Embedded Control

High-order electrical RLC oscillators - PID and LQR control

Hochschule Ravensburg-Weingarten
University of Applied Sciences

28.01.2022

Olti Cano
Anton Gres
Kastriot Thaqi
Michael Schichta
David Schlumberger

Contents

List of Figures

List of Tables

1 Problem statement

The interest of this work is to design an PID and LQR controller for a high-order electrical RLC oscillator. For this purpose, state space equations are first formed for the cases $n = 1, n = 2, n = 3$ and $n = n$. After that, the general stability behaviour is checked and simulated. Finally, the PID and LQR controls are created.

2 Part I: Research (by David Schlumberger)

The aim of this chapter is to find out how the state space representation of an electrical RLC oscillator system changes if one or multiple RLC oscillators are added to the system. In order to get this finding, the state space representations of a first order, a second order and a third order electrical RLC oscillator are set up.

Based on the results of this chapter, an algorithm could be defined in the next chapter.

2.1 First Order System

We investigate a first order electrical RLC oscillator as shown in Figure 2.1.1.

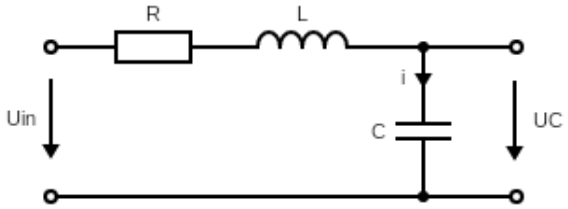


Figure 2.1.1: First order RLC oscillator

We assume the value of each component to be known, as well as the voltage over each component as

$$\begin{aligned} u_R(t) &= Ri(t), \\ u_L(t) &= L\frac{\partial i(t)}{\partial t}, \\ u_C(t) &= \frac{1}{C}\int i(\tau)d\tau. \end{aligned} \tag{2.1.1}$$

By reshaping Equation (2.1.1) we obtain the current

$$i(t) = C\frac{\partial u(t)}{\partial t} = C\dot{u}(t).$$

The mesh equation is noted as

$$\begin{aligned} u_{in}(t) &= u_R(t) + u_L(t) + u_C(t) = Ri(t) + L\frac{\partial i(t)}{\partial t} + u_C(t) \\ &= RC\dot{u}_C(t) + LC\ddot{u}_C(t) + u_C(t). \end{aligned} \tag{2.1.2}$$

We introduce new variables, namely

$$\begin{aligned} \text{input } u(t) &= u_{in}(t), \\ \text{state } z(t) &= u_C(t) \end{aligned}$$

which we insert into the mesh Equation (2.1.2) to yield

$$LC\ddot{z}(t) + RC\dot{z}(t) = u(t) - z(t). \tag{2.1.3}$$

We now have a representation of the system in the form

$$M\ddot{z}(t) + N\dot{z}(t) = \tilde{A}z(t) + \tilde{B}u(t).$$

The desired state space notation is

$$\dot{x}(t) = Ax(t) + Bu(t). \tag{2.1.4}$$

We reshape our representation by again introducing new variables, namely

$$\begin{aligned} x_1(t) &= z(t), \\ x_2(t) &= \dot{z}(t) = \dot{x}_1(t) \end{aligned}$$

which we insert into Equation (2.1.3) to yield

$$LC\dot{x}_2(t) = -x_1(t) - RCx_2(t) + u(t).$$

We can now describe the system by two equations

$$\begin{aligned} \dot{x}_1(t) &= x_2(t), \\ LC\dot{x}_2(t) &= -x_1(t) - RCx_2(t) + u(t) \end{aligned}$$

or by one equation in matrix notation

$$\begin{pmatrix} 1 & 0 \\ 0 & LC \end{pmatrix} \begin{pmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & -RC \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u(t).$$

We now have a representation of the system in the form

$$S\dot{\vec{x}}(t) = T\vec{x}(t) + \hat{B}u(t). \tag{2.1.5}$$

Since matrix S has obviously full rank, we can write

$$\dot{\vec{x}}(t) = S^{-1}T\vec{x}(t) + S^{-1}\hat{B}u(t).$$

We invert matrix S of Equation (2.1.5)

$$S^{-1} = \frac{1}{LC} \begin{pmatrix} LC & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{LC} \end{pmatrix}$$

and finally we can note our system in the desired state space notation (2.1.4)

$$\dot{\vec{x}}(t) = \begin{pmatrix} 0 & 1 \\ -\frac{1}{LC} & -\frac{RC}{LC} \end{pmatrix} \vec{x}(t) + \begin{pmatrix} 0 \\ \frac{1}{LC} \end{pmatrix} u(t)$$

and thus the stability of the system could be checked by eigenvalue computation, and classical control methods like state controller and state observer could be applied.

2.2 Second Order System

We investigate a second order electrical RLC oscillator as shown in Figure 2.2.1.

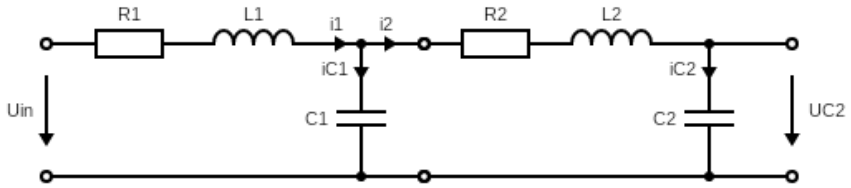


Figure 2.2.1: Second order RLC oscillator

We assume the value of each component to be known, as well as the voltage over each component as

$$u_{R_n}(t) = Ri_n(t), \tag{2.2.1}$$

$$u_{L_n}(t) = L_n \frac{\partial i_n(t)}{\partial t}, \tag{2.2.2}$$

$$u_{C_n}(t) = \frac{1}{C_n} \int i_{Cn}(\tau) d\tau \tag{2.2.3}$$

for $n \in \{1, 2\}$.

The node equation is

$$i_1(t) = i_2(t) + i_{C1}(t) = i_{C2}(t) + i_{C1}(t). \tag{2.2.4}$$

We insert Equation (2.2.3) into (2.2.4) and obtain the currents

$$i_1(t) = C_1 \frac{\partial u_{C1}(t)}{\partial t} + C_2 \frac{\partial u_{C2}(t)}{\partial t} = C_1 \dot{u}_{C1}(t) + C_2 \dot{u}_{C2}(t),$$
$$i_2(t) = C_2 \frac{\partial u_{C2}(t)}{\partial t} = C_2 \dot{u}_{C2}(t).$$

The first mesh Equation is denoted as

$$\begin{aligned} u_{in}(t) =& u_{R1}(t) + u_{L1}(t) + u_{C1}(t) \\ =& R_1 i_1(t) + L_1 \frac{\partial i_1(t)}{\partial t} + u_{C1}(t) \\ =& R_1 (C_1 \dot{u}_{C1}(t) + C_2 \dot{u}_{C2}(t)) \\ & + L_1 (C_1 \ddot{u}_{C1}(t) + C_2 \ddot{u}_{C2}(t)) + u_{C1}(t) \\ =& R_1 C_1 \dot{u}_{C1}(t) + R_1 C_2 \dot{u}_{C2}(t) \\ & + L_1 C_1 \ddot{u}_{C1}(t) + L_1 C_2 \ddot{u}_{C2}(t) + u_{C1}(t). \end{aligned} \tag{2.2.5}$$

The second mesh Equation is denoted as

$$\begin{aligned} u_{C1}(t) &= u_{R2}(t) + u_{L2}(t) + u_{C2}(t) \\ &= R_2 i_2(t) + L_2 \frac{\partial i_2(t)}{\partial t} + u_{C2}(t) \\ &= R_2 C_2 \dot{u}_{C2}(t) + L_2 C_2 \ddot{u}_{C2}(t) + u_{C2}(t). \end{aligned} \tag{2.2.6}$$

We introduce new variables, namely

$$\begin{aligned} \text{input } u(t) &= u_{in}(t), \\ \text{states } z_1(t) &= u_{C1}(t) \\ \text{and } z_2(t) &= u_{C2}(t) \end{aligned}$$

which we insert into the first and second mesh Equation (2.2.5) (2.2.6) to yield

$$\begin{aligned} L_1C_1\ddot{z}_1(t) + L_1C_2\ddot{z}_2(t) + R_1C_1\dot{z}_1(t) + R_1C_2\dot{z}_2(t) &= -z_1(t) + u(t), \\ L_2C_2\ddot{z}_2(t) \qquad\qquad + R_2C_2\dot{z}_2(t) &= \quad z_1(t) - z_2(t). \end{aligned}$$

We note the resulting mesh equations in matrix-vector notation as

$$\begin{aligned} &\begin{pmatrix} L_1C_1 & L_1C_2 \\ 0 & L_2C_2 \end{pmatrix} \begin{pmatrix} \ddot{z}_1(t) \\ \ddot{z}_2(t) \end{pmatrix} + \begin{pmatrix} R_1C_1 & R_1C_2 \\ 0 & R_2C_2 \end{pmatrix} \begin{pmatrix} \dot{z}_1(t) \\ \dot{z}_2(t) \end{pmatrix} \\ &= \begin{pmatrix} -1 & 0 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} z_1(t) \\ z_2(t) \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \end{pmatrix} u(t). \end{aligned}$$

We now have a representation of the system in the form

$$M\ddot{\vec{z}}(t) + N\dot{\vec{z}}(t) = \tilde{A}\vec{z}(t) + \tilde{B}u(t). \tag{2.2.7}$$

The desired state space notation is

$$\dot{\vec{x}}(t) = A\vec{x}(t) + B\vec{u}(t). \tag{2.2.8}$$

We reshape our representation by again introducing new variables, namely

$$\begin{aligned} \vec{x}_1(t) = \vec{z}(t) &= \begin{pmatrix} z_1(t) \\ z_2(t) \end{pmatrix}, \\ \vec{x}_2(t) = \dot{\vec{z}}(t) &= \begin{pmatrix} \dot{z}_1(t) \\ \dot{z}_2(t) \end{pmatrix} = \dot{\vec{x}}_1(t) \end{aligned}$$

which we insert into Equation (2.2.7) to yield

$$M\dot{\vec{x}}_2(t) = \tilde{A}\vec{x}_1(t) - N\vec{x}_2(t) + \tilde{B}u(t). \tag{2.2.9}$$

We can now describe the system by two equations

$$\begin{aligned} I\dot{\vec{x}}_1(t) &= I\vec{x}_2(t), \\ M\dot{\vec{x}}_2(t) &= \tilde{A}\vec{x}_1(t) - N\vec{x}_2(t) + \tilde{B}u(t) \end{aligned}$$

or by one equation in matrix notation

$$\begin{pmatrix} I & 0 \\ 0 & M \end{pmatrix} \begin{pmatrix} \dot{\vec{x}}_1(t) \\ \dot{\vec{x}}_2(t) \end{pmatrix} = \begin{pmatrix} 0 & I \\ \tilde{A} & -N \end{pmatrix} \begin{pmatrix} \vec{x}_1(t) \\ \vec{x}_2(t) \end{pmatrix} + \begin{pmatrix} 0 \\ \tilde{B} \end{pmatrix} u(t)$$

which is equivalent to

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & L_1C_1 & L_1C_2 \\ 0 & 0 & 0 & L_2C_2 \end{pmatrix} \begin{pmatrix} \dot{x}_{11}(t) \\ \dot{x}_{12}(t) \\ \dot{x}_{21}(t) \\ \dot{x}_{22}(t) \end{pmatrix}$$

$$= \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & -R_1C_1 & -R_1C_2 \\ 1 & -1 & 0 & -R_2C_2 \end{pmatrix} \begin{pmatrix} x_{11}(t) \\ x_{12}(t) \\ x_{21}(t) \\ x_{22}(t) \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} u(t). \tag{2.2.10}$$

We now have a representation of our second order electrical RLC oscillator in the form

$$S\dot{\vec{x}}(t) = T\vec{x}(t) + \hat{B}u(t). \tag{2.2.11}$$

The system is not yet completely in the desired state space notation (2.2.8), but we stop here with our research for the second order system, since a trend can already be determined. The final step would only be of computational effort, what we can delegate to MATLAB in the next chapter when designing the algorithm.

If matrix S of Equation (2.2.11) is invertible, the final step would be to write

$$\dot{\vec{x}}(t) = S^{-1}T\vec{x}(t) + S^{-1}\hat{B}u(t)$$

which would be equivalent to the desired state space notation (2.2.8) and thus the stability of the system could be checked and classical control methods could be applied.

2.3 Third Order System

We investigate a third order electrical RLC oscillator as shown in Figure 2.3.1.

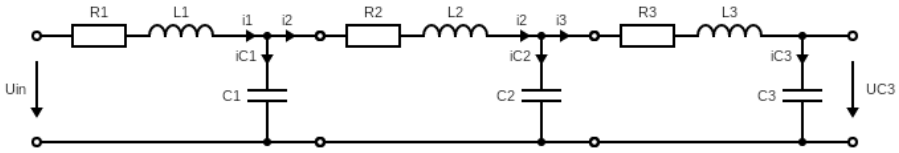


Figure 2.3.1: Third order RLC oscillator

We assume the value of each component to be known, as well as the voltage over each component as in Equation (2.2.1), (2.2.2) and (**??**) for $n \in \{1, 2, 3\}$.

We set up the node equations as we did in the previous chapter in Equation (**??**) and insert Equation (**??**) to obtain the currents

$$\begin{aligned} i_1 &= C_1\dot{u}_{C1} + C_2\dot{u}_{C2} + C_3\dot{u}_{C3}, \\ i_2 &= C_2\dot{u}_{C2} + C_3\dot{u}_{C3}, \\ i_3 &= C_3\dot{u}_{C3}. \end{aligned}$$

We set up the mesh equations as we did in the previous chapter in Equation (2.2.5) and (2.2.6) to yield

$$\begin{aligned} u_{in} &= R_1C_1\dot{u}_{C1} + R_1C_2\dot{u}_{C2} + R_1C_3\dot{u}_{C3} \\ &\quad + L_1C_1\ddot{u}_{C1} + L_1C_2\ddot{u}_{C2} + L_1C_3\ddot{u}_{C3} + u_{C1}, && (2.3.1) \\ u_{C1} &= R_2C_2\dot{u}_{C2} + R_2C_3\dot{u}_{C3} + L_2C_2\ddot{u}_{C2} + L_2C_3\ddot{u}_{C3} + u_{C2}, && (2.3.2) \\ u_{C2} &= R_3C_3\dot{u}_{C3} + L_3C_3\ddot{u}_{C3} + u_{C3} && (2.3.3) \end{aligned}$$

which we can reshape by introducing new state variables in order to get a representation of the system in the form (2.2.7) as

$$\begin{aligned} L_1C_1\ddot{z}_1(t) + L_1C_2\ddot{z}_2(t) + L_1C_3\ddot{z}_3(t) + R_1C_1\dot{z}_1(t) + R_1C_2\dot{z}_2(t) + R_1C_3\dot{z}_3(t) &= u(t) - z_1(t), \\ L_2C_2\ddot{z}_2(t) + L_2C_3\ddot{z}_3(t) \qquad\qquad + R_2C_2\dot{z}_2(t) + R_2C_3\dot{z}_3(t) &= z_1(t) - z_2(t), \\ L_3C_3\ddot{z}_3(t) \qquad\qquad\qquad\qquad + R_3C_3\dot{z}_3(t) &= z_2(t) - z_3(t) \end{aligned}$$

noted in matrix notation

$$\begin{pmatrix} L_1C_1 & L_1C_2 & L_1C_3 \\ 0 & L_2C_2 & L_2C_3 \\ 0 & 0 & L_3C_3 \end{pmatrix} \begin{pmatrix} \ddot{z}_1(t) \\ \ddot{z}_2(t) \\ \ddot{z}_3(t) \end{pmatrix} + \begin{pmatrix} R_1C_1 & R_1C_2 & R_1C_3 \\ 0 & R_2C_2 & R_2C_3 \\ 0 & 0 & R_3C_3 \end{pmatrix} \begin{pmatrix} \dot{z}_1(t) \\ \dot{z}_2(t) \\ \dot{z}_3(t) \end{pmatrix}$$
$$= \begin{pmatrix} -1 & 0 & 0 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \end{pmatrix} \begin{pmatrix} z_1(t) \\ z_2(t) \\ z_3(t) \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} u(t).$$

Finally, after again introducing new variables, we obtain a representation of our third order electrical RLC oscillator according to Equation (2.2.11) as

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & L_1C_1 & L_1C_2 & L_1C_3 \\ 0 & 0 & 0 & 0 & L_2C_2 & L_2C_3 \\ 0 & 0 & 0 & 0 & 0 & L_3C_3 \end{pmatrix} \begin{pmatrix} \dot{x}_{11}(t) \\ \dot{x}_{12}(t) \\ \dot{x}_{13}(t) \\ \dot{x}_{21}(t) \\ \dot{x}_{22}(t) \\ \dot{x}_{23}(t) \end{pmatrix}$$

$$= \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & -R_1C_1 & -R_1C_2 & -R_1C_3 \\ 1 & -1 & 0 & 0 & -R_2C_2 & -R_2C_3 \\ 0 & 1 & -1 & 0 & 0 & -R_3C_3 \end{pmatrix} \begin{pmatrix} x_{11}(t) \\ x_{12}(t) \\ x_{13}(t) \\ x_{21}(t) \\ x_{22}(t) \\ x_{23}(t) \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} u(t). \quad (2.3.4)$$

We now have a representation of our third order electrical RLC oscillator in the form

$$S\dot{\vec{x}}(t) = T\vec{x}(t) + \hat{B}u(t). \quad (2.3.5)$$

The system is not yet completely in the desired state space notation, but we stop here with our research for the third order system, since a trend can already be determined. The final step would only be of computational effort, what we can delegate to MATLAB in the next chapter when designing the algorithm.

