PANE

P. J. Phys. Edu. Vol 1(1), 2026 PANE Journal of Physics Education

A TUTORIAL FOR SPARSE IDENTIFICATION OF NONLINEAR DYNAMICAL SYSTEM (SINDy) FRAMEWORK TO DISCOVER GOVERNING EQUATIONS FROM RAW DATA

Britan Singh[1], Mani K Chettri[1], Vivek Shrivastav[1], Hemam D Singh[2], and Rupak Mukherjee[*1]

[1]*Department of Physics, Sikkim University, Gangtok, Sikkim, India*
[2]*Department of Physics, Netaji Subhas University of Technology, New Delhi, India*

[*]Corresponding Author: rmukherjee@cus.ac.in

Abstract: The discovery of governing equation from time-series data is a challenge in the analysis of nonlinear dynamical systems. In this work, a clear and systematic exposition of the Sparse Identification of Nonlinear Dynamical System (SINDy) framework is presented for data-driven equation discovery. The method is formulated through system representation, construction of a candidate function library, and identification of sparse governing terms. The approach is validated using 1D, 2D, and 3D systems, including the logistic model, nonlinear pendulum, and Lorenz system. The coefficients we found shows that only the actual terms from the true equation are important, unnecessary terms are correctly ignored. These results show that SINDy is an effective method for discovering the governing equations of nonlinear physical systems directly from data.

Keywords: SINDy; Dynamical System; Nonlinear System; Governing Equations

(Received 14 February 2026; accepted 14 March 2026; published 6 April 2026)

1 Introduction

The identification of governing equations has been a fundamental objective in the physical sciences. Traditionally, physical laws are derived from first principles through analytical reasoning, symmetry considerations, and conservation laws [1]. This approach worked well for simple systems, but it runs into problems when dealing with complex, nonlinear behavior [1]. In these cases, the interactions are so complicated that it becomes very difficult to derive the governing equations using traditional methods. However, the advancement of experimental tools, and numerical simulations provide us data rich research environments [2, 3]. Because traditional methods struggle with nonlinear systems, researchers are now turning to data-driven methods. These methods let the data itself reveal the underlying physical law, without needing to start from first principles [4].

Early work in data-driven methods focused on techniques that automatically discover equations from data, such as symbolic regression [5, 6]. These methods showed that it is possible to discover meaningful physical laws, without having prior knowledge of the governing equations. Symbolic regressions frameworks have been successfully applied to recover conservation laws and dynamical models from experimental and simulated datasets [7]. However, these early approaches come with challenges. They often have to search through a large number of possible equations, which demands significant computational power and time. They can also produce very complicated equations that are hard to understand or interpret physically.

In recent years, Sparse Identification of Nonlinear Dynamical System (SINDy) framework has emerged as a powerful for data-driven model discovery [1, 2, 3, 4]. The main idea behind SINDy is that the governing equations are sparse in nature [3, 4]. SINDy works by first estimating derivatives from time series data, then building a large

library of possible candidate functions, and finally using regression technique to pick out only the important terms [3]. This results in a simple, compact model that is easy to interpret. Compared to older methods like symbolic regression, SINDy is much faster and less likely to produce overly complicated equations [3, 4]. Because of this, SINDy is especially useful for uncovering the rules that govern nonlinear and complex systems.

In this work, we present a clear and straightforward explanation of the SINDy method, focusing on how it can be used to discover the governing equations of physical systems. We use classical benchmark systems, including the logistic model, simple pendulum, and Lorenz system, as illustrative examples [5].

2 Dynamical Systems and Time-Series Data

Many physical systems can be described within the framework of dynamical system, where the state of the system evolves with time according to an underlying governing law [1]. Mathematically a dynamical system can be represented as

$$\frac{dX}{dt} = f(X). \tag{1}$$

Here, $X(t) = (x_1, x_2, x_3....)$ denotes the state of the given system, and $f(X)$ in an unknown function that determines how the system evolves. Traditionally the $f(X)$ is derived from first principle based on physical law [5]. In contrast, data-driven approaches assume that the governing function is unknown and must be derived directly from observational data [3, 4]. In real experiments or simulations, we collect time-series data by measuring the system's state at specific time points and these collected data can be arranged into a data-matrix

$$\mathbf{X} = \begin{bmatrix} X(t_1) \\ X(t_2) \\ \vdots \\ . \\ X(t_m) \end{bmatrix}, \tag{2}$$

where m is the number of time samples [4]. In practical situations,the time derivatives of the state variables are usually not measured directly. Therefore, the derivatives must be estimated numerically from the available time-series data [3]. This is an important step because any error in derivative estimation can affect the accuracy of the discovered governing equations [3, 4]. The estimated derivatives can be arranged into a derivative matrix given by

$$\dot{\mathbf{X}} = \begin{bmatrix} \dot{X}(t_1) \\ \dot{X}(t_2) \\ \vdots \\ . \\ \dot{X}(t_m) \end{bmatrix}, \tag{3}$$

where $\dot{X}(t_i)$ represents the time derivative of the state vector at time t_i. The main goal is to determine the relationship between $f(X)$ and $\dot{X}$ using the available time series data.

