On the Numerical Solution of Differential Equations

Contents

The solution of a differential equation is a function of time (in the context of this text), in other words, the time course of a quantity, time dependence, signal. This function of time can generally be sought as an *analytical solution*, meaning that we use analytical methods to find a mathematical expression of the function. Another option is to seek a *numerical solution*, where we look for a sequence of numerical values (numbers) assigned to time data such that it can be shown that the resulting time sequence of values can be considered a representation of the time function that is the solution of the differential equation.

1 Pendulum as an Example of a Dynamic System for Finding a Numerical Solution

Consider a pendulum whose oscillations are damped by viscous friction with a coefficient β [kg m^2 s^{-1}]. The pendulum is shown in Fig. 1, where a mass point with mass m [kg] attached to an arm with negligible mass and length l [m] oscillates, o denotes the axis of rotation perpendicular to the plane in which the pendulum oscillates, the angle between the vertical and the pendulum arm is denoted by φ [rad], and the gravitational acceleration is g [m s^{-2}].

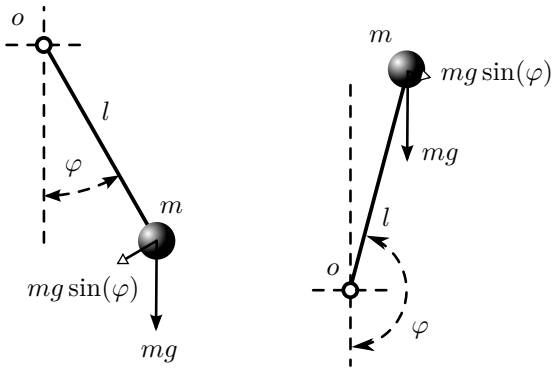

Figure 1: Pendulum

The equation of motion describing the dynamics of the rotational motion of the pendulum is in the form

$$ml^2\ddot{\varphi}(t) + \beta\dot{\varphi}(t) + mgl\sin(\varphi(t)) = u(t) \tag{1a}$$

$$ml^2\ddot{\varphi}(t) = -\beta\dot{\varphi}(t) - mgl\sin(\varphi(t)) + u(t) \tag{1b}$$

where $u(t)$ [kg m^2 s^{-2}] is the external torque acting on the pendulum arm, $\dot{\varphi}(t)$ [rad s^{-1}] is the angular velocity, and $\ddot{\varphi}(t)$ [rad s^{-2}] is the angular acceleration of the pendulum arm. The numerical values of the pendulum parameters are given in Table 1.

Table 1: Pendulum Parameters

Parameter	Value	Units
m	1	kg
l	1	m
g	9.81	m s^{-2}
β	$2 \cdot 0.5 \cdot \sqrt{g/l}$	kg m^2 s^{-1}

1.1 Dynamic System Model

Equation (1) is the model of the considered dynamic system. A model is a mathematical representation of a physical system in this case. The model allows us to consider the system and predict how it will behave.

1.1.1 Input-Output Model of the System

The given model describes the input-output behavior of the dynamic system, where the input is the external torque $u(t)$ and the output is the angle $\varphi(t)$. However, we will also work with the state-space description of the system.

1.1.2 State-Space Description of the System

The state of a system is a set of variables (quantities) that summarize the past of the system for the purpose of predicting the future of the system. For a physical system, the state consists of variables needed to compute changes in mass, momentum, and energy. A key question in modeling is how accurately this change should be described.

State variables form a vector $x(t) \in \mathbb{R}^n$, called the *state vector*. Inputs, through which the system is controlled, form an input vector $u(t) \in \mathbb{R}^p$, and measurable outputs of the system form an output vector $y(t) \in \mathbb{R}^q$. In this case, we have $p = q = 1$. The dynamic system can then be represented by equations in the form

$$\dot{x}(t) = f(x(t), u(t)) \tag{2a}$$

$$y(t) = h(x(t), u(t)) \tag{2b}$$

where $f : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^n$ and $h : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^q$ are smooth functions. A model in this form is called a *state-space model*.

