stochastic Documentation

Release 0.4.0

Christopher Flynn

Dec 28, 2018
# Contents

1 Installation 3

2 Dependencies 5

3 Compatibility 7

4 Performance 9

5 Documentation 11
   5.1 General Usage 11
   5.2 Base Classes 13
   5.3 Discrete-time Processes 14
   5.4 Noise Processes 18
   5.5 Continuous-time Processes 25
   5.6 Diffusion Models 40
   5.7 Bibliographical Sources 43
   5.8 Release Notes 44

6 Indices 47
Stochastic is a python package for generating realizations of stochastic processes.
Stochastic is a python package available on pypi and can be installed using pip:

```
pip install stochastic
```
Stochastic depends on numpy for most calculations and scipy for certain random variable generation.
Stochastic is tested on Python versions 2.7, 3.4, 3.5, and 3.6.
This package uses numpy and scipy wherever possible for faster computation. For improved performance under Monte Carlo simulation, some classes will store results of intermediate computations for faster generation on subsequent simulations.
5.1 General Usage

5.1.1 Processes

This package offers a number of common discrete-time, continuous-time, and noise process objects for generating realizations of stochastic processes as *numpy* arrays.

The diffusion processes are approximated using the Euler–Maruyama method.

Here are the currently supported processes and how to access their classes:

- stochastic
  - continuous
    - BesselProcess
    - BrownianBridge
    - BrownianExcursion
    - BrownianMeander
    - BrownianMotion
    - CauchyProcess
    - FractionalBrownianMotion
    - GammaProcess
    - GeometricBrownianMotion
    - InverseGaussianProcess
    - MixedPoissonProcess
    - MultifractionalBrownianMotion
• PoissonProcess
• SquaredBesselProcess
• VarianceGammaProcess
• WienerProcess

– diffusion
• ConstantElasticityVarianceProcess (or CEVProcess)
• CoxIngersollRossProcess (or CIRProcess)
• OrnsteinUhlenbeckProcess (or OUProcess)
• VasicekProcess

– discrete
• BernoulliProcess
• ChineseRestaurantProcess
• MarkovChain
• MoranProcess
• RandomWalk

– noise
• BlueNoise
• BrownianNoise
• ColoredNoise
• PinkNoise
• RedNoise
• VioletNoise
• WhiteNoise
• FractionalGaussianNoise
• GaussianNoise

5.1.2 Usage patterns

The sample() method

To use stochastic, import the process you want and instantiate with the required parameters. Every process class has a sample method for generating realizations. The sample methods accept a parameter n for the quantity of steps in the realization, but others (Poisson, for instance) may take additional parameters. Parameters can be accessed as attributes of the instance.

```python
from stochastic.discrete import BernoulliProcess

bp = BernoulliProcess(p=0.6)
s = bp.sample(16)
success_probability = bp.p
```
Continuous processes provide a default parameter, \( t \), which indicates the maximum time of the process realizations. The default value is 1. The sample method will generate \( n \) equally spaced increments on the interval \([0, t]\).

**The sample_at() method**

Some continuous processes also provide a `sample_at()` method, in which a sequence of time values can be passed at which the object will generate a realization. This method ignores the parameter, \( t \), specified on instantiation.

```python
from stochastic.continuous import BrownianMotion

bm = BrownianMotion(drift=1, scale=1, t=1)
times = [0, 3, 10, 11, 11.2, 20]
s = sample_at(times)
```

**The times() method**

Continuous-time processes also provide a method `times()` which generates the time values (using `numpy.linspace`) corresponding to a realization of \( n \) steps. This is particularly useful for plotting your samples.

```python
import matplotlib.pyplot as plt
from stochastic.continuous import FractionalBrownianMotion

fbm = FractionalBrownianMotion(hurst=0.7, t=1)
s = fbm.sample(32)
times = fbm.times(32)
plt.plot(times, s)
plt.show()
```

**The algorithm option**

Some processes provide an optional parameter `algorithm`, in which one can specify which algorithm to use to generate the realization using the `sample()` or `sample_at()` methods. See class-specific documentation for implementations.

```python
from stochastic.noise import FractionalGaussianNoise

fgn = FractionalGaussianNoise(hurst=0.6, t=1)
s = fgn.sample(32, algorithm='hosking')
```

## 5.2 Base Classes

Base classes meant to be subclassed or mixed in.

- `stochastic.base.Checks`
- `stochastic.base.Continuous`

### class stochastic.base.Checks

Mix-in class containing input value checking functions.
class stochastic.base.Continuous(t=1)
    Base class to be subclassed to most process classes.
    Contains properties and functions related to times and continuous-time processes.

    sample(*args, **kwargs)
        Sample the process.

        Raises
        -----
        NotImplementedError

    t
        End time of the process.

times(n, zero=True)
    Generate times associated with n increments on [0, t].

    Parameters
    ----------
    * n (int) -- the number of increments
    * zero (bool) -- if True, include \( t = 0 \)

5.3 Discrete-time Processes

The stochastic.discrete module provides classes for generating discrete-time stochastic processes.

- stochastic.discrete.BernoulliProcess
- stochastic.discrete.ChineseRestaurantProcess
- stochastic.discrete.MarkovChain
- stochastic.discrete.MoranProcess
- stochastic.discrete.RandomWalk

class stochastic.discrete.BernoulliProcess(p=0.5)
    Bernoulli process.

    A Bernoulli process consists of a sequence of Bernoulli random variables. A Bernoulli random variable is

    * 1 with probability \( p \)
    * 0 with probability \( 1 - p \)

    Parameters
    ----------
    p -- in \([0, 1]\), the probability of success of each Bernoulli random variable
A Chinese restaurant process consists of a sequence of arrivals of customers to a Chinese restaurant. Customers may be seated either at an occupied table or a new table, there being infinitely many customers and tables.

The first customer sits at the first table. The \( n \)-th customer sits at a new table with probability \( 1/n \), and at each already occupied table with probability \( t_k/n \), where \( t_k \) is the number of customers already seated at table \( k \). This is the canonical process with \( discount = 0 \) and \( strength = 1 \).

