ELEC3035 mindstorms

Ivan Markovsky

School of Electronics and Computer Science
University of Southampton
im@ecs.soton.ac.uk

Abstract

The exercises described in this document are inspired by Seymour Papert's book "Mindstorms: Children, Computers, And Powerful Ideas". Their purpose is to stimulate curiosity in systems and control related concepts via independent work on computer based projects. The projects cut across several disciplines—physics, mathematics, computer programming, as well as the subject of ELEC3035—systems and control theory.

Mindstorms (optional background information)

> What kind of curve is the trajectory of a stone thrown in the air?
> How should one throw a stone in order to reach as far as possible?

These are curiosity driven questions, which can be answered by first turning them into well defined mathematical problems (using high school physics) and then solving the mathematical problems (using analytical and numerical methods, studied in school as well as in the university). Solving mathematical problems is part of the ELEC3035 exercises, but it is not the main part and not the most important part either. More important than finding answers to math problems is the habit to ask questions (*i.e.*, aiming for open enquiring rather than following of authority) and the ability to turn vague questions into well defined problems (there is no recipe for this; it is more of an art than science). Only in combination with these skills, the ability to reason rigorously in the search for solutions becomes a powerful tool for solving real-life problems.

The philosophy of "mindstorms", described by Seymour Papert in his books [1, 2, 3], is that

> the only way to acquire any type of knowledge is to immerse in an environment that offers ample possibilities to practice this knowledge.

Using Papert's analogy, in the same way as you need France for learning french, you need "control land" for learning control theory. These exercises aim to serve the role of "control land" for ELEC30335. I like the exercise and hope you do too. There are however many possibilities for improvement. Your feedback (criticism, corrections of mistakes, and ideas for additional exercises) is a critical element in this process.

One needs motivation in order to pursue the search for answers to difficult questions. Such motivation comes only through personal interest and involvement with the questions. Unfortunately, the formal educational system reverses this causal relation—everyone is expected to acquire certain "pure" knowledge (the curriculum of one's education) in order to be prepared for solving "real-life" problems (or just for beginning an educated person). The "real-life" problems are usually not encountered until the education is over.

In ELEC3035, you are given a possibility to practice your systems and control knowledge by working on a series of exercises which are formulated as vague questions rather than as well defined problems. These questions are curiosity driven for me personally and I hope you too will be motivated to find answers. You have full freedom to choose the problem formulation and solution method that you deem relevant for the question at hand. Moreover, you are encouraged to find your own style in dealing with the assignments and ask questions that are not part of the assignments!

Given such freedom, you may come up with different answers to the original questions (as well as answers to questions that were never asked). As long as your answers are correct solutions to relevant problems, related to the original question, they are "valid answers". Clearly, you may come up with different valid answers. This is fine. In fact, this is what happens in real-life engineering practice.

Introduction and notation

In the exercises you will be dealing with questions related to free and controlled flight of an object in a gravitational field. For example, throwing a stone, we apply for a (short) period of time a force on the stone, which net effect is to give the stone an initial velocity. The magnitude and direction of the initial velocity is all that matters. How they are achieved is not important for the subsequent flight. From the moment of departing from the hand, the stone is falling freely; the forces that act on the stone are the gravitational force and a force from the impact with the air (friction and wind). To begin with, assume that the stone is flying in vacuum, so that the the only force acting on the stone is the gravitational force.

Note that the stone remains throughout its free falling flight in the plane determined by the initial velocity and the gravitational force. Therefore, although the stone is flying in a three dimensional space, it actually remains in a plane. Let $p(t)$ be the position of the stone at time t. We choose a reference moment of time $t = 0$ to be the moment when the stone starts its free fall and an orthogonal coordinate system in the plane of motion with vertical axis along the negative of the gravitational force and a perpendicular horizontal axis at the ground level, see Figure 1.

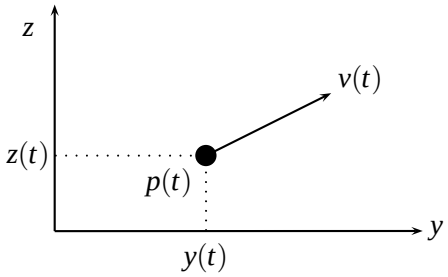

Figure 1: Problem setup.

The horizontal displacement of the stone at time t with respect to the origin is denoted by $y(t)$ and the vertical displacement by $z(t)$. More notation, used later on, is:

$p(t) = \begin{bmatrix} y(t) \\ z(t) \end{bmatrix}$	—	object's position and its coordinates at time t
$v(t)$	—	object's velocity at time t
p_{ini}, v_{ini}	—	initial (time $t = 0$) position and velocity
$x(t)$	—	state (position and velocity) at time t
m	—	object's mass
g	—	gravitational constant
$m\mathbf{g}$	—	gravitational force ($\mathbf{g}$ is a vector in the "negative vertical" direction with norm equal to g)

By the second law of Newton, for any $t > 0$, the position of the stone is described by the differential equation

$$m\ddot{p} = m\mathbf{g}, \qquad \text{where} \quad p(0) = p_{\text{ini}} \text{ and } \dot{p}(0) = v_{\text{ini}}. \tag{1}$$

Here p_{ini} is the initial position (the place from where the stone is thrown) and v_{ini} is the initial velocity (by which the stone is thrown). Note the following facts.

- (1) is a linear differential equation with constant coefficients. (This is good news because such equations can be solved explicitly.)
- (1) is of second order and is a vector equation (there is a scalar equation for y and another scalar equation for z).
- The equation does not depend on the mass m (since m cancels). This means that any stone, more generally any object, falling in a gravitational field without friction has the same trajectory.[1]

[1] The fact that the trajectory does not depend on the mass may be surprising and counter intuitive. Here is an instance when physics and mathematics reveal something that is not obvious.