The dimension of the state vector is called the *order of the system*. The system (2) is called *time-invariant* because the functions f and h do not directly depend on time t. For time-variant systems, they do. The model consists of two functions: the function f determines the rate of change of the state vector as a function of the state $x(t)$ and the input $u(t)$, and the function h determines the measurable outputs as a function of the state $x(t)$ and the input $u(t)$.

Linear System

A system is called linear if the functions f and h are linear with respect to x and u. A linear state-space model has the form

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{3a}$$

$$y(t) = Cx(t) + Du(t) \tag{3b}$$

where A, B, C, and D are constant matrices. Such a system is called linear and time-invariant, abbreviated as LTI. The matrix A is called the system matrix, the matrix B is called the input matrix, the matrix C is called the output matrix, and the matrix D is called the direct transmission matrix. The vast majority of systems do not have a direct transmission term, which means that the input does not directly affect the output.

A linear dynamic system can be written in the form of a linear ordinary differential equation

$$\frac{\mathrm{d}^n y(t)}{\mathrm{d}t^n} + a_{n-1}\frac{\mathrm{d}^{(n-1)} y(t)}{\mathrm{d}t^{(n-1)}} + \cdots + a_0 y(t) = b_0 u(t) \tag{4}$$

where t is the independent variable (time), $y(t)$ is the dependent variable (output), and $u(t)$ is the input. The notation $\frac{\mathrm{d}^n y}{\mathrm{d}t^n}$ denotes the n-th derivative of $y(t)$ with respect to time t. We say that equation (4) is an n-th order differential equation that models the dynamics of an n-th order system. Conversion to a state-space model is, for example, by defining the state vector in the form

$$x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y(t) \\ \frac{\mathrm{d}y(t)}{\mathrm{d}t} \\ \vdots \\ \frac{\mathrm{d}^{(n-1)}y(t)}{\mathrm{d}t^{(n-1)}} \end{bmatrix} \tag{5}$$

then the state-space model can be written in the form

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{bmatrix} = \begin{bmatrix} x_{n-1}(t) \\ x_{n-2}(t) \\ \vdots \\ -a_{n-1}x_n(t) - \cdots - a_0 x_1(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ b_0 u(t) \end{bmatrix} \tag{6a}$$

$$y(t) = x_1(t) \tag{6b}$$

which, after appropriate definition of the matrices A, B, C, and D, has the form (3). For example, it is possible that the output will be a linear combination of all state variables (we assume that the output does not directly depend on the input), thus

$$y(t) = c_1 x_1(t) + c_2 x_2(t) + \cdots + c_n x_n(t) \tag{7}$$

Then the state-space model is

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ -a_0 & -a_1 & -a_2 & \cdots & -a_{n-2} & -a_{n-1} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_{n-2}(t) \\ x_{n-1}(t) \\ x_n(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ b_0 \end{bmatrix} u(t) \tag{8a}$$

$$y(t) = \begin{bmatrix} c_1 & c_2 & \dots & c_{n-2} & c_{n-1} & c_n \end{bmatrix} x(t) \tag{8b}$$

Nonlinear System

We have an equation describing the dynamics of the rotational motion of the pendulum. The equation is in the form

$$ml^2\ddot{\varphi}(t) + \beta\dot{\varphi}(t) + mgl\sin(\varphi(t)) = u(t) \tag{9}$$

This equation (equation (**??**)) can also be rewritten in the form

$$\ddot{\varphi}(t) = -\frac{\beta}{ml^2}\dot{\varphi}(t) - \frac{g}{l}\sin(\varphi(t)) + \frac{1}{ml^2}u(t) \tag{10}$$

which in this case is just a cosmetic adjustment, but relatively useful, as will be shown.

Any higher-order differential equation can be written as a system of first-order equations. For example, a system of two first-order equations can generally be written in the form

$$\begin{bmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\end{bmatrix} = F\left(t, \begin{bmatrix}x_1(t)\\ x_2(t)\end{bmatrix}, \dots\right) \tag{11}$$

In such a new system of equations, new quantities (signals) generally arise, which may differ from the original quantities (signals) in the original higher-order equation.

