Dublin R Workshop on Linear Dynamical Systems

Mick Cooney
michael.cooney@applied.ai

Oct 20, 2015

`https://bitbucket.org/kaybenleroll/dublin_r_workshops`.

The content from this workshop is largely taken from Stanford's EE263 course on Introduction to Linear Dynamical Systems. The homepage for the course as a whole is available at

`http://floatium.stanford.edu/ee263/`.

Despite the title, it is also a course on applied linear algebra.

In the event that you would like a more basic course on linear algebra, I recommend Gilbert Strang's course from MIT:

`http://ocw.mit.edu/courses/mathematics/18-06-linear-algebra-spring-2010/video-lectures/`

The video lectures for these courses are also on YouTube.

1. Introduction

A *continuous-time* linear dynamical system (CT LDS) has the form

$$\dot{x} = \frac{dx}{dt} = A(t)x(t) + B(t)u(t), \quad y(t) = C(t)x(t) + D(t)u(t),$$

where

- $t \in \mathbb{R}$
- $x(t) \in \mathbb{R}^n$ is the *state*
- $u(t) \in \mathbb{R}^m$ is the *input* or *control*
- $y(t) \in \mathbb{R}^p$ is the *output*
- $A(t) \in \mathbb{R}^{n \times n}$ is the *dynamics matrix*
- $B(t) \in \mathbb{R}^{n \times m}$ is the *input matrix*
- $C(t) \in \mathbb{R}^{p \times n}$ is the *output* or *sensor matrix*
- $D(t) \in \mathbb{R}^{p \times m}$ is the *feedthrough matrix*

These are often written more conveniently as

$$\dot{x} = Ax + Bu, \quad y = Cx + Du,$$

and is a first-order vector *differential equation*. They are also referred to as *state equations* or an 'm-input, n-state, p-output' linear dynamical system.

The *discrete-time* linear dynamical system (DT LDS) has the form:

$$x(t+1) = A(t)x(t) + B(t)u(t), \quad y(t) = C(t)x(t) + D(t)u(t),$$

where

- $t \in \mathbb{Z} = 0, \pm 1, \pm 2, ...$
- (vector) signals x, u, y are *sequences*

DT LDS is a first-order vector *recursion*.

2. Problem: A Simple Power Control Algorithm for a Wireless Network

PROBLEM:
We consider a network of n transmitter/receiver pairs. Transmitter i transmits at power level p_i (which is positive). The path gain from transmitter j to receiver i is G_{ij} (which are all nonnegative, and G_{ii} are positive).

The signal power at receiver i is given by $s_i = G_{ii}p_i$. The noise plus interference power at receiver i is given by

$$q_i = \sigma + \sum_{j \neq i} G_{ij} p_j$$

where $\sigma > 0$ is the self-noise power of the receivers (assumed to be the same for all receivers). The *signal to interference plus noise ratio* (SINR) at receiver i is defined as $S_i = \frac{s_i}{q_i}$.

For signal reception to occur, the SINR must exceed some threshold value γ (often in the range 3 – 10). Various *power control algorithms* are used to adjust the powers p_i to ensure that $S_i \geq \gamma$ (so that each receiver can receive the signal transmitted by its associated transmitter).

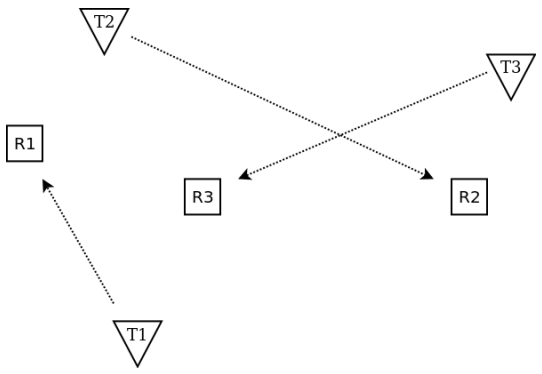
In this problem, we consider a simple power control update algorithm.

