Using MATLAB for teaching control design and analysis

J.A. Rossiter *

* *Department of Automatic Control and Systems Engineering, University of Sheffield, South Yorkshire, U.K. S1 3JD. (e-mail: j.a.rossiter@shef.ac.uk).*

Abstract: Control design and analysis are numerically demanding and thus can be done efficiently only with suitable computer software. This paper looks at the use of MATLAB for supporting student learning of two types of control methodologies: (i) classical control in frequency response and (ii) model predictive control. The paper will first discuss the pedagogical background of how MATLAB supports learning and why MATLAB is the software of choice. This is followed by a description of some of the MATLAB tools that have been created. The paper is completed by some student evaluation of their experiences.

Keywords: Learning control basics, independent learning, laboratories

1. INTRODUCTION

This paper is focussed on the context of University education rather than industrial practice. As such, the principle aim of the lecturer is to encourage active student participation in learning and to structure course delivery to encourage deep Race (2005) rather than surface approaches to learning. For example, with many engineering topics which require high level mathematical algorithms, it can be too easy for staff to set assessments which require the implementation of algorithms by rote; students then learn that they can get a good enough mark by memorisation of procedures, but significantly, with little real understanding.

This paper looks at the teaching strategies which encourage students away from rote learning and into being inquisitive learners (inquiry based learning CILASS (2005)). In this model, students take much more control of their own learning and develop understanding by themselves; the lecturer becomes a facilitator rather than a teacher. It is recognised that student progress to a base level of competence can be slower, but their understanding is much deeper and hence they make better progress in the long term. They are also developing important skills needed for the workplace.

1.1 Systems engineering

Within the author's department there are numerous topics where there is an opportunity to make students more active in the learning process and to move them away from rote based learning approaches. In recent years therefore, there has been proactive development of appropriate assessments, some of which have already been discussed in the literature e.g. Rossiter (2007); Rossiter and Gray (2009). In this paper the focus is placed on specific topics where heavy numeric computation is required and thus historically academics were restricted to only using elementary examples and/or crude graphical approaches with pen and paper. The argument is very simple, now that computational power and software are cheap, for numerically demanding topics, it makes far more pedagogical sense to assess students in a situation where they have access to that software. Thus the examiner can focus on the students ability to understand and implement concepts, without being hamstrung by the need to do tedious numerical calculations on paper, because the computer will do these instantaneously Becerra (2008).

As the author is within a systems and control department, there are many key topics that fit the paragraph above, for example:

(1) Modelling and simulation of ODEs.
(2) Simulation using numerical methods (e.g. Runge Kutta).
(3) Control loop analysis, simulation and design.
(4) Core mathematics topics.
(5) Most higher level modules.

The department decided on two key pieces of software that all students should learn, because these were in common use within industry and also served the educational purposes; these were C programming and MATLAB/SIMULINK. C is used mainly for low level programming of microchips, embedded control, etc., whereas MATLAB/SIMULINK is used for simulation and analysis and thus is the topic discussed in this paper.

1.2 Paper contribution

This paper will focus on the use of MATLAB/SIMULINK to support two control analysis and design topics within the curriculum (although the software tools are also used

widely in other modules). There will be a discussion of where and how the software is used and also on the pedagogical reasoning for this. Finally there is some evaluation of student experiences with the software.

2. INTEGRATION OF MATLAB INTO THE FEEDBACK DESIGN CURRICULUM

This section will give a description of where and how MATLAB is incorporated whereas the next section will give more details of actual code and activities. The focus will be on two main topics:

(1) Classical control analysis and design using frequency response and time domain methods.
(2) Model predictive control (MPC).

2.1 Using MATLAB to support learning of classical control

Classical control is mainly taught in year 2 for systems engineers, aerospace engineers and biomedical engineers. The focus of the core module is on giving students a thorough grounding in the principles and concepts required to understand, or analyse, behaviour of feedback loops [1] .

The module covers topics such as block diagrams and closed-loop transfer functions, closed-loop poles and links to expected response, steady-state errors, root-loci, bode and nyquist diagrams, margins and analysis of lead/lag compensators (aided by mechanistic design rules).

