Ingegneria dell ’Automazione - Sistemi in Tempo Reale

Selected topics on discrete-time and sampled-data systems

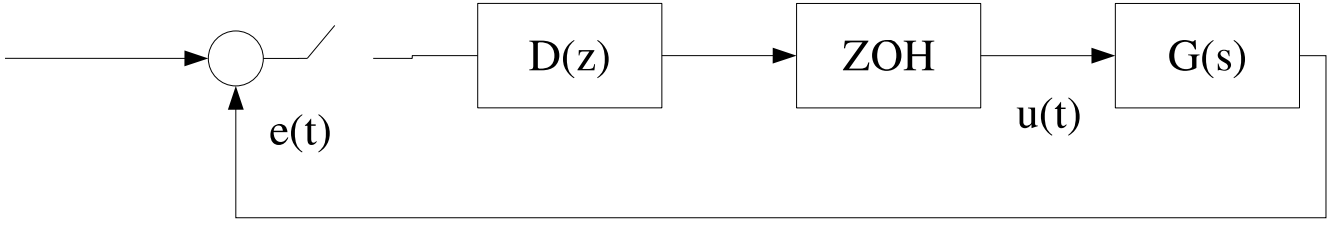
Luigi Palopoli

palopoli@sssup.it - Tel. 050/883444

Outline

- Block diagram analysis of sampled-data systems
- Discrete equivalents and their implementation

Stucture of a digital control system



Heterogeneous diagram: the different blocks have different meaning

Homegeneous blocks

- We aim at a block diagram where blocks are homogeneous
- Consider a sampled signal $e^*(t)$ (a sequence of δ)
- We can model the block $D(z)$ as if it operated on periodic spectrum signal (i.e., transforming sequences of δ into sequences of δ) whereas $D(z)$ operates on sequences of numbers
- The transfer function is given by:

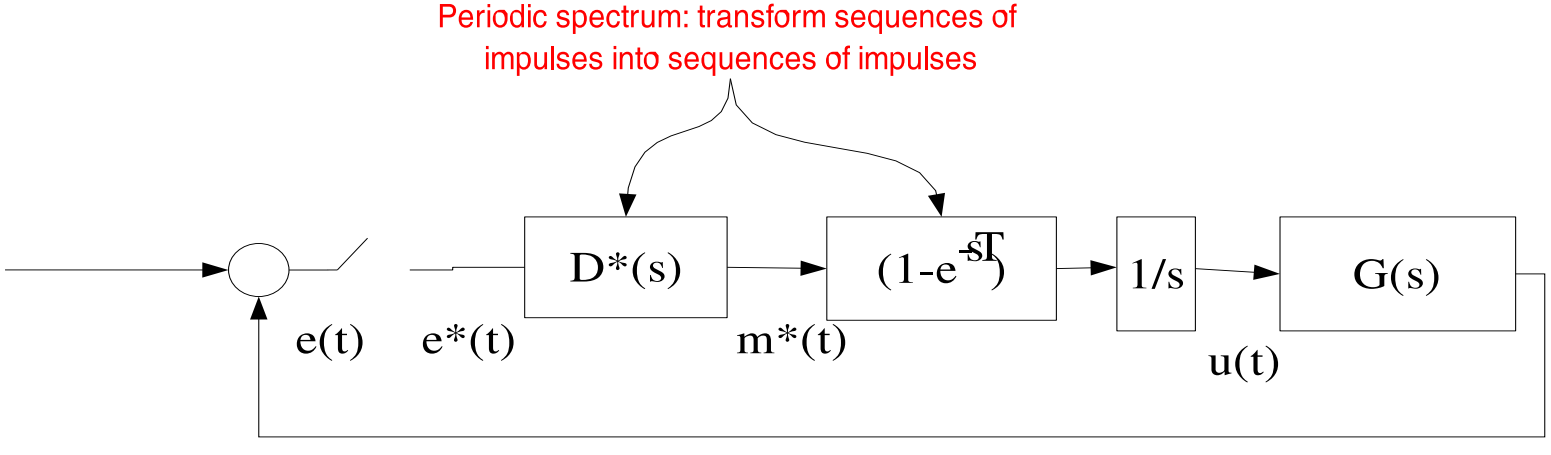
$$D^*(s) = \int_0^{+\infty} \sum_k \delta(t - kT) d(kT) e^{-sT} dt =$$
$$= \sum_k \int_0^{+\infty} \delta(t - kT) d(kT) e^{-sT} dt = \sum_k d(kT) e^{-(sT)k} = D(z)|_{z=e^{sT}}$$

- A useful fact

$$\{E^*(s)G(s)\}^* = E^*(s)G^*(s)$$

(the periodicity of $E^*(s)$ is crucial in the proof).

