# Data-Driven System Identification of the Barabási–Albert Model

David Contreras Franco
Departamento de Electrónica y Ciencias de la Computación, Pontificia Universidad Javeriana Cali
davidcontreras@javerianacali.edu.co

***Abstract*—This work compares the Sparse Identification of Nonlinear Dynamics (SINDy) and Nonlinear AutoRegressive Moving Average with eXogenous input (NARMAX) methods in the task of dynamics identification. The comparison considered on Ordinary Differential Equation (ODE) systems, such as chaotic systems, nonlinear transfer functions with random input, and node degree evolution of the Barabási–Albert (BA) model, with the extension of their capability to identify different equations between two nodes with different *fitness* values in a Bianconi-Barabábsi model. The comparison metrics selected focus on model representation, sparsity, estimation error, and time. The comparison shows an initial indication that SINDy finds sparser models, while NARMAX finds models with minimised estimation error.**

***Index Terms*—Barabási–Albert Model, System Identification, Data-driven methods, Sparsity**

## I. Introduction

There are numerous examples of complex interconnected systems, whose collective behaviour cannot be explained by understanding the isolated behaviour of individual components. The inability to fall back on reductionism, makes it unfeasible to use the classical approach to model a system. Additionally, as systems increase complexity, this model tuning requires more and more time and resources. Consequently, novel approaches for system identification that automate this process are required.

The lofty aim of system identification methods for nonlinear systems is to facilitate, from available data, the estimation of nonlinear dynamics of a system. To better understand and interpret complex interconnected systems, this work proposes to identify the equations that govern the evolution of systems of interconnected entities based on a well-studied mathematical framework. In particular, we want to recreate identifying a system using data generated by the Barabási–Albert (BA) model, a stochastic network model with simple rules. The aim is to compare the performance of two different system identification methods.

## II. Theoretical Background

This work over-viewed methods, systems, and metrics. Among the methods for consideration are all of those which can correctly identify the proper underlying behaviour of the system; this means that all the linear methods, piece-wise linear approximations, and similar, are not considered.

### A. NARMAX

Nonlinear AutoRegressive Moving Average with eXogenous input (NARMAX) consists of a general case definition of the identification of a nonlinear system, defined by a function that depends only on previous input and output values and stochastic terms, where the goal is to find the simplest model possible that can be related to the underlying system, instead of the generalisation and estimation [1]. The main idea is to build the underlying model by iteratively picking the most important nonlinear term from a library of candidate functions until the model is adequate and related to the underlying system. This is achieved via the Orthogonal Least Squares (OLS) estimator in par with the Error Reduction Ratio (ERR) test. In the sense of sparse regression, it is paramount to note that Orthogonal Matching Pursuit (OMP) is a slight variation of OLS.

$$\begin{aligned} y_k = F[&y_{k-1}, \dots, y_{k-n_y}, \\ &x_{k-d}, \dots, x_{k-d-n_x}, \\ &e_{k-1}, \dots, e_{k-n_e}] + e_k \end{aligned} \tag{1}$$

Where $F[\cdot]$ is some nonlinear function of the input and output with a time delay $d$, $x_k \in \mathbb{R}^{n_x}$ is the input, $y_k \in \mathbb{R}^{n_y}$ is the output, and $e_k \in \mathbb{R}^{n_e}$ is the stochastic term at discrete time $k \in \mathbb{N}^n$, with $n_y, n_x, n_e \in \mathbb{N}$ as the maximum lags for the input, output, and noise, respectively. the function $F[\cdot]$ will be described from a Polynomial basis of degree five. The selection for the basis does not represent the limits of the NARMAX method; extreme nonlinear behaviours may be represented via a Bilinear, Rational, Radial basis functions, Wavelet Decomposition, among many possible basis functions, which NARMAX defines as the Extended Model Set, composed by a mix of different basis functions.

*1) SysIdentPy:* SysIdentPy is the Python package that implements the NARMAX method. Since SysIdentPy implements a Polynomial NARMAX model and the Forward Regression Orthogonal Least Squares (FROLS) algorithm, usage of this package is simplified. Least Squares (LS) and Recursive Least Squares (RLS) were used for the estimator. Regarding the information criterion, SysIdentPy has implemented: Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC), Law of Iterated Logarithm Criterion (LILC), and Final Prediction Error (FPE); from which BIC is selected due to consistently resulting in sparser models during testing.