The powers are all updated synchronously at a fixed time interval, denoted by $t = 0, 1, 2, \ldots$. Thus the quantities p, q, and S are discrete-time signals, so for example $p_3(5)$ denotes the transmit power of transmitter 3 at time epoch $t = 5$.

What we would like is

$$S_i(t) = \frac{s_i(t)}{q_i(t)} = \alpha\gamma,$$

where $\alpha > 1$ is an SINR safety margin (of, for example, one or two dB). Note that increasing $p_i(t)$ (power of the i-th transmitter) increases S_i but decreases all other S_j.



A very simple power update algorithm is given by

$$p_i(t+1) = p_i(t)\left(\frac{\alpha\gamma}{S_i(t)}\right). \tag{1}$$

This scales the power at the next time step to be the power that would achieve $S_i = \alpha\gamma$, if the interference plus noise term were to stay the same. But unfortunately, changing the transmit powers also changes the interference powers, so its not that simple!

Finally, we get to the problem:

(a) Show that the power control algorithm (1) can be expressed as a linear dynamical system with constant input, i.e., in the form $p(t+1) = Ap(t) + b$, where $A \in R^{nn}$ and $b \in R^n$ are constant. Describe A and b explicitly in terms of σ, γ, α and the components of G.

(b) Simulation. Use R to simulate the power control algorithm (1), starting from various initial (positive) power levels. Use the problem data:

$$G = \begin{bmatrix} 1.0 & 0.2 & 0.1 \\ 0.1 & 2.0 & 0.1 \\ 0.3 & 0.1 & 3.0 \end{bmatrix}, \quad \gamma = 3, \quad \alpha = 1.2, \quad \sigma = 0.01. \tag{2}$$

Plot S_i and p as a function of t, and compare it to the target value $\alpha\gamma$. Repeat for $\gamma = 5$. Comment briefly on what you observe.

Comment: You'll soon understand what you see.

WORKSHOP:
We can manipulate the definitions to get some sense of the update rule:

$$\begin{aligned} p_i(t+1) &= p_i(t)\left(\frac{\alpha\gamma}{S_i(t)}\right) \\ &= \frac{\alpha\gamma p_i(t) q_i(t)}{s_i(t)} \\ &= \frac{\alpha\gamma p_i(t)\left[\sigma + \sum_{j\neq i} G_{ij} p_j(t)\right]}{G_{ii} p_i(t)} \\ &= \frac{\alpha\gamma\left[\sigma + \sum_{j\neq i} G_{ij} p_j(t)\right]}{G_{ii}} \end{aligned}$$

In matrix form:

$$\begin{bmatrix} p_1(t+1) \\ p_2(t+1) \\ p_3(t+1) \end{bmatrix} = \begin{bmatrix} 0 & \frac{\alpha\gamma G_{12}}{G_{11}} & \frac{\alpha\gamma G_{13}}{G_{11}} \\ \frac{\alpha\gamma G_{21}}{G_{22}} & 0 & \frac{\alpha\gamma G_{23}}{G_{22}} \\ \frac{\alpha\gamma G_{31}}{G_{33}} & \frac{\alpha\gamma G_{32}}{G_{33}} & 0 \end{bmatrix} \begin{bmatrix} p_1(t) \\ p_2(t) \\ p_3(t) \end{bmatrix} + \begin{bmatrix} \frac{\alpha\gamma\sigma}{G_{11}} \\ \frac{\alpha\gamma\sigma}{G_{22}} \\ \frac{\alpha\gamma\sigma}{G_{33}} \end{bmatrix}$$

So with the substitutions for A and b we get

$$p_i(t+1) = Ap(t) + b$$

```
G <- matrix(c(1.0, 0.2, 0.2,
              0.1, 2.0, 0.4,
              0.3, 0.1, 3.0), ncol = 3, byrow = TRUE);
```

```
gamma <- 3.0;
alpha <- 1.2;
sigma <- 0.01;
```

Exercise 2.1 Using the basic set up described in the question, plot the SINR and power output over time

Exercise 2.2 Create a function that allows you to produce the data for the plots as a function

Exercise 2.3 Using the same setup but different initialisations, check what happens for different starting power initialisations.

