Interactive applications to explore the parametric space of multivariable controllers

Yves Piguet * **Roland Longchamp** **

* *Calerga Sàrl, Av. de la Chablière 35, 1004 Lausanne, Switzerland (e-mail: yves.piguet@calerga.com).*
** *Laboratoire d'automatique, EPFL, 1015 Lausanne, Switzerland (e-mail: roland.longchamp@epfl.ch).*

Abstract: This paper describes interactive applications used to illustrate an advanced course on multivariable control. The direct manipulation of features in graphics, such as eigenvectors in the phase plane or observer eigenvalues in the complex plane, gives the student the means to explore not only the effect of parameters in the ranges usually considered as realistic, but also to observe the consequences of more extreme choices and get a better understanding of the design methods. The applications are implemented in Sysquake to provide a simple user interface, consistent across a wide range of examples from classical to advanced control.

Keywords: Interactive programs, computer graphics, teaching, automatic control engineering.

1. INTRODUCTION

In automatic control in particular, experience shows the importance of software applications as a complement to courses, books and practice with laboratory experiments (Guzmán et al. (2006)). At the Swiss Federal Institute of Technology in Lausanne (EPFL), after a mandatory course for undergraduate students of several disciplines, an elective course on advanced control is offered to master students. It covers polynomial controllers designed with pole placement, adaptive control, and multivariable control.

A new edition of the textbook the whole course is based on was published in September 2010 (Longchamp (2010)). It includes a CD-ROM with 64 interactive applications for Windows and Mac OS X, which have three purposes (Piguet and Longchamp (2006)):

- Reproduce textbook examples. Each application is an enhanced version of the static figures printed in the book. The student can observe the effect of each parameter by manipulating it with the mouse.
- Explore examples in depth. The description of textbook examples cannot cover all aspects of a problem. A lot can be learned by exploring the problem for parameters out of the range of what could be considered as "sensible" or by displaying optional graphics.
- Solve other problems. The models corresponding to the textbook examples can be changed to help students understand their homework exercises or their lab projects.

New interactive applications have been developed to complement the material introduced in the new edition (some of them can be found at `http://la.epfl.ch/tp` or `www.calerga.com/contrib/cnsd`). They preserve the principles applied for the previous edition and validated by positive student feedback: a simple user interface consistent across all the applications (notations, colors, menus); applications created with Sysquake (Piguet (2007)), an interactive scientific application with a programming language compatible with Matlab.

The remainder of the paper describes two important topics for multivariable control, phase plane in Section 2 and feedback design using eigenvalue assignment or LQR (Linear Quadratic Regulator) in Section 3. The purpose is not to describe in depth the theory, but rather to explain why they are relevant to the students and how software applications can enhance the learning process.

2. PHASE PLANE

Single-input single-output linear time-invariant systems are often represented by transfer functions, in the Laplace domain for continuous time and in the z domain for discrete time, from an input signal to an output signal. Their time response is commonly represented by the output as a function of time for a typical input, such as a step or a ramp.

Multivariable systems (systems with more than one input and one output) are usually described by state-space models. For linear time-invariant discrete-time systems without algebraic constraints, the model is a set of first-order ordinary difference equations which can be written in matrix form:

$$x(k+1) = Ax(k) + Bu(k) \tag{1}$$

$$y(k) = Cx(k) + Du(k) \tag{2}$$

where $x(k)$ is the state vector at time sample k, $u(k)$ the system input vector, $y(k)$ the output vector, and A, B, C and D are constant matrices. Stability and damping are determined by matrix A only. Matrix B defines how inputs influence the system, and matrix C how the state affects

the output. Matrix D is zero for strictly causal systems (i.e. all physical systems).

Because of the crucial role of the state, its time response, independently of any input and output, is important to understand. The two-dimension case $x \in \mathbb{R}^2$ is interesting: it can be represented easily in a graphic, yet the understanding students can acquire is easily generalized to higher-order systems. *Phase plane* represents the trajectory of the state in the state plane. In discrete time, the trajectory is made of discrete points, starting from the initial state $x(0)$.

