Sistemi di Discreti

Corrado Santoro

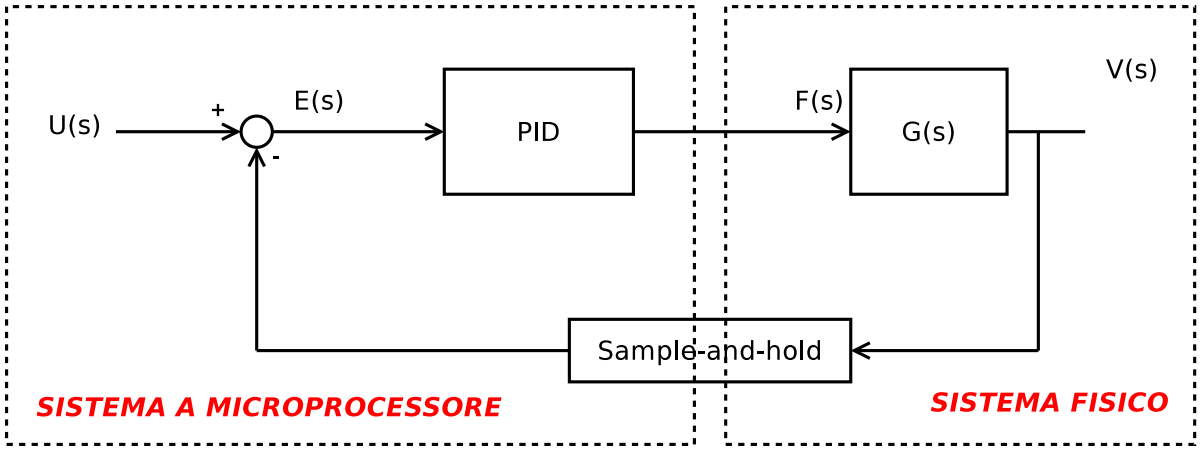
ARSLAB - Autonomous and Robotic Systems Laboratory
Dipartimento di Matematica e Informatica - Università di Catania, Italy
santoro@dmi.unict.it

Programmazione Sistemi Robotici

Dal continuo al discreto

- L'implementazione di un sistema di controllo (ad es. il controllore PID) può essere effettuata con un approccio
  - **tempo-continuo**
  - **tempo-discreto**
- **tempo-continuo:** l'implementazione si effettua sintetizzando direttamente la funzione di trasferimento $C(s)$ tramite componenti elettronici analogici detti **amplificatori operazionali**
- **tempo-discreto:** l'implementazione si effettua usando sistemi a microprocessore che:
  - campionano il segnale ad un intervallo prefissato
  - applicano la $C(s)$ tramite un algoritmo software
  - generano il segnale d'uscita
- Occorre tuttavia trasformare il sistema da tempo-continuo a **tempo-discreto**

Dal continuo al discreto



- Il **sample-and-hold** è un (generico) circuito che "campiona" la grandezza da controllare; esso può essere:
  - un **convertitore analogico-digitale** (ADC) se il segnale proviene da un sensore analogico
  - un'**interfaccia dati** (SPI, I^2C, RS232) se il segnale proviene da un sensore digitale
  - un altro tipo di circuiteria (es. **contatore** o **timer**) dipendente da come il sensore scelto fornisce (elettricamente) la grandezza da controllare

Discretizzazione di un sistema

- L'operazione di discretizzazione implica "campionare" tutti i segnali con un intervallo di tempo prefissato T:

$$
\begin{aligned}
x(t), t \in \mathcal{R} &\Rightarrow x(kT), k \in \mathcal{N} \\
\dot{x}(t), t \in \mathcal{R} &\Rightarrow \frac{x((k+1)T) - x(kT)}{T}, k \in \mathcal{N} \\
x(t) &\Rightarrow x(k) \\
\dot{x}(t) &\Rightarrow \frac{x(k+1) - x(k)}{T}
\end{aligned}
$$

