Statistical Analysis of Diffusion over Fractal Sets

František Gašpar

September 4, 2024

FNSPE CTU in Prague

Waypoints

- Dimension(s)
- Diffusion over fractal sets
- Alternative numerical approach

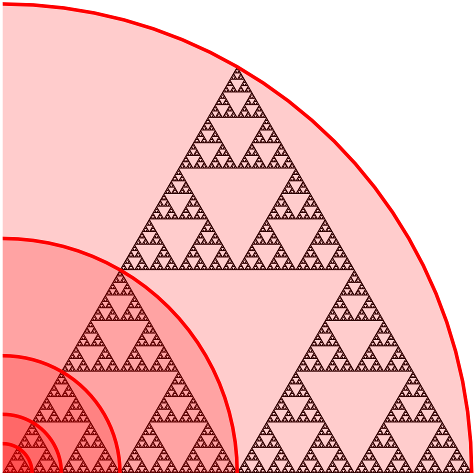
Fractal Dimension

- Sierpinski Gasket
- double the distance, triple the mass
- $d_f = \log 3 / \log 2$

Mass Scaling

$$M \sim L^{d_f}$$

- Similarity dimension
- Hausdorff dimension



Diffusion on Fractals

Fractal Dimension

$$M \sim L^{d_\mathrm{f}}$$

Walk Dimension

$$t \sim L^{d_\mathrm{w}}$$

Spectral Dimension

$$d_\mathrm{s} = 2\,\frac{d_\mathrm{f}}{d_\mathrm{w}}$$

Observables

- Return Probability

$$\Pr(X_t = \boldsymbol{x}_0) \sim t^{-d_\mathrm{s}/2}$$

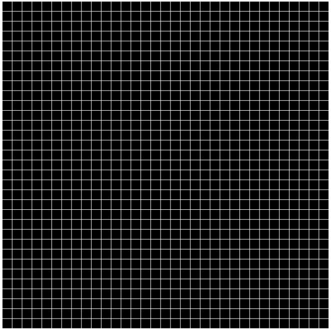
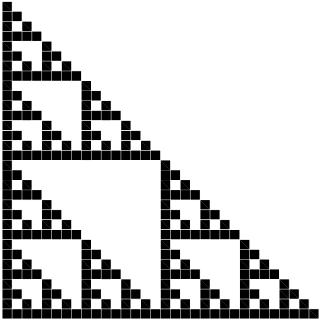
- Absolute Moments

$$\mathrm{E}\,||X_t - \boldsymbol{x}_0||_2^\alpha \sim t^{\alpha/d_\mathrm{w}}$$

Porous Media Diffusion

https://en.wikipedia.org/wiki/Porous_medium

Grid Based Models

$d = 2$	$d_\mathrm{f} = \log 3 \big/ \log 2$
	

Random Walk on Graph

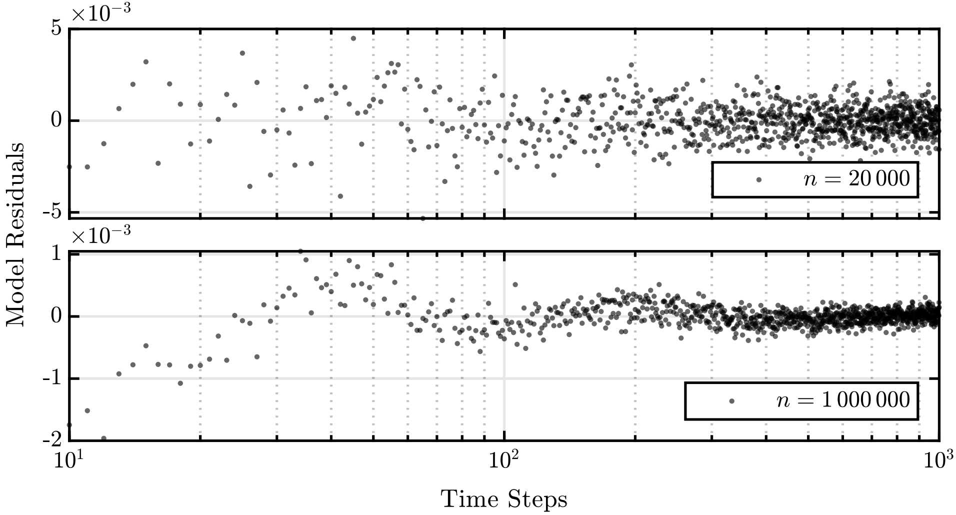
$$\mathcal{G} = (\mathcal{F}, \mathcal{E}), \quad \mathcal{F} \subset \mathbb{Z}^d, \quad \mathcal{T} \subset \mathbb{Z}^d$$

