# Dynamical systems and control mindstorms

Ivan Markovsky

*Abstract*— **An unorthodox programme for teaching systems and control is developed and tested at the School of Electronics and Computer Science of the University of Southampton. Motivation for the employed teaching methods is Moore's method and S. Papert's book "Mindstorms: children, computers, and powerful ideas". The teaching is shifted from lecture instruction to independent work on computer based projects and physical models. Our experience shows that involvement with projects is more effective in stimulating curiosity in systems and control related concepts and in achieving understanding of these concepts. The programme consists of two parts: 1) analytical and computational exercises, using Matlab/Octave, and 2) laboratory exercises, using programmable Lego mindstorms models. Both activities cut across several disciplines—physics, mathematics, computer programming, as well as the subject of the programme—systems and control theory.**

## I. Introduction

> 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 exercises and the laboratory, described in the paper, 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 of asking questions, *i.e.*, aiming for open enquiry rather than following of authority, and the ability to turn vague questions into well defined problems. 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 a student needs France for learning french, the student needs "systems and control land" for learning dynamical systems and control theory. The exercises and laboratory described in the paper aim to serve this role.

This work is supported by ERC grant number 258581.

I. Markovsky is with School of Electronics and Computer Science, University of Southampton, Southampton, SO17 1BJ, UK. `im@ecs.soton.ac.uk`

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.

Our aim is to give the students an opportunity to practice systems and control knowledge by working on a series of exercises which are formulated as vague questions rather than well defined problems. These questions are curiosity driven, so that the student should be motivated to find answers. The students have full freedom to choose the problem formulation and solution method that they deem relevant for the question at hand. Moreover, the students are encouraged to find their own style in dealing with the assignments and ask questions that are not part of the assignments.

Given such freedom, the students may come up with different answers to the original questions (as well as answers to questions that were never asked). As long as the answers are correct solutions to relevant problems, related to the original question, they are "valid answers".

## II. Setup of the exercises

In the exercises the students are 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 determine the subsequent free fall. How they are achieved is not important for the purpose of studying the free fall. 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). Initially the simplifying assumption that the object is flying in vacuum is made, so that the the only force acting on the stone is the gravitational force.

Throughout its free falling flight the object remains in the plane determined by the initial velocity and the gravitational force. Therefore, although the object is flying in a three dimensional space, its motion can be described in a plane. Let $p(t)$ be the position of the object at time $t$. We choose a reference moment of time $t = 0$ to be the moment when the object 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.

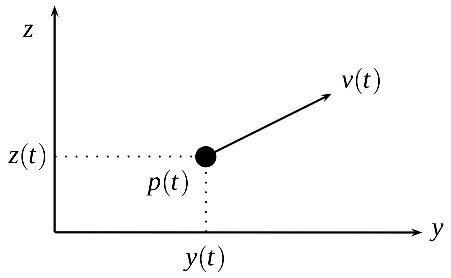

Fig. 1. Problem setup.

The horizontal displacement of the object at time $t$ with respect to the origin is denoted by $y(t)$ and the vertical displacement by $z(t)$. More notation, used in the exercises 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 $t$ |
| $p_{\text{ini}}$, $v_{\text{ini}}$ | — | 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 object is described by the differential equation

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

Here $p_{\text{ini}}$ is the initial position (the place from where the object is thrown) and $v_{\text{ini}}$ is the initial velocity (by which the object is thrown). Note the following facts about (1). It is

- a linear differential equation with constant coefficients;
- a second order, vector equation;
- the right hand side is constant; and
- does not depend on the mass $m$.

The last item implies that any object falling in a gravitational field without friction has the same trajectory. The fact that the trajectory does not depend on the mass may be surprising and counter intuitive for some students. This is a welcome instance when physics and mathematics reveal something that is not obvious.