2.1 State trajectory

Let a discrete-time linear time-invariant state-space model without input or output, defined by $x(k+1) = A\,x(k)$. The state trajectory can be computed iteratively. An explicit expression is given by

$$x(k) = \sum_{i=1}^{n} c_i \lambda_i^k v_i \tag{3}$$

where λ_i are the eigenvalues of A and v_i the corresponding eigenvectors, and c_i are constants defined by the initial conditions such that $x(0) = x_0$:

$$x_0 = \sum_{i=1}^{n} c_i v_i \tag{4}$$

This shows that the dynamical behavior of the system is directly related to the eigenvalues and eigenvectors of A.

The phase plane representation assumes $n = 2$ ($A \in \mathbb{R}^{2\times 2}$), but (3) and (4) are valid for any $n \in \mathbb{N}^\star$.

2.2 Application

An interactive application illustrates the kinds of state trajectories a system can exhibit in the phase plane.

The initial system is

$$A = \begin{bmatrix} 0.83 & -0.08 \\ -0.16 & 0.75 \end{bmatrix}$$

Its eigenvalues are $\lambda_1 = 0.91$ and $\lambda_2 = 0.67$ with corresponding eigenvectors $v_1 = [\,1\ -1\,]^{\mathrm{T}}$ and $v_2 = [\,1\ 2\,]^{\mathrm{T}}$. Fig. 1 shows the main window of the application, with matrix A which can be modified with sliders (a menu entry lets also the user enter it with Matlab's syntax), corresponding eigenvectors, and two graphics: one for eigenvalues in the complex plane, and one for the phase plane. In the phase plane, eigenvectors are shown as infinite straight lines, because their magnitude and sign is arbitrary and irrelevant. Small arrows show the vector field $Ax - x$. The discrete state trajectory, starting from the initial state, is tangent to this vector field. In addition to the state trajectory represented by circles, a continuous support line is obtained by replacing integer k with a real number in (3). Matching between eigenvalues and eigenvectors is shown by colors.

Figures containing elements which can be manipulated interactively with the mouse have a red frame. In the phase plane, the initial state can be moved anywhere. Since the

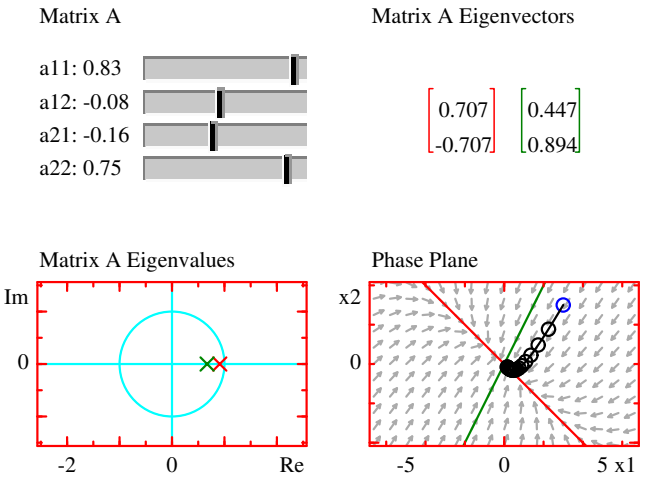

Fig. 1. Phase plane, 2nd-order system with stable real eigenvalues.

green (second) eigenvalue is better damped than the red one (it is smaller), the trajectory converges faster along the green eigenvector and is tangent to the red eigenvector for larger times. This remains true for any initial state.

Eigenvalues can also be manipulated on the real axis; a new matrix A is computed which keeps the same eigenvectors. When eigenvalues are close together, the trajectory goes more or less directly to zero. Perfectly identical eigenvalues are prevented, because they result in a discontinuity in the eigenvectors. For this reason, it is not possible to switch to the complex case without changing directly matrix A.

Fig. 2 shows how the trajectory is modified when one eigenvalue is moved beyond the stability limit 1 on the positive real axis. The two modes are clearly recognized: one of them remains the same, stable, and the trajectory is attracted to the line defined by the other eigenvector. This illustrates well (3) where only one of the eigenvalues λ_i is changed.

Another situation which can be easily discovered interactively is a negative real eigenvalue (Fig. 3). Since it causes a ringing effect which one usually wants to avoid, it is not often illustrated. But the student who reaches this case "by accident" can remember the relationship between the pole of a transfer function and the corresponding impulse response. These kinds of unexpected discoveries reinforce his or her understanding of modeling and control.

The last interactive elements are the eigenvectors which can be rotated around the origin. A new matrix A is computed which preserves the eigenvalues.

When eigenvectors become close, what happens? From (3), the problem is ill-conditioned and results in large values unless the initial state lies on the direction defined by the eigenvectors; see Fig. 4.