3 Mathematical formulation of "SINDy" method

The "Sparse Identification of Nonlinear Dynamical system" ("SINDy") method is a data-driven framework for method governing equation directly from time-series data [1, 3, 4]. As discussed in earlier section, the system dynamics can be expressed as

$$\dot{X} = f(X), \tag{4}$$

where $X(t) \in \mathbb{R}^n$ denotes the state-vector of the system and $f(X)$ is an unknown nonlinear function governing the temporal equation of the system [4, 8]. In many practical cases, the explicit analytical form of $f(X)$ is not available and must be inferred directly from observational or simulated time-series data. The central idea of the SINDy framework is to approximate the unknown function $f(X)$ using a set of candidate basis functions constructed from the measured data. These candidate functions are organized into library matrix, denoted by

$\Theta(X)$, which contains various possible linear and nonlinear combinations of the state variables. A general form of the library matrix can be written as

$$\Theta(X) = \begin{bmatrix} 1 & x_1(t_1) & x_2(t_1) & x_1^2(t_1) & x_1(t_1)x_2(t_1) & \cdots & \cdot \\ 1 & x_1(t_2) & x_2(t_2) & x_1^2(t_2) & x_1(t_2)x_2(t_2) & \cdots & \cdot \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \\ \cdot & \cdot & \cdot & \cdot & \cdot & & \\ 1 & x_1(t_m) & x_2(t_m) & x_1^2(t_m) & x_1(t_m)x_2(t_m) & \cdots & \cdot \end{bmatrix}. \tag{5}$$

Where each row corresponds to a time snapshot of the system and each column represents a candidate function constructed from the state variables [3, 4]. The first column usually denotes a constant term, while the remaining columns include linear terms, polynomial terms, and other nonlinear functions depending on the expected physics of the system.

The key assumption underlying the SINDy method is that the true governing dynamics are sparse in the space of candidate functions. This means that, although the library matrix may contain a large number of possible terms, only a few small subset of these terms are required to accurately describe our system dynamics [1]. Based on this assumption, the system dynamics can be expanded as a linear combination of the candidate library functions,

$$\dot{X} = \Theta(X)\,\Xi, \tag{6}$$

where Ξ is a matrix of coefficients that tells which terms from the library are actually important. The above equation has a form similar to the standard regression problem, therefore the identification of the governing equations can be treated as a regression problem.

To make this formulation more explicit, the time-series measurements are first arranged into a state data matrix and a corresponding time-derivative matrix. The derivative matrix is given by

$$\dot{X} = \begin{bmatrix} \dot{x}_1(t_1) & \dot{x}_2(t_1) & \cdots & \cdot & \dot{x}_n(t_1) \\ \dot{x}_1(t_2) & \dot{x}_2(t_2) & \cdots & \cdot & \dot{x}_n(t_2) \\ \vdots & \vdots & \ddots & \vdots & \\ \dot{x}_1(t_m) & \dot{x}_2(t_m) & \cdots & \cdot & \dot{x}_n(t_m) \end{bmatrix}, \tag{7}$$

where m denotes the number of time snapshots and n is the number of state variables. Each row corresponds to a time instant, while each column represents the temporal evolution of an individual state variable [3, 4].

Similarly, the library matrix $\Theta(X)$ is constructed from candidate nonlinear functions of the measured state variables and can be written in a general form as

$$\Theta(X) = \begin{bmatrix} 1 & \theta_1(X(t_1)) & \theta_2(X(t_1)) & \cdots & \cdot & \theta_p(X(t_1)) \\ 1 & \theta_1(X(t_2)) & \theta_2(X(t_2)) & \cdots & \cdot & \theta_p(X(t_2)) \\ \vdots & \vdots & \vdots & \ddots & \vdots & \\ 1 & \theta_1(X(t_m)) & \theta_2(X(t_m)) & \cdots & \cdot & \theta_p(X(t_m)) \end{bmatrix}, \tag{8}$$

where p is the number of candidate basis functions, including constant, linear, polynomial, and other nonlinear terms depending on the expected physics of the system [5, 9]