### B. SINDy

The Sparse Identification of Nonlinear Dynamics (SINDy) method considers the problem of model discovery from the compressed sensing perspective, whereas out of many candidate functions, only a few of them are needed to describe the dynamics; *ergo*, the terms of the governing dynamics equations are sparse in the nonlinear functions space. Considering the vector $\mathbf{x} \in \mathbb{R}^n$ represents the state of an n-dimensional system at time $t$, the dynamics are described by the function $\mathbf{f}$.

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t)) \qquad \dot{\mathbf{X}} = \begin{bmatrix} \dot{\mathbf{x}}^T(t_1) \\ \vdots \\ \dot{\mathbf{x}}^T(t_m) \end{bmatrix} \tag{2}$$

To determine $\mathbf{f}$, from the states evolution through time matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$, the matrix $\dot{\mathbf{X}} \in \mathbb{R}^{n \times m}$ can either be measured or numerically defined from $\mathbf{X}$ when considering a continuous system with differential equations as dynamics. The library of candidate functions $\mathcal{D}$ contains all possible basis functions that describe the dynamics. With each basis function $\theta_i \in \mathcal{D}$ evaluated on the matrix $\mathbf{X}$ as $\theta_i(\mathbf{X}) \in \mathbb{R}^{n \times m}$, the complete library $\Theta(\mathbf{X})$ describes all possible dynamics of $\mathbf{X}$ given $\mathcal{D}$.

The sparse vectors of coefficients $\Xi = [\xi_1 \dots \xi_i \dots]$ determine which basis functions are present in the dynamics. Here, the sparse regression problem gives a solution to $\Xi$, and may be solved through known algorithms like Least Absolute Shrinkage and Selection Operator (LASSO) or OMP. This method has shown to be quite robust and flexible in a wide range of applications [2], but mainly aims at interpretability and directly solves the main problem of this work.

$$\dot{\mathbf{X}} = \Theta(\mathbf{X})\Xi \qquad \Theta(\mathbf{X}) = [\theta_1(\mathbf{X}) \dots \theta_i(\mathbf{X}) \dots] \tag{3}$$

*1) PySINDy:* PySINDy is the Python package made by the original authors of the method [3]. It is straightforward to use and flexible regarding the selection of the differentiation method, the library of basis functions, and the optimiser, the algorithm that solves the sparse regression problem. Regarding the differentiation method, only finite differences is used for comparison, and the library of basis functions was restricted to a Polynomial basis of degree five, with the additional custom basis function for the BA model as described in section III-C. Finally, the Sequential Thresholded Least-Squares (STLSQ) algorithm was used as an optimiser, given its computational efficiency and fast convergence. STLSQ is a type of Iterative Hard-Thresholding (IHT) algorithm, where a solution for $\Xi$ is iteratively found with LS, and then thresholded by a value $\lambda$ with the element-wise hard-thresholding operator $H_\lambda$.

$$H_\lambda(\Xi) = \begin{cases} 0 & \Xi_{ij} \leq \lambda \\ \Xi_{ij} & \Xi_{ij} > \lambda \end{cases} \tag{4}$$

$$\Xi^{(n+1)} = H_\lambda \left( L_s \left( \Theta(\mathbf{X}), \dot{\mathbf{X}}, \Xi^{(n)} \right) \right) \tag{5}$$

### C. Differential Equation Systems

As a part of the cross-comparison systems, a set of Ordinary Differential Equations (ODEs) systems was chosen. The following examples were extracted from the extensive documentation for the package PySINDy [3].

1) Linear 2D ODE from equations 6 in a time range $[0, 25]$ with $dt = 0.01$ and a test size of $75\%$.
2) Cubic 2D ODE from equations 7 in a time range $[0, 25]$ with $dt = 0.001$ and a test size of $75\%$.
3) Linear 3D ODE from equations 8 in a time range $[0, 50]$ with $dt = 0.01$ and a test size of $60\%$.
4) Lorenz System ODE from equations 9 with $\sigma = 10$, $\rho = 28$, and $\beta = \frac{8}{3}$, in a time range $[0, 25]$ with $dt = 5e-4$ and a test size of $75\%$.
5) Rössler Attractor from equations 10 with $\alpha = 0.2$, $\beta = 0.2$, and $\gamma = 5.7$, in a time range $[0, 50]$ with $dt = 1e-2$ and a test size of $50\%$.

