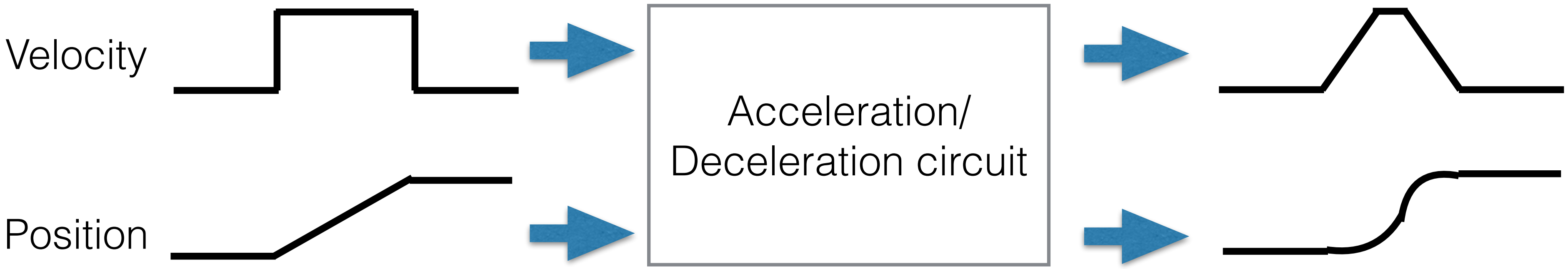
Implementation of digital moving average filter on microcontroller

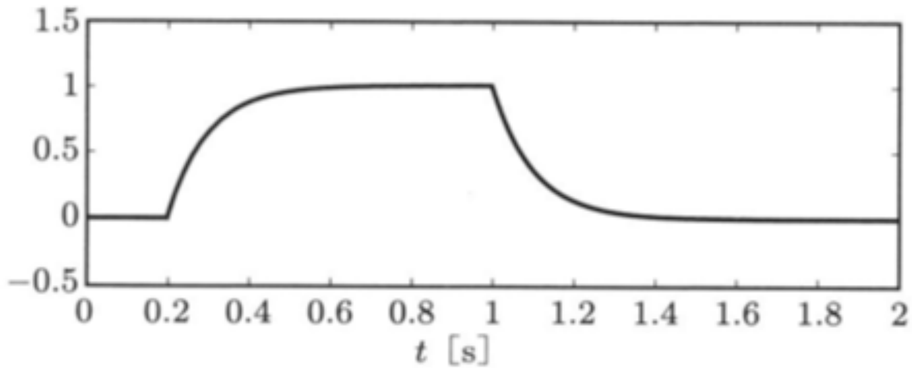
Acceleration/Deceleration circuit



- In acceleration control (e.g. CNC controller), acceleration/deceleration circuit is used to shape the velocity command
- There are three kinds of acceleration circuit

1) Exponential acceleration & deceleration

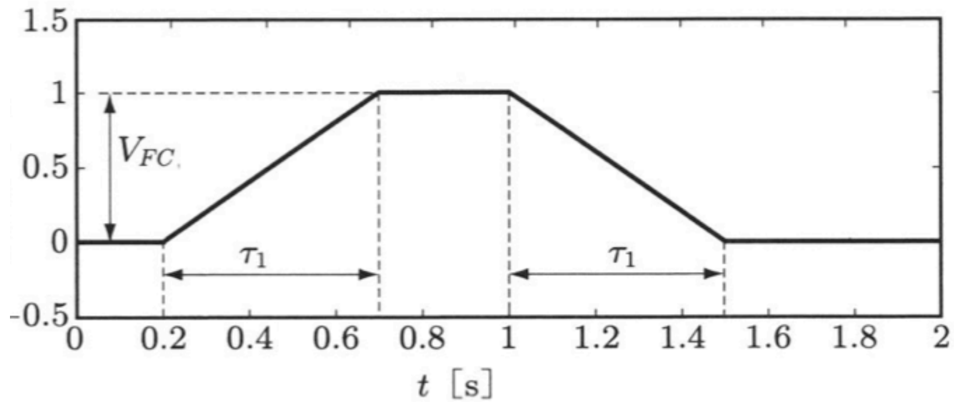
$$G(s) = \frac{1}{\tau_{lc}s + 1}$$



- This is equivalent to RC circuit you developed
  - We are not using this one

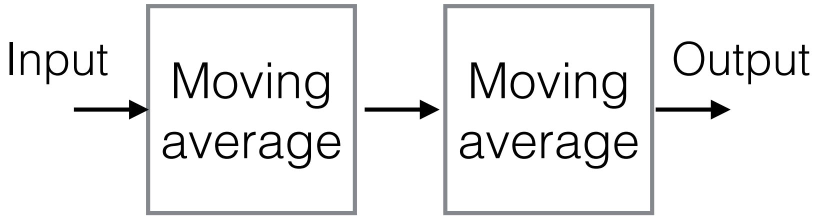
2) Moving average acceleration (i.e. Linear acceleration)