Discretizzazione di un sistema

Il sistema

$$\begin{cases} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{cases}$$

discretizzato con tempo di campionamento T diventa:

$$\begin{cases} \frac{x(k+1)-x(k)}{T} &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) + Du(k) \end{cases}$$

$$\begin{cases} x(k+1) - x(k) &= ATx(k) + BTu(k) \\ y(k) &= Cx(k) + Du(k) \end{cases}$$

$$\begin{cases} x(k+1) &= (AT + I)x(k) + BTu(k) \\ y(k) &= Cx(k) + Du(k) \end{cases}$$

Discretizzazione di un sistema

Il sistema

$$\begin{cases} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{cases}$$

discretizzato con tempo di campionamento T diventa:

$$\begin{cases} x(k+1) &= A'x(k) + B'u(k) \\ y(k) &= Cx(k) + Du(k) \end{cases}$$

dove:

$$\begin{aligned} A' &= AT + I \\ B' &= BT \end{aligned}$$

Sebbene la forma analitica sia diversa, i **due sistemi sono equivalenti**

In un sistema discreto:

$$\begin{cases} x(k+1) &= A'x(k) + B'u(k) \\ y(k) &= Cx(k) + Du(k) \end{cases}$$

scompare l'equazione differenziale che descrive l'evoluzione dello stato nel *tempo*

Essa è sostituita dalla **equazione alle differenze**:

$$x(k+1) = A'x(k) + B'u(k)$$

che indica qual è il valore che assumerà lo stato al *successivo istante di campionamento* (legge di aggiornamento dello stato)

Sistemi discreti: algoritmo (pseudo-codice)

$$\begin{cases} x(k+1) &= A'x(k) + B'u(k) \\ y(k) &= Cx(k) + Du(k) \end{cases}$$

```
MATRIX A_prime;
VECTOR B_prime, C, X;
SCALAR D;

SCALAR discrete_system(SCALAR U)
{
  SCALAR Y = C * X + D * U;          // compute output
  X = A_prime * X + B_prime * U;     // update the state
  return Y;                          // return the output
}

void main()
{
  repeat at period T {
    SCALAR U = sample_input();
    SCALAR Y = discrete_system(U);
    generate_output(Y);
  }
}
```

Autovalori e Stabilità

In un sistema discreto:

$$\begin{cases} x(k+1) &= A'x(k) + B'u(k) \\ y(k) &= Cx(k) + Du(k) \end{cases}$$

la **stabilità** è comunque legata agli autovalori λ della **matrice di stato** $A' = AT + I$:

- Se tutti gli autovalori sono $|\lambda| < 1$, il sistema è **asintoticamente stabile**
- Se gli autovalori sono $|\lambda| < 1$, tranne qualcuno che è $|\lambda| = 1$, ma essi sono **radici semplici**, il sistema è **semplicemente stabile**
- Se esiste almeno un autovalore tale che $|\lambda| > 1$, il sistema è **instabile**

Trasformata Z

Per lo studio analitico dei sistemi discreti si fa uso della **trasformata** z.

Trasformata Z

Sia data una **sequenza** $x(0), x(1), \ldots, x(k), \ldots$, si la trasformata Z è il seguente funzionale

$$X(z) = \mathcal{Z}[x(k)] = \sum_{k=0}^{+\infty} x(k) z^{-k}$$

con z variabile complesssa.

La trasformata Z è la **duale** della trasformata di Laplace nel discreto e gode di proprietà analoghe.

Trasformata Z: Traslazione nel tempo

$$\begin{aligned} \mathcal{Z}[x(k-n)] &= z^{-n}\mathcal{Z}[x(k)] \quad \text{Ritardo} \\ \mathcal{Z}[x(k+n)] &= z^{n}(\mathcal{Z}[x(k)] - \sum_{n=0}^{k-1} x(n)z^{-n}) \quad \text{Anticipo} \end{aligned}$$

Spesso si adotta la seguente convenzione (tuttavia analiticamente non corretta):

$$\begin{aligned} z^{n}x(k) &= x(k+n) \\ z^{-n}x(k) &= x(k-n) \end{aligned}$$

Sistemi discreti e funzione di trasferimento

La trasformata Z si utilizza per rappresentare la **funzione di trasferimento** di un sistema discreto:

$$\begin{cases} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) + Du(k) \end{cases}$$