$$\begin{aligned} \dot{x} &= -0.1x + 2y \\ \dot{y} &= -2x - 0.1y \end{aligned} \tag{6}$$

$$\begin{aligned} \dot{x} &= -0.1x^3 + 2y^3 \\ \dot{y} &= -2x^3 - 0.1y^3 \end{aligned} \tag{7}$$

$$\begin{aligned} \dot{x} &= -0.1x + 2y \\ \dot{y} &= -2x - 0.1y \\ \dot{z} &= -0.3z \end{aligned} \tag{8}$$

$$\begin{aligned} \dot{x} &= \sigma(y - x) \\ \dot{y} &= x(\rho - z) - y \\ \dot{z} &= xy - \beta z \end{aligned} \tag{9}$$

$$\begin{aligned} \dot{x} &= -y - z \\ \dot{y} &= x + \alpha y \\ \dot{z} &= \beta + z(x - \gamma) \end{aligned} \tag{10}$$

### D. Nonlinear Functions

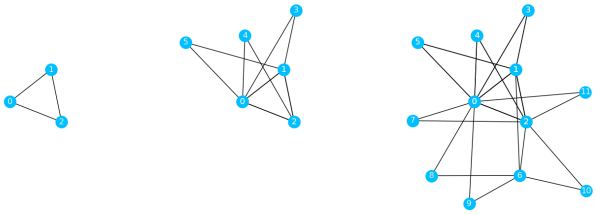
Additionally for the cross-comparison, the methods were tested on the transfer functions with Single-Input Single-Output (SISO) as in equation 11 and Multiple-Input Single-Output (MISO) as in equation 12, both with a uniformly distributed random variable $x$ as input and some added white noise $e$ to the output [4]. Also, in both cases, $1000$ samples were generated, from which $10\%$ was used for testing the identified model. The equations have been rewritten to show time delays more evidently.

$$y[k+1] = 0.2y[k] + 0.1y[k]x[k] + 0.9x[k-1] + e[k] \tag{11}$$

$$\begin{aligned} y[k+1] =& 0.4y^2[k] + 0.1y[k]x_1[k] + 0.6x_2[k] \\ & - 0.3x_1[k]x_2[k-1] + e[k] \end{aligned} \tag{12}$$

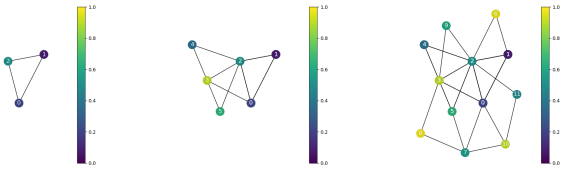
### E. Barabási–Albert Model

The Barabási–Albert model can generate scale-free networks that show emergence behaviour and, therefore, represent many types of complex systems [5]. This scale-free property is achieved through two primary rules in the BA model: Growth and Preferential Attachment. Growth means that a new node is



(a) $n = 3$ (b) $n = 6$ (c) $n = 12$

Figure 1: Example evolution of the Barabási–Albert model with $m = 2$.



(a) $n = 3$ (b) $n = 6$ (c) $n = 12$

Figure 2: Example evolution of the Bianconi-Barabási model with $m = 2$ and random uniform distribution for the fitness. The *fitness* level for each node is expressed in the colour-bar.

added to the network with $m$ new connections or links at each step, and preferential attachment defines which nodes the new node will most likely connect. Understanding that the number of links in a node is the degree of the node, in equation 13, $k_i$ refers to the degree of the $i$-th node. The network is generated following the next steps:

1) Start with $m_0$ nodes, with arbitrary links between them.
2) Add a new node with $m$ links to the existing nodes in the network. The probability $\prod(k_i)$ that node $i$ is linked to the new node is defined by equation 13.
3) Repeat step 2 for growth.

$$\prod(k_i) = \frac{k_i}{\sum_j k_j} \tag{13}$$

*1) Degree Dynamics:* The dynamics intended to be identified from data by this work is the time evolution of the degree of a single node, in contrast to the analytical approach [6]. Considering a network in which new nodes will link to the network with $m$ edges, the time evolution of the network following this model, *i.e.* the rate at which the degree of a specific node $i$ changes over time, is given by equation 14. Considering $t \gg 1$, we obtain the degree dynamics expressed in equation 15.

