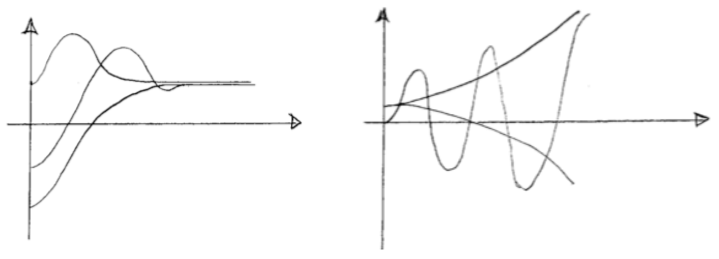
2.3 A few things on stability

As mentioned in the introduction, the need for feedback control comes our desire to have some kind of consistency in the obtained behavior. This is what is referred to as dynamic stability, or stability for short. There are quite a few ways to define stability for dynamical systems. In the present case, we will consider stability with respect to changes in initial conditions.

Intuitively, stability with respect to initial conditions, which we will simply call here stability, can be represented graphically by the figure 2.3a below.



(a) Stable case (b) Unstable case

Figure 2.3: Stable vs. Unstable

In the above figure, the horizontal axis is the time, and the vertical one is the amplitude of a considered variable, say output $y(t)$. In this case, stable will mean that, taking several signals $y(t)$ initialized differently, they will all eventually reach the same point after a transient. Obviously, a contrario, the other figure 2.3b would represent an unstable system.

Stability and independence with respect to initial conditions can also be examined mathematically through the simple example that follows.

Example: First-order linear scalar system

Consider the basic first-order linear differential equation

$$\dot{x}(t) = -x(t), \qquad x(0) = x_0 \tag{2.8}$$

To check that the system is stable, we just compute its solution given by

$$x(t) = x_0 e^{-t} \tag{2.9}$$

and remark that, when $t \to \infty$, the "final" behavior of the system is

$$\lim_{t\to\infty} x(t) = \lim_{t\to\infty} x_0 e^{-t} = 0 \tag{2.10}$$

and this no matter what the initial condition of the system is! Hence this system is obviously stable. □

Many systems are not naturally stable (think of the upright position of a pendulum for example). In this case, one can render or make this system stable by using information on the state to send corrections on the control input so that the system in closed loop is stable. The controller that does that is called feedback stabilizer or feedback controller (see figure 2.4 below).

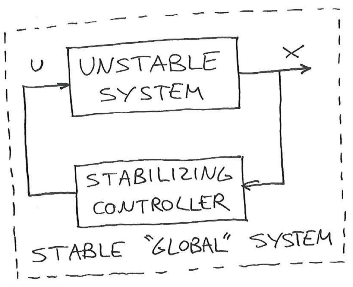


Figure 2.4: Stabilizing an unstable plant

Let us now consider another scalar system, albeit an unstable one, and see how it can be stabilized by feedback.

Example: Stabilizing an unstable scalar system

Consider the system modelled by

$$\dot{x}(t) = x(t) + u(t), \qquad x(0) = x_0 \tag{2.11}$$

Without control input, ie when

$$u(t) = 0, \qquad \text{we have} \qquad x(t) = x_0 e^t \tag{2.12}$$

which means that,

$$\begin{cases} \text{if} \quad x_0 > 0 \Rightarrow \lim_{t\to\infty} x(t) = +\infty \\ \text{if} \quad x_0 = 0 \Rightarrow \lim_{t\to\infty} x(t) = 0 \\ \text{if} \quad x_0 < 0 \Rightarrow \lim_{t\to\infty} x(t) = -\infty \end{cases} \tag{2.13}$$

that is the final behavior is hightly dependent on the initial condition, implying that the system is not stable.

The basic idea behind stabilization by feedback consists in choosing a control expression u (also called *control law*) as a function of the state x so that the global system, also called *system in closed-loop*, is stable.
Indeed, let

$$u(t) = -2x(t) \tag{2.14}$$

Then, replace signal $u(t)$ in (2.11) with control law (2.14), which gives

$$\dot{x}(t) = x(t) - 2x(t) = -x(t) \tag{2.15}$$

so that the system in closed-loop is stable. Hence the simple linear controller (2.14) with *proportional gain* 2 was able to stabilize system (2.11). □

2.3.1 Stability of Linear Time-Invariant systems

What we would like is a general tool/criterion to assess whether a system represented by a linear state-space representation is stable or not.
Before doing so, let us brush up on a stability result used in classical control theory.

