Part II – Decision Systems for Engineering Design
Laboratory A: Sampling Plans and Knowledge Discovery

Robin Purshouse
University of Sheffield

Spring Semester 2023/24

1 Introduction to the laboratories

1.1 Overview

In the laboratories for Part II of the module, you will use decision system technologies to design a digital control system for a plant. In your assignment, you will report on your lab activities and findings (see the separate Assignment Briefing for more details about the assessment requirements).

1.2 Aim of Laboratory A

In Laboratory A, your aim is to explore the design space for the problem: revealing the relationships that exist between design variables and performance criteria (using data modelling approaches where appropriate) and communicating these relationships clearly through visualisation methods.

1.3 Materials

In preparation for this laboratory, go to the ACS6124 Blackboard site and download the following files from the *Decision Systems* Laboratories area, and place them in your working directory (or other convenient location on the Matlab path):

- `evaluateControlSystem.m` and its utility functions `getPeakInfo.m`, `getRiseTime.m`, `getSettlingTime.m`, `getShoots.m`, and `getSSError.m`—these are the files needed to evaluate a control system design;
- `sampling.zip`—this is a set of open-source sampling plan routines and utilities developed for engineering design problems by Alexander Forrester and colleagues from the University of Southampton.

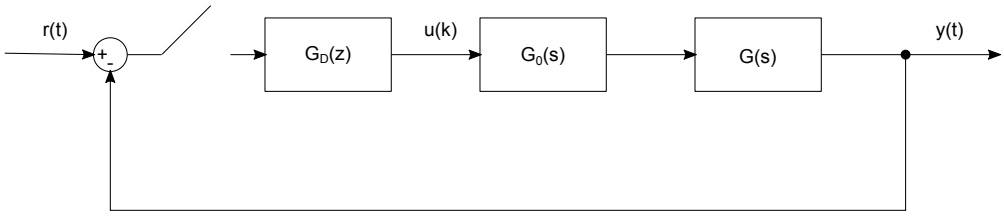

Figure 1: Digital control system

2 Background to the design problem

2.1 System architecture

The architecture for the problem is pre-defined. The digital control system, $G_D(z)$, is a Proportional-Integral (PI) controller situated in a unity-gain feedback arrangement. The plant, $G(s)$, is an analogue system and so a digital-to-analogue converter, $G_0(s)$ is needed between the controller and the plant. For analogue-to-digital conversion, the plant is sampled at 1 second intervals. The overall arrangement is shown in Figure 1.

2.2 Design variables

The control system design problem is to choose appropriate proportional and integral gains, K_P and K_I, for the PI controller.

2.3 Performance criteria

Ten performance criteria have been identified for this system:

1. Stability criteria:
   - (i) Magnitude of the largest pole in the closed-loop transfer function in the z-domain;
   - (ii) Gain margin;
   - (iii) Phase margin;
2. Transient performance criteria for $y(t)$ in response to a unit step in $r(t)$:
   - (iv) Rise time—from 10% to 90% of the final value of $y(t)$;
   - (v) Peak time;
   - (vi) Maximum overshoot;
   - (vii) Maximum undershoot;
   - (viii) Settling time—to within 2% of the final value of $y(t)$;
3. Steady-state performance criterion for $y(t)$ in response to a unit step in $r(t)$:

(ix) Steady-state error.

4. Sustainability performance criterion:

   (a) Aggregate control effort, $u(k)$, after 100 time steps.

2.4 Evaluation function

The performance criteria for a candidate controller design are evaluated using a difference equation model that has been implemented in Matlab. For the purposes of the Part II laboratories, you should treat the evaluation function as a closed-box (i.e. you do not have direct access to the transfer functions).

3 Task A.0 Preliminaries

Evaluations are made by calling the Matlab function `evaluateControlSystem`. This function takes a pair of decision variables, i.e. controller gains $[K_P \; K_I]$, as inputs and returns as output the performance of the system under those gains. The performance is a row vector, where the criteria are provided in the same order as presented in Section 2.3.

Note that multiple pairs of controller gains can be evaluated by in one function call, with each pair entered as a row in the input matrix.

Enter your initial guess for a suitable pair of controller gains into the function and check the performance against the criteria that you obtain. Does your controller deliver a stable system (i.e. is the magnitude of the largest pole in criterion one less than unity)? If so, does it perform well in terms of transient and steady-state performance?

4 Task A.1: Sampling Plan

4.1 Generating a plan

Your first task is to develop a space-filling sampling plan for the controller gains K_P and K_I. Hopefully this will not be too difficult, given that we have only two design variables! Allow yourself a budget of 100 candidate designs.

The Forrester functions contained in `sampling.zip` offer some options here:

- `fullfactorial` – generates a full factorial sampling plan on the unit hypercube;
- `rlh` – generates a plan according to a Latin hypercube design.

You could also use the Statistics & Machine Learning Toolbox function `lhsdesign` to generate a Latin hypercube design. Sobol sampling is also available in Matlab using the `sobolset` function and associated utilities.

Use the `help` command to learn more about the functions and how they need to be called. Note that, in all cases, you will need to scale the sampling plan to be on the

range you want for K_P and K_I. We also recommend you use logarithmic scaling so that you obtain good coverage of controller gains across different orders of magnitude.

4.2 Assessing a plan

Recall from Lecture 2 that different plans have different space-filling properties. It would be good to assess the capability of the plan that you've created. The Φ_q metric is one way of doing this and is available from the Forrester toolkit as the function `mmphi`.

Generate some different plans and assess their space-filling properties. What is the best plan that you can identify?

5 Task A.2: Knowledge Discovery

When you are happy with your sampling plan, use `evaluateControlSystem` to evaluate the designs in the plan.

Once the designs are evaluated, your objectives are to:

- Identify the relationships between the design variables and performance criteria for the problem;
- Present these findings to a Chief Engineer in a compelling way.

This part of the lab is open-ended and you are free to explore any options that you feel are the most promising, including methods not covered in the lectures. The Statistics & Machine Learning Toolbox offers some commands that can help you, and you may also wish to explore the Mathworks File Exchange for other options or perform visualisation outside of the Matlab environment (e.g. using D3[1]).

Options available within Matlab include:

- `plotmatrix` and `gplotmatrix` – matrix scatter plots;
- `parallelcoords` – parallel coordinates;
- `glyphplot` – glyphing approaches, even including the infamous Chernoff faces;
- `pca` – principal component analysis;
- `clusterdata` and `kmeans` – clustering algorithms.

Make sure you save all your visualisations with the Matlab `print` function using a good quality image format such as EPS, PNG or PDF.

[1] `https://d3js.org`