Introduction to feedback linearization under digital control

PART I

Mohamed Elobaid

March 24, 2023

What this talk is

- Basic, introductory and somewhat informal
- Feedback linearization in continuous time
- Why (Emulation) holding the control constant during the sampling period may not be enough
- Feedback linearization redesign in the digital domain
- Examples (motivations and applications)

What this talk is not

- Comprehensive and exhaustive
- Proofs focused (although some formal statements will be recalled)
- 1^{st}, higher order and generalized holding schemes
- Rigorous

Prelude

A motivating example

For the system;

$$
\begin{aligned}
\dot{x}_1 &= \frac{-(x_2 - \sin x_1)}{\sin x_1 - 2} \\
\dot{x}_2 &= -2x_1 - x_2 - \frac{\cos x_1 (x_2 - \sin x_1)}{\sin x_1 - 2} - \cos x_1 + \sin x_1 \\
&\quad + (2x_1 + \cos x_1)(x_2 - \sin x_1) + u \\
y &= 2x_1 + \cos x_1
\end{aligned}
$$

Find a control u to make the output follow a given reference y_d

A motivating example

Looking at the output

$$\begin{aligned}
\dot{x}_1 &= \frac{-(x_2 - \sin x_1)}{\sin x_1 - 2} \\
\dot{x}_2 &= -2x_1 - x_2 - \frac{\cos x_1 (x_2 - \sin x_1)}{\sin x_1 - 2} - \cos x_1 + \sin x_1 \\
&\quad + (2x_1 + \cos x_1)(x_2 - \sin x_1) + u \\
&\boxed{y = 2x_1 + \cos x_1}
\end{aligned}$$

A motivating example

Looking at the output

$$
\begin{aligned}
\dot{x}_1 &= \frac{-(x_2 - \sin x_1)}{\sin x_1 - 2} \\
\dot{x}_2 &= -2x_1 - x_2 - \frac{\cos x_1 (x_2 - \sin x_1)}{\sin x_1 - 2} - \cos x_1 + \sin x_1 \\
&\quad + (2x_1 + \cos x_1)(x_2 - \sin x_1) + u \\
&\boxed{y = 2x_1 + \cos x_1} \\
\dot{y} &= 2\dot{x}_1 - \sin x_1 \dot{x}_1 \\
&= \frac{-2(x_2 - \sin x_1)}{\sin x_1 - 2} + \frac{\sin x_1 (x_2 - \sin x_1)}{\sin x_1 - 2} \qquad \text{not a function of } u
\end{aligned}
$$

A motivating example

Looking at the output

$$
\begin{aligned}
\dot{x}_1 &= \frac{-(x_2 - \sin x_1)}{\sin x_1 - 2} \\
\dot{x}_2 &= -2x_1 - x_2 - \frac{\cos x_1 (x_2 - \sin x_1)}{\sin x_1 - 2} - \cos x_1 + \sin x_1 \\
&\quad + (2x_1 + \cos x_1)(x_2 - \sin x_1) + u \\
&\boxed{y = 2x_1 + \cos x_1} \\
\dot{y} &= 2\dot{x}_1 - \sin x_1 \dot{x}_1 \\
&= \frac{-2(x_2 - \sin x_1)}{\sin x_1 - 2} + \frac{\sin x_1 (x_2 - \sin x_1)}{\sin x_1 - 2} \\
&= x_2 - \sin x_1 \\
\ddot{y} &= \color{red}{\dot{x}_2} \color{black}{- \dot{x}_1 \cos x_1} \quad \color{red}{\text{a function of the control } u} \color{black}{\implies r = 2}
\end{aligned}
$$