Since it is no longer relevant that the thrown object is a stone, from now on, we will call it more generally "an object".

- The right hand side is constant. This further simplifies the solution of the equation.

Equation (1) "says" everything about the motion of the object. It is a concise and unambiguous description of any object falling in a gravitational field, starting from an arbitrary initial conditions—position and velocity. The equation, however, is not as clear as an explicit solution that exhibits the nature of the trajectory (p as a function of time). Therefore, your first task is to find the solution explicitly; both analytically (by hand) and numerically (using the computer). If you did not know the answer of the opening question "What kind of curve is the trajectory of a stone thrown in the air?", then you will discover it yourself, which is the best way to learn something.

Exercise 1: Trajectory of a freely falling object

In this exercise, we consider the trajectory of an object with initial condition $x(0) = x_{\text{ini}}$ (position and velocity) when no external force f is applied (*i.e.*, a freely falling object, thrown from some location $p(0) = p_{\text{ini}}$ with some initial velocity $v(0) = v_{\text{ini}}$). Using any method you deem appropriate, find an analytic expression for the resulting trajectory. Then, write a function (in your favourite programming language) that takes as an input the initial condition and returns as an output samples of the trajectory and a time vector of the corresponding moments of time. Test your function for some initial conditions and plot the resulting trajectories.

Solution: The second order vector differential equation (1) can be written as a first order equation

$$\dot{x} = Ax, \qquad p = Cx, \qquad x(0) = x_{\text{ini}}, \tag{2}$$

where

$$x := \begin{bmatrix} y \\ \dot{y} \\ z \\ \dot{z} \\ x_5 \end{bmatrix}, \qquad x_{\text{ini}} := \begin{bmatrix} y_{\text{ini}} \\ v_{\text{ini},1} \\ z_{\text{ini}} \\ v_{\text{ini},2} \\ -g \end{bmatrix}, \qquad A := \begin{bmatrix} 0 & 1 & & & \\ 0 & 0 & & & \\ & & 0 & 1 & \\ & & 0 & 0 & 1 \\ & & & & 0 \end{bmatrix} \qquad \text{and} \qquad C := \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}. \tag{3}$$

We have taken into account the constant input **g** in (1) by defining an extra state variable x_5, with equation $\dot{x}_5(t) = 0$ and initial condition $x_5(0) = -g$. In system theoretic terms, this turns the inhomogeneous equation (1) into an autonomous state space model.

The following code chunk defines the A and C matrices, which specify the state space model (2) (up to unknown initial conditions).

3 ⟨*State-space model* 3⟩≡ (4e 7a 9 12–15)

```
a1 = [0 1; 0 0]; a = blkdiag(a1, a1, 0); a(4, 5) = 1;
c  = zeros(2, 5); c(1, 1) = 1; c(2, 3) = 1;
```

Using the material in Lecture 3, we know that the trajectory p of (2) is

$$p(t) = Ce^{At}x_{\text{ini}}. \tag{4}$$

This is an explicit formula (a closed form solution) for the trajectory p resulting from the initial condition x_{ini}, however, for the particular model, given by (3), it can be further simplified.

The matrix exponential is by definition the series

$$e^{At} = \sum_{i=0}^{\infty} \frac{A^i t^i}{i!}.$$

Note that

$$A^2 = \begin{bmatrix} 0 & 1 & & & \\ 0 & 0 & & & \\ & & 0 & 1 & \\ & & 0 & 0 & 1 \\ & & & & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & & & \\ 0 & 0 & & & \\ & & 0 & 1 & \\ & & 0 & 0 & 1 \\ & & & & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & & & \\ 0 & 0 & & & \\ & & 0 & 0 & 1 \\ & & 0 & 0 & 0 \\ & & & & 0 \end{bmatrix}$$

and $A^i = 0$, for all i greater than two. Therefore,

$$e^{At} = \begin{bmatrix} 1 & t & & & \\ 0 & 1 & & & \\ & & 1 & t & \frac{1}{2}t^2 \\ & & 0 & 1 & t \\ & & & & 1 \end{bmatrix}. \tag{5}$$

Note that the dynamics along the y and z axis is independent (decoupled). Using the explicit formula for the matrix exponential, we obtain explicit formulae for the solution (4):

$$y(t) = v_{\text{ini},1}t + y_{\text{ini}} \qquad \text{and} \qquad z(t) = \big(v_{\text{ini},2} - \frac{1}{2}gt\big)t + z_{\text{ini}}. \tag{6}$$

which shows that along y, the motion is linear with the initial speed and along z, the motion is quadratic.

In the numerical implementation, we use (4) rather than the explicit formulae (6), because it is applicable for any linear time-invariant model. In order to do the computation, we discretize the continuous-time equation. If the discretization time is t_s,

4a ⟨*Define* `ts` 4a⟩≡ (4b)
```
ts = 0.01;
```

the corresponding discrete-time equation is

$$p(t_s k) = CA_{\text{d}}^k x_{\text{ini}}, \qquad \text{where} \quad A_{\text{d}} := e^{At_s}. \tag{7}$$

4b ⟨*Discretization* 4b⟩≡ (4e 6a 12–15) 12b▷
```
⟨Define ts 4a⟩
ad = expm(a * ts);
```

The simulation is continued till the object falls on the ground, *i.e.*, $z(t) > 0$,

4c ⟨*Simulation* 4c⟩≡ (4e 6a)
```
⟨Define g 4d⟩
x = [xini; -g];
while x(3,end) >= -eps
    x = [x ad * x(:,end)];
end
p = c * x;
T = size(x, 2); t = 0:ts:(T - 1) * ts;
```

where the gravitational constant g is defined as follows

4d ⟨*Define* `g` 4d⟩≡ (4 7 9 12d 15)
```
g = 9.81;
```

Putting everything together, we have the following function for simulation of a freely falling object with a given initial condition.

