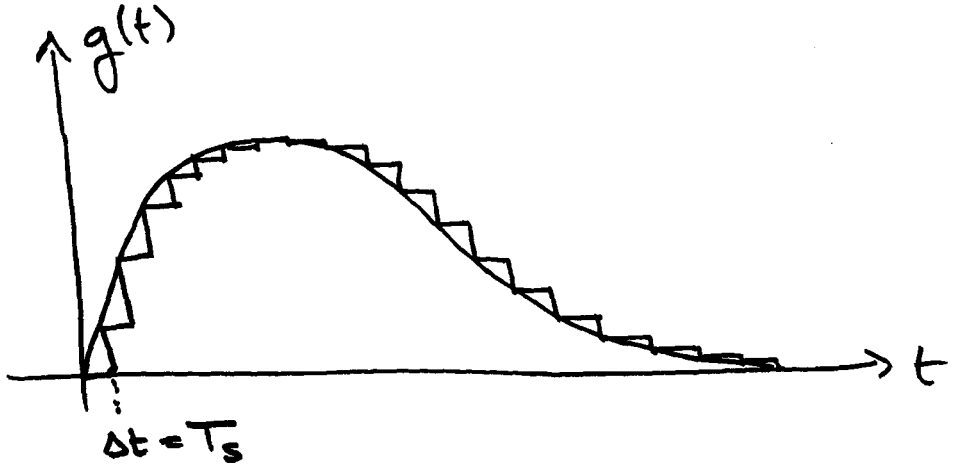
Impulse Response:

Given a linear strictly-proper SISO system. We already know that

$$g(t) = \underline{c}' \cdot e^{At} \cdot \underline{b}$$

is its impulse response. How can we compute this in practice? Let us approximate $g(t)$ by rectangles:



Let: $\Delta t = T_s$ be its sampling rate.

$$\Rightarrow g_0 = g(t=\emptyset) = \underline{c}' \cdot \underline{b}$$
$$g_1 = g(t = 1 \cdot \Delta t) = \underline{c}' \cdot e^{AT_s} \cdot \underline{b}$$

Let: $F = e^{A \cdot T_s}$

$$\Rightarrow \quad g_1 = \underline{c}' \cdot F \cdot \underline{b}$$

$$g_2 = g(t = 2 \cdot \Delta t) = \underline{c}' e^{A \cdot 2T_s} \cdot \underline{b}$$

$$\equiv \underline{c}' \cdot (e^{A \cdot T_s})^2 \cdot \underline{b} = \underline{c}' \cdot F^2 \cdot \underline{b}$$

$$\Rightarrow \boxed{g_i = g(i \cdot \Delta t) = \underline{c}' \cdot F^i \cdot \underline{b}}$$

Matlab:

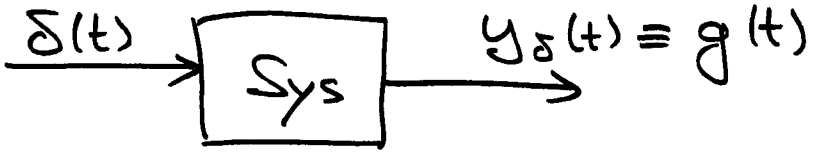
```
g = c*b;
F = expm(A*Ts);
aux = eye(n);                % n = system order
for i=1:k,                   % k = # of steps
    aux = F*aux;
    g = [g; c*aux*b];
end
```

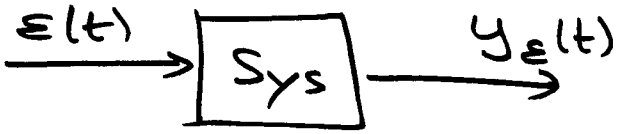
or:

```
d = 0;
sys = ss(A,b,c,d);
t = 0:Ts:k*Ts;
g = impulse(sys,t);
```

Step Response:

Given a linear strictly-proper SISO system.