Exercise 2.4 Redo the previous work using $\gamma = 5$. What difference do you notice?

Exercise 2.5 Check the eigenvalues of the two constructed matrices A. Does anything stand out as a possible explanation for the difference in behaviour?

Exercise 2.6 Repeat the work using a different value for G.

3. Problem: Temperatures in a Multi-core Processor

PROBLEM:

We are concerned with the temperature of a processor at two critical locations. These temperatures, denoted $T = (T_1, T_2)$ (in degrees C), are affine functions of the power dissipated by three processor cores, denoted $P = (P_1, P_2, P_3)$ (in watts). We make 4 measurements. In the first, all cores are idling, and dissipate 10W. In the next three measurements, one of the processors is set to full power, 100W, and the other two are idling. In each experiment we measure and note the temperatures at the two critical locations.

P_1	P_2	P_3	T_1	T_2
10W	10W	10W	27C	29C
100W	10W	10W	45C	37C
10W	100W	10W	41C	49C
10W	10W	100W	35C	55C

Suppose we operate all cores at the same power, p. How large can we make p, without T_1 or T_2 exceeding 70C?

You must fully explain your reasoning and method, in addition to providing the numerical solution.

WORKSHOP:

Exercise 3.1 Set up the linear system as given in the problem.

Exercise 3.2 Find the maxium power that all cores can use without exceeding 70C.

Exercise 3.3 How does this change for 80C?

Exercise 3.4 Suppose we need to run a processor at 100W, and the other two at 50W. Assume all processors are equal, how do we do this to minimise the largest temperature?

4. Problem: Concentration of Chemicals in Reaction Kinetics

PROBLEM:

When a linear system has no input, it is called an *autonomous* system:

$$\dot{x} = Ax,$$

which can be a good model for the concentration x_i of chemical i in some reaction chains.

Suppose we have a reaction chain:

$$C_1 \xrightarrow{k_1} C_2 \xrightarrow{k_2} C_3$$

then we can model it as the following linear system:

$$\dot{x} = \begin{bmatrix} -k_1 & 0 & 0 \\ k_1 & -k_2 & 0 \\ 0 & k_2 & 0 \end{bmatrix} x$$

To model this as a discrete-time system we can see that:

$$x(t+1) = (I + hA)x(t)$$

where h is the size of the timestep and is typically 'small'.

WORKSHOP:

Exercise 4.1 Plot the concentrations x_i over time t with $k_1 = k_2 = 1$ and $h = 0.01$.

Exercise 4.2 Repeat for $h = 0.001$ and $h = 0.1$ and $h = 1$.

Exercise 4.3 Repeat for $k_1 = k_2 = 0.1$ and the various timestep sizes we have used. Is there a connection with previous outputs?

Exercise 4.4 Create a new dynamics matrix A that allows for more complex interactions. Are there any constraints we must impose for it to make sense?

Exercise 4.5 Create an analogous system for $N = 5$. Are there any qualitative differences from the $N = 3$ system?

5. Problem: Optimal Control of a Unit Mass

PROBLEM:

In this problem you will use R to solve an optimal control problem for a force acting on a unit mass. Consider a unit mass at position $p(t)$ with velocity $\dot{p}(t)$, subjected to force $f(t)$, where $f(t) = x_i$ for $i1 < t \leq i$, for $i = 1, ..., 10$.