$$\frac{dk_i}{dt} = m\prod(k_i) = m\frac{k_i}{\sum_{j=1}^{N-1} k_j} = m\frac{k_i}{2mt - m} \tag{14}$$

$$\frac{dk_i}{dt} = \frac{k_i}{2t} \tag{15}$$

### F. Bianconi-Barabási Model

In the Bianconi-Barabási model nodes also have a *fitness* level $\eta$ that makes a node more or less attractive to receive new connections, aside from the degree; this model will be referred to as the BA *Fitness* model. This new attachment rule means that a new node will connect with $m$ links and fitness $\eta_i$. The fitness $\eta_i$ is a random number from the distribution $\rho(\eta)$, and it does not change once set. To describe this new property of the nodes, the probability of receiving a link from equation 13 is changed to equation 16. This change in the equation leads to the degree dynamics expressed in equation 17. For this work, a simple case with two fitness levels, the consideration lies in whether the methods can differentiate the fitness levels by the identified dynamics.

$$\prod(k_i) = \frac{\eta_i k_i}{\sum_j \eta_j k_j} \tag{16}$$

$$\frac{\partial k_i}{\partial t} = m\frac{\eta_i k_i}{\sum_j \eta_j k_j} \tag{17}$$

### G. Set of Basis Functions

Considering that from the dictionary $\mathcal{D}$ of $m$ basis functions, the set of functions $\mathcal{M}_i \subset \mathcal{D}$ exactly describe the dynamics of $x_i$, where $x_i$ is the time evolution of state $i$ of the system, where $i \in \{1, \ldots, n\}$. If a method estimates that the set of dynamics that describe $x_i$ is $\mathcal{E}_i$, the method correctly identified the system's dynamics *iff* $\forall_i\ \mathcal{M}_i \subseteq \mathcal{E}_i$. Success in the individual and overall system identification is measured as in equations 18 and 19; these metrics will be referred to as Correct Individual Identification (CII) and Correct Overall Identification (COI).

$$\mathrm{C_i} = [\mathcal{M}_1 \subseteq \mathcal{E}_1 \ldots \mathcal{M}_n \subseteq \mathcal{E}_n]^T \tag{18}$$

$$\mathrm{C_o} = \forall_i\ \mathcal{M}_i \subseteq \mathcal{E}_i \tag{19}$$

It is helpful in this project to show whether the methods correctly identified the dynamics directly; in further works where a solution is unknown, actual implementations of these methods, it is better to rely on sparse and error based metrics.

### H. Sparsity

The number of non-zero entries in a vector is called sparsity and can be obtained via the $l_0$-norm in equation 20. The $l_0$-norm can then calculate the sparsity of a resulting model set of dynamics. Considering a dictionary $\mathcal{D}$ of $m$ basis functions, vector $\xi_i \in \mathbb{R}^m$ represents the multiplicative coefficients of the basis functions in $\mathcal{D}$ that describe the dynamics of $x_i$. Then for a system with $n$ states, the sparsity of the dynamics $s_i$ for state $i$ can be obtained via $\|\xi_i\|_0$, which means that $s \in \mathbb{Z}^n$ represents the sparsity of the model for the $n$-dimensional system.

$$\|\mathbf{x}\|_0 = \sum_{n=1}^{N} \mathbf{1}\{x_n \neq 0\} \tag{20}$$

$$s = [\|\xi_1\|_0 \dots \|\xi_n\|_0]^T \qquad \mathcal{C} = \sum_{n=1}^{N} s_n \tag{21}$$

Furthermore, $\Delta s$ represents the difference between the sparsity of the true model and of the identified model. Ideally, $\Delta s$ would be zero, but it is critical to keep in mind that this metric only represents how much more complex the identified model is to the system and alone does not represent the true success of the method. A final indicator of the model's simplicity, the complexity $\mathcal{C}$, is defined as the sum over the sparsity.

### I. Error Estimation

Given the measurements, $Y \in \mathbb{R}^{k\times n}$, where there are $n$ states and $k$ measurements time-wise, as the validation data generated from simulating the system with starting conditions $Y_0$, and the predicted data $\widehat{Y} \in \mathbb{R}^{k\times n}$ generated from the identified model considering $Y_0$ as starting conditions, the quality of the model as a predictor of the system may be evaluated through the following metrics.