4e ⟨`sim_ini` 4e⟩≡
```
function [p, t, x] = sim_ini(xini)
⟨State-space model 3⟩
⟨Discretization 4b⟩
⟨Simulation 4c⟩
```

The function is tested with the following script

4f ⟨`test_sim_ini` 4f⟩≡ 4g▷
```
xini = [0 1 0 2]';
[p, t] = sim_ini(xini); plot(p(1,:), p(2,:)), hold on
```

comparing the numerical solution with the solution obtained by the analytic expression (6)

4g ⟨`test_sim_ini` 4f⟩+≡ ◁4f 6b▷
```
⟨Define g 4d⟩
plot(xini(1) + xini(2) * t, xini(3) + xini(4) * t - 0.5 * g * t .^2, '-r')
print_figure(1)
```

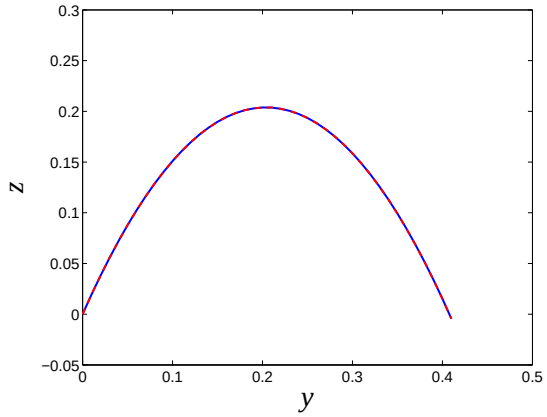

Figure 2: Example of a simulated trajectory with initial condition $x_{\text{ini}} = (0,1,0,2)$. (Solid blue line — numerical solution, Dashed red line — analytic solution.)

The test script produces the plot shown in Figure 2. □

Exercise 2: Free fall with friction

In this exercise, we relax the assumption that the object is in vacuum. The net effect of the air on the object is a force f, acting on the object. The model equation (1) becomes

$$m\ddot{p} = m\mathbf{g} + f, \qquad \text{where} \quad p(0) = p_{\text{ini}} \text{ and } \dot{p}(0) = v_{\text{ini}}. \tag{8}$$

Without wind and turbulence, the force f is due to friction with the air and can be approximated by a linear function of the velocity, *i.e.*, we take

$$f = -\gamma v, \tag{9}$$

where γ is a constant depending on the physical properties of the environment as well as the size and shape of the object. Repeat exercise 1 for the case when a friction force (9) is present. Experiment with different values for the mass m and the friction constant γ.

Solution: The modified equations (8) and (9) result in a modified autonomous state space model (2). The only difference between the model in Exercise (1) and the model in this exercise is the A matrix

$$A := \begin{bmatrix} 0 & 1 & & & \\ 0 & -\gamma/m & & & \\ & & 0 & 1 & \\ & & 0 & -\gamma/m & 1 \\ & & & & 0 \end{bmatrix}. \tag{10}$$

5 ⟨*State-space model with friction* 5⟩≡ (6a 7c)

```
a1 = [0 1; 0 -gamma/m]; a = blkdiag(a1, a1, 0); a(4, 5) = 1;
c  = zeros(2, 5); c(1, 1) = 1; c(2, 3) = 1;
```

Note that now the dynamics depends on the mass of the object, which is consistent with our experience. It is still possible but more involved to find the matrix exponential and the trajectory p in closed form. (Derivation of the closed form expression is to be added.) The trajectory turns out to be qualitatively different — the horizontal motion is a decaying exponential (converging to a constant) and the vertical motion is a decaying exponential, converging to a linear function.

Contrary to the analytical approach that gets more complicated, the numerical approach needs only trivial modifications. The only difference in the simulation function of the model with friction and the simulation function

`sim_ini`, defined in Exercise 1, is that the state space model is different:

6a ⟨`sim_ini_friction` 6a⟩≡
```
function [p, t, x] = sim_ini(xini, m, gamma)
⟨State-space model with friction 5⟩
⟨Discretization 4b⟩
⟨Simulation 4c⟩
```

The function is tested by comparing the trajectory with and without fiction

6b ⟨`test_sim_ini` 4f⟩+≡ ◁4g
```
[p, t] = sim_ini_friction(xini, 1, 1); plot(p(1,:), p(2,:))
print_figure(6)
```

The result is shown in Figure 3. The free fall with friction indeed does not reach as far as the corresponding one without friction and has the expected ballistic shape and rather than the symmetric shape of a parabola. □

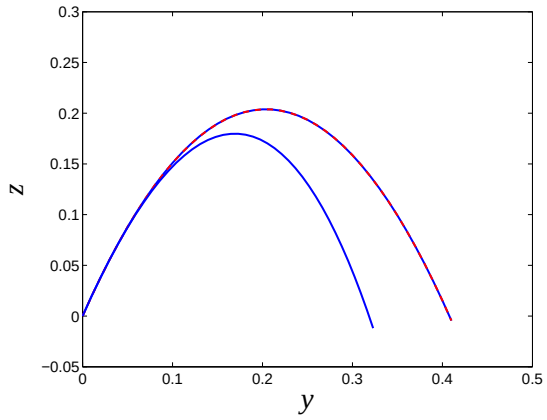

Figure 3: Example of a simulated trajectories with (solid blue) and without (dashed red) friction. The initial condition is $x_{\text{ini}} = (0, 1, 0, 2)$, the mass is $m = 1$, and the friction coefficient is $\gamma = 1$.

