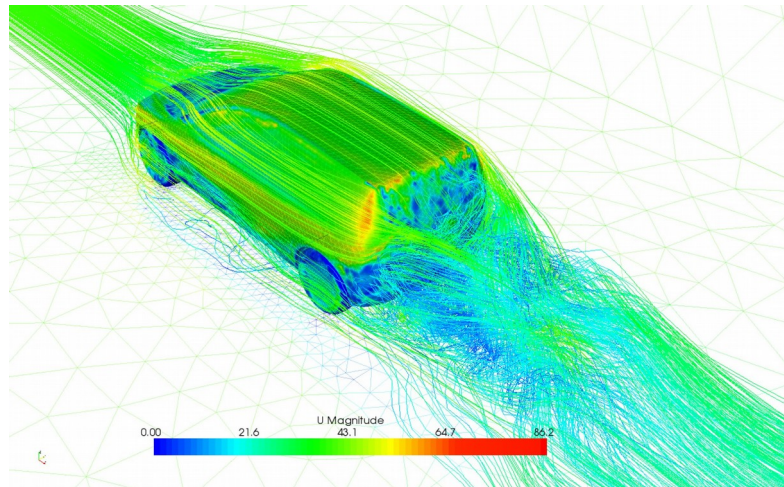


Equação diferencial estocástica Navier-Stokes e p-model



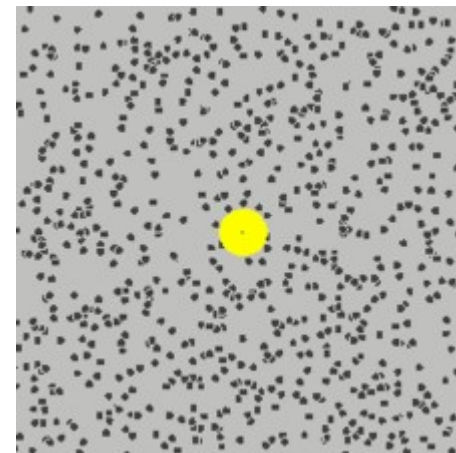
Equação diferencial estocástica

É uma equação diferencial em que uma ou mais variáveis são um processo estocástico, resultando numa solução que também é um processo estocástico.

São usadas para modelar diversos fenômenos.

Os primeiros trabalhos sobre equações diferenciais estocásticas descrevem o movimento Browniano (movimento aleatório das partículas suspensas num fluido).

Exemplo de equação diferencial estocástica: Navier-Stokes (escoamento de fluidos).



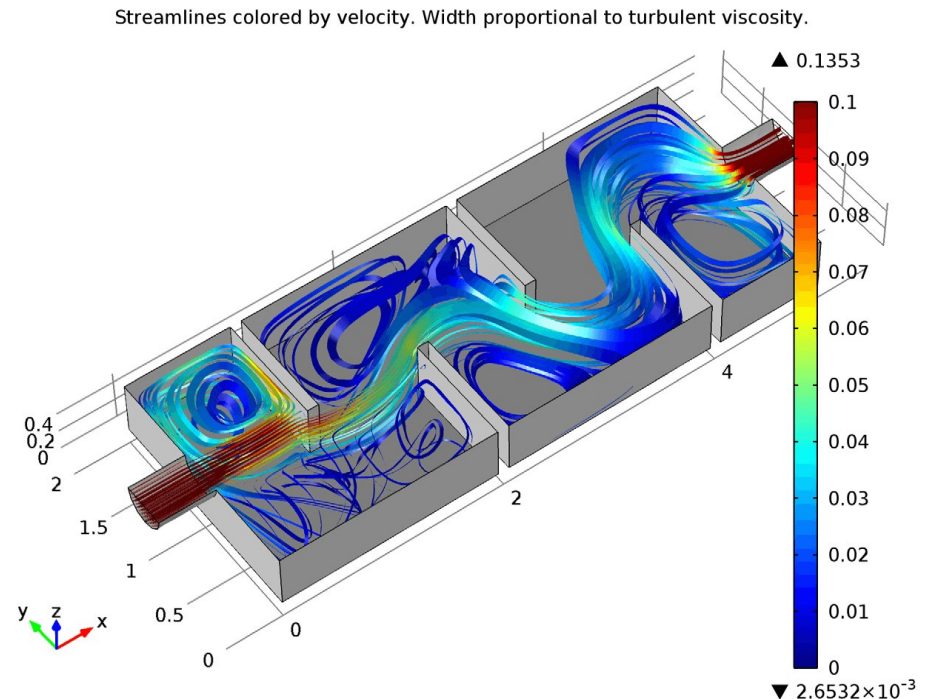
Navier-Stokes

São equações diferenciais que descrevem o escoamento de fluídos.