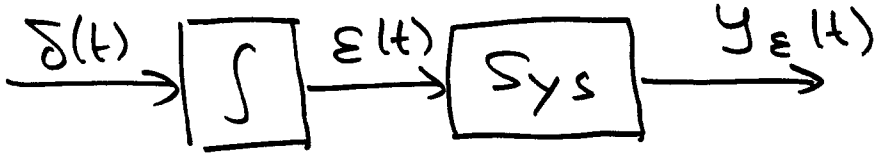
$y_\delta(t) = g(t)$ = impulse response

$y_\varepsilon(t)$ = step response

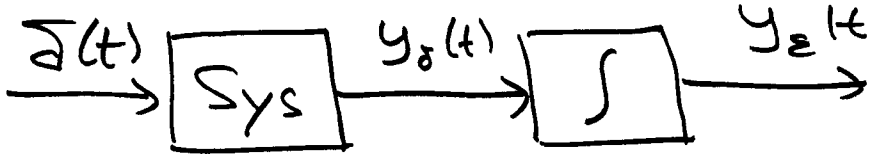
However, we already know that

$$\delta(t) = \frac{d}{dt}\{\varepsilon(t)\}$$

$\Rightarrow$



Because of linearity:



(Linear SISO system in a cascade are commutative.)

$$\Rightarrow y_\varepsilon(t) = \int_{0-}^{t} y_\delta(t)\,dt = \int_{0-}^{t} g(t)\,dt$$

In practice:

$$y_{\varepsilon_0} = y_\varepsilon(t=\phi_+) \equiv \phi$$

(because the system was assumed to be strictly proper.)

$$y_{\varepsilon_1} = y_\varepsilon(t = 1\cdot\Delta t) = g_0 \cdot T_s$$

$$y_{\varepsilon_2} = y_\varepsilon(t = 2\cdot\Delta t) = g_0 T_s + g_1 T_s = (g_0 + g_1)\cdot T_s$$

etc.

$$\Rightarrow \boxed{y_{\varepsilon_i} = T_s \cdot \sum_{j=\phi}^{i-1} g_j}$$

Matlab:

```
ye= 0;
aux = 0;
for i= 1:k,
    aux = aux + g(i);
    ye= [ye; Ts* aux];
end
```

or:

```
d=0;
sys= ss(A, b, c, d);
t = 0 : Ts : k*Ts;
ye= step (sys, t);
```

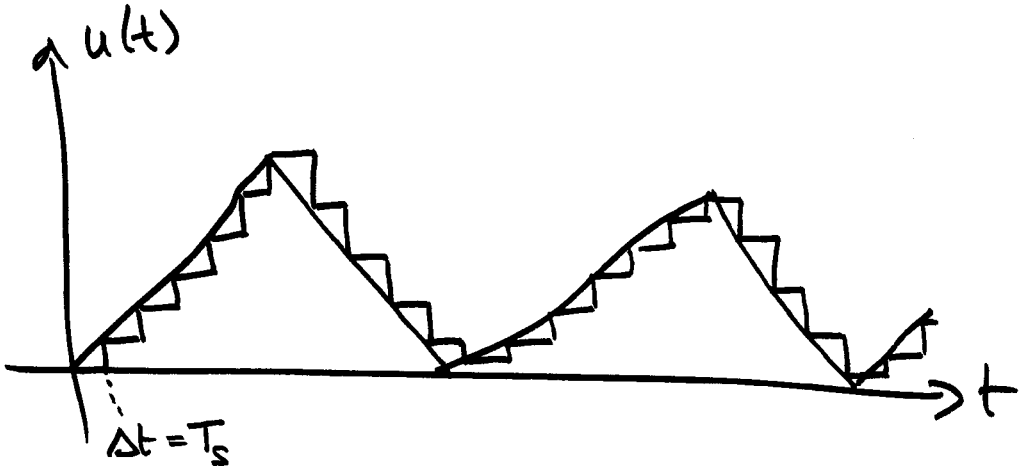
Input Response:

Given a linear strictly-proper SISO system. We already know that:

$$y_u(t) = g(t) * u(t)$$

$$= \int_{0^-}^{t} g(t-\tau) \cdot u(\tau)\, d\tau$$

Let $u(t)$ be any function that does not contain Dirac impulses. We discretize $u(t)$ in the same way as $g(t)$:



$$\Rightarrow y_u(t) = \int_{0^-}^{t} g(t-\tau) \cdot u(\tau)\, d\tau$$

can be approximated by a sum:

$$y_{u_i} = y_u(t = i \cdot \Delta t) = T_s \cdot \sum_{j=0}^{i-1} g_{i-j-1} \cdot u_j$$

$y_{u_0} \equiv 0$

$y_{u_1} = T_s \cdot g_0 \cdot u_0$

$y_{u_2} = T_s \cdot (g_1 \cdot u_0 + g_0 \cdot u_1)$

$y_{u_3} = T_s \cdot (g_2 \cdot u_0 + g_1 \cdot u_1 + g_0 \cdot u_2)$

etc.