A stability result for Transfer Functions

Consider a linear single-input single-output (SISO) system modelled by the transfer function

$$\frac{y(s)}{u(s)} = \frac{b_m s^m + ... + b_1 s + b_0}{s^n + a_{n-1}s^{n-1} + ... + a_1 s + a_0} \tag{2.16}$$

where the a_i and b_j parameters are constant, with $m \leq n$.

An important result from classical control theory states that **system (2.16) is stable if and only if each root of the denominator $a(s)$ (called *characteristic polynomial*) has a strictly negative real part**.

Remark that this result is meant for an input-output notion of stability, and therefore that initial conditions are not explicitly considered. In addition, in the way this result is stated, it requires the presence of an input signal, which, as we have seen is not always the case (think of a state-space representation with the matrix $\mathbf{B} = 0$).

Stability criterion for non-controlled state-space representations

Consider the system modelled by

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t), \qquad \mathbf{x}(0) = \mathbf{x}_0 \tag{2.17}$$

Similarly to the scalar cases we have seen in section 2.1, we want to have a simple criterion allowing us to conclude on the stability of system (2.17) by considering only matrix $\mathbf{A}$.

A simple but neat way to obtain such a criterion consists first in applying the Laplace transform on system (2.17) so that we get

$$s.\mathbf{x}(s) - \mathbf{x}_0 = \mathbf{A}\mathbf{x}(s) \Rightarrow (s\mathbf{I} - \mathbf{A})\mathbf{x}(s) = \mathbf{x}_0 \tag{2.18}$$

where $\mathbf{x}(s)$ is the Laplace transform of $\mathbf{x}(t)$ and $\mathbf{I}$ is the identity matrix.

From there, we have

$$\mathbf{x}(s) = (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{x}_0 \tag{2.19}$$

and if we rewrite the inverse of matrix $s\mathbf{I}-\mathbf{A}$ in terms of how it can be computed, we get

$$\mathbf{x}(s) = \frac{\text{adj}(s\mathbf{I} - \mathbf{A})}{\det(s\mathbf{I} - \mathbf{A})}\mathbf{x}_0 \tag{2.20}$$

Note that, under this form, expression (2.20) looks a lot like the transfer function expression (2.16). Indeed, in this case, $\mathbf{x}(s)$ in (2.20) corresponds to $y(s)$

in (2.16) while x_0 plays in (2.20) the same role as input $u(s)$ in (2.16). Interestingly, a constant like $\mathbf{x}_0$ in the Laplace domain is an impulse of amplitude $\mathbf{x}_0$ in the time domain.

Now we know from section 2.3.1 that transfer function (2.20) is stable (for all inputs and therefore for the impulse of amplitude x_0) if and only if each root of the denominator of (2.20), ie the so-called *characteristic polynomial*, $\det(s\mathbf{I}-\mathbf{A})$ has its real part strictly negative, or, using other words, if each solution λ of the equation $\det(\lambda\mathbf{I}-\mathbf{A}) = 0$ has a strictly negative real part.

Recall from maths: The solutions λ of the equation $\det(\lambda\mathbf{I}-\mathbf{A}) = 0$ are called the *eigenvalues* of matrix $\mathbf{A}$.

The above means that, in order to assess the stability of system $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}$, one can directly compute the eigenvalues of matrix $\mathbf{A}$ without having to compute the corresponding transfer function.
This finally leads to the following result.