Applichiamo la $Z-$trasformata:

$$\begin{cases} zX(z) &= AX(z) + BU(z) \\ Y(z) &= CX(z) + DU(z) \end{cases}$$

Risolvendo....

$$\begin{cases} zX(z) - AX(z) &= BU(z) \\ Y(z) &= CX(z) + DU(z) \end{cases}$$

$$Y(z) = (C(zI - A)^{-1}B + D)U(z)$$

La funzione di trasferimento è

$$G(z) = C(zI - A)B + D$$

Poli e Zeri di una Funzione di Trasferimento discreta

$$G(z) = C(zI - A)^{-1}B + D = \frac{N(z)}{D(z)}$$

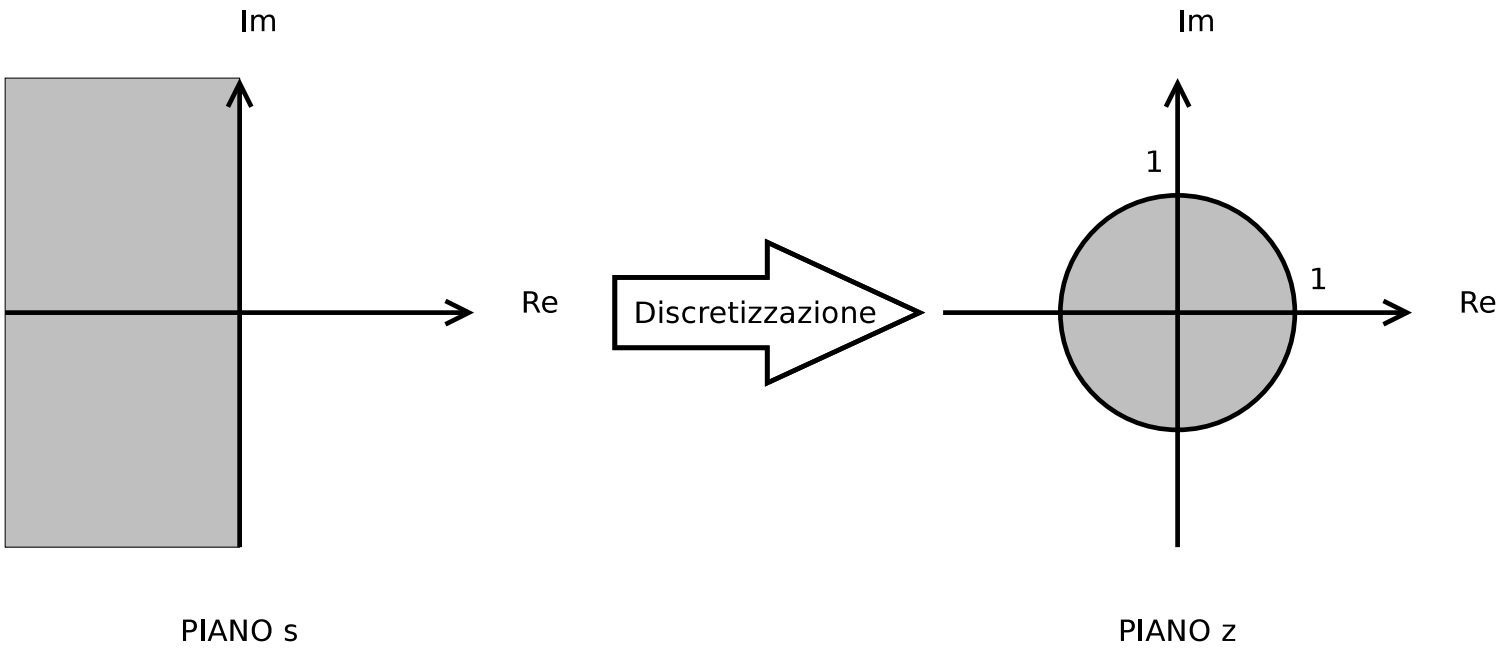
Poli e Zeri

Si definiscono **zeri del sistema** i valori di z che annullano $N(z)$.

Si definiscono **poli del sistema** i valori di z che annullano $D(z)$.

I poli coincidono con gli autovalori di A perchè $D(z)$ è coincide con il polinomio caratteristico di A.