When A is defined to have complex eigenvalues, eigenvectors are complex, too, and cannot be represented in the phase plane. Trajectories become oscillatory, as illustrated in Fig. 5 where the eigenvalues are inside the unit disk (stable system).

Eigenvalues can be moved outside the unit circle, resulting in an oscillating, unstable trajectory. Fig. 6 shows the limit

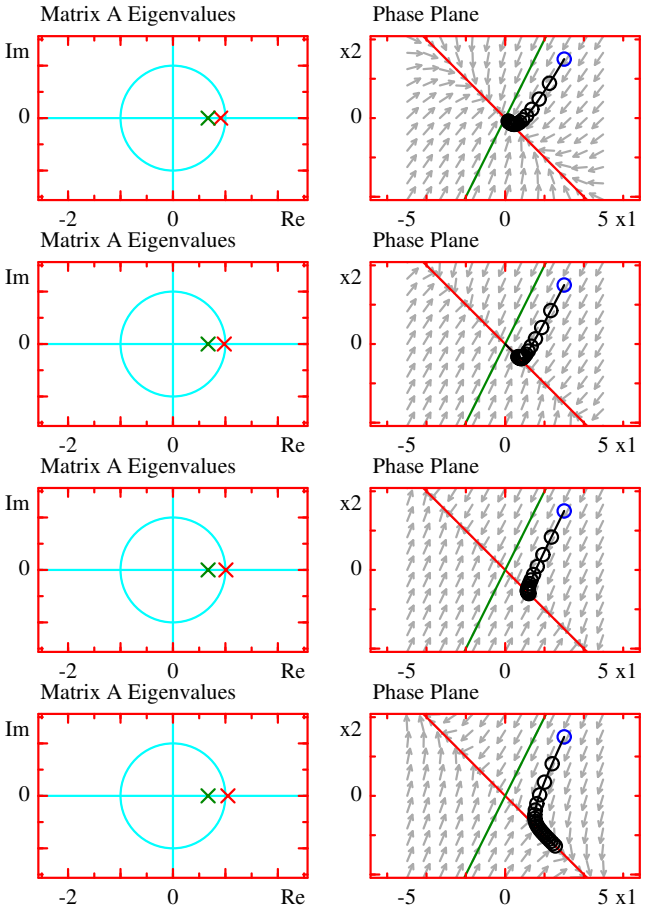

Fig. 2. Phase plane, 2nd-order system with real eigenvalues; one of the eigenvalues is moved from stable to unstable region.

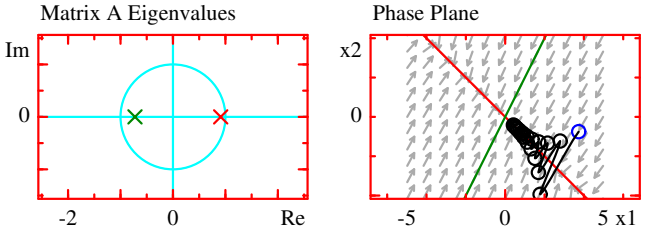

Fig. 3. Phase plane, 2nd-order system with a real negative eigenvalue.

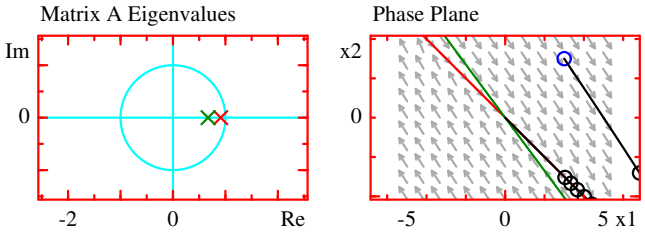

Fig. 4. Phase plane, 2nd-order system with stable real eigenvalues and eigenvectors almost aligned.

case of a marginally stable system. To help the user, when the mouse is close enough to the unit circle, eigenvalues are attracted to be placed exactly on it. The eigenvalue argument can still be set freely, showing how successive states are placed more or less closely on an ellipse.