Derivadas parciais para determinar pressão e velocidade em um escoamento.

São usadas para modelar:

- o clima
- correntes oceânicas
- fluxos da água em oceanos, estuários, lagos e rios
- movimentos das estrelas dentro e fora da galáxia
- fluxo ao redor de aerofólios (asas) de automóveis e de aviões
- propagação de fumaça em incêndios e em chaminés industriais (dispersão)



Navier-Stokes

Fluxo em fluidos incompressíveis

$$\rho \left(\frac{\partial u}{\partial t} + u \cdot \nabla u \right) = \nabla P + \mu \nabla^2 u + \rho F$$

Navier-Stokes

Massa

Aceleração

Força

$$\rho \left(\frac{\partial u}{\partial t} + u \cdot \nabla u \right) = \underbrace{\nabla P + \mu \nabla^2 u + \rho F}_{\text{Força}}$$

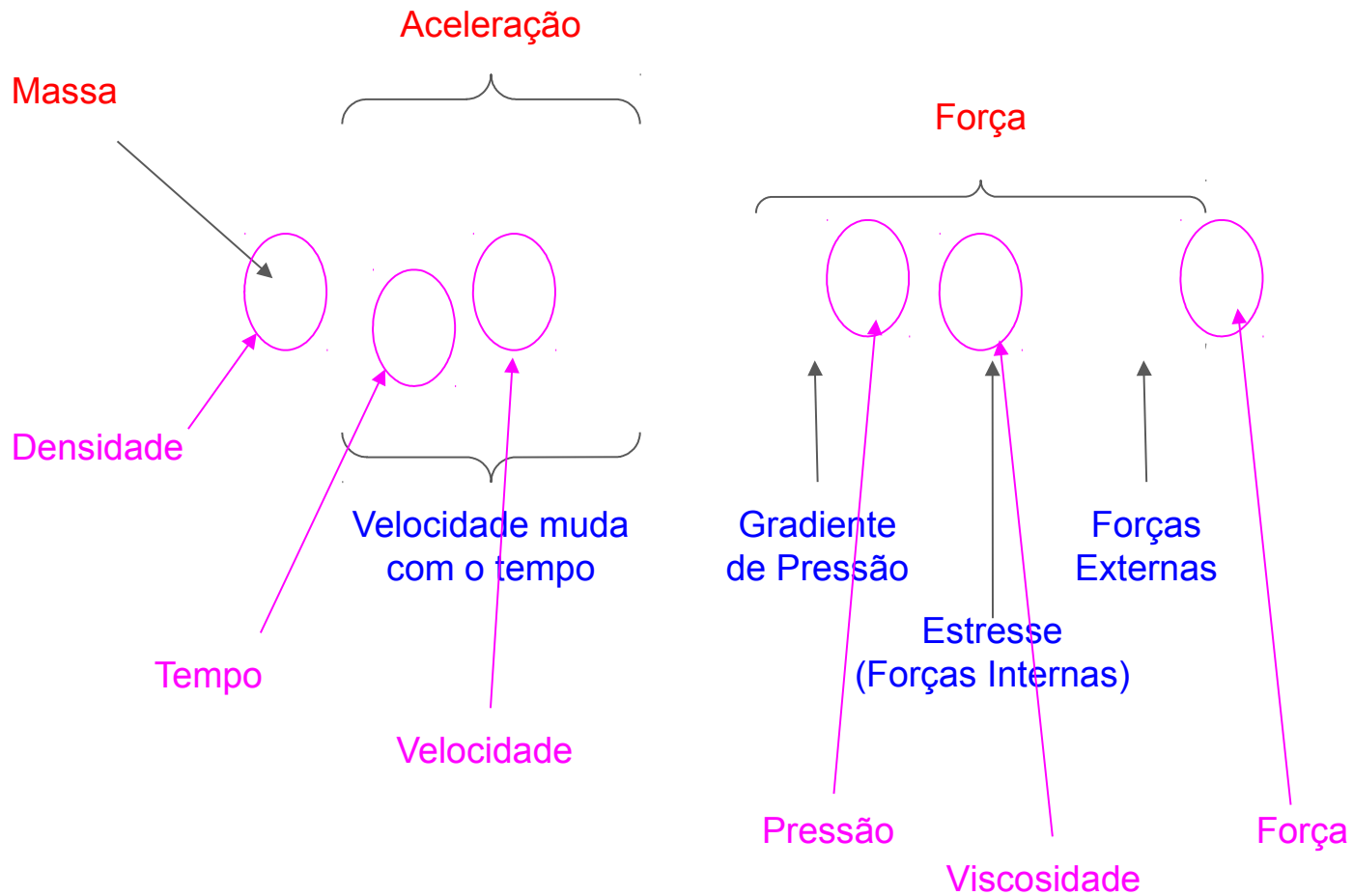
Navier-Stokes

The diagram illustrates the Navier-Stokes equation with the following components and annotations:

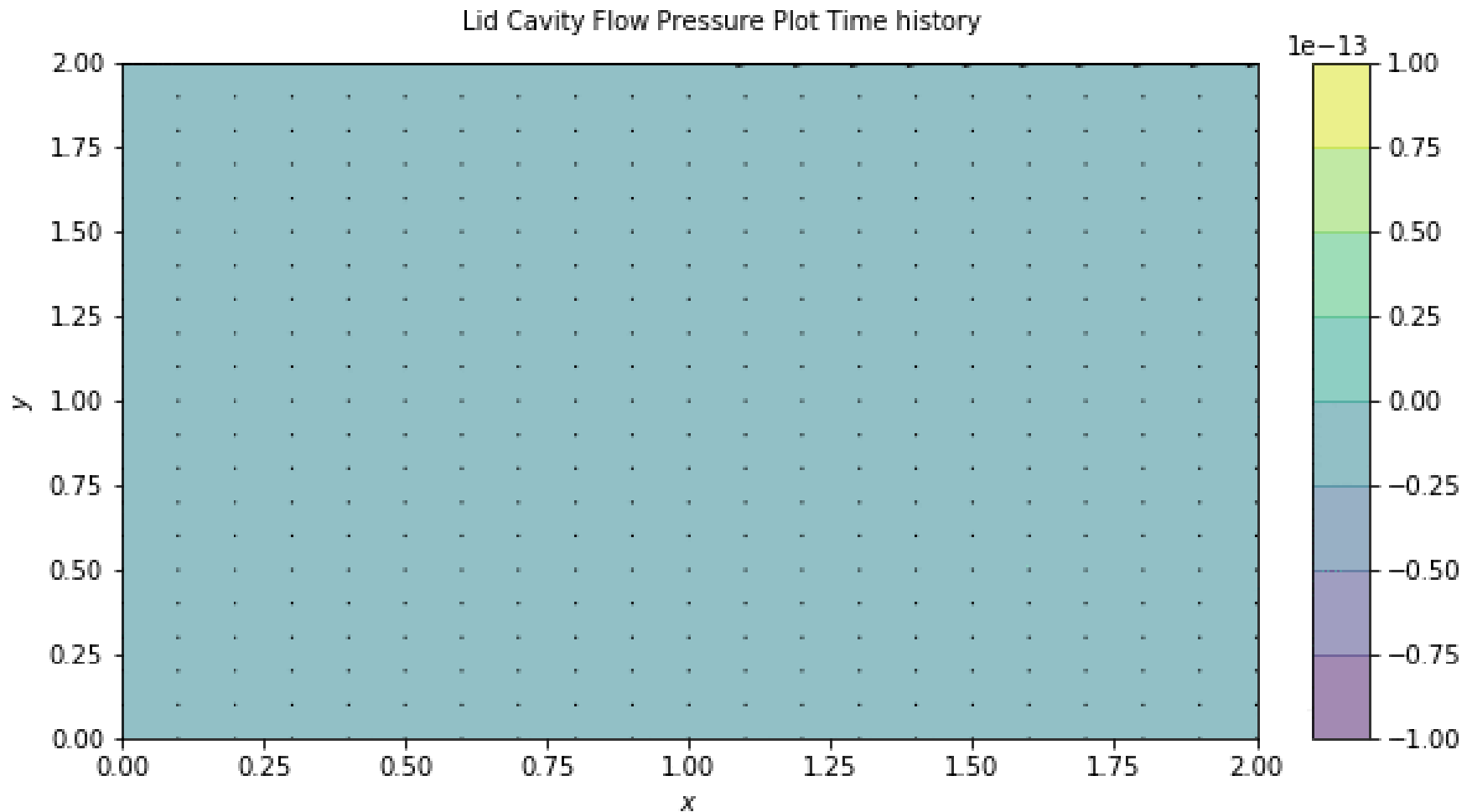
- Massa** (Mass): An arrow points to the density term ρ .
- Aceleração** (Acceleration): A bracket above the acceleration term $\left(\frac{\partial u}{\partial t} + u \cdot \nabla u \right)$.
- Força** (Force): A bracket above the force terms $\nabla P + \mu \nabla^2 \mu + \rho F$.
- Velocidade muda com o tempo** (Velocity changes with time): A bracket below the acceleration term $\left(\frac{\partial u}{\partial t} + u \cdot \nabla u \right)$.
- Gradiente de Pressão** (Pressure Gradient): An arrow points from the pressure term ∇P to this label.
- Estresse (Forças Internas)** (Stress (Internal Forces)): An arrow points from the viscosity term $\mu \nabla^2 \mu$ to this label.
- Forças Externas** (External Forces): An arrow points from the body force term ρF to this label.