Dualità $\mathcal{L}$ e $\mathcal{Z}$: Poli e Stabilità



- In *s*, la regione di stabilità è il **semipiano sinistro**
- In *z*, la regione di stabilità è il **cerchio unitario**

Trasformazione $\mathcal{L} \rightarrow \mathcal{Z}$

- E' possibile trasformare un **sistema continuo** (in s) in un **sistema discreto** (in z) e viceversa attraverso una trasformazione che mappa il dominio s sul dominio z
- Una delle trasformazioni più usate è la **bilineare**:

$$z = \frac{1 + \frac{T}{2}s}{1 - \frac{T}{2}s} \qquad s = \frac{2}{T}\frac{z-1}{z+1}$$

dove T è il tempo di campionamento

Implementazione funzioni di trasferimento in Z

Sia dato un sistema:

$$Y(z) = G(z)U(z) \tag{1}$$

La sua implementazione *software* si effettua trasformando la relazione (1) in un'**equazione alle differenze** con una dipendenza dal *passato* dell'ingresso e dell'uscita:

$$\begin{aligned} y(k) &= a_1 y(k-1) + a_2 y(k-2) + a_3 y(k-3) + \dots \\ &= \dots + b_0 u(k) + b_1 y(k-1) + a_2 y(k-2) + \dots \end{aligned}$$

Implementazione funzioni di trasferimento in Z: Esempio

Sia dato un sistema:

$$\begin{aligned} G(z) &= \frac{2}{z-0.5} \\ Y(z) &= \frac{2}{z-0.5}U(z) \\ (z-0.5)Y(z) &= 2U(z) \\ zY(z)-0.5Y(z) &= 2U(z) \end{aligned}$$

Ricordando che il prodotto per z^n (z^{-n}) corrisponde ad un **anticipo** (**ritardo**) di n, si ha:

$$\begin{aligned} y(k+1)-0.5y(k) &= 2U(k) \\ y(k+1) &= 2u(k)+0.5y(k) \end{aligned}$$

Implementazione funzioni di trasferimento in Z: Esempio

$$y(k+1) = 2u(k) + 0.5y(k)$$

Trasliamo tutto il sistema di uno step all'indietro:

$$y(k) = 2u(k-1) + 0.5y(k-1)$$

```
float y_1 = 0, u_1 = 0;

float fdt(float u)
{
  float y;
  y = 2 * u_1 + 0.5 * y_1;
  y_1 = y;
  u_1 = u;
  return y;
}
```

Implementazione funzioni di trasferimento in Z: Esempio

Sia dato un sistema:

$$
\begin{aligned}
G(z) &= \frac{z-1}{z^2-0.9z+0.2} \\
Y(z) &= \frac{z-1}{z^2-0.9z+0.2}U(z) \\
(z^2-0.9z+0.2)Y(z) &= (z-1)U(z) \\
z^2Y(z)-0.9zY(z)+0.2Y(z) &= zU(z)-U(z) \\
y(k+2)-0.9y(k+1)+0.2y(k) &= u(k+1)-u(k) \\
y(k+2) = u(k+1)-u(k) &+ 0.9y(k+1)-0.2y(k)
\end{aligned}
$$

Trasliamo tutto il sistema di due step all'indietro:

$$y(k) = u(k-1)-u(k-2)+0.9y(k-1)-0.2y(k-2)$$

Implementazione funzioni di trasferimento in Z: Esempio

$$y(k) = u(k-1) - u(k-2) + 0.9y(k-1) - 0.2y(k-2)$$

```
float y_1 = 0, y_2 = 0, u_1 = 0, u_2 = 0;

float fdt(float u)
{
  float y;
  y = u_1 - u_2 + 0.9 * y_1 - 0.2 * y_2; // calcolo uscita
  // aggiornamento memoria ingresso
  u_2 = u_1;
  u_1 = u;
  // aggiornamento memoria uscita
  y_2 = y_1;
  y_1 = y;
  return y;
}
```

Sistemi di Discreti

Corrado Santoro

ARSLAB - Autonomous and Robotic Systems Laboratory
Dipartimento di Matematica e Informatica - Università di Catania, Italy
santoro@dmi.unict.it

Programmazione Sistemi Robotici