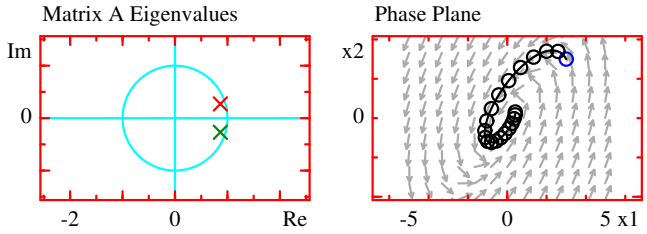

Fig. 5. Phase plane, 2nd-order system with stable complex conjugate eigenvalues.

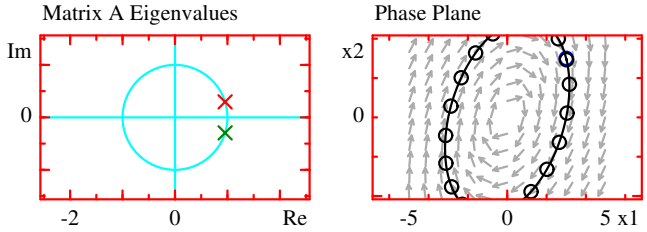

Fig. 6. Phase plane, 2nd-order system with marginally stable complex conjugate eigenvalues.

3. MULTIVARIABLE CONTROL

Since the dynamical behavior of a system depends mainly on its eigenvalues, a controller, in order to fulfill performance requirements, must move them to a suitable region in the complex plane. Only a feedback loop can achieve that. The simplest controller uses directly the state as its input. Two approaches are detailed in a standard course: eigenvalue placement, where one assigns directly the closed-loop eigenvalues so that absolute and relative damping specifications are met. And the Linear Quadratic Regulator (LQR), where a weighted quadratic function of states and system inputs is minimized.

In practice, the full state is usually not directly available. With output feedback, the output $y(k)$ of (2) is used with the system input to estimate the state with an *observer*. The observer itself adds its own dynamics and can also be designed with eigenvalue placement.

3.1 Eigenvalue assignment

Substituting state feedback $u(k) = -Lx(k)$ into (1) yields the the following closed-loop state-space equation:

$$x(k+1) = (A - BL)\, x(k)$$

The closed-loop dynamical behavior depends therefore on the eigenvalues of $A-BL$. For a single-input system, there is a unique state feedback vector L for a given choice of eigenvalues. L can be computed with the controllability matrix of the system or the Ackermann equation, among others.

Fig. 7 shows an application which illustrates eigenvalue placement for a state feedback controller. The discrete-time system represents a servomotor controlled in position. Closed-loop eigenvalues in the top-left figure can be moved anywhere in the complex plane (complex eigenvalues must form conjugate pairs). As a guide for the user, the region satisfying absolute and relative damping is displayed in red. The time responses of the input and state are displayed for a particular initial condition, which can also be changed.

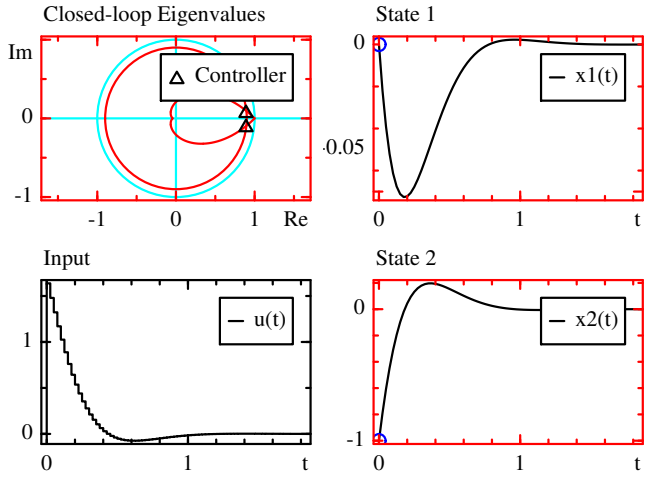

Fig. 7. Eigenvalue assignment for a two-state one-input state feedback controller.

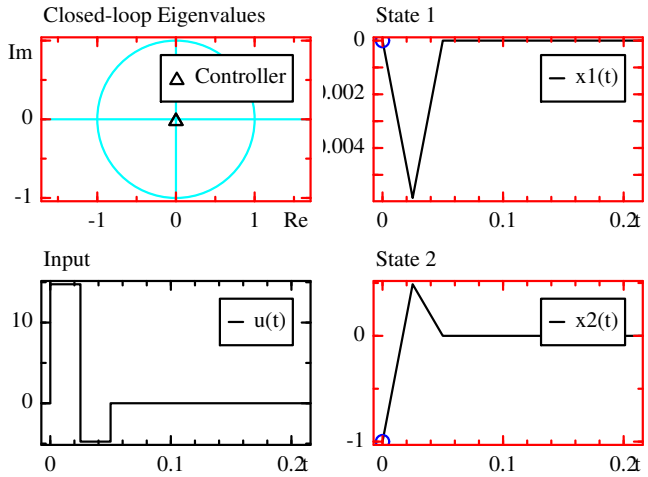

Fig. 8. Deadbeat controller (state feedback, eigenvalue assignment to 0).