*1) Mean Bias Error:* The Mean Bias Error (MBE) represents the predictor's forecast bias; it shows whether the model tends to over or under predict. An MBE of zero means an unbiased estimator, not necessarily without error. MBE is calculated as the average of the actual value minus the predicted value, as seen in equation 22.

$$\text{MBE}(\widehat{Y}) = \frac{1}{nk}\sum_{i=1}^{n}\sum_{j=1}^{k} y_{ij} - \widehat{y}_{ij} \tag{22}$$

*2) Mean Squared Error:* Mean Squared Error (MSE) gives an excellent idea of the amount of error in the prediction by averaging the squares of the errors as seen in equation 23. We can also decompose the MSE through the variability and bias of the prediction $\widehat{Y}$, so the predictor needs to have both high accuracy and high precision for it to have a low MSE. Naturally, an MSE of zero would represent a perfect predictor.

$$\text{MSE}(\widehat{Y}) = \frac{1}{nk}\sum_{i=1}^{n}\sum_{j=1}^{k} (y_{ij} - \widehat{y}_{ij})^2 = \text{Var}(\widehat{Y}) + \text{Bias}(\widehat{Y}, Y)^2 \tag{23}$$

*3) Root Relative Squared Error:* A metric expanding the MSE is the Root Relative Squared Error (RRSE), in which the error of the predictor is compared to the error of the predictions made by a naive predictor as observed in equation 25, the average of the values from equation 24. Given the importance of finding a simple model that still predicts the system's behaviour correctly, this metric may give insight into the complexity of the model.

$$\overline{y} = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k} y_{ij} \tag{24}$$

$$\text{RRSE}(\widehat{Y}) = \sqrt{\frac{\sum_{i=1}^{n}\sum_{j=1}^{k}(y_{ij} - \widehat{y}_{ij})^2}{\sum_{i=1}^{n}\sum_{j=1}^{k}(y_{ij} - \overline{y})^2}} \tag{25}$$

## III. Implementation

This project was done in Python 3.9.7 with PySINDy [3] version 1.4.3 for the SINDy method, SysIdentPy [4] version 0.1.8 for the NARMAX method, and NetworkX version 2.6.3 as a baseline for the simulation of the network models.

### A. Networks Generation

An expansion of NetworkX's BA generator network generator was done, that calculated the degree evolution of all the nodes in the network as it was being created; additionally, a generator for the BA *Fitness* was also implemented. The generation of each consisted of 10000 nodes, each one with two new links; in a single realisation of a network given its stochastic nature and on the average of 100 networks to compare performance on the expected degree evolution. In general the following models were generated:

- Barabási–Albert
- Bianconi-Barabási with *Fitness* levels $[0.991, 0.223]$.
- Bianconi-Barabási with *Fitness* levels $[0.75, 0.25]$.
- Bianconi-Barabási with *Fitness* levels $[0.625, 0.375]$.
- Bianconi-Barabási with *Fitness* levels $[0.5625, 0.4375]$.
- Bianconi-Barabási with *Fitness* levels $[0.53125, 0.46875]$.

### B. SysIdentPy Adjustments

Given the limitation of basis functions set in the SysIdentPy version, the multiplicative inverse time was used as the input of the transfer functions to identify the rational relation from equation 15. Additionally, the NARMAX method found a transfer function for every state, in which the estimated transfer function of the state had as an output the state estimated and all the other states as inputs; generalising this process allowed for simple usage of NARMAX on systems like the State-Space models from section II-C.

### C. PySINDy Adjustments

For integrating time into the dynamics, equation 26 was the auxiliary basis function added to the library of candidate functions. Additionally, time was considered as the input to the system, using SINDy with Model Predictive Control (MPC) [7]. SINDy with control also allowed the usage of time-delays to find the solution to the systems described in section II-D; here, the inputs were time-shifted to account for different time-delays.

$$f_c(x, y) \triangleq x/y \tag{26}$$

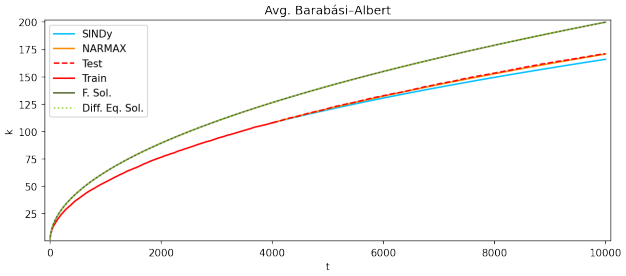



Figure 3: Estimations of the SINDy and NARMAX methods from the identified model of the initial node from the BA model averaged over 100 realisations.