$$\rho \left(\frac{\partial u}{\partial t} + u \cdot \nabla u \right) = \nabla P + \mu \nabla^2 \mu + \rho F$$

Navier-Stokes



Barba et al.. Module for learning CFD Python - 12 steps to Navier-Stokes (github)



Relação entre Navier-Stokes e p-model

- “Since the direct solution of the Navier-Stokes equations is not feasible for the full velocity field, a traditional and useful way of gaining insight into the geometry of turbulent flows has been the construction of models making analytic treatment possible. (...) the experimental results are much more consistent with the recent multifractal models: (...) the multifractal spectra derived from the p model are in accord with the experimental results for the generalized dimensions determined from measurements of the dissipation field.” (Vicsek, 1981)
- “A classical example from physics is the Navier-Stokes equation for fluid dynamics (10). In the turbulent regime, this nonlinear equation generates a multifractal output (...) (Amaral, 2012)
- “(...) show that there are some indications that multifractal sets are indeed necessary to describe the properties of structure functions in turbulent flows” (Benzi, 1984)
- “This leads to multifractal “p models,” originally proposed by Meneveau and Sreenivasan (1987) to simulate the highly intermittent spatial fluctuations of the kinetic energy dissipation in turbulence.” (Davis, 1997)

p-model

É um modelo usado para gerar séries temporais com estrutura fractal.

O modelo p em si, só pode produzir séries temporais estacionárias, isto é, séries temporais em que a variância é finita se você extrapolar seu espectro de potência para escalas grandes infinitas.

Opcionalmente, é possível filtrar o resultado do modelo p no espaço de Fourier para obter outro declive fractal, por exemplo para torná-lo contínuo e não-estacionário. Isso também é chamado de integração fracionária.