Among the controller properties the student can observe, bringing the eigenvalues closer to 0, which makes the closed-loop system faster, increases the amplitude of the input. The limit case is the deadbeat controller with all closed-loop eigenvalues placed at 0 (Fig. 8).

3.2 Observer

When the full state cannot be measured directly, it must be estimated with an observer. The observer is basically a simulated version of the system where the measured output is used to force the estimated state to converge to the same value as the true system state.

The estimated state $\hat{x}(k)$ is given by

$$\hat{x}(k+1) = (A - KC)\hat{x}(k) + Bu(k) + Ky(k)$$

where K is the observer gain. Like the state feedback gain, it can be calculated by setting the eigenvalues of $A - KC$.

Fig. 9 shows eigenvalue assignment for a state observer. Eigenvalues, represented by green triangles in the complex plane, can be moved interactively. While observer eigenvalues are usually chosen to be well damped to provide a good state estimate for the state feedback controller, the student is free to explore other values, such as the complex conjugate poorly-damped pair illustrated in Fig. 10.

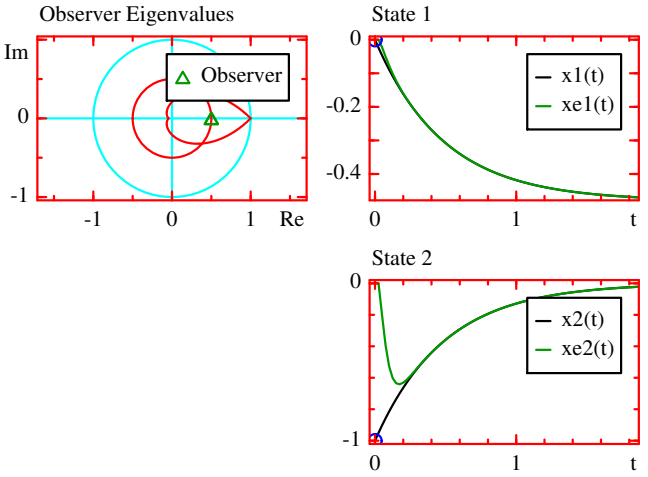

Fig. 9. Eigenvalue assignment for a well-damped observer ("xe" is the estimated state $\hat{x}$).

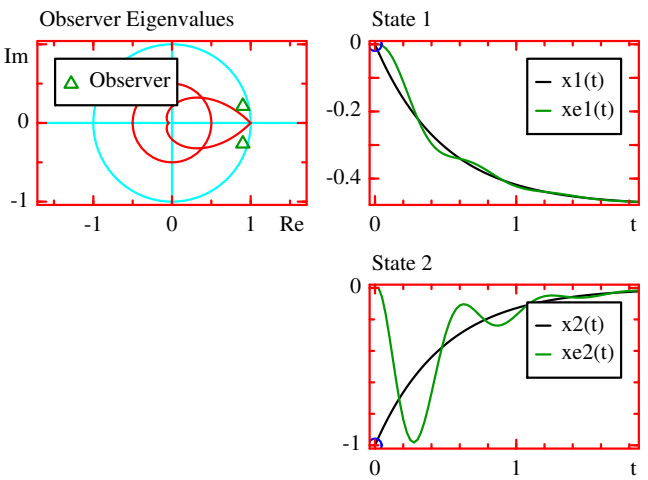

Fig. 10. Eigenvalue assignment for a poorly damped oscillating observer ("xe" is the estimated state $\hat{x}$).

3.3 Output feedback

The complete output controller is obtained by combining a state feedback controller with an observer. The estimated state is used in lieu of the system state to compute the control signal $u(k) = -L\hat{x}(k)$. The whole closed-loop system can be represented as a state-space model of order $2n$:

$$\begin{bmatrix} x(k+1) \\ \hat{x}(k+1) \end{bmatrix} = \begin{bmatrix} A & -BL \\ KC & A - KC - BL \end{bmatrix} \begin{bmatrix} x(k) \\ \hat{x}(k) \end{bmatrix}$$

An application where the eigenvalues of the observer and the state feedback can be placed interactively in the complex plane is shown in Fig. 11.

