Stochastic Price Path Simulators

Janice Wu

October 12, 2025

Overview

simulator.py provides NumPy implementations for simulating financial price paths under several standard models and a utility for converting prices to per–period log returns. These simulations can later be used to evaluate the performance of time–series generation models.

- prices_to_logreturns: convert prices to $\log(S_{t+1}/S_t)$.
- gbm: Geometric Brownian Motion (GBM).
- cev: Constant Elasticity of Variance (CEV) with Milstein/Euler schemes and boundary handling.
- merton: Merton jump–diffusion with Poisson jumps and lognormal jump sizes.
- variance_gamma: Variance–Gamma (VG) pure-jump model via gamma subordination, with optional martingale correction to enforce $\mathbb{E}[S_t] = S_0 e^{(r-q)t}$.
- heston: Heston stochastic volatility with Full–Truncation Euler (variance) and log–Euler (price); supports time–varying parameters and correlated shocks.
- garch_ret: GARCH(1,1) (Normal or standardized Student-t innovations) with optional AR(1) mean; returns integrated to prices.

All simulators accept scalars or time–varying vectors for parameters (broadcast over N paths), return arrays with shapes documented below, and expose a seed for reproducibility.

Notation

Throughout, N denotes number of paths, T the number of steps, and Δt the step size $dt > 0$. We use $Z_t \sim \mathcal{N}(0,1)$ i.i.d., and for Merton jumps $K_t \sim \text{Poisson}(\lambda \Delta t)$ with jump sizes $J = \exp(Y)$, $Y \sim \mathcal{N}(m_J, s_J^2)$.

1 prices_to_logreturns

Signature. `prices_to_logreturns(prices, axis=1, check_positive=True)`

Purpose. Convert prices to per–step log returns:

$$r_t \equiv \log \frac{S_{t+1}}{S_t} = \log S_{t+1} - \log S_t.$$

Parameters.

- `prices`: array-like with time along `axis` (shape $(N, T+1)$, $(T+1,)$, or broadcastable).
- `axis` (`int`, default 1): time axis; if your data is $(T+1, N)$, set `axis=0`.
- `check_positive` (`bool`, default `True`): if `True`, raises `ValueError` when any price in a ratio is ≤ 0.

Returns. Array of log returns with the same shape as `prices` except length reduced by 1 along `axis`.

Notes. Works for single series or batched paths. Uses `np.diff(np.log(prices), axis=axis)`; with `check_positive`, validates positivity of prices used in ratios.

2 `gbm`

Signature. `gbm(N,T,mu,sigma,S0=100.0,dt=1.0,seed=None)`

Model. Geometric Brownian Motion:

$$dS_t = \mu S_t\, dt + \sigma S_t\, dW_t \quad \Rightarrow \quad \log \frac{S_{t+1}}{S_t} \sim \mathcal{N}\Big(\big(\mu - \tfrac{1}{2}\sigma^2\big)\,\Delta t,\ \sigma^2 \Delta t\Big).$$

Parameters. $N, T \in \mathbb{N}$. μ, σ can be scalars or length-T arrays (broadcast across paths). $S_0 > 0$. $dt > 0$. `seed` for RNG.

Returns. $S \in \mathbb{R}^{N\times(T+1)}$ with $S_{\cdot,0} = S_0$.

Implementation. Exact log step:

$$\log S_{t+1} = \log S_t + \big(\mu - \tfrac{1}{2}\sigma^2\big)\Delta t + \sigma\sqrt{\Delta t}\, Z_t, \qquad Z_t \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1).$$

3 `cev`

Signature. `cev(N,T,mu,sigma,beta,S0=100.0,dt=1.0,scheme='milstein', boundary='truncate',seed=None)`

Model. Constant Elasticity of Variance:

$$dS_t = \mu_t S_t\, dt + \sigma_t S_t^{\beta_t} dW_t, \qquad \beta \in \mathbb{R}.$$

Discretizations.

- **Euler–Maruyama**: $S_{t+1} = S_t + \mu_t S_t\,\Delta t + \sigma_t S_t^{\beta_t}\sqrt{\Delta t}\, Z_t$.
- **Milstein** (recommended for state–dependent diffusion):

$$S_{t+1} = S_t + \mu_t S_t\,\Delta t + \sigma_t S_t^{\beta_t}\sqrt{\Delta t}\, Z_t + \tfrac{1}{2}\sigma_t^2\,\beta_t\, S_t^{2\beta_t - 1}\,(\Delta t)\,(Z_t^2 - 1).$$

Boundary handling. After proposing S_{t+1}, apply:

- `'truncate'`: set negative values to a small floor (e.g., $\varepsilon > 0$).
- `'absorb'`: once $S \le 0$, keep it at zero thereafter.
- `'reflect'`: reflect through zero, $S \leftarrow |S|$.