With these definitions, Eq. (6) represents a linear regression problem in a high-dimensional feature space, where the objective is to determine the sparse coefficient matrix

$$\Xi = \begin{bmatrix} \xi_{11} & \xi_{12} & \cdots & \xi_{1n} \\ \xi_{21} & \xi_{22} & \cdots & \xi_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \xi_{p1} & \xi_{p2} & \cdots & \xi_{pn} \end{bmatrix}. \tag{9}$$

Each column of Ξ corresponds to the governing equation of a specific state variable, indicating which candidate functions actively contribute to the system dynamics, while the remaining coefficients remain close to zero due to the sparsity assumption.

4 Least Squares Identification

In the present work, the coefficient matrix x_i is obtained using a classical least squares approach. After constructing the library matrix $\Theta(X)$ and the derivative vector $\dot{X}$ from the time series data, the governing equation

$$\dot{X} = \Theta(X)\Xi, \tag{10}$$

is treated as linear regression problem. specifically, the coefficients are computed by solving the normal equations

$$\Theta(X)^T\Theta(X)\,\Xi = \Theta(X)^T\dot{X}, \tag{11}$$

which correspond to the standard ordinary least square solution [4]. In the numerical implementation, the matrices $\Theta^T\Theta$ and $\Theta^T\dot{X}$ are explicitly constructed and the resulting linear system in solved using Gaussian elimination.

Although ordinary least squares regression does not explicitly enforce sparsity, it provides a straightforward and computationally efficient method for identifying the important terms in the governing equations. More broadly, modern machine learning and data-driven modeling approaches have played a significant role in discovering governing equations directly from data [10, 11, 12].

Steps	Description
1	Collect time-series data of the state variables $X(t)$ from experiments or numerical simulations.
2	Estimate the time derivatives $\dot{X}(t)$ using an appropriate numerical differentiation method.
3	Construct the candidate function library matrix $\Theta(X)$ using linear, polynomial, or nonlinear combinations of the state variables.
4	Formulate the regression problem in the form $\dot{X} = \Theta(X)\Xi$.
5	Compute the initial coefficient matrix Ξ using regression methods.
6	Reconstruct the governing equations from the identified coefficients, consistent with sparse representation and compressed sensing principles [13].

Table 1: Step-by-step algorithm of the SINDy method for equation discovery

The SINDy formulation $\dot{X} = \Theta(X)\Xi$ follows the standard framework of sparse regression for dynamical systems described in data-driven modeling literature [4, 10, 14].

5 Illustrative Examples with Code

To demonstrate the effectiveness of the SINDy method [14] we use several classical dynamical systems as examples [1]. These benchmark systems are chosen because their governing equations are well known, which allows us direct comparison between the identified model and the true model.

5.1 Logistic Model

The logistic growth equation,

$$\frac{dN}{dt} = rN\left(1 - \frac{N}{K}\right), \tag{12}$$

is a classical nonlinear model that describes population growth in the presence of limited resources [1], where r is the intrinsic growth rate and K is the "carrying capacity" of the system [5]. The term rN represents exponential growth, while the quadratic term $-\frac{r}{K}N^2$ introduces nonlinear saturation as the population approaches the carrying capacity.

Due to the presence of a simple quadratic nonlinearity, the logistic model serves as an ideal benchmark for testing data-driven discovery methods such as genetic programming, symbolic regression and SINDy [8, 15, 16, 17].

To demonstrate the capability of the SINDy method in discovering governing equations from data, we first generate time-series data $N(t)$ using the logistic growth model. The corresponding time derivative $\dot{N}(t)$ is then computed.

In this implementation, a candidate library is chosen as

$$\Theta(X) = [1,\ x,\ x^2,\ x^3,\ x^4,\ x^5,\ \sin(x),\ \cos(x)] \tag{13}$$

which includes polynomial as well as trigonometric terms, The purpose of choosing an extended library is to test whether the SINDy framework can correctly identify the true governing dynamics even in the presence of many irrelevant candidate functions. The SINDy algorithm then determines the coefficient vector associated with each term in the library.

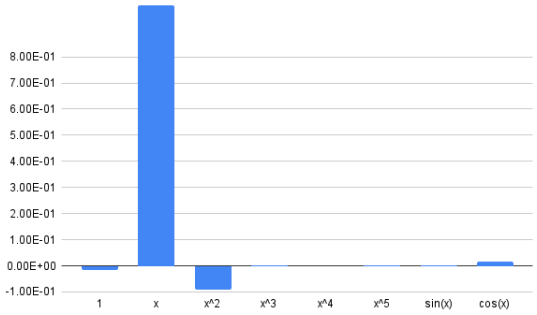

Figure 1: Coefficients identified by SINDy for the logistic growth model.