The generalized process gives the \( n \)-th customer a probability of \( (strength + T \cdot discount)/(n - 1 + strength) \) to sit at a new table and a probability of \( (t_k - discount)/(n - 1 + strength) \) of sitting at table \( k \). \( T \) is the number of occupied tables.

Samples provide a sequence of tables selected by a sequence of customers.

**Parameters**

- **discount** (*float*) – the discount value of existing tables. Must be strictly less than 1.
- **strength** (*float*) – the strength of a new table. If discount is negative, strength must be a multiple of discount. If discount is nonnegative, strength must be strictly greater than the negative discount.

**discount**

Discount parameter.

**partition_to_sequence** *(partition)*

Create a sequence from a partition.

**Parameters** **partition** – a Chinese restaurant partition.

**sample** *(n)*

Generate a Chinese restaurant process with \( n \) customers.

**Parameters** **n** – the number of customers to simulate.

**sample_partition** *(n)*

Generate a Chinese restaurant process partition.

**Parameters** **n** – the number of customers to simulate.
sequence_to_partition (sequence)
Create a partition from a sequence.

Parameters sequence – a Chinese restaurant sample.

strength
Strength parameter.

class stochastic.discrete.MarkovChain (transition=[[0.5, 0.5], [0.5, 0.5]], initial=None)
Finite state Markov chain.

A Markov Chain which changes between states according to the transition matrix.

Parameters

• transition – a square matrix representing the transition probabilities between states.

• initial – a vector representing the initial state probabilities. If not provided, each state has equal initial probability.

initial
Vector of initial state probabilities.

sample (n)
Generate a realization of the Markov chain.

Parameters n (int) – the number of steps of the Markov chain to generate.

transition
Transition probability matrix.

class stochastic.discrete.MoranProcess (maximum)
Moran process.
A neutral drift Moran process, typically used to model populations. At each step this process will increase by one, decrease by one, or remain at the same value between values of zero and the number of states, \( n \). The process ends when its value reaches zero or the maximum valued state.

**Parameters**

- **maximum** (int) – the maximum possible value for the process.

  - Maximum value.

- **sample** \((n, start)\)

  Generate a realization of the Moran process.

  Generate a Moran process until absorption occurs (state 0 or `maximum`) or length of process reaches length \( n \).

  **Parameters**

  - **n** (int) – the maximum number of steps to generate assuming absorption does not occur.
  - **start** (int) – the initial state of the process.

**class stochastic.discrete.RandomWalk**(steps=[-1, 1], weights=None)

Random walk.

A random walk is a sequence of random steps taken from a set of step sizes with a probability distribution. By default this object defines the steps to be \([-1, 1]\) with probability 1/2 for each possibility.

**Parameters**

- **steps** – a vector of possible deltas to apply at each step.
- **weights** – a corresponding vector of weights associated with each step value. If not provided each step has equal weight/probability.

**P**

Step probabilities, normalized from `weights`.

- **sample** \((n, zero=True)\)

  Generate a sample random walk.

  **Parameters**

  - **n** (int) – the number of steps to generate
  - **zero** (bool) – if True include the step at \( t = 0 \)

- **sample Increments** \((n)\)

  Generate a sample of random walk increments.

  **Parameters**

  - **n** (int) – the number of increments to generate.
steps
Possible steps.
weights
Step weights provided.

5.4 Noise Processes

The stochastic.noise module provides classes for generating noise processes.

Gaussian increments
- stochastic.noise.GaussianNoise
- stochastic.noise.FractionalGaussianNoise

Colored noise
- stochastic.noise.BlueNoise
- stochastic.noise.BrownianNoise
- stochastic.noise.ColoredNoise
- stochastic.noise.RedNoise
- stochastic.noise.PinkNoise
- stochastic.noise.VioletNoise
- stochastic.noise.WhiteNoise

5.4.1 Gaussian increments

Noise processes which are increments of their continuous counterparts.

class stochastic.noise.GaussianNoise(t=1)
Gaussian noise process.

Generate a sequence of Gaussian random variables.

Parameters
- **t** (float) – the right hand endpoint of the time interval [0, t] for the process

---

Generate a realization of Gaussian noise.

Generate a Gaussian noise realization with n increments.
Parameters $n (int)$ – the number of increments to generate.

`sample_at (times)`
Generate Gaussian noise increments at specified times from zero.

**Parameters times** – a vector of increasing time values for which to generate noise increments.

t
End time of the process.

`times (n, zero=True)`
Generate times associated with $n$ increments on $[0, t]$.

**Parameters**
- $n (int)$ – the number of increments
- `zero (bool)` – if True, include $t = 0$

**class stochastic.noise.FractionalGaussianNoise (hurst=0.5, t=1)**
Fractional Gaussian noise process.

Generate sequences of fractional Gaussian noise.

Hosking’s method:

Davies Harte method:

**Parameters**
- `hurst (float)` – The Hurst parameter value in $(0, 1)$.
- `t (float)` – the right hand endpoint of the time interval $[0, t]$ for the process

`hurst`
Hurst parameter.

`sample (n, algorithm='daviesharte')`
Generate a realization of fractional Gaussian noise.

**Parameters**
- $n (int)$ – number of increments to generate
- `algorithm (str)` – either ‘daviesharte’ or ‘hosking’ algorithms
\textbf{t}

End time of the process.

\textbf{times}(n, zero=True)

Generate times associated with n increments on [0, t].

\begin{itemize}
  \item \textbf{n (int)} – the number of increments
  \item \textbf{zero (bool)} – if True, include \( t = 0 \)
\end{itemize}

\subsection{5.4.2 Colored noise}

Signals with spectral densities proportional to the power law.

\begin{itemize}
  \item \textbf{BlueNoise}(t=1)
\end{itemize}

Blue noise.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{blue_noise.png}
\caption{Blue noise}
\end{figure}

Colored noise, or power law noise with spectral density exponent \( \beta = -1 \).

\begin{itemize}
  \item \textbf{t (float)} – the right hand endpoint of the time interval \([0, t]\) for the process
  \item \textbf{sample}(n)
\end{itemize}

Generate a realization of colored noise.

\begin{itemize}
  \item \textbf{n (int)} – the number of increments to generate.
\end{itemize}

\textbf{t}

End time of the process.