$$p \in (0,1), \quad c(\boldsymbol{x}) = \operatorname{card} \mathcal{N}(\boldsymbol{x}), \quad \mathcal{N}(\boldsymbol{x}) = \big(\boldsymbol{x} \oplus \mathcal{T}\big) \cap \mathcal{F}$$

$$\Pr\big(X_{t+1} = \boldsymbol{y} \,|\, X_t = \boldsymbol{x}\big) = \begin{cases} p, & \boldsymbol{y} \in \mathcal{N}(\boldsymbol{x}) \\ 1 - c(\boldsymbol{x})\, p, & \boldsymbol{y} = \boldsymbol{x} \\ 0, & \text{otherwise} \end{cases}$$

$$\boldsymbol{x}, \boldsymbol{y} \in \mathcal{F}$$

Monte Carlo Simulations: Return Probability Model



Full Space Numerical Approach

- $\mathrm{P}_t : \mathbb{Z}^d \mapsto [0, 1]$
- $\mathrm{P}_t(\boldsymbol{x}) = \Pr(X_t = \boldsymbol{x} | X_0 = \boldsymbol{x}_0)$
- $\mathrm{P}_0(\boldsymbol{x}) = 1(\boldsymbol{x} = \boldsymbol{x}_0)$

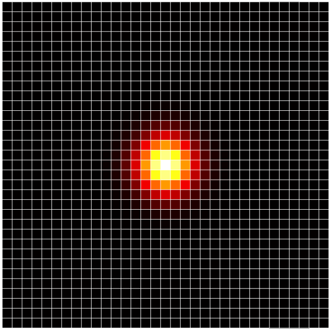
Full Grid Probability Evolution

$$\mathrm{P}_{t+1} = \mathrm{N} * \mathrm{P}_t$$

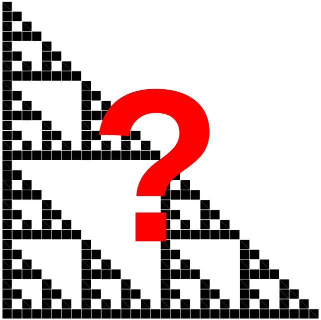
- $\mathrm{N}_{\text{square}} = \begin{pmatrix} 0 & p & 0 \\ p & 1-4p & p \\ 0 & p & 0 \end{pmatrix}$
- $\mathrm{N}_{\text{hexa}} = \begin{pmatrix} p & p & 0 \\ p & 1-6p & p \\ 0 & p & p \end{pmatrix}$

Random Walk Over Grid Based Models

$d = 2$



$d_f = \log 3 / \log 2$



Constrained Convolution Schema

- $\mathsf{P}_t, \mathsf{F}, \mathsf{C} : \mathbb{Z}^d \mapsto \mathbb{R}$
- $\mathsf{P}_t(\boldsymbol{x}) = \Pr(X_t = \boldsymbol{x} | X_0 = \boldsymbol{x}_0)$
- $\mathsf{P}_0(\boldsymbol{x}) = \mathsf{I}(\boldsymbol{x} = \boldsymbol{x}_0)$
- $\mathsf{F}(\boldsymbol{x}) = \mathsf{I}(\boldsymbol{x} \in \mathcal{F})$
- $\mathsf{N}(\boldsymbol{x}) = \mathsf{I}(\boldsymbol{x} \in \mathcal{T})$
- $\mathsf{C} = \mathsf{F}\,(\mathsf{N} * \mathsf{F})$

Sparse Grid Probability Evolution

$$\mathsf{P}_{t+1} = \mathsf{P}_t\,(1 - p\,\mathsf{C}) + p\,\mathsf{F}\,(\mathsf{N} * \mathsf{P}_t)$$

- $\mathsf{N}_{\text{square}} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$
- $\mathsf{N}_{\text{hexa}} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$

Constrained Convolution Schema

- Markov Chain transition probabilities

$$p_{t+1}(\boldsymbol{x}) = \sum_{\boldsymbol{y} \in \mathcal{F}} p_t(\boldsymbol{y})\, p(\boldsymbol{y} \to \boldsymbol{x}) \tag{1}$$

$$= p_t(\boldsymbol{x})\, p(\boldsymbol{x} \to \boldsymbol{x}) + \sum_{\boldsymbol{y} \in \mathcal{F}, \boldsymbol{y} \neq \boldsymbol{x}} p_t(\boldsymbol{y})\, p(\boldsymbol{y} \to \boldsymbol{x}) \tag{2}$$

$$= p_t(\boldsymbol{x}) \left(1 - p\, c(\boldsymbol{x})\right) + \sum_{\boldsymbol{y} \in \mathcal{F}} p_t(\boldsymbol{y})\ p\, \mathbb{I}\Big(\boldsymbol{y} \in \mathcal{N}(\boldsymbol{x})\Big) \tag{3}$$