Fig. 1 illustrates the identified coefficients for the logistic growth model. It is clearly observed that the dominant coefficients correspond to the linear term x and the quadratic term x^2 while the higher order polynomial and trigonometric terms have coefficients close to zero.

5.2 Simple Pendulum

The simple pendulum is a well known nonlinear dynamical system widely studied in classical mechanics and nonlinear dynamics [1], and its equation of motion is given by

$$\ddot{\theta} + \frac{g}{L}\sin(\theta) = 0 \tag{14}$$

where θ denotes the angular-displacement, g, the acceleration-due-to-gravity, and L, the length of the pendulum. Unlike the linearized small angle approximation, the presence of the $\sin(\theta)$ term makes the system inherently nonlinear, providing an ideal model for testing data driven method [5].

In order to apply the SINDy framework, the above second order differential equation (14), must be converted into a system of two first order equations. This is achieved by defining the state variables as

$$x = \theta \tag{15}$$

$$y = \dot{\theta} \tag{16}$$

with this transformation, the simple pendulum dynamics can be expressed as a two dimensional dynamical system, In this case the candidate library is constructed using constant, polynomial and trigonometric terms of the state variables

$$\Theta(X) = [1, x, y, xy, x^2, y^2, \sin(x), \sin(y), ...] \tag{17}$$

This library allows the SINDy algorithm to identify the relevant nonlinear terms governing the pendulum dynamics.

$$\dot{x} = y \tag{18}$$

$$\dot{y} = -\frac{g}{L}\sin(x) \tag{19}$$

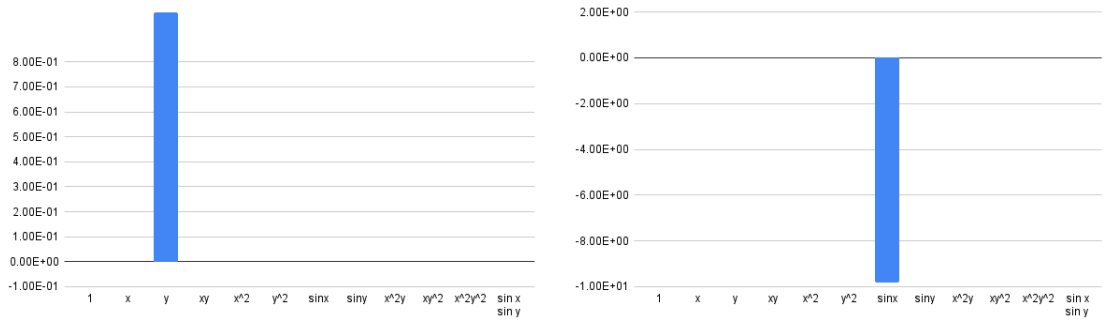

Figure 2: Identified coefficients for the Simple pendulum system using SINDy method.

Fig. 2 Presents the identified coefficient vectors obtained using the SINDy method for the simple pendulum system. The coefficient plot clearly shows that the dominant term in the first equation corresponds to the linear velocity y, which correctly represents the relation $\dot{x} = y$. In the second equation, the largest coefficient is associated with the trigonometric term $\sin(x)$, while the constant, polynomial, and cross terms exhibit coefficients that are close to zero.

5.3 Lorenz System

The "Lorenz system" is a famous example of a nonlinear dynamical system showing chaos [1]. Due to its strong nonlinear coupling and sensitive dependence on initial conditions, it is widely used as a benchmark problem in nonlinear dynamics and data-driven model discovery [5, 14, 18]. The governing equations of the Lorenz system are given by

$$\dot{x} = \sigma\,(y - x) \tag{20}$$
$$\dot{y} = x\,(\rho - z) - y \tag{21}$$
$$\dot{z} = xy - \beta z, \tag{22}$$

where σ, ρ, and β are positive parameters that control the behavior of the system [5].

In this example, synthetic data for the state variables $x(t)$, $y(t)$, and $z(t)$ are generated numerically from the Lorenz equations using standard parameter values. The data are then re-arranged into a state matrix $X(t) = [x(t),\ y(t),\ z(t)]^T$, and the corresponding time derivative are computed to construct $\dot{X}(t)$. A candidate library is formed using constant, linear, quadratic, and intersection terms of the state variables [1, 3, 4]. The SINDy method is then applied to obtain coefficient matrices corresponding to the three governing equations.