Exercise 3: Bungee jumping

The object now is attached to one end of an elastic rope. The other end of the rope is fixed at a given location p_{r}, see Figure 4. The force exerted on the object from the rope is

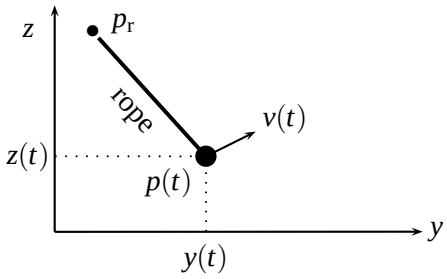

Figure 4: Setup for Exercise 3.

$$e(t) = \gamma\big(p_{\text{r}} - p(t)\big), \tag{11}$$

where γ is a positive constant. Find the trajectory resulting from an initial condition x_{ini}. Compare the trajectories with and without friction in the air.

Solution: To be added. □

In order to make the setup more realistic, modify the elastic force (11) taking into account the rope's length r, *i.e.*, the force is zero before the rope gets stretched. Can you still find the trajectory p in closed form? Revise the numerical simulation function for this case and observe how the trajectory differs from before.

Solution: To be added. □

Exercise 4: Hitting a stationary object

Knowing what the trajectory of a freely falling object is and being able to simulate it on a computer, our next objective is to choose the initial velocity v_{ini}, so that the object starting from initial position $p_{\text{ini}} = 0$ reaches a given position p_{des} at a given moment of time $t = T$, *i.e.*, $p(T) = p_{\text{des}}$. Solve the problem analytically in the case of no friction. Then write a function that takes as an input p_{des} and T and returns as an output the initial velocity v_{ini} that achieves a trajectory, such that $p(T) = p_{\text{des}}$. Test your function for some p_{des} and T.

Solution: The mapping from an initial state

$$x(0) = x_{\text{ini}} = \begin{bmatrix} y_{\text{ini}} & v_{\text{ini},1} & z_{\text{ini}} & v_{\text{ini},2} & -g \end{bmatrix}^\top$$

to the output $p(T) = p_{\text{des}}$ of the autonomous model (2) is given by

$$p_{\text{des}} = \Phi x_{\text{ini}}, \qquad \text{where} \quad \Phi := Ce^{AT}.$$

This is a linear system of two equations with two unknowns $v_{\text{ini},1}$ and $v_{\text{ini},2}$. Therefore, provided the submatrix of Φ obtained by selecting the 2nd and the 4th columns is nonsingular, there is a unique solution. This solution give us the unique initial velocity that achieves the desired trajectory. Using (5), we actually have an explicit expression for Φ

$$\Phi = \begin{bmatrix} t & \\ & t \end{bmatrix}$$

which shows that for ant t greater than zero, Φ is nonsingular.

The above solution is implemented in the following Matlab function:

7a ⟨reach 7a⟩≡
```
function vini = reach(pdes, T)
⟨State-space model 3⟩
⟨Define g 4d⟩
phi  = c * expm(a * T);
vini = phi(:, [2, 4]) \ (pdes + g * phi(:, 5));
```

The function is tested with the following script

7b ⟨test_reach 7b⟩≡ 7d▷
```
xini = zeros(4, 1); xini([2, 4]) = reach([1; 0], 1);
[p, t] = sim_ini(xini); plot(p(1,:), p(2,:)), print_figure(2)
```

which produces the plot shown in Figure 5, left. □

Modify your solution for the case when there is friction in the air.

Solution: The only modification needed in the `reach` function is to replace the state space model:

7c ⟨reach_friction 7c⟩≡
```
function vini = reach_friction(pdes, T, m, gamma)
⟨State-space model with friction 5⟩
⟨Define g 4d⟩
phi  = c * expm(a * T);
vini = phi(:, [2, 4]) \ (pdes + g * phi(:, 5));
```

The function is tested with the following script

7d ⟨test_reach 7b⟩+≡ ◁7b
```
xini = zeros(4, 1); xini([2, 4]) = reach_friction([1; 0], 1, 1, 1);
[p, t] = sim_ini_friction(xini, 1, 1); plot(p(1,:), p(2,:)), print_figure(7)
```

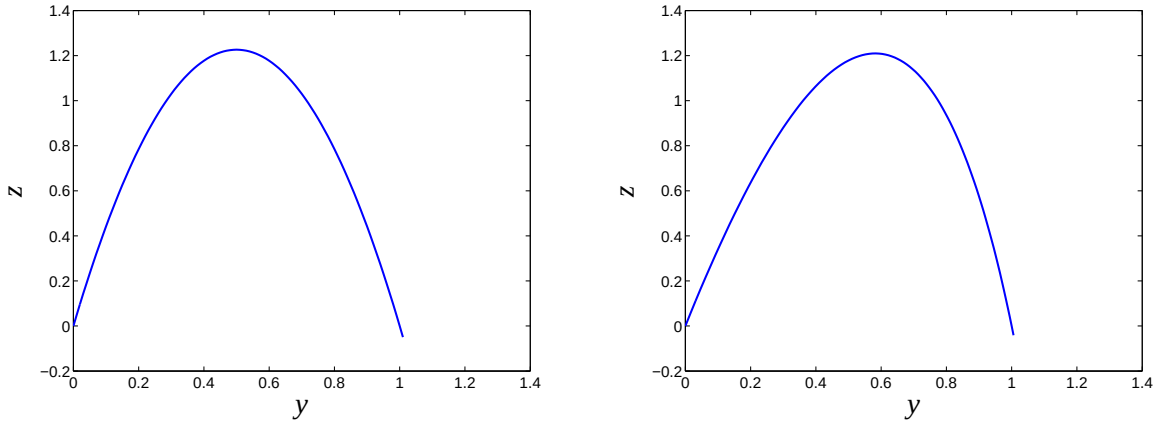

Figure 5: Example of hitting a stationary object. Left plot: without friction. Right plot: with friction.

which produces the plot shown in Figure 5, right. □

Exercise 5: Throwing an object as far as possible

How far can you throw an object? In order to formulate the question mathematically, assume (which is a reasonable assumption) that you can give a maximal initial velocity to the object (by accelerating it for a period of time, after which period the object is freely falling). The question then is what direction should the initial velocity have in order for the object to reach as far as possible when it lands on the ground. The problem is considerably simpler when $z_{\text{ini}} = 0$, *i.e.*, the object is thrown from the ground and there is no friction. Assume that this is the case. Once you come up with an answer, use your free falling simulation function to compute and plot the trajectory. Try some alternative admissible trajectories to make sure that your solution gives the best result.

