SBAM

*a **S**parse **B**ayesian **A**uxiliary **M**odule for PySINDy*

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

group 2219:

Gaudio Raffaele
Giorgetti Luca
Vero Emerson
Zanoli Alessandro

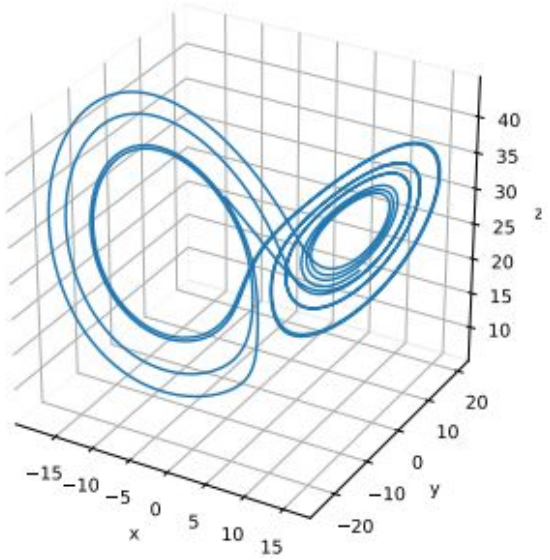
SINDy

Sparse Identification of Nonlinear Dynamical systems

SINDy is a framework devised for identifying the governing set of equations describing the dynamics of a particular system.

The goal of the method is to use data to obtain a model describing the system's dynamics while promoting sparsity among a list of "candidate features" for the time evolution of each variable.

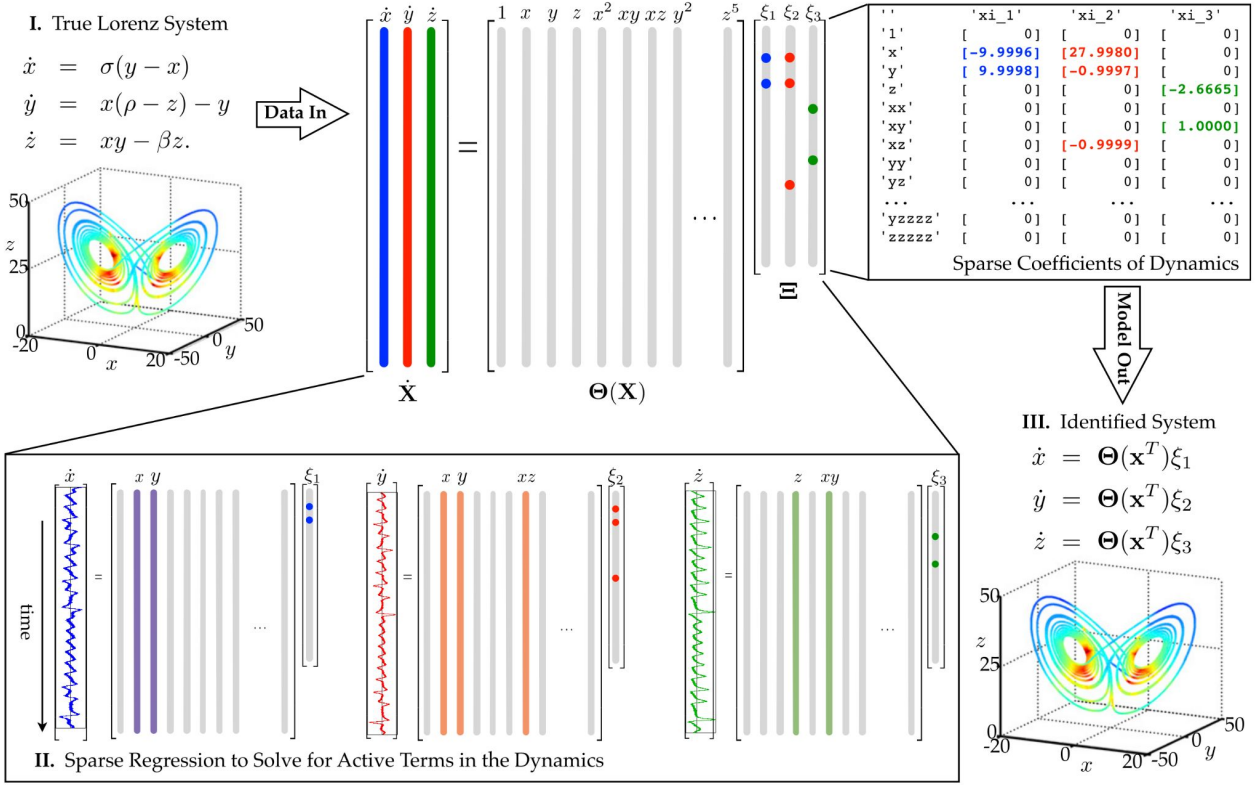
Sparsity is desirable as the goal of this method is to construct a model that's easily interpretable and generalizable by us.



SINDy

Sparse Identification of Nonlinear Dynamical systems

The key is to reframe the problem as a linear regression one, transforming the system's evolution into a higher dimensional manifold *(the library)*, hoping to find a hyperplane that is parallel to most of the features' axis.

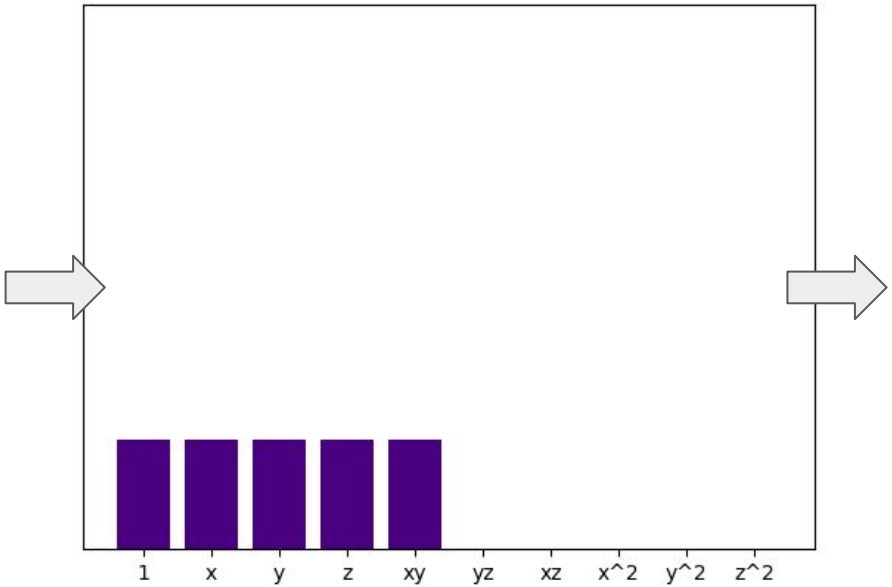


SINDy

Bayesian Approach

From a Bayesian standpoint, the act itself of building the library is a preliminary prior distribution that assigns a uniform probability to a finite set of features out of an infinite number of possible terms.