If matrix S of Equation 2.3.5 is invertible, the final step would be to write

$$\dot{\vec{x}}(t) = S^{-1}T\vec{x}(t) + S^{-1}\hat{B}u(t)$$

which would be equivalent to the desired state space notation and thus the stability of the system could be checked and classical control methods could be applied.

The amount of added RLC subsystems can be increased even more. The behaviour of such higher order systems is treated in the following chapter.

3 Part II: Algorithm (by Anton Gres)

3.1 N-th order system

Continuing with the results from chapter 2, an equation for an nth order electrical RLC oscillator system shall be created. The attempt to find a general approach is demonstrated using a third order system or $n = 3$ as an example, seen in Figure 3.1.1.

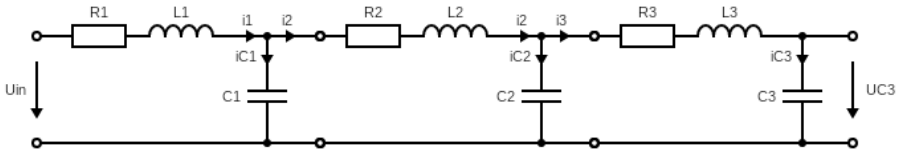


Figure 3.1.1: Third order RLC oscillator

With the help of Kirchhoff's current law and Kirchhoff's voltage law one can define the nodes and meshes in Figure 3.1.2. From this Figure one can conclude that

1. the number of mesh and node equations increase linear with the order number,
2. that the mesh and node equations have a predictable pattern with increasing order number and
3. that the mesh equation per mesh is based on the previous voltage and the node equation per node is based on the upcoming current.

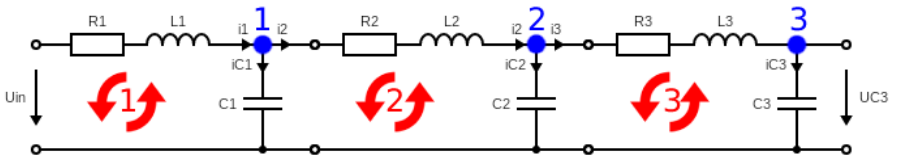


Figure 3.1.2: Graphic representation of the meshes and nodes

Therefore, the following rules for creating a state space representation for a nth order system can be established:

1. **Node equations**
   For n equations the following node equations can be noted as

$$i_k(t) = \sum_{j=k}^{n} C_j \frac{\partial u_{Cj}(t)}{\partial t} = \sum_{j=k}^{n} C_j \dot{u}_{Cj}(t) \quad \text{with } k \in N : 0 < k \leq n \tag{3.1.1}$$

2. **Mesh equations**
   For n equations with $u_{C0}(t) = u_{in}$ the following mesh equations can be noted as

$$\begin{gathered} u_{Cm-1}(t) = R_m i_m(t) + L_m \frac{\partial i_m(t)}{\partial t} + u_{Cm}(t) \\ \text{with} \quad m \in N : 0 < m \leq n \end{gathered} \tag{3.1.2}$$

3. **Setting up the general matrix**
   All generated equations can be merged into the form $M\ddot{\vec{z}}(t) + N\dot{\vec{z}}(t) = \tilde{A}\vec{z}(t) + \tilde{B}u(t)$ with the new variables names $u(t) = u_{in}$ and $z_m(t) = u_{Cm}(t)$.

4. **Setting up the state space equation**
   For the desired state space equation $\dot{\vec{x}}(t) = A_C\vec{x}(t) + B_C u(t)$, the general matrix is reshaped again with the new variables $x_1(t) = z(t)$ and $x_2(t) = \dot{z}(t) = \dot{x}_1(t)$. This yields an equation of the form $S\dot{\vec{x}}(t) = T\vec{x}(t) + \hat{B}u(t)$ and thus the state space equation $\dot{\vec{x}}(t) = S^{-1}[T\vec{x}(t) + \hat{B}u(t)] = A_C\vec{x}(t) + B_C u(t)$.

3.2 Algorithm: Example on an N-th order system

In order to form a state space notation of an N-th order system, the previously discussed rules must be applied. In particular, the following node equations can be derived from (3.1.1)

$$\begin{aligned} i_N(t) &= C_N \dot{u}_{CN}(t) \\ i_{N-1}(t) &= C_{N-1}\dot{u}_{CN-1}(t) + C_N \dot{u}_{CN}(t) \\ &\vdots \\ i_1(t) &= \sum_{n=1}^{N} C_n \dot{u}_{Cn}(t). \end{aligned} \tag{3.2.1}$$

The mesh equations can be obtained from (3.1.2) by

$$\begin{aligned} u_{in} = u_{C0}(t) &= R_1 i_1(t) + L_1 \frac{\partial i_1(t)}{\partial t} + u_{C1}(t) \\ u_{C1}(t) &= R_2 i_2(t) + L_2 \frac{\partial i_2(t)}{\partial t} + u_{C2}(t) \\ &\vdots \\ u_{CN-1}(t) &= R_N i_N(t) + L_N \frac{\partial i_N(t)}{\partial t} + u_{CN}(t). \end{aligned} \tag{3.2.2}$$

The systems of equations (3.2.1) and (3.2.2) can be joined together

$$
\begin{aligned}
u_{in} &= R_1 \left[\sum_{n=1}^{N} C_n \dot{u}_{Cn}(t)\right] + L_1 \left[\sum_{n=1}^{N} C_n \ddot{u}_{Cn}(t)\right] + u_{C1}(t) \\
u_{C1}(t) &= R_2 \left[\sum_{n=2}^{N} C_n \dot{u}_{Cn}(t)\right] + L_2 \left[\sum_{n=2}^{N} C_n \ddot{u}_{Cn}(t)\right] + u_{C2}(t) \\
&\vdots \\
u_{CN-1}(t) &= R_N C_N \dot{u}_{CN}(t) + L_N C_N \ddot{u}_{CN}(t) + u_{CN}(t).
\end{aligned}
\tag{3.2.3}
$$

After the conversion of (3.2.3) with $z_m(t) = u_{Cm}(t)$, the system of equations can be put into a more readable form

$$
\begin{aligned}
L_1C_1\ddot{z}_1(t) + L_1C_2\ddot{z}_2(t) + \cdots + L_1C_N\ddot{z}_N(t) + R_1C_1\dot{z}_1(t) + R_1C_2\dot{z}_2(t) + \cdots + R_NC_N\dot{z}_N(t) &= u(t) - z_1(t) \\
L_2C_2\ddot{z}_2(t) + \cdots + L_2C_N\ddot{z}_N(t) \qquad + R_2C_2\dot{z}_2(t) + \cdots + R_2C_N\dot{z}_N(t) &= z_1(t) - z_2(t) \\
\vdots \qquad\qquad\qquad \vdots \\
L_NC_N\ddot{z}_N(t) \qquad + R_NC_N\dot{z}_N(t) &= z_{N-1}(t) - z_N(t)
\end{aligned}
$$

from which the matrix form $M\ddot{\vec{z}}(t) + N\dot{\vec{z}}(t) = \tilde{A}\vec{z}(t) + \tilde{B}u(t)$ can be easily read out

$$
\begin{aligned}
&\begin{pmatrix} L_1C_1 & L_1C_2 & \cdots & L_1C_N \\ 0 & L_2C_2 & \cdots & L_2C_N \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & L_NC_N \end{pmatrix}
\begin{pmatrix} \ddot{z}_1(t) \\ \ddot{z}_2(t) \\ \vdots \\ \ddot{z}_N(t) \end{pmatrix}
+
\begin{pmatrix} R_1C_1 & R_1C_2 & \cdots & R_1C_N \\ 0 & R_2C_2 & \cdots & R_3C_N \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & R_NC_N \end{pmatrix}
\begin{pmatrix} \dot{z}_1(t) \\ \dot{z}_2(t) \\ \vdots \\ \dot{z}_N(t) \end{pmatrix} \\
&\qquad = \begin{pmatrix} -1 & 0 & \cdots & 0 \\ 1 & -1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 1 & -1 \end{pmatrix}
\begin{pmatrix} z_1(t) \\ z_2(t) \\ \vdots \\ z_N(t) \end{pmatrix}
+ \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} u(t).
\end{aligned}
\tag{3.2.4}
$$

With new variable names $x_1(t) = z(t)$ and $x_2(t) = \dot{z}(t) = \dot{x}_1(t)$ the final state space notation $S\dot{\vec{x}}(t) = T\vec{x}(t) + \hat{B}u(t)$ can be constructed from (3.2.4) as follows

$$
\begin{pmatrix} I & 0 \\ 0 & M \end{pmatrix}
\begin{pmatrix} \dot{\vec{x}}_1(t) \\ \dot{\vec{x}}_2(t) \end{pmatrix}
=
\begin{pmatrix} 0 & I \\ \tilde{A} & -N \end{pmatrix}
\begin{pmatrix} \vec{x}_1(t) \\ \vec{x}_2(t) \end{pmatrix}
+ \begin{pmatrix} 0 \\ \tilde{B} \end{pmatrix} u(t).
\tag{3.2.5}
$$

Of these derivations, the following MATLAB algorithm can be written to calculate the matrices $M, N, \tilde{A}$ and $\tilde{B}$. The $N \times N$ zero matrices and zero vectors are created and then populated as described in (3.2.4).

```
function [M,N,A,B] = RLC_osc(R, L, C)
    % Get size of array(s)
    order = length(R);

    % Create base matrices
    M = zeros(order);
    N = zeros(order);
    A = -1*eye(order);
    B = zeros(order, 1);

    for row = 1:order
        for column = row:order
            % fill matrices with corresponding value pairs
            M(row, column) = L(row)*C(column);
            N(row, column)= R(row)*C(column);

            if row > 1
                % fill matrix A with ones under the main diagonal
                A(row, row - 1) = 1;
            end
        end
    end

    B(1) = 1;
end
```

The matrices can be used in the subsequent algorithm to create the state space notation $S\dot{\vec{x}}(t) = T\vec{x}(t) + \hat{B}u(t)$ and consequently $\dot{\vec{x}}(t) = S^{-1}(T\vec{x}(t) + \hat{B}u(t)) = A_C\vec{x}(t) + B_C u(t)$.

For this the matrices $M, N, \tilde{A}$ and $\tilde{B}$ are used to fill the matrices S, T and $\hat{B}$ as described in (3.2.5). Thereby the matrices A_C and B_C can be devised.

```
function [Ac,Bc] = RLC_ss(M, N, A, B)
    % Get order of resulting matrices
    order = length(B);

    % Build matrices
    S = [eye(order), zeros(order);
        zeros(order), M];

    T = [zeros(order), eye(order);
        A, -N];

    % Calculate Ac and Bc
    Ac = S\T;
    Bc = S\[zeros(order, 1); B];
end
```

4 Part III: Stability (by Kastriot Thaqi)

In this chapter the stability of the RLC system will be tested. For this the algorithm explained in the chapter 3.2 is going to be used. First the stability of a higher order system with constant values for the R, L and C components is checked.

Then different variants will be tested for example rising or falling values for R, then for L and lastly for C while the other components stay the same. Afterwards constant high and low values for C and L are going to be examined for stability. And lastly a real example of a possible low-pass circuit is tested for stability.

To find out if the system is stable the eigenvalues of the system matrix A have to be calculated. And if the real part of all eigenvalues is negative then the system is said to be asymptotically stable [1].

Proofing the stability by calculating the A matrix is possible because of the state space representation of the RLC system and it being linearly as well as time-invariant.

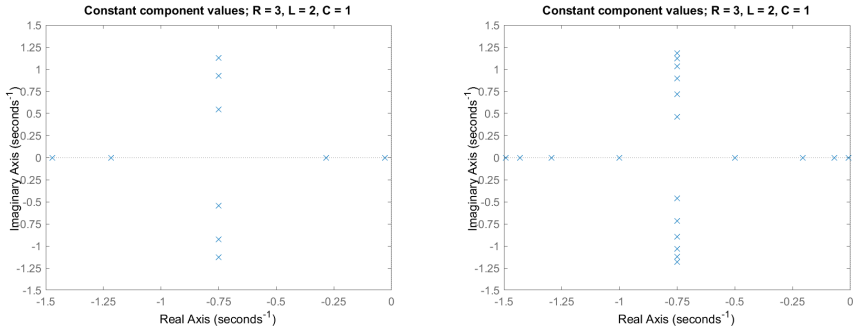
4.1 Stability of higher order RLC systems

Calculating the eigenvalues of the state matrix A analytically is too complex for higher-order systems. This is why the algorithm created in chapter 3.2 is used with the addition of the MATLAB function eig(A).

This function returns all eigenvalues of a matrix with real and imaginary part. The eigenvalues are then shown on a map, called eigenvalue spectrum, for better visualization. If all eigenvalues are on the left half of the plane then the system is stable.

It can be compared to the s-plane when displaying poles and zeros in transfer function representation. As it can be assumed that the poles in the transfer function are the same as the eigenvalues of the system matrix A [1].

For the following Figures the values for the components are $R = 3$, $L = 2$ and $C = 1$. These values are constant and only the order of the system increases. The values are also chosen arbitrary.



(a) 5th order (b) 10th order

Figure 4.1.1: Eigenvalue spectrum with constant R, L and C values

Now through the eigenvalue spectrum the behaviour of the system can be estimated. For example as seen in Figure 4.1.1 the position of the eigenvalues on the left half plane define how fast a system is. Poles that are farther away from the origin make the system react quicker. Imagining an impulse that is applied to the system the poles that are farthest away make the system return much faster to its original state.

In this case the pole which is at $x = -0.0275$ (x being the real axis) on the left graph is the slowest reacting pole while the pole at $x = -1.47$ is the fastest. It can also be assumed that poles influence each other meaning that the pole that is closest to the origin is the more dominant pole for the system behaviour. This makes the system react more slowly even though it has the fast reacting pole at $x = -1.47$.

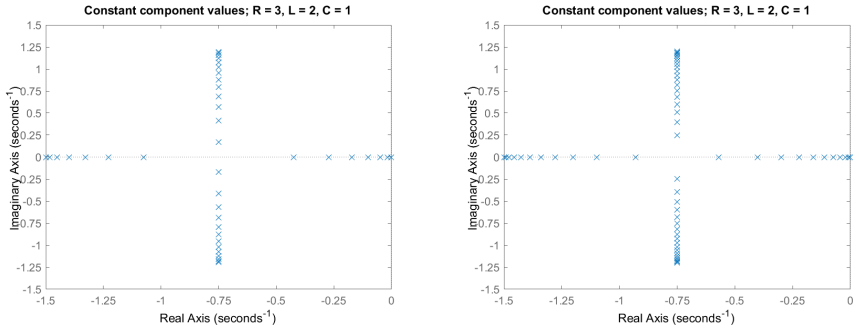
By looking at the poles that have a real part and an imaginary part it is possible to tell that the system has a damped oscillation when the pole is on the left half of the s-plane. The higher the imaginary part is the greater is the amplitude of the oscillation as it has a smaller damping value.