Main result: The system represented by $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}(t)$ is stable if and only if the real part of each eigenvalue of $\mathbf{A}$ is strictly negative. □

Example: Two-dimensional system

Consider the following system

$$\dot{\mathbf{x}} = \begin{bmatrix} -1 & 0 \\ 1 & -1 \end{bmatrix} \mathbf{x} \tag{2.21}$$

Computing the eigenvalues of matrix $\mathbf{A}$ starts by calculating

$$\begin{aligned} \det(\lambda\mathbf{I}-\mathbf{A}) &= \det\left(\begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} - \begin{bmatrix} -1 & 0 \\ 1 & -1 \end{bmatrix}\right) \\ &= \det\left(\begin{bmatrix} \lambda+1 & 0 \\ -1 & \lambda+1 \end{bmatrix}\right) \\ &= (\lambda+1)^2 = 0 \end{aligned} \tag{2.22}$$

From equation (2.22), we can see that we have a double eigenvalue $\lambda = -1$, whose real part is strictly negative. Hence the system (2.21) is stable.
Take now another system:

$$\dot{\mathbf{x}} = \begin{bmatrix} -1 & -1 \\ 1 & -1 \end{bmatrix} \mathbf{x} \tag{2.23}$$

The eigenvalues of matrix $\mathbf{A}$ are given this time by $\lambda_1 = -1 + i$ and $\lambda_2 = -1 - i$. Both eigenvalues have a strictly negative real part, the system again is stable. □

2.4 Stabilization of LTI systems by state feedback

Consider a linear system with inputs $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$. As we have seen in the previous section, if not all the eigenvalues of $\mathbf{A}$ have their real part strictly negative, then the system is not stable. Then, similarly to our discussion of in the introduction, we can use the state $\mathbf{x}(t)$ as information fed back to the control input, and to define a controller of the form

$$\mathbf{u}(t) = -\mathbf{K}.\mathbf{x}(t) \tag{2.24}$$

where, in a general multi-input context, we would have

$$\mathbf{u}(t) \in \mathbb{R}^m, \qquad \mathbf{x}(t) \in \mathbb{R}^n \tag{2.25}$$

which implies, for matrix $\mathbf{K}$:

$$\mathbf{K} \in \mathbb{R}^{m \times n} \tag{2.26}$$

The controller or control law (2.24) is called a *state-feedback controller.*
If we put back the expression of this controller into the state-space representation and instead of the control input $\mathbf{u}(t)$, we have the dynamics of the closed-loop system written as

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}(-\mathbf{K}\mathbf{x}(t)) \tag{2.27}$$

or

$$\dot{\mathbf{x}}(t) = (\mathbf{A} - \mathbf{B}\mathbf{K})\mathbf{x}(t) \tag{2.28}$$

Clearly the so-called *closed-loop dynamics* (2.28) is stable if each eigenvalue of $\mathbf{A} - \mathbf{B}\mathbf{K}$ has a strictly negative real part.
Hence, in this case, *the role of the control engineer is to pick values of matrix* $\mathbf{K}$ *such that its closed-loop dynamics is stable*!

Example: A multiple-input system

Consider the two-input system represented by

$$\dot{\mathbf{x}} = \begin{bmatrix} 2 & 0 & 0 \\ 1 & -1 & 2 \\ 0 & 2 & -1 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 1 & 0 \\ 0 & 2 \\ 0 & 0 \end{bmatrix} \mathbf{u} \tag{2.29}$$

and assume that we would like to stabilize this system by state-feedback (first, one can also check that this system is not stable when $\mathbf{u} = 0$, which it is not).
Since $\dim(\mathbf{x}) = 3$ and $\dim(\mathbf{u}) = 2$, we have that $\mathbf{K} \in \mathbb{R}^{2\times 3}$.
Let us rewrite (2.29) in component-form as follows

$$\begin{cases} \dot{x}_1 = 2x_1 + u_1 \\ \dot{x}_2 = x_1 - x_2 + 2x_3 + 2u_2 \\ \dot{x}_3 = 2x_2 - x_3 \end{cases} \tag{2.30}$$

A possible control law (but there are many other possibilities) could be $u_1 = -3x_1$ and $u_2 = -x_3$. This would give the closed-loop dynamics

$$\begin{cases} \dot{x}_1 = -x_1 \\ \dot{x}_2 = x_1 - x_2 \\ \dot{x}_3 = 2x_2 - x_3 \end{cases} \tag{2.31}$$

or

$$\dot{\mathbf{x}} = \begin{bmatrix} -1 & 0 & 0 \\ 1 & -1 & 0 \\ 0 & 2 & -1 \end{bmatrix} \mathbf{x} \tag{2.32}$$

which is a cascade of 3 stable subsystems. Since the first system in x_1 is stable and is not influenced by anything else, than it will gradually go to 0. In the second subsystem, this also has the consequence that the term $x_1(t)$ will disappear, so that, combined with the stability of this subsystem, $x_2(t)$ will also go eventually to zero. Applying the same reasoning for the third and last subsystem, we can finally conclude that the overall system is stable (one can also check the eigenvalues of closed-loop system (2.32)).
Hence the system was stabilized by state-feedback with the *gain matrix* **K** written as

$$\mathbf{K} = \begin{bmatrix} -3 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{2.33}$$