Constrained Convolution Schema

$$\mathsf{P}_{t+1}(\boldsymbol{x}) = \Big(\mathsf{P}_t\,(1 - p\,\mathsf{C})\Big)(\boldsymbol{x}) + p \sum_{\boldsymbol{y}\in\mathbb{Z}^d} \mathsf{P}_t(\boldsymbol{y})\,\mathsf{I}\big(\boldsymbol{y} \in \mathcal{N}(\boldsymbol{x})\big)\,\mathsf{I}\big(\boldsymbol{x} \in \mathcal{F}\big) \tag{4}$$

$$= \Big(\mathsf{P}_t\,(1 - p\,\mathsf{C})\Big)(\boldsymbol{x}) + p \sum_{\boldsymbol{y}\in\mathbb{Z}^d} \mathsf{P}_t(\boldsymbol{y})\,\mathsf{I}\Big(\boldsymbol{y} \in \big(\boldsymbol{x} \oplus \mathcal{T}\big) \cap \mathcal{F}\Big)\,\mathsf{F}(\boldsymbol{x}) \tag{5}$$

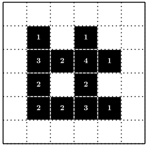
$$= \Big(\mathsf{P}_t\,(1 - p\,\mathsf{C})\Big)(\boldsymbol{x}) + p\,\mathsf{F}(\boldsymbol{x}) \sum_{\boldsymbol{y}\in\mathbb{Z}^d} \mathsf{P}_t(\boldsymbol{y})\,\mathsf{I}\Big(\boldsymbol{y} \in \big(\boldsymbol{x} \oplus \mathcal{T}\big)\Big)\,\mathsf{I}\big(\boldsymbol{y} \in \mathcal{F}\big) \tag{6}$$

$$= \Big(\mathsf{P}_t\,(1 - p\,\mathsf{C})\Big)(\boldsymbol{x}) + p\,\mathsf{F}(\boldsymbol{x}) \sum_{\boldsymbol{y}\in\mathbb{Z}^d} \mathsf{P}_t(\boldsymbol{y})\,\mathsf{I}\big((\boldsymbol{x} - \boldsymbol{y}) \in \mathcal{T}\big)\,\mathsf{F}(\boldsymbol{y}) \tag{7}$$

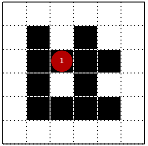
$$= \Big(\mathsf{P}_t\,(1 - p\,\mathsf{C})\Big)(\boldsymbol{x}) + p\,\mathsf{F}(\boldsymbol{x}) \sum_{\boldsymbol{y}\in\mathbb{Z}^d} \mathsf{P}_t(\boldsymbol{y})\,\mathsf{N}(\boldsymbol{x} - \boldsymbol{y}) \tag{8}$$

$$= \Big(\mathsf{P}_t\,(1 - p\,\mathsf{C})\Big)(\boldsymbol{x}) + p\,\mathsf{F}(\boldsymbol{x})\big(\mathsf{P}_t * \mathsf{N}\big)(\boldsymbol{x}) \tag{9}$$

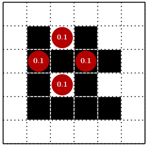
Time Evolution (Sub)Steps: Square Topology



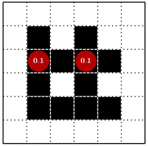
F, C



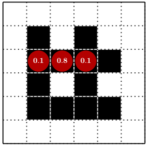
P_0



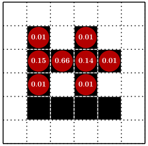
$p\left(P_0 * N\right)$



$p\,F\left(P_0 * N\right)$

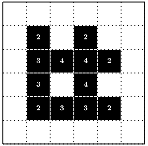


P_1

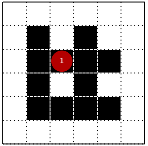


P_2

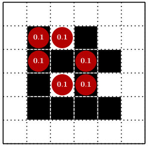
Time Evolution (Sub)Steps: Hexagonal Topology