The scientist's role is therefore ineradicable, as the construction of the library itself is a subjective approximation.



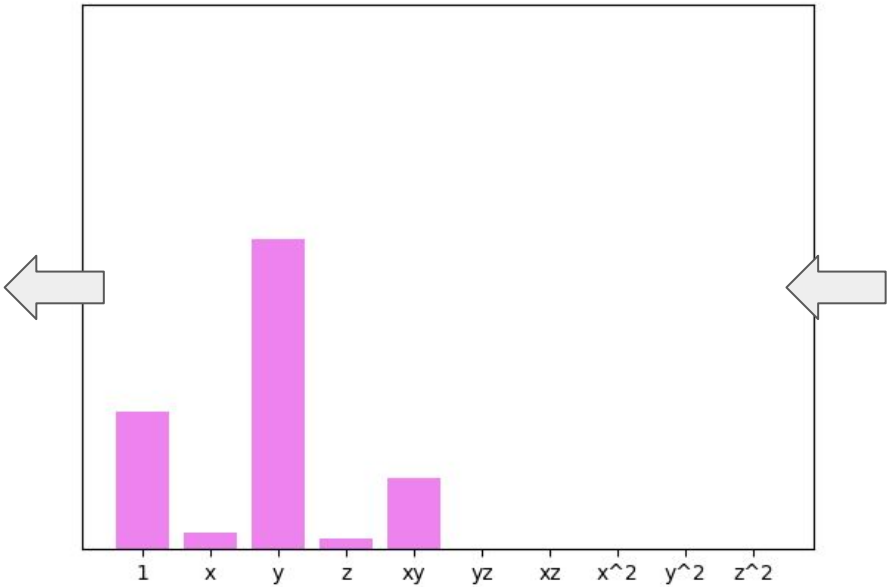
But how can we further model the sparsity assumption within a Bayesian framework?

How can we use data to *update* this prior belief of variable selection?

SINDy

Bayesian Approach

The goal of our project is to build a Bayesian model that will yield a posterior distribution for the feature inclusion, *given* inescapable prior that arises when building the library.



But how can we further model the sparsity assumption within a Bayesian framework?

How can we use data to *update* this prior belief of variable selection?

A Bayesian approach to linear regression

The core idea of the Bayesian approach to linear regression is the assumption that the likelihood of the observed data follows a distribution like:

$$P(\mathbf{y}|\mathbf{X}, \beta, \sigma^2) \propto \mathcal{N}\left(\mathbf{X}^T\beta, \sigma^2\right)$$

Specifying a prior belief for β (the vector of regression coefficients), given some observations (y,X), a posterior distribution for the value of β is obtained via Bayes theorem:

$$P(\beta|\mathbf{y}, \mathbf{X}, \sigma^2) \propto P(\mathbf{y}|\mathbf{X}, \beta, \sigma^2)P(\beta)$$

Spike and Slab prior

In a sense, variable selection is a Bernoulli process: either a feature is relevant or it isn't. How do we model this dichotomy? How to quantify our belief of sparsity and how could we ever infer it from data?

Enter the Spike and Slab prior: the key is to assign a *finite* probability to P(β=0). In order to do this, we need to "split" the prior distribution for the vector β:

$$P(\beta_j|z) = (1 - z_j)\delta_0 + z_j P_{slab}$$

According to a (vector of) "masking variable" z, which is drawn from a Bernoulli discrete distribution:

$$P(z_j) = Bern(\theta)$$

The prior distribution for z is determined by the parameter θ, which is the same for every entry of the masking vector. It represents the fraction of features that are important for the prediction, i.e. those whose coefficients are non-zero. It is itself an unknown variable, for which we can establish a prior distribution, but that will eventually be learned from data.

But what is P_{SLAB} ? That is our prior distribution for the regression coefficient, given that is deemed active by the z “coin toss”. This distribution is also common for each feature, since we have no reasonable prior assumption for different features.

A common choice would be a normal distribution centered on zero, commonly referred to as L2 regularization:

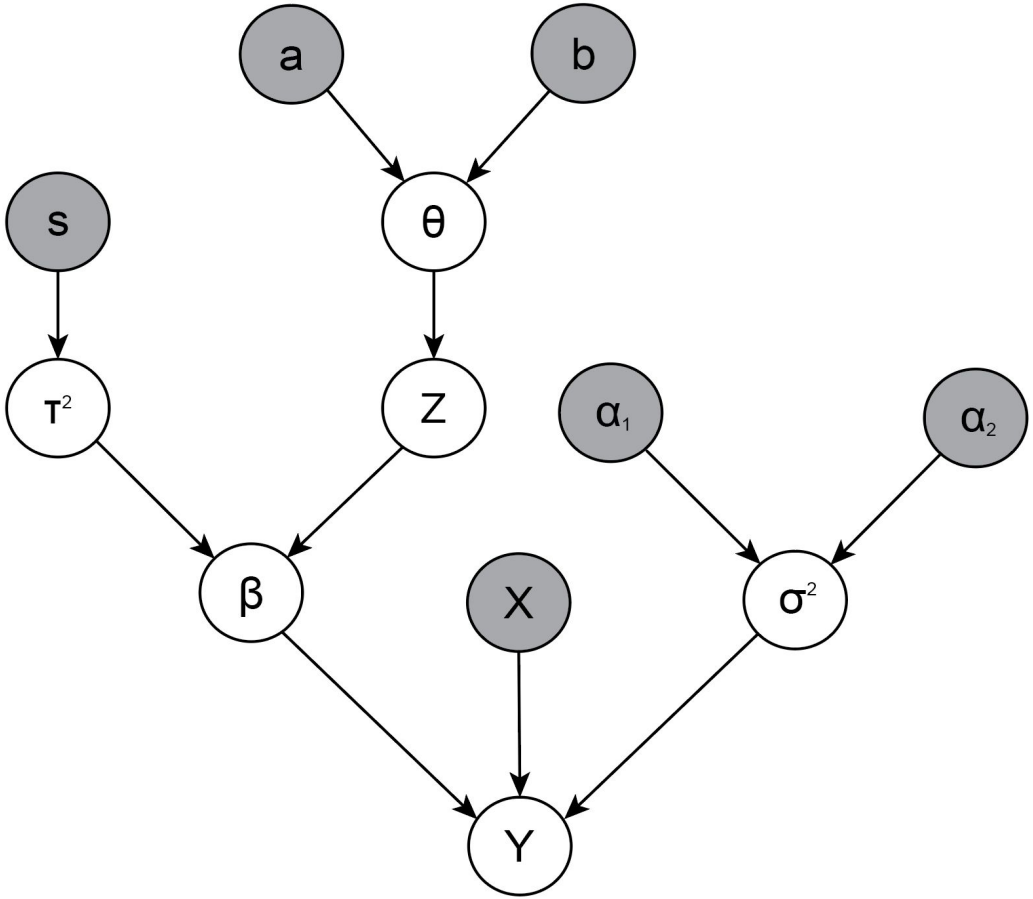
$$P(\beta_j|z_j = 1) \sim \mathcal{N}\left(0, \sigma^2\tau^2\right)$$