□

In the above example, we have proceeded by simple guessing and examination of the system subdynamics for choosing the values of gain matrix **K**. The effect of choosing these values could only be checked *a posteriori* by stability assessment.
While this way of doing things might very well work in quite a few cases, it might not be so for all systems, and it would be interesting of have a more systematic method.
Luckily enough, such algorithms do exist, and there quite a few of them. One of these techniques, or a class thereof, is referred to as *pole placement* or *eigenvalue assignment*. Instead of picking some value(s) for **K** and checking afterwards whether the corresponding closed-loop dynamics is stable, one can also choose what eigenvalues we would like the system in closed-loop to have, i.e. we want

$$\boldsymbol{\lambda}_{cl} = \mathrm{eig}(\mathbf{A} - \mathbf{BK}) \tag{2.34}$$

where $\boldsymbol{\lambda}_{cl}$ is a vector containing all desired eigenvalues that the system in closed-loop should have. There are several algorithms allowing to compute the corresponding **K** given $\boldsymbol{\lambda}_{cl}$. The most well-known is probably the so-called Ackerman method. In Matlab, one simply type

```
K = acker(A,B,lambda_cl)
```
(2.35)

One can also use the command `place` in Matlab, which can be summoned in the same way as the Ackerman method. While the Ackerman results works fine for small systems, numerical issues can appear for larger ones. The `place` command usually works fine, as long as the desired eigenvalues are chosen differently from one another.

2.5 The Linear Quadratic Regulator (LQR)

An alternative to the eigenvalue assignment is the famous Linear Quadratic Regulator, or LQR for short. Indeed, instead of choosing desired eigenvalues and obtaining the corresponding gain **K**, we would like to find a control input vector $\mathbf{u}(t)$ defined on the interval $[0, T]$ such that the obtained $\mathbf{u}(t)$ minimizes the cost function

$$J = \int_0^T \left[\mathbf{x}^T(\tau)\mathbf{Q}\mathbf{x}(\tau) + \mathbf{u}^T(\tau)\mathbf{R}\mathbf{u}(\tau)\right] d\tau \tag{2.36}$$

where $\mathbf{Q}$, $\mathbf{R}$ are strictly positive and symmetric constant matrices used to tune the controller. Roughly speaking, these two matrices are used to balance between minizing the distance between the state $\mathbf{x}$ and the origin (through the quadratic form $\mathbf{x}^T\mathbf{Q}\mathbf{x}$, generalization of $q||\mathbf{x}||^2 = q\mathbf{x}^T\mathbf{x}$, with q a scalar), and minimizing the energy consumption (through the quadratic form $\mathbf{u}^T\mathbf{R}\mathbf{u}$, which, again, can be seen as a generalization of $||\mathbf{u}||^2 = \mathbf{u}^T\mathbf{u}$ multiplied with a weighting scalar).
Interestingly, when $T \to \infty$, ie. when J in (3.136) is changed into

$$J = \int_0^\infty \left[\mathbf{x}^T(\tau)\mathbf{Q}\mathbf{x}(\tau) + \mathbf{u}^T(\tau)\mathbf{R}\mathbf{u}(\tau)\right] d\tau, \tag{2.37}$$

then control signal $u(t)$ can be expressed as

$$\mathbf{u}(t) = -\mathbf{K}\mathbf{x}(t) \tag{2.38}$$

which is nothing but the state-feedback form we have seen in the previous section! Hence, the difference lies essentially in how the matrix $\mathbf{K}$ is obtained. In the case of the LQR controller, we have

$$\mathbf{K} = \mathbf{R}^{-1}\mathbf{B}^T\mathbf{P} \tag{2.39}$$

where symmetric and positive definite matrix $\mathbf{P}$ is the solution of the following algebraic matrix equation

$$\mathbf{P}\mathbf{A} + \mathbf{A}^T\mathbf{P} - \mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{P} + \mathbf{Q} = 0 \tag{2.40}$$