In Fig. 3 the results indicate that the dominant terms in the recovered equations correspond to the linear coupling between x and y, the nonlinear interaction term xy, and the linear damping term in the z equation. All the other polynomial and higher order library terms have coefficients close to zero, showing that only a few terms significantly contribute to the system dynamics.

In addition to the coefficient analysis, phase space plot are presented to validate the dynamical behavior of the identified model. For chaotic systems such as the Lorenz system, phase portraits provide an effective way to compare the qualitative dynamics between the true system and the SINDy recovered system. From the phase portrait plot (Fig. 4), it is observed that the trajectories obtained from the SINDy model closely follow the true Lorenz attractor in all state space projection. This close agreement indicates that the SINDy identified model successfully reproduces the essential nonlinear and chaotic dynamics of the system.

6 Conclusion

In this work, we provided a clear and step by step explanation of the SINDy method, which is used to discover the equations governing a dynamical system directly from data. We started by explaining how dynamical systems are typically described mathematically in the framework of nonlinear dynamics [1]. Then, we showed how to build a

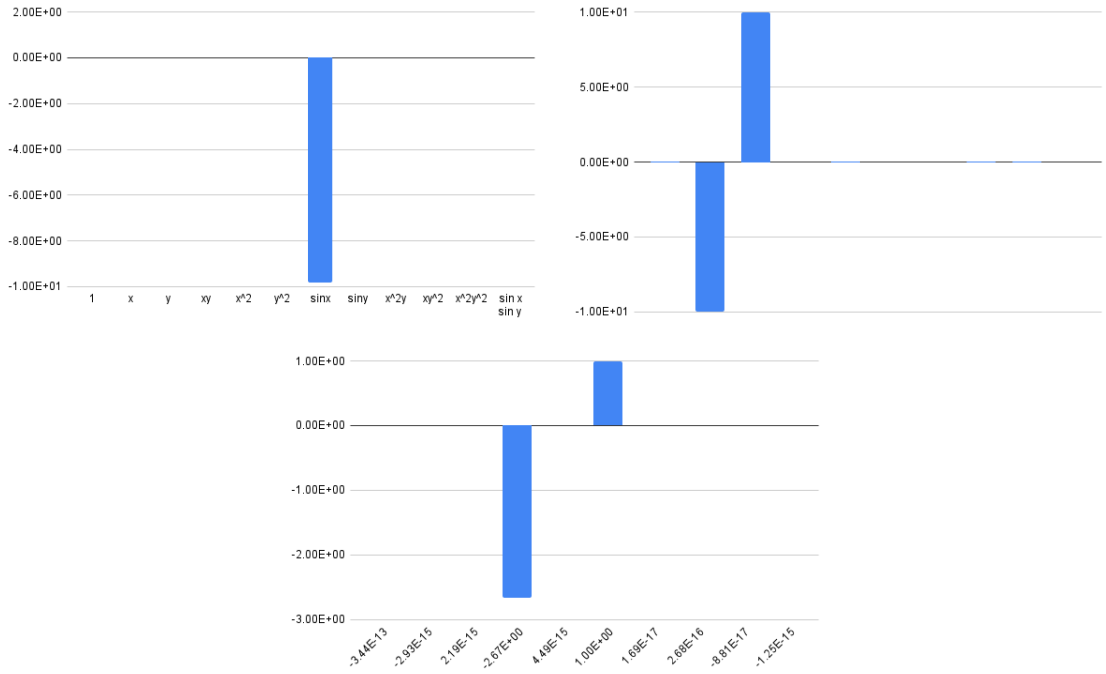

Figure 3: Identified coefficients for the Lorenz system using the SINDy method.

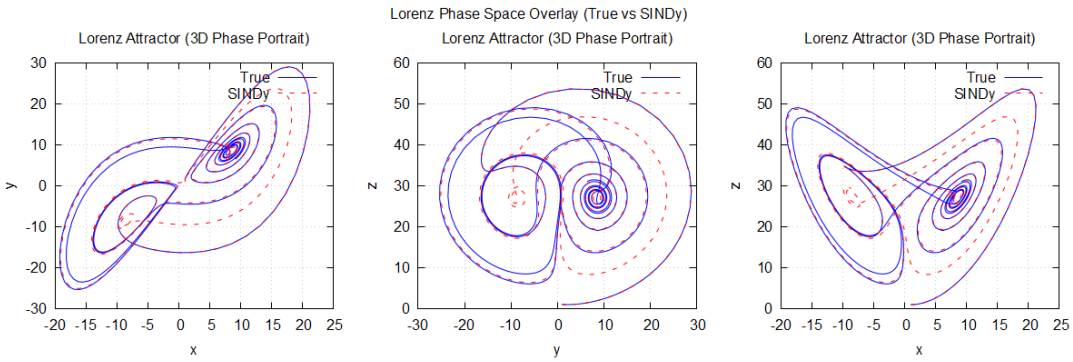

Figure 4: Comparison of true and SINDy reconstructed Lorenz attractor in phase space projections.