\textbf{times}(n, zero=True)

Generate times associated with n increments on [0, t].

\begin{itemize}
  \item \textbf{n (int)} – the number of increments
  \item \textbf{zero (bool)} – if True, include \( t = 0 \)
\end{itemize}

\begin{itemize}
  \item \textbf{BrownianNoise}(t=1)
\end{itemize}

Brownian (red) noise.
Colored noise, or power law noise with spectral density exponent $\beta = 2$.

**Parameters**

$t$ (*float*) – the right hand endpoint of the time interval $[0, t]$ for the process

**sample**(n)

Generate a realization of colored noise.

Generate a colored noise realization with $n$ increments.

**Parameters**

$n$ (*int*) – the number of increments to generate.

$t$

End time of the process.

**times**(n, zero=False)

Generate times associated with $n$ increments on $[0, t]$.

**Parameters**

• $n$ (*int*) – the number of increments
• $zero$ (*bool*) – if True, include $t = 0$

**class** stochastic.noise.ColoredNoise(beta=0, t=1)

Colored noise processes.

Also referred to as power law noise, colored noise refers to noise processes with power law spectral density. That is, their spectral density per unit bandwidth is proportional to $(1/f)^\beta$, where $f$ is frequency with exponent $\beta$.

Uses the algorithm from:

Generates a normalized power-law spectral noise.

Parameters

- **beta** *(float)* – the power law exponent for the spectral density, with 0 being white noise, 1 being pink noise, 2 being red noise (Brownian noise), -1 being blue noise, -2 being violet noise. Default is 0 (white noise).

- **t** *(float)* – the right hand endpoint of the time interval \([0, t]\) for the process

**beta**

Power law exponent.

sample*(n)*

Generate a realization of colored noise.

Generate a colored noise realization with *n* increments.

Parameters

- **n** *(int)* – the number of increments to generate.

**t**

End time of the process.

**times** *(n, zero=True)*

Generate times associated with *n* increments on \([0, t]\).

Parameters

- **n** *(int)* – the number of increments

- **zero** *(bool)* – if True, include \(t = 0\)

class stochastic.noise.RedNoise*(t=1)*

Red (Brownian) noise.

Colored noise, or power law noise with spectral density exponent \(\beta = 2\).

Parameters **t** *(float)* – the right hand endpoint of the time interval \([0, t]\) for the process

**sample** *(n)*

Generate a realization of colored noise.

Generate a colored noise realization with *n* increments.

Parameters **n** *(int)* – the number of increments to generate.

**t**

End time of the process.

**times** *(n, zero=True)*

Generate times associated with *n* increments on \([0, t]\).
Parameters

- **n** (*int*) – the number of increments
- **zero** (*bool*) – if True, include \( t = 0 \)

```python
class stochastic.noise.PinkNoise(t=1)
Pink (flicker) noise.
```

Colored noise, or power law noise with spectral density exponent \( \beta = 1 \).

**Parameters**
- **t** (*float*) – the right hand endpoint of the time interval \([0, t]\) for the process
- **sample** (*n*)
  - Generate a realization of colored noise.
- **times** (*n, zero=True*)
  - Generate times associated with \( n \) increments on \([0, t]\).

```python
t
End time of the process.
```

```python
class stochastic.noise.VioletNoise(t=1)
Violet noise.
```

Colored noise, or power law noise with spectral density exponent \( \beta = -2 \).
Parameters \( t (\text{float}) \) – the right hand endpoint of the time interval \([0, t]\) for the process

\[ \text{sample}(n) \]
Generate a realization of colored noise.
Generate a colored noise realization with \( n \) increments.

Parameters \( n (\text{int}) \) – the number of increments to generate.

\( t \)
End time of the process.

\[ \text{times}(n, \text{zero}=\text{True}) \]
Generate times associated with \( n \) increments on \([0, t]\).

Parameters

- \( n (\text{int}) \) – the number of increments
- \( \text{zero} (\text{bool}) \) – if True, include \( t = 0 \)

\textbf{WhiteNoise}(t=1)
White noise.

Colored noise, or power law noise with spectral density exponent \( \beta = 0 \).

Parameters \( t (\text{float}) \) – the right hand endpoint of the time interval \([0, t]\) for the process

\[ \text{sample}(n) \]
Generate a realization of colored noise.
Generate a colored noise realization with \( n \) increments.

Parameters \( n (\text{int}) \) – the number of increments to generate.

\( t \)
End time of the process.

\[ \text{times}(n, \text{zero}=\text{True}) \]
Generate times associated with \( n \) increments on \([0, t]\).

Parameters

- \( n (\text{int}) \) – the number of increments
- \( \text{zero} (\text{bool}) \) – if True, include \( t = 0 \)
5.5 Continuous-time Processes

The `stochastic.continuous` module provides classes for generating (discretely sampled) continuous-time stochastic processes.

- `stochastic.continuous.BesselProcess`
- `stochastic.continuous.BrownianBridge`
- `stochastic.continuous.BrownianExcursion`
- `stochastic.continuous.BrownianMeander`
- `stochastic.continuous.BrownianMotion`
- `stochastic.continuous.CauchyProcess`
- `stochastic.continuous.FractionalBrownianMotion`
- `stochastic.continuous.GammaProcess`
- `stochastic.continuous.GeometricBrownianMotion`
- `stochastic.continuous.InverseGaussianProcess`
- `stochastic.continuous.MixedPoissonProcess`
- `stochastic.continuous.MultifractionalBrownianMotion`
- `stochastic.continuous.PoissonProcess`
- `stochastic.continuous.SquaredBesselProcess`
- `stochastic.continuous.VarianceGammaProcess`
- `stochastic.continuous.WienerProcess`

```python
class stochastic.continuous.BesselProcess(dim=1, t=1)
Bessel process.
```

The Bessel process is the Euclidean norm of an \( n \)-dimensional Wiener process, e.g. \( \| W_t \| \).

Generate Bessel process realizations using `dim` independent Brownian motion processes on the interval \([0, t]\).