The new quantities appearing in the system of equations are collectively referred to as the state of the system (state variables of the system) in system theory. If we know the current state of the system, we can generally determine the past and future states.

For example, in the pendulum equation, the quantities (signals) $\ddot{\varphi}(t)$, $\dot{\varphi}(t)$, and $\varphi(t)$ appear. It is clear (though perhaps not immediately obvious) that the state of the system can be chosen as the quantities $\varphi(t)$ and $\dot{\varphi}(t)$, i.e., the position and angular velocity of the pendulum. If we know these, we know the entire history and future of the pendulum's motion.

There may be multiple choices for state variables. For linear systems, there are infinitely many possibilities (infinitely many state spaces). However, from a practical point of view, only some choices are meaningful - for example, for motion systems like the pendulum, these are naturally position, velocity, acceleration, jerk, etc., depending on the order of the system.

One way to convert a higher-order equation to a system of first-order equations is the following procedure. In this case, coincidentally, the result is also a practically usable state space (state variables $\varphi(t)$ and $\dot{\varphi}(t)$). Let

$$x_1(t) = \varphi(t) \tag{12}$$

then

$$\dot{x}_1(t) = \dot{\varphi}(t) \tag{13}$$

Next, let

$$\dot{x}_1(t) = \dot{\varphi}(t) = x_2(t) \tag{14}$$

and that means

$$\dot{x}_2(t) = \ddot{\varphi}(t) \tag{15}$$

Thus, we have obtained the quantities $x_1(t) = \varphi(t)$ and $x_2(t) = \dot{\varphi}(t)$. It is possible to form the state vector $x = \begin{bmatrix}x_1(t) & x_2(t)\end{bmatrix}^\mathsf{T}$ and thus $\dot{x} = \begin{bmatrix}\dot{x}_1(t) & \dot{x}_2(t)\end{bmatrix}^\mathsf{T}$.

As we indicated in connection with (11), the goal is actually to specify the function F in the equation

$$\dot{x} = F(t, x, \dots) \tag{16}$$

which is a compact notation of the system

$$\dot{x}_1(t) = F_1\left(t, x_1(t), x_2(t), \dots\right) \tag{17}$$
$$\dot{x}_2(t) = F_2\left(t, x_1(t), x_2(t), \dots\right) \tag{18}$$

The first equation in this case is:

$$\dot{x}_1(t) = x_2(t) \tag{19}$$

The second equation follows from the observation that the original second-order equation can be written in the form

$$\begin{aligned}\ddot{\varphi}(t) &= -\frac{\beta}{ml^2}\dot{\varphi}(t) - \frac{g}{l}\sin\left(\varphi(t)\right) + \frac{1}{ml^2}u(t)\\ &= -\frac{\beta}{ml^2}x_2(t) - \frac{g}{l}\sin\left(x_1(t)\right) + \frac{1}{ml^2}u(t)\end{aligned} \tag{20}$$

where the newly introduced state variables $x_1(t)$ and $x_2(t)$ are used. It is clear that the second equation of the system is

$$\dot{x}_2(t) = -\frac{\beta}{ml^2}x_2(t) - \frac{g}{l}\sin\left(x_1(t)\right) + \frac{1}{ml^2}u(t) \tag{21}$$

and thus the pendulum equations in the state space are

$$\begin{bmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\end{bmatrix} = \begin{bmatrix} x_2(t) \\ -\frac{\beta}{ml^2}x_2(t) - \frac{g}{l}\sin\left(x_1(t)\right)\end{bmatrix} + \begin{bmatrix}0\\ \frac{1}{ml^2}\end{bmatrix} u(t) \tag{22}$$

thus the function F is clearly specified.

The state of the pendulum consists of two quantities: the angle of rotation of the pendulum arm $\varphi(t)$ and the angular velocity of the pendulum arm $\dot{\varphi}(t)$. The state vector therefore has two elements $x^{\mathsf{T}}(t) = \begin{bmatrix}x_1(t) & x_2(t)\end{bmatrix}$, where $x_1(t) = \varphi(t)$ and $x_2(t) = \dot{\varphi}(t)$. The pendulum model in the state space is in the form

$$\begin{bmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\end{bmatrix} = \begin{bmatrix} x_2(t) \\ -\frac{\beta}{ml^2}x_2(t) - \frac{g}{l}\sin(x_1(t))\end{bmatrix} + \begin{bmatrix}0\\ \frac{1}{ml^2}\end{bmatrix} u(t) \tag{23a}$$

$$\varphi(t) = x_1(t) \tag{23b}$$

This is a nonlinear time-invariant second-order system.