Whose variance depends on a parameter τ, that is being multiplied by the variance of the data in order to have the gaussian scale with the outcome and isolate the variance of the coefficients.

Again, τ is an unknown parameter, and given a prior distribution of it, it will be inferred from data.

The model

Given that σ is also an unknown parameter with its own prior distribution, we can finally summarize what turns out to be a whole Bayesian hierarchical model with a directed acyclic graph (DAG), to clarify all of the dependencies that we will need in order to factorize the joint distribution.



Since we will use Gibbs sampling to sample from the joint distribution, we need conditional posterior probabilities, whose dependencies can be deduced from the DAG:

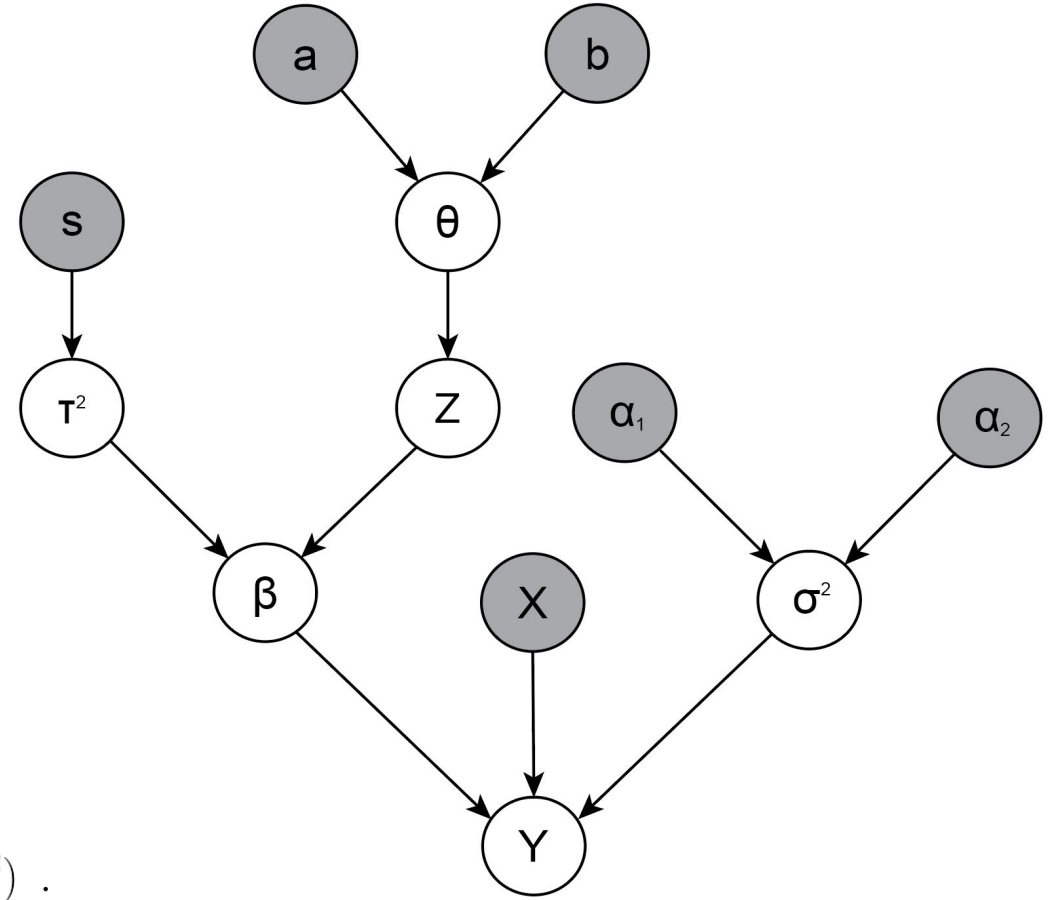
$$P(\theta \mid \mathbf{y}, \beta, z, \tau^2, \sigma^2) = P(\theta \mid z)$$

$$P(\tau^2 \mid \mathbf{y}, \beta, z, \theta, \sigma^2) = P(\tau^2 \mid \beta, z)$$

$$P(\sigma^2 \mid \mathbf{y}, \beta, z, \theta, \tau^2) = P(\sigma^2 \mid \mathbf{y}, \beta)$$