Parameters
- `dim (int)` – the number of underlying independent Brownian motions to use
- `t (float)` – the right hand endpoint of the time interval \([0, t]\) for the process

```python
sample(n, zero=True)
Generate a realization.
```
Parameters

- \( n (\text{int}) \) – the number of increments to generate
- \( \text{zero} (\text{bool}) \) – if True, include \( t = 0 \)

\texttt{sample\_at}(\texttt{times})

Generate a realization using specified times.

Parameters \texttt{times} – a vector of increasing time values at which to generate the realization

\( t \)

End time of the process.

\texttt{times}(\texttt{n, zero=\text{True}})

Generate times associated with \( n \) increments on \([0, t]\).

Parameters

- \( n (\text{int}) \) – the number of increments
- \( \text{zero} (\text{bool}) \) – if True, include \( t = 0 \)

\texttt{class}\ \texttt{stochastic\_continuous.BrownianBridge}(b=0, t=1)

Brownian bridge.

\[ \text{Brownian bridge (b=0)} \]

A Brownian bridge is a Brownian motion with a conditional value on the right endpoint of the process.

Parameters

- \( b (\text{float}) \) – the right endpoint value of the Brownian bridge at time \( t \)
- \( t (\text{float}) \) – the right hand endpoint of the time interval \([0, t]\) for the process

\( b \)

Right endpoint value.

\texttt{sample}(\texttt{n, zero=\text{True}})

Generate a realization.

Parameters

- \( n (\text{int}) \) – the number of increments to generate
- \( \text{zero} (\text{bool}) \) – if True, include \( t = 0 \)

\texttt{sample\_at}(\texttt{times, b=\text{None}})

Generate a realization using specified times.

Parameters

- \( \texttt{times} \) – a vector of increasing time values at which to generate the realization
• \( b (\text{float}) \) – the right endpoint value for \( \text{times} [-1] \)

\[
t
\]
End time of the process.

\textbf{times} \( (n, \text{zero}=\text{True}) \)
Generate times associated with \( n \) increments on \([0, t]\).

\textbf{Parameters}

• \( n (\text{int}) \) – the number of increments

• \( \text{zero} (\text{bool}) \) – if True, include \( t = 0 \)

\textbf{class} \ stochastic.continuous.BrownianExcursion \( (t=1) \)
Brownian excursion.

A Brownian excursion is a Brownian bridge from \((0, 0)\) to \((t, 0)\) which is conditioned to be nonnegative on the interval \([0, t]\).

Generated using method by


\textbf{Parameters} \( t (\text{float}) \) – the right hand endpoint of the time interval \([0, t]\) for the process

\textbf{sample} \( (n, \text{zero}=\text{True}) \)
Generate a realization.

\textbf{Parameters}

• \( n (\text{int}) \) – the number of increments to generate.

• \( \text{zero} (\text{bool}) \) – if True, include \( t = 0 \)

\textbf{sample_at} \( (\text{times}) \)
Generate a realization using specified times.

\textbf{Parameters} \( \text{times} \) – a vector of increasing time values at which to generate the realization

\[
t
\]
End time of the process.

\textbf{times} \( (n, \text{zero}=\text{True}) \)
Generate times associated with \( n \) increments on \([0, t]\).
class stochastic.continuous.BrownianMeander(t=1)
Brownian meander process.

A Brownian motion conditioned such that the process is nonnegative.

Generated using method by


Parameters

- **t** *(float)* – the right hand endpoint of the time interval \([0, t]\) for the process

**sample**(n, b=None, zero=True)
Generate a realization.

Parameters

- **n** *(int)* – the number of increments to generate
- **b** *(float)* – the nonnegative right hand endpoint of the meander. If not provided, one is randomly selected from a \(\sqrt{2E}\) random variable where \(E\) is exponential.
- **zero** *(bool)* – if True, include time \(t = 0\)

**sample_at**(times, b=None)
Generate a realization using specified times.

Parameters

- **times** – a vector of increasing time values at which to generate the realization
- **b** *(float)* – the right endpoint value for **times**[-1]. If not provided, one is randomly selected from a \(\sqrt{2E}\) random variable where \(E\) is exponential and \(t\) is **times**[-1].

**t**
End time of the process.

**times**(n, zero=True)
Generate times associated with n increments on \([0, t]\).

Parameters
• \( n \) (int) – the number of increments
• \( \text{zero} \) (bool) – if True, include \( t = 0 \)

```python
class stochastic.continuous.BrownianMotion(drift=0, scale=1, t=1)
```
Brownian motion.

A standard Brownian motion (discretely sampled) has independent and identically distributed Gaussian increments with variance equal to increment length. Non-standard Brownian motion includes a linear drift parameter and scale factor.

**Parameters**
- **drift** (float) – rate of change of the expected value
- **scale** (float) – scale factor of the Gaussian process
- **\( t \)** (float) – the right hand endpoint of the time interval \([0, t]\) for the process

```
drift
Drift parameter.
```

```
sample \((n, \text{zero}=\text{True})\)
Generate a realization.
```

**Parameters**
- **\( n \)** (int) – the number of increments to generate
- **\( \text{zero} \)** (bool) – if True, include \( t = 0 \)

```
sample_at \((\text{times})\)
Generate a realization using specified times.
```

**Parameters**
- **\( \text{times} \)** – a vector of increasing time values at which to generate the realization

```
scale
Scale parameter.
```

```
t
End time of the process.
```

```
times \((n, \text{zero}=\text{True})\)
Generate times associated with \( n \) increments on \([0, t]\).
```

**Parameters**
- **\( n \)** (int) – the number of increments
- **\( \text{zero} \)** (bool) – if True, include \( t = 0 \)
The symmetric Cauchy process is a Brownian motion with a Levy subordinator using location parameter 0 and scale parameter $t^2/2$.

**Parameters**

- **t (float)** – the right hand endpoint of the time interval $[0, t]$ for the process

**sample (n, zero=True)**

Generate a realization.

**Parameters**

- **n (int)** – the number of increments to generate.
- **zero (bool)** – if True, include $t = 0$

**sample_at (times)**

Generate a realization using specified times.

**Parameters**

- **times** – a vector of increasing time values at which to generate the realization

**t**

End time of the process.