Equation (1) described completely 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, the first task in the series of exercises is to find the solution explicitly; both analytically (by hand) and numerically (using the computer). The subsequent exercises are studying

- free fall with friction,
- bungee jumping,
- hitting a stationary object,
- throwing an object as far as possible,
- deducing the initial state from observed trajectory,
- hitting a moving object,
- effect of disturbances,
- use of feedback control to reduce the effect of the disturbances, and
- optimal open loop control.

Description of the selected exercises and indicative solutions are given in the Appendix.

## III. Laboratory

The aim of the laboratory is to expose the students to a complete control system design cycle, starting with modeling of the plant, proceeding to design of a controller based on the model, implementation of the controller on a microprocessor, and finishing by experimental verification of the designed control system. Significant part of the work is on a computer and involves Matlab and C programming. Matlab is used for the parameter estimation and controller design and C is used for implementation of the control algorithm on the microcontroller.

The assignment is to design a controller for a cart, build from an Lego Mindstorms NXT constructor, see Figure 2. The cart includes a micro controller that is connected to an ultrasonic distance sensor and two electrical motors — one for the left wheels and one for the right wheels.

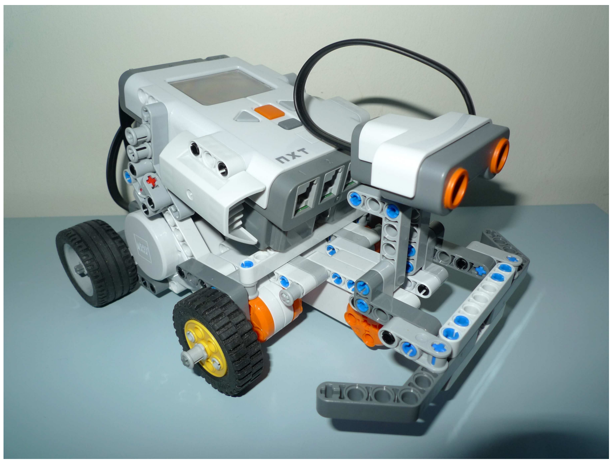

Fig. 2. Test platform: a cart build from Lego Mindstorms NXT constructor.

The control objective is to keep the cart at a constant distance from a moving object. For simplicity the cart and the object are assumed to move along a line, so that the problem has one degree of freedom (back-forth motion). Therefore, the same power is applied to both motors (resulting in the cart moving back-forth).

The first task of the laboratory is to identify a model for the relation between the power level applied to the motors (the control signal $u$) and the distance of the cart to the object (the controlled signal $y$). A parameterized model structure

$$H(s) = \frac{1}{s}\frac{b}{s+a}$$

is postulated from physical considerations and the parameters $a$ and $b$ are estimated from a measured response of the cart to a step input. The objective of the parameter estimation is to obtain a model that accurately reproduces the actual dynamical relation. This is validated by independent measurements and simulation of the model.

The controller design is initially carried out in Matlab (second task), where the controller is tested on the plant's model. Then, the designed controller is tested on the physical plant (third task) and the behaviour of the actual closed-loop system is compared to this observed in simulation.

Different programming languages can be used for implementing the discrete-time controller on the LEGO Mindstorms NXT microcontroller. In this laboratory, the students are given a program written in a C-like language, called Not eXactly C (NXC) [4]. NXC is specially created for development of Lego Mindstorms NXT applications and is supported by a powerful integrated programming environment, called the Bricx Command Center.

The skills learnt/practised in the laboratory are:

- data preprocessing by a moving average filter,
- parameter estimation (least squares optimization),
- PID controller tuning,
- discretization of a continuous-time plant,
- programming of a microcontroller in C,
- use of multitasking and concurency in C, and
- use of Matlab, Simulink, and the Control Toolbox.