Solution: Without loss of generality, we can assume that the initial velocity has unit magnitude, *i.e.*,

$$v_{\text{ini}} = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}, \tag{12}$$

where θ is the to-be-determined parameter. The problem is

$$\text{minimize} \quad (\text{over } \theta \in (-\pi, \pi]) \quad |y(T)| \quad \text{subject to} \quad z(T) = 0$$
$$\text{and } p = (y,z) \text{ satisfies (1) with } p_{\text{ini}} = 0 \text{ and } v_{\text{ini}} = \left[\begin{smallmatrix} \cos\theta \\ \sin\theta \end{smallmatrix}\right].$$

From (6) and (12), we have

$$z(T) = 0 \quad \Longleftrightarrow \quad v_{\text{ini},2} - \frac{1}{2} g T = 0 \quad \Longleftrightarrow \quad T = \frac{2}{g} \cos\theta$$

and

$$|y(T)| = |v_{\text{ini},1} T| = \frac{2}{g} |\sin\theta \cos\theta| = \frac{4}{g} |\sin 2\theta|.$$

Therefore, the maximum is achieved for $\sin 2\theta = \pm 1$ or $\theta_1 = \pi/4$ and $\theta_2 = 3\pi/4$.

The test script

8 ⟨test_opt_sol 8⟩≡

```
xini = zeros(4, 1);
th = pi/4; xini([2, 4]) = [cos(th); sin(th)];
[p, t] = sim_ini(xini); plot(p(1,:), p(2,:), '-b'), hold on
th = pi/3; xini([2, 4]) = [cos(th); sin(th)];
[p, t] = sim_ini(xini); plot(p(1,:), p(2,:), '-r')
th = pi/5; xini([2, 4]) = [cos(th); sin(th)];
[p, t] = sim_ini(xini); plot(p(1,:), p(2,:), '-r')
print_figure(3)
```

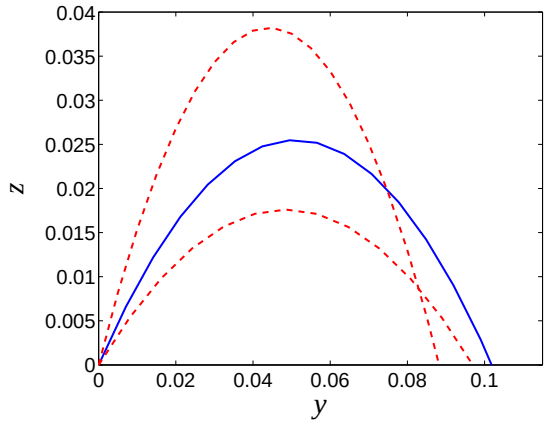

Figure 6: Optimal (solid blue) and suboptimal (dashed red) throws.

produces the plots shown in Figure 6. □

Exercise 6: Deducing the initial state from observed trajectory

Suppose you observe a free falling object for a period of time. More precisely, you are given the positions at given moments of time

$$p(t_1), p(t_2), \ldots, p(t_N), \qquad \text{where} \quad 0 < t_1 \leq t_2 \leq \cdots \leq t_N. \tag{13}$$

Can you find from this data the initial position $p(0)$ and the initial velocity $v(0)$? If so, write a function that accepts as an input data the positions (13) and returns as an output the initial state $x(0)$. Test your function for some simulated trajectories.

Solution: The basic equation (4), gives us a system of equations for the relation between the observed data and the initial state

$$\begin{bmatrix} p(t_1) \\ p(t_2) \\ \vdots \\ p(t_N) \end{bmatrix} = \underbrace{\begin{bmatrix} Ce^{At_1} \\ Ce^{At_2} \\ \vdots \\ Ce^{At_N} \end{bmatrix}}_{\mathscr{O}} x_{\text{ini}}.$$

A solution for x_{ini} is unique if the matrix $\mathscr{O}$ is full column rank. Under this condition, which depends on both (C,A) and $t_1,\ldots,t_N$, the problem is solvable. Since the pair (A,C) is observable

```
>> rank([c; c * a; c * a^2; c * a^3; c * a^4])

ans =

     5
```

the only condition is that $N > 3$.) The above solution can be improved by taking into account the fact that $x_{\text{ini},5} = -g$ is known. (This implies that $N = 2$ exact observations are enough to deduce the initial state.)

The following function estimates the initial condition from given data.

9 ⟨est_xini 9⟩≡
```
function xini = est_xini(p, t)
⟨State-space model 3⟩
⟨Define g 4d⟩
N = length(t); O = [];
for i = 1:N
    O = [O; c * expm(a * t(i))];
end
xini = O(:, 1:4) \ (p(:) + g * O(:, 5));
```

The function is illustrated with the following script

10a ⟨test_est_xini 10a⟩≡
```
[p, t] = sim_ini([1 2 3 4]'); est_xini(p(:, 1:4), t(1:4))
```
which gives:

```
ans =

    1.0000
    2.0000
    3.0000
    4.0000
```

□

It is unrealistic to assume that the moments of time $t_1,\ldots,t_N$ when the observations are taken are known relative to the moment of time when the target is thrown, but you can assume that the object was thrown from the ground, *i.e.*, $z_{\text{ini}} = 0$. In order to relax the assumptions, redo the problem when $t_1 = 0,\ldots,t_N = t_{\text{obsv}}$ are known relative to the moment of time when the first observation is taken. Can you still find the place and velocity from and by which the object is thrown as well as the moment of time, relative to the first observation, when this happened?