Many of these topics are difficult to do by hand because they involve significant numerical computation, for example:

- Plotting closed-loop responses to validate steady-state errors requires root calculations, partial fractions and inverse Laplace.
- Computation of closed-loop poles cannot be done algebraically except for trivial examples.
- Root-loci cannot be done explicitly, say to validate a student sketch or insight, without a computer.
- Bode and Nyquist plots can only be sketched by hand and even rough sketches can require substantial computations.
- Margin computations cannot be done by hand except for a very small subset of problems or approximately from sketches.
- Compensator design and testing can only be done approximately from sketches and again would be very time consuming by hand.

In summary, the lecturer has a very limited set of examples they can get students to do themselves on paper because of the requirement for numbers to be straightforward for paper and pen exercises. Moreover, historically, students would have to spend long periods focussed on number crunching instead of focussing on concepts.

Within Sheffield it was decided that student learning and understanding of concepts would be improved significantly if they were able to do the numeric computations instantaneously and thus quickly test the impact of changes or their assumptions. Consequently, MATLAB was integrated explicitly into the curriculum as a tool for students to self validate their own learning, for example:

- Find the predicted steady-state error using the final value theorem (and easy paper computation) and then validate with MATLAB by plotting the closed-loop response.
- Sketch the root-loci by hand using the basic rules only and then validate by producing the actual plot with MATLAB.
- Sketch Bode and Nyquist plots using largely asymptote information (thus very quick) and validate with MATLAB.
- Use margins and encirclement information to infer expected closed-loop behaviour and then validate with MATLAB.
- Implement compensator designs on MATLAB and discuss.

Students were actively encouraged to use MATLAB to support their learning by two very simple tools.

(1) An assignment worth 20% of a 20 credit module was based entirely on effective use of software to generate answers. Students were given 'typical questions' to practise on in advance and supported tutorial slots throughout term so that well prepared students had an incentive and opportunity to practise. Students then had an invigilated assessment in a PC laboratory at the end of semester where they had 3 hours to produce numerous answers, all of which required substantial computation and figure production and thus could not be done on pen and paper.
(2) Several of the tutorial sheets embedded within the notes implicitly assumed that students would access MATLAB to test their answers and reinforce learning. The appropriate MATLAB scripts were demonstrated in lectures and provided in the handouts.

In order to support the students further still, numerous MATLAB files and GUIs were also made available for some tasks, a word document gave an overview of the key commands they would need and a FLASH lecture Rossiter (2010) was also provided to help learn MATLAB for control for the novice. Readers can find the resources suggested at the OER website OER (2010).

More detailed exemplars of tasks and MATLAB usage will be given in the next section.

2.2 Using MATLAB to support learning of model predictive control

The predictive control module is taught in a short fat fashion, which means students are present for a single week only with exams being several months later. Consequently the department has veered away from using assignments because students may not have access to the relevant software or support once the intense week has finished. There is also nervousness about setting assignments which are easy to plagiarise because each student has the same specification and it is impractical in general to give different assignments to different students. Hence, the module is assessed by exam only and focusses very much on stu-

[1] There are changes planned to introduce more design from 2010-11 which until now was done separately.

dents understanding of key concepts rather than numeric computations.

Nevertheless, it is clear that student understanding of concepts is reinforced by experimentation and therefore MATLAB laboratories are embedded within the intense week for students to try out some of the algorithms and concepts presented within the lectures. Students can of course access these MATLAB scripts later as well should they wish too OER (2010). The author demonstrates the scripts provided in an integrated fashion within the lectures so that students get the twofold approach of first demonstration followed by supported laboratories with worksheets.

Anecdotal feedback from the students about this provision is very positive and it was clear that the mechanics of going through an algorithm in a laboratory with the staff member present, but without having to do the detailed computations by hand, really helped some students understand the key steps better. Once again, the accessibility and broad usage of MATLAB within the teaching environment meant that providing students with this learning experience was straightforward.

3. EXEMPLARS OF MATLAB USAGE FOR CLASSICAL CONTROL

This section will give some detailed illustrations of how MATLAB is integrated into control analysis and design teaching for an introductory module. Each subsection details a different example although readers should consider that in terms of lecture delivery this is done in a more integrated fashion.

3.1 Computation of steady-state errors

Students are given a page of problems to solve on pen and paper, for example a typical question would be: given a system G(s) and controller K(s), compute steady-state errors to a step in the set point. Students are encouraged to try this first and then the next page of the notes gives solutions by way of the MATLAB code required to find and plot the numeric solutions. A simple numerical example is given next.