**times (n, zero=True)**

Generate times associated with n increments on $[0, t]$.

**Parameters**

- **n (int)** – the number of increments
- **zero (bool)** – if True, include $t = 0$
A fractional Brownian motion (discretely sampled) has correlated Gaussian increments defined by Hurst parameter $H$. When $H = 1/2$, the process is a standard Brownian motion. When $H > 1/2$, the increments are positively correlated. When $H < 1/2$, the increments are negatively correlated.

Hosking’s method:


Davies Harte method:


Parameters

- **hurst (float)** – the Hurst parameter on the interval (0, 1)
- **t (float)** – the right hand endpoint of the time interval $[0, t]$ for the process

`hurst`  
Hurst parameter.

`sample (n, zero=True)`  
Generate a realization.

Parameters

- **n (int)** – the number of increments to generate
- **zero (bool)** – if True, include $t = 0$

`t`  
End time of the process.

`times (n, zero=True)`  
Generate times associated with $n$ increments on $[0, t]$.

Parameters

- **n (int)** – the number of increments
- **zero (bool)** – if True, include $t = 0$

`class stochastic.continuous.GammaProcess (mean=None, variance=None, rate=None, scale=None, t=1)`  
Gamma process.
A Gamma process (discretely sampled) is the summation of stationary independent increments which are distributed as gamma random variables. This class supports instantiation using the mean/variance parametrization or the rate/scale parametrization.

**Parameters**

- **mean** (*float*) – mean increase per unit time; supply with **variance**
- **variance** (*float*) – variance of increase per unit time; supply with **mean**
- **rate** (*float*) – the rate of jump arrivals; supply with **scale**
- **scale** (*float*) – the size of the jumps; supplie with **rate**
- **t** (*float*) – the right hand endpoint of the time interval \([0, t]\) for the process

**mean**
Mean increase per unit time.

**rate**
Rate of jump arrivals.

**sample** \((n, \text{zero}=\text{True})\)
Generate a realization.

**Parameters**

- **n** (*int*) – the number of increments to generate
- **zero** (*bool*) – if True, include \(t = 0\)

**sample_at** \((\text{times})\)
Generate a realization at specified times.

**Parameters**

- **times** (*int*) – the times at which to generate the realization

**scale**
Scale parameter for jump sizes.

**t**
End time of the process.

**times** \((n, \text{zero}=\text{True})\)
Generate times associated with \(n\) increments on \([0, t]\).

**Parameters**

- **n** (*int*) – the number of increments
- **zero** (*bool*) – if True, include \(t = 0\)
A geometric Brownian motion $S_t$ is the analytic solution to the stochastic differential equation with Wiener process $W_t$:

$$dS_t = \mu S_t dt + \sigma S_t dW_t$$

and can be represented with initial value $S_0$ in the form:

$$S_t = S_0 \exp \left( \left( \mu - \frac{\sigma^2}{2} \right) t + \sigma W_t \right)$$

**Parameters**

- **drift** *(float)* – the parameter $\mu$.
- **volatility** *(float)* – the parameter $\sigma$.
- **t** *(float)* – the right hand endpoint of the time interval $[0, t]$ for the process.

**Sample** *(n, initial=1, zero=True)*

Generate a realization.

**Parameters**

- **n** *(int)* – the number of increments to generate.
- **initial** *(float)* – the initial value of the process $S_0$.
- **zero** *(bool)* – if True, include $t = 0$.

**Sample at** *(times, initial=1)*

Generate a realization using specified times.

**Parameters**

- **times** – a vector of increasing time values at which to generate the realization.

**T**

End time of the process.

**Times** *(n, zero=True)*

Generate times associated with n increments on $[0, t]$.

**Parameters**
• **n (int)** – the number of increments

• **zero (bool)** – if True, include \( t = 0 \)

**volatility**
Geometric Brownian motion volatility parameter.

class stochastic.continuous.InverseGaussianProcess (mean=None, scale=1, t=1)
Inverse Gaussian process.

An inverse Gaussian process has independent increments which follow an inverse Gaussian distribution with parameters defined by a monotonically increasing function, \( \Gamma(t) \). E.g. for increment \([s, t]\):

\[
\mathcal{IG}(\Gamma(t) - \Gamma(s), \eta(\Gamma(t) - \Gamma(s))^2)
\]

Uses a method for generating inverse Gaussian variates from:


**Parameters**

• **mean (callable)** – a callable with one argument \( \Gamma(t) \) such that \( \Gamma(t') > \Gamma(t) \) \( \forall t' > t \). Default is the identity function.

• **scale (float)** – scale factor of the shape parameter of the inverse gaussian, or \( \eta \) from the above equation.

• **t (float)** – the right hand endpoint of the time interval \([0, t]\) for the process

**mean**
Mean function.

**sample (n, zero=True)**
Generate a realization.

**Parameters**

• **n (int)** – the number of increments to generate

• **zero (bool)** – if True, include \( t = 0 \)

**sample_at (times)**
Generate a realization using specified times.

**Parameters**

• **times** – a vector of increasing time values at which to generate the realization

**scale**
Scale parameter.
t
End time of the process.

times \(n, \text{zero}=\text{True}\)
Generate times associated with \(n\) increments on \([0, t]\).

Parameters

- \(n\) (int) – the number of increments
- \(\text{zero}\) (bool) – if True, include \(t = 0\)

class stochastic.continuous.MixedPoissonProcess(rate_func, rate_args=(), rate_kwargs={})
Mixed poisson process.

A mixed poisson process is a Poisson process for which the rate is a scalar random variate. The sample method will generate a random variate for the rate before generating a Poisson process realization with the rate. A Poisson process with rate \(\lambda\) is a count of occurrences of i.i.d. exponential random variables with mean \(1/\lambda\). Use the \(\text{rate}\) attribute to get the most recently generated random rate.

Parameters

- \(\text{rate\_func}\) (callable) – a callable to generate variates of the random rate
- \(\text{rate\_args}\) (tuple) – positional args for \(\text{rate\_func}\)
- \(\text{rate\_kwargs}\) (dict) – keyword args for \(\text{rate\_func}\)

rate
The most recently generated rate.