It should be noted that the closed-loop eigenvalues as design parameters play the same role as the closed-loop poles of a polynomial single-input single-output controller, and the observer eigenvalues correspond to the poles of the observer polynomial. The application is very similar to the one which illustrates pole placement (Fig. 12). The student sees how different approaches are closely related.

3.4 Linear Quadratic Regulator (LQR)

The LQR controller is obtained by minimizing the cost function J defined as

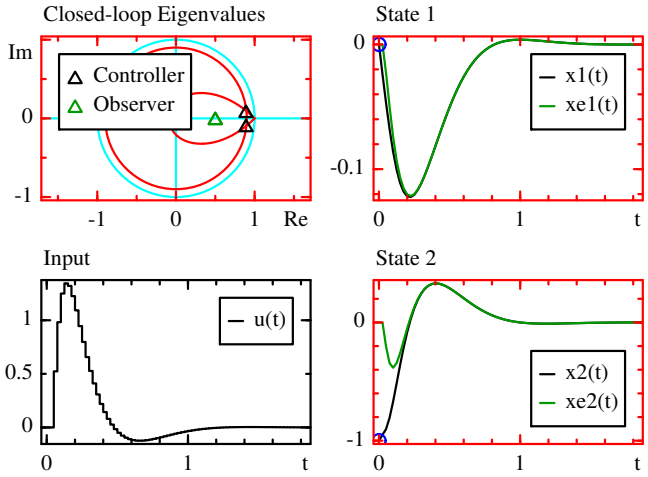

Fig. 11. Assignment of state feedback and observer eigenvalues ("xe" is the estimated state $\hat{x}$).

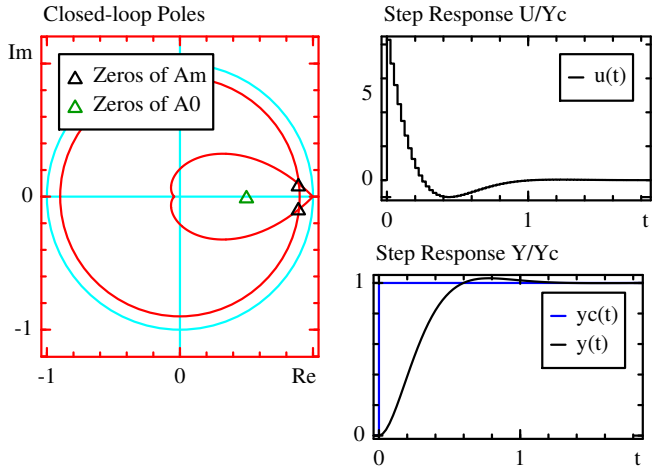

Fig. 12. Pole placement for a polynomial controller.

$$J = \frac{1}{2}x^{\mathrm{T}}(N)Sx(N) + \frac{1}{2}\sum_{k=0}^{N-1}\left(x^{\mathrm{T}}(k)Qx(k) + u^{\mathrm{T}}(k)Ru(k)\right) \quad (5)$$

where N is the control horizon; S, Q are symmetric positive semi-definite weighting matrices that penalize the state at the terminal time N and the state before terminal time, respectively; and R is a symmetric positive definite weighting matrix that penalizes the system input. Weighting matrices S, Q and R determine a compromise between the damping of the state and the magnitude of the control.

With a fixed starting time, the result is a linear time-variant feedback gain $L(k)$. In practice, a simplification is often made with an infinite control horizon, where the cost function is

$$J = \frac{1}{2}\sum_{k=0}^{\infty}\left(x^{\mathrm{T}}(k)Qx(k) + u^{\mathrm{T}}(k)Ru(k)\right)$$

Under mild conditions, an infinite control horizon ensures a stable closed-loop system. The controller is obtained by solving the discrete-time algebraic Riccati equation

$$P = A^{\mathrm{T}}PA - A^{\mathrm{T}}PB\left(B^{\mathrm{T}}PB + R\right)^{-1}B^{\mathrm{T}}PA + Q$$