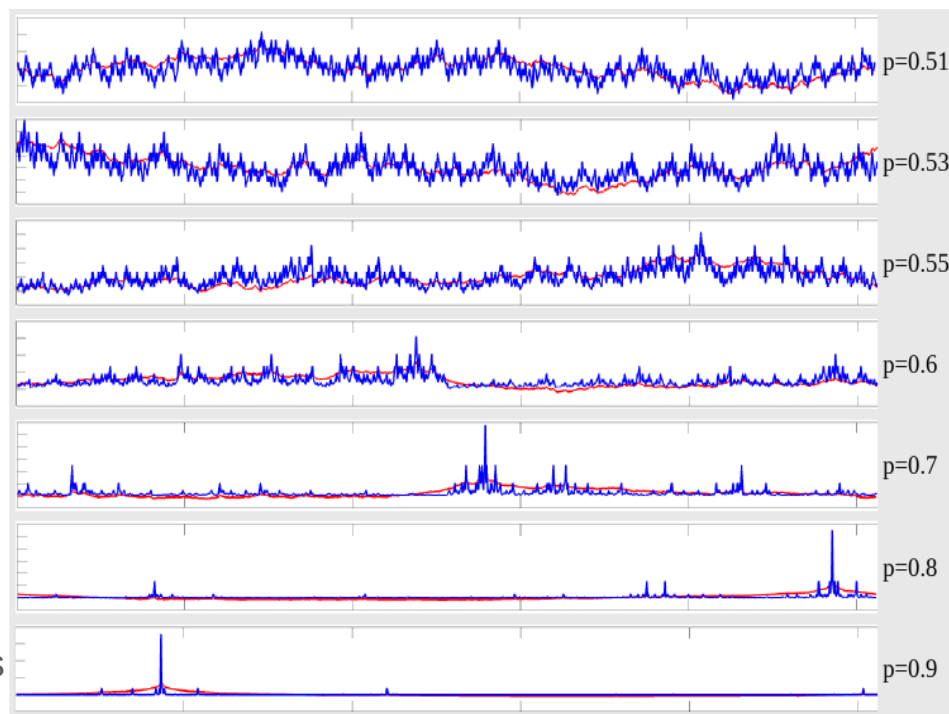
a figura ao lado foi gerada no Octave usando

```
[F, P] = pmodel(1024, p, -2);
```

onde

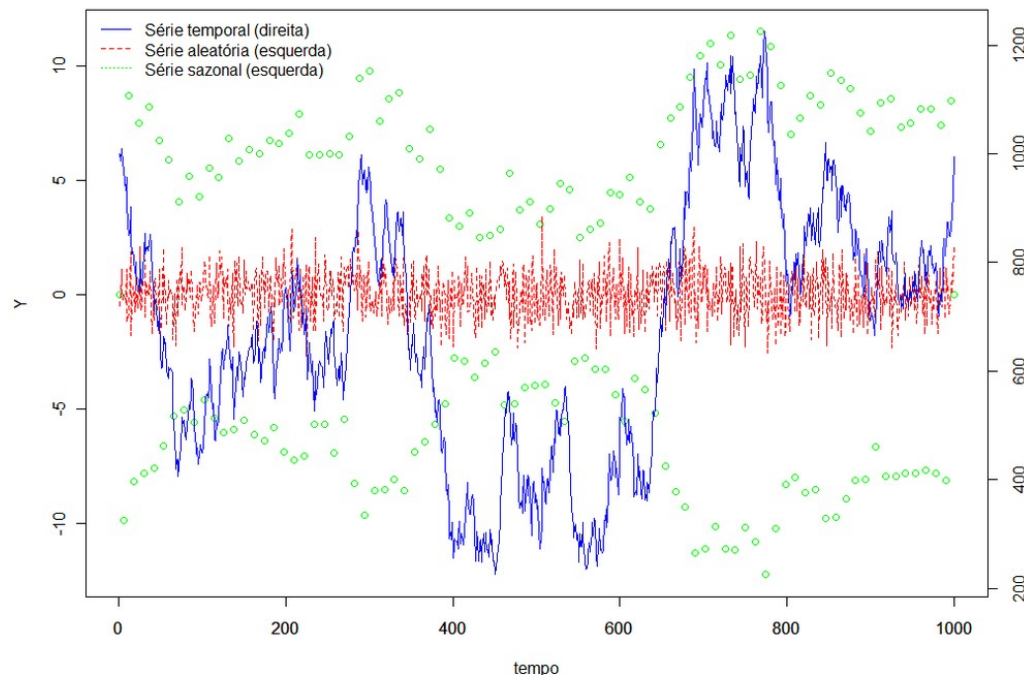
F(vermelho) = fractional Integrated Time Series

P(azul) = p Model Time Series



Série temporal

Coleção de observações feitas sequencialmente ao longo do tempo, onde observações vizinhas são dependentes.