Attempting to get the rate prior to generating a sample will raise an AttributeError.

rate_args
Positional arguments for the rate function.

rate_func
Current rate’s distribution.

rate_kwargs
Keyword arguments for the rate function.

sample \(n=\text{None}, length=\text{None}, \text{zero}=\text{True}\)
Generate a realization.

Exactly one of \(n\) and \(length\) must be provided. Generates a random variate for the rate, then generates a Poisson process realization using this rate.

Parameters
A multifractional Brownian motion generalizes a fractional Brownian motion with a Hurst parameter which is a function of time, \( h(t) \). If the Hurst is constant, the process is a fractional Brownian motion. If Hurst is constant equal to 0.5, the process is a Brownian motion.

Approximate method originally proposed for fBm in

Adapted to approximate mBm in

**Parameters**

- **hurst (float)** – a callable with one argument \( h(t) \) such that \( h(t') \in (0, 1) \forall t' \in [0, t] \). Default is \( h(t) = 0.5 \).
- **t (float)** – the right hand endpoint of the time interval \([0, t]\) for the process

**hurst**
Hurst function.

**sample (n, zero=True)**
Generate a realization.

**Parameters**

- **n (int)** – the number of increments to generate
- **zero (bool)** – if True, include \( t = 0 \)

**t**
End time of the process.

**times (n, zero=True)**
Generate times associated with \( n \) increments on \([0, t]\).

**Parameters**
• \(n\) (int) – the number of increments
• \(zero\) (bool) – if True, include \(t = 0\)

class stochastic.continuous.PoissonProcess(rate=1)
Poisson process.

A Poisson process with rate \(\lambda\) is a count of occurrences of i.i.d. exponential random variables with mean \(1/\lambda\).

This class generates samples of times for which cumulative exponential random variables occur.

Parameters rate (float) – the parameter \(\lambda\) which defines the rate of occurrences of the process

rate
Rate parameter.

sample (n=None, length=None, zero=True)
Generate a realization.

Exactly one of \(n\) and \(length\) must be provided.

Parameters

• \(n\) (int) – the number of arrivals to simulate
• \(length\) (int) – the length of time to simulate; will generate arrivals until length is met or exceeded.
• \(zero\) (bool) – if True, include \(t = 0\)

class stochastic.continuous.SquaredBesselProcess(dim=1, t=1)
Squared Bessel process.

The square of a Bessel process: \(\|W_t\|^2\).

The Bessel process is the Euclidean norm of an \(n\)-dimensional Wiener process, e.g. \(\|W_t\|\)
Parameters

- `dim (int)` – the number of underlying independent Brownian motions to use
- `t (float)` – the right hand endpoint of the time interval \([0, t]\) for the process

`dim`
Dimensions, or independent Brownian motions.

`sample (n, zero=True)`
Generate a realization.

Parameters

- `n (int)` – the number of increments to generate
- `zero (bool)` – if True, include \(t = 0\)

`sample_at (times)`
Generate a realization using specified times.

Parameters `times` – a vector of increasing time values at which to generate the realization

`t`
End time of the process.

```python
class stochastic.continuous.VarianceGammaProcess (drift=0, variance=1, scale=1, t=1)
Variance Gamma process.
```

A variance gamma process has independent increments which follow the variance-gamma distribution. It can be represented as a Brownian motion with drift subordinated by a Gamma process:

\[
\theta \Gamma(t; 1, \nu) + \sigma W(\Gamma(t; 1, \nu))
\]

Parameters

- `drift (float)` – the drift parameter of the Brownian motion, or \(\theta\) above
- `variance (float)` – the variance parameter of the Gamma subordinator, or \(\nu\) above
- `scale (float)` – the scale parameter of the Brownian motion, or \(\sigma\) above
- `t (float)` – the right hand endpoint of the time interval \([0, t]\) for the process

`drift`
Drift parameter.

`sample (n, zero=True)`
Generate a realization.

Parameters
• n (int) – the number of increments to generate  
• zero (bool) – if True, include t = 0

sample_at (times)  
Generate a realization using specified times.

Parameters times – a vector of increasing time values at which to generate the realization

scale  
Scale parameter.

t  
End time of the process.

variance  
Variance parameter.

class stochastic.continuous.WienerProcess (t=1)  
Wiener process, or standard Brownian motion.

Parameters t (float) – the right hand endpoint of the time interval [0, t] for the process

sample (n, zero=True)  
Generate a realization.

Parameters  
• n (int) – the number of increments to generate  
• zero (bool) – if True, include t = 0

sample_at (times)  
Generate a realization using specified times.

Parameters times – a vector of increasing time values at which to generate the realization

t  
End time of the process.

times (n, zero=True)  
Generate times associated with n increments on [0, t].

Parameters  
• n (int) – the number of increments  
• zero (bool) – if True, include t = 0
5.6 Diffusion Models

The `stochastic.diffusion` module provides classes for generating (discretely sampled) continuous-time diffusion processes using the Euler–Maruyama method.

- `stochastic.diffusion.ConstantElasticityVarianceProcess`
- `stochastic.diffusion.CoxIngersollRossProcess`
- `stochastic.diffusion.OrnsteinUhlenbeckProcess`
- `stochastic.diffusion.VasicekProcess`

```python
class stochastic.diffusion.ConstantElasticityVarianceProcess(mu=1, sigma=1, gamma=1, t=1):
    Constant elasticity of variance process.
```

The process $X_t$ that satisfies the following stochastic differential equation with Wiener process $W_t$:

$$dX_t = \mu X_t dt + \sigma X_t^\gamma dW_t$$

Realizations are generated using the Euler-Maruyama method.

Parameters
- `mu` (`float`) – the drift coefficient, or $\mu$ above
- `sigma` (`float`) – the volatility coefficient, or $\sigma$ above
- `gamma` (`float`) – the volatility-price exponent, or $\gamma$ above
- `t` (`float`) – the right hand endpoint of the time interval $[0, t]$ for the process

```python
sample(n, initial=1, zero=True)
```

Generate a realization.