A motivating example

Coordinates change $z = \phi(x)$ is a *Homeomorphism*, preferably a *Diffeomorphism*

$$\dot{x}_1 = \frac{-(x_2 - \sin x_1)}{\sin x_1 - 2}$$

$$\dot{x}_2 = -2x_1 - x_2 - \frac{\cos x_1 (x_2 - \sin x_1)}{\sin x_1 - 2} - \cos x_1 + \sin x_1 + (2x_1 + \cos x_1)(x_2 - \sin x_1) + u$$

$$\boxed{y = 2x_1 + \cos x_1}$$

$$\boxed{\dot{y} = x_2 - \sin x_1}$$

$$z_1 = y = 2x_1 + \cos x_1$$

$$z_2 = \dot{y} = x_2 - \sin x_1$$

A motivating example

Coordinates change $z = \phi(x)$ is a *Homeomorphism*, preferably a *Diffeomorphism*

$$
\begin{aligned}
\dot{x}_1 &= \frac{-(x_2 - \sin x_1)}{\sin x_1 - 2} \\
\dot{x}_2 &= -2x_1 - x_2 - \frac{\cos x_1 (x_2 - \sin x_1)}{\sin x_1 - 2} - \cos x_1 + \sin x_1 \\
&\quad + (2x_1 + \cos x_1)(x_2 - \sin x_1) + u \\
&\boxed{y = 2x_1 + \cos x_1} \\
z = \phi(x) &= \begin{pmatrix} 2x_1 + \cos x_1 \\ x_2 - \sin x_1 \end{pmatrix}, \qquad D_{\phi(x)} = \begin{pmatrix} 2 - \sin x_1 & 0 \\ -\cos x_1 & 1 \end{pmatrix}
\end{aligned}
$$

Defined almost everywhere !

A motivating example

Writing the system in the new coordinates z

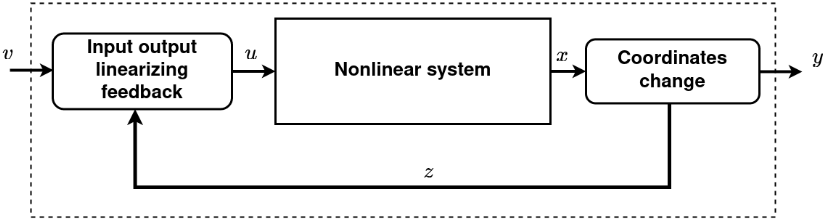
$$\dot{z}_1 = z_2$$
$$\dot{z}_2 = -z_1 - z_2 + z_1 z_2 + u$$
$$\boxed{y = z_1}$$

Now we can apply input-output feedback linearization

$$u = z_1 + z_2 - z_1 z_2 + v$$

A motivating example

$$\begin{aligned} \dot{z}_1 &= z_2 \\ \dot{z}_2 &= v \\ y &= z_1 \end{aligned}$$



$$y = \frac{1}{s^2} v$$

Figure: Input output link is rendered linear (a double integrator)

A motivating example

Matlab break

Interlude

Why did emulation fail: ex cont.

Looking at the system after the coordinates change;

$$
\begin{aligned}
\dot{z}_1 &= z_2 \\
\dot{z}_2 &= -z_1 - z_2 + z_1 z_2 + u \\
\Aboxed{y &= z_1}
\end{aligned}
$$

It admits the following Taylor expansion at the sampling instants (sampling rate δ)

$$
\begin{aligned}
z_1(k+1) &= z_1(k) + \delta \dot{z}_1(k) + \frac{\delta^2}{2!}\ddot{z}_1(k) + \dots \\
&= z_1(k) + \delta z_2(k) + \frac{\boldsymbol{\delta^2}}{2!}\left(-z_1(k) - z_2(k) + z_1(k) z_2(k) + {\color{red}u(k)}\right) + {\color{blue}O(\delta^3)} \\
z_2(k+1) &= z_2(k) + \delta\left(-z_1(k) - z_2(k) + z_1(k) z_2(k) + {\color{red}u(k)}\right) + {\color{blue}O(\delta^2)} \\
y(k) &= z_1(k)
\end{aligned}
$$

Why did emulation fail: ex cont.

$$
\begin{aligned}
z_1(k+1) &= z_1(k) + \delta \dot{z}_1(k) + \frac{\delta^2}{2!}\ddot{z}_1(k) + \dots \\
&= z_1(k) + \delta z_2(k) + \frac{\boldsymbol{\delta^2}}{2!}\left(-z_1(k) - z_2(k) + z_1(k)z_2(k) + {\color{red}u(k)}\right) + {\color{blue}O(\delta^3)} \\
z_2(k+1) &= z_2(k) + \delta\left(-z_1(k) - z_2(k) + z_1(k)z_2(k) + {\color{red}u(k)}\right) + {\color{blue}O(\delta^2)} \\
y(k) &= z_1(k)
\end{aligned}
$$

But the continuous-time linearizing feedback, held constant at the sampling instants, is

$$
u(k) = z_1(k) + z_2(k) - z_1(k)z_2(k) + v(k)
$$

Why did emulation fail: ex cont.

