Implementation of pole at origin

If $p_c = \phi$ (comp pole at origin), then clearly

$$\alpha = \exp[\phi T_s] = 1$$

in the implementation eq'n. However $\beta = \frac{(1-1)}{\phi}$ is indeterminate.

If we look more carefully at $\lim_{p_c \to \phi} \left[\frac{1 - \exp[p_c T_s]}{-p_c} \right]$

This yields the correct value $\beta = T_s$ for this case.

Thus for $\dot{x}(t) = e(t)$

we have $x(t_{k+1}) = x(t_k) + T_s\, e(t_k)$

i.e. $x_{k+1} = x_k + T_s e_k$

A closer look

$$\dot{x}(t) = e(t) \Rightarrow x_{k+1} = x_k + T_s e_k$$

$$\Rightarrow \quad x(t+dt) = x(t) + dt\, e(t)$$

So our ZOH discretization strategy is equivalent to a simple (and not terribly accurate) <u>Euler method</u> for numerically integrating

Better idea:

$$x(t+dt) = x(t) + \frac{dt}{2}\left[e(t) + e(t+dt)\right]$$

i.e. a <u>trapezoidal</u> numerical approximation

Equivalent discrete equations

$$x(t+dt) = x(t) + \frac{dt}{2}\left[e(t) + e(t+dt)\right]$$

$$\Rightarrow \quad x_{K+1} = x_K + \frac{T_s}{2}\left[e_K + e_{K+1}\right]$$

Which seems to require knowledge of future (e_{K+1})

But:

Let $\quad z_K = x_K - \frac{T_s}{2} e_K$

Then $\quad z_{K+1} = x_{K+1} - \frac{T_s}{2} e_{K+1}$

$$= x_K + \frac{T_s}{2} e_K + \cancel{\frac{T_s}{2} e_{K+1}} - \cancel{\frac{T_s}{2} e_{K+1}}$$

$$\Rightarrow z_{K+1} = z_K + T_s e_K$$

Trapezoidal ("Tustin") Discretization

So $\dot{x}(t) = e(t)$ can more accurately be discretized with the pair of equations

$$\boxed{\begin{aligned} z_{k+1} &= z_k + T_s e_k \\ x_k &= z_k + \frac{T_s}{2} e_k \end{aligned}}$$

Extension to general 1st order DEs is known as

"Tustin's method"

Generally more accurate than simple ZOH.

$\Rightarrow$ most commonly used in practice

Straightforward to calculate, but algebraically tedious

Repeated/complex poles

When $H(s)$ has complex, or repeated poles,

algebraic calculations for both ZOH + Tustin get even <u>more</u> involved.

Fortunately, Matlab has some built-in functions which will crunch the numbers for us.

$\Rightarrow$ c2d(H, T_s)

$\Rightarrow$ c2d(H, T_s, 'tustin')

Need to more carefully understand how to use the outputs from these functions.