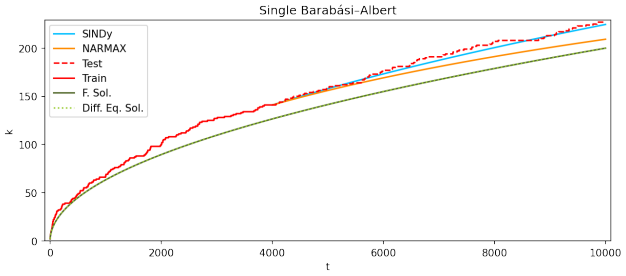



Figure 4: Estimations of the SINDy and NARMAX methods from the identified model of the initial node from the single BA model realisation.

Table I: Metrics data frame for the methods estimating the evolution of the initial node of the BA model averaged over 100 realisations.

| Method | CII | $\Delta s$ | MBE | MSE | RRSE | Time (s) |
|---|---|---|---|---|---|---|
| SINDy | [T] | [0] | 2.94 | 1.11e+1 | 1.14e+2 | 3.20e-2 |
| NARMAX | [T] | [3] | 4.44e-1 | 2.65e-1 | 1.10e+2 | 1.77e-1 |

Table II: Metrics data frame for the methods estimating the evolution of the initial node of the single BA model realisation.

| Method | CII | $\Delta s$ | MBE | MSE | RRSE | Time (s) |
|---|---|---|---|---|---|---|
| SINDy | [T] | [0] | 2.27 | 1.04e+1 | 1.11e+2 | 1.27e-2 |
| NARMAX | [T] | [2] | 9.20 | 1.15e+2 | 1.18e+2 | 1.67e-1 |

## IV. Results

All the systems were simulated, and the generated data was split into an initial set for training, identification, and subsequent set for validating the identified model. The following results do not show COI, sparsity, and complexity for simplicity; detailed results are available in this project's repository [8].

### A. Barabási–Albert Model

The objective was to identify the degree evolution of the first node in the simulated data of the BA model according to equation 15. The methods had the first 40% of the evolution for training and attempted to predict the rest. Identification was done on the ensemble average of 100 networks considering the degree evolution as a random process using the NARMAX and SINDy methods, and on the realisation of a single network.

Both methods successfully identify the specific term of the evolution according to equation 15 in the ensemble average case, with NARMAX adding additional terms that allow it to improve its predictions significantly and, therefore, its estimation error. For reference, the curve describing the dynamics is plotted with "Diff. Eq. Sol." showing the evolution according to equation 15 and "F. Sol." showing the corresponding time function.

The single realisation case gives an insight into the capability of the methods on model identification when the data is from a stochastic nature. SINDy shows excellent performance in this case, but NARMAX's tendency to overfit leads it to a high order term that significantly affects the prediction. This error can be omitted by reducing the order of polynomials considered, but this would mean giving more *a priori* information than the one SINDy needed.

### B. Bianconi-Barabási Model

It is essential to consider that the BA *Fitness* model has different dynamics to BA's model. Accordingly, the identification is considered successful if the time-degree relationship can be identified and the corresponding coefficient is significantly different between the bigger and smaller *fitness* values. For reference, a solution based on equation 17 is plotted in green along with the simulated and predicted values. In each plot, the bigger *fitness* value is referred to as "upper" and the smaller one as "lower".

For the ensemble average identification, both methods successfully identify a dynamic equation that relates the degree with the inverse time, with coefficients that correspond to the *fitness* levels; higher *fitness* level with a higher coefficient. In the single realisation case NARMAX completely fails to identify a sensible dynamic for the bigger *fitness* value. Also, there is no estimation error advantage of NARMAX over the SINDy method. SINDy shows overall a balance in model complexity and small estimation error. These results do not show that the methods identify the actual *fitness* value but that different equations can be identified between different *fitness* values, as close as $0.0625$.