Solution: The solution above can still be used. It finds, however, $x_{\text{ini}} = x(0)$, which is the position and velocity at the time of the first observation. Now the formula (4), can be used to propagate the "initial" state x_{ini} *backwards* in time, *i.e.*, find the state x for negative time. The time when the object was thrown is found from the condition $x_3(t) = z(t) = z_{\text{ini}} = 0$. □

In practice, the observations are noisy and are not generated by a linear time-invariant system of low order. Test your estimation method with data (13) obtained as samples of a true trajectory $\bar{p}$ plus zero mean Gaussian additive noise. The estimated initial position and velocity fix an estimated trajectory $\widehat{p}$. Validate the accuracy of the method by evaluating the relative error function

$$e := \frac{\sqrt{\sum_{i=1}^{N} \|\bar{p}(t_i) - \widehat{p}(t_i)\|_2^2}}{\sqrt{\sum_{i=1}^{N} \|\bar{p}(t_i)\|_2^2}}.$$

Solution: To be added. □

Exercise 7: Hitting a moving object

Using the insights and functions that we obtained from the solution of exercises 2 and 4, we now turn to the seemingly challenging problem of hitting a moving object by an object thrown from initial position $p^1_{\text{ini}} = 0$. (The superscript index 1 shows that this is the "first" object; the one that is thrown. The target object's position, velocity, and state are denoted with superscript 2.) Suppose that we have observed the free fall of the target for a given period of time T_{ini}. The aim is to determine the initial velocity $v^1_{\text{ini}} = v^1(T_{\text{ini}})$, such that $p^1(T) = p^2(T)$ for some given $T > T_{\text{ini}}$, *i.e.*, the thrown object hits the target object. Write a function that implements the solution and test it on some simulated data.

Solution: This problem is direct application of the solutions of exercises 1, 2, and 4.

10b ⟨reach2 10b⟩≡
```
function v = reach2(pini, tini, T)
Tini = tini(end);                          % end of the observation period
[p, t] = sim_ini(est_xini(pini, tini)); % extrapolates pini further in time
v = reach(p(:, find(t == T)), T - Tini);
```

The function `reach2` is tested by the following script

11 ⟨test_reach2 11⟩≡
```
[p2, t2] = sim_ini([1 -2 0 1]');
Tini = t2(4); T = t2(end - 4);
vini1 = reach2(p2(:, 1:4), t2(1:4), T);
[p1, t1] = sim_ini([0; vini1(1); 0; vini1(2)]); t1 = t1 + Tini;
plot(p1(1, find(t1 <= T + eps)), p1(2, find(t1 <= T + eps)), 'r'), hold on
plot(p2(1, find(t2 <= T + eps)), p2(2, find(t2 <= T + eps)), 'b')
print_figure(4)
```

which produces the plot shown in Figure 7.

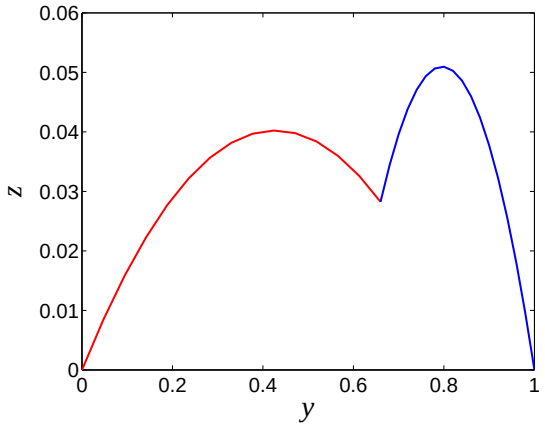

Figure 7: Example for hitting a moving object.

□

Exercise 8: Effect of disturbances

In exercises 1–5, the trajectory of the object was described by equation (1). In this exercise, we simulate a free falling trajectory with a disturbance force

$$m\ddot{p} = m\mathbf{g} + d, \qquad \text{where} \quad p(0) = p_{\text{ini}} \text{ and } \dot{p}(0) = v_{\text{ini}}. \tag{14}$$

Think of the disturbance d as an effect of wind, exerting a force on the object (as a real wind does). The disturbance d is modelled by a zero mean white Gaussian random variable, acting in a horizontal direction.

- The zero mean property implies that for a long period of time the net effect of the disturbance is nil.
- The white noise property implies that the intensities of the wind in sequential moments of time are unrelated.
- The Gaussian property implies that the distribution of the wind intensity in any moment of time is the famous bell shaped curve.

Although these assumptions are somewhat arbitrary and certainly do not match all real wind scenarios, they are a convenient theoretical model to study and capture well some realistic types of disturbances.

Modify the free falling simulation function to correspond to the noisy setup (14). The noise d, described above, is determined by a single parameter—the noise variance v. The parameter v as well as the object's mass m be inputs to the simulation function. Experiment with different values of the parameters and comment on the results. Find the mean and variance of $p(T)$, corresponding to initial condition x_{ini}, *i.e.*, find the effect of the initial condition and disturbance on the landing location.

Solution: The state-space model, corresponding to the model with disturbance is

$$\dot{x} = Ax + Bd, \qquad p = Cx, \qquad \text{where} \quad B = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \end{bmatrix}^\top$$

and A and C are as in (2).

