System Identification – Practical Assignment 7
Pseudo-random binary sequences

Logistics are as before, see previous labs.

In this assignment we will study the creation and properties of pseudo-random binary sequences, PRBS. See the course material, Parts VI: *Input Signals*.

You will develop a function with the exact signature:

```
[index, pe3, pe10, u3, theta3, y10] = prbsidentify
```

Each student is assigned an index number in the set 1-7, which needs to be saved to variable `index` at the beginning of the function. A function `system_simulator` is available, which obtains experimental data from the system given an input signal. For the purposes of this lab, since we do not have access to the real system, this simulation takes the place of the real experiment, for both identification and validation. The signature of the function is `data = system_simulator(index, input)` where `index` is your index number, `input` is the input sequence (in discrete time), and `data` is the returned experimental data as a standard object of type `iddata`. Moreover, a predefined validation input `u` is provided in datafile `uval.mat`, containing the single variable `u`.

From prior knowledge, it is known that the system to be identified has order not larger than 4, and that the disturbance does not satisfy the structural assumptions of the ARX model. Nevertheless, due to its simplicity we choose to identify an ARX model, and to compensate for the disturbance structure we take large values for the orders: $na = nb = 15$. Note that in order to identify an ARX model, the input signal should satisfy a certain persistence of excitation condition, see the lecture for the details.

Requirements:

- Generate first a validation dataset with `system_simulator`. You will use this dataset to validate the models found below.
- Write code that generates an input signal of length N using a maximum-length PRBS with a register of a given length m, and which switches between given values a and b. Length m is limited to the range $3, 4, \ldots, 10$. Initialize the coefficients a using the table in the lecture, to obtain a maximal-period PRBS; note that if $N > P$, the period of the PRBS, then the input signal will consist of several repetitions of the maximum-length PRBS (this is normal). Test this function for some values of N, m, a and b. Hints: ◦ Using the state space representation of the linear shift feedback register, it is easy to create a vectorized implementation in Matlab (but you can also do an explicit implementation rather easily). ◦ You can use function `mod` to implement the modulo-2 summation. ◦ It is better to write a function for this, below your main solution function.
- Compute the orders of persistent excitation for a PRBS signal with $m = 3$ and $m = 10$ bits, respectively. Return the two orders in variables `pe3`, `pe10`.
- Generate an identification input signal of length $N = 300$ with $m = 3$, taking values $a = 0.5$ and $b = 1$, and an initial state equal to the binary representation of your index. E.g. if your index is 2, then the initial state should be $[0, 1, 0]^T$. Return this signal in `u3`. Note: only the first 10 values of this signal are checked by Grader.
- Apply this signal to the system using `system_simulator`, and use the resulting dataset to identify an ARX model. Return the parameter vector of this model in `theta3`. Simulate the model using the validation input, and compare the resulting output with the true output of the system.

Does the input data have a sufficient order of PE to identify the ARX model? Use the validation results to support your conclusion. Hints: ◦ To make things easy, use the Matlab function `arx` instead of your own code from the ARX lab (it will make the Grader solution cleaner and easier to debug). ◦ Check the template solution for ideas on how to simulate and how to pick up the parameter vector.

- Repeat the previous two bullet points, but now for $m = 10$, and returning instead the simulated model output in `y10`. Does this new data have a sufficient order of persistent excitation?
- Optionally, if you still have time: repeat the experiment, but now using the default input signal generator in the System Identification toolbox, `idinput`. Study the interface of this function and generate a PRBS taking values $a = 0.5$ and $b = 1$, without limiting its frequency content; note that the order m of the register is chosen automatically. Study also the possibilities for generating other types of input (white noise, sum of sines).