Equation (2.40) is known as the matrix Riccati equation. Note that, in Matlab, you do not have to write a program to solve equation (2.40). In order to obtain the gain matrix $\mathbf{K}$, you can just use the command `lqr`.
Similarly to the Ziegler-Nichols rule which was used to find initial values for tuning a PID controller, the Bryson's rule can be used to find how to obtain preliminary tunings for matrices $\mathbf{Q}$ and $\mathbf{R}$. The latter are diagonal matrices that will be set as

$$Q_{ii} = 1/\text{maximum acceptable value of } x_i^2, \tag{2.41}$$

with $i \in \{1, 2, ..., n\}$, and

$$R_{jj} = 1/\text{maximum acceptable value of } u_j^2, \tag{2.42}$$

with $j \in \{1, 2, ..., m\}$.
There is also a discrete-time version of the LQR controller. In this case, considering discrete-time linear state-space representation

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k), \tag{2.43}$$

cost function (2.37) is replaced with

$$J = \sum_0^\infty \left[\mathbf{x}^T(k)\mathbf{Q}\mathbf{x}(k) + \mathbf{u}^T(k)\mathbf{R}\mathbf{u}(k))\right]. \tag{2.44}$$

2.6 Stabilization of a linear system around an equilibrium point

So far, when stabilizing a linear system represented by $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$, we have mostly used the control law $\mathbf{u} = -\mathbf{K}\mathbf{x}$, meaning the state $\mathbf{x}(t)$ of the system in closed loop was converging to 0.
First, note that in doing so, we have implicitly stated that the equilibrium point of the system in closed loop is $\mathbf{x}^* = 0$. This is indeed the case, ie we have that

$$0 = \mathbf{A}'\mathbf{x}^* = (\mathbf{A} - \mathbf{B}\mathbf{K})\mathbf{x}^* \tag{2.45}$$

has the solution $\mathbf{x}^* = 0$ for *any* matrix $\mathbf{A}'$.

But what if we want to stabilize the system on another equilibrium point? As we have seen previously, for systems with inputs, equilibrium points $\mathbf{x}^*$ and their associated $\mathbf{u}^*$ should be the solution of the algebraic equation

$$0 = \mathbf{A}\mathbf{x}^* + \mathbf{B}\mathbf{u}^* \tag{2.46}$$

ie for some $\mathbf{x}^*$, there will be a corresponding control input $\mathbf{u}^*$. And since we want to stabilize around $\mathbf{x}^*$, let us define

$$\Delta\mathbf{x}(t) := \mathbf{x}(t) - \mathbf{x}^* \quad \text{and} \quad \Delta\mathbf{u}(t) := \mathbf{u}(t) - \mathbf{u}^* \tag{2.47}$$

where $\Delta\mathbf{x}(t)$ represents the "error" around $\mathbf{x}^*$. In order to stabilize the system, $\Delta\mathbf{u}(t)$ the control input changes around the nominal $\mathbf{u}^*$. In this case, what is important is to drive the error variable $\Delta\mathbf{x}(t)$ to 0. To see this, first compute its time-derivative:

$$\frac{d}{dt}(\Delta\mathbf{x}(t)) = \frac{d}{dt}(\mathbf{x}(t) - \mathbf{x}^*) = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{2.48}$$

and, since $\mathbf{A}\mathbf{x}^* + \mathbf{B}\mathbf{u}^* = 0$, we can also write

$$\frac{d}{dt}(\Delta\mathbf{x}(t)) = \mathbf{A}\mathbf{x} - \mathbf{A}\mathbf{x}^* + \mathbf{B}\mathbf{u} - \mathbf{B}\mathbf{u}^* \tag{2.49}$$

which finally gives

$$\frac{d}{dt}(\Delta\mathbf{x}(t)) = \mathbf{A}\Delta\mathbf{x}(t) + \mathbf{B}\Delta\mathbf{u}(t) \tag{2.50}$$

