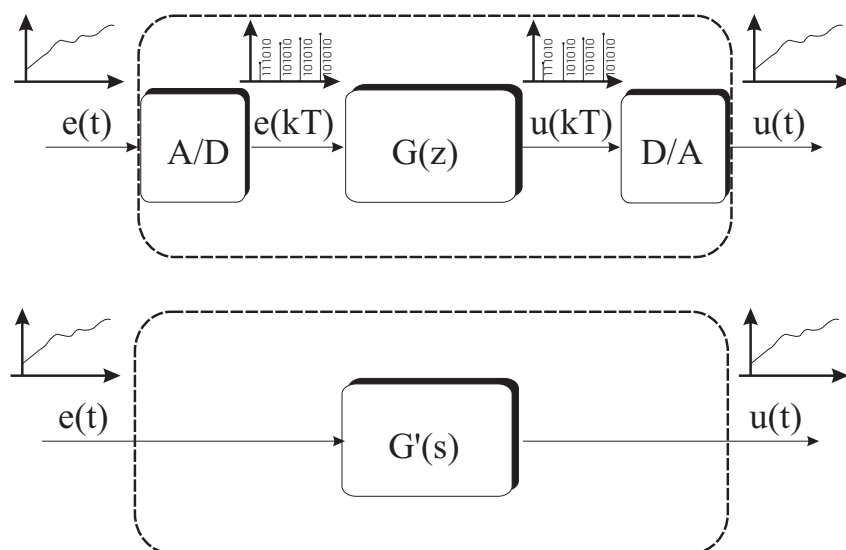


Discretizzazione di sistemi continui

Il metodo di progetto per “discretizzazione”

- Equivalenza tra segnali rappresentati mediante \mathcal{L} -trasformate e \mathcal{Z} -trasformate.



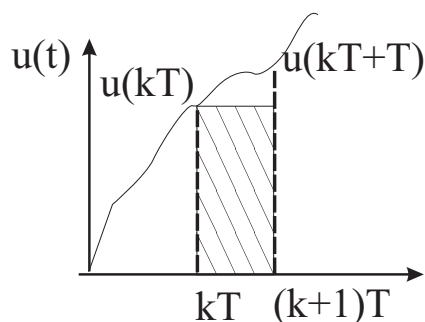
Discretizzazione

- La discretizzazione è il processo attraverso cui è possibile trasformare la rappresentazione $G(s)$ di un sistema a tempo continuo nella rappresentazione $G(z)$ a tempo discreto.

$$G(s) \xrightarrow{z=f(s)} G'(z)$$

Occorre determinare la funzione
 $z = f(s)$

Metodi di discretizzazione - 1) Forward difference



$$I(t) = \int_0^t u(\tau) d\tau \Rightarrow I(s) = \frac{1}{s} u(s)$$

$$I(kT + T) = I(kT) + \int_{t=kT}^{t=kT+T} u(\tau) d\tau$$

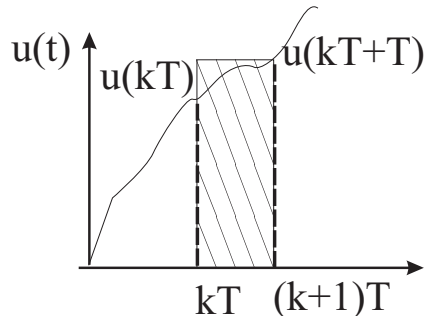
$$I(kT + T) \stackrel{\text{Approx.}}{\approx} I(kT) + T u(kT)$$

$$zI(z) - I(z) = T u(z)$$

$$I(z) = \frac{T}{z-1} u(z)$$

$$s = \frac{z-1}{T}$$

Metodi di discretizzazione - 2) Backward difference



$$I(t) = \int_0^t u(\tau) d\tau \Rightarrow I(s) = \frac{1}{s} u(s)$$

$$I(kT + T) = I(kT) + \int_{t=kT}^{t=kT+T} u(\tau) d\tau$$

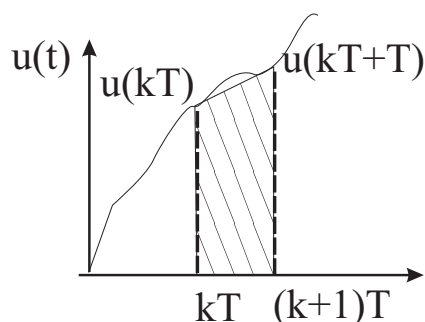
$$I(kT + T) \stackrel{Approx.}{\approx} I(kT) + T u(kT + T)$$