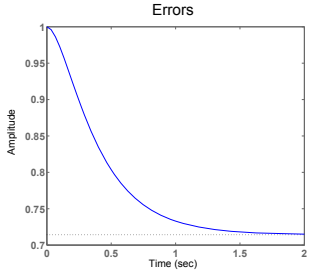
Problem solution on paper

$$\left.\begin{array}{c} G = \dfrac{1}{(s+10)(s+2)} \\ K = 8 \end{array}\right\} \Rightarrow \; e = \frac{1}{1+G(0)K(0)} = \frac{1}{1+8/20}$$

Problem solution with MATLAB

```
G=tf(1,poly([-2,-10]));
K=8;
step(feedback(1,G*K));
title('Errors')
```

MATLAB CODE



MATLAB FIGURE

The notes also give many more examples and code for computing errors to ramps which are more tricky on MATLAB as a ramp is a divergent signal. Critically the code is simple so students should be able to do this!

3.2 Closed-loop behaviour and links to pole positions

To compute poles and system responses is very tedious if not impossible on pen and paper. However, with MATLAB one can get students to speculate first, demonstrate they understand concepts and then check their expectations against what actually happens, thus reinforcing or correcting their understanding. A typical tutorial question is:

Problem statement

Compute the closed-loop poles for the following systems and comment on the expected behaviour as K varies. Validate your answers by explicit computation of: (i) Closed-loop step responses and (ii) Checking closed-loop poles on MATLAB.

$$\left.\begin{array}{c} G = \dfrac{1}{s^3+3s^2+3s+1} \\ K = 0.5 \end{array}\right\} \Rightarrow \; p_c = K + s^3 + 3s^2 + 3s + 1 \tag{1}$$

Problem solution in MATLAB CODE

The following code defines G(s) and then uses a loop to vary K so students get a dump to the command window of all the closed-loop poles for each different value of K. A closed-loop step response plot is also given at the same time. Students need to press enter to move from one value of K to the next.

```
G=tf(1,[1 3 3 1]);
for K=[0.1,0.5,1,2,5,10];
Gc=feedback(G*K,1);
disp(['K = ',num2str(K),' and poles are']);
[poles]=pzmap(Gc)
step(feedback(G*K,1));
disp('Press return'); pause
end
```

3.3 Bode asymptotes

Students spend a few weeks learning how to sketch Bode diagrams as these give huge insight into the performance expectations and thus design principles for lead and lag compensators as well as are easier to use than Nyquist diagrams. The expectations in the authors department are largely biased towards asymptotic information because this helps students understand key trends. Students are encouraged to generate asymptote plots by hand and then some software is provided for them to check their answers against. The author provides some ready made scripts to help this process OER (2010), including a GUI which builds asymptotes one part at a time so they can go through the process slowly for different systems.

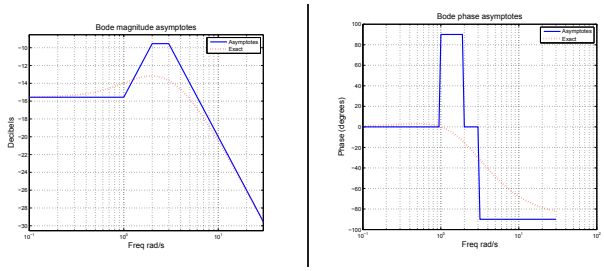
Problem statement

Compute the Bode asymptotes for G(s), improve the sketches with explicit computations at 2 key frequencies and then validate against MATLAB.

$$G = \frac{s+1}{(s+2)(s+3)} \tag{2}$$

Problem solution in MATLAB CODE and PLOTS

G=tf([1 1],[1 5 6]); bodeasymptote(G)



3.4 Lead design

The most complex problem in the introductory module is the lead design. However, MATLAB is an excellent tool for doing this in a mechanistic fashion while giving students highly visual information on the impact of each step. This paper illustrates the simplicity of the steps required by the students as well as the quality of the easy to generate figures that come out. The key design is undertaken on sisotool and then the compensator is exported to the workspace for overlaying with other compensators for comparison [2] .

A typical question during a laboratory assessment could be to compare the rise time and maximum input for a lead compensator design compared to a simple gain design. This would lead into a typical exam question which ask students to discuss in general the differences between lead designs and simple gain designs.