In the case of Figure 4.1.1 the complex conjugate pole pair with the greatest y value has a damping factor of *0.553*. This causes an overshoot before the system output settles.

Looking at the graph on the right of Figure 4.1.1 the poles of a 10th order system can be seen. Here almost the same characteristic of the system is visible as in the graph on the left but it can be estimated that the system reacts slower than before.

Due to the addition of eigenvalues on the real axis and more being closer to the origin. The pole closest to the origin has the value $x = -0.00748$ while the pole farthest away is at $x = -1.49$.

The increase in the order also effected the complex conjugate poles on the imaginary axes. Where the poles causing the greatest oscillation have a damping factor of 0.536. What can also be seen is that the poles on the real axis don't exceed the value of 1.5.



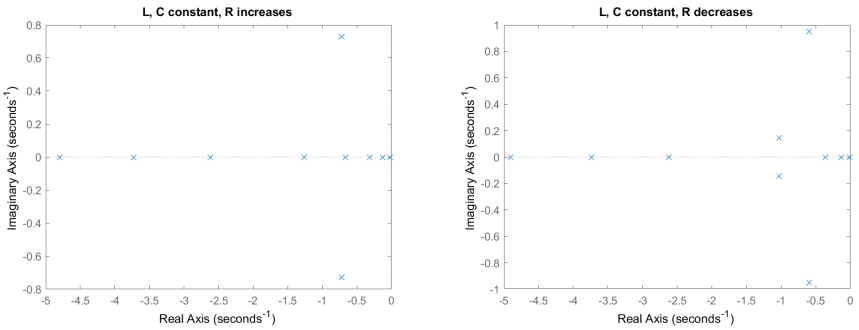
(a) 20th order (b) 30th order

Figure 4.1.2: Constant R, L and C values

Figure 4.1.2 shows how the poles behave when increasing the order of the system even more. On the left side the system has an order of 20 and on the right side the system has an order of 30. In general the behaviour is still the same as the poles only marginally increase on the imaginary axis and on the real axis the eigenvalues get closer to the origin and -1.5.

4.2 Stability with varying R component

Now the stability is tested with constant L and C components but the value of the R component varies. First it increases for every subsequent circuit. This means that the value of R in the first circuit is 2 then in the adhering circuit the value is 4 and so on. Afterwards the poles are compared to a decreasing R component which is done in a similar way where the first value of R is 10 and for every subsequent circuit the value decreases by 2.



(a) 5th order increasing R

(b) 5th order decreasing R

Figure 4.2.1: Varying R values with constant L = 2 and C = 1

	Increasing R value	Decreasing R value
1st circuit	2	10
2nd circuit	4	8
3rd circuit	6	6
4th circuit	8	4
5th circuit	10	2

Table 4.2.1: Overview of resistor values with order of 5

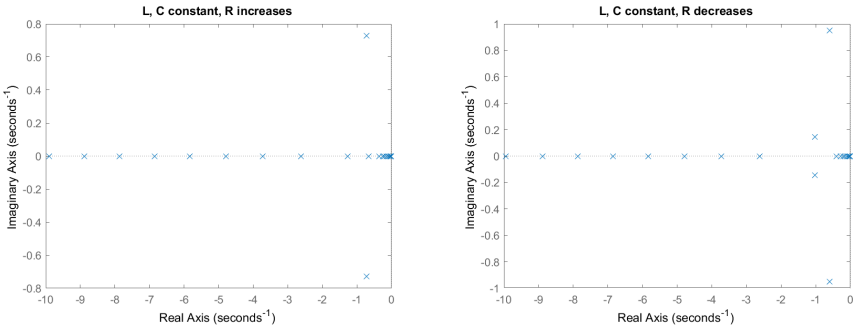
In Figure 4.2.1 the eigenvalue spectrum with the values for the resistor stated in the Table 4.2.1 is displayed.

Both graphs show that with an increasing and decreasing resistor value the system stays stable. The eigenvalues that are closer to the origin are again more dominant and in turn make the system react slower.

Additionally both systems have complex conjugate eigenvalues. For the system with the increasing R value has there is only one eigenvalue pair on the imaginary axis while for the system with the decreasing R value there are two. As a result the oscillation is estimated to be greater than for the system with one conjugate complex pole pair.

Comparing this Figure with the Figures from the chapter before, for example 4.1.1, it is visible that an increasing as well as a decreasing R component value affects the oscillation. Both oscillations are slower and have a smaller amplitude.

The next Figure 4.2.2 shows the poles of a 10th order system. Here it is visible that the left most pole has a value of almost -10 and the pole close to the origin has now a value of -0.0015 where before it was at -0.01. The conjugate complex pole pairs on the other hand stayed at the same position with only a very minimal change which can be neglected.



(a) 10th order increasing R

(b) 10th order decreasing R

Figure 4.2.2: Varying resistor values with constant L = 2 and C = 1

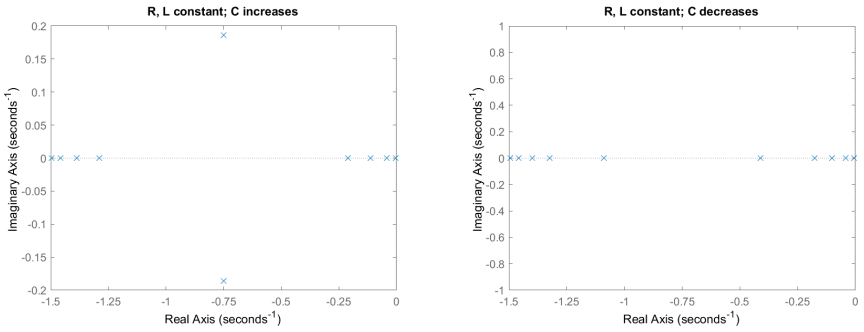
	Increasing R value	Decreasing R value
1st circuit	2	20
2nd circuit	4	18
...	...	...
9th circuit	18	4
10th circuit	20	2

Table 4.2.2: Overview of resistor values for order of 10

This means that the oscillation is still as great as in the system with the order of 5 but with even more poles closer to the origin the system should react slower.

4.3 Stability with varying C component

Coming next is checking the stability of an RLC system where the R and L component values are constant only the capacitor component value varies. The variation is done in the same way as in the chapter 4.2.



(a) 5th order increasing C

(b) 5th order decreasing C

Figure 4.3.1: Varying C values with constant R = 3 and L = 2

	Increasing C value	Decreasing C value
1st circuit	2	10
2nd circuit	4	8
3rd circuit	6	6
4th circuit	8	4
5th circuit	10	2

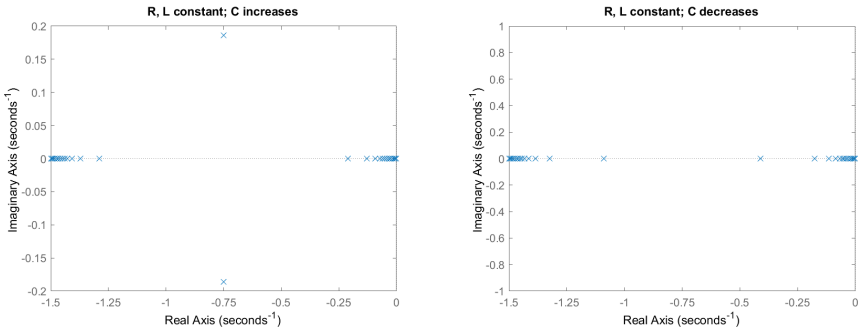
Table 4.3.1: Overview of capacitor values with order of 5

Figure 4.3.1 shows that with a varying C component the system is still stable. What is interesting to see is that the system has very little or no oscillation. Looking at the left graph where C increases only one complex conjugate pole pair is present with an imaginary value of ± 0.185. This means that the oscillation is damped more and because of the position on the real axis the oscillation is relatively fast.

When the C value decreases on the other hand there is no visible oscillation in place. All poles are on the real axis with the more dominant poles close to the origin causing the system to react more slowly.

Increasing the order of the system as seen in Figure 4.3.2 to 15 has no effect on the stability. The system is still stable with a similar characteristic as with the order of 5.

The complex conjugate pole is still at the same position with the same oscillation when C increases. The poles on the real axis have shifted closer to the origin and the value of -1.5.



(a) 15th order increasing C

(b) 15th order decreasing C

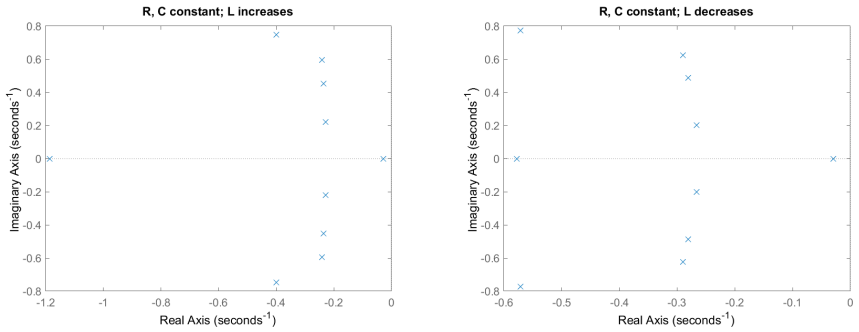
Figure 4.3.2: Varying capacitor values with constant R = 3 and L = 2

	Increasing C value	Decreasing C value
1st circuit	2	30
2nd circuit	4	28
...	...	...
14th circuit	28	4
15th circuit	30	2

Table 4.3.2: Overview of capacitor values with order of 15

4.4 Stability with varying L component

Now the stability of a system with a varying inductor component is checked. This means that the R and C component values are constant throughout the circuit. Only the value of the L component increases or decreases.



(a) 5th order increasing L

(b) 5th order decreasing L

Figure 4.4.1: Varying L values with constant R = 3 and C = 1

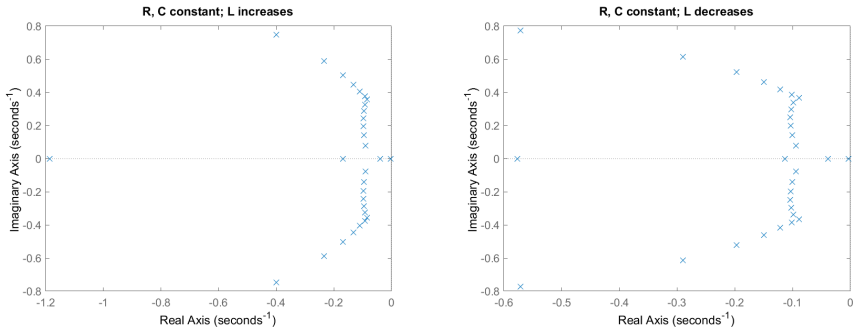
	Increasing L value	Decreasing L value
1st circuit	2	10
2nd circuit	4	8
3rd circuit	6	6
4th circuit	8	4
5th circuit	10	2

Table 4.4.1: Overview of inductor values with order of 5

In Figure 4.4.1 it is visible that the system is stable for a decreasing or increasing inductor value as all poles are on the left half of the plane. The eigenvalue spectrum of both cases also show that there are only two eigenvalues without an imaginary part. Also the value of the smallest real pole almost doubles when the L value increases. This could affect the reaction of the system in general.

Also an oscillation is visible through the poles that are on the imaginary axis. The poles from both graphs seem to have a similar oscillation characteristic with the only difference being that the highest pole on the imaginary axis which causes a less damped oscillation is farther away from the origin in the decreasing inductor graph.

This means that the pole has less of an effect on the oscillation due to more poles being closer to the origin. When looking at the Figure 4.4.2 where the order of the system is at 15 a pattern of the complex conjugate poles is visible and it being still stable.



(a) 15th order increasing L

(b) 15th order decreasing L

Figure 4.4.2: Varying L values with constant R = 3 and C = 1

	Increasing L value	Decreasing L value
1st circuit	2	30
2nd circuit	4	28
...	...	...
14th circuit	28	4
15th circuit	30	2

Table 4.4.2: Overview of inductor values with order of 15

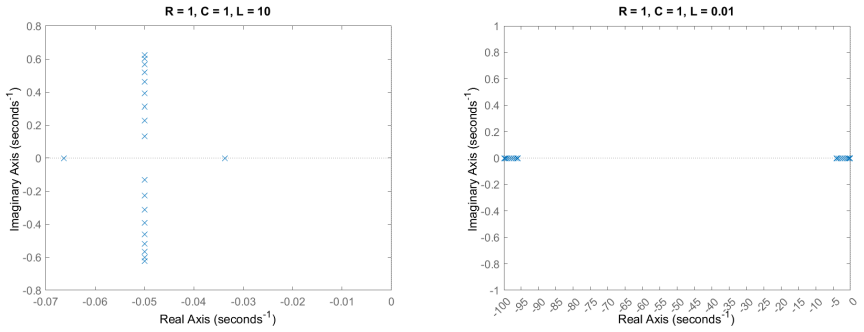
In this case most of the eigenvalues are on the imaginary axis. With the majority of them being around the value -1 on the real axis. The complex conjugate eigenvalues from graph 4.4.2a and 4.4.2b are clustered up around ± 0.1 to ± 0.3 on the real axis. This causes a more damped and slow oscillating behaviour.

Although the order has been increased to 15 the maximum values on the real axis did not exceed -1.2 when the inductor value is increased or -0.6 when the inductor value is decreased. Another difference is the two additional poles on the real axis which are close to the origin.

4.5 Comparing the stability of high and low values of the inductor and capacitor

In this chapter the stability of the system is checked with a relatively high and low value for the capacitor and the inductor. Only one of these elements is changed while the other two components are constant.

The order chosen to look at the stability of this system is 10. The reason being that already at the 10th order a certain pattern can be assumed so when increasing the order the eigenvalues follow that pattern.



(a) 10th order with L = 10

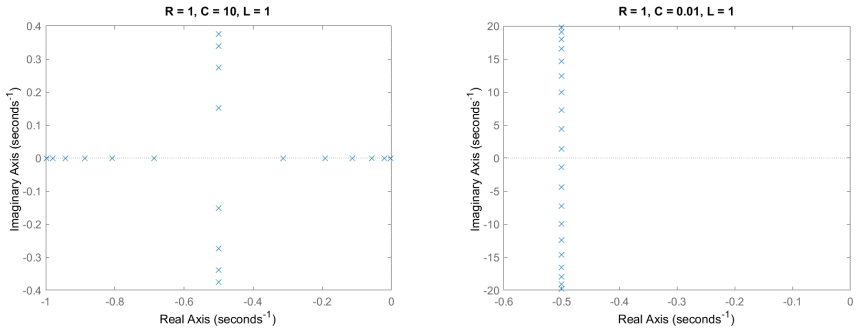
(b) 10th order with L = 0.01

Figure 4.5.1: Constant C, R and high/low L values

The stability of the system with a constant resistance and capacitance value, in this case the values are $R = 1$ and $C = 1$, and a high as well as low inductor value can be seen in Figure 4.5.1. Both graphs show a stable system. But with a low inductor value of 0.01 there is no visible oscillation.

The poles on the real axis also have a great distance to each other with the lowest pole being at -99.9 and the pole closest to the origin at -0.02. This makes the system react slow.

Through an L value of 10 the behaviour changes a lot. Now there is a less damped oscillation taking place and all poles are very close to the origin. With the maximum negative pole being at -0.066. This would show a very slow oscillation with a great amplitude due to the relatively high imaginary value.



(a) 10th order with C = 10 (b) 10th order with C = 0.01

Figure 4.5.2: Constant L, R and high/low C values

Figure 4.5.2 displays the eigenvalue spectrum of the varying C component with a high and a low value. Here the system is as well stable for both cases. What can be seen is that the characteristic of the eigenvalues is similar to Figure 4.1.1.

There the amount of eigenvalues that are on the real axis and on the imaginary axis is almost the same. But here less eigenvalues are complex conjugate.

On the contrary with a small capacitor value it shows a very strong oscillation with a high amplitude as there are no eigenvalues on the real axis to dampen it. Also the highest imaginary value is at almost ± 20.