A different block diagram



In this case , blocks are homegeneuos

A strange transfer function

- From the diagram we read:

$$
\begin{aligned}
E(s) &= R(s) - Y(s) \\
M^*(s) &= D^*(s)E^*(s) \\
U(s) &= M^*(s)\left[\frac{1-e^{-sT}}{s}\right] \\
Y(s) &= G(s)U(s),
\end{aligned}
$$

- ...from which

$$
\begin{aligned}
&E^*(s) = R^*(s) - Y^*(s) \\
&U^*(s) = M^*(s)\text{samples} \rightarrow \text{ZoH} \rightarrow \text{sampling} \rightarrow \text{samples} \\
&Y^*(s) = \{G(s)U(s)\}^* = \left\{G(s)M^*(s)\left[\frac{1-e^{-sT}}{s}\right]\right\}^* = \\
&= \left\{\frac{G(s)}{s}M^*(s)(1-e^{-sT})\right\}^*, \text{Using} \{E^*(s)G(s)\}^* = E^*(s)G^*(s)) \text{ we get} \\
&Y^*(s) = (1-e^{-sT})M^*(s)\left\{\frac{G(s)}{s}\right\}^*
\end{aligned}
$$

A strange transfer function - I

- using $M^*(s) = D^*(s)E^*(s)$, we get

$$Y^*(s) = (1 - e^{-sT})D^*(s)E^*(s)\left\{\frac{G(s)}{s}\right\}^* =$$
$$(1 - e^{-sT}D^*(s)\left\{\frac{G(s)}{s}\right\}^*(R^*(s) - Y^*(s))$$

- setting $H^*(s) = (1 - e^{-sT})D^*(s)\left\{\frac{G(s)}{s}\right\}^*$, we get

$$Y^*(s) = \frac{H^*(s)}{H^*(s) + 1}R^*(s)$$

- This is the usual form for a transfer function!!!
- Note, though, that the sampling operation is not time-invariant: this is a lucky coincidence!

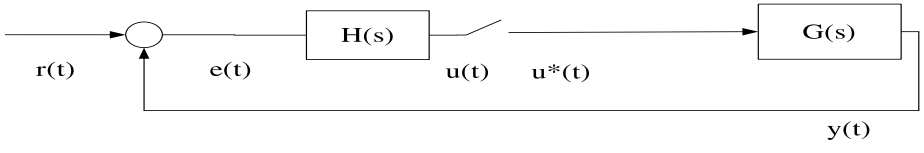
A numeric example

- Plant: $G(s) = \frac{a}{s+a}$
- Controller: $u(kT) = u((k-1)T) + k_0 e(kT) \rightarrow D(z) = \frac{k_0 z}{z-1}$
- $D^*(s) = D(z)|_{z=e^{sT}} = \frac{k_0 e^{sT}}{e^{sT}-1}$
- $H^*(s) = (1-e^{-sT})D^*(s)\left\{\frac{G(s)}{s}\right\}^* = (1-e^{-sT})D^*(s)\left\{\frac{1}{s} - \frac{1}{s+a}\right\}^* =$
  $(1-e^{-sT})D^*(s)\left(\frac{1}{1-e^{-sT}} - \frac{1}{1-e^{-aT}e^{-sT}}\right)$
- Now we are able to compute staic and dyanmic responses on H^*
- It is easy to show that

$$Y(s) = \frac{(1-e^{-sT})R^*(s)D^*(s)}{1+H^*(s)}\frac{G(s)}{s}$$

- From the above it is possible to compute the intersample behaviour. Notice that the first factor produces a train of δ impulses: in the intersampling the systems generates a sum of scaled and traslated step responses.

A different case

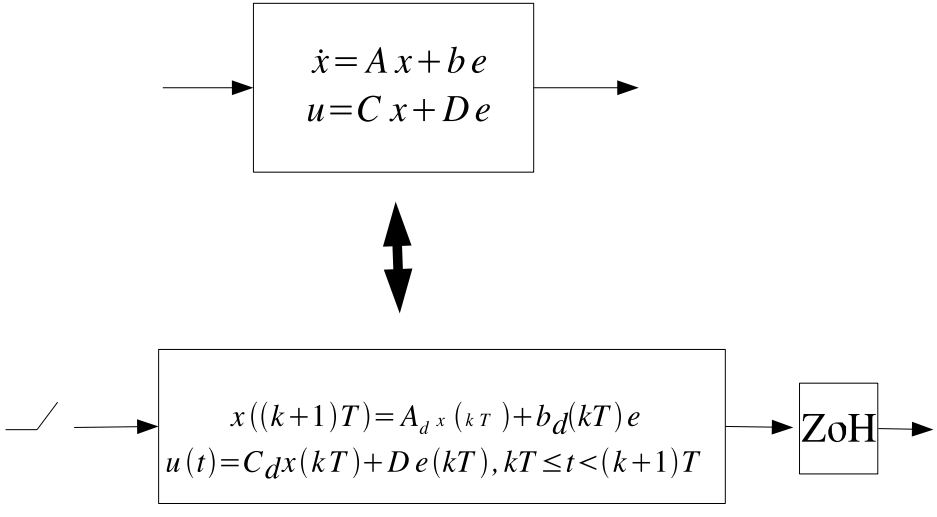


- In a case like this we get: $Y^*(s) = \frac{\{H(s)R(s)\}^*}{1+\{H(s)G(s)\}^*}G^*(s)$
- we haven't got any transfer function structure (R does not enter the system only *via* its samples.

