ECE 486: Control Systems

Lecture 10C: Control Law Implementation

Key Takeaways

It is common to implement controllers on a microprocessor.

This lecture discusses some of the details associated with this implementation:

- Sample a measurement at specific (discrete) time invervals
- Update the control input u at each sample time.
- Hold the control input u constant until the next update.

The update equation is chosen to approximate the properties of the designed (ODE) controller. The update equation can be implemented on a microprocessor with a few lines of code.

Continuous-Time Control Design

- This course focuses on continuous-time control design.
- We use ODEs to model the plant and obtain a controller in the form of an ODE or transfer function:

  PI Control: $K(s) = \frac{K_p s + K_i}{s}$.

- It is common to implement the controller on a microprocessor using discrete-time updates.

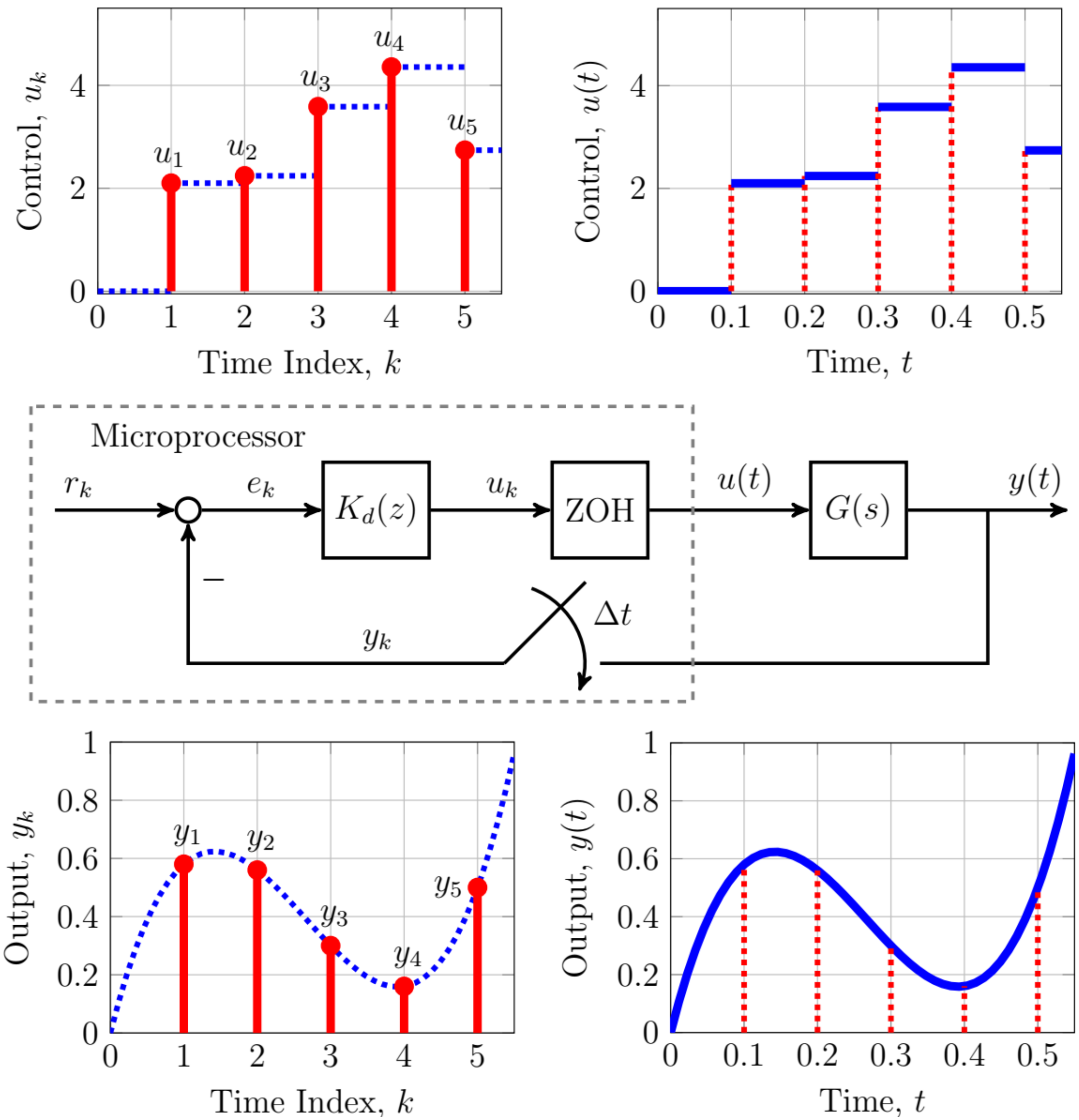
1. Sample the measurement every Δt seconds.
2. Compute the error and use a difference equation to update the control input u.
3. Hold the control input u constant until the next update.