Table III: Metrics data frame for the methods estimating the evolution of initial node of the BA *Fitness* model with discrete fitness levels $[0.53125, 0.46875]$ averaged over 100 realisations estimating a node with *Fitness* $0.46875$.

| Method | CII | $\Delta s$ | MBE | MSE | RRSE | Time (s) |
|---|---|---|---|---|---|---|
| SINDy | [T] | [0] | 1.44 | 2.86 | 1.13e+2 | 8.01e-2 |
| NARMAX | [T] | [2] | 1.67 | 4.74 | 1.15e+2 | 5.31 |

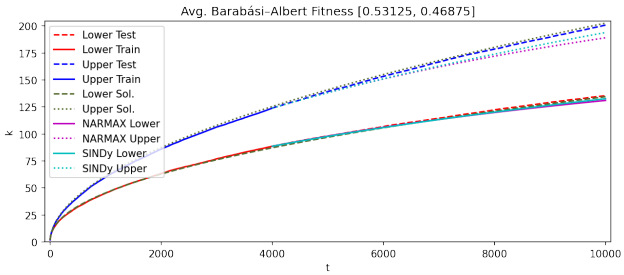



Figure 5: Estimations of the SINDy and NARMAX methods from the identified model of the initial node from the BA *Fitness* model with discrete fitness levels $[0.53125, 0.46875]$ averaged over 100 realisations.

Table IV: Metrics data frame for the methods estimating the evolution of initial node of the BA *Fitness* model with discrete fitness levels $[0.53125, 0.46875]$ averaged over 100 realisations estimating a node with *Fitness* $0.53125$.

| Method | CII | $\Delta s$ | MBE | MSE | RRSE | Time (s) |
|---|---|---|---|---|---|---|
| SINDy | [T] | [0] | 3.41 | 1.57e+1 | 1.14e+2 | 1.26e-2 |
| NARMAX | [T] | [2] | 4.80 | 3.51e+1 | 1.18e+2 | 2.50 |

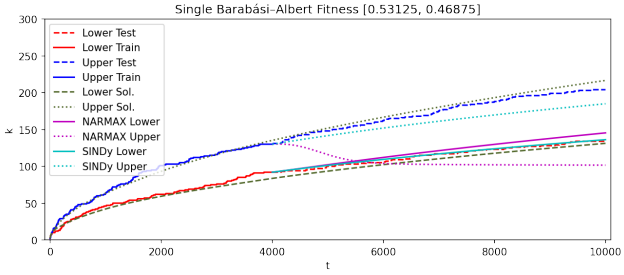



Figure 6: Estimation from the SINDy and NARMAX methods of the BA *Fitness* model with discrete fitness levels $[0.53125, 0.46875]$.

Table V: Metrics data frame for the methods estimating the BA *Fitness* model with discrete fitness levels $[0.53125, 0.46875]$ estimating a node with *Fitness* $0.46875$.

| Method | CII | $\Delta s$ | MBE | MSE | RRSE | Time (s) |
|---|---|---|---|---|---|---|
| SINDy | [T] | [0] | -1.37 | 3.06 | 1.11e+2 | 4.03e-2 |
| NARMAX | [T] | [5] | -5.72 | 3.93e+1 | 9.90e+1 | 9.75e-2 |

Table VI: Metrics data frame for the methods estimating the BA *Fitness* model with discrete fitness levels $[0.53125, 0.46875]$ estimating a node with *Fitness* $0.53125$.

| Method | CII | $\Delta s$ | MBE | MSE | RRSE | Time (s) |
|---|---|---|---|---|---|---|
| SINDy | [T] | [0] | 1.35e+1 | 2.21e+2 | 1.13e+2 | 1.17e-2 |
| NARMAX | [T] | [3] | 6.64e+1 | 5.28e+3 | 8.15e+1 | 7.08e-2 |

## V. Conclusions

SINDy and NARMAX were selected, aiming for sparse-based methods that broadly represented the standard nonlinear system identification methods. Which were compared with sparse and error based metrics, as well as per term model identification success; these were selected to give a broad insight into model representation, model sparsity to evaluate the simplicity of the models, accuracy estimation as the most common metrics, and time for insight into real-time applications. As a baseline for testing, the methods were tested on ODE systems, such as chaotic systems, nonlinear transfer functions with random input, and the BA model with the extension of BA with *fitness*. Both methods perform pretty well and correctly identify the underlying dynamics; SINDy tends for sparser models while NARMAX for minimised estimation errors.