library of possible terms and use regression techniques to identify which ones actually appears in the true equations. The effectiveness of the SINDy method was demonstrated using several dynamical systems, including the logistic model, the nonlinear pendulum, and the Lorenz system which serve as representative nonlinear dynamical benchmarks [17]. For each case, the identified coefficient structures showed dominant contributions from the true governing terms, while higher order and redundant library terms were suppressed. This indicates that the method successfully isolates the physically relevant dynamics even when an overcomplete candidate library is used. In, particular, the phase space comparisons for the Lorenz system showed close agreement between the true trajectories and the SINDy reconstructed trajectories which confirms that the identified model successfully captures the structure of the chaotic attractor. Overall, the SINDy framework is a simple, interpretable, and computationally efficient tool for data-driven equation discovery in nonlinear physical systems, consistent with modern machine learning-based scientific modeling approaches [2].

7 Acknowledgment

All the computational work reported in this paper was carried out using the ”Brahmagupta” HPC facility at Sikkim University. One of the authors R.M. acknowledges support provided by IUCCA, Pune, through the visiting asso-

ciate program.

A Appendix

This appendix provides the complete Fortran implementations used for the illustrative examples presented in this work. The codes generate time-series data, construct the candidate library matrix, and compute the coefficient vector. Separate codes are provided for the logistic growth model, nonlinear pendulum, and Lorenz system to ensure reproducibility of the results.

A.1 Logistic Model code

```
program sindy_least_square_logistic_model
  implicit none
  integer, parameter::N = 100
  real(kind=8),dimension(N)::t, x,xdot
  real(kind=8),dimension(N,8)::Theta
  real(kind=8),dimension(8)::xi_x
  real(kind=8)::dt,r,K
  integer::i

 dt=0.01d0
  x(1)=1.0d0
  t(1)=0.0d0
  r=1.0d0
  K=10.0d0
  do i=1,N
     xdot(i)= x(i)*(r-x(i)/K)
     x(i+1)= x(i)+xdot(i)*dt
     t(i+1)= t(i) + dt
  end do

  do i=1,N
     Theta(i,1)= 1.0d0
     Theta(i,2)=x(i)
     Theta(i,3)=x(i)**2
     Theta(i,4)=x(i)**3
     Theta(i,5)=x(i)**4
     Theta(i,6)=x(i)**5
     Theta(i,7)=sin(x(i))
     Theta(i,8)=cos(x(i))
  end do

  call least_squares(N,8,Theta,xdot,xi_x)

  print *, "Sparse vector"
  do i=1,8
     print*,xi_x(i)
  end do

end program

  subroutine least_squares(m,n,A,b,x)
    integer:: m, n
    real(kind=8)::A(m,n),b(m)
    real(kind=8)::x(n)
```

```
    real(kind=8),allocatable::AtA(:,:),Atb(:)
    integer::i,j

    allocate(AtA(n,n),Atb(n))
    AtA=0.0d0
    Atb=0.0d0
do i=1,n
 do j=1,n
        AtA(i,j)=sum(A(:,i)*A(:,j))
       end do
    end do

    do i=1,n
       Atb(i)=sum(A(:,i)*b(:))
    end do

    call solve_linear(n,AtA,Atb,x)
    deallocate(AtA,Atb)
  end subroutine

  subroutine solve_linear(n,A,b,x)
    integer:: n
    real(kind=8)::A(n,n)
    real(kind=8)::b(n)
    real(kind=8)::x(n)
    real(kind=8)::factor
    integer::i,j,k

    do k =1,n-1
       do i=k+1,n
          factor=A(i,k)/A(k,k)
          A(i,k:n)=A(i,k:n)-factor*A(k,k:n)
          b(i) = b(i) - factor*b(k)
       end do
    end do

    do i=n,1,-1
       x(i) = (b(i) - sum(A(i,i+1:n)*x(i+1:n)))/A(i,i)
    end do
  end subroutine
```