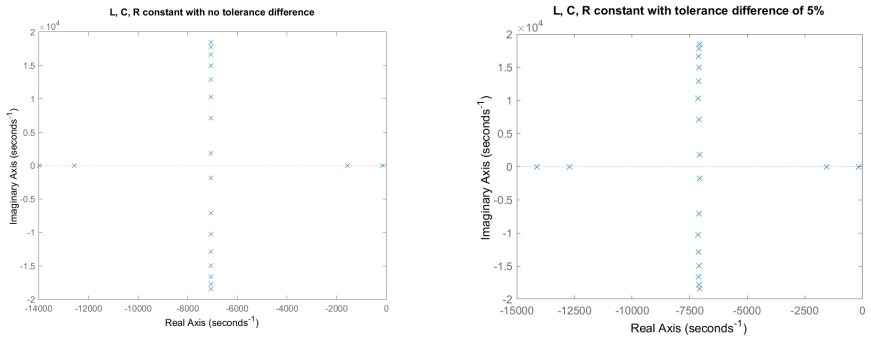
4.6 Stability with influence of component tolerance

Lastly the stability with the influence of component tolerance is going to be tested. For this an RLC low-pass filter with the values of $R = 14.14\Omega$, $L = 1mH$ and $C = 10\mu F$ is picked. To these values a random tolerance of 5%, 10% and 20% has been added.

How the random tolerance is calculated and added to each value in MATLAB can be seen below with the example of a 5% tolerance

```
R = 14.14 + (14.14*0.05)*(randn(1,1)/4);
L = 1e-3 + (1e-3*0.05)*(randn(1,1)/4);
C = 10e-6 + (10e-6*0.05)*(randn(1,1)/4);
```

In this case the order of the system is set to be 10. The following Figure 4.6.1 shows the eigenvalue spectrum of the system without a tolerance and with a tolerance of 5%.



(a) no tolerance

(b) 5% tolerance

Figure 4.6.1: Eigenvalue spectrum of components with tolerance

It is visible that the system in general is stable. The eigenvalue closest to the origin for Figure 4.6.1a is at -159.7 while the farthest eigenvalue on the real axis is at -13980.2. This already shows that the system will react really fast.

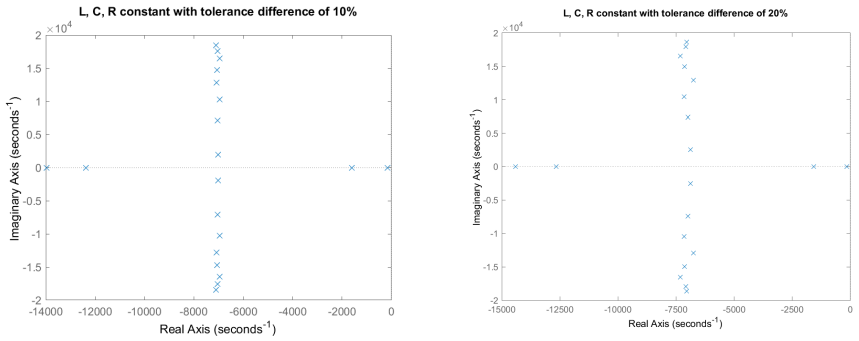
The poles with an imaginary part are at -7070 on the real axis and these conjugate complex poles have a maximum value of ± 18469.69. This causes the oscillation of the system to reach a very high amplitude.

For the system with a 5% tolerance for each component the eigenvalue spectrum looks similar. The poles have moved slightly with the maximum real pole being at -14134.69 and the pole closest to the origin being at -160.2.

While the complex conjugate eigenvalues are now placed at around -7133 to -7075. The behaviour of the system should be similar as the values of the poles did not change that much. Considering the values of the components it is interesting to see how much larger the values for the poles are.

Because of this it is to be expected as stated before that the system will react really quick. Figure 4.6.2 shows what a difference of 10% and 20% tolerance can make for the poles of the system.

The stability of the system is still ensured as all poles are on the left half of the plane. But comparing it to the eigenvalue spectrum from Figure 4.6.1 it is observable that the poles have moved even more.



(a) 10% tolerance

(b) 20% tolerance

Figure 4.6.2: Eigenvalue spectrum of components with tolerance

The complex conjugate eigenvalues have moved in a range from -7264 to -6891 on the real axis for the system with 20% tolerance. This movement affects the reaction of the system as well as the oscillation.

It is also noticeable that the poles on the real axis have moved greatly. The closest pole to the origin is now at -147 and the outer most left pole is at -14468.

5 Part IV: Simulation of RLC systems (by Kastriot Thaqi)

In this chapter the simulation consisting of the step response for some RLC systems that have been mentioned in the chapter 4 is shown. The simulation is done through MATLAB.

In the simulation every step response from the output of every circuit is plotted. This means that the step response of the first circuit is displayed up to the step response of the last circuit in the RLC system.

For this case the order of the system is 10 which in turn results in 10 step responses shown in the Figures. Additionally the change over time of every corresponding step response is included in these Figures.

Looking at Figure 5.0.1 the step response of an RLC system with a constant R, L and C value show a relatively slow reacting system. The change over time has been magnified for better visibility and can be seen in the red rectangle.

Only for the example with the constant R, L, C values the order for the step response is increased to 30. To see how much the system response is affected by an increasing order. And through Figure 5.0.2 it is readable that the system is almost 10 times slower than with an order of 10.

Following the constant R, L, C step responses is first the example of a high value for the inductor and then the example of a low value for the capacitor shown in Figure 5.0.3 and 5.0.5. Lastly the simulation of the system with real values plus a tolerance is displayed by Figure 5.0.8.

Each Figure after the constant R, L, C step responses show in more detail the first output and the last output. The emphasis lies on these two because it is interesting to see how an output changes for every consecutive circuit until it reaches the end.

Most noticeable from the step responses following after Figure 5.0.1 and Figure 5.0.2 is the time it takes to reach the amplitude 1. Also the increase in oscillation is perfectly displayed in Figure 5.0.3 and 5.0.5 where many or all eigenvalues are complex conjugates.

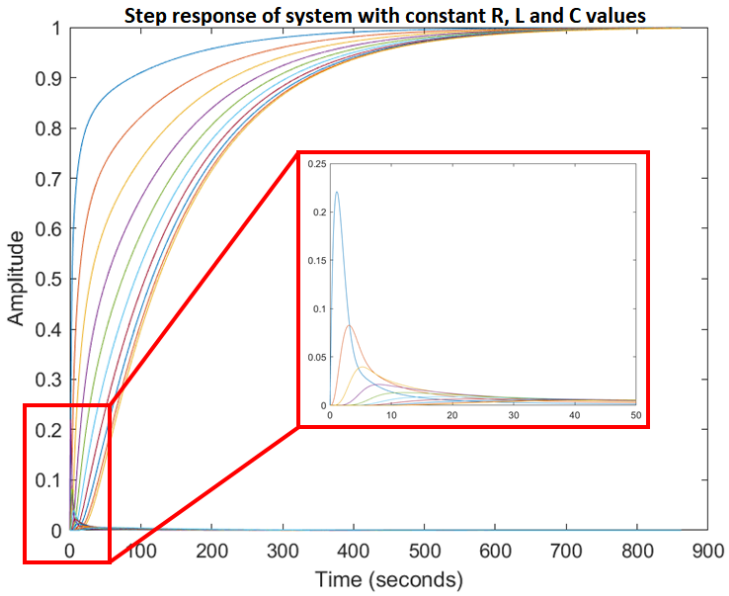


Figure 5.0.1: Step Response of system with constant R, L, C and order of 10 with change over time highlighted in red square

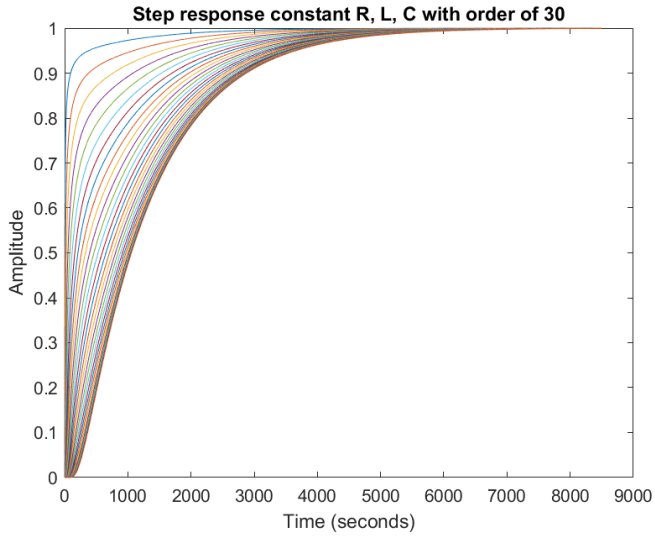


Figure 5.0.2: Same system components as in Figure 5.0.1 but with the order of 30

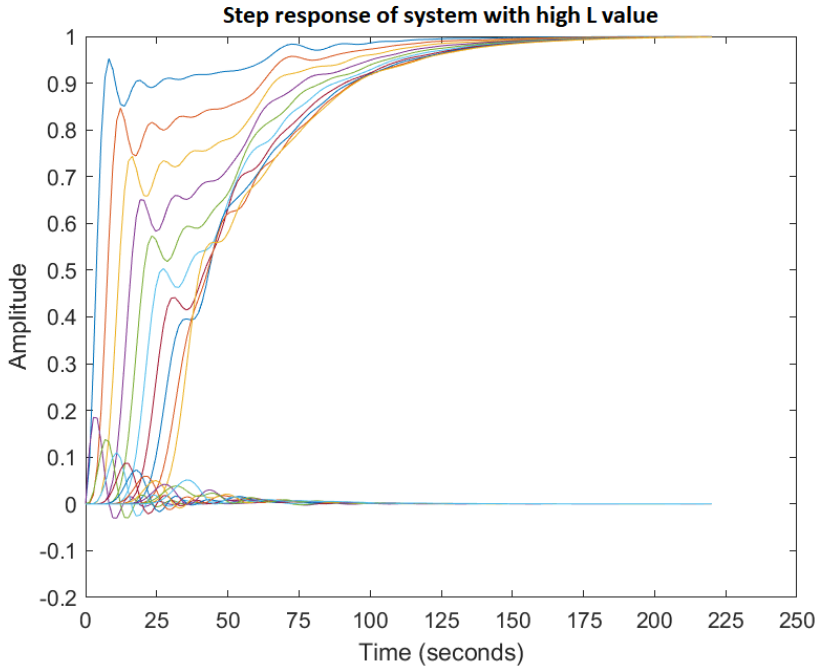
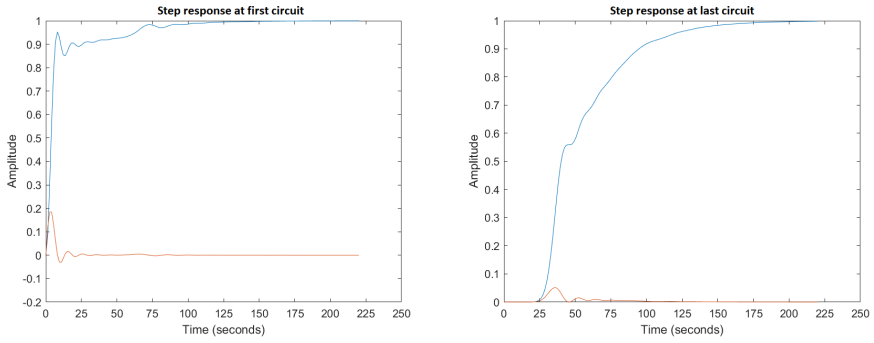


Figure 5.0.3: Step Response of Figure 4.5.1(a)



(a) (b)

Figure 5.0.4: Closer look at (a) first and (b) last circuit step response with each corresponding step response of 5.0.3

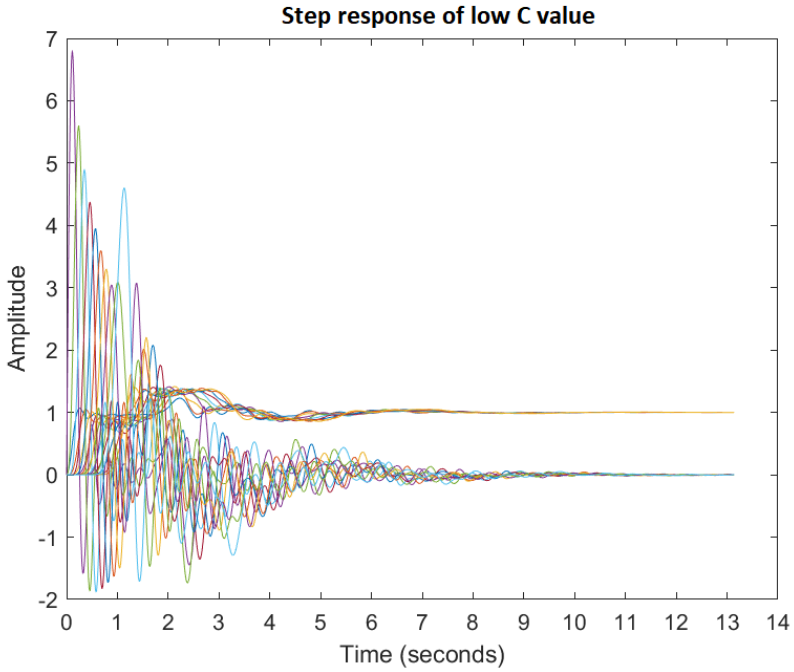
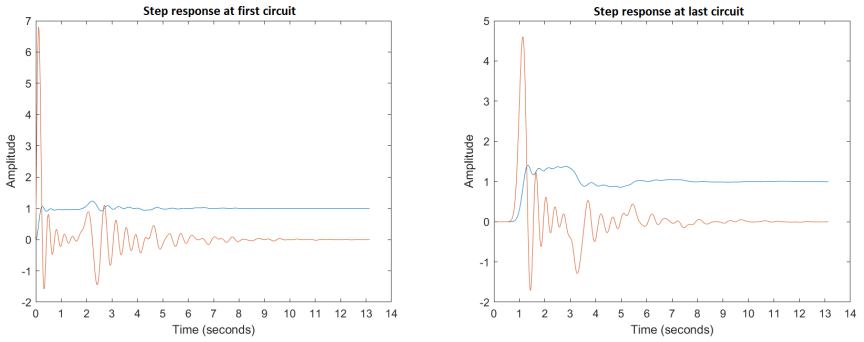


Figure 5.0.5: Step Response of Figure 4.5.2(b)



(a) (b)