The methods' performance on the Barabási–Albert model were compared, giving insight into their capability to identify node degree evolution, both on an ensemble average of the model and a single realisation. Additionally, the methods were compared in their capability to identify different equations between two nodes with different *fitness* values in a Bianconi-Barabási model. Both methods succeeded at identifying the underlying dynamics of the BA model. Although SINDy successfully identifies distinct equations between two nodes with different *fitness* values in a BA Fitness model, NARMAX failed when only a single realisation was considered. Overall, there is some indication that SINDy performs better than NARMAX at this task, and given the flexibility of the model for extensions, some ideas implemented in the NARMAX method could improve SINDy, for example, the usage of information criterion and inputs with lag. Additionally, SINDy consistently identified the dynamics faster than NARMAX, which may allow real-time implementation on some applications.

In conclusion, the main contributions of this work are:

1) This work over-viewed different methods and metrics.
2) Modified the generator for the BA model that gives the degree evolution of the nodes in the network.
3) Created a generator for the BA *Fitness* model that also returns the degree evolution of the nodes in the network.
4) Used the current SINDy with control to consider lag in the input.
5) Adapted the current NARMAX implementation to find the governing equations for multidimensional systems.
6) It proposed an approachable way to include rational basis functions for the Polynomial NARMAX basis library current implementation in SysIdentPy.
7) It proposes a metrics data frame for nonlinear system identification methods comparison.

## ABBREVIATIONS

| | |
|---|---|
| AIC | Akaike Information Criterion |
| BA | Barabási–Albert |
| BIC | Bayesian Information Criterion |
| CII | Correct Individual Identification |
| COI | Correct Overall Identification |
| ERR | Error Reduction Ratio |
| FPE | Final Prediction Error |
| FROLS | Forward Regression Orthogonal Least Squares |
| IHT | Iterative Hard-Thresholding |
| LASSO | Least Absolute Shrinkage and Selection Operator |
| LILC | Law of Iterated Logarithm Criterion |
| LS | Least Squares |
| MBE | Mean Bias Error |
| MISO | Multiple-Input Single-Output |
| MPC | Model Predictive Control |
| MSE | Mean Squared Error |
| NARMAX | Nonlinear AutoRegressive Moving Average with eXogenous input |
| ODE | Ordinary Differential Equation |
| OLS | Orthogonal Least Squares |
| OMP | Orthogonal Matching Pursuit |
| RLS | Recursive Least Squares |
| RRSE | Root Relative Squared Error |
| SINDy | Sparse Identification of Nonlinear Dynamics |
| SISO | Single-Input Single-Output |
| STLSQ | Sequential Thresholded Least-Squares |

## REFERENCES


[1] S. A. Billings, *Nonlinear System Identification: NARMAX Methods in the Time, Frequency, and Spatio–Temporal Domains*. Hoboken, New Jersey: John Wiley & Sons, Ltd, 2013.

[2] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proceedings of the National Academy of Sciences*, vol. 113, no. 15, pp. 3932–3937, 2016.

[3] B. de Silva, K. Champion, M. Quade, J.-C. Loiseau, J. Kutz, and S. Brunton, "Pysindy: A python package for the sparse identification of nonlinear dynamical systems from data," *Journal of Open Source Software*, vol. 5, no. 49, p. 2104, 2020.

[4] W. R. Lacerda, L. P. C. da Andrade, S. C. P. Oliveira, and S. A. M. Martins, "Sysidentpy: A python package for system identification using narmax models," *Journal of Open Source Software*, vol. 5, no. 54, p. 2384, 2020.

[5] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, p. 509–512, Oct 1999.

[6] A.-L. Barabási, R. Albert, and H. Jeong, "Mean-field theory for scale-free random networks," *Physica A: Statistical Mechanics and its Applications*, vol. 272, no. 1, pp. 173–187, 1999.

[7] U. Fasel, E. Kaiser, J. N. Kutz, B. W. Brunton, and S. L. Brunton, "Sindy with control: A tutorial," 2021.

[8] D. Contreras Franco, "Data-driven Dynamics Understanding." https://github.com/DavidContrerasFranco/Data-driven-Dynamics-Understanding, 2021.