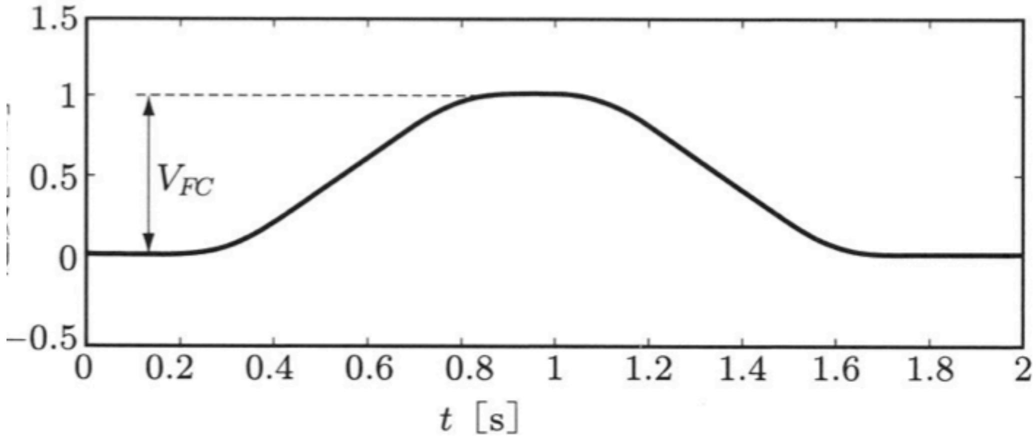
$$f_{\text{out}}(t) = \frac{1}{\tau_1}\int_{t-\tau_1}^{t} f_{\text{in}}(\tau)d\tau$$
$$= \frac{1}{\tau_1}\left[\int_0^t \{f_{\text{in}}(\tau)\,d\tau - f_{\text{in}}(\tau-\tau_1)\}\,d\tau\right]$$



- We use this one
  - V_{FC} is the constant input
  - τ_1 is the time to complete acceleration (9 seconds in our case)

3) Two-stage moving average acceleration (i.e. S-curve acceleration)





- By chaining two moving average filters, the velocity profile becomes S-curve
  - We don't use this one

Let's calculate continuous transfer function G

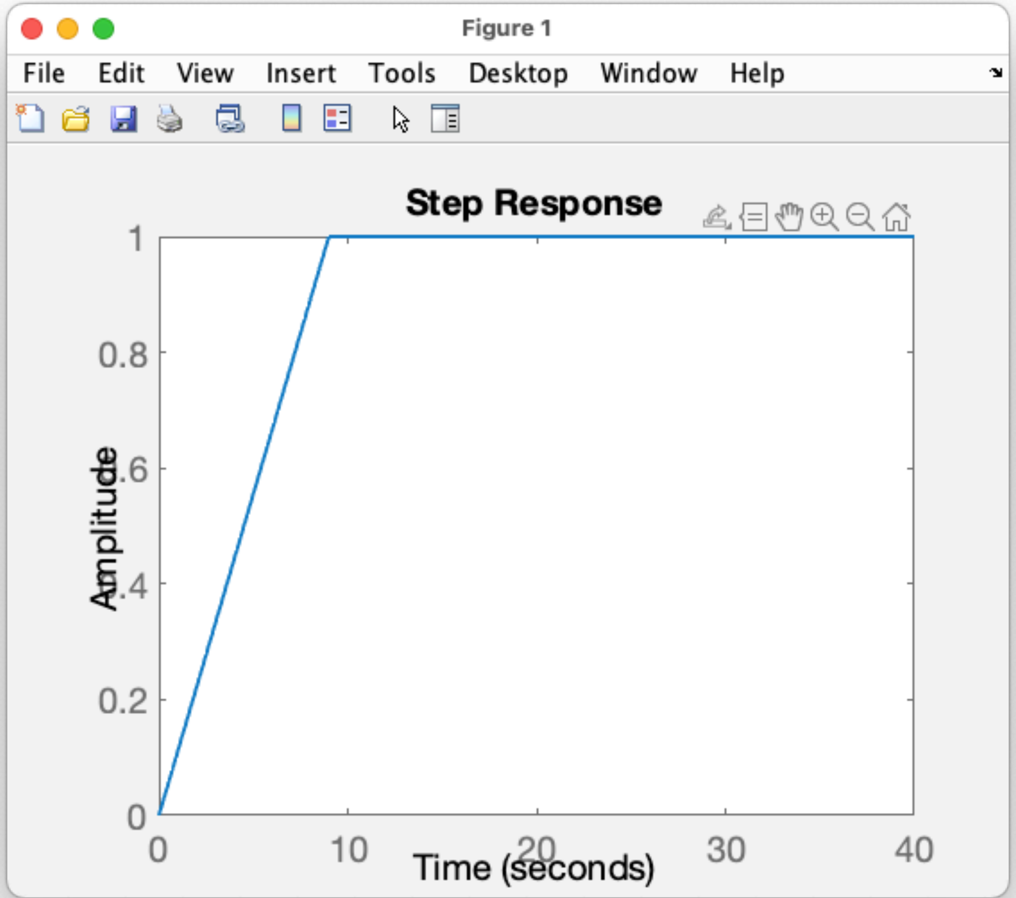
$$f_{\mathrm{out}}(t) = \frac{1}{\tau_1}\left[\int_0^t \{f_{\mathrm{in}}(\tau)\,d\tau - f_{\mathrm{in}}(\tau-\tau_1)\}\,d\tau\right]$$