Discrete-Time Implementation

The diagram shows the three main steps:

1. Sampling
2. Control Update
3. Zero-Order Hold

We will describe these in detail on the next few slides.



Sampling

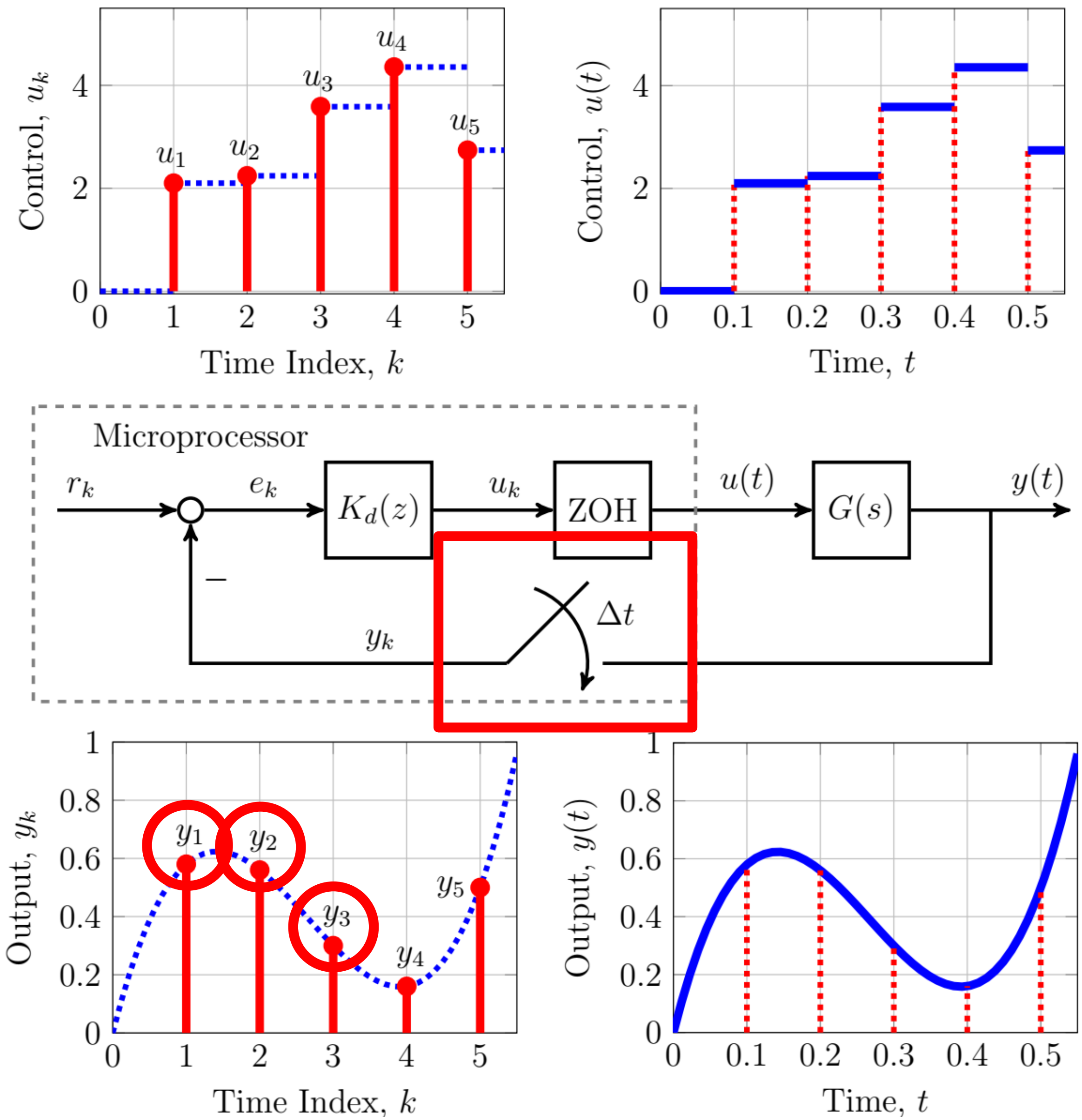
- Plant output *y(t)* is a continuous-time signal.
- Microprocessor samples every Δt seconds:

$$y_1 := y(\Delta t)$$

$$y_2 := y(2\Delta t)$$

$$y_3 := y(3\Delta t)$$

- $y_k := y(k \cdot \Delta t)$ is a discrete-time signal
- Typically assume "fast" sampling, 10x faster than relevant dynamics.