12a ⟨*Disturbance input* 12a⟩≡ (12c 13a)
```
b  = [0; 1; 0; 0; 0];
```

The discretized state-space model with disturbance input is

$$x\big((k+1)t_s\big) = A_d x(kt_s) + B_d d(kt_s), \qquad x(0) = x_{\text{ini}}$$
$$p(kt_s) = Cx(kt_s),$$

where

$$B_d = \int_0^{t_s} e^{A\tau} B \mathrm{d}\tau = A^{-1}(e^{At_s} - I)B,$$

and A_d is in (7). (Work out the details in the derivation of B_d.)

12b ⟨*Discretization* 4b⟩+≡ (4e 6a 12–15) ◁4b
```
if exist('b'), bd = pinv(a) * (expm(a * ts) - eye(size(a, 1))) * b; end
```

The function for noisy simulation has the same structure as `sim_noise`

12c ⟨`sim_ini_noise` 12c⟩≡
```
function [p, t, x] = sim_ini_noise(xini, m, v)
```
⟨*State-space model* 3⟩
⟨*Disturbance input* 12a⟩
⟨*Discretization* 4b⟩
⟨*Noisy simulation* 12d⟩

with a modification of the "simulation" chunk as follows

12d ⟨*Noisy simulation* 12d⟩≡ (12c)
⟨*Define* g 4d⟩
```
x = [xini; -g];
while x(3,end) >= -eps
    x = [x ad * x(:,end) + bd * (sqrt(v) / m) * randn];
end
p = c * x;
T = size(x, 2); t = 0:ts:(T - 1) * ts;
```

The function is tested with the following script

12e ⟨`test_sim_ini_noise` 12e⟩≡
⟨*Simulation parameters* 12f⟩
```
[p, t] = sim_ini_noise(xini, m, v); plot(p(1,:), p(2,:)), print_figure(5)
```

which produces for the simulation parameter

12f ⟨*Simulation parameters* 12f⟩≡ (12 13b)
```
xini = [0 1 0 2]'; m = 1; v = 0.5;
```

the plot shown in Figure 8. As evident from (14), an increase of the noise variance v results in an increase of the randomness in the trajectory p, while an increase of the object's mass decreases the randomness. This is clear from a physical considerations as well: a heavy object is less affected by wind than a light one.

The map from x_{ini} and d to $p(T)$ is linear (because the system is linear) and, by assumption the disturbance is zero mean. These facts imply that the mean of $y(T)$ is equal to the value of $y(T)$ in the noise free simulation. In order to verify this statement empirically, in the following script, we compute the mean of $y(T)$

12g ⟨*Empirical computation of the mean of* $y(T)$ 12g⟩≡
⟨*Simulation parameters* 12f⟩
```
yT = [];
for i = 1:100
  p = sim_ini_noise(xini, m, v);
  yT = [yT p(1, end)];
end
p_e = sim_ini(xini);
[p_e(1, end), mean(yT)]
```

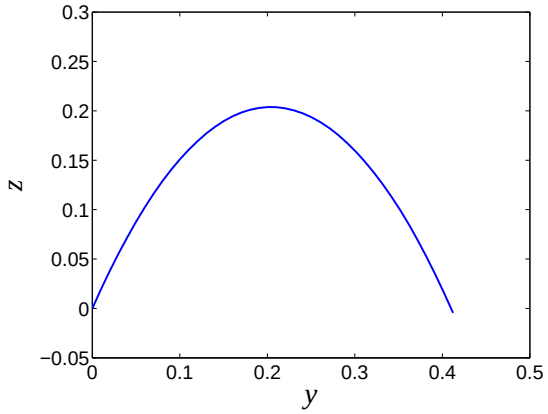

Figure 8: Example of a simulated trajectory with initial condition $x_{\text{ini}} = (0,1,0,2)$, noise variance $v = 0.5$ and mass $m = 1$.

which gives $y(T) = 0.41$ for the exact value and 0.3939 for the mean over 100 noisy simulations. Experiment and observe that the two values get closer when the number of simulations, used for the computation of the mean is increased. Can you explain why this happens?

From Lecture 4, page 5, we have the following representation of the discrete-time system's dynamics

$$p(kt_s) = CA_d x_{\text{ini}} + C \underbrace{\begin{bmatrix} A_d^{k-1}B_d & \cdots & A_dB_d & B_d \end{bmatrix}}_{\mathscr{C}_k} \begin{bmatrix} d(t_s) \\ d(2t_s) \\ \vdots \\ d(kt_s) \end{bmatrix}.$$

Therefore the covariance of $p(kt_s)$ is equal to

$$\operatorname{cov}\left(p(kt_s)\right) = C\mathscr{C}_k V_d \mathscr{C}_k^\top C^\top, \qquad \text{where} \quad V_d = \operatorname{diag}\left(v/m^2, \ldots, v/m^2\right).$$

(Show this.) The computation of the variance of $y(T)$ is implemented in the following function.

13a ⟨var_yt 13a⟩≡
```
function vyt = var_yt(xini, m, v)
p = sim_ini(xini); T = size(p, 2);
⟨State-space model 3⟩
⟨Disturbance input 12a⟩
⟨Discretization 4b⟩
C = bd;
for i = 1:T-1
  C = [C ad * C(:, end)];
end
vpt = v/m^2 * c * C * C' * c';
vyt = vpt(1, 1);
```

It is tested against the empirical estimate by the following script.

13b ⟨Empirical computation of the variance of $y(T)$ 13b⟩≡
```
⟨Simulation parameters 12f⟩
yT = [];
for i = 1:100
  p = sim_ini_noise(xini, m, v);
  yT = [yT p(1, end)];
end
[var_yt(xini, m, v), var(yT)]
```

The theoretical value is 2.3821×10^{-5} and the empirical value is 2.8360×10^{-5}. □

Exercise 9: Use of feedback control to reduce the effect of the disturbances

An effective and often used way to deal with disturbances (as well as model uncertainty) is feedback. Feedback requires measurement of the object's state or output (position) and possibility to apply corrective action via an input. In the setup of Exercise 2, the control input is an external force f. (Think of the object as a rocket equipped with a thruster.) In this exercise, the force acts in horizontal direction (thus counteracting the wind force).