An up-to-date description of the laboratory and related files are available from:

`www.ecs.soton.ac.uk/~im/lego.html`

## IV. Conclusion, discussion, and outlook

A dynamical systems and control programme was presented that departs from the traditional lecture style and employs curiosity driven, discovery based, learning. The medium for doing this is a series of exercises with loosely defined questions that leave freedom for the students to formulate precise problems and choose the relevant tools or methods for solving the problems. Our experience shows that this teaching method is more enjoyable and engaging for both students and teachers than the standard lecture based instruction. A negative side of the new method is that it requires more teachers and teaching assistants per student than the standard lecture based instruction and covers less material in a given period of time. We believe, however, that these shortcomings are far surpassed by the advantage of increasing the interest in the subject and achieving deeper understating for students of all abilities by the new method.

In future we aim to extend the exercises with more curiosity driven questions of diverse engineering nature and to add laboratories with different test models, comparing different control methods. The challenge that we faced in the reported preliminary work and expect to have in the future development is that inevitably the subject matter grows beyond the topic of systems and control. Indeed free enquiry is not limited to one subjects but cuts across many related and (seemingly) unrelated subjects.

## V. Acknowledgments

Mark French encouraged me to revise the second year control laboratories and pointed out the use of Lego Mindstorms models by Glenn Vinnicombe in Cambridge. Mark and Chris Freeman contributed with suggestions about the development of the Lego lab presented in the paper.

The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC Grant agreement number 258581 "Structured low-rank approximation: Theory, algorithms, and applications".

## References

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

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

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

[4] J. Hansen, *LEGO Mindstorms NXT Power Programming: Robotics in C*. Variant Press, 2007.

[5] F. Jones, "The Moore method," *American Mathematical Monthly*, vol. 84, pp. 273–277, 1977.

[6] J. Parker, *R. L. Moore: Mathematician and Teacher*. Mathematical Association of America, 2005.

[7] M. Mansour and W. Schaufelberger, "Software and laboratory experiments using computers in control education," *IEEE Control Systems Magazine*, vol. 9, pp. 19–24, 1989.

[8] S. Skelton and T. Iwasaki, "Increased roles of linear algebra in control education," *IEEE Control Systems Magazine*, vol. 15, pp. 76–90, 1995.

[9] M. Smith, "United Kingdom control education," *IEEE Control Systems Magazine*, vol. 16, pp. 51–56, 1989.

[10] D. Bernstein, "Enhancing undergraduate control education," *IEEE Control Systems Magazine*, vol. 19, pp. 40–43, 1999.

## Appendix

### 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

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

$$\text{and} \quad C := \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}. \tag{3}$$

We have taken into account the constant input $\mathbf{g}$ 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).

⟨*State-space model* 4a⟩≡

```
  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 $x_{\text{ini}}$, 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

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

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):

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

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 $t_s$,

⟨*Define* `ts` 4b⟩≡

```
  ts = 0.01;
```

the corresponding discrete-time equation is

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

⟨*Discretization* 4c⟩≡
  ⟨*Define* `ts` 4b⟩

```
  ad = expm(a * ts);
```

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

⟨*Simulation* 4d⟩≡
  ⟨*Define* `g` 4e⟩

```
  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

⟨*Define* `g` 4e⟩≡

```
  g = 9.81;
```

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

⟨`sim_ini` 4f⟩≡

```
  function [p, t, x] = sim_ini(xini)
```

  ⟨*State-space model* 4a⟩
  ⟨*Discretization* 4c⟩
  ⟨*Simulation* 4d⟩

The function is tested with the following script

⟨`test_sim_ini` 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)

⟨`test_sim_ini` 4g⟩+≡
  ⟨*Define* `g` 4e⟩

```
  plot(xini(1) + xini(2) * t, ...
  xini(3) + xini(4) * t - 0.5 * g * t .^2, '-r')
  print_figure(1)
```

The test script produces the plot shown in Figure 3. □

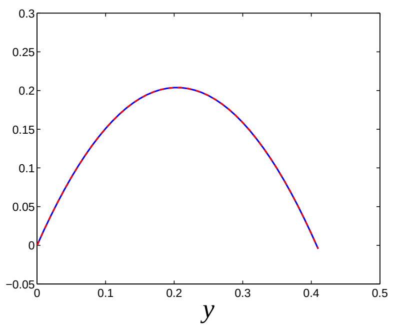

Fig. 3. Example of a simulated trajectory with initial condition $x_{\text{ini}} = (0,1,0,2)$. (Solid blue line — numerical solution, Dashed red line — analytic solution.)