The state feedback gain L is given by

$$L = \left(B^{\mathrm{T}}PB + R\right)^{-1}B^{\mathrm{T}}PA$$

Sometimes, only the first control signal $u(0)$ of the solution which minimizes (5) is used and remaining values are discarded; at the next sampling time, all the values are shifted by one sampling period (receding horizon control). The result is a time-invariant state feedback gain L which depends only on the model A, B and C, the weighting matrices Q, R and S, and the control horizon N:

$$P_N = S$$

$$P_{k-1} = A^{\mathrm{T}}\left(P_k - P_kB\left(R + B^{\mathrm{T}}P_kB\right)^{-1}B^{\mathrm{T}}P_k\right)A + Q$$

$$L = \left(B^{\mathrm{T}}P_1B + R\right)^{-1}B^{\mathrm{T}}P_1A$$

3.5 Applications

Two interactive applications illustrates the LQR design and the role of the weights, one for infinite control horizon, and one for receding horizon. As a simplification, weighting matrices are diagonal (correlation of state and input vectors would make the user interface needlessly complicated).

Fig. 13 shows an LQR controller for a system with $x \in \mathbb{R}^2$ and $u \in \mathbb{R}^2$:

$$x(k+1) = \begin{bmatrix} 0.99 & 0 \\ 0 & 0.98 \end{bmatrix}x(k) + \begin{bmatrix} 0.995 & 0.995 \\ -0.248 & 0.743 \end{bmatrix}u(k)$$

Changing the weights, the following observations can be made:

- Increasing q_i makes the corresponding state $x_i(k)$ converge faster.
- Increasing r_i makes the corresponding input $u_i(k)$ smaller.
- In both cases, weights have also an effect on all other states and inputs.
- An R small relatively to Q makes the feedback gain L larger.
- The closed-loop system is stable for any choice of the weights.

Fig. 14 shows an LQR controller with receding horizon. The length of the control horizon can also be set with a slider. The infinite-horizon LQR with the same weighting matrices is also displayed to show the difference. It can be observed that a smaller control horizon has a less pronounced low-pass effect; consequently, the feedback gain is larger and the closed-loop response is faster.

Like with most of the applications, the student can replace the default system with others, including higher-order ones.

3.6 Discussion

Eigenvalue assignment and LQR are two synthesis methods for multivariable systems with different strengths. With eigenvalue assignment, design parameters are more directly related to the frequency- and time-domain performance of the closed-loop system. They are the equivalent of poles in the monovariable case, which student are familiar with.

LQR's design parameters are higher-level and offer a way to fine-tune the controller action on each state separately. It is also suitable to systems with multiple inputs.

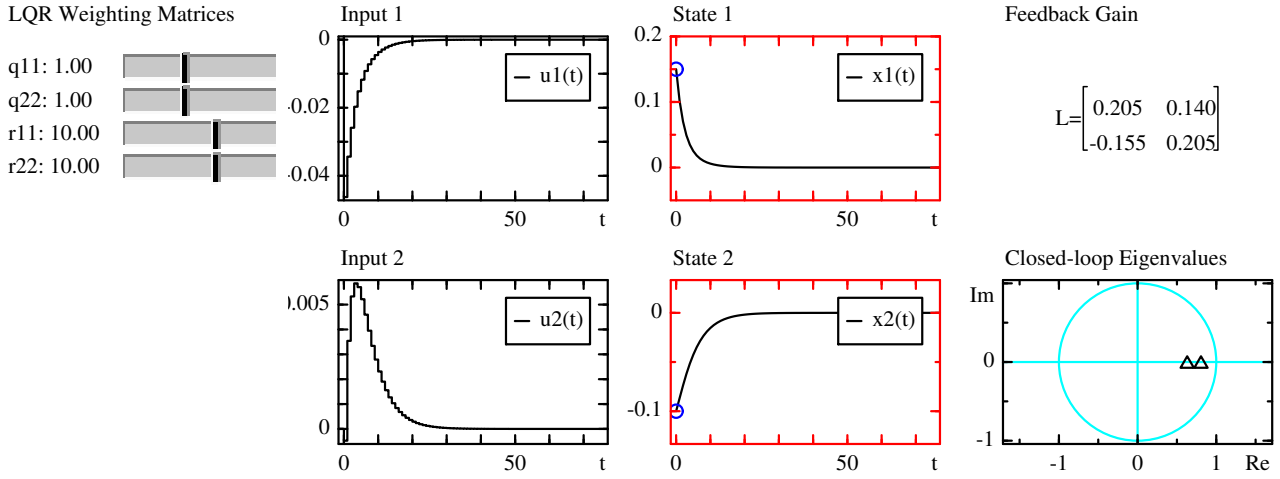

Fig. 13. LQR for a system with two states and two inputs.