This can be written in matrix form as follows:

$$\begin{bmatrix} y_{u_0} \\ y_{u_1} \\ y_{u_2} \\ \vdots \\ y_{u_k} \end{bmatrix} = \underbrace{\begin{bmatrix} \phi & \phi & \phi & \cdots & \phi \\ g_0 & \phi & \phi & \cdots & \phi \\ g_1 & g_0 & \phi & \cdots & \phi \\ \vdots & g_1 & g_0 & \cdots & \phi \\ g_{k-1} & g_{k-2} & g_{k-3} & \cdots & g_0 \end{bmatrix}}_{T_g} \cdot \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{k-1} \end{bmatrix} \cdot T_s$$

T_g = Toeplitz matrix over vector g.

[A Toeplitz matrix is a matrix that has the same value along any diagonal.]

Here: $T_g \in \mathbb{R}^{(k+1) \times k}$

Special case: $u_i \equiv 1$

$$y_{u_0} = \phi$$

$$y_{u_1} = T_s \cdot g_0 \cdot 1 = T_s \cdot g_0$$

$$y_{u_2} = T_s \cdot (g_1 \cdot 1 + g_0 \cdot 1) = T_s (g_0 + g_1)$$

$\Rightarrow$ step response.

Matlab offers a function

$$z = \text{conv}(x, y)$$

with the following semantics:

$$x \in \mathbb{R}^n \; ; \; y \in \mathbb{R}^m$$

$$\Rightarrow z \in \mathbb{R}^{n+m-1}$$

$$z(1) = x(1) \cdot y(1)$$
$$z(2) = x(2) \cdot y(1) + x(1) \cdot y(2)$$
$$z(3) = x(3) \cdot y(1) + x(2) \cdot y(2) + x(1) \cdot y(3)$$

etc.

Example:

$$x = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \; ; \; y = \begin{bmatrix} 4 \\ 5 \\ 6 \\ 7 \end{bmatrix}$$

$$\Rightarrow z = \begin{bmatrix} 4 \\ 13 \\ 28 \\ 34 \\ 32 \\ 21 \end{bmatrix}$$

In matrix notation:

$$\begin{bmatrix} 4 \\ 13 \\ 28 \\ 34 \\ 32 \\ 21 \end{bmatrix} = \begin{bmatrix} 1 & \phi & \phi & \phi \\ 2 & 1 & \phi & \phi \\ 3 & 2 & 1 & \phi \\ \phi & 3 & 2 & 1 \\ \phi & \phi & 3 & 2 \\ \phi & \phi & \phi & 3 \end{bmatrix} \cdot \begin{bmatrix} 4 \\ 5 \\ 6 \\ 7 \end{bmatrix}$$

or:

$$\begin{bmatrix} 4 \\ 13 \\ 28 \\ 34 \\ 32 \\ 21 \end{bmatrix} = \begin{bmatrix} 4 & \phi & \phi \\ 5 & 4 & \phi \\ 6 & 5 & 4 \\ 7 & 6 & 5 \\ \phi & 7 & 6 \\ \phi & \phi & 7 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

because: $x * y \equiv y * x$

Using this function, the input response can be computed conveniently:

$$yu = [\phi ; Ts * conv(g, u)]$$

The approximation assumes that $g(t)$ has essentially decayed to a value close to zero at the end of the vector $\underline{g}$, i.e.

$$g(t) \approx 0 \;, \quad \forall\, t > k * Ts \;.$$

A more precise approximation can be obtained using:

```
sys = ss(A,b,c,d);      % d can exist!
t = 0 : Ts : k*Ts;
u = sin(t);             % assuming u=sin(t)
y = lsim(sys,u,t);
```

If an initial condition is also present:

```
y = lsim(sys,u,t,x0);
```