Figure 5.0.6: Emphasis on the first (a) and last (b) step response of 5.0.5

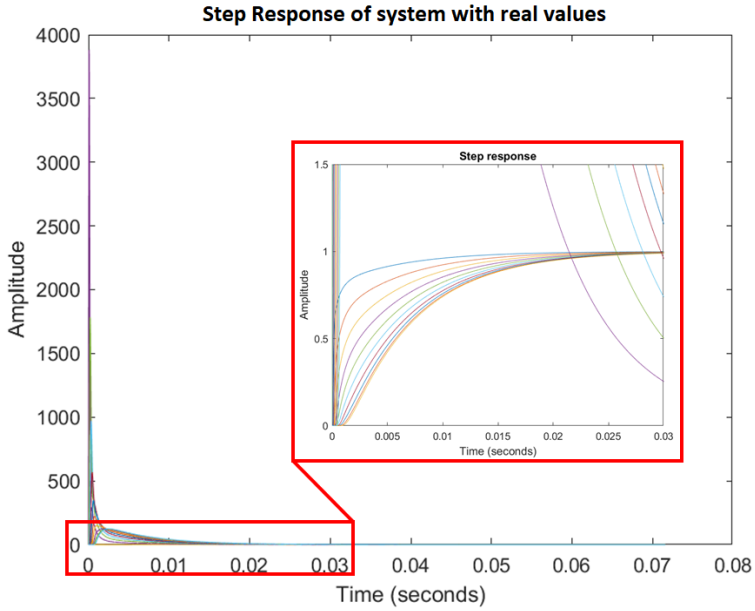


Figure 5.0.7: Step Response of Figure 4.6.1(a) seen in red square

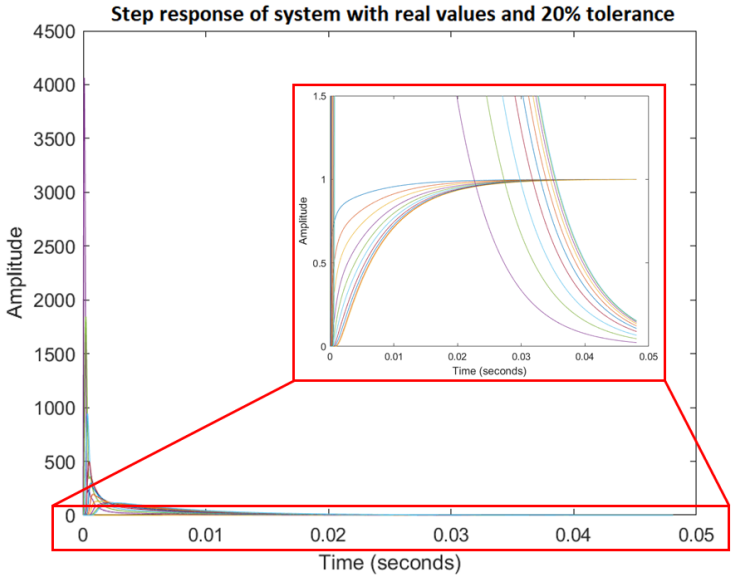
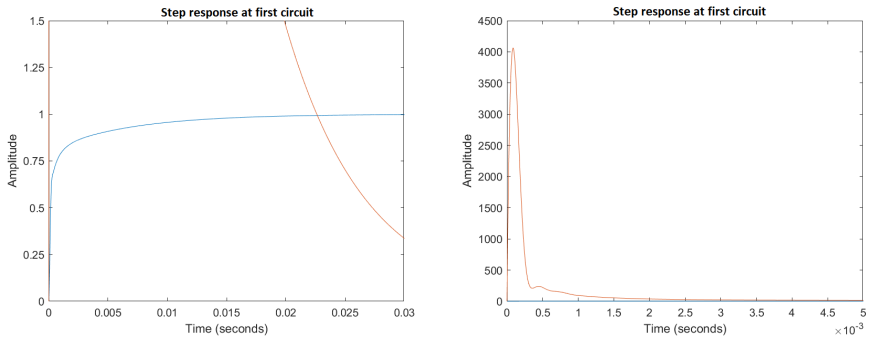
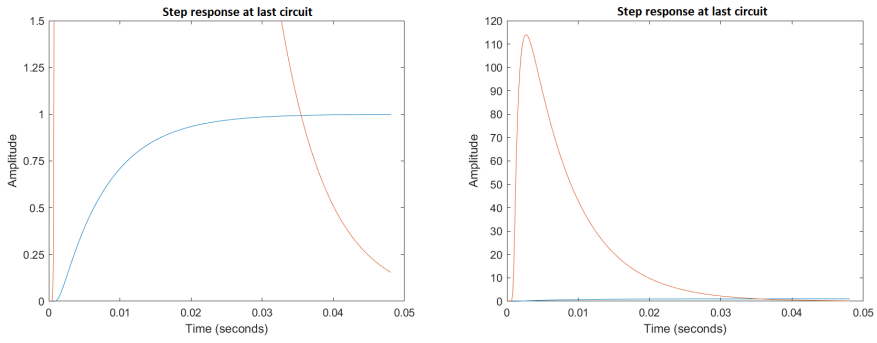


Figure 5.0.8: Step Response of Figure 4.6.2(b) shown in red square



(a) (b)

Figure 5.0.9: Focus on first step response (a) with change over time (b) of 5.0.8



(a) (b)

Figure 5.0.10: Focus on last step response (a) with change over time (b) of 5.0.8

6 Part V: PID Controller (by Michael Schichta)

Within this chapter the tuning and implementation of a PID controller is covered. The implementation in C is integrated into the MATLAB environment as a part of a software-in-the-loop routine.

6.1 Computing the transfer functions

As presented in chapter 2 and then 3, n-coupled RLC circuits are translated in to their respective state space model via the developed MATLAB functions. Before we can construct the transfer functions of the system we first have to construct the remaining matrices C and D for the state space model.

As an example a circuit of three coupled RLC circuits is assumed. This results in the following matrices for A_c and B_c.

```
R = [1, 1, 1];
L = [1, 1, 1];
C = [1, 1, 1];

% using the developed functions from the team
[M,N,A,B] = RLC_osc(R,L,C);

[Ac, Bc] = RLC_ss(M,N,A,B);
```

This results in

$$A_c = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -2 & 1 & 0 & -1 & 0 & 0 \\ 1 & -2 & 1 & 0 & -1 & 0 \\ 0 & 1 & -1 & 0 & 0 & -1 \end{pmatrix} \text{ and } B_c = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}.$$

With A_c and B_c being the state transition matrix and input coefficient matrix, respectively, the output coefficient matrix C and the direct path coefficient matrix D are still to be set up. While we can choose D to be a 0, C needs to be constructed so we only get a single output for our multi-input matrix [2]. So we set declare

$$D = 0 \text{ and } C = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

and implement them in MATLAB as stated in the following code listing

```
% construct matrix C and D so they are compatible
% for calculating the transfer function
C = zeros(1, size(Ac,1));
C(length(C)/2) = 1;
D = [0];
```

We remember the state space model[3] representation and the formula for the transfer function

$$\dot{\vec{x}}(t) = A\vec{x}(t) + B\vec{u}(t), \quad \vec{y}(t) = C\vec{x}(t), \quad G_p = \frac{Y(s)}{U(s)}. \tag{6.1.1}$$

In order to get the transfer function we take the Laplace transform on the state space model

$$sX(s) = AX(s) + BU(s) \tag{6.1.2}$$
$$Y(s) = CX(s). \tag{6.1.3}$$

And since we are interested in the relation of output *Y(s)* to input *X(s)* we solve for X

$$\begin{aligned} sX(s) - AX(s) &= BU(s) \\ \Leftrightarrow X(s) &= (sI - A)^{-1}BU(s). \end{aligned}$$

Inserting this into the output equation yields

$$Y(s) = C(sI - A)^{-1}BU(s).$$

Which finally becomes a formula by which we can convert state space models to transfer functions

$$\Leftrightarrow G_p = \frac{Y(s)}{U(s)} = C(sI - A)^{-1}B.$$

This procedure is utilized by MATLAB when using the ss2tf-function. The next step is to calculate first the open-loop transfer function and then the closed loop transfer function. But in order to calculate them we need to set the gains for the PID controller which should be used in our system.

6.2 Controller design

6.2.1 Ziegler-Nichols method

For the scope of this project a heuristic approach was chosen for calculating the individual gains for the PID controller. In this case the *Ziegler-Nichols* approach was used.[4]

This procedure uses the step response of the plant. The values which are necessary for this approximation are derived from the time constant T_g, dead time T_u and gain K_s. Before the approximated values for the controller are calculated the time constants are determined, as seen in Figure 6.2.1

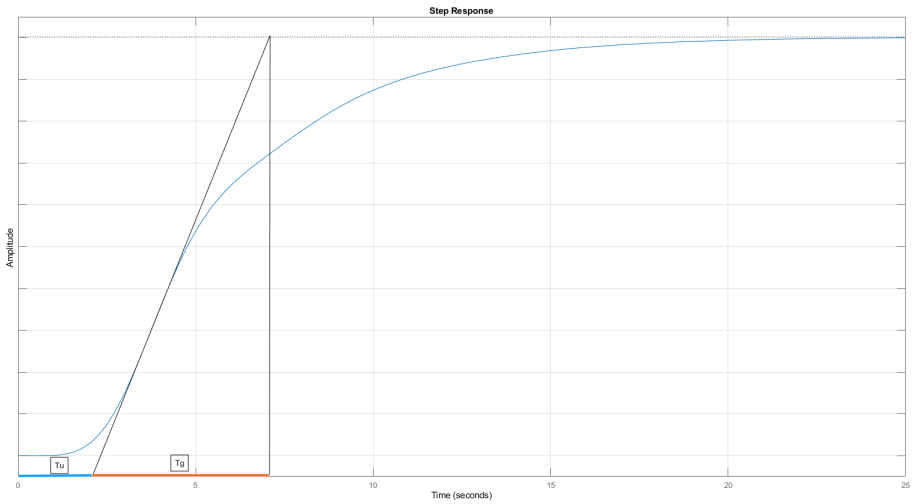


Figure 6.2.1: The step response of the plant transfer function

With

$$K_s = 1, \quad T_u = 2.28 \quad \text{and} \quad T_g = 4.8$$

we set up Equation 6.2.1, 6.2.2 and 6.2.3 as

$$K_p = \frac{1.2}{K_s} \frac{T_g}{T_u} = 2.52, \tag{6.2.1}$$

$$T_n = 2\, T_u, \tag{6.2.2}$$

$$T_V = \frac{1}{2}\, T_u = 1.14. \tag{6.2.3}$$

Following the Ziegler-Nichols method, we can solve

$$T_n = K_p\, T_I \quad \text{for} \quad T_I \tag{6.2.4}$$

$$\text{and } T_v = \frac{T_D}{K_p} \quad \text{for} \quad T_D. \tag{6.2.5}$$

in order to arrange Equation 6.2.6,

$$u(t) = K_p e(t) + \frac{1}{T_I} \int_0^t e(\tau)d\tau + T_D \frac{d}{dt} e(t), \tag{6.2.6}$$

as the differential equation for the PID controller. Equation 6.2.6

$$\Leftrightarrow K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t). \tag{6.2.7}$$

Solving Equations 6.2.4 and 6.2.5 we get the following values for 6.2.7,

$$K_p = 2.52, \quad K_i = 0.5526, \quad K_d = 2.87.$$

After calculating the gains for the PID controller that should be established, the transfer function of the controller G_c, then the open-loop transfer function G_o and finally the closed-loop transfer function G_{cl} are computed as in Listing 6.2.1.

```
1 G_c = pid(Kp, Ki, Kd, 0);
2 % Open Loop transfer function
3 G_o = G_p * G_c;
4 % Closed Loop transfer function
5 G_cl = G_o/(1 + G_o);
```

Investigating the step response of G_{cl} shows strong oscillating behavior as seen in Figure 6.2.2 which implies bad practicalities.

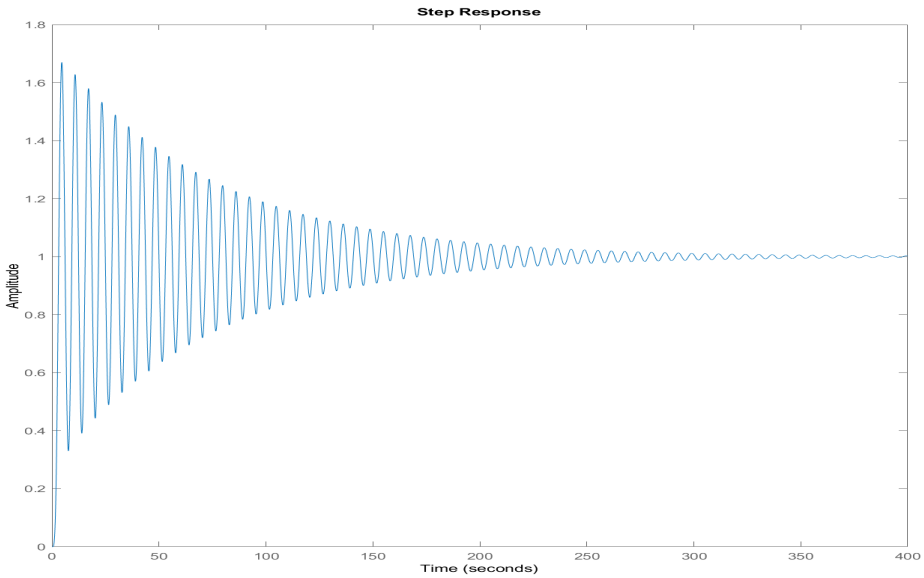


Figure 6.2.2: The step response of the closed-loop transfer function

6.2.2 Iterative method

Another method which seemed to lead to usable results is to find the gains for the transfer functions in an iterative matter. Its implementation is easy and fast but it is clearly flawed. The working principle are three nested for-loops for K_p, K_i and K_d each. Retrieving gains iteratively, calculating the transfer function and finally checking for the poles to assure stability proves itself useless as a method. Because many combinations of gains can prove to feature stable behavior for a given plant, but still be without beneficial properties in terms of non-oscillating behavior or no steady-state errors.

```
sys_p = ss(Ac, Bc, C, D);
G_p = tf(sys_p);

for Kd = 0:0.1:10
    for Ki = 0:0.1:10
        for Kp = 0:0.1:10
            % Time constant of the first-order derivative
            % filter Tf = 0 => no filter on derivative action
            G_c = pid(Kp, Ki, Kd, 0);

            % Open Loop transfer function
            G_o = G_p * G_c;

            % Closed Loop transfer function
            G_cl = G_o/(1 + G_o)

            % calc poles of G_cl
            % poles(G_cl)
            [p,z] = pzmap(G_cl)
            if (p <= 0)
                disp ("Gains found!")
                step(G_cl)
                return;
            end
        end
    end
end
```

For these reasons another method is tested to identify the gains for the to be implemented PID controller.

6.2.3 Random gain probing

Determining gain values of the controller from a range of random numbers is another attempt to design the to be used controller. Hereby the same problem is present as in the other methods. After calculating the closed-loop transfer function the poles are investigated to check for stability. The poles feature all negative valued and conjugated complex results except for one pole with a real and imaginary part of zero which makes it marginally stable. While this would imply that this controller is not suitable in real world applications due to instability in case of noise, calculating the zeros of the transfer function featured a zero on top of the pole which compensates it[5], as seen in Figure 6.2.3.

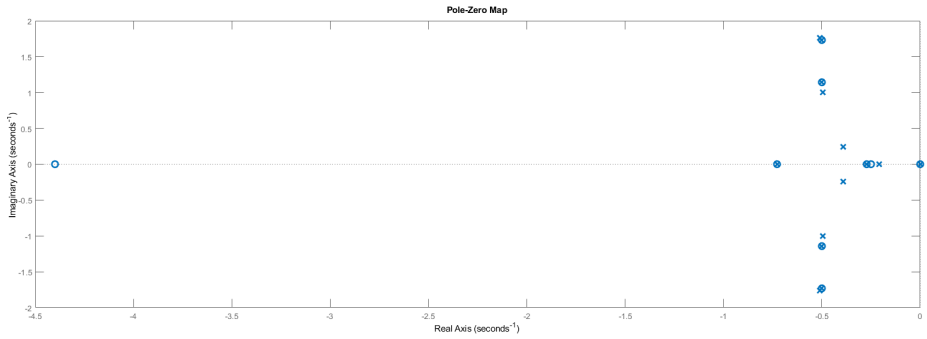


Figure 6.2.3: The system's Pole-Zero-Map

For this reason when checking if the transfer function with the randomly selected gains features such a constellation some conditions are introduced to make sure the pole is compensated by a zero as seen in Listing 6.2.3.

```
while(1)
    % select a random float between 0 and 50
    Kd = 0 + (50-0).*rand();
    Ki = 0 + (50-0).*rand();
    Kp = 0 + (50-0).*rand();

    % Time constant of the first-order derivative filter Tf = 0 =>
    % no filter on derivative action
    G_c = pid(Kp, Ki, Kd, 0);
    % Open Loop transfer function
    G_o = G_p * G_c;
    % Closed Loop transfer function
    G_cl = G_o/(1 + G_o);

    % calc poles of G_cl
    [p,z] = pzmap(G_cl)
    if p <= 0
        if p(1) <= 0 && z(1) <= 0
            % for some reasing connecting the conditions with
            % logical operators didnt work with all the cond.
            if p(2:end) < 0
                if z(2:end) < 0
                    step(G_cl);
                    return;
                end
            end
        end
    end
end
```

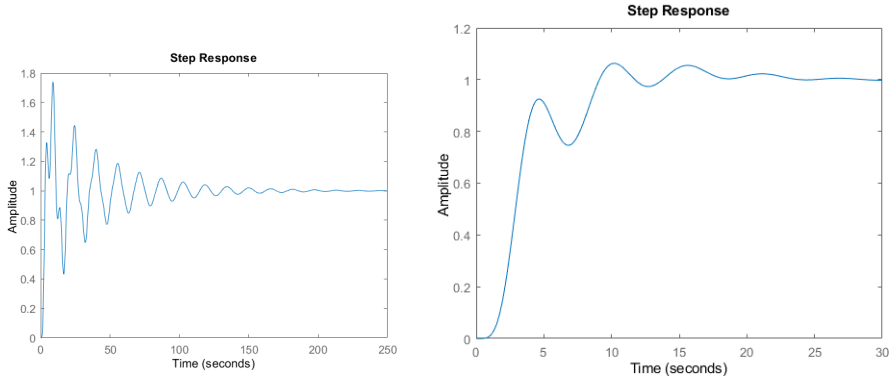


Figure 6.2.4: Examples step responses of closed-loop (K_p, K_i, K_d) left: 3.8, 1.2, 0.83; right: 1.2, 0.3, 1.89

While this methods does not aim to be a reliable way to calculate the gains in its current state, it is able to determine both, gains which lead to a satisfying step response and gains which do not, as seen in Figure 6.2.4 and 6.2.5.