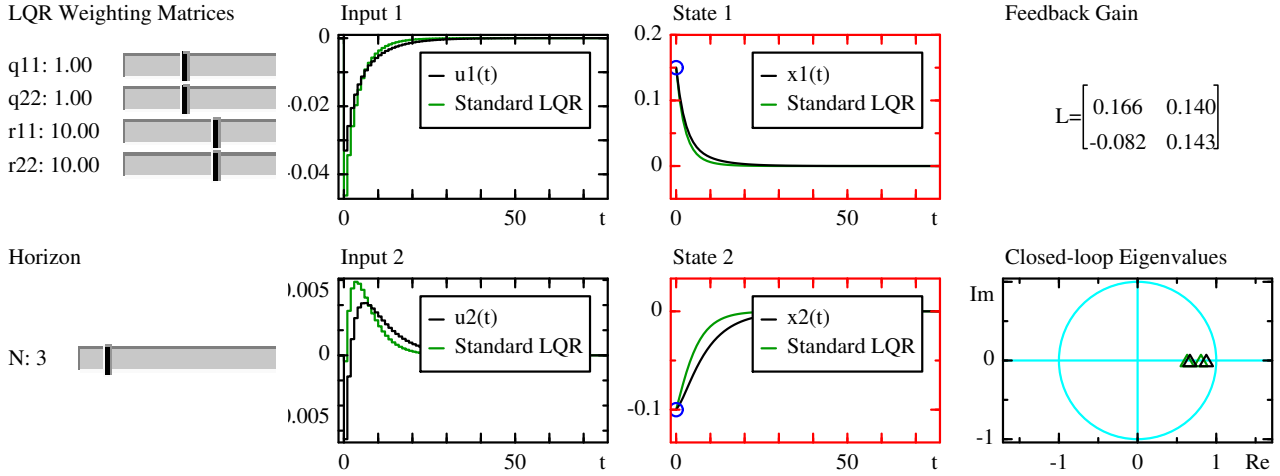

Fig. 14. LQR with receding horizon; the time responses and closed-loop eigenvalues for the infinite-horizon LQR controller based on the same weighting matrices are displayed in green.

In both cases, interactive applications allow the students to quickly grasp the effect of the controller. The user interface is very similar, which lets them concentrate on the meaning of the graphics. The LQR application also shows the eigenvalues in the complex plane; weighting matrices become just an alternative way of tuning the controller, but the same analysis tools remain.

4. CONCLUSION

Students need to acquire their own understanding, not only to be able to reproduce what they have been taught but to have a broader view of dynamical systems and their properties. Even if the course itself follows a well-defined progression, the process of knowledge consolidation is specific to each student. Control engineering provides different approaches — continuous-time and discrete-time, transfer functions and state-space models, poles or eigenvalues, frequency responses and time responses — which are not disjoint, but offer many opportunities to get insight into the field.

For instance, increasing feedback gain often leads to faster closed-loop responses, larger control signals, and a risk of overshoot, instability, and saturation. Is it still the case for multivariable systems? What can be learned for choosing tuning parameters?

Interactive applications let students answer these questions, and others, at their own pace.

Feedback from the students shows the the applications, well integrated with the textbook, are considered as a very valuable complement to understand theory and analyze other problems related to homework exercises and laboratory experiments.

REFERENCES

Guzmán, J.L., Åström, K., Dormido, S., Hägglund, T., Berenguel, M., and Piguet, Y. (2006). Interactive learning modules for PID control. *Control Systems Magazine, IEEE*, 28(5), 118–134.

Longchamp, R. (2010). *Commande numérique de systèmes dynamiques.* Presses polytechniques et universitaires romandes, Lausanne, Switzerland, 3rd edition.

Piguet, Y. (2007). *Sysquake User Manual.* Calerga, Lausanne, Switzerland, 5th edition.

Piguet, Y. and Longchamp, R. (2006). Interactive applications in a mandatory control course. In *Proc. of 7th IFAC Symposium on Advances in Control Education*, 11–22. Madrid, Spain.