2 ODE solver

To find the numerical solution, we will use an ODE solver. ODE stands for ordinary differential equations.

The task of an ODE solver is to find a numerical solution based on the equation (differential), which can generally be written in the form

$$\dot{x}(t) = f\left(t, x(t), \ldots\right) \tag{24}$$

where f is a function whose arguments are time t, naturally, the output (sought, unknown) signal $x(t)$, and possibly other parameters or quantities - for example, an external input. The given equation literally prescribes what the time change of the signal $x(t)$ is. The time change of the signal, in other words, the time derivative (derivative with respect to time) is denoted as $\dot{x}(t)$.

Numerical Integration

If we substitute the values of the arguments (time, signal $x(t)$, and possibly others) into the function f, we obtain the value of the time change $\dot{x}(t)$. Based on the information about $\dot{x}(t)$, which corresponds to the current (substituted) signal $x(t)$, we can determine the value of $x(t)$ at some later time. This new value of $x(t)$ can again be substituted into the function f and subsequently find another value even further in time - and so on. The ODE solver uses this simple principle to gradually find the values (numerical values) of the signal $x(t)$.

In general, the mentioned principle is called numerical integration. The ODE solver thus numerically integrates. There are many methods for numerical integration, which differ in the way they solve problems related to the process of numerical integration itself (choice (optimization) of the integration time step, consideration of the mathematical properties of the given type of differential equations, and others). ODE solvers can also differ in the implementation of some of the numerical integration methods. A more detailed description of the ODE solver is beyond the scope of this text.

2.1 Using the ODE Solver

An ODE solver as a function in a program can have, for example, the following inputs (arguments) and outputs:

```
x = odesolver(fcnF, init, timeVect)
```

where `x` is, of course, the sought numerical solution. The first argument is a function named `fcnF`, which implements the system of differential equations in the sense of the previous text. `init` denotes the initial values of the state variables. `timeVect` denotes the time moments (samples) at which we are looking for the values of the numerical solution.

2.1.1 MATLAB

MATLAB contains several ODE solvers. Here we will use `ode45`.

Autonomous System (without input signal)

Let's create a function that implements the system of differential equations (22), but consider that the input signal $u(t)$ is zero. That is, do not consider the input signal at all. In other words, the external torque is zero, $u(t) = 0$, and therefore we can write

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -\frac{\beta}{ml^2} x_2 - \frac{g}{l} \sin(x_1) \end{bmatrix} \tag{25}$$

This is an autonomous nonlinear time-invariant second-order system. Its behavior depends only on the initial state at the beginning of the considered time.

The function that implements the given system can be as follows:

Complete file PravaStr.m

```
function dotx = PravaStr(t,x)

global m l g beta

dotx1 =  x(2);
dotx2 = - (beta/m*l^2)*x(2)  -  (g/l)*sin(x(1));

dotx = [dotx1; dotx2];

end
```

Let's create a "main script" in which we set everything necessary and in which we will call the ODE solver. First, let the global variables (in this case, the pendulum parameters) be:

Part of the file hlSkript.m

```
global m l g beta

m = 1; %kg
l = 1; %m
g = 9.81; %m/s^2
beta = 2*0.5*sqrt(g/l); %kgm^2/s
```

Define the time vector, which will determine for which time moments the ODE solver will return the numerical solution:

Part of the file hlSkript.m

```
timeVect = 0:0.1:5;
```

Call the ODE solver, while it remains to choose the initial conditions - the initial state of the pendulum. Let the initial state be $x_1(0) = 0.25$ [rad] and $x_2(0) = 0$ [rad/s].