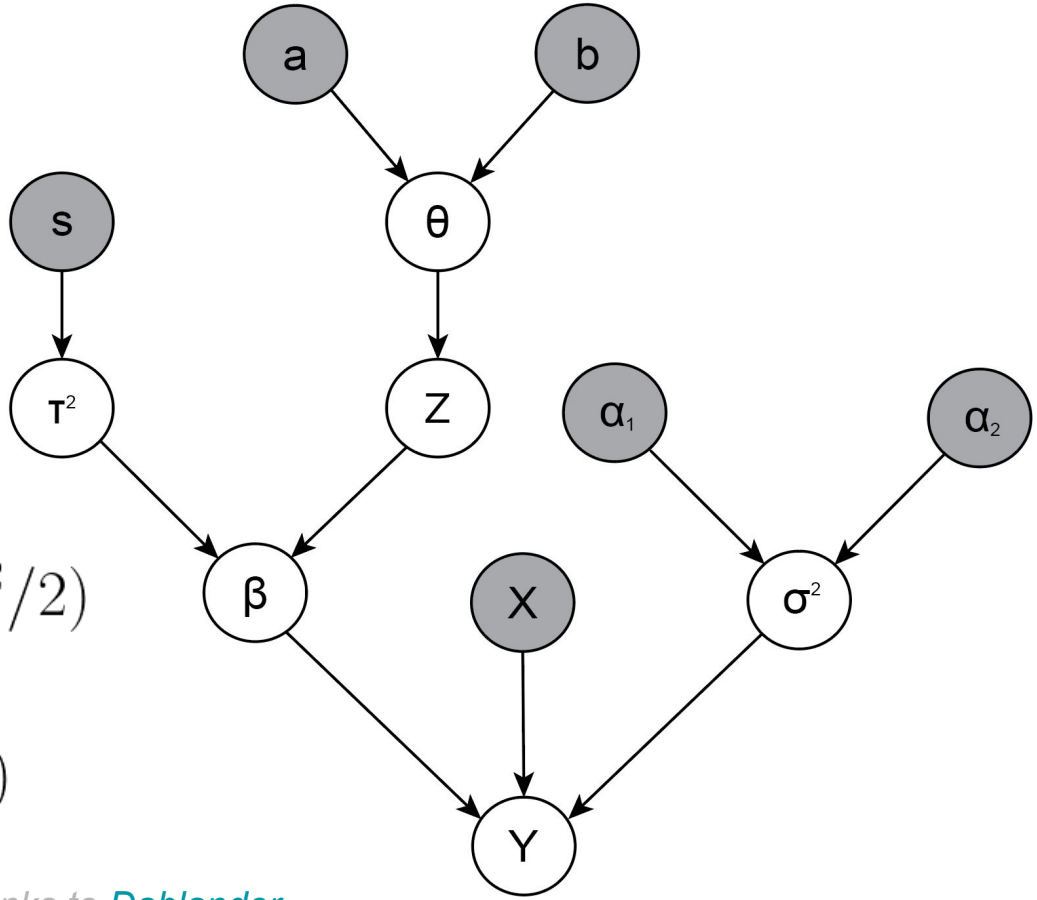
$$P(\pi \mid \mathbf{y}, \beta, \theta, \tau^2, \sigma^2) = P(z \mid \beta, \theta, \tau^2)$$

$$P(\beta \mid \mathbf{y}, z, \theta, \tau^2, \sigma^2) = P(\beta \mid \mathbf{y}, z, \tau^2, \sigma^2) \ .$$



Priors specification

Finally, we define the prior distributions, in order to derive the conditional posteriors analytically.



$$\tau^2 \sim \text{Inverse-Gamma}(1/2, s^2/2)$$
$$\theta \sim \text{Beta}(a, b)$$
$$\sigma^2 \sim \text{Inverse-Gamma}(\alpha_1, \alpha_2)$$

priors specification and analytical results thanks to Dablander

Conditional posterior distributions

$$\theta \mid z \sim \text{Beta}\left(a + \sum_{j=1}^{p} z_j, b + \sum_{j=1}^{p}(1 - z_j)\right)$$

$$\tau^2 \mid \beta, z \sim \text{Inverse-Gamma}\left(\frac{1}{2} + \frac{\sum_{j=1}^{p} z_j}{2}, \frac{s^2}{2} + \frac{\beta^T \beta}{2\sigma^2}\right)$$

$$\sigma^2 \mid \mathbf{y}, \beta \sim \text{Gamma}\left(\alpha_1 + \frac{n}{2}, \alpha_2 + \frac{(\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta)}{2}\right)$$

with p = number of features, n = number of observations and

$$\beta \mid \mathbf{y}, z, \tau^2, \sigma^2 \sim \mathcal{N}\left(\left(\mathbf{X}^T\mathbf{X}\frac{1}{\sigma^2} + \mathbf{I}\frac{1}{\sigma^2\tau^2}\right)^{-1}\mathbf{X}^T\mathbf{y}\frac{1}{\sigma^2}, \left(\mathbf{X}^T\mathbf{X}\frac{1}{\sigma^2} + \mathbf{I}\frac{1}{\sigma^2\tau^2}\right)^{-1}\right)$$

this last one being a multivariate normal also conditioned on z because we then apply

$$\beta = \beta \circ z$$

Conditional posterior distributions

each z_j is sampled from a Bernoulli process with $P(z_j=0)$ given by

$$1-\xi_j = \frac{(1-\theta)}{\left(\sigma^2\tau^2\right)^{-\frac{1}{2}}\exp\left(\frac{\left(\sum_{i=1}^{n} x_i u_i\right)^2}{2\sigma^2\left(\sum_{i=1}^{n} x_i^2+\frac{1}{\tau^2}\right)}\right)\left(\frac{\sigma^2}{\left(\sum_{i=1}^{n} x_i^2+\frac{1}{\tau^2}\right)}\right)^{\frac{1}{2}}\theta+(1-\theta)}$$

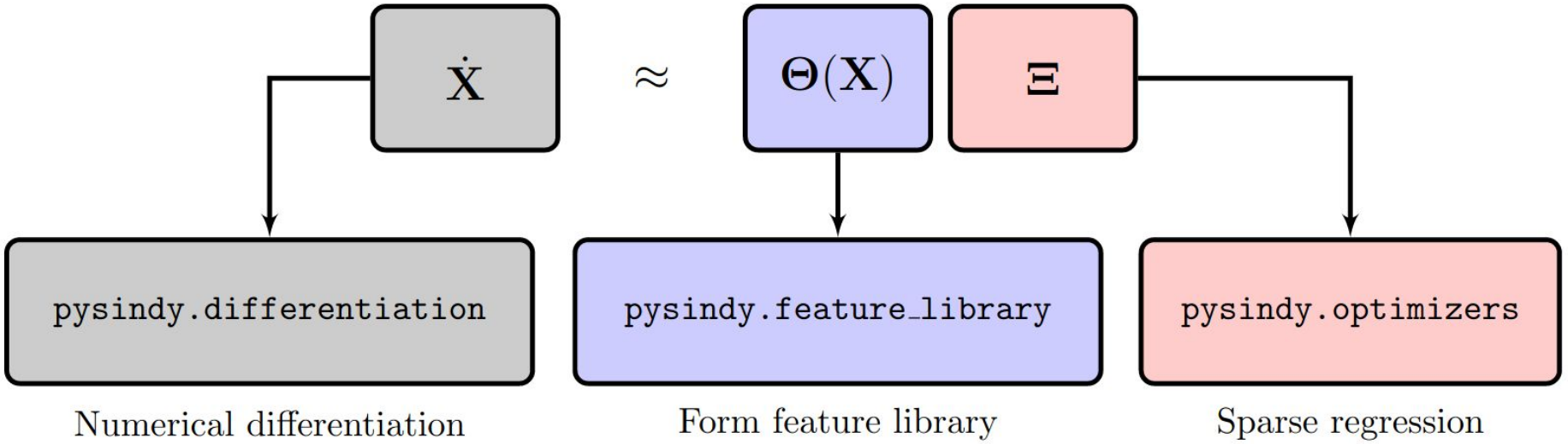
with x_i being the i-th observation of the j-th feature and u_i being the regression residuals without the j-th feature:

$$\mathbf{u} = \mathbf{y} - \mathbf{X}_{-j}\beta_{-j}$$

PySINDy

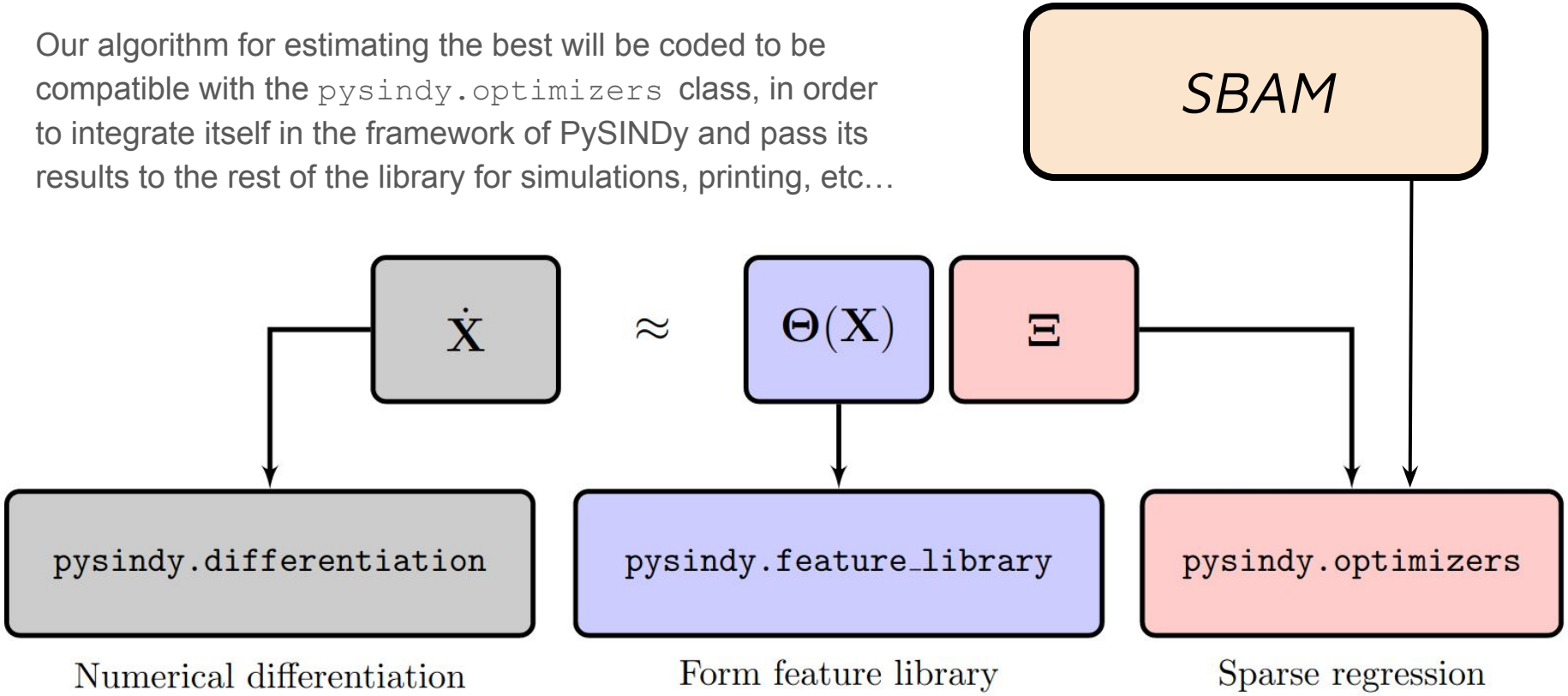
PySINDy is a sparse regression package with several implementations for the Sparse Identification of Nonlinear Dynamical systems (SINDy). The package is designed for compatibility with the **`sk-learn`** suite and takes care of the three main components that make up the SINDy workflow:

differentiation, library construction, regression



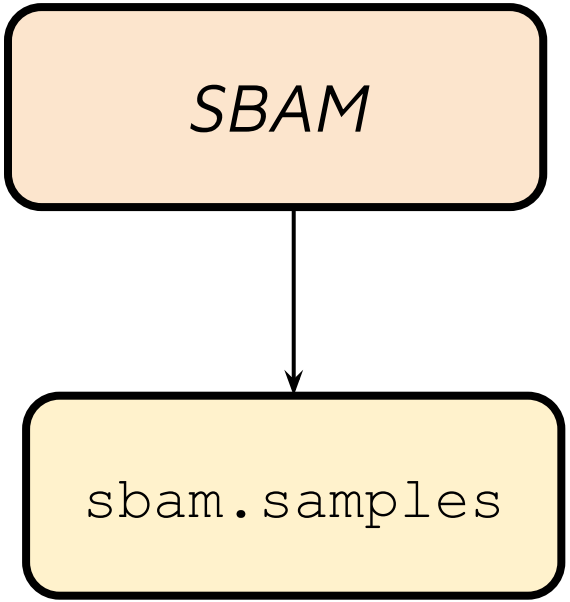
PySINDy + SBAM

Our algorithm for estimating the best will be coded to be compatible with the `pysindy.optimizers` class, in order to integrate itself in the framework of PySINDy and pass its results to the rest of the library for simulations, printing, etc…

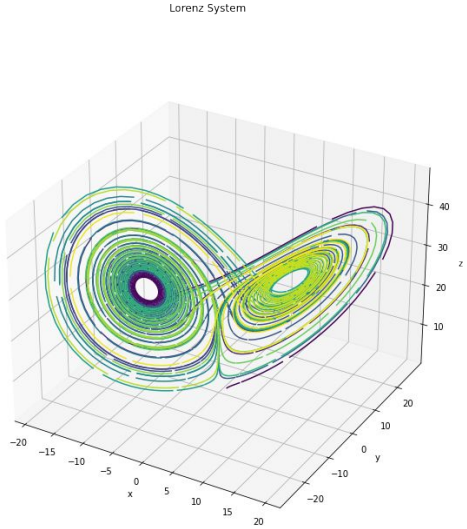


PySINDy + SBAM

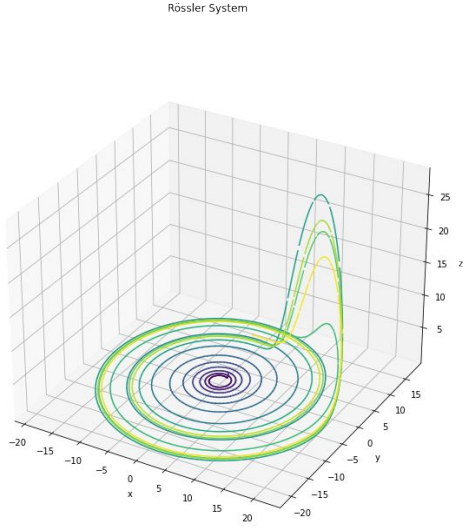
- PySINDy isn’t designed to expect a probability distribution over the coefficients;
- Our method involves Gibbs sampling of the posterior distribution in order to compute interesting statistical indicators
- This is why our module will retain the samples in the form of a Pandas dataframe as an internal attribute, accessible for statistical analysis