These dynamics are called *error dynamics*. Note that, in (2.50), if $\Delta\mathbf{u}(t) = 0$ (which also means if $\mathbf{u}(t) = \mathbf{u}^*$), then we have the corresponding equilibrium point $\Delta\mathbf{x}^* = 0$.
Then, similarly to the previous section, we can simply stabilize this system with the linear state-feedback controller

$$\Delta\mathbf{u}(t) = -\mathbf{K}.\Delta\mathbf{x}(t) \tag{2.51}$$

while the control input applied to the system, is also given by

$$\mathbf{u}(t) = -\mathbf{K}(\mathbf{x}(t) - \mathbf{x}^*) + \mathbf{u}^* \tag{2.52}$$

If we rewrite expression (2.52) slightly (and remove obvious time-dependencies), we get

$$\mathbf{u} = -\mathbf{K}\mathbf{x} + \mathbf{u}^* + \mathbf{K}\mathbf{x}^* \tag{2.53}$$

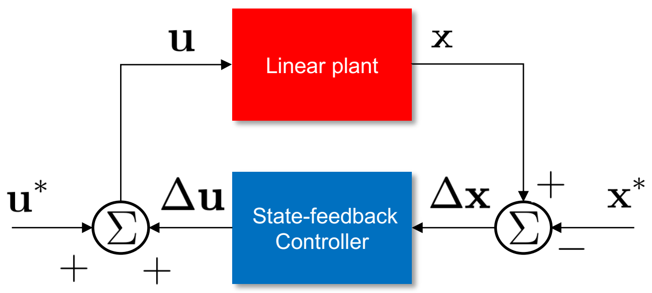


Figure 2.5: State-feedback controller for a linear system

where we can see that the first part of the RHS term of (2.53) is dedicated to feedback while the rest is a feedforward term. While this expression works very well, it would be of interest, instead of stabilizing the state around an equilibrium point, of making a chosen output $y(t)$ reach a constant reference signal r of same dimension as the output. In other words, we would like our state-feedback controller to take the form

$$\mathbf{u} = -\mathbf{K}\mathbf{x} + \bar{\mathbf{N}}\mathbf{r}. \tag{2.54}$$

To do so, start again with the equilibrium relation, to which we adjoin the output equation, ie we have (we assume here that $\mathbf{D} = 0$)

$$\begin{cases} 0 = \mathbf{A}\mathbf{x}^* + \mathbf{B}\mathbf{u}^* \\ \mathbf{y}^* = \mathbf{C}\mathbf{x}^* \end{cases} \tag{2.55}$$

where recall that we want to obtain

$$\mathbf{y}^* = \mathbf{r} \tag{2.56}$$

Rewrite then (2.55) in matrix form

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}^* \\ \mathbf{u}^* \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{r} \end{bmatrix} \tag{2.57}$$

which can be inverted as

$$\begin{bmatrix} \mathbf{x}^* \\ \mathbf{u}^* \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ \mathbf{r} \end{bmatrix}. \tag{2.58}$$

Define now the new matrix $\mathbf{N}$ defined as

$$\mathbf{N} := \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & 0 \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{N}_{11} & \mathbf{N}_x \\ \mathbf{N}_{21} & \mathbf{N}_u \end{bmatrix}, \tag{2.59}$$

which can be inserted in (2.58) to get

$$\begin{bmatrix} \mathbf{x}^* \\ \mathbf{u}^* \end{bmatrix} = \begin{bmatrix} \mathbf{N}_{11} & \mathbf{N}_x \\ \mathbf{N}_{21} & \mathbf{N}_u \end{bmatrix} \begin{bmatrix} 0 \\ \mathbf{r} \end{bmatrix} = \begin{bmatrix} \mathbf{N}_x\mathbf{r} \\ \mathbf{N}_u\mathbf{r} \end{bmatrix} \tag{2.60}$$

From here, we can rewrite control law (2.53) as

$$\mathbf{u} = -\mathbf{K}\mathbf{x} + \mathbf{N}_u\mathbf{r} + \mathbf{K}\mathbf{N}_x\mathbf{r} \tag{2.61}$$

and we finally obtain controller expression (2.54) with

$$\bar{\mathbf{N}} := \mathbf{N}_u + \mathbf{K}\mathbf{N}_x. \tag{2.62}$$