Substituting that feedback in our series expansion we have

$$
\begin{aligned}
z_1(k+1) &= z_1(k) + \delta z_2(k) + \frac{\delta^2}{2!} v(k) + O(\delta^3) \\
z_2(k+1) &= z_2(k) + \delta v(k) + O(\delta^2) \\
y(k) &= z_1(k)
\end{aligned}
$$

Even though still linear, the system matrices changed!

$$
\begin{aligned}
z(k+1) &= A_d z(k) + b_d v(k) + O(\Delta^2) \\
y(k) &= c z(k) \\
A_d &= \begin{pmatrix} 1 & \delta \\ 0 & 1 \end{pmatrix}, \quad b_d = \begin{pmatrix} \frac{\delta^2}{2} \\ \delta \end{pmatrix} \\
c &= \begin{pmatrix} 1 & 0 \end{pmatrix}
\end{aligned}
$$

Why did emulation fail: ex cont.

$$
\begin{aligned}
z(k+1) &= A_d z(k) + b_d v(k) + \color{red}{O(\Delta^2)} \\
y(k) &= c z(k) \\
A_d &= \begin{pmatrix} 1 & \delta \\ 0 & 1 \end{pmatrix}, \quad b_d = \begin{pmatrix} \frac{\delta^2}{2} \\ \delta \end{pmatrix} \\
c &= \begin{pmatrix} 1 & 0 \end{pmatrix}
\end{aligned}
$$

Even though still linear, the system matrices changed!

The same v held constant is no longer guaranteed to get the job done, specially if δ increases/not-constant

Is it enough just to redesign v given the new discrete time representation ?

Why did emulation fail: ex cont.

Matlab break

An approximate sampled-data approach

As we have seen, it is **not** enough to simply redesign the linear control v based on the discrete time representation.

An approximate sampled-data approach

$$
\begin{aligned}
z_1(k+1) &= z_1(k) + \delta z_2(k) + \frac{\delta^{\mathbf{2}}}{2!}\left(-z_1(k) - z_2(k) + z_1(k)z_2(k) + u(k)\right) + O(\delta^3)\\
z_2(k+1) &= z_2(k) + \delta\left(-z_1(k) - z_2(k) + z_1(k)z_2(k) + u(k)\right) + O(\delta^2)\\
y(k) &= z_1(k)
\end{aligned}
$$

The problem is also emphasized by the fact that the output at $t = k+1$ is directly influenced by the input (compared to its second derivative in ct)

$$
y(k+1) = z_1(k+1) = z_1(k) + \delta z_2(k) + \frac{\delta^{\mathbf{2}}}{2!}\left(-z_1(k) - z_2(k) + z_1(k)z_2(k) + u(k)\right) + O(\delta^3)
$$

Can we move the influence of the control to $y(k+2)$ to mimic the continuous time second derivative

An approximate sampled-data approach

Having a problem with complexity, always think about coordinates changes !

