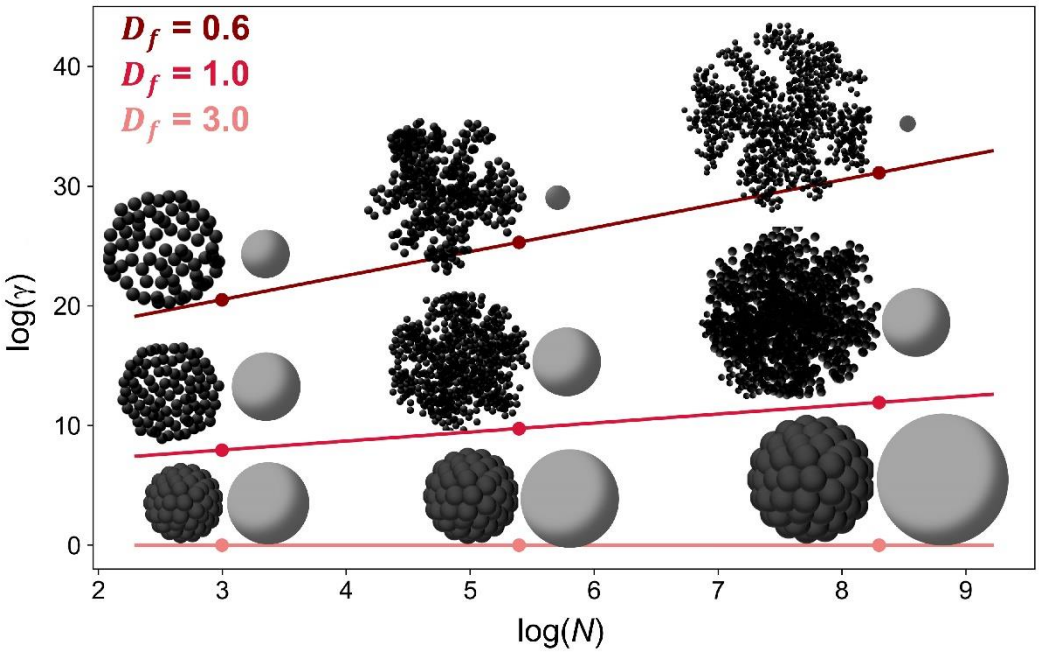

Figure S2. monomer number log(N_S) and fractal dimension D_f versus log(γ). The different color intensity of solid lines represents different fractal dimensions D_f.The black markers and their sizes represent for the aerosol cluster and its optical equivalent radius. While the gray markers represent for those aerodynamical spherical equivalent radius.

According to Fig. S2 (Figure 2 in the previous draft) we show the most intrinsically different feature of fractal clusters compared to previous aerosol particles with a default dimension of 3: i.e., as the aerosol size (N) grows, the optically equivalent radius of the clusters grows faster than the aerodynamically equivalent radius and the ratio γ of the two is not a linear factor, but rather there is a power-law difference, which is related to the cluster's fractal geometry feature: the fractal dimension of the cluster: D_f. The further D_f deviates from 3, the larger the value of log γ, i.e., a cluster with a smaller aerodynamic equivalent radius also has a stronger optical equivalent radius, which results in a stronger atmospheric scattering signature.

Exceptionally, when D_f is smaller than 1 (a situation that holds only in theoretical derivations, and no such exaggerated examples seem to have been confirmed to have been found in the current sampling observations), it is even possible to have a situation in which the optically equivalent radius of a cluster grows rapidly as the monomers continue to coalesce while the aerodynamically equivalent radius decreases at the same time.

The relevant theoretical derivation is in Sec.2.2 of the new manuscript.