Mechanistic design rules [For within sisotool]

(1) Use 'drag' of the Bode plot [3] to get a 60 degree phase margin. Determine K and the associated gain cross over frequency w_g.
(2) Assume the lead compensator can double the gain cross over frequency to $w_l = 2w_g$.
(3) Find the gain K_l which makes w_l the gain cross over frequency - again this is done by simple drag of the Bode gain plot.
(4) Detemine the phase rotation required to make the phase margin 60 degrees and hence choose required ratio β of pole to zero (from a table, e.g. $\phi = 20 \Rightarrow \beta = 2, \phi = 30 \Rightarrow \beta = 3$, etc.) and then the lead compensator as:

$$K_{lead} = K_l\sqrt{\beta}\frac{s + w_l/\sqrt{\beta}}{s + w_l\sqrt{\beta}} \quad (3)$$

(5) Check the new phase margin is as expected. [In case of errors in data entry]

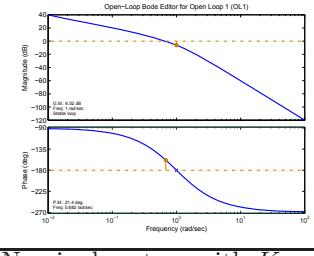
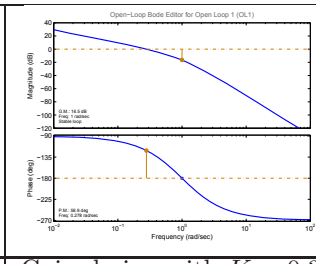
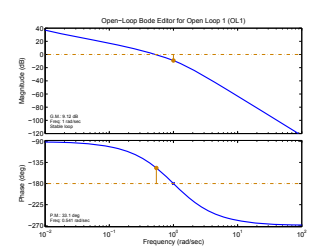
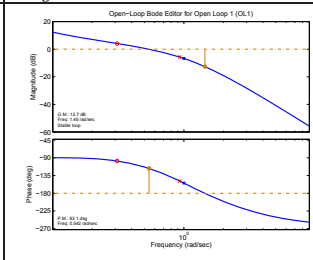
Illustration The following figures use a simple system

$$G = \frac{1}{s(s+1)^2} \quad (4)$$

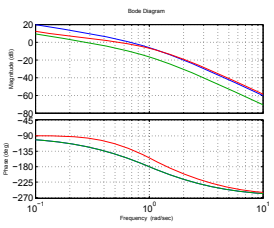
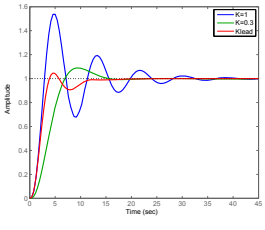
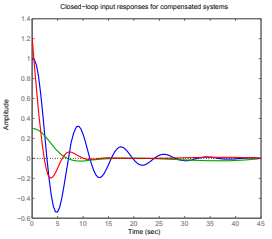
and show how easy this procedure is for a student on MATLAB while giving them high quality insight into frequency domain analysis and design. The MATLAB code required to get started is two simple lines:

[2] One weakness of SISOTOOL is its inability to store and display more than one design at a time.
[3] You can move the gain plot up and down manually and hence achieve a desired phase margin without computation.

G=tf(1,[1 2 1 0]);
sisotool(G);
Students now implement the mechanistic design rules within sisotool to generate the plots and values below.

	
Nominal system with $K = 1$. Poor margins.	Gain design with $K = 0.3$. $w_g = 0.27$. Good margins.
	
Nominal system with $K = 0.7$. Phase margin=33, $w_l = 0.54$	$K_{lead} = 0.7\sqrt{3}\frac{s+0.54/\sqrt{3}}{s+0.54\sqrt{3}}$ Phase margin =63.

Finally students should overlay the Bode and time responses of the lead, gain and uncompensated systems to give an overview. Students export K_{lead} back to the command window and then use two more simple MATLAB statements to do this:
*bode(G,G*0.3,G*Klead);*
overlaymany(G,1,0.3,Klead);
The latter of these files is a simple function file written by the author to make comparison of numerous compensator alternatives easy for students without the need for clumsy coding OER (2010).

	
Bode diagrams	Output responses
	Plots for uncompensated ($K = 1$ in blue), simple gain (green) and lead design (red).
Input responses	

It is clear that students can compare attributes of each scheme easily from these plots and moreover experiment

with several different systems. The insight gained can be assessed in a written exam with no computer available.

3.5 Summary

The key observation here is that the MATLAB code required for students to do all the key computations for classical control is largely trivial and therefore, getting to grips with MATLAB should not be an obstacle. Instead, students who make an effort to use the MATLAB tools find that their learning of classical control becomes much more real and concepts easier to understand. Students can experiment quickly and easily and thus test their understanding of both concepts and simple procedures. In summary, the key factor is that students are active and not passive and this is what learning theory says we need to aim for.