Control Update

The microprocessor:

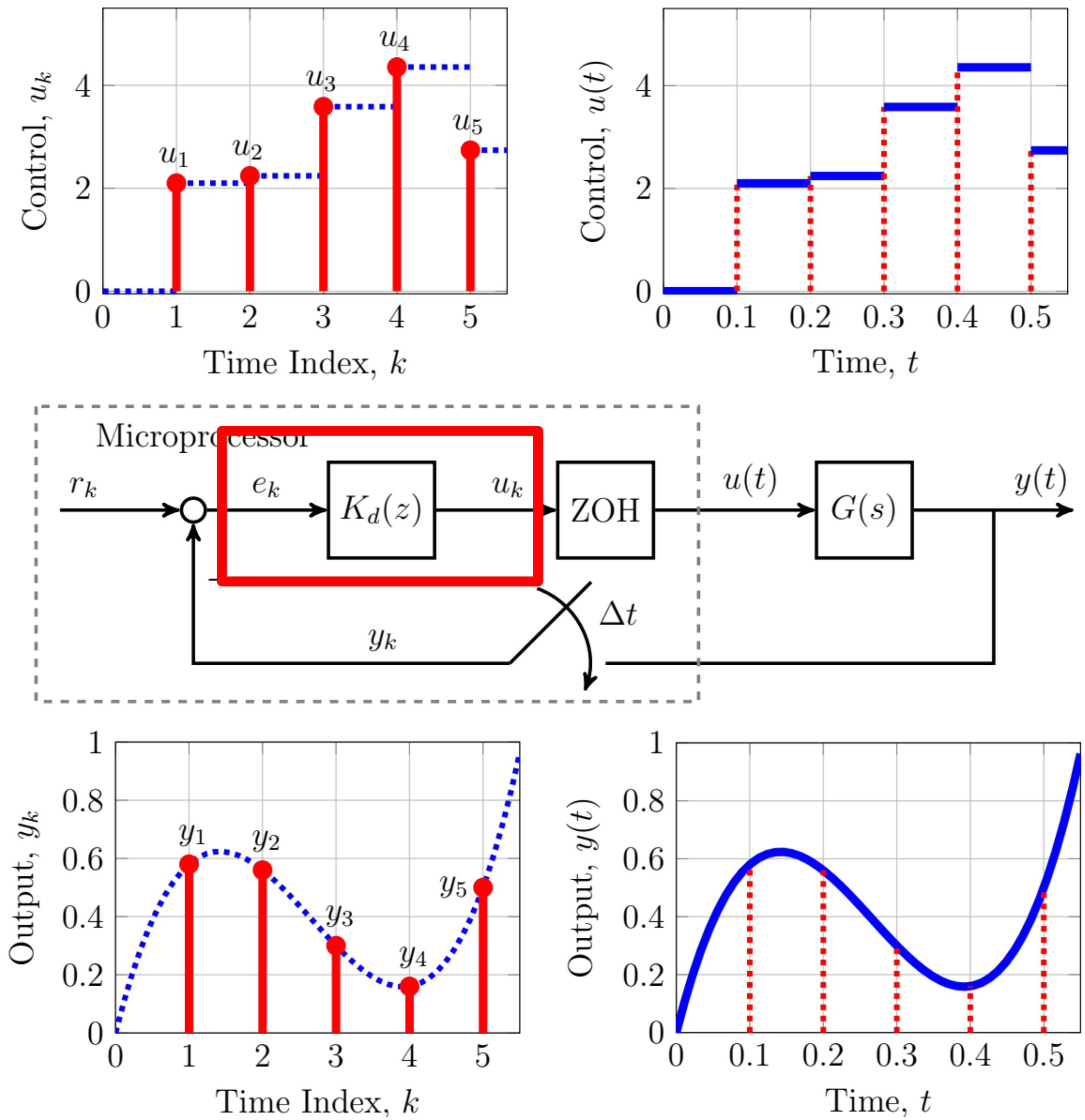
- Computes the error

  $e_k = r_k - y_k$

- Updates the discrete-time control with a difference equation,

  for example:

$$u_k = u_{k-1} + 5e_k - 4.9e_{k-1}$$



The discrete-time update, denoted $K_d(z)$, is chosen to approximate $K(s)$.