Part of the file hlSkript.m

```
[t,x] = ode45(@(t,x) PravaStr(t,x), timeVect, [0.25; 0]);
```

The variable `x` now contains two columns - the first column is the first state variable and the second column is the second state variable. To plot the calculated solution:

Part of the file hlSkript.m

```
figure(1)
plot(t,x)
```

The resulting numerical solution is graphically shown in Fig. 2.

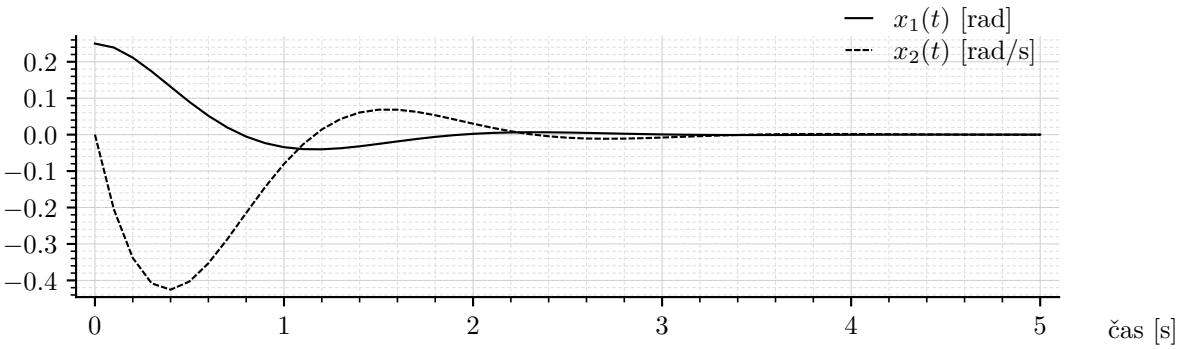

Figure 2: Graphical representation of the numerical solution.

In Fig. 2, however, it is just a basic display. It might be more meaningful, for example, if we plotted only the pendulum position (deflection) separately and additionally not in radians but in degrees – see Fig. 3. For such a figure, you can add to the main script:

Part of the file hlSkript.m

```
11      figure(2)
12      plot(t,x(:,1)*180/pi)
```

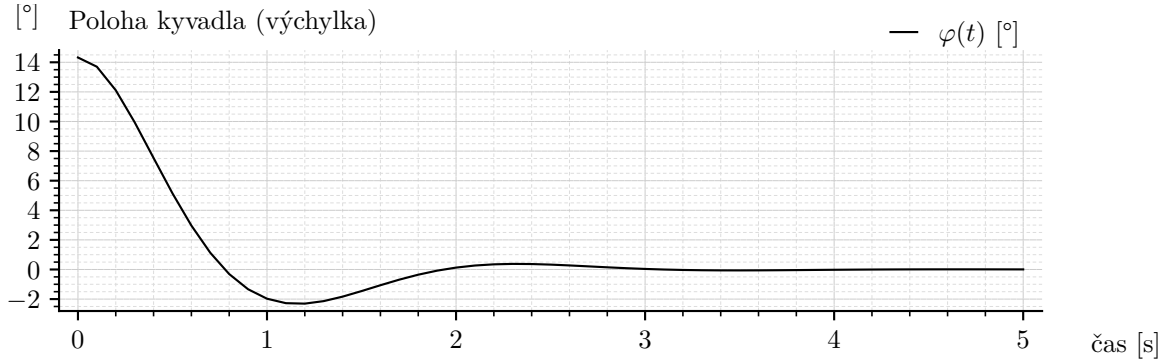

Figure 3: Graphical representation of the pendulum position.

System with Input Signal

Let's modify the original function and script in MATLAB to simulate a non-zero input signal $u(t)$.