$$G_{ma}(s) = \frac{1}{\tau_1}\frac{1-e^{-\tau_1 s}}{s} \qquad \leftarrow G_{ma} = F_{out}(s)/F_{in}(s)$$

- Transfer function G(s) is calculated as
  $$G(s) = \frac{1}{\tau_1}\frac{1-e^{-\tau_1 s}}{s}$$

Matlab simulation

```
tau1 = 9;
s = tf('s');
G = 1/tau1*(1-exp(-tau1*s))/s;
step(G)
```



It is successfully accelerated in 9 seconds

Discretize transfer function G to H

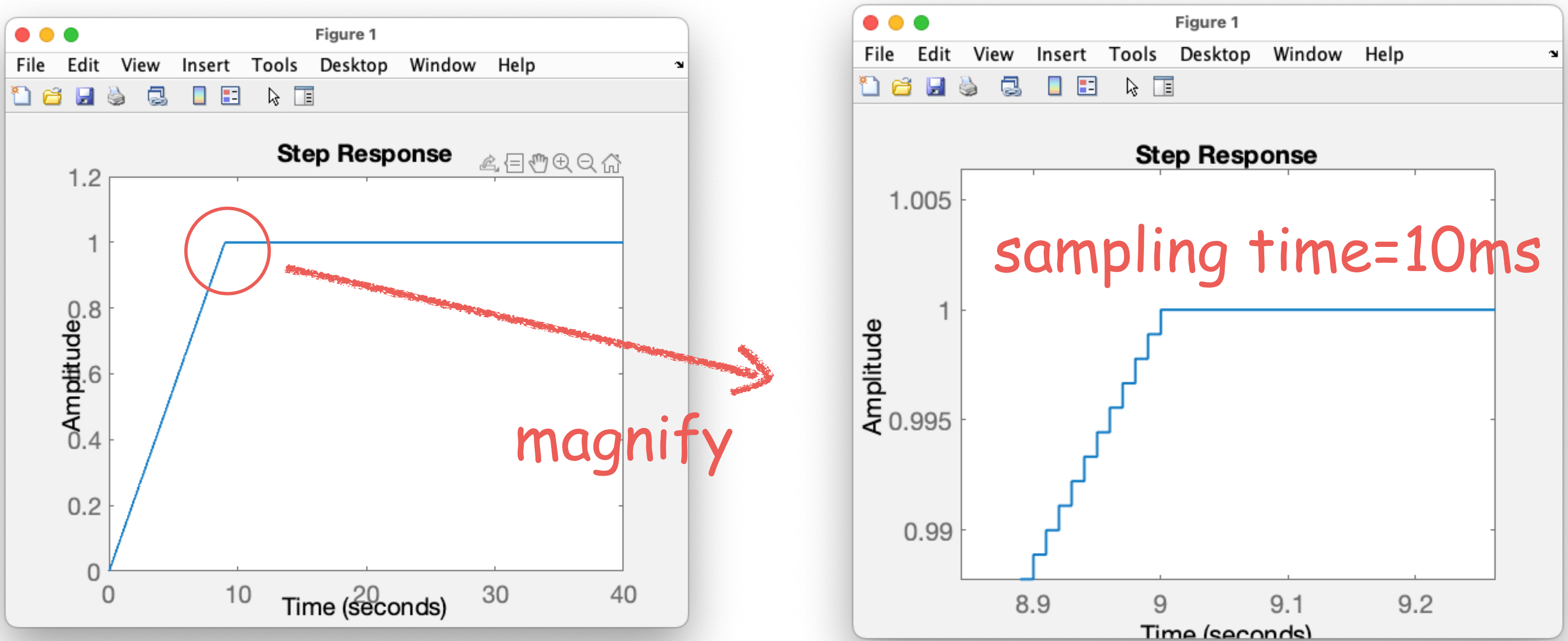
- We need to discretize transfer function to implement the algorithm on the microcontroller
- In Matlab, we can use **`c2d()`** function

```
tau1 = 9;
s = tf('s');
G = 1/tau1*(1-exp(-tau1*s))/s;
sampling_time = 0.01;
H = c2d(G,sampling_time,'zoh');
step(H);
```