A.2 Simple Pendulum code

```
    program sindy_least_square_Pendulum
  implicit none
  integer, parameter:: N=100
  real(kind=8),dimension(N)::t,x,xdot,y,ydot
  real(kind=8),dimension(N,12)::Theta
  real(kind=8),dimension(12)::xi_x,xi_y
  real(kind=8)::dt,g,L
  integer :: i

  dt=0.1d0
  x(1)= 1.0d0
  y(1)= 1.0d0
  t(1)= 0.0d0
```

```
  g=9.81d0
  L=1.0d0

  do i=1,N
     xdot(i)=y(i)
     ydot(i)=-(g/L)*sin(x(i))

     x(i+1)= x(i)+xdot(i)*dt
     y(i+1)= y(i)+ydot(i)*dt
     t(i+1)= t(i)*dt
  end do

  do i = 1, N
     Theta(i,1)= 1.0d0
     Theta(i,2)=x(i)
     Theta(i,3)=y(i)
     Theta(i,4)=x(i)*y(i)
     Theta(i,5)=x(i)**2
     Theta(i,6)=y(i)**2
     Theta(i,7)=sin(x(i))
     Theta(i,8)=sin(y(i))
     Theta(i,9)=x(i)**2*y(i)
     Theta(i,10)=y(i)**2*x(i)
     Theta(i,11)=x(i)**2*y(i)**2
     Theta(i,12)=sin(y(i))*sin(x(i))

  end do

  call least_squares(N,12,Theta,xdot,xi_x)
  call least_squares(N,12,Theta,ydot,xi_y)

  print *, "Sparse vector"
  do i=1,12
     print*,xi_x(i)
  end do

   print *, "Sparse vector"
  do i=1,12
     print*,xi_y(i)
  end do

end program

  subroutine least_squares(m,n,A,b,x)
    integer:: m,n
    real(kind=8):: A(m,n),b(m)
    real(kind=8)::x(n)
    real(kind=8),allocatable::AtA(:,:),Atb(:)
    integer::i,j

    allocate(AtA(n,n),Atb(n))
    AtA=0.0d0
    Atb=0.0d0
do i = 1,n
 do j = 1,n
        AtA(i,j)=sum(A(:,i)*A(:,j))
```

```
      end do
   end do

   do i=1,n
      Atb(i)=sum(A(:,i)*b(:))
   end do

   call solve_linear(n,AtA,Atb,x)
   deallocate(AtA,Atb)
end subroutine

subroutine solve_linear(n,A,b,x)
   integer:: n
   real(kind=8)::A(n,n)
   real(kind=8)::b(n)
   real(kind=8)::x(n)
   real(kind=8)::factor
   integer::i,j,k

   do k =1,n-1
      do i=k+1,n
         factor=A(i,k)/A(k,k)
         A(i,k:n)=A(i,k:n)-factor*A(k,k:n)
         b(i)=b(i) - factor*b(k)
      end do
   end do

   do i=n,1,-1
      x(i) = (b(i)-sum(A(i,i+1:n)*x(i+1:n)))/A(i,i)
   end do
end subroutine
```