Parameters. $N, T \in \mathbb{N}$. μ, σ, β scalars or length-T. $S_0 > 0$, $dt > 0$. `scheme`$\in\{$`'milstein'`,`'euler'`$\}$. `boundary`$\in\{$`'truncate'`,`'absorb'`,`'reflect'`$\}$. `seed` optional.

Returns. $S \in \mathbb{R}^{N\times(T+1)}$.

Notes. For $\beta = 1$ with Milstein, the scheme reduces toward GBM behavior; for $\beta < 1$ the diffusion weakens as $S \to 0$, making boundary handling important.

4 `merton`

Signature. `merton(N,T,mu,sigma,lamb,mJ,sJ,S0=100.0,dt=1.0,adjust_drift=True,seed=None)`

Model. Merton jump–diffusion:

$$\frac{dS_t}{S_t} = \mu\, dt + \sigma\, dW_t + (J-1)\, dN_t, \qquad \ln J \sim \mathcal{N}(m_J, s_J^2),\ N_t \sim \text{Poisson}(\lambda t).$$

Exact discrete log step

$$\log \frac{S_{t+1}}{S_t} = \underbrace{\left(\mu - \tfrac{1}{2}\sigma^2 - \lambda\kappa\right)\Delta t}_{\text{if adjust\_drift}} + \sigma\sqrt{\Delta t}\, Z_t + \sum_{i=1}^{K_t} Y_i,$$

with $K_t \sim \text{Poisson}(\lambda\Delta t)$, $Y_i \sim \mathcal{N}(m_J, s_J^2)$, and $\kappa = \mathbb{E}[J-1] = e^{m_J + \frac{1}{2}s_J^2} - 1$. When `adjust_drift=False`, the drift term omits $-\lambda\kappa$.

Parameters. $N, T \in \mathbb{N}$. $\mu, \sigma, \lambda, m_J, s_J$ scalars or length-T. $S_0 > 0$, $dt > 0$. `seed` optional.

Returns. $S \in \mathbb{R}^{N\times(T+1)}$.

Notes. The jump sum conditional on $K_t = k$ is Normal with mean $k\, m_J$ and variance $k\, s_J^2$ (additivity of Normals), enabling an efficient draw per step.

5 `variance_gamma`

Signature. `variance_gamma(N, T, dt, S0=100.0, r=0.0, q=0.0, theta=-0.1, sigma=0.2, nu=0.2, seed=None, martingale=True)`

Model. The Variance–Gamma (VG) process is a Brownian motion with drift θ and scale σ evaluated at a gamma subordinator G_t with variance rate ν:

$$G_{\Delta t} \sim \Gamma\Big(\text{shape} = \tfrac{\Delta t}{\nu},\ \text{scale} = \nu\Big), \qquad Z \sim \mathcal{N}(0,1),$$

$$\Delta X = \theta\, G_{\Delta t} + \sigma \sqrt{G_{\Delta t}}\, Z.$$

The (log) price update is

$$\log S_{t+\Delta t} = \log S_t + (r - q + \omega)\,\Delta t + \Delta X,$$

with the martingale correction

$$\omega = \begin{cases} \nu^{-1} \log\big(1 - \theta\nu - \frac{1}{2}\sigma^2\nu\big), & \text{if } \texttt{martingale} = \text{True}, \\ 0, & \text{otherwise.} \end{cases}$$

Setting ω as above enforces $\mathbb{E}[S_t] = S_0 e^{(r-q)t}$ under the risk-neutral measure.

Parameters.

- `martingale` If `True`, applies ω so that the discounted price is a martingale.

Returns. By default the function returns prices $S \in \mathbb{R}^{N\times(T+1)}$ with $S_{\cdot,0} = S_0$. (Optionally, you may also return the VG process X if desired; see the example.)

Notes.

- The increment construction is i.i.d. across steps; the model is pure-jump with infinite activity and finite variance.
- When $\texttt{martingale} = \texttt{True}$, ensure $1 - \theta\nu - \frac{1}{2}\sigma^2\nu > 0$ so that ω is well-defined.

6 `heston`

Signature. `heston(N, T, dt, mu=0.0, kappa=2.0, theta=0.04, xi=0.5, rho=-0.7, S0=100.0, v0=None, seed=None, clip_eps=1e-12)`

Model. Heston (1993) stochastic volatility:

$$\begin{aligned} dS_t &= \mu\, S_t\, dt + \sqrt{v_t}\, S_t\, dW_t^S, \\ dv_t &= \kappa\,(\theta - v_t)\, dt + \xi\,\sqrt{v_t}\, dW_t^v, \qquad \text{corr}\Big(dW_t^S, dW_t^v\Big) = \rho. \end{aligned}$$

Discretization (this implementation).