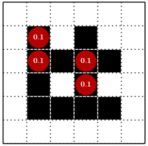
F, C



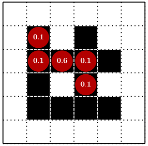
P_0



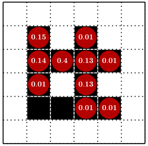
$p\left(P_0 * N\right)$



$p\,F\left(P_0 * N\right)$



P_1



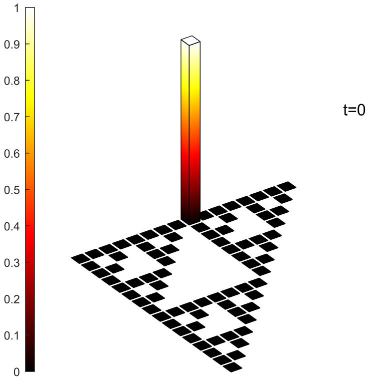
P_2

MATLAB Implementation

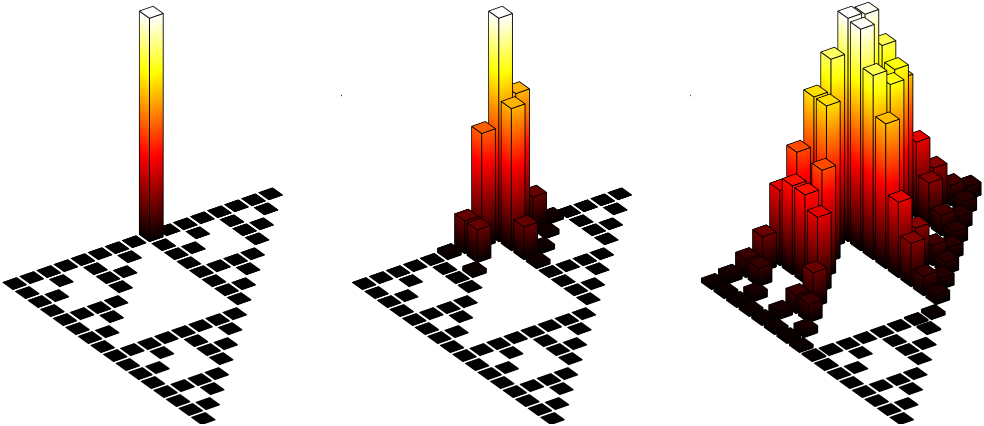
```
% parameters
time_steps = 1e2;
p = 0.1;

% load set model
data = load("data_SierpGasket.mat");
F = data.F;
start_idx = data.start_idx;
% neighbourhood
N = [1, 1, 0;
     1, 0, 1;
     0, 1, 1;];
% precalculated arrays
C = conv2(F, N, "same") .* F;
S = 1 - p .* C;
pF = p .* F;

% initial probability
P = zeros(size(F));
P(start_idx) = 1;
```

```
% checks
assert(sum(P, "all") == 1)
assert(sum(P .* F, "all") == 1)

% resulting return probabilities
ret_probabs = nan(time_steps, 1);
% GPU support
gpuType = @(a) gpuArray(double(a));
if canUseGPU()
    P = gpuType(P);
    pF = gpuType(pF);
    N = gpuType(N);
    S = gpuType(S);
    ret_probabs = gpuType(ret_probabs);
end

for t = 1:time_steps
    P = P .* S + pF .* conv2(P, N, "same");
    ret_probabs(t) = P(start_idx);
end
```

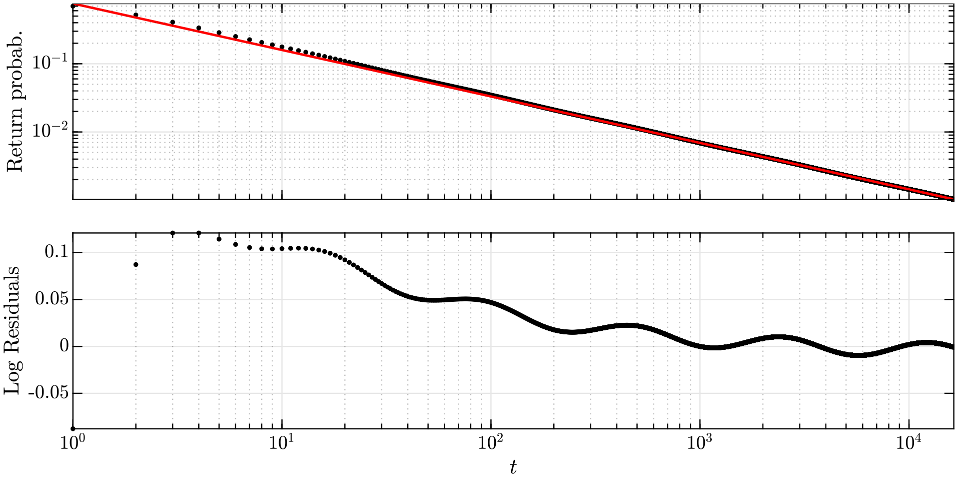
Probability on Sierpinski Gasket



Probability on Sierpinski Gasket



Return Probability



- Dimension(s)
- Random walks
- Constained convolution schema

Thank you for your attention!