Numerical experiments



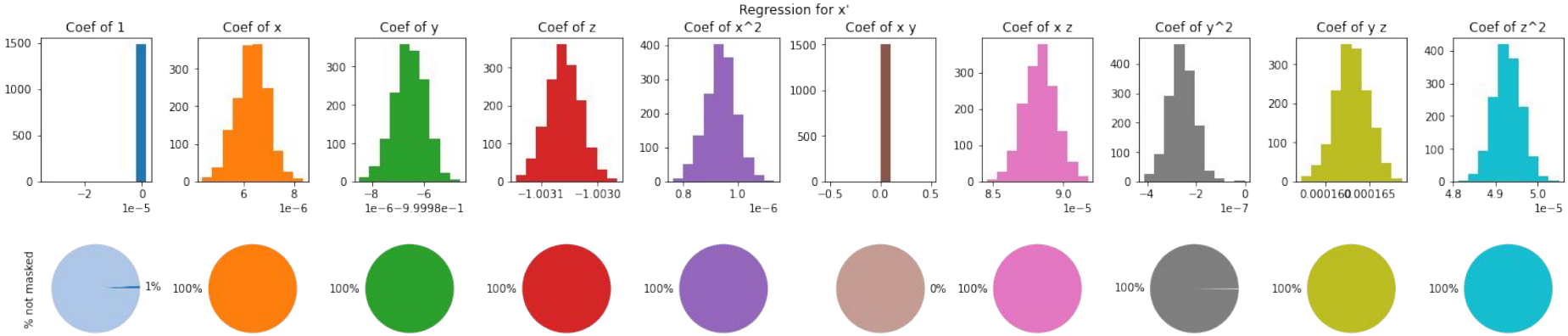
$$\begin{cases} \dot{x} = \sigma(y - x) \\ \dot{y} = x(\rho - z) - y \\ \dot{z} = xy - \beta z \end{cases} \qquad \begin{cases} \sigma = 10 \\ \rho = 28 \\ \beta = \frac{8}{3} \end{cases}$$



$$\begin{cases} \dot{x} = -y - z \\ \dot{y} = x + ay \\ \dot{z} = b + z(x - c) \end{cases} \qquad \begin{cases} a = 0.1 \\ b = 0.1 \\ c = 14 \end{cases}$$

Rössler system *ẋ(t) regression*

uniform prior on $\theta \in [0,1]$:



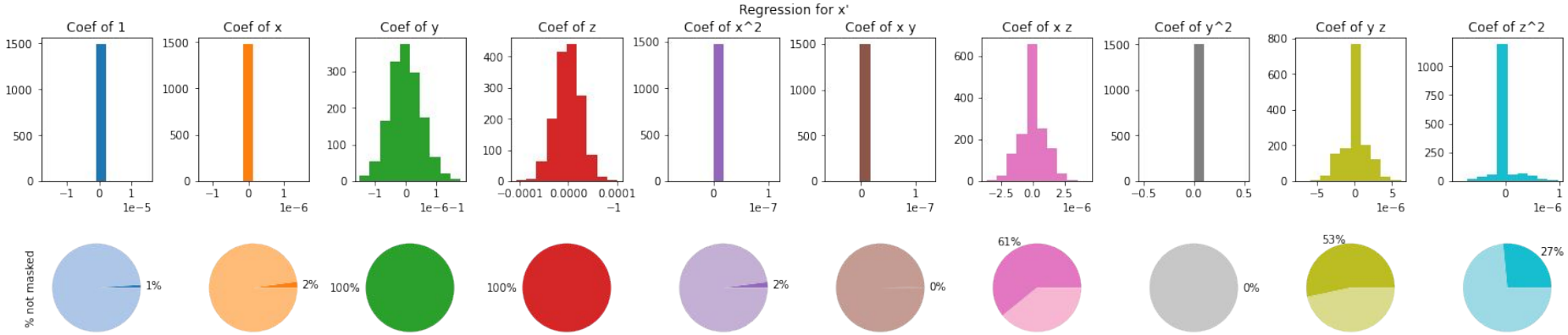
The sampler immediately lands on the non-zero distribution for most of the features' coefficients, even when initialized in different ways.

The library construction may introduce some bias in the hyperplane that is enough for the likelihood to completely disrupt the "coin toss" in favour of the non-zero distribution.

$$\begin{cases} \dot{x} = -y - z \\ \dot{y} = x + ay \\ \dot{z} = b + z(x - c) \end{cases} \qquad \begin{cases} a = 0.1 \\ b = 0.1 \\ c = 14 \end{cases}$$

Rössler system *ẋ(t) regression*

uniform prior on $\theta \in [0,1]$: `ps.FiniteDifference(order=12)`



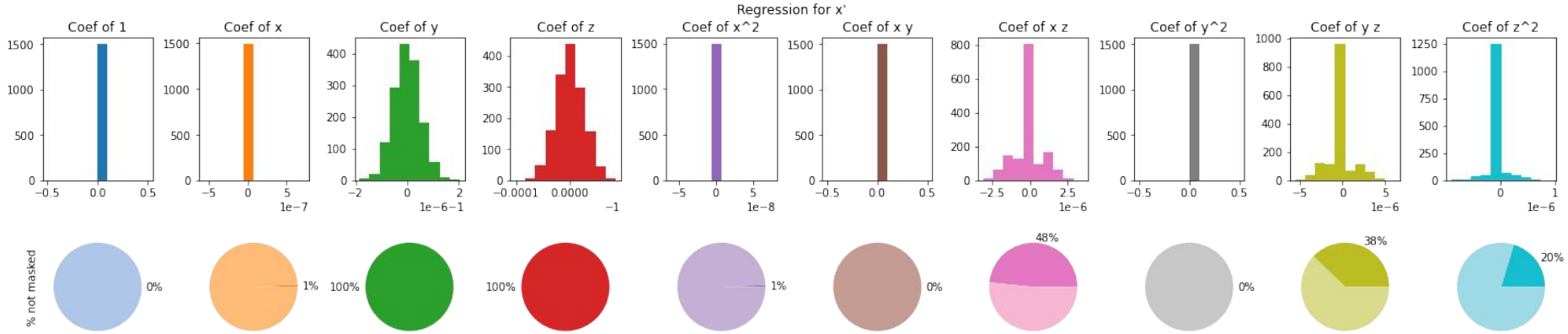
By changing the order of the finite differences algorithm we calculate the derivatives including more data points and this apparently reduces the bias.