←10 [ms]

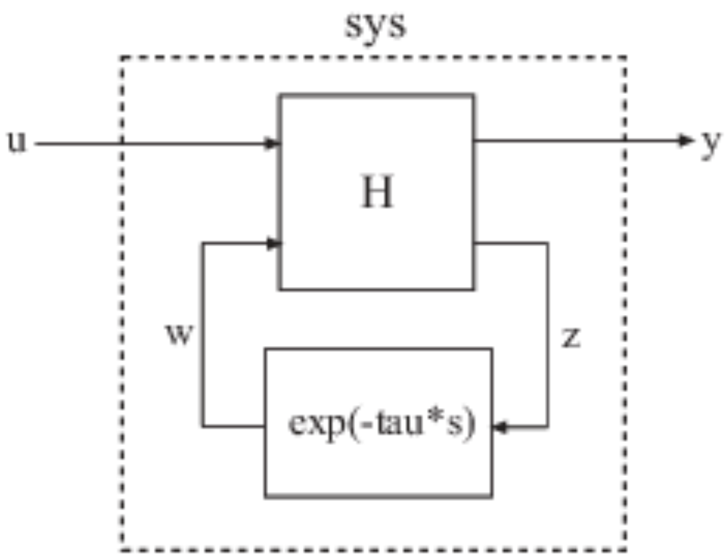
Zero-Order Hold

Check discrete step response



It is successfully discretized

Represent discrete transfer function H as a difference equation (= State space representation)



$$Ex[k+1] = Ax[k] + B_1u[k] + B_2w[k]$$
$$y[k] = C_1x[k] + D_{11}u[k] + D_{12}w[k]$$
$$z[k] = C_2x[k] + D_{21}u[k] + D_{22}w[k]$$
$$w[k] = z[k-\tau]$$

```
[A,B1,B2,C1,C2,D11,D12,D21,D22,E,tau] = getDelayModel(sys)
```

Ref. https://www.mathworks.com/help/control/ref/ss.getdelaymodel.html

- getDelayModel() converts transfer function H to difference equation given above
- u is our voltage input V_{in}, y is our voltage output V_{out}

Matlab simulation (1/2)

```
tau1 = 9;
sampling_time = 0.01;
s = tf('s');
G = 1/tau1*(1-exp(-tau1*s))/s;
H = c2d(G,sampling_time,'zoh');
[A,B1,B2,C1,C2,D11,D12,D21,D22,E,tau] =
getDelayModel(H);
max_simulation_time = 20; %[s]
kmax = round(max_simulation_time/sampling_time);
x = zeros(1,kmax);
y = zeros(1,kmax);
z = zeros(1,kmax);
w = zeros(1,kmax);
Vmax = 5; %[V]
u = Vmax*ones(1,kmax);
```