Lets look at an output for which our requirement hold;

$$
\begin{aligned}
y^{\delta}(k) &= z_1(k) - \frac{\delta}{2} z_2(k) \\
y^{\delta}(k+1) &= z_1(k+1) - \frac{\delta}{2} z_2(k+1) \\
&= z_1(k+1) - \frac{\delta}{2} z_2(k+1) \\
&= z_1(k) + \delta z_2(k) + \cancel{\frac{\delta^{\mathbf{2}}}{2!}\left(-z_1(k) - z_2(k) + z_1(k) z_2(k) + u(k)\right)} + O(\delta^3) \\
&- \frac{\delta}{2}\left(z_2(k) + \cancel{\delta\left(-z_1(k) - z_2(k) + z_1(k) z_2(k) + u(k)\right)} + O(\delta^2)\right) \\
&= z_1(k) + \frac{\delta}{2} z_2(k) + O(\delta^3)
\end{aligned}
$$

An approximate sampled-data approach

Having a problem with complexity, always think about coordinates changes !
Using this output to define the following linear coordinates change;

$$z_1^\delta = z_1 - \frac{\delta}{2} z_2$$
$$z_2^\delta = z_2$$

In the new coordinates, the dynamics looks like

$$z_1^\delta(k+1) = z_1^\delta(k) + \delta z_2^\delta(k) + O(\delta^3)$$
$$z_2^\delta(k+1) = z_2^\delta(k) + \delta\left(v(k)\right) + O(\delta^2)$$

An approximate sampled-data approach

Leaving the input-output linearizing feedback unchanged and only adding a coordinates change and redesigning v based on y^δ.

Our feedback linearized system now is characterized by the model

$$
\begin{aligned}
z^\delta(k+1) &= A_s z^\delta + b_s v + O(\Delta^2) \\
y^\delta(k) &= c_s z^\delta \\
A_s &= \begin{pmatrix} 1 & \delta \\ 0 & 1 \end{pmatrix}, \quad b_s = \begin{pmatrix} 0 \\ \delta \end{pmatrix} \\
c_s &= \begin{pmatrix} 1 & 0 \end{pmatrix}
\end{aligned}
$$

Also, the orginal output in these new coordinates is characterized by the matrix

$$
c_d = \begin{pmatrix} 1 & \frac{\delta}{2} \end{pmatrix}
$$

An approximate sampled-data approach

Matlab break

Higher orders of approximate feedback linearization

$$
\begin{aligned}
z^\delta(k+1) &= A_s z^\delta + b_s v + \color{red}{O(\Delta^2)} \\
y^\delta(k) &= c_s z^\delta \\
A_s &= \begin{pmatrix} 1 & \delta \\ 0 & 1 \end{pmatrix}, \quad b_s = \begin{pmatrix} 0 \\ \delta \end{pmatrix} \\
c_s &= \begin{pmatrix} 1 & 0 \end{pmatrix}
\end{aligned}
$$

Notice that

$$
O(\Delta^2) = \begin{pmatrix} O(\delta^3) \\ O(\delta^2) \end{pmatrix}
$$

Higher orders of approximate feedback linearization

Can we do even better, i.e. $O(\Delta^3)$?

$$
\begin{aligned}
z_1^\delta(k+1) &= z_1(k+1) - \frac{\delta}{2} z_2(k+1) \\
&= z_1 + \delta z_2 - \frac{\delta^3}{12}\big((z_1-1)(-z_1-z_2+z_1z_2+u) + z_2(z_2-1)\big) + O(\delta^4) \\
z_2^\delta(k+1) &= z_2 + \delta\left(-z_1-z_2+z_1z_2+u\right) \\
&\quad + \frac{\delta^2}{2!}\big((z_1-1)(-z_1-z_2+z_1z_2+u) + z_2(z_2-1)\big) + O(\delta^3)
\end{aligned}
$$

Can we move the influence of the control on the dynamics to higher orders of δ even more?

Since typically $\delta << 1$, the further we remove it, the better ! (**But how?**)

Higher orders of approximate feedback linearization

Going one term more in the power series expansion (dropping the (k));

$$\begin{aligned}
z_1^\delta(k+1) &= z_1 + \delta z_2 - \frac{\delta^3}{12}\big((z_1-1)(-z_1-z_2+z_1z_2+u)+z_2(z_2-1)\big) + O(\delta^4)\\
z_2^\delta(k+1) &= z_2 + \delta\left(-z_1-z_2+z_1z_2+u\right)\\
&\quad + \frac{\delta^2}{2!}\big((z_1-1)(-z_1-z_2+z_1z_2+u)+z_2(z_2-1)\big) + O(\delta^3)
\end{aligned}$$