(a) Assume the mass has zero initial position and velocity: $p(0) = \dot{p}(0) = 0$. Find x that minimises

$$\int_0^{t=10} f(t)^2 \, dt$$

subject to the following specifications: $p(10) = 1$, $\dot{p}(10) = 0$, and $p(5) = 0$. Plot the optimal force f and the resulting p and $\dot{p}$. Make sure the specifications are satisfied. Give a short intuitive explanation for what you see.

(b) Assume the mass has initial position $p(0) = 0$ and velocity $\dot{p}(0) = 1$. Our goal is to bring the mass near or to the origin at $t = 10$, at or near rest, i.e. we want

$$J_1 = p(10)^2 + \dot{p}(10)^2$$

small, while keeping

$$J_2 = \int_0^{t=10} f(t)^2 \, dt$$

small, or at least not too large. Plot the optimal trade-off curve between J_1 and J_2. Check that the end-points make sense to you. *Hint:* the parameter μ has to cover a very large range, so it usually works better in practice to give it a logarithmic spacing, using e.g. `logspace` in Matlab. You do not need more than 50 or so points on the trade-off curve.

Your solution to this problem should consist of a clear written narrative that explains what you are doing, and gives formulas symbolically; the Matlab source code you devise to find the numerical answers, along with comments explaining it all; and the final plots produced by Matlab

WORKSHOP:

To show how we set this system up as a linear system, we will work through it over 4 timesteps. The key to this problem is to find the linear relationship between the forces f_i and the final position and velocity (at $t = 4$) $x(4)$ and $\dot{x}(4)$ respectively. In other words, we should find the 2×4 matrix A such that

$$\begin{bmatrix} \dot{x}(4) \\ x(4) \end{bmatrix} = A \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix}$$

First consider the time interval $0 \leq t < 1$. We have

$$\frac{d^2x}{dt^2} = f_1, \; x(0) = 0, \; \dot{x}(0) = 0.$$

This differential equation gives

$$\dot{x}(t) = f_1 t, \; x(t) = \frac{1}{2} f_1 t^2,$$

so at $t = 1$ the position and velocity become $x(1) = \frac{1}{2} f_1$ and $\dot{x}(1) = f_1$.

Now in the time interval $1 \leq t < 2$

$$\frac{d^2x}{dt^2} = f_2,\ x(1) = \frac{1}{2}f_1,\ \dot{x}(1) = f_1,$$

which gives

$$\dot{x}(t) = f_2(t-1) + f_1,\ x(t) = \frac{1}{2}f_2(t-1)^2 + f_1(t-1) + \frac{1}{2}f_1,$$

and therefore $x(2) = \frac{1}{2}f_2 + \frac{3}{2}f_1$ and $\dot{x}(2) = f_1 + f_2$.

For $2 \leq t < 3$

$$\frac{d^2x}{dt^2} = f_3,\ x(2) = \frac{1}{2}f_2 + \frac{3}{2}f_1,\ \dot{x}(2) = f_1 + f_2,$$

and we get

$$\begin{aligned} \dot{x}(3) &= f_1 + f_2 + f_3 \\ x(3) &= \frac{5}{2}f_1 + \frac{3}{2}f_2 + \frac{1}{2}f_3, \end{aligned}$$

Proceeding in the same way we finally have:

$$\begin{bmatrix} \dot{x}(4) \\ x(4) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ \frac{7}{2} & \frac{5}{2} & \frac{3}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix}.$$

Finally, note that $\int_0^{t=10} f(t)^2\,dt$ is nothing but $||x||^2$ since

$$\begin{aligned} \int_0^{t=10} &= \int t = 1_0 x_1^2\,dt + \int_1^{t=2} x_2^2\,dt + ... + \int t = 10_9 x_{10}^2\,dt \\ &= x_1^2 + x_2^2 + ... + x_{10}^2 \\ &= ||x||^2. \end{aligned}$$

(a) We can now express the constraints $p(10) = 1$, $\dot{p}(10) = 0$, and $p(5) = 0$ can be expressed as

$$\begin{bmatrix} p(10) \\ \dot{p}(10) \\ p(5) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 9.5 & 8.5 & 7.5 & 6.5 & 5.5 & 4.5 & 3.5 & 2.5 & 1.5 & 0.5 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 4.5 & 3.5 & 2.5 & 1.5 & 0.5 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} x. \tag{3}$$

Thus the optimal x is given by the minimiser of $||x||$ subject to (3). In other words, we want the minimum norm solution of (3).