Implementação em Python - p-model

baseado em:

<http://www2.meteo.uni-bonn.de/staff/venema/themes/surrogates/pmodel/pmodel.m>

Algoritmo

Conforme descrito em “Fast algorithm for a three-dimensional synthetic model of intermittent turbulence” (Malara et al., 2016) : No p-model uma distribuição espacial 1D de fluxo de energia em diferentes escalas, é construída através de um processo multiplicativo. Um redemoinho em uma escala L quebra em 2 redemoinhos na escala $L/2$, e o fluxo de energia E associado com o parente do redemoinho é desigualmente distribuído em dois redemoinhos filhos, com frações dadas por $2pE$ e $2(1-p)E$, respectivamente, com $0,5 \leq p \leq 1$.

```
def next_step_1d(y, p):  
    length = len(y)  
    y2 = np.zeros(length * 2)  
    sign = np.random.uniform(0, 1, length) - 0.5  
    sign = sign / abs(sign)  
    y2[::2] = y + sign * (1 - 2 * p) * y  
    y2[1::2] = y - sign * (1 - 2 * p) * y  
    return y2
```

O parâmetro do p-model é “p” e com valores próximos de 0 ou 1 existem muitos picos. Com valores perto de 0,5 é mais calmo. Em 0,5 o resultado é um vetor constante.

“y” é a série temporal gerada pelo p-model.

Inicialmente y tem um elemento 1. Vamos supor $p=0.52$.

```
y = [1]
for n in range(1, noOrders + 1, 1):
    y = next_step_1d(y, p)
```

Neste caso “next_step_1d” é chamado com os parâmetros [1] e 0.52.

```
>>> y = [1]
```

```
>>> length = len(y)
```

```
>>> len (y)
```

```
1
```

A variável `length` recebe a quantidade atual de elementos na série (no caso 1).

É criado um array temporário com dois elementos vazios:

```
>>> y2 = np.zeros(length * 2)
```

```
>>> y2
```

```
array([ 0.,  0.])
```

Cria-se um array de um elemento contendo um número aleatório:

```
>>> sign = np.random.uniform(0, 1, length) - 0.5
```

```
>>> sign
```

```
array([ 0.3706906])
```

É extraída a informação se é positivo ou negativo, retornando 1 ou -1:

```
>>> sign = sign / abs(sign)
```

```
>>> sign
```

```
array([ 1.])
```


São calculadas as “frações $2pE$ e $2(1-p)E$ ”

```
>>> y2[:,2] = y + sign * (1 - 2 * p) * y
```

```
>>> y2
```

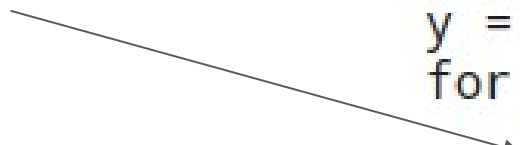
```
array([ 1.04, 0.  ])
```

```
>>> y2[1:,2] = y - sign * (1 - 2 * p) * y
```

```
>>> y2
```

```
array([ 1.04, 0.96])
```

```
return y2
```