(Details later.)

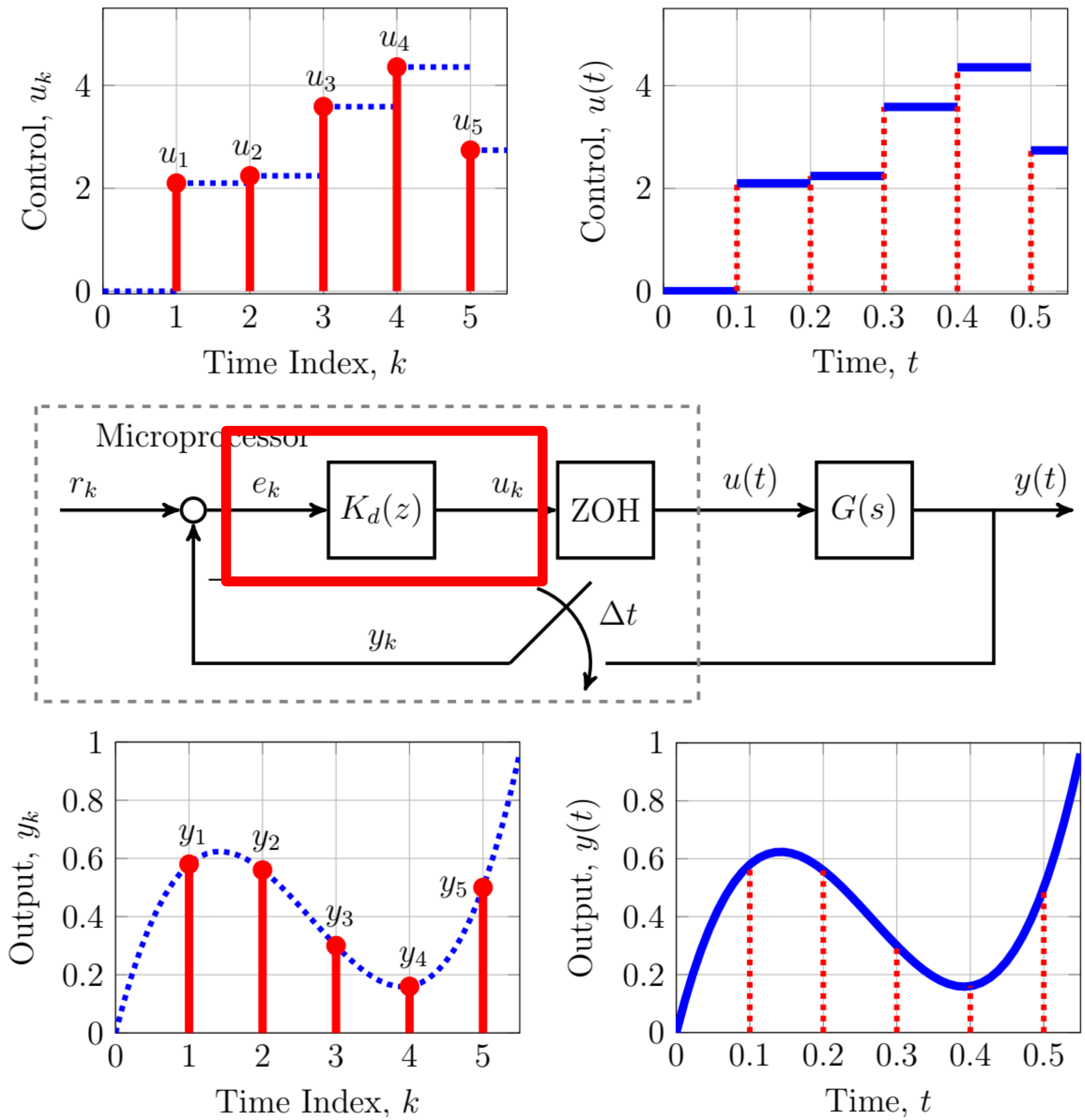
Control Update

Difference Equation:

$$u_k = u_{k-1} + 5e_k - 4.9e_{k-1}$$

Pseudo-code for the update is:

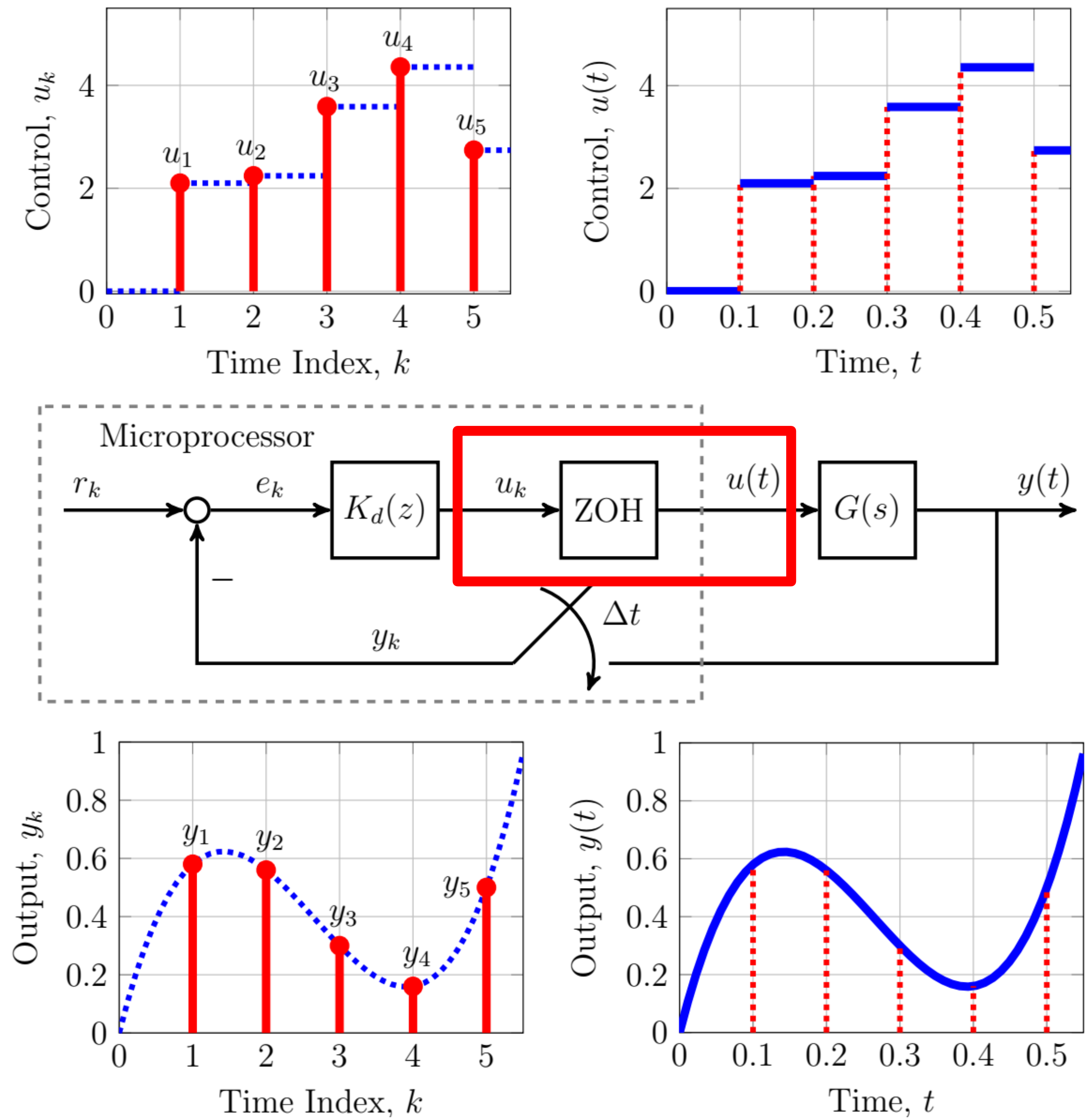
```
Initialize: uprev=0, eprev=0
while(1)
  Obtain new samples: (r,y)
  Compute error: e = r - y
  Update control:
    u = uprev + 5*e - 4.9*eprev
  Update previous values:
    uprev = u, eprev = e
end
```