Parameters
- `n` (`int`) – the number of increments to generate
- `initial` (`float`) – the initial value of the process
- `zero` (`bool`) – if True, include $t = 0$

`t` End time of the process.

```python
times(n, zero=True)
```

Generate times associated with $n$ increments on $[0, t]$. 
Parameters

- \( n (\text{int}) \) – the number of increments
- \( \text{zero} (\text{bool}) \) – if True, include \( t = 0 \)

class stochastic.diffusion.CEVProcess(mu=1, sigma=1, gamma=1, t=1)

Alias for ConstantElasticityVarianceProcess.

class stochastic.diffusion.CoxIngersollRossProcess(speed=1, mean=1, vol=1, t=1)

Cox-Ingersoll-Ross process. A model for instantaneous interest rate.

The process \( X_t \) that satisfies the following stochastic differential equation with Wiener process \( W_t \):

\[
dX_t = \theta X_t(\mu - t) dt + \sigma \sqrt{X_t} dW_t
\]

Realizations are generated using the Euler-Maruyama method.

Parameters

- \( \text{speed} (\text{float}) \) – the speed of reversion, or \( \theta \) above
- \( \text{mean} (\text{float}) \) – the mean of the process, or \( \mu \) above
- \( \text{vol} (\text{float}) \) – volatility coefficient of the process, or \( \sigma \) above
- \( t (\text{float}) \) – the right hand endpoint of the time interval \([0, t]\) for the process

sample \((n, \text{initial}=1, \text{zero}=True)\)

Generate a realization.

Parameters

- \( n (\text{int}) \) – the number of increments to generate
- \( \text{initial} (\text{float}) \) – the initial value of the process
- \( \text{zero} (\text{bool}) \) – if True, include \( t = 0 \)

\( t \)

End time of the process.

times \((n, \text{zero}=True)\)

Generate times associated with \( n \) increments on \([0, t] \).

Parameters

- \( n (\text{int}) \) – the number of increments
- \( \text{zero} (\text{bool}) \) – if True, include \( t = 0 \)
class stochastic.diffusion.CIRProcess(speed=1, mean=1, vol=1, t=1)
Alias for CoxIngersollRossProcess.

class stochastic.diffusion.OrnsteinUhlenbeckProcess(speed=1, mean=1, vol=1, t=1)
Ornstein-Uhlenbeck process.

The process $X_t$ that satisfies the following stochastic differential equation with Wiener process $W_t$:

$$dX_t = \theta X_t(\mu - t)dt + \sigma dW_t$$

Realizations are generated using the Euler-Maruyama method.

Parameters

- **speed** (float) – the speed of reversion, or $\theta$ above
- **mean** (float) – the mean of the process, or $\mu$ above
- **vol** (float) – volatility coefficient of the process, or $\sigma$ above
- **t** (float) – the right hand endpoint of the time interval $[0, t]$ for the process

```python
sample(n, initial=1, zero=True)
```
Generate a realization.

Parameters

- **n** (int) – the number of increments to generate
- **initial** (float) – the initial value of the process
- **zero** (bool) – if True, include $t = 0$

```
t
```
End time of the process.

```python
times(n, zero=True)
```
Generate times associated with $n$ increments on $[0, t]$.

Parameters

- **n** (int) – the number of increments
- **zero** (bool) – if True, include $t = 0$

class stochastic.diffusion.OUProcess(speed=1, mean=1, vol=1, t=1)
Alias for OrnsteinUhlenbeckProcess.

class stochastic.diffusion.VasicekProcess(speed=1, mean=1, vol=1, t=1)
Vasicek process.
A model for instantaneous interest rate.

The Vasicek process $X_t$ that satisfies the following stochastic differential equation with Wiener process $W_t$:

$$dX_t = \theta X_t(\mu - t)\,dt + \sigma dW_t$$

Realizations are generated using the Euler-Maruyama method.

**Parameters**

- **speed** (float) – the speed of reversion, or $\theta$ above
- **mean** (float) – the mean of the process, or $\mu$ above
- **vol** (float) – volatility coefficient of the process, or $\sigma$ above
- **t** (float) – the right hand endpoint of the time interval $[0, t]$ for the process

**sample** ($n$, initial=1, zero=True)

Generate a realization.

**Parameters**

- **n** (int) – the number of increments to generate
- **initial** (float) – the initial value of the process
- **zero** (bool) – if True, include $t = 0$

**t**

End time of the process.

**times** ($n$, zero=True)

Generate times associated with $n$ increments on $[0, t]$.

**Parameters**

- **n** (int) – the number of increments
- **zero** (bool) – if True, include $t = 0$

### 5.7 Bibliographical Sources


5.8 Release Notes

5.8.1 Contributing

Stochastic is an open source python package.

If you have additional processes, generalizations, or algorithms that you think would be suitable for this package, please let me know on this project’s GitHub page.

5.8.2 License

MIT License

Copyright (c) 2018 Christopher Flynn

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
5.8.3 Contributors

Author

• crflynn

Contributions

• Gabinou

5.8.4 Release History

0.4.0 (2018-08-19)

• Added a MixedPoissonProcess (thanks to Gabinou)

0.3.0 (2018-07-22)

• Introduced breaking changes that move the t argument of all processes to the end of the __init__ signature
• Added support for inverse Gaussian process

0.2.0 (2018-07-11)

• Added support for colored noise processes (generalized power law, violet, blue, white, pink, red/Brownian)
• Added support for multifractional brownian motion
• Added more citations and bibliographical source page to docs

0.1.0 (2018-01-04)

• First release.
• Support for multiple continuous-time, discrete-time, diffusion, and noise processes.
Indices

- genindex
- modindex
- search
Index

B
b (stochastic.continuous.BrownianBridge attribute), 26
BernoulliProcess (class in stochastic.discrete), 14
BesselProcess (class in stochastic.continuous), 25
beta (stochastic.noise.ColoredNoise attribute), 22
BlueNoise (class in stochastic.noise), 20
BrownianBridge (class in stochastic.continuous), 26
BrownianExcursion (class in stochastic.continuous), 27
BrownianMeander (class in stochastic.continuous), 28
BrownianMotion (class in stochastic.continuous), 29
BrownianNoise (class in stochastic.noise), 20
CauchyProcess (class in stochastic.continuous), 29
CEVProcess (class in stochastic.diffusion), 41
Checks (class in stochastic.base), 13
ChineseRestaurantProcess (class in stochastic.discrete), 15
CIRProcess (class in stochastic.diffusion), 41
ColoredNoise (class in stochastic.noise), 21
ConstantElasticityVarianceProcess (class in stochastic.diffusion), 40
Continuous (class in stochastic.base), 14
CoxIngersollRossProcess (class in stochastic.diffusion), 41