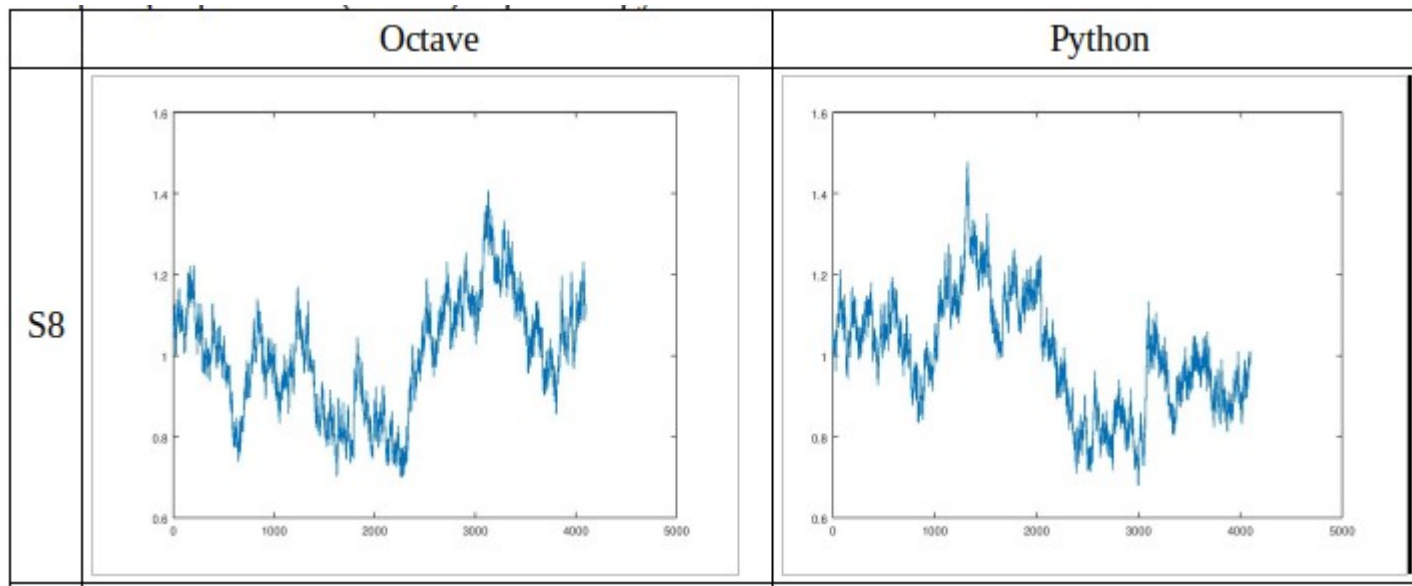


```
y = [1]  
for n in range(1, noOrders + 1, 1):  
    y = next_step_1d(y, p)
```

Resultados e considerações finais

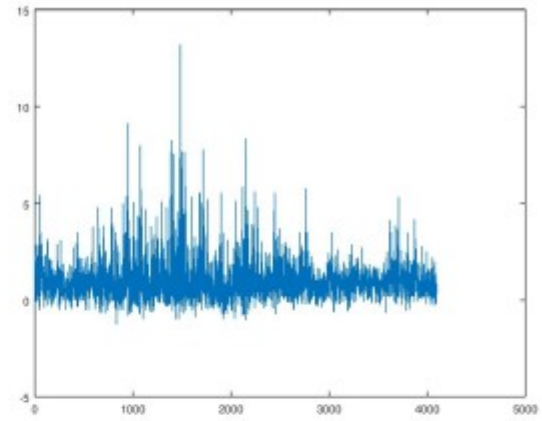
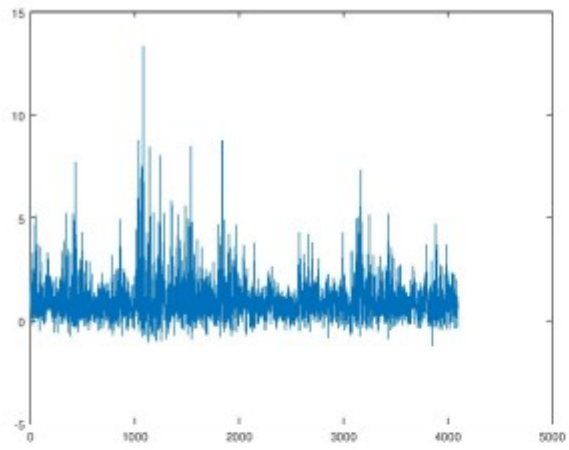
Comparação pmodel.m(Octave) X pmodel.py

```
>>> S8 = pmodel(noValues=2**12, p=0.52, slope=-1.66)
>>> S9 = pmodel(noValues=2**12, p=0.62, slope=-0.45)
>>> S10 = pmodel(noValues=2**12, p=0.72, slope=-0.75)
```



	Octave	Python
--	--------	--------

S9



S10