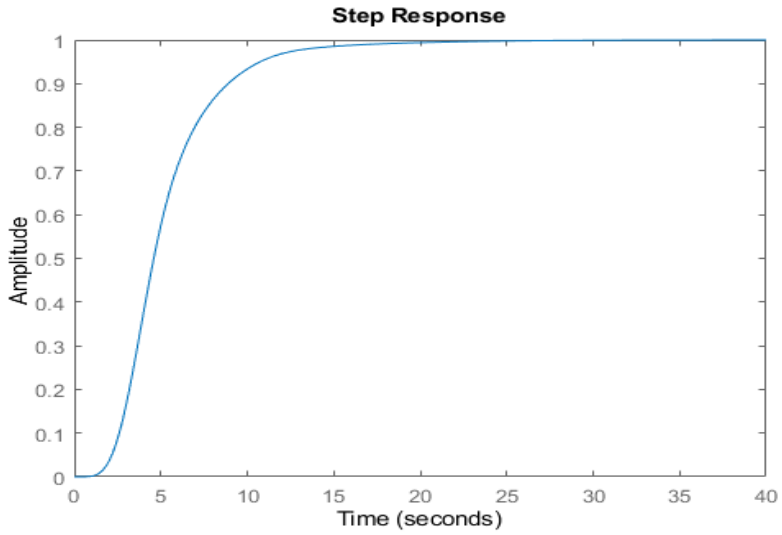


Figure 6.2.5: The step response of the closed-loop transfer function

6.3 Implementation of the PID Controller in C

The goal of introducing a PID controller is to be able to set the voltage over the n-th capacitor for a given reference R. Looking at Figure 6.3.1 it is clear that the controllers output U aims to drive the plant so that its output $Y = R$. This is done by comparing the systems' output Y and the reference R, and applying the resulting error E as feedback to generate a corrective action.

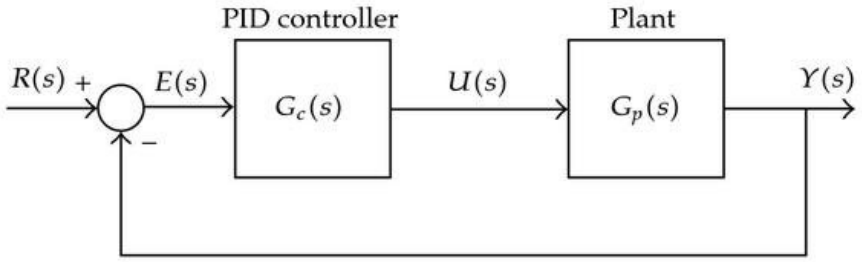


Figure 6.3.1: Closed-loop control system with PID Controller

Implemented as software, the working principle of generating said corrective action, output U, remains. The implemented function needs to solve Equation 6.2.7 but as a discrete time system.[6]

Given $e(t)$ as the continuous-time error signal, the discrete-time error signal $e[k]$ is defined as $e(t)$ sampled at $t = kT_s$ with T_s being the sampling time. With this the integral error-term of the PID control can be approximated to

$$e_i(t) \equiv \int_0^t e(\tau)d\tau \approx \sum_{n=0}^{k-1} e[n]T_s.$$

The derivative error-term of the PID control is approximated using finite differences

$$e_d(t) \equiv \frac{d}{dt}e(t) \approx \frac{e(t) - e(t - T_s)}{T_s} \equiv e_d[k] \equiv \frac{e[k] - e[k-1]}{T_s}.$$

In summery the syntax which represents the three error-terms can be realized as in the following code listing.

```
prev_err = err;
err = ref - y;
int_err = int_err + (err*Ts);
div_err = (err - prev_err)/Ts;

%my c-function called from matlab
u = cpid_mex(0.7922, 0.274, 0.1704, err, int_err, div_err);
```

Additionally, the function for the controller which is executed when called during the MATLAB script is realized in a separate C-file shown in the listing below.

```
#include <stdio.h>
#include "ECSpidmatlab.h"

double main(float Kp, float Ki, float Kd,
            double err, double  div_err, double  int_err) {
    return Kp * err + Ki * int_err + Kd * div_err;
}
```

6.3.1 Software-in-the-loop simulation

We conclude the C-implementation of the PID controller by using it in a software-in-the-loop simulation where setting up the state space model matrices and calculating the transfer function is done in MATLAB, but during the actual simulation MATLAB calls C-functions and hands over the current error values in order to get the output $u[k]$.
This particular loop is composed of the following steps[6]:

- Calculating error by comparing the plant's output and the reference.
- Computing control signal.
- Using the discretized differential equations of the plant to calculate its output $y(kT_s)$.
- updating controller variables.

We discretize the state space model, previously describing our plant for continuous time, with the sampling time T_s. Numerical integration as a numerical method is used to find the approximation to the solution of the differential equation.

The equations

$$\vec{x}_d[k+1] \approx A_d\ \vec{x}_d[k] + B_d\ u$$

and

$$y_d[k+1] \approx D_d\ \vec{x}_d[k+1]$$

are implemented in MATLAB as in the the following listing.

```
x_d(:,i+1) = Ad*x_d(:,i)+Bd*u;
y=Cd*x_d(:,i+1)
```

Running this simulation loop for a reference value of 5 and plotting the output y of the system generated the plot seen in Figure 6.3.2

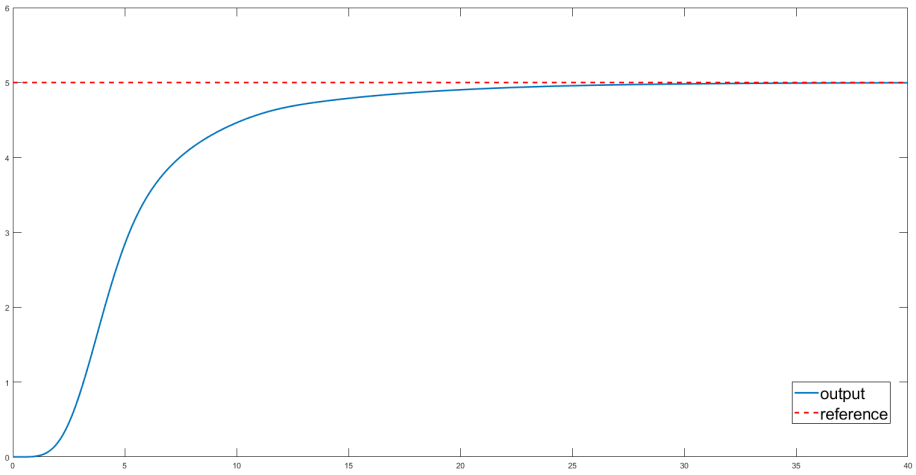


Figure 6.3.2: The system's output

6.4 Conclusion

The plot in Figure 6.3.2 features a response which does not have any over-shoot or oscillating behavior but reaches the set reference value only after around 25 seconds. Considering the inefficient way of acquiring the gains for the controller, its performance is acceptable. Having said this, we have to consider that this was the result for a system composed of only three coupled RLC-circuits. For higher-coupled system a more sophisticated approach for acquiring and tuning the controller would be needed.

7 Part VI: LQR Controller (by Olti Cano)

7.1 Introduction

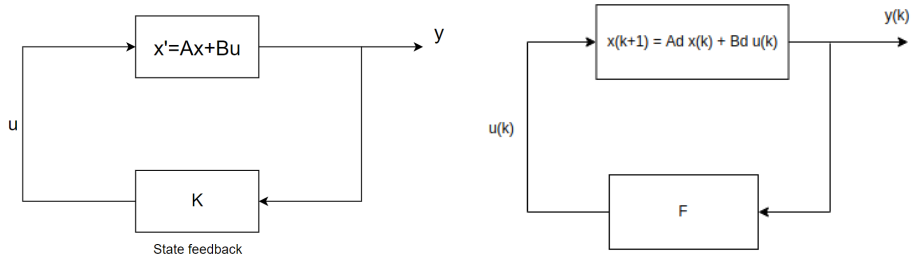
LQR is a robust controller that is designed to minimize the cost function by implementing different adjustable weighting factors. For a continuous-time linear system (see Figure 7.1.1a) $\dot{x}(t) = Ax(t) + Bu(t)$ the cost function is defined as

$$J = \int_0^\infty (x^T Q x + u^T R u) dt. \tag{7.1.1}$$

For a discrete system (see Figure 7.1.1b) $x(k+1) = A_D x(k) + B_D u(k)$ the cost function is defined as

$$J = \sum_{k=0}^{\infty} (x_k^T Q x_k + u_k^T R u_k) \tag{7.1.2}$$

The cost function is defined as the sum of deviations of some measurements from the desired values.



(a) Continuous-time linear system

(b) Discrete system

Figure 7.1.1: System block diagrams

7.2 Continuous-time LQR

In this section, a first order electrical RLC oscillator will be investigated by implementing a continuous-time LQR as shown in Figure 2.1.1.

At first a general system of first order in state space notation is considered

$$\dot{\vec{x}}(t) = \begin{pmatrix} 0 & 1 \\ -\frac{1}{LC} & -\frac{RC}{LC} \end{pmatrix} \vec{x}(t) + \begin{pmatrix} 0 \\ \frac{1}{LC} \end{pmatrix} u(t).$$

In addition, the cost function for a continuous-time is defined. In order to simplify the complexity of the calculations the weighting factor N will be neglected.

$$J = \int_0^\infty (x^T Q x + u^T R u) dt = \int_0^\infty ((x_1 \quad x_2) \begin{pmatrix} q_{11} & 0 \\ 0 & q_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + r u^2) dt,$$

and is equivalent to

$$J = \int_0^\infty (q_{11} x_1^2 + q_{22} x_2^2 + r u^2) dt, \tag{7.2.1}$$

subject to

$$\dot{x}(t) = Ax(t) + Bu(t),$$

with the initial conditions $\mathbf{x(0)=0}$.

In the following the meaning and the appropriate values of each weighting factor is explained.

The $\mathbf{R}$ corresponds to the weighting matrix for the control effort. In all the cases the matrix $\mathbf{R}$ should always contain this properties, $R = R^T > 0$, be Hermitian, invertible and positive semi-definite. Initially the matrix $\mathbf{R}$ is equal to '1' or the identity matrix.

$$R = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = I.$$

The $\mathbf{Q}$ corresponds to the weighting matrix for the performance. In all the cases the matrix $\mathbf{Q}$ should always contain this properties, $Q = Q^T \geq 0$, be positive semi-definite and Hermitian. $\mathbf{Q}$ is usually interpreted as multiple of the identity matrix.

$$Q = C \cdot I = \begin{pmatrix} q_{11} & 0 \\ 0 & q_{22} \end{pmatrix}.$$

7.2.1 State feedback law and Riccati equation

By solving the **Riccati** equation for the matrix **P**, it is possible to calculate the gain factor **K**, which minimizes the cost function **J**. The **Riccati** equation has the form:

$$A^T P + PA - (PB + N)R^{-1}(B^T P + N^T) + Q = 0. \tag{7.2.2}$$

Due to its calculation complexity the matrix **P** will be calculated with the help of MATLAB function called icare. To determine the gain factor **K** the following equation is used

$$K = R^{-1}(B^T P + N^T), \tag{7.2.3}$$

where the feedback control input is $u = -Kx$.

After determining the gain factor, it is possible to define the closed loop equation from

$$\dot{x} = Ax + Bu = Ax + B(-Kx) = (A - BK) \cdot x,$$

and it is equal to

$$A_{cl} = A - BK. \tag{7.2.4}$$

Furthermore, the expression $\dot{x} = A_{cl}x, \;\; x(0) = \begin{pmatrix} x_{01} \\ x_{02} \end{pmatrix}$, is considered, where x_0 corresponds to an initial condition point. In conclusion, optimal control problem is solved to compute the feedback gain **K** to steer the system with feedback law $u(t) = -Kx(t)$ to zero.

7.2.2 The feedforward filter

Finally, a **feedforward** filter is implemented as shown in the Figure 7.2.1. The **v** corresponds to the feedforward input to the system and the state **W** corresponds to the inverse system of closed loop system $W \cdot T = 1$.

For the feedforward filter the state space system

$$\dot{x} = Ax + Bu, \tag{7.2.5}$$

$$y = Cx, \tag{7.2.6}$$

is considered with the feedback control law input

$$u = -Kx + Wv.$$

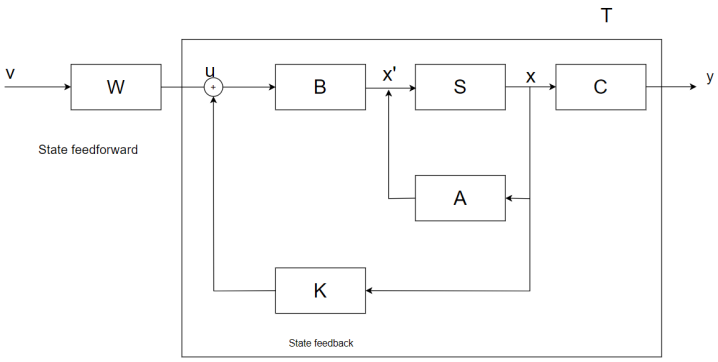


Figure 7.2.1: System block diagram with feedforward filter

By substituting the u on the equation (7.2.5) and we get

$$\dot{x} = (A - Bk)x + BWu. \tag{7.2.7}$$

When $t \to \infty$ the $\dot{x} \to 0$, so the (7.2.7) will be expressed as

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} = (A - BK)x + BWv, \tag{7.2.8}$$

and

$$y(t) = v(t), \tag{7.2.9}$$

which is equal to

$$x(t) = -(A - BK)^{-1} + BW \cdot v. \tag{7.2.10}$$

By substituting equation (7.2.10) into the equation (7.2.6), the resulting equation is

$$y(t) = -C(A - BK)^{-1}BW \cdot v \overset{!}{=} v(t), \tag{7.2.11}$$

with

$$I = -C(A - BK)^{-1}BW.$$

As a result, the equation for the inverse system of closed loop system is achieved

$$W = -[C(A - BK)^{-1}B]^{-1}I. \tag{7.2.12}$$

7.3 Discrete-time LQR

7.3.1 State feedback law and Riccati equation

Same as in the previous section, again a first order electrical RLC oscillator will be investigated by implementing an continuous-time LQR as shown in Figure 2.1.1.

