

Lecture 18: Digital Implementation of Continuous-Time Controllers

Cleveland State University
Mechanical Engineering
Hanz Richter, PhD

MCE441 – p.1/10

Digital vs. Continuous (Analog) Control

Suppose we found a “satisfactory” control transfer function by classical design techniques followed by simulation studies:

$$K(s) = 2 + \frac{3}{s}$$

Since this is a PI controller, we may just buy an analog PID box off the shelf and tune the proportional gain to 2 and the integral gain to 3.

Internally, the analog PID is made of resistors, inductors, capacitors and other physical devices capable of “materializing” the control TF, from error to control input.

What if the controller was

$$K(s) = \frac{2s^5 + 3s^4 + s^3 + s^2 + 2s + 1}{s^2(s^2 + s + 1)(s + 8)}$$

Can we find off-the-shelf analog circuitry which has this TF?

MCE441 – p.2/10

Digital vs. Continuous (Analog) Control

On the other hand, we may want to use a digital computer to realize the behavior of our control transfer function between error and control input. This affords a lot of flexibility, since changes to control parameters (or mathematical form) are simply done by editing lines of code.

Since digital computers operate on discrete time (a defined paced cycle for sensing, computation and actuation, which cannot happen simultaneously), certain continuous mathematical operations like differentiation and integration must be performed by approximation, in what is called *discretization*.

MCE441 – p.3/10

Discretizing derivatives

Suppose we are using a PID law which contains the term \dot{e} . One way to approximate \dot{e} is to use the *backward difference rule*:

$$\dot{e} \approx \frac{e(t) - e(t - T)}{T}$$

In other words, the cyclic part of the code might look like:

```
read current_r %from the user command
read current_y %from the sensor
current_e=current_r-current_y;
edot=(current_e-previous_e)/T;
.....
.....
previous_e=current_e
```

MCE441 – p.4/10

Discretizing integrals

The integral term can be approximated (crudely) by using the *rectangular rule*:

```
read current_r %from the user command
read current_y %from the sensor
current_e=current_r-current_y;
current_eint=previous_eint+current_e*T;
.....
.....
previous_eint=current_eint
```

MCE441 – p.5/10

Discretization of state derivatives

Suppose our control TF is converted to state-space format:

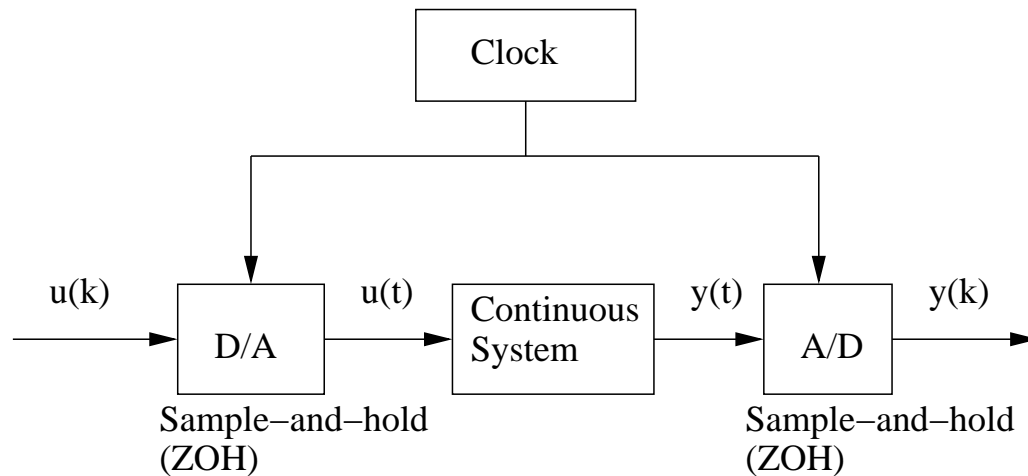
$$\begin{aligned}\dot{x}_1 &= a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n + b_1e \\ \dot{x}_2 &= a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n + b_2e \\ &\vdots \\ \dot{x}_n &= a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n + b_ne \\ u &= c_1x_1 + c_2x_2 + \dots + c_nx_n + d_1e_1 + d_2e_2 + \dots + d_ne_n\end{aligned}\tag{1}$$

Then we can discretize each one of the derivatives and create recursive expressions of the form

```
u_ctrl=h(x_1,x_2,...x_n,e);
x_1_next=f_1(x_1,x_2,...x_n,e);
...
x_n_next=f_n(x_1,x_2,...x_n,e);
```

MCE441 – p.6/10

Zero-Order Hold Discretization



MCE441 – p.7/10

Zero-Order Hold Discretization

A discretization method that matches the timed sample-and-hold process in a digital processor is the Zero-Order Hold (ZOH) method. In this introductory course we are only concerned with obtaining the ZOH discretization of a given continuous-time TF or SS system by using Matlab:

```
>>sysK=tf(numK,denK) %suppose you have a control
%transfer function [numK,denK]
>>Ts=1e-3; %define the sampling period
>>sysK_d=c2d(sysK,Ts,'zoh'); %we go from continuous to
%discrete using the ZOH method
>>[Ad,Bd,Cd,Dd]=ssdata(sysK_d); %extract the state-space
%matrices for a state-space implementation
```

MCE441 – p.8/10

A Word of Caution

Designing the controller in continuous-time, followed by discretization and deployment is called *emulation design*.

Many factors outside the scope of this course, including sample rate selection, determine whether the implemented controller will operate with a performance similar to the one seen in continuous-time simulations.

Ideally, one should discretize the plant TF and conduct the design using digital techniques (*direct design*). (MCE603/703)

It can be safely said, however, that using a high sample rate in the control implementation reduces the possibilities of failure when using emulation design.

MCE441 – p.9/10

Demonstration-DC Servomotor Control

We now demonstrate the operations of discretizing a TF designed in continuous-time, conducting a simulation to validate the emulation design under digital operation and finally deploying the controller in real time.

MCE441 – p.10/10