Outline

- Block diagram analysis of sampled-data systems
- Discrete equivalents and their implementation

Discrete Equivalent



- Given a continuous time controler (A, B, C, D) find a discrete time controller (A_d, B_d, C_d, D_d) that is a good approximation for the CT controller.
- Roughly speaking the problem is finding a good numeric integration algorthm....
- There are different ways for doing it:
  - Forward integration rule
  - Backward integration rule
  - Trapezoidal integration rule
  - ZoH equivalent

Forward integration

- Approximate

$$\dot{x} \approx \frac{x((k+1)T) - x(kT)}{T}$$

- ..that leads to:

$$\frac{x((k+1)T)-x(kT)}{T} = Ax(kT) + Be(kT), u(kT) = Cx(kT) + De(kT) \rightarrow$$
$$x((k+1)T) = (I + AT)x(kT) + Be(kT), u(kT) = Cx(kT) + De(kT)$$

- The rule maps the left laplace half-plane into a square of side 1: a stable system might become unstable!

Backward integration

- Approximation: $dotx(t) \approx \frac{x(kT)-x((k-1)T)}{T}$
- Looking at the state evolution: $\frac{x(kT)-x((k-1)T)}{T} = Ax(kT) + Be(kT)$
- ...from which

$$(I - AT)x(kT) = x((k-1)T) + BTe(kT)$$

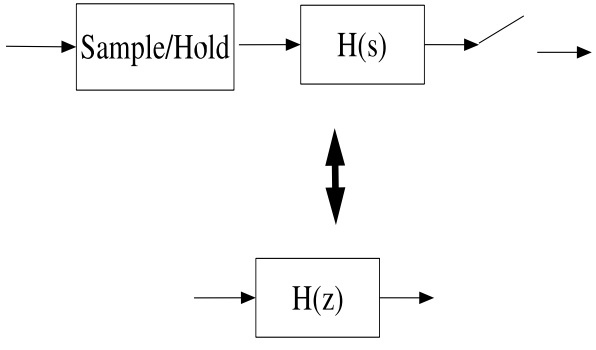
- Introduce $w(kT) = x((k-1)T)$, we get:

$$w((k+1)T) = x(kT) = (I - AT)^{-1}w(kT) + (I - AT)^{-1}BTe(KT)$$

- Output of the controller

$$y(kT) = Cw((k+1)T) + De(kT) = C(I-AT)^{-1}w(kT) + (C(I-AT)^{-1}BT + D)e(kT)$$

ZoH equivalent



- The idea is here to find a discrete equivalent replicating the behaviour of the continuous time controller if it was framed into a ZoH-Sampler scheme
- we know how to do this....

Implementation Scheme

Data structure

```
struct DTSys {
        int nx, ny, nu;
        double * A;
        double * B;
        double * C;
        double * D;
        double * x; /* Current State */
        double * lx; /* Working variable */
}
```

Integration Step

```
void DTStep(struct DTsys * s, double * e, double * u) {
        outputUpdate(s, e, u);
        stateUpdate(s, e);
}
void outputUpdate(struct DTsys * s, double * e, double * u) {
        int i,j; double * x = s->x;
        for (i=0; i < s->ny;i++) { /*output update */
                u[i] = 0;
                for (j=0; j < s->nx;j++)
                        u[i] += s->C[i*s->nx+j]*x[i];
                for (j=0; j < s->nu;j++)
                        u[j] += s->D[i*s->nu+j]*e[j];
        }
};
```

Integration step (continued)

```
void stateUpdate(struct DTsys * s, double * e) {
    int i,j; double * x = s->x; double * lx = s->lx;
    for (i=0; i < s->nx;i++) { /*State update */
        lx[i] = 0;
        for (j=0; j < s->nx;j++)
            lx[i] += s->A[i*s->nx+j]*x[i];
        for (j=0; j < s->nu;j++)
            lx[i] += s->B[i*s->nu+j]*e[j];
        }
    }
    s->x = lx;s->lx = x;
    }
}
```

Initialization

```
double A = {.,.,.,.};/*rows and columns of A*/
double B ={.,.,.,.};/*rows and columns of B*/
double C = {.,.,.,.};
double D = {.,.,.,.};
double x = {.,.};/*Initial state vector */
double lx = {.,.};/*Working variable as big as the state vector */
struct DTsys mySystem = {/*Create and initialize a system structure */
      2,2,2,A,B,C,D,x,lx
}
```

Task structure

```
TASK() {
        double e[2];
        double u[2];
        ...
        for(;;) {
                getE(e);/* Get sensor readings in e*/
                /* e is ouput of the plant
                and input to the controller *
                DTStep(&mySystem,e,u);
                putU(u);/*Write to the actuators */
        }
}
```

Consideration

- Is this the best we can do?
- Not at all!!
- We are unnecssearily delaying the emission of the new output!
- A better solution:

```
TASK() {
        ...
        for(;;) {
                getE(e);/* Get sensor readings in e*/
                outputUpdate(&mySystem,e,u);
                putU(u);/*Write to the actuators */
                stateUpdate(&mySystem,e);
        }
}
```