Initially, a general discrete system of first order in state space notation is taken into consideration

$$x(k+1) = e^{A\Delta T} + x(k)\int_0^{\Delta T}(e^{A(\Delta T-\tau)}Bu(\tau))d\tau,$$

$$x(k+1) = e^{A\Delta T} + x(k) + (e^{A\Delta T}\int_0^{\Delta T} e^{-A\tau)}d\tau B)\cdot u(k),$$

$$x(k+1) = A_D x(k) + B_D u(k), \quad x(0) = x_0,$$

$$x(k+1) = \begin{pmatrix} 0 & 1 \\ -\frac{1}{LC} & -\frac{RC}{LC} \end{pmatrix} x(k) + \begin{pmatrix} 0 \\ \frac{1}{LC} \end{pmatrix} u(k).$$

In addition, the cost function for a continuous-time is defined. In order to simplify the complexity of the calculations, the weighting factor N will be neglected. The cost function **J** is given as follows

$$J = \sum_{k=0}^{\infty}(x_k^T Q x_k + u_k^T R u_k) = \sum_{k=0}^{\infty}(\begin{pmatrix} x_{k_1} & x_{k_2} \end{pmatrix}\begin{pmatrix} q_{k_{11}} & 0 \\ 0 & q_{k_{22}} \end{pmatrix}\begin{pmatrix} x_{k_1} \\ x_{k_2} \end{pmatrix} + r u_k^2),$$

which is simplified to the form

$$J = \sum_{k=0}^{\infty}(q_{k_{11}} x_{k_1}^2 + q_{k_{22}} x_{k_2}^2 + r u_k^2). \tag{7.3.1}$$

The optimal feedback control input is given by the expression

$$u_k = -F x_k,$$

and the gain coefficient **F** is given by the following equation

$$F = (R + B_D^T P B_D)^{-1}(B_D^T P A_D). \tag{7.3.2}$$

Same as for the continuous-time LQR, the unique positive definite matrix **P** will be calculated by solving the algebraic **Riccati** equation.

$$P = A^T P A - A^T P B(R + B^T P B)^{-1} B^T P A + Q \tag{7.3.3}$$

Due to high calculation complexity the equation will be calculated with the help of a MATLAB function called idare.

After determining the gain factor **F**, it is possible to define the closed loop equation, which is

$$Ad_{cl} = A_D - B_D \cdot F.$$

Furthermore, the expression $x(k+1) = Ad_{cl}x_k, \;\; x(0) = \begin{pmatrix} x_{01} \\ x_{02} \end{pmatrix}$, is considered, where x_0 corresponds to an initial condition point. In conclusion, optimal control problem is solved to compute the feedback gain **F**, to steer the system with feedback law $u_k = -Fx_k$, to the desired value.

7.3.2 The feedforward filter

Finally, a **feedforward** filter is implemented as shown in the Figure 7.2.1. The **v** corresponds to the feedforward input to the system and the state **W** corresponds to the inverse system of closed loop system $W \cdot T = 1$.

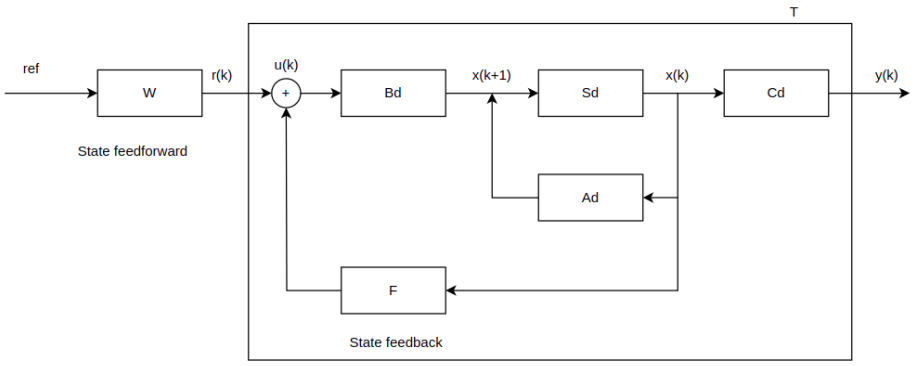


Figure 7.3.1: System block diagram with feedforward filter

For the feedforward filter, we consider the following state space system

$$x(k+1) = A_D x(k) + B_D u(k), \tag{7.3.4}$$

$$y(k) = C_D x(k), \tag{7.3.5}$$

with feedback control law input

$$u(k) = -Fx(k) + Wr(k). \tag{7.3.6}$$

By substituting equation (7.3.6) into the equation (7.3.4), the resulting equation is

$$x(k+1) = (A_D - B_D F)x(k) + B_D W r(k). \tag{7.3.7}$$

From equation (7.3.7), the resulting closed loop equation is,

$$A_D cl = A_D - B_D F.$$

Furthermore, the limits of the continuous and discrete system are defined as in following,

$$\lim_{t\to\infty} \dot{x}(t) = 0 \to \lim_{k\to\infty} x(k+1) = x(k).$$

By substituting the closed loop equation in the equation (7.3.7), we get

$$x(k+1) = A_D cl \cdot x(k) + B_D W r(k) = x(k), \quad for \quad k \to \infty, \tag{7.3.8}$$

which can be rewritten as

$$B_D W r(k) = (I - A_D cl) \cdot x(k).$$

The equation above is solved for the $x(k)$ and is defined as,

$$x(k) = (I - A_D cl)^{-1} B_D W r(k). \tag{7.3.9}$$

On the other side, the output y(k) from equation (7.3.5) is expressed as,

$$y(k) = C_D \cdot x(k) = C_D (I - A_D cl)^{-1} B_D W \cdot r(k) = I \cdot r(k). \tag{7.3.10}$$

Finally, from the equation (7.3.10) the feedback filter equation is achieved and is defined as following

$$W = [C_D (I - A_D cl)^{-1} B_D]^{-1}.$$

8 Part VII: Simulation of LQR for n-order systems (by Olti Cano)

In this section, the simulation of different order systems for continuous and discrete systems will be considered. In addition, the results will be analysed in detail and the controlling system will be adapted in order to increase the efficiency. Furthermore, the changes between the time-continuous and discrete system will be highlighted and explained.

8.1 First order system

8.1.1 First order linear time-continuous system

For this example the values of the RLC circuit are, R=1, L=2 and C=3. The state space representation for this circuit is,

$$A = \begin{pmatrix} 0 & 1 \\ \frac{-1}{LC} & \frac{-(RC)}{(LC)} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ \frac{-1}{6} & \frac{-(1)}{(2)} \end{pmatrix},$$

$$B = \begin{pmatrix} 0 \\ \frac{1}{LC} \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{1}{6} \end{pmatrix},$$

$$C = \begin{pmatrix} 1 & 0 \end{pmatrix},$$

$$R = 1 \;\; and \;\; Q = 100 \cdot I.$$

In addition, by solving the Riccati equation, the unknown factor P is calculated. By substituting the P in the feedback law equation, it is possible to calculate the gain factor K.

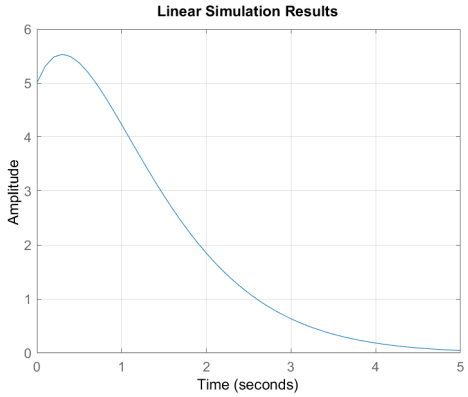
$$K = R^{-1}B^T P = \begin{pmatrix} 9.0499 & 11.7512 \end{pmatrix}.$$

The closed loop for this system is,

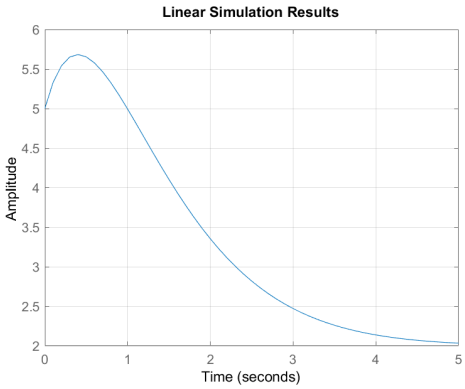
$$Acl = A - BK = \begin{pmatrix} 0 & 1 \\ \frac{-1}{6} & \frac{-(1)}{(2)} \end{pmatrix} - \begin{pmatrix} 0 \\ \frac{1}{6} \end{pmatrix} \begin{pmatrix} 9.0499 & 11.7512 \end{pmatrix}.$$

As a result, the time response of the system can be shown graphically. Furthermore, the feedforward filter is implemented to the system, and the time response of the system is represented graphically.

$$W = -[C(A - BK)^{-1}B]^{-1}I.$$



(a) Time response of Cont. LQR controller



(b) Time response of Cont. LQR with feedforward filter

Figure 8.1.1: Time response of a first order linear time-continuous system

8.1.2 First order discrete system

For this example, the values of the RLC circuit are, R=1, L=2 and C=3. The state space representation for this circuit is,

$$A = \begin{pmatrix} 0 & 1 \\ \frac{-1}{LC} & \frac{-(RC)}{(LC)} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ \frac{-1}{6} & \frac{-(1)}{(2)} \end{pmatrix},$$

$$B = \begin{pmatrix} 0 \\ \frac{1}{LC} \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{1}{6} \end{pmatrix},$$

$$C = \begin{pmatrix} 1 & 0 \end{pmatrix},$$

$$R = 1 \ \ and \ \ Q = 100 \cdot I.$$

The initial point for this example is $x(0) = [5, 4]$.

In order to convert the continuous system to discrete system the c2d function and a sampling time, **Ts=1** as in the following is implemented.

```
Ts=1;
sys_c = ss(A, B, C, 0);
sys_d = c2d(sys_c, Ts) % Convert to discrete time

Ad = sys_d.A
Bd = sys_d.B
Cd = sys_d.C
```

The resulting discrete system defined by MATLAB is

$$A_D = \begin{pmatrix} 0.9299 & 0.7654 \\ -0.1276 & 0.5473 \end{pmatrix},$$

$$B_D = \begin{pmatrix} 0.070073 \\ 0.127558 \end{pmatrix},$$

$$C_D = \begin{pmatrix} 1 & 0 \end{pmatrix},$$

In addition, by solving the algebraic Riccati equation for discrete time systems, the unknown factor P is calculated. By substituting the P in the feedback law equation, it is possible to calculate the gain factor F.

$$F = (R + B_D^T P B_D)^{-1}(B_D^T P A_D) = \begin{pmatrix} 3.1892 & 5.4108 \end{pmatrix}.$$

The closed loop for this system is,

$$\begin{aligned} A_D cl &= A_D - B_D F, \\ A_D cl &= \begin{pmatrix} 0.9299 & 0.7654 \\ -0.1276 & 0.5473 \end{pmatrix} - \begin{pmatrix} 0.070073 \\ 0.127558 \end{pmatrix} \begin{pmatrix} 3.1892 & 5.4108 \end{pmatrix}, \\ A_D cl &= \begin{pmatrix} 0.7064 & 0.3862 \\ -0.5344 & -0.1429 \end{pmatrix}. \end{aligned}$$

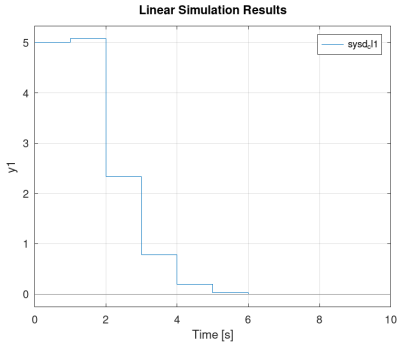
As a result the time response of the system can be shown graphically in the Figure 8.1.2a.

Furthermore, the feedforward filter is implemented to the system, and the time response of the system is represented graphically in the Figure 8.1.2b.

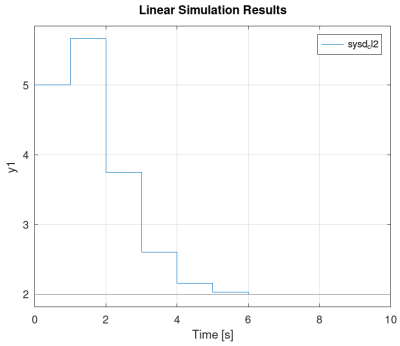
$$W = -[C(A - BK)^{-1}B]^{-1} I = 4.1892.$$

The figures show graphically how the system is steered to the desired value in a certain time. The MATLAB function used to draw these graphs is called lsim.

```
% time response of dynamic systems for arbitrary inputs.
lsim(sysd_cl1, u_in_1, tsteps, [5,4])
```



(a) Time response of Dis. LQR controller



(b) Time response of Dis. LQR with feedforward filter

Figure 8.1.2: Time response of first order discrete system

8.2 Third order system

8.2.1 Third order linear time-continuous system

For this example the values of the RLC circuit are,

$$R1 = 1, \;\; R2 = 3, \;\; R3 = 2,$$

$$L1 = 2, \;\; L2 = 1, \;\; L3 = 3,$$

$$C1 = 3, \;\; C2 = 2, \;\; C3 = 1.$$

The state space representation for this circuit is,

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -1/2 & 1/3 & 0 & -1/2 & 5/3 & 5/6 \\ 1/2 & -2/3 & 1/6 & 0 & -3 & -7/6 \\ 0 & 1/3 & -1/3 & 0 & 0 & -2/3 \end{pmatrix},$$

$$B = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1/6 \\ 0 \\ 0 \end{pmatrix},$$

$$C = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$R = 1 \;\; and \;\; Q = 100 \cdot I.$$

In addition, by solving the Riccati equation, the unknown factor P is calculated. By substituting the P in the feedback law equation, it is possible to calculate the gain factor K.

$$K = R^{-1} B^T P =$$

$$= \begin{pmatrix} 32.3007 & 14.9618 & 6.5188 & 34.3712 & 12.1114 & 5.3700 \end{pmatrix}.$$

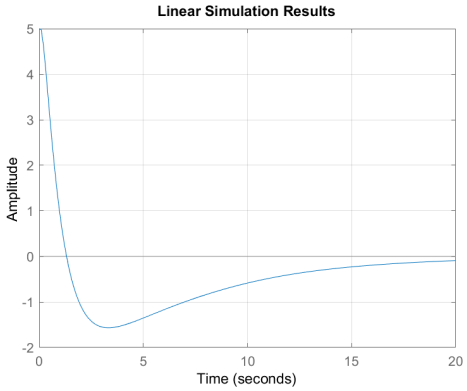
The closed loop for this system is,

$$Acl = A - BK$$

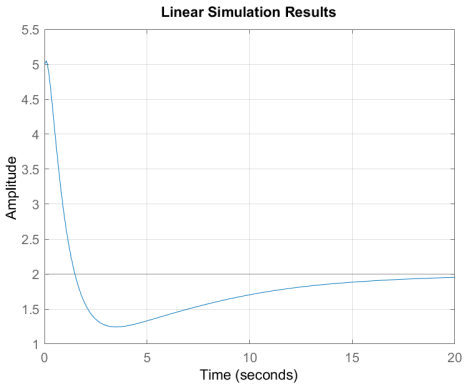
$$= \begin{pmatrix} 0 & 0 & 0 & 1.0000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.0000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.0000 \\ -5.8835 & -2.1603 & -1.0865 & -6.2285 & -0.3519 & -0.0617 \\ 0.5000 & -0.6667 & 0.1667 & 0 & -3.0000 & -1.1667 \\ 0 & 0.3333 & -0.3333 & 0 & 0 & -0.6667 \end{pmatrix}.$$

As a result the time response of the system can be shown graphically. Furthermore, the feedforward filter is implemented to the system, and the time response of the system is represented graphically.

$$W = -[C(A - BK)^{-1}B]^{-1}I = 54.7814.$$



(a) Time response of Cont. LQR controller



(b) Time response of Cont. LQR with feedforward filter

Figure 8.2.1: Time response of a third order linear time-continuous system

8.2.2 Third order discrete system

For this example the values of the RLC circuit are,

$$R1 = 1, \quad R2 = 3, \quad R3 = 2,$$

$$L1 = 2, \quad L2 = 1, \quad L3 = 3,$$

$$C1 = 3, \quad C2 = 2, \quad C3 = 1.$$

The state space representation for this circuit is,

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -1/2 & 1/3 & 0 & -1/2 & 5/3 & 5/6 \\ 1/2 & -2/3 & 1/6 & 0 & -3 & -7/6 \\ 0 & 1/3 & -1/3 & 0 & 0 & -2/3 \end{pmatrix},$$

$$B = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1/6 \\ 0 \\ 0 \end{pmatrix},$$

$$C = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$R = 1 \ \ and \ \ Q = 100 \cdot I.$$

The initial point for this example is $x(0) = [5, 4, 3, 2, 1, 0]$.