Higher orders of approximate feedback linearization

$$
\begin{aligned}
z_1^{\delta}(k+1) &= z_1 + \delta z_2 - \frac{\delta^3}{12}\big((z_1-1)(-z_1-z_2+z_1z_2+u)+z_2(z_2-1)\big) + O(\delta^4)\\
z_2^{\delta}(k+1) &= z_2 + \delta\,(-z_1-z_2+z_1z_2+u)\\
&\quad + \frac{\delta^2}{2!}\big((z_1-1)(-z_1-z_2+z_1z_2+u)+z_2(z_2-1)\big) + O(\delta^3)
\end{aligned}
$$

Now write the control as

$$u = u_0 + \delta u_1$$

Higher orders of approximate feedback linearization

We have;

$$
\begin{aligned}
z_1^\delta(k+1) &= z_1 + \delta z_2 - \frac{\delta^3}{12}\big((z_1-1)(-z_1-z_2+z_1z_2+{\color{red}u_0+\delta u_1}) + z_2(z_2-1)\big) + {\color{blue}O(\delta^4)} \\
z_2^\delta(k+1) &= z_2 + \delta\left(-z_1 - z_2 + z_1z_2 + {\color{red}u_0+\delta u_1}\right) \\
&\quad + \frac{\delta^2}{2!}\big((z_1-1)(-z_1-z_2+z_1z_2+{\color{red}u_0+\delta u_1}) + z_2(z_2-1))\big) + {\color{blue}O(\delta^3)}
\end{aligned}
$$

Higher orders of approximate feedback linearization

Focusing only in terms in $O(\Delta^3)$;

$$z_1^\delta(k+1) = z_1 + \delta z_2 - \frac{\delta^3}{12}\Big((z_1-1)(-z_1-z_2+z_1z_2+u_0+\cancel{\delta u_1}) + z_2(z_2-1)\Big) + O(\delta^4)$$

$$\begin{aligned} z_2^\delta(k+1) &= z_2 + \delta\left(-z_1 - z_2 + z_1z_2 + u_0 + \delta u_1\right) \\ &\quad + \frac{\delta^2}{2!}\Big((z_1-1)(-z_1-z_2+z_1z_2+u_0+\cancel{\delta u_1}) + z_2(z_2-1))\Big) + O(\delta^3) \end{aligned}$$

Higher orders of approximate feedback linearization

$$
\begin{aligned}
z_1^\delta(k+1) &= z_1 + \delta z_2 - \frac{\delta^3}{12}\big((z_1-1)(-z_1-z_2+z_1z_2+u_0)+z_2(z_2-1)\big) + O(\delta^4) \\
z_2^\delta(k+1) &= z_2 + \delta\left(-z_1-z_2+z_1z_2+u_0+\delta u_1\right) \\
&\quad + \frac{\delta^2}{2!}\big((z_1-1)(-z_1-z_2+z_1z_2+u_0)+z_2(z_2-1))\big) + O(\delta^3)
\end{aligned}
$$

Now let us set u_0 to be the continuous time linearizing feedback, i.e.

$$
u_0 = z_1 + z_2 - z_1 z_2 + v
$$

Higher orders of approximate feedback linearization

we are left with

$$
\begin{aligned}
z_1^\delta(k+1) &= z_1 + \delta z_2 - \frac{\delta^3}{12}\big((z_1-1)v + z_2(z_2-1)\big) + O(\delta^4) \\
z_2^\delta(k+1) &= z_2 + \delta\left(v + \delta u_1\right) + \frac{\delta^2}{2!}\big((z_1-1)v + z_2(z_2-1))\big) + O(\delta^3)
\end{aligned}
$$

When in doubt, look for a coordinates change!

$$
\eta = \Psi(z^\delta) = \begin{pmatrix} z_1^\delta \\ z_2^\delta - \frac{\delta^2}{12}\big((z_1-1)v + z_2(z_2-1)\big) \end{pmatrix}
$$