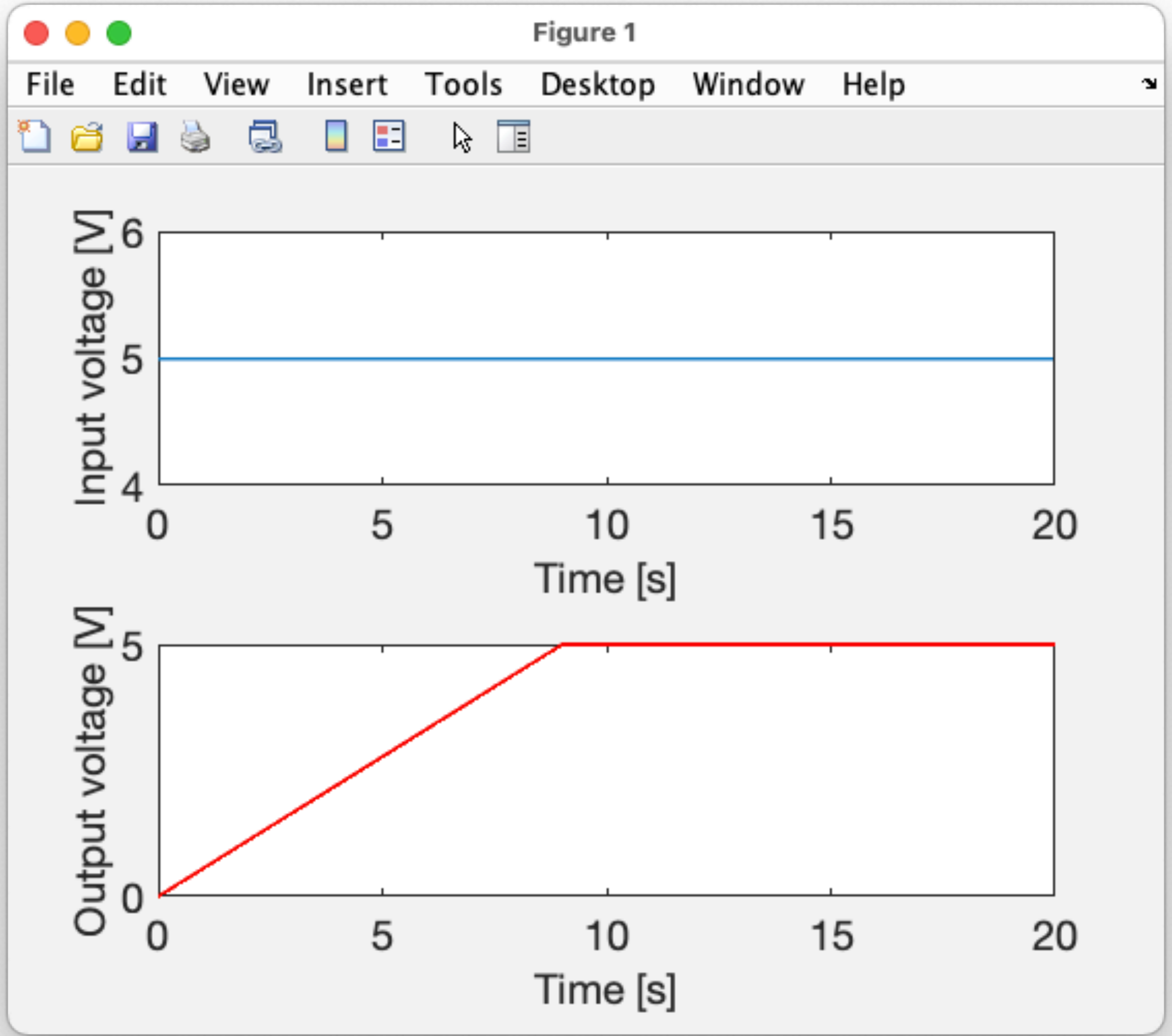
Matlab simulation (2/2)

```
for k=1:kmax
  x(k+1) = A*x(k) + B1*u(k) + B2*w(k);
  y(k) = C1*x(k) + D11*u(k) + D12*w(k);
  z(k) = C2*x(k) + D21*u(k) + D22*w(k);
  w(k+1) = z(max(1,k-tau));
end

t = (1:kmax)*sampling_time;
subplot(211);
plot(t,u);
xlabel('Time [s]');
ylabel('Input voltage [V]');

subplot(212);
plot(t,y,'r');
xlabel('Time [s]');
ylabel('Output voltage [V]');
```

Result



The digital moving average filter is successfully implemented

Next step

- Implement the logic in Arduino (or any other MCU)
- What you need is to use the following code

```
for k=1:kmax
  x(k+1) = A*x(k) + B1*u(k) + B2*w(k);
  y(k) = C1*x(k) + D11*u(k) + D12*w(k);
  z(k) = C2*x(k) + D21*u(k) + D22*w(k);
  w(k+1) = z(max(1,k-tau));
end
```

```
A = 1
B1 = 0.0100
B2 = 0
C1 = 0.1111
C2 = -0.1111
D11 = 0
D12 = 1
D21 = 0
D22 = 0
tau = 900
```

N.B. The loop must be run in 100[Hz] in this case

Good luck!