A.3 Lorenz system code

```
   program sindy_lorenz_least_squares
implicit none
integer, parameter::N= 1000
real(kind=8),dimension(N)::t,x,y,z,xdot,ydot,zdot
real(kind=8),dimension(N,10)::Theta
real(kind=8),dimension(10)::xi_x, xi_y,xi_z
real(kind=8):: dt,sigma,rho,beta
integer::i

sigma=10.0d0
beta=8.0d0/3.0d0
rho=28.0d0
dt=0.01d0

x(1)= 1.0d0
y(1)= 1.0d0
z(1)= 1.0d0
t(1)= 0.0d0

do i=1,N
   xdot(i) = sigma*(y(i)-x(i))
   ydot(i) = x(i)*(rho-z(i))-y(i)
```

```
      zdot(i) = x(i)*y(i)-beta*z(i)

      x(i+1)=x(i)+dt*xdot(i)
      y(i+1)=y(i)+dt*ydot(i)
      z(i+1)=z(i)+dt*zdot(i)
      t(i+1)=t(i)+dt
   end do

   do i = 1,N
      Theta(i,1)= 1.0d0
      Theta(i,2)=x(i)
      Theta(i,3)=y(i)
      Theta(i,4)=z(i)
      Theta(i,5)=x(i)**2
      Theta(i,6)=x(i)*y(i)
      Theta(i,7)=x(i)*z(i)
      Theta(i,8)=y(i)**2
      Theta(i,9)=y(i)*z(i)
      Theta(i,10)=z(i)**2
   end do

   call least_squares(N,10,Theta,xdot,xi_x)
   call least_squares(N,10,Theta,ydot,xi_y)
   call least_squares(N,10,Theta,zdot,xi_z)

   print*, "spars vector"
   do i = 1,10
      print *,xi_x(i)
   end do

   print*, "sparse vector"
   do i = 1,10
      print *, xi_y(i)
   end do

   print*, "sparse vector"
   do i = 1,10
      print *, xi_z(i)
   end do

end program

subroutine least_squares(m,n,A,b,x)
   implicit none
   integer:: m,n
   real(kind=8):: A(m,n), b(m)
   real(kind=8):: x(n)
   real(kind=8),allocatable :: AtA(:,:), Atb(:)
   integer :: i,j

   allocate(AtA(n,n),Atb(n))
   AtA=0.0d0
   Atb=0.0d0

   do i=1,n
      do j=1,n
```

```
        AtA(i,j)=sum(A(:,i)*A(:,j))
     end do
     Atb(i)=sum(A(:,i)*b(:))
  end do

  call solve_linear(n,AtA,Atb,x)
  deallocate(AtA,Atb)
end subroutine

subroutine solve_linear(n,A,b,x)
  implicit none
  integer::n,i,j,k
  real(kind=8):: A(n,n),b(n),x(n),factor

  do k=1,n-1
     do i=k+1,n
        factor=A(i,k)/A(k,k)
        A(i,k:n)=A(i,k:n)-factor*A(k,k:n)
        b(i)=b(i) - factor*b(k)
     end do
  end do

  do i=n,1,-1
     x(i) = (b(i)- sum(A(i,i+1:n)*x(i+1:n)))/A(i,i)
  end do
end subroutine
```

References

[1] S. H. Strogatz, *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering* (Westview Press, Boulder, 1994).

[2] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science* **349**(6245), 255–260 (2015).

[3] V. Marx, "Biology: The big challenges of big data," *Nature* **498**(7453), 255–260 (2013).

[4] S. L. Brunton and J. N. Kutz, *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control* (Cambridge University Press, Cambridge, 2019).

[5] S. L. Brunton, J. H. Tu, I. Bright, and J. N. Kutz, "Compressive sensing and low-rank libraries for classification of bifurcation regimes in nonlinear dynamical systems," *SIAM Journal on Applied Dynamical Systems* **13**(4), 1716–1732 (2014).

[6] J. Bongard and H. Lipson, "Automated reverse engineering of nonlinear dynamical systems," *Proceedings of the National Academy of Sciences USA* **104**(24), 9943–9948 (2007).

[7] C. W. Rowley, I. Mezić, S. Bagheri, P. Schlatter, and D. S. Henningson, "Spectral analysis of nonlinear flows," *Journal of Fluid Mechanics* **645**, 115–127 (2009).

[8] M. Schmidt and H. Lipson, "Distilling free-form natural laws from experimental data," *Science* **324**(5923), 81–85 (2009).

[9] B. C. Daniels and I. Nemenman, "Efficient inference of parsimonious phenomenological models of cellular dynamics using S-systems and alternating regression," *PLoS ONE* **10**(3), e0119821 (2015).

[10] N. Zolman, C. Lagemann, U. Fasel, J. N. Kutz, and S. L. Brunton, "SINDy-RL: Interpretable and Efficient Model-Based Reinforcement Learning," *Nature Communications* **16.1**, 10714 (2025).

[11] J. L. Proctor, S. L. Brunton, B. W. Brunton, and J. N. Kutz, "Exploiting sparsity and equation-free architectures in complex systems (Invited Review)," *European Physical Journal Special Topics* **223**, 2665–2684 (2014).

[12] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B* **58**(1), 267–288 (1996).

[13] D. L. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory* **52**(4), 1289–1306 (2006).

[14] 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 USA* **113**(15), 3932–3937 (2016).

[15] J. R. Koza, "Genetic Programming: On the Programming of Computers by Means of Natural Selection," MIT Press, Cambridge, MA (1992).

[16] J. R. Koza, "Genetic Programming II: Automatic Discovery of Reusable Programs," MIT Press, Cambridge, MA (1994).

[17] A. Sen, D. P. Ahalpara, A. Thyagaraja, and G. S. Krishnaswami, "A KdV-like advection–dispersion equation with some remarkable properties," *Communications in Nonlinear Science and Numerical Simulation* **17**(11), 4117–4127 (2012).

[18] P. J. Schmid, "Dynamic mode decomposition of numerical and experimental data," *Journal of Fluid Mechanics* **656**, 5–28 (2010).