The function that implements the system of differential equations (22) with the input signal $u(t)$:

Complete file PravaStr_u.m

```
1   function dotx = PravaStr_u(t,x, u)
2
3   global m l g beta
4
5   dotx1 =  x(2);
6   dotx2 = - (beta/m*l^2)*x(2) - (g/l)*sin(x(1)) + (1/m*l^2) * u;
7
8   dotx = [dotx1; dotx2];
9
10  end
```

Let's create a "main script" where we set everything necessary and call the ODE solver:

File hlSkript_u.m

```
1   global m l g beta
2
3   m = 1; %kg
4   l = 1; %m
5   g = 9.81; %m/s^2
```

```
beta = 2*0.5*sqrt(g/l); %kgm^2/s

u = 3

[t,x] = ode45(@(t,x) PravaStr_u(t,x,u), [0 10], [0; 0]);

figure(3)
plot(t,x(:,1)*180/pi)
```

Let's simulate the case when, for example, $u(t) = 3$ [kg m^2 s^{-2}] (note: for better clarity, let's consider zero initial conditions). The simulation result is shown in Fig. 4.

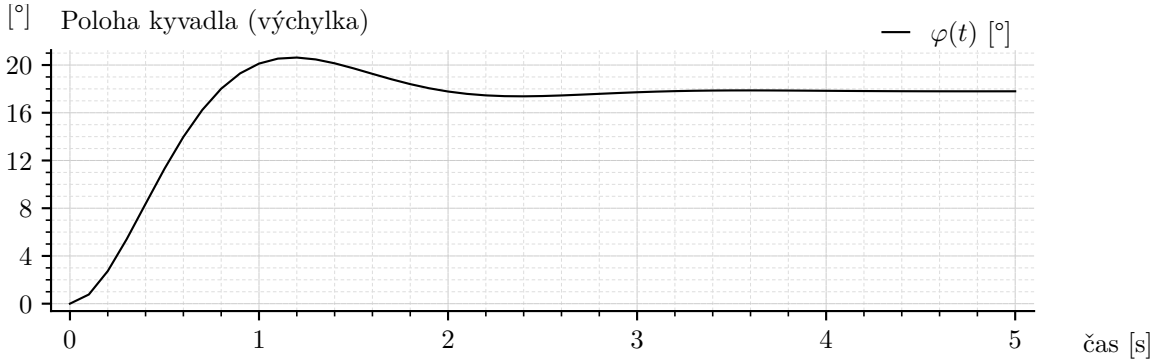

Figure 4: Graphical representation of the pendulum position.

2.1.2 Python

The following would be the search for a numerical solution in the Python language.

The library SciPy, specifically scipy.integrate contains an ODE solver called `odeint`. Let's create a script using this ODE solver:

Python Script

```
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt

m = 1.0
l = 1.0
g = 9.81
beta = 2 * 0.5 * np.sqrt(g/l)

def fcn_rovniceKyvadla(x, t, u):
    x_1, x_2 = x
    dotx_1 = x_2
    dotx_2 = -(beta/m*l**2) * x_2 - (g/l) * np.sin(x_1) + (1.0/m*l
    **2) * u
    return [dotx_1, dotx_2]


timeVect = np.arange(0, 5.1, 0.1)

u = 0

x = odeint(fcn_rovniceKyvadla,
           [np.pi/4, 0],   # initial conditions
           timeVect,
           args=(u,),
           )

plt.figure(1)
plt.plot(timeVect, x)
plt.xlabel(u'time [s]')
plt.legend(['$x_1(t)$ [rad]', '$x_2(t)$ [rad/s]'])

plt.figure(2)
plt.plot(timeVect, x[:,0]*180/np.pi)
plt.xlabel(u'time [s]')
plt.ylabel(u'$x_1(t)$ [degrees]')
```

3 Questions and Tasks

1. Explain the concept of *numerical solution* of an ordinary differential equation.
2. What is the difference between an analytical and a numerical solution of a differential equation?
3. Rewrite the system of equations

$$\begin{aligned}\dot{x}_1(t) &= x_2(t)\\ \dot{x}_2(t) &= -a_0 x_1(t) - a_1 x_2(t) + b_0 u(t)\\ y(t) &= x_1(t)\end{aligned}$$

   into matrix form:

$$\begin{aligned}\dot{x}(t) &= Ax(t) + bu(t)\\ y(t) &= c^{\mathsf{T}} x(t)\end{aligned}$$

   (define the signal vector $x(t)$, matrix A, and vectors b and c).