Design a feedback controller that keeps the object "on target" despite of disturbances. "On target" means that the object is supposed to land as close as possible to the specified location y_{des}. Measure the effectiveness of the feedback control by computing the bias (mean value of $y_{\text{des}} - y(T)$) and the variance of $y(T)$. Compare different control design methods with the "no feedback control" solution of Exercise 2.

Solution: To be added. □

Exercise 10: Optimal open loop control

The object now is equipped with two thrusters: one acting (as in Exercise 7) in horizontal direction and the second one in vertical direction. The two thrusters can be controlled independently. Let $u_1(t)$ be the force that the first thruster applies on the object at time t and $u_2(t)$ the force that the second thruster applies on the object at time t. Modify the initial conditions simulation function for the present scenario. Then, find the minimum energy control, *i.e.*, the inputs u_1 and u_2 that minimize the function

$$\|u\| = \sqrt{\int_0^T u_1^2(t) + u_2^2(t)\mathrm{d}t}$$

and transfers the object from a given initial state $x(0) = x_{\text{ini}}$ to a given final state $x(T) = x_{\text{des}}$ for a given time T. Implement the solution in a function and experiment with different values for x_{ini}, x_{des}, and T. Can you explain the shape of the optimal control curve? Think of the function computing the optimal control as a mapping

$$(x_{\text{ini}}, x_{\text{des}}, T) \mapsto u_{\text{ln}}, \tag{15}$$

where u_{ln} is the minimum norm control achieving the transfer. Find input values $(x_{\text{ini}}, x_{\text{des}}, T)$ for which $u_{\text{ln}} = 0$ and describe *all* such values. What kind of map is (15)? Find the kernel of (15) and relate it to the verbal description of the input values $(x_{\text{ini}}, x_{\text{des}}, T)$, for which $u_{\text{ln}} = 0$.

Solution: A state space representation of the model with horizontal and vertical thrusters is

$$\dot{x} = Ax + Bu, \qquad y = Cx, \qquad \text{where} \quad B = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}^\top.$$

and A, C are defined in (3) (for the case with friction (10)).

14a ⟨*Control input* 14a⟩≡ (14b 15b)

```
b  = [0 0; 1 0; 0 0; 0 1; 0 0];
```

The discretization process is equivalent to this in Exercise 6, so that the simulation function is

14b ⟨`sim_ini_input` 14b⟩≡

```
function [p, t, x] = sim_ini_input(u, xini, m)
⟨State-space model 3⟩
⟨Control input 14a⟩
⟨Discretization 4b⟩
⟨Simulation with input 15a⟩
```

where

15a ⟨*Simulation with input* 15a⟩≡ (14b)
```
⟨Define g 4d⟩
x  = [xini; -g];
Td = length(u);
for k = 1:Td
    x = [x ad * x(:, end) + bd * u(:, k)];
end
p = c * x; t = 0:ts:(Td * ts);
```

The optimal control input for the discretized system is computed by solving a least norm problem, as explained on page 10 in Lecture 4.

15b ⟨`opt_control` 15b⟩≡
```
function u = opt_control(xini, xdes, T, m)
⟨State-space model 3⟩
⟨Control input 14a⟩
⟨Discretization 4b⟩
⟨Define g 4d⟩
⟨Construct the controllability matrix 15c⟩
⟨Compute the optimal control 15d⟩
```

where

15c ⟨*Construct the controllability matrix* 15c⟩≡ (15b)
```
C = bd; Td = round(T / ts);
for i = 1:(Td - 1),
    C = [C, ad * C(:, (end - 1):end)];
end
```

and

15d ⟨*Compute the optimal control* 15d⟩≡ (15b)
```
xc =  [xdes; -g] - ad ^ Td * [xini; -g];
u = C(1:4, :)' * inv(C(1:4, :) * C(1:4, :)') * xc(1:4);
u = fliplr(reshape(u, 2, Td));
```

The following code test the functions `sim_ini_input` and `opt_control`.

15e ⟨`test_opt_control` 15e⟩≡
```
xini = [1; 1; 1; 1]; xdes = [2; 1; 1; 1]; T = 1; m = 1;
u = opt_control(xini, xdes, T, m);
[p, t, x] = sim_ini_input(u, xini, m);
plot(p(1, :), p(2, :)), print_figure(8)
```

The result is the trajectory shown in Figure 9. (By varying the initial and desired state, you can get interesting plots. The challenge is to explain why the optimal trajectory looks like the way it does.) □

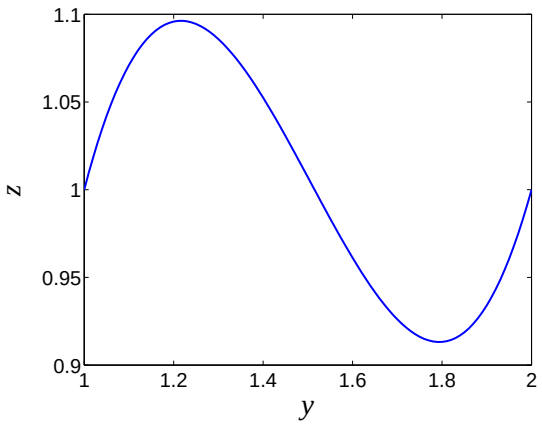

Figure 9: Optimal state transfer.

References

[1] S. Papert. *Mindstorms: Children, Computers, And Powerful Ideas*. Basic Books, 1993.

[2] S. Papert. *The Children's Machine: Rethinking School In The Age Of The Computer*. Basic Books, 1994.

[3] S. Papert. *The Connected Family: Bridging the Digital Generation Gap*. Longstreet Press, 1996.