In order to convert the continuous system to discrete system the c2d function and a sampling time, **Ts=1** as in the following is implemented.

```
Ts=1;
sys_c = ss(A, B, C, 0);
sys_d = c2d(sys_c, Ts) % Convert to discrete time

Ad = sys_d.A
Bd = sys_d.B
Cd = sys_d.C
```

The resulting discrete system defined by MATLAB is

$$A_D = \begin{pmatrix} 0.860457 & 0.0695239 & 0.00116966 & 0.740266 & 0.315229 & 0.145833 \\ 0.104286 & 0.830741 & 0.0631371 & 0.0376392 & 0.293281 & -0.171173 \\ 0.00350898 & 0.126274 & 0.870195 & 0.000720012 & 0.0242831 & 0.683473 \\ -0.212519 & 0.0852139 & 0.00392719 & 0.490324 & 0.357615 & 0.15307 \\ 0.127821 & -0.240032 & 0.105938 & 0.0854663 & 0.0136298 & -0.133543 \\ 0.0117816 & 0.211876 & -0.223777 & 0.00314898 & 0.0546248 & 0.386816 \end{pmatrix},$$

$$B_D = \begin{pmatrix} 0.0688492 \\ 0.00183646 \\ 2.19642e-05 \\ 0.123378 \\ 0.0062732 \\ 0.000120002 \end{pmatrix},$$

$$C_D = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

In addition, by solving the algebraic Riccati equation for discrete time systems, the unknown factor P is calculated. By substituting the P in the feedback law equation, it is possible to calculate the gain factor F.

$$F = (R + B_D^T P B_D)^{-1}(B_D^T P A_D) =$$

$$= \begin{pmatrix} 3.1245 & 2.1774 & 0.899275 & 5.40523 & 3.50601 & 1.56896 \end{pmatrix}.$$

The closed loop for this system is,

$$A_D cl = A_D - B_D F =$$

$$= \begin{pmatrix} 0.645338 & -0.0803886 & -0.0607448 & 0.36812 & 0.0738427 & 0.0378113 \\ 0.0985479 & 0.826742 & 0.0614856 & 0.0277127 & 0.286842 & -0.174054 \\ 0.00344036 & 0.126226 & 0.870175 & 0.00060129 & 0.0242061 & 0.683438 \\ -0.598012 & -0.183429 & -0.107023 & -0.176561 & -0.074948 & -0.0405047 \\ 0.10822 & -0.253691 & 0.100296 & 0.0515582 & -0.00836409 & -0.143385 \\ 0.0114066 & 0.211614 & -0.223885 & 0.00250034 & 0.0542041 & 0.386628 \end{pmatrix}$$

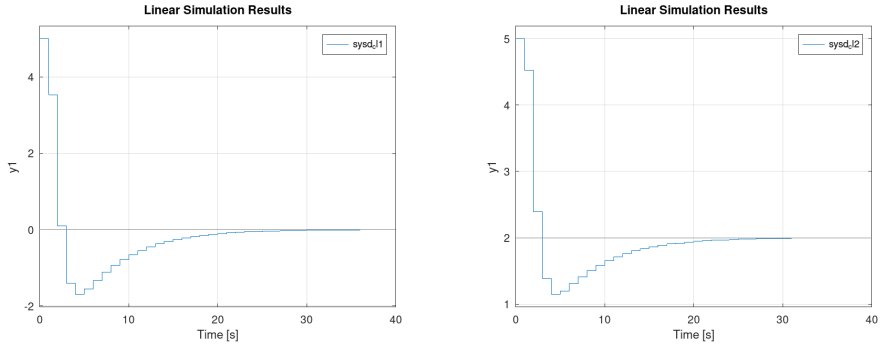
As a result the time response of the system can be shown graphically in the Figure 8.2.2a.

Furthermore, the feedforward filter is implemented to the system, and the time response of the system is represented graphically in the Figure 8.2.2b.

$$W = -[C(A - BK)^{-1}B]^{-1}I = 7.20118.$$

The figures show graphically how the system is steered to the desired value in a certain time. The MATLAB function used to draw these graphs is called lsim.

```
% time response of dynamic systems for arbitrary inputs.
lsim(sysd_cl1, u_in_1, tsteps, [5,4,3,2])
```



(a) Time response of Dis. LQR controller

(b) Time response of Dis. LQR with feedforward filter

Figure 8.2.2: Time response of a third order discrete system

8.3 Fourth order system

8.3.1 Fourth order linear time-continuous system

For this example the values of the RLC circuit are,

$$R1 = 1, \;\; R2 = 3, \;\; R3 = 2, \;\; R4 = 4,$$

$$L1 = 2, \;\; L2 = 1, \;\; L3 = 3, \;\; L4 = 4,$$

$$C1 = 3, \;\; C2 = 2, \;\; C3 = 1, \;\; C4 = 4.$$

The state space representation for this circuit is,

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -1/2 & 1/3 & 0 & 0 & -1/2 & 5/3 & 5/6 & 10/3 \\ 1/2 & -2/3 & 1/6 & 0 & 0 & -3 & -7/6 & -14/3 \\ 0 & 1/3 & -7/12 & 1/4 & 0 & 0 & -2/3 & 4/3 \\ 0 & 0 & 1/16 & -1/16 & 0 & 0 & 0 & -1 \end{pmatrix},$$

$$B = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1/6 \\ 0 \\ 0 \\ 0 \end{pmatrix},$$

$$C = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$R = 1 \;\; and \;\; Q = 100 \cdot I.$$

In addition, by solving the Riccati equation, the unknown factor P is calculated. By substituting the P in the feedback law equation, it is possible to calculate the gain factor K.

$$K = R^{-1} B^T P =$$

$$= \begin{pmatrix} 31.7695 & 12.0004 & 4.3627 & 14.1209 & 34.2858 & 11.1200 & 3.3186 & 8.6996 \end{pmatrix}.$$

The closed loop for this system is,

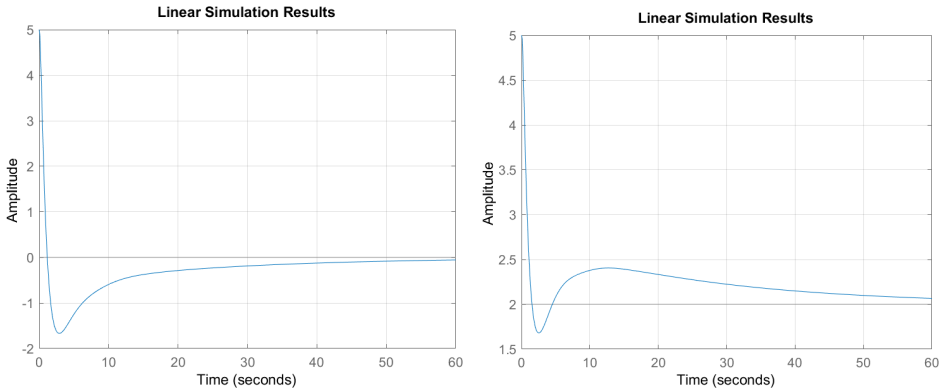
$$Acl = A - BK =$$

$$= \begin{pmatrix} 0 & 0 & 0 & 0 & 1.0000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.0000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.0000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0000 \\ -5.7949 & -1.6667 & -0.7271 & -2.3535 & -6.2143 & -0.1867 & 0.2802 & 1.8834 \\ 0.5000 & -0.6667 & 0.1667 & 0.0000 & 0 & -3.0000 & -1.1667 & -4.6667 \\ 0 & 0.3333 & -0.5833 & 0.2500 & 0 & 0 & -0.6667 & 1.3333 \\ 0 & 0 & 0.0625 & -0.0625 & 0 & 0 & 0 & -1.0000 \end{pmatrix}.$$

As a result the time response of the system can be shown graphically.

Furthermore, the feedforward filter is implemented to the system, and the time response of the system is represented graphically.

$$W = -[C(A - BK)^{-1}B]^{-1}I = 63.2535.$$



(a) Time response of Con. LQR controller

(b) Time response of Dis. LQR with feedforward filter

Figure 8.3.1: Time response of a fourth order linear time-continuous system

8.3.2 Fourth order discrete system

For this example the values of the RLC circuit are,

$$R1 = 1, \quad R2 = 3, \quad R3 = 2, \quad R4 = 4,$$

$$L1 = 2, \quad L2 = 1, \quad L3 = 3, \quad L4 = 4,$$

$$C1 = 3, \quad C2 = 2, \quad C3 = 1, \quad C4 = 4.$$

The state space representation for this circuit is,

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -1/2 & 1/3 & 0 & 0 & -1/2 & 5/3 & 5/6 & 10/3 \\ 1/2 & -2/3 & 1/6 & 0 & 0 & -3 & -7/6 & -14/3 \\ 0 & 1/3 & -7/12 & 1/4 & 0 & 0 & -2/3 & 4/3 \\ 0 & 0 & 1/16 & -1/16 & 0 & 0 & 0 & -1 \end{pmatrix},$$

$$B = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1/6 \\ 0 \\ 0 \\ 0 \end{pmatrix},$$

$$C = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

$$R = 1 \ \ and \ \ Q = 100 \cdot I.$$

The initial point for this example is $x(0) = [5, 4, 3, 2, 1, 0, -1, -2]$.

In order to convert the continuous system to discrete system the c2d function and a sampling time, **Ts=1** as in the following is implemented.

```
Ts=1;
sys_c = ss(A, B, C, 0);
sys_d = c2d(sys_c, Ts) % Convert to discrete time

Ad = sys_d.A
Bd = sys_d.B
Cd = sys_d.C
```

The resulting discrete system defined by MATLAB is

$$A_D = \begin{pmatrix} 0.860 & 0.069 & 0.001 & 0 & 0.740 & 0.315 & 0.145 & 0.582 \\ 0.104 & 0.830 & 0.061 & 0.001 & 0.037 & 0.293 & -0.170 & -0.755 \\ 0.003 & 0.123 & 0.785 & 0.087 & 0 & 0.023 & 0.656 & 0.368 \\ 7.63e-06 & 0 & 0.021 & 0.977 & 1.107e-06 & 8.33e-05 & 0.006 & 0.627 \\ -0.212 & 0.085 & 0.003 & 5.48e-05 & 0.490 & 0.357 & 0.153 & 0.608 \\ 0.127 & -0.240 & 0.101 & 0.004 & 0.085 & 0.013 & -0.132 & -0.714 \\ 0.011 & 0.203 & -0.359 & 0.141 & 0.003 & 0.053 & 0.320 & 0.485 \\ 4.11e-05 & 0.002 & 0.035 & -0.037 & 7.08e-06 & 0 & 0.017 & 0.358 \end{pmatrix},$$

$$B_D = \begin{pmatrix} 0.0688492 \\ 0.00183646 \\ 2.18657e-05 \\ 2.4736e-08 \\ 0.123378 \\ 0.0062732 \\ 0.000119267 \\ 1.84599e-07 \end{pmatrix},$$

$$C_D = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

In addition, by solving the algebraic Riccati equation for discrete time systems, the unknown factor P is calculated. By substituting the P in the feedback law equation, it is possible to calculate the gain factor F.

$$F = (R + B_D^T P B_D)^{-1}(B_D^T P A_D) =$$

$$= \begin{pmatrix} 3.030 & 1.799 & 0.6444 & 1.897 & 5.358 & 3.365 & 1.337 & 4.751A \end{pmatrix}.$$

The closed loop for this system is,

$$A_D cl = A_D - B_D F =$$

$$= \begin{pmatrix} 0.652 & -0.054 & -0.043 & -0.131 & 0.371 & 0.084 & 0.054 & 0.255 \\ 0.099 & 0.827 & 0.061 & -0.002 & 0.028 & 0.287 & -0.174 & -0.765 \\ 0.003 & 0.124 & 0.785 & 0.087 & 0.001 & 0.024 & 0.657 & 0.368 \\ 0 & 0.001 & 0.022 & 0.978 & 0 & 0 & 0.007 & 0.628 \\ -0.586 & -0.137 & -0.076 & -0.234 & -0.171 & -0.058 & -0.012 & 0.022 \\ 0.109 & -0.251 & 0.098 & -0.008 & 0.052 & -0.007 & -0.143 & -0.744 \\ 0.011 & 0.203 & -0.356 & 0.141 & 0.002 & 0.053 & 0.32 & 0.485 \\ 0 & 0.002 & 0.035 & -0.038 & 0 & 0 & 0.017 & 0.358 \end{pmatrix}.$$

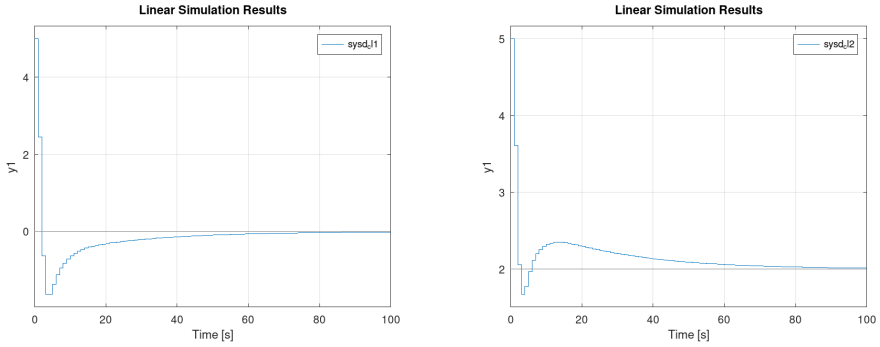
As a result the time response of the system can be shown graphically in the Figure 8.2.2a.

Furthermore, the feedforward filter is implemented to the system, and the time response of the system is represented graphically in the Figure 8.2.2b.

$$W = -[C(A - BK)^{-1}B]^{-1} I = 8.37063.$$

The Figures show graphically how the system is steered to the desired value in a certain time. The MATLAB function used to draw these graphs is called lsim.

```
% time response of dynamic systems for arbitrary inputs.
lsim(sysd_cl1, u_in_1, tsteps, [5,4,3,2,1,0,-1,-2])
```



(a) Time response of Dis. LQR controller

(b) Time response of Dis. LQR with feedforward filter

Figure 8.3.2: Time response of a fourth order discrete system

8.4 Conclusions

8.4.1 Implementation of the feedforward filter

The implementation of the feedforward filter into the oscillating system maintains a more stabilized behaviour by preventing the problems before they occur.

Another important reason of using the feedforward filter is the possibility to steer the system to the desired set of points.

Using only the feedback system is not sufficient for building a good controlling system especially in critical controlling tasks such as air-crafts, driving assistance etc.

8.4.2 Time continuous vs. Time discrete system

Nowadays, the time continuous controllers are found only in few technical applications. They are usually implemented as electronics (via OP-Amps or FPGAs).

Usually, the time continuous systems are implemented in safety-critical systems (air-crafts, ADAS, etc.) because circuits can be tested more easily than software and pure electronics are rather real-time capable.

However, implementing a controller in electronics results in high financial and time costs. Alternatively, it is much easier to rewrite some lines of code instead of redesigning and reproducing a circuit.

The time discrete controllers are mostly implemented in the standard applications. Modern controllers are becoming more complex because techniques such as Artificial Neural Networks controllers are being implemented. As a consequence, implementing this type of controllers in electronics is very complicated and requires a lot of effort.

References

[1] *Control systems/state-space stability*, Wikibooks, 07.01.2022. [Online]. Available: https://en.wikibooks.org/wiki/Control_Systems/State-Space_Stability.

[2] J. O. S. III, *Introduction to linear state space models*, 2019. [Online]. Available: https://ccrma.stanford.edu/~jos/StateSpace/StateSpace_4up.pdf.

[3] P. S. Solvang, *State space model based pid controller tuning*, Master's Thesis 2019 Industrial IT and Automation, 2019.

[4] *Faustformelverfahren (automatisierungstechnik)*. [Online]. Available: https://de.wikipedia.org/wiki/Faustformelverfahren_(Automatisierungstechnik).

[5] *Regelkreis, bedeutung der pole und nullstellen der übertragungsfunktion eines übertragungssystems*. [Online]. Available: https://de.wikipedia.org/wiki/Regelkreis#Bedeutung_der_Pole_und_Nullstellen_der_\%C3\%9Cbertragungsfunktion_eines_\%C3\%9Cbertragungssystems.

[6] *Pid controller*. [Online]. Available: https://en.wikipedia.org/wiki/PID_controller#PID_tutorials.