Zero-Order Hold

- The microprocessor only updates u_k when it receives a new sample.
- The discrete-time signal u_k must be converted to continuous-time $u(t)$.
- Zero-Order Hold (ZOH)

$$u(t) = u_k \quad \text{for } t \in [k\Delta t,\ (k+1)\Delta t)$$



Discretization

- (Continuous-Time) PI Controller

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau$$

- Evaluate at two consecutive time steps:

$$u((k-1)\Delta t) = K_p e((k-1)\Delta t) + K_i \int_0^{(k-1)\Delta t} e(\tau)\, d\tau$$

$$u(k\Delta t) = K_p e(k\Delta t) + K_i \int_0^{k\Delta t} e(\tau)\, d\tau$$

- Subtract these equations and use $u_k \coloneqq u(k \cdot \Delta t)$, etc:

$$u_k - u_{k-1} = K_p e_k - K_p e_{k-1} + K_i \int_{(k-1)\Delta t}^{k\Delta t} e(\tau)\, d\tau$$

Discretization

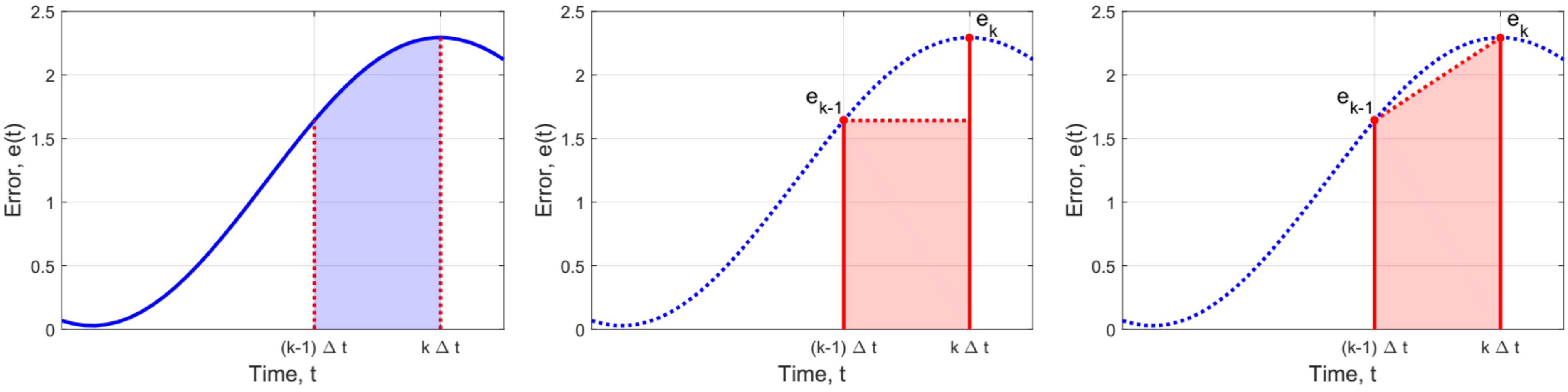
- Need to approximate the integral:

$$u_k - u_{k-1} = K_p e_k - K_p e_{k-1} + K_i \boxed{\int_{(k-1)\Delta t}^{k\Delta t} e(\tau)\, d\tau}$$

$$\approx 0.5 \cdot (e_k + e_{k-1})\, \Delta t$$

- Final difference equation approximation:

$$u_k = u_{k-1} + \left(K_p + K_i \frac{\Delta t}{2}\right) e_k - \left(K_p - K_i \frac{\Delta t}{2}\right)\, e_{k-1}$$



Matlab Example

The discretization method generalizes to n^th-order controllers *K(s)*. The function `c2d` automates this.

```
>> K = tf([1 2 3],[4 5 6]);
>> DeltaT = 0.01;
>> Kd = c2d(K,DeltaT,'foh')
Kd =
  0.2509 z^2 - 0.4968 z + 0.246
  -----------------------------
     z^2 - 1.987 z + 0.9876
Sample time: 0.01 seconds
Discrete-time transfer function.
```

$$K(s) = \frac{s^2 + 2s + 3}{4s^2 + 5s + 6}$$

$$u_k - 1.987u_{k-1} + 0.9876u_{k-2} = 0.2509e_k - 0.4968e_{k-1} + 0.246e_{k-2}$$

$$u_k = 1.987u_{k-1} - 0.9876u_{k-2} + 0.2509e_k - 0.4968e_{k-1} + 0.246e_{k-2}$$