D
dim (stochastic.continuous.SquaredBesselProcess attribute), 38
discount (stochastic.discrete.ChineseRestaurantProcess attribute), 15
drift (stochastic.continuous.BrownianMotion attribute), 29
drift (stochastic.continuous.GeometricBrownianMotion attribute), 33
drift (stochastic.continuous.VarianceGammaProcess attribute), 38

F
FractionalBrownianMotion (class in stochastic.continuous), 30
FractionalGaussianNoise (class in stochastic.noise), 19
GammaProcess (class in stochastic.continuous), 31
GaussianNoise (class in stochastic.noise), 18
GeometricBrownianMotion (class in stochastic.continuous), 33
hurst (stochastic.continuous.FractionalBrownianMotion attribute), 31
hurst (stochastic.continuous.MultifractionalBrownianMotion attribute), 36
hurst (stochastic.noise.FractionalGaussianNoise attribute), 19
initial (stochastic.discrete.MarkovChain attribute), 16
InverseGaussianProcess (class in stochastic.continuous), 34
MarkovChain (class in stochastic.discrete), 16
maximum (stochastic.discrete.MoranProcess attribute), 17
mean (stochastic.continuous.GammaProcess attribute), 32
mean (stochastic.continuous.InverseGaussianProcess attribute), 34
MixedPoissonProcess (class in stochastic.continuous), 35
MoranProcess (class in stochastic.discrete), 16
MultifractionalBrownianMotion (class in stochastic.continuous), 36

O
OrnsteinUhlenbeckProcess (class in stochastic.diffusion), 42
OUProcess (class in stochastic.diffusion), 42
Index
Index

T

- sample_increments() (stochastic.discrete.RandomWalk method), 17
- sample_partition() (stochastic.discrete.ChineseRestaurantProcess method), 15
- scale (stochastic.continuous.BrownianMotion attribute), 29
- scale (stochastic.continuous.GammaProcess attribute), 32
- scale (stochastic.continuous.InverseGaussianProcess attribute), 34
- scale (stochastic.continuous.VarianceGammaProcess attribute), 39
- sequence_to_partition() (stochastic.discrete.ChineseRestaurantProcess method), 15
- SquaredBesselProcess (class in stochastic.continuous), 37
- steps (stochastic.discrete.RandomWalk attribute), 17
- strength (stochastic.discrete.ChineseRestaurantProcess attribute), 16
- T
  - t (stochastic.base.Continuous attribute), 14
  - t (stochastic.continuous.BesselProcess attribute), 26
  - t (stochastic.continuous.BrownianBridge attribute), 27
  - t (stochastic.continuous.BrownianExcursion attribute), 27
  - t (stochastic.continuous.BrownianMeander attribute), 28
  - t (stochastic.continuous.BrownianMotion attribute), 29
  - t (stochastic.continuous.CauchyProcess attribute), 30
  - t (stochastic.continuous.FractionalBrownianMotion attribute), 31
  - t (stochastic.continuous.GammaProcess attribute), 32
  - t (stochastic.continuous.GeometricBrownianMotion attribute), 32
  - t (stochastic.continuous.InverseGaussianProcess attribute), 34
  - t (stochastic.continuous.MultifractionalBrownianMotion attribute), 36
  - t (stochastic.continuous.SquaredBesselProcess attribute), 38
  - t (stochastic.continuous.VarianceGammaProcess attribute), 39
  - t (stochastic.continuous.WienerProcess attribute), 39
- times() (stochastic.base.Continuous method), 14
- times() (stochastic.continuous.BesselProcess method), 27
- times() (stochastic.continuous.BrownianBridge method), 27
- times() (stochastic.continuous.BrownianExcursion method), 27
- times() (stochastic.continuous.BrownianMeander method), 28
- times() (stochastic.continuous.BrownianMotion method), 29
- times() (stochastic.continuous.CauchyProcess method), 30
- times() (stochastic.continuous.FractionalBrownianMotion method), 31
- times() (stochastic.continuous.GammaProcess method), 32
- times() (stochastic.continuous.GeometricBrownianMotion method), 33
- times() (stochastic.continuous.InverseGaussianProcess method), 35
- times() (stochastic.continuous.MultifractionalBrownianMotion method), 36
- times() (stochastic.continuous.WienerProcess method), 39
- times() (stochastic.diffusion.ConstantElasticityVarianceProcess method), 40
- times() (stochastic.diffusion.CoxIngersollRossProcess method), 41
- times() (stochastic.diffusion.OrnsteinUhlenbeckProcess method), 42
- times() (stochastic.diffusion.VasicekProcess method), 43
- times() (stochastic.noise.BlueNoise method), 20
- times() (stochastic.noise.BrownianNoise method), 21
- times() (stochastic.noise.ColoredNoise method), 22
- times() (stochastic.noise.FractionalGaussianNoise method), 20
- times() (stochastic.noise.GaussianNoise method), 19
- times() (stochastic.noise.PinkNoise method), 23
- times() (stochastic.noise.RedNoise method), 22
- times() (stochastic.noise.VioletNoise method), 24
- times() (stochastic.noise.WhiteNoise method), 24
- transition (stochastic.discrete.MarkovChain attribute), 16

V

- variance (stochastic.continuous.GammaProcess attribute), 32
- variance (stochastic.continuous.VarianceGammaProcess attribute), 39
- VarianceGammaProcess (class in stochastic.continuous), 38
VasicekProcess (class in stochastic.diffusion), 42
VioletNoise (class in stochastic.noise), 23
volatility (stochastic.continuous.GeometricBrownianMotion
        attribute), 34

W
weights (stochastic.discrete.RandomWalk attribute), 18
WhiteNoise (class in stochastic.noise), 24
WienerProcess (class in stochastic.continuous), 39