There are several other GUIs and scripts provided but not mentioned above and the interested reader is referred to the website to view these OER (2010).

4. PREDICTIVE CONTROL ILLUSTRATIONS

The open nature of MATLAB means it is easy to set up script and function files which are transparent to the USER. With predictive control the author has assumed MSc level competence and therefore not worried too much about writing code that is in a GUI form. Instead he expects sufficient competence from students to write their own script files to experiment with what is provided. Some examples are given next - again the reader is referred to OER (2010) for the complete set of code and instruction sheet provided.

4.1 Prediction with CARIMA models

One of the topics within MPC that students struggle to get their heads around is how to predict and the structure of the predictions. The author runs one laboratory where students need to generate predictions with MATLAB using the function files provided. Predictions typically take the form:

$$y_{fut} = H\Delta u_{fut} + P\Delta u_{past} + Qy_{past} \qquad (5)$$

where $H = C_A^{-1}C_b$, $C_A^{-1}H_b, C_A^{-1}H_A$. Without going through all these definitions in fine detail, students need to understand how matrices C_A, C_b, H_A, H_b are defined and to validate the answers they get.

Two files are provided: *caha.m and mpc_predmat.m*, one of which forms H, P, Q directly and one which allows the user to create these themselves using the interim steps. Students have a complete set of 'correct' data within the notes and are asked to reproduce these using MATLAB.

Other files are also available for state space model prediction and predictions with a T-filter for those who want to go to the next level of complexity.

4.2 Understanding GPC tuning

A big part of the MPC course is getting students to understand the impact of the different 'tuning' parameters on the closed-loop behaviour, and ironically, convincing students that what are viewed as tuning parameters are really degrees of freedom which need to be selected wisely to ensure that you get a sensible result from your MPC optimisation.

The best way to convince students of this and to give them an opportunity to own this knowledge via experience, that is experience of doing numerous MPC designs and seeing for themselves the impact of different changes. Typical parameters students may wish to investigate are the input and output horizons n_u, n_y, the control weighting λ, the choice of T-filter, the magnitude of the constraints and even of course the underlying dynamics of the process.

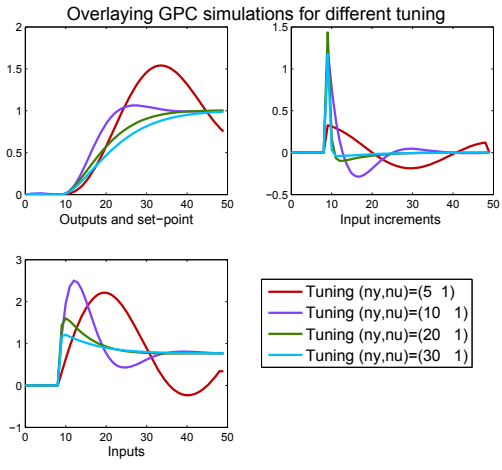
Consequently the author supplies *example*.m* script files which simply define all the data required to run a simulation and then, run the simulation and produce plots. For example the first few lines of one file are given next:

```
% Constraints
umax=1;
umin=-1;
Dumax=.5;
% Model
A=[1 -1.8 .81];
B=[1,.3]/100;
% Tuning parameters
lambda =1;
ny=15;
nu=3;
Tfilt=[1 -0.8];
```

The aim is that students can easily edit any data and then re-run the simulation to see the impact. Once students are comfortable with the basics, other code is provided to make it easy for students to overlay the results of different stragies, for instance code from within the file *overlayvarioustuning.m* looks as follows:

```
% Enter tuning parameters
% Ensure all definitions have same number of terms
% Suggest no more than 6
NY=[5,10,20,30];
NU=[1 1 1 1];
lambda={Wu,Wu,Wu,Wu}; % notice a cell array
```

A loop is then used to find all the closed-loop responses and produce a plot (as seen below).



Again, the emphasis is on providing tools that are transparent to the 'informed' user and quick to manipulate into the form the user requires to enhance their learning.

4.3 Other predictive control strategies

The author provides other files, including some guis, for students to experiment with dual-mode MPC strategies, independent model prediction and more. The interested reader is referred to OER (2010).