Higher orders of approximate feedback linearization

In the new coordinates (we mix coordinates for readability), we are left with

$$\begin{aligned}
\eta_1(k+1) &= \eta_1(k) + \delta\eta_2(k) + O(\delta^4) \\
\eta_2(k+1) &= z_2 + \delta\left(v + \delta u_1\right) + \frac{\delta^2}{2!}\Big((z_1 - 1)v + z_2(z_2 - 1)\Big) \\
&\quad - \frac{\delta^2}{2}((z_1(k+1) - 1)v(k+1) + z_2^\delta(k+1)(z_2^\delta(k+1) - 1)) + O(\delta^3)
\end{aligned}$$

Higher orders of approximate feedback linearization

Then we can use u_1 to cancel the terms in the underbraces!

$$\eta_1(k+1) = \eta_1(k) + \delta\eta_2(k) + O(\delta^4)$$

$$\eta_2(k+1) = z_2 + \delta v + \delta^2 u_1 \underbrace{+\frac{\delta^2}{2!}\big((z_1-1)v + z_2(z_2-1)\big)}$$

$$\underbrace{-\frac{\delta^2}{2}((z_1(k+1)-1)v(k+1) + z_2^\delta(k+1)(z_2^\delta(k+1)-1))} + O(\delta^3)$$

Higher orders of approximate feedback linearization

And with that we have

$$\begin{aligned}\eta_1(k+1) &= \eta_1(k) + \delta\eta_2(k) + O(\delta^4)\\ \eta_2(k+1) &= z_2 + \delta v + O(\delta^3)\end{aligned}$$

Equivalently;

$$\begin{aligned}\eta(k+1) &= A_s\eta(k) + b_s v(k) + O(\Delta^3)\\ y^\delta(k) &= c\eta(k)\end{aligned}$$

What is the catch ?

Higher orders of approximate feedback linearization

Summarizing what we did:

- define a coordinates change $\phi(\cdot) : z \mapsto z^\delta$
- redesign the external linear control v based on the new discrete time system matrices
- to get higher orders approximation, an additional coordinates change was defined $\Psi : z^\delta \mapsto \eta$, and the control was split into linearizing part u_0 and corrective term u_1

Higher orders of approximate feedback linearization

Can this process of defining successive coordinates changes and correcting feedback terms be made more systematic and general?

Let us see some robotics motion control application

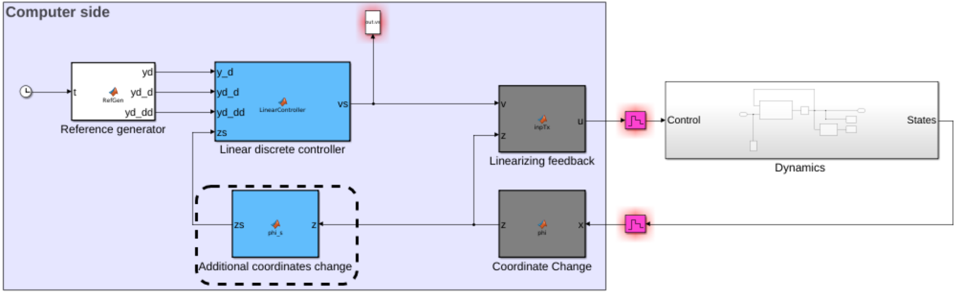
Can we do even better? (PART 2)!

Let's get more systematic

Computing the digital coordinates transformation

we know how to compute the continuous-time linearizing coordinates change and feedback (explicit formula) for generic nonlinear systems with well defined relative degree.

Can we say the same about the digital additional coordinates change ?



Fortunately, YES, and its a linear transformation as a bonus!