(b) We now have two competing objectives: keep J_1 small and keep $J_2 = ||x||^2$ small. To express J_1 we rewrite the equations of motion with $p(0) = 0$ and $\dot{p}(0) = 1$. It is easy to verify that this gives us

$$\begin{bmatrix} p(10) \\ \dot{p}(10) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 9.5 & 8.5 & 7.5 & 6.5 & 5.5 & 4.5 & 3.5 & 2.5 & 1.5 & 0.5 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} x + \begin{bmatrix} 10 \\ 1 \end{bmatrix} \tag{4}$$

Therefore, to minimise J_1 we have to minimise $||Ax - y||^2$, where

$$y = \begin{bmatrix} -10 \\ -1 \end{bmatrix}$$

So we need to calculate the minimiser of

$$J_1 + \mu J_2 = ||Ax - y||^2 + \mu ||x||^2$$

We can show that for a given value of μ, the minimiser for this, x_μ is given by

$$x_\mu = (A^T A + \mu I)^{-1} A^T y.$$

Exercise 5.1 Calculate the minimum norm solution x_{ln}.

Exercise 5.2 Draw plots of f, p and $\dot{p}$ over t.

Exercise 5.3 Ensure all these calculated values are internally consistent.

Exercise 5.4 Write a function that calculates x_μ for a given μ. *HINT:* You can use the function `solve()` to find the inverse of a matrix.

Exercise 5.5 Graph the tradeoff chart between J_1 and J_2. What different methods can we use visualise this?

6. Problem: The n-Transmitter, m-Receiver System

PROBLEM: The signal transmitted by n sources is measured at m receivers. The signal transmitted by each of the sources at sampling period k, for $k = 1, 2, ..., p$, is denoted by an n-vector $x(k) \in \mathbb{R}^n$. The gain from the j-th source to the i-th receiver is denoted by $a_{ij} \in \mathbb{R}$. The signal measured at the receivers is then

$$y(k) = Ax(k) + v(k), \quad k = 1, ..., p,$$

where $v(k) \in \mathbb{R}^m$ is a vector of sensor noises, and $A \in \mathbb{R}^{mn}$ is the matrix of source to receiver gains. However, we do not know the gains a_{ij}, nor the transmitted signal $x(k)$, nor even the number of sources present n. We only have the following additional a priori information:

- We expect the number of sources to be less than the number of receivers (i.e., $n < m$, so that A is skinny);
- A is full-rank and well-conditioned;
- All sources have roughly the same average power, the signal $x(k)$ is unpredictable, and the source signals are unrelated to each other; Hence, given enough samples (i.e., p large) the vectors $x(k)$ will point in all directions;
- The sensor noise $v(k)$ is small relative to the received signal $Ax(k)$.

We first will create this data so that we can better understand exactly what it is we need to do. This should also help us ensure we are not making any mistakes.

Exercise 6.1 Create a dataset for 10 receivers based on 3 transmitters with an r^2-distance attenuation. For the moment we ignore noise, and we can use the data loaded from the file `threetrans_inital.data.R` as our inputs.

Exercise 6.2 Use SVD methods on the matrix A can we relate this data to what we know about the data generation process.

Exercise 6.3 Repeat the above steps but with the data in `threetrans_sinewaves.data.R`. Can you explain what is happening here?

Exercise 6.4 Using the SVD output and some matrix algebra, try to recreate the signals and see how they compare with our 'known' signals.