But is a uniform prior on θ the proper choice to impose our sparsity beliefs? That would mean we expect about half of our features to be relevant, but that is definitely not the case…

$$\begin{cases} \dot{x} = -y - z \\ \dot{y} = x + ay \\ \dot{z} = b + z(x - c) \end{cases} \qquad \begin{cases} a = 0.1 \\ b = 0.1 \\ c = 14 \end{cases}$$

Rössler system *ẋ(t) regression* $\theta \sim$ Beta(a,b) a≪b

```
ps.FiniteDifference(order=12)
```



By tuning the hyperparameters to have a distribution for θ much more skewed towards zero, we find that the mode of the posterior distribution for z matches exactly the true system.

```
model.print()

(x)' = -1.000 y + -1.000 z
(y)' = 1.000 x + 0.100 y
(z)' = 0.100 1 + -14.000 z + 1.000 x z
```

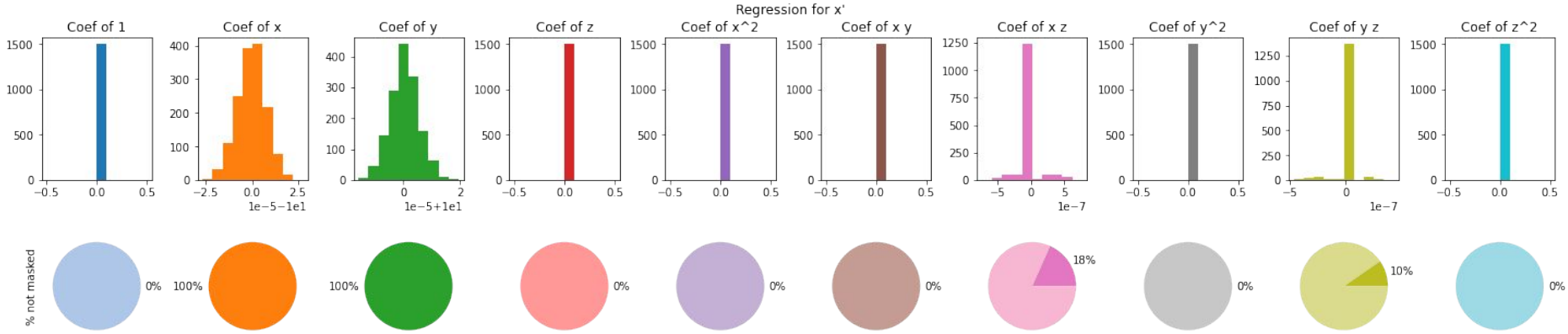
$$\begin{cases} \dot{x} = -y - z \\ \dot{y} = x + ay \\ \dot{z} = b + z(x - c) \end{cases} \qquad \begin{cases} a = 0.1 \\ b = 0.1 \\ c = 14 \end{cases}$$

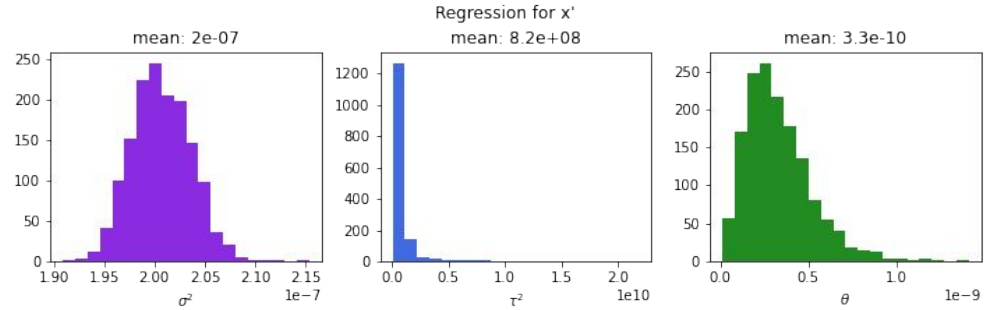
Lorentz system *ẋ(t) regression*

θ ~ Beta(a,b) a≪b

```
ps.FiniteDifference(order=12)

N_data=10000
```





$$\begin{cases} \dot{x} = \sigma(y - x) \\ \dot{y} = x(\rho - z) - y \\ \dot{z} = xy - \beta z \end{cases}$$

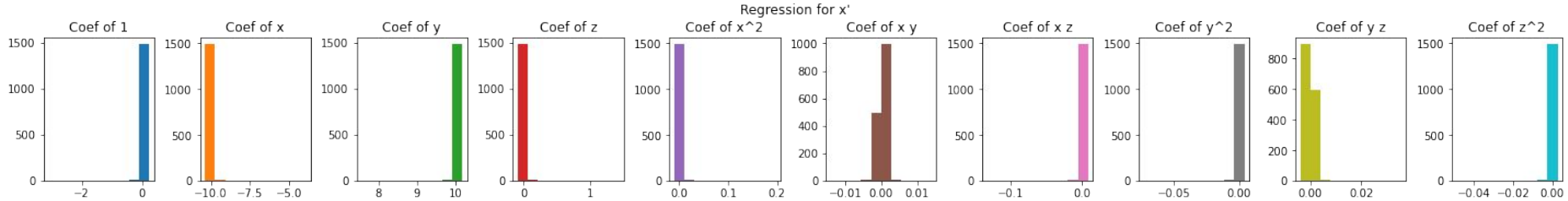
$$\begin{cases} \sigma = 10 \\ \rho = 28 \\ \beta = \frac{8}{3} \end{cases}$$