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

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

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

⟨*State-space model with friction* 5a⟩≡
```
  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. 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 , is that the state space model is different:

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

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

⟨`test_sim_ini` 4g⟩+≡
```
  [p, t] = sim_ini_friction(xini, 1, 1);
  plot(p(1,:), p(2,:))
  print_figure(6)
```
The result is shown in Figure 4. 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. □

## 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 $p_\text{r}$, see Figure 5. The force exerted on the object from the rope is

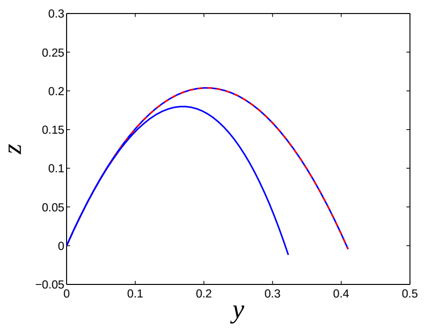

Fig. 4. Example of a simulated trajectories with (solid blue) and without (dashed red) friction. The initial condition is $x_\text{ini} = (0, 1, 0, 2)$, the mass is $m = 1$, and the friction coefficient is $\gamma = 1$.

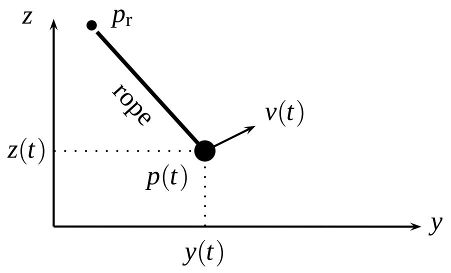

Fig. 5. Setup for Exercise 3.

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

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

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.

## 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 $v_\text{ini}$, so that the object starting from initial position $p_\text{ini} = 0$ reaches a given position $p_\text{des}$ 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 $p_\text{des}$ and $T$ and returns as an output the initial velocity $v_\text{ini}$ that achieves a trajectory, such that $p(T) = p_\text{des}$. Test your function for some $p_\text{des}$ and $T$. Modify your solution for the case when there is friction in the air.

## 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.*,

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

where $\theta$ is the to-be-determined parameter. The problem is

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

From (6) and (12), we have

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

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

⟨test_opt_sol 6a⟩≡
```
  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)
```

produces the plots shown in Figure 6. □

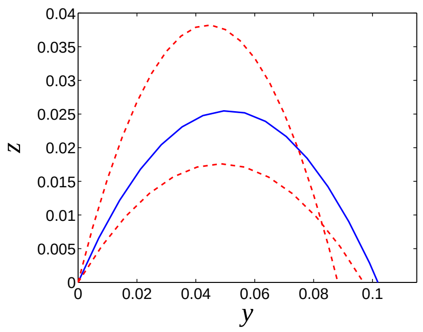

Fig. 6. Optimal (solid blue) and suboptimal (dashed red) throws.

**Exercise 6: Deducing initial state from 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

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

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 $x_{\text{ini}}$ 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.

⟨est_xini 6b⟩≡
```
  function xini = est_xini(p, t)
  ⟨State-space model 4a⟩
  ⟨Define g 4e⟩
  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

⟨test_est_xini 6c⟩≡
```
  [p, t] = sim_ini([1 2 3 4]');
  est_xini(p(:, 1:4), t(1:4))
```

□