$$zI(z) - I(z) = T z u(z)$$

$$I(z) = \frac{zT}{z-1} u(z)$$

$$s = \frac{z-1}{zT}$$

Metodi di discretizzazione - 3) Bilinear



$$I(t) = \int_0^t u(\tau) d\tau \Rightarrow I(s) = \frac{1}{s} u(s)$$

$$I(kT + T) = I(kT) + \int_{t=kT}^{t=kT+T} u(\tau) d\tau$$

$$I(kT + T) \stackrel{Approx.}{\approx} I(kT) + \frac{T}{2} (u(kT) + u(kT + T))$$

$$zI(z) - I(z) = \frac{T}{2} (u(z) + zu(z))$$

$$I(z) = \frac{T}{2} \frac{z+1}{z-1} u(z)$$

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

Mappatura piano s -piano z .

- Il semipiano sinistro del piano di Gauss s corrisponde a:

$$\Re \left(\frac{2z-1}{Tz+1} \right) < 0 \Rightarrow \Re \left(\frac{z-1}{z+1} \right) < 0$$

- Ponendo $z = \sigma + j\omega$

$$\Re \left(\frac{z-1}{z+1} \right) = \Re \left(\frac{\sigma + j\omega - 1}{\sigma + j\omega + 1} \right) = \Re \left(\frac{\sigma^2 - 1 + \omega^2 + j2\omega}{(\sigma + 1)^2 + \omega^2} \right) < 0$$

$$\sigma^2 + \omega^2 < 1$$

- Quindi il semipiano sinistro nel piano s viene mappato nel cerchio di raggio unitario nel piano z , e quindi le proprietà di stabilità vengono mantenute

Equazioni alle differenze

- L'implementazione finale richiede di ricavare l'equazione alle differenze che implementa il regolatore dalla funzione di trasferimento discreta.

$$u(z) = \frac{a_n z^{n-m} + \dots + a_1 z^{1-m} + a_0 z^{-m}}{b_m + \dots + b_1 z^{1-m} + b_0 z^{-m}} e(z), \quad n < m$$

$$\Downarrow$$

$$b_m u(kT) + \dots + b_1 u(kT - mT + T) + b_0 u(kT - mT) =$$

$$= a_n e(kT - mT + nT) + \dots + a_1 e(kT - mT + T) + a_0 e(kT - mT)$$

Una implementazione di primo tentativo

```
#include <stdio.h>
#include <conio.h>
#define NMAX 1000 /* max num elem denominatore f.d.t. */
#define MMAX 1000 /* max num elem numeratore f.d.t. */
#define EMAX 1000 /* max num elem segnale d'ingresso */
#define UMAX 1000 /* max num elem segnale d'uscita */

int n; /* numero di elementi del vettore a (den f.d.t.) */
int m; /* numero di elementi del vettore b (num f.d.t.) */
/* nb: questi due parametri sono +1 rispetto agli omonimi teorici */
int ne; /* numero di elementi del vettore e (ingresso) */
double u[UMAX]; /* vettore di uscita */
double a[NMAX]; /*denominatore f.d.t. */
double b[MMAX]; /* numeratore f.d.t. */
double e[EMAX]; /* vettore d'ingresso */
```

```
void acquisisci_datastiera(void)
{
    int i;
    printf("inserisci il numero di elementi del numeratore: ");
    scanf("%d",&m);
    for (i=0; i<m; i++)
    {
        printf("inserisci b[%d]: ",i);
        scanf("%lf",&b[i]);
    }
    printf("inserisci il numero di elementi del denominatore: ");
    scanf("%d",&n);
    for (i=0; i<n; i++)
    {
        printf("inserisci a[%d]: ",i);
        scanf("%lf",&a[i]);
    }
    printf("inserisci i primi %d valori dell'uscita\n",n-m);
    for (i=0; i<n-m; i++)
    {
        printf("inserisci u[%d]: ",i);
        scanf("%lf",&u[i]);
    }

    printf("inserisci il numero di elementi dell'ingresso: ");
    scanf("%d",&ne);
    for (i=0; i<ne; i++)
```

```
{    printf("inserisci e[%d]: ",i);  
    scanf("%lf",&e[i]); }  
}
```

```
void main(void)  
{ int k;          /* istante corrente */  
  int i,j;        /* indici cicli */  
  double sommab,sommaa;  
  acquisisci_datastiera();  
  for (k=0; k<n-m; k++) printf("u[%d]=%lf\n",k,u[k]);  
  
  for (k=n-m; k<ne-m+n; k++)  
  {    sommab=0; j=m-1;  
    for (i=k+m-n; i>=k-n && i>=0; i--)  
      { sommab=sommab+b[j]*e[i]; j--;}  
    sommaa=0; j=n-2;  
    for (i=k-1; i>=k-n && i>=0; i--)  
      { sommaa=sommaa+a[j]*u[i]; j--; }  
    sommaa=-(sommaa/a[n-1]);  
    u[k]=sommaa+sommab;  
    printf("u[%d]=%lf\n",k,u[k]);  }  }
```