Computing the digital coordinates transformation

Without going into derivation (check Ch.7, and IEEE Letters)

$$z^\delta = T_r(\delta) z$$

where r is the relative degree, and;

$$T_r(\delta) = \begin{pmatrix} c_r^\delta \\ \frac{1}{\delta} c_r^\delta (A_r^\delta - I_r) \\ \vdots \\ \frac{1}{\delta^{r-1}} c_r^\delta (A_r^\delta - I_r)^{r-1} \end{pmatrix}$$

$$A_r^\delta = \begin{pmatrix} 1 & \delta & \dots & \frac{\delta^{r-1}}{(r-1)!} \\ 0 & 1 & \dots & \frac{\delta^{r-2}}{(r-2)!} \\ & & \ddots & \\ 0 & 0 & \dots & 1 \end{pmatrix}, \quad b_r^\delta = \begin{pmatrix} \frac{\delta^r}{r!} \\ \frac{\delta^{r-1}}{(r-1)!} \\ \vdots \\ \delta \end{pmatrix}$$

$$c_r^\delta = \delta^r \begin{pmatrix} 0 & 0 & \dots & 1 \end{pmatrix} \begin{pmatrix} b_r^\delta & A_r^\delta b_r^\delta & \dots & (A_r^\delta)^{r-1} b_r^\delta \end{pmatrix}$$

Computing the digital coordinates transformation

Fortunately, the coordinate change only depends on the relative degree of the continuous time system and δ. **Can be precomputed offline!**

Example In our example $r = 2$

$$A_2^\delta = \begin{pmatrix} 1 & \delta \\ 0 & 1 \end{pmatrix}, \qquad b_2^\delta = \begin{pmatrix} \frac{\delta^2}{2} \\ \delta \end{pmatrix}$$

Computing the digital coordinates transformation

Fortunately, the coordinate change only depends on the relative degree of the continuous time system and δ. **Can be precomputed offline!**

Example In our example $r = 2$

$$
A_2^\delta = \begin{pmatrix} 1 & \delta \\ 0 & 1 \end{pmatrix}, \qquad b_2^\delta = \begin{pmatrix} \frac{\delta^2}{2} \\ \delta \end{pmatrix}
$$

$$
\begin{aligned}
c_2^\delta &= \delta^2 \begin{pmatrix} 0 & 1 \end{pmatrix} \begin{pmatrix} b_2^\delta & A_2^\delta b_2^\delta \end{pmatrix} \\
&= \delta^2 \begin{pmatrix} 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{\delta^2}{2!} & \frac{3\delta^2}{2!} \\ \delta & \delta \end{pmatrix}^{-1} \\
&= \begin{pmatrix} 1 & -\frac{\delta}{2} \end{pmatrix}
\end{aligned}
$$

Computing the digital coordinates transformation

Fortunately, the coordinate change only depends on the relative degree of the continuous time system and δ. **Can be precomputed offline!**

Example In our example $r = 2$

$$
\begin{aligned}
A_2^\delta &= \begin{pmatrix} 1 & \delta \\ 0 & 1 \end{pmatrix}, \qquad b_2^\delta = \begin{pmatrix} \frac{\delta^2}{2} \\ \delta \end{pmatrix} \\
c_2^\delta &= \delta^2 \begin{pmatrix} 0 & 1 \end{pmatrix} \begin{pmatrix} b_2^\delta & A_2^\delta b_2^\delta \end{pmatrix} \\
&= \delta^2 \begin{pmatrix} 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{\delta^2}{2!} & \frac{3\delta^2}{2!} \\ \delta & \delta \end{pmatrix}^{-1} \\
&= \begin{pmatrix} 1 & -\frac{\delta}{2} \end{pmatrix} \\
T_2(\delta) &= \begin{pmatrix} c_r^\delta \\ \frac{1}{\delta} c_r^\delta (A_r^\delta - I_r) \end{pmatrix} = \begin{pmatrix} 1 & -\frac{\delta}{2} \\ 0 & 1 \end{pmatrix}
\end{aligned}
$$

Computing the digital coordinates transformation

Fortunately, the coordinate change only depends on the relative degree of the continuous time system and δ. **Can be precomputed offline!**

So, we have for $r = 2, 3, \ldots$

$$T_2(\delta) = \begin{pmatrix} 1 & -\frac{\delta}{2} \\ 0 & 1 \end{pmatrix}$$

$$T_3(\delta) = \begin{pmatrix} 1 & -\delta & \frac{\delta^2}{3} \\ 0 & 1 & -\frac{\delta}{2} \\ 0 & 0 & 1 \end{pmatrix}$$

$$\vdots$$