5. STUDENT PERCEPTIONS

On the whole, students are highly appreciative of the way MATLAB has been integrated into the curriculum Rossiter (2006), though it must be said it takes a few until the end of semester to reach this conclusion. For example, in December 2009, a survey of a class of about 100 (in attendance that day) 0% disagreed with the requirement for them to learn and use MATLAB and only 5% thought it was difficult to learn.

Historical and ongoing student comments are that the assessed laboratory, with implicit requirement to use MATLAB, is the part of the module that has really helped them see how everything fits together; the best part of the module! Some other comments include: *(i) Matlab - easy if you use logical steps and the reference material available. (ii) I think it was a good idea to use computer lab to teach about Matlab instead of lectures. It is more practical and interesting; (iii) Found it quite difficult at first, but then realised I have to work on my own to really master Matlab; (iv) I think the MATLAB assignment is a very good idea and think it is a shame that the module does't focus more on this. I find doing something with a pen and paper when MATLAB can do a much better job and quicker fairly tedious, although I do appreciate that some background is required.*

6. CONCLUSIONS

This paper had focussed on giving a description of how MATLAB tools are used in education to support learning of control analysis and design. From the perspective of the academic a few observations are key:

- MATLAB is widely used in industry and thus it is easy to convince colleagues and students of the need to engage.
- MATLAB is widely available within the University network and cheap to purchase as a student.
- The interface and general ease of use makes it easy and efficient for staff to produce quality learning resources to enhance the student learning experience.
- Once students get over their generic fear of 'code', the transparency and simplicity of MATLAB code makes it easy for them to use and solve difficult problems. This opens up and brings alive Gallop et al. (2005); Khan and Vlacic (2006) parts of the curriculum that previously were inaccessible due to the numerical requirements.

Although clearly the author is very positive about MATLAB as a teaching tool for control design, it is also necessary to be honest about some of the problems encountered and alternatives.

- **Expense:** In order to use MATLAB with large classes (circa 200) it is necessary to have about 100 licenses with relevant toolboxes. This is not cheap and is beyond the finances of a single department.
- **Student version or full version:** Although the student version is cheaper, many staff and research students need the full version and thus University are caught in a trap of eithering having 2 separate licenses with limited numbers on each or just paying for the full version and trying to get a few more licenses; the full version is too expensive to provide sufficient licenses for all students. Having just a student version, which is cheaper and inludes the key toolboxes needed by undergraduates, is not viable as this does not support much of the research requirement.
- **Toolboxes provision:** These are licensed as an add on and it is expensive to have large numbers. Licensing is based on peak usage not average usage, whereas teaching has localised peaks of a week or two and then almost no usage for the remainder of the year. We have annual problems with there being insufficient toolbox licenses at assignment times, but overall usage does not justify increasing this number. This causes real stress to staff and students who have little flexibility due to the timetables.

There are other solutions available (e.g. Guzman et al. (2006)) which do not have the same licensing or cost implications. However, more expertise and therefore staff preparation time is currently required to produce these and thus there is not a clear cut choice to be made here.

REFERENCES

Becerra, V. (2008). Experiences at teaching advanced control at reading using matlab based computer assessment. In *IEEE Colloquium/workshop on Control Education and sharing education resources*.

CILASS (2005). *Centre for excellence in inquiry based learning in the arts and social sciences*. http://www.shef.ac.uk/cilass.

Gallop, R., Bell, V., and Barnes, S. (2005). Using online activities to encourage active learning. In *HEA annual conference*.

Guzman, J., Astrom, K., Dormido, S., Hagglund, T., and Piguet, Y. (2006). Interactive learning modules for pid control. In *Advances in Control Education*.

Khan, A. and Vlacic, L. (2006). Teaching control: benefits of animated tutorials from viewpoint of control students. In *Advances in Control Education*.

OER (2010). *Open educational resources*. http://controleducation.group.shef.ac.uk.

Race, P. (2005). *Making Learning Happen: A Guide for Post-Compulsory Education*. Sage (Paul Chapman Publications).

Rossiter, J. (2006). Blended learning: some case studies from control engineering. In *Advances in control education*.

Rossiter, J. (2007). *Special interest group in control education website*. http://controleducation.group.shef.ac.uk/.

Rossiter, J. (2010). Facilitating independent learning of control basics. In *Submitted to UKACC*.

Rossiter, J. and Gray, L. (2009). Curriculum design for learning in a systems engineering department. In *Blended Learning Conference*.