- **Variance:** Full–Truncation Euler (FTE). Use $v_t^+ = \max(v_t, 0)$ in both drift and diffusion, step v_{t+1}, then floor at 0.
- **Price:** Log–Euler with the same v_t^+: $S_{t+1} = S_t \exp\big((\mu - \frac{1}{2}v_t^+)\Delta t + \sqrt{v_t^+}\,\Delta W_t^S\big)$.
- **Correlation:** Build $\Delta W_t^S = \rho\,\Delta W_t^v + \sqrt{1-\rho^2}\, Z_t$ with independent standard Normal Z_t.

Parameters.

- `clip_eps` Tiny positive floor applied to prices for numerical hygiene.

Returns. $S \in \mathbb{R}^{N\times(T+1)}$ with $S_{\cdot,0} = S_0$.

7 `garch_ret`

Signature. `garch_ret(N,T,omega,alpha,beta,mu=0.0,phi=0.0,dist='normal',nu=8,S0=100.0,v0=None,seed=None)`

Model. Returns follow a mean model with GARCH(1,1) volatility:

$$r_t = \mu + \phi\, r_{t-1} + \sigma_t\, z_t, \qquad z_t \sim \begin{cases} \mathcal{N}(0,1) & \text{if dist} = \text{normal}, \\ t_\nu \text{ (standardized)} & \text{if dist} = \text{t}, \end{cases}$$
$$\sigma_t^2 = \omega + \alpha r_{t-1}^2 + \beta \sigma_{t-1}^2.$$

Stationarity requires $\omega > 0$, $\alpha \geq 0$, $\beta \geq 0$, and $\alpha + \beta < 1$.

Initialization. If `v0` is omitted, the unconditional variance $\sigma_\infty^2 = \omega/(1-\alpha-\beta)$ is used to start σ_0^2. For `dist='t'`, innovations are scaled to unit variance.

Integration to prices. Prices are formed from log returns:

$$S_0 \text{ given}, \quad \log S_{t+1} = \log S_t + r_t \quad \Rightarrow \quad S_{t+1} = S_t\, e^{r_t}.$$

Parameters.

- $N, T \in \mathbb{N}$.
- $\omega, \alpha, \beta > 0$ (real scalars).
- μ (intercept), ϕ (AR(1) coefficient).
- `dist`$\in\{$`'normal'`,`'t'`$\}$; $\nu > 2$ (df for t).
- $S_0 > 0$. Optional `v0` for initial variance. `seed` optional.

Returns. $S \in \mathbb{R}^{N\times(T+1)}$. (You can recover the simulated returns by applying `prices_to_logreturns` .)

Minimal Examples

```
import numpy as np
from simulator import gbm, cev, merton, garch_ret, prices_to_logreturns

# 1) GBM: 5 paths, 252 steps, 10% drift, 20% vol, daily dt
S = gbm(N=5, T=252, mu=0.10, sigma=0.20, S0=100.0, dt=1/252, seed=42)
r = prices_to_logreturns(S, axis=1) # shape (5, 252)

# 2) CEV with Milstein, beta=0.7 and reflective boundary
S_cev = cev(N=3, T=100, mu=0.05, sigma=0.25, beta=0.7,
           S0=50.0, dt=1/12, scheme="milstein", boundary="reflect", seed=7)

# 3) Merton jumps (lambda=0.5/yr, lognormal jumps mJ=-0.05, sJ=0.20)
S_m = merton(N=2, T=252, mu=0.08, sigma=0.15, lamb=0.5,
            mJ=-0.05, sJ=0.20, S0=100.0, dt=1/252, adjust_drift=True, seed=0)
```

```python
# 4) Variance Gamma: daily steps, risk-neutral
S_vg = variance_gamma(N=5, T=252, dt=1/252, S0=100.0, r=0.02, q=0.0,
                      theta=-0.1, sigma=0.2, nu=0.2, seed=7, martingale=True)

# 5) Heston (FTE) with time-homogeneous parameters, daily steps
S_heston = heston(N=5, T=252, dt=1/252, mu=0.08, kappa=2.0, theta=0.04,
                  xi=0.5, rho=-0.7, S0=100.0, v0=0.04, seed=42)

# 6) GARCH(1,1) returns with Student-t innovations, integrated to price
S_g = garch_ret(N=4, T=1000, omega=1e-6, alpha=0.05, beta=0.9,
                mu=0.0, phi=0.0, dist="t", nu=8, S0=100.0, seed=123)
```

Practical Tips

- Use small Δt (e.g., 1/252) for daily steps; prefer Milstein for `cev`.
- For `merton`, `adjust_drift=True` makes the expected growth under the jump component neutral via $-\lambda\kappa$.
- For `garch_ret`, prefer $\alpha + \beta < 1$ well below 1 to avoid extremely persistent volatility; when using `dist='t'`, choose $\nu > 4$ for finite kurtosis of returns.