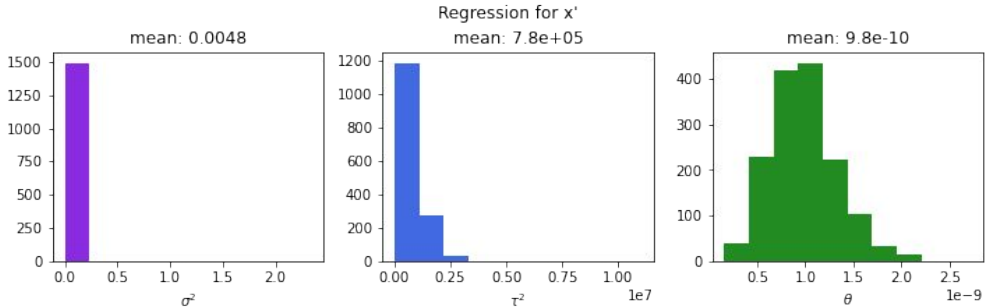
Lorentz system *ẋ(t) regression*

θ ~ Beta(a,b) a≪b

```
ps.FiniteDifference(order=12)

N_data=100
```





$$\begin{cases} \dot{x} = \sigma(y - x) \\ \dot{y} = x(\rho - z) - y \\ \dot{z} = xy - \beta z \end{cases}$$

$$\begin{cases} \sigma = 10 \\ \rho = 28 \\ \beta = \frac{8}{3} \end{cases}$$

Lorentz system

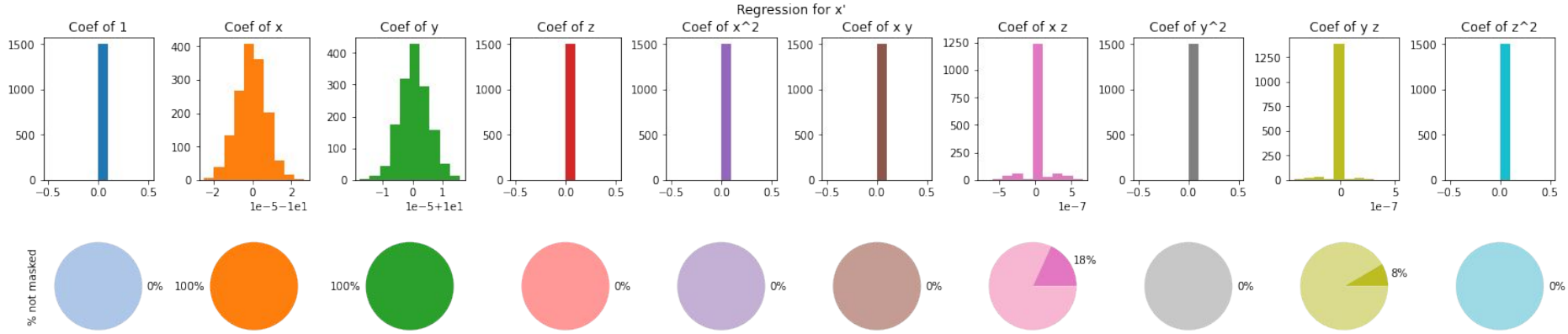
$\dot{x}(t)$ regression

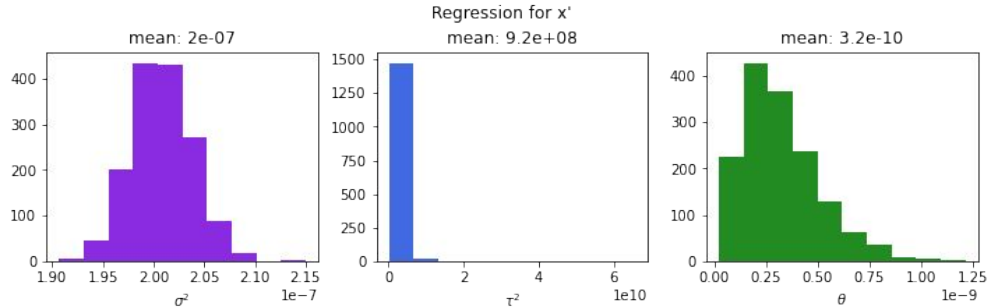
θ ~ Beta(a,b) a≪b

```
ps.FiniteDifference(order=12)

N_data=10000

noise = np.random.normal(0,1000)
```





$$\begin{cases} \dot{x} = \sigma(y - x) \\ \dot{y} = x(\rho - z) - y \\ \dot{z} = xy - \beta z \end{cases}$$

$$\begin{cases} \sigma = 10 \\ \rho = 28 \\ \beta = \frac{8}{3} \end{cases}$$

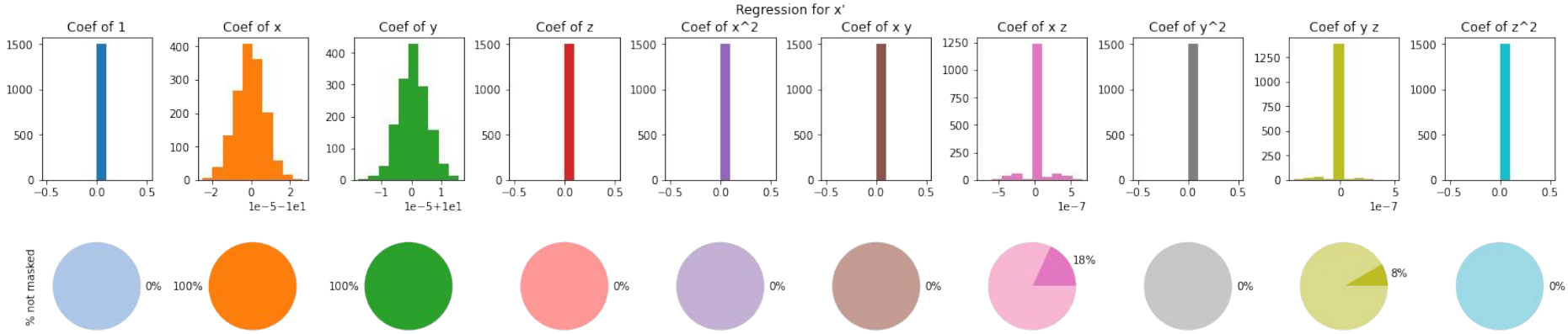
Lorentz system *ẋ(t) regression*

θ ~ Beta(a,b) a≪b

```
ps.FiniteDifference(order=12)

N_data=10000

noise = np.random.normal(0,1000)
```




```
model.print()


(x)' = -10.000 x + 10.000 y
(y)' = 28.000 x + -1.000 y + -1.000 x z
(z)' = -2.667 z + 1.000 x y
```

$$\begin{cases} \dot{x} = \sigma(y - x) \\ \dot{y} = x(\rho - z) - y \\ \dot{z} = xy - \beta z \end{cases} \qquad \begin{cases} \sigma = 10 \\ \rho = 28 \\ \beta = \frac{8}{3} \end{cases}$$

Conclusions

- The role of the thresholding value for the standard SINDy algorithm is somehow replaced by the prior specification for θ; the choice of a thresholding algorithm balances the presence of a bias in the regression model, and our choice or theta prior does the same in influencing the activation of a certain variable.