Implementazione finale

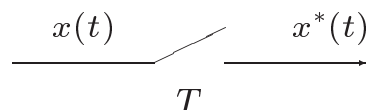
```
static float u[n],e[m];
static float a[n], b[m];

controllore(e[0])
{ int i;float num,den;
/* Algoritmo di controllo */
  u[0] = 1/b[m]*(-(b[m-1]*u[1] + ... + b[0]*u[m])) + ...
  ... + 1/b[m] * (a[n]*e[m-n] + a[n-1]*e[m-n+1]+... + a[0]*e[m]);

/* Aggiornamento campioni precedenti */
u[1]=u[0]; u[2]=u[1]; ... u[n]=u[n-1];
e[1]=e[0]; e[2]=e[1]; ... e[m]=e[m-1];
return(u[0])
}
```

Analisi del processo di campionamento

- Il processo di campionamento trasforma il segnale continuo $x(t)$ nel segnale campionato (definito ancora a tempo continuo) $x^*(t)$.



- Il processo di campionamento può essere rappresentato come:

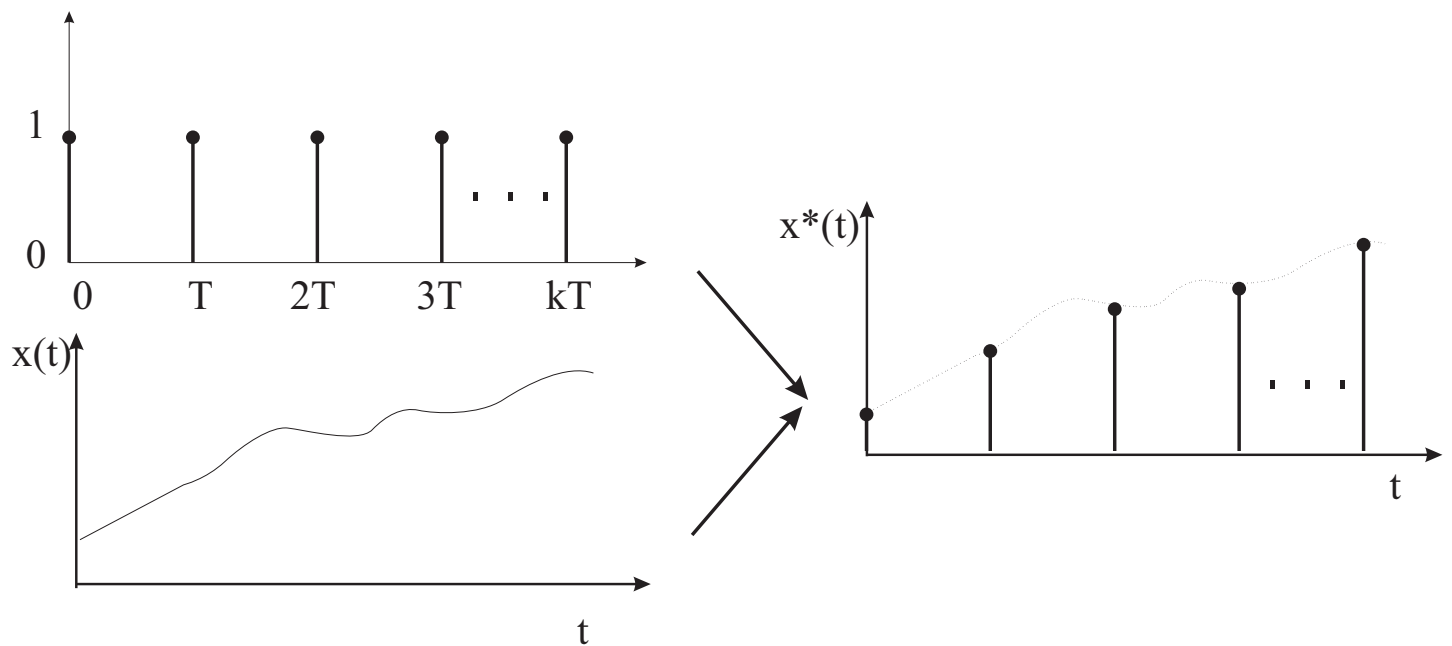
$$x^*(t) = \sum_{k=-\infty}^{\infty} x(t) \delta(t - kT) = x(t) \sum_{k=-\infty}^{\infty} \delta(t - kT) =$$

Note sul lucido 217

Nel seguito consideriamo la seguente terminologia:

- T Periodo di campionamento.
- $f_s = 1/T$ frequenza di campionamento.
- $\omega_s = 2\pi f_s = 2\pi/T$ pulsazione di campionamento.

Campionatore



Spettro del segnale campionato.

1. Il segnale campionato $x^*(t)$ è un segnale periodico e quindi rappresentabile mediante **serie di Fourier**:

$$\sum_{k=-\infty}^{\infty} \delta(t - kT) = \sum_{n=-\infty}^{\infty} C_n e^{j(n\omega_s)t} = \sum_{n=-\infty}^{\infty} C_n (\cos(n\omega_s t) + j \sin(n\omega_s t))$$

2. dove il coefficiente C_n si calcola come:

$$C_n = \frac{1}{T} \int_{-T/2}^{T/2} \sum_{k=-\infty}^{\infty} \delta(t - kT) e^{-j(n\omega_s)t} dt$$

3. Il solo impulso compreso nell'intervallo di integrazione è quello nell'origine (per cui è $k = 0$):

$$C_n = \frac{1}{T} \int_{-T/2}^{T/2} \delta(t) e^{-j(n\omega_s)t} dt$$

4. In base alla proprietà della funzione $\delta(t)$:

$$\int_{-T/2}^{T/2} \delta(t) e^{-jn(2\pi t/T)} dt = \left[e^{-j(n\omega_s)t} \right]_{t=0} = 1$$

5. si ha:

$$C_n = \frac{1}{T}$$

6. Per cui otteniamo la rappresentazione secondo trasformata di Fourier del campionatore ideale:

$$\sum_{k=-\infty}^{\infty} \delta(t - kT) = \frac{1}{T} \sum_{n=-\infty}^{\infty} e^{j(n\omega_s)t}$$

7. Quindi:

$$x^*(t) = x(t) \sum_{k=-\infty}^{\infty} \delta(t - kT) = x(t) \frac{1}{T} \sum_{n=-\infty}^{\infty} e^{j(n\omega_s)t}$$

8. Calcoliamo la trasformata di Laplace $\mathcal{L}[x^*(t)] = X^*(s)$ di un campionatore ideale:

$$X^*(s) = \int_0^{\infty} x^*(t) e^{-st} dt = \int_0^{\infty} \left[x(t) \frac{1}{T} \sum_{n=-\infty}^{\infty} e^{j(n\omega_s)t} \right] e^{-st} dt$$

9. Integrando la serie termine a termine (i.e. scambiando fra loro l'integrale e la somma):

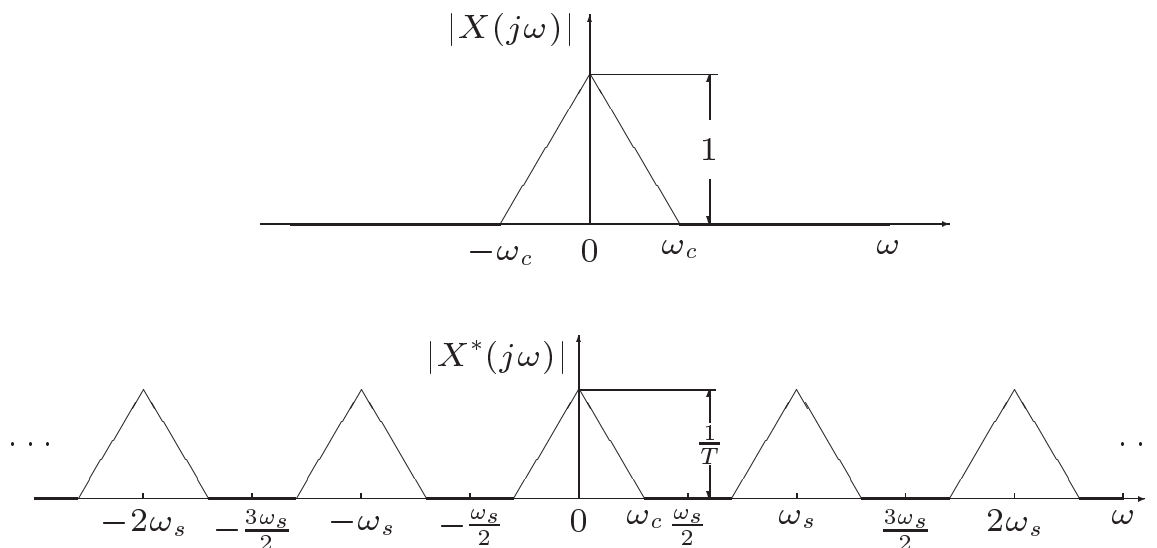
$$X^*(s) = \frac{1}{T} \sum_{n=-\infty}^{\infty} \int_0^{\infty} x(t) e^{jn\omega_s t} e^{-st} dt = \frac{1}{T} \sum_{n=-\infty}^{\infty} \int_0^{\infty} x(t) e^{-(s-jn\omega_s)t} dt$$

10. Poniamo $s' = s - jn\omega_s$, si ottiene:

$$X^*(s) = \frac{1}{T} \sum_{n=-\infty}^{\infty} \int_0^{\infty} x(t) e^{-s't} dt = \frac{1}{T} \sum_{n=-\infty}^{\infty} X(s') = \frac{1}{T} \sum_{n=-\infty}^{\infty} X(s - jn\omega_s)$$

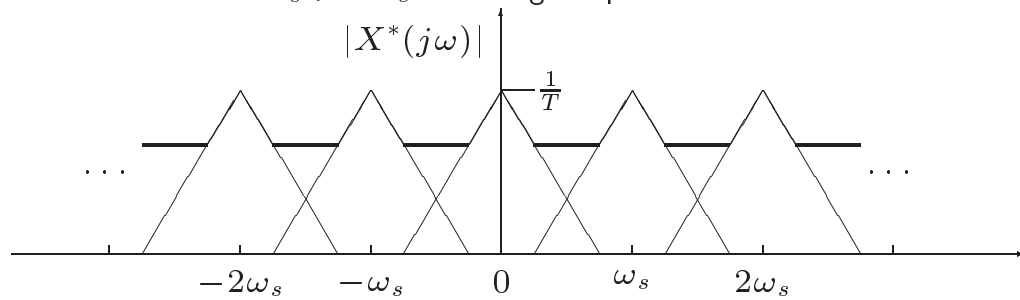
- A meno della costante moltiplicativa $1/T$, la trasformata di Laplace $X^*(s)$ del segnale campionato si ottiene dalla somma degli infiniti termini, $X(s - jn\omega_s)$, ciascuno dei quali si ottiene dalla $X(s)$ mediante traslazione di $jn\omega_s$ nel campo complesso.
- L'andamento spettrale del segnale campionato vale:

$$X^*(j\omega) = \frac{1}{T} \sum_{n=-\infty}^{\infty} X(j\omega - jn\omega_s)$$



- La condizione $\omega_s > 2\omega_c$ mantiene distinto lo spettro originario dalle componenti complementari per cui, mediante filtraggio, è possibile ricostruire completamente il segnale $x(t)$.

- Nel caso in cui la condizione $\omega_s > 2\omega_c$ non venga rispettata:



- Lo spettro originario è parzialmente sovrapposto alle componenti complementari contigue per cui mediante filtraggio non è più possibile ricavare il segnale originario a partire dal segnale campionato

Teorema di Shannon

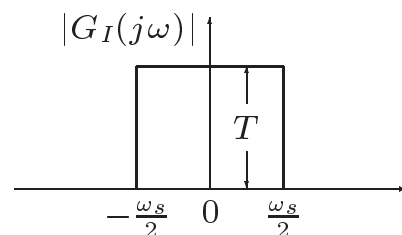
- **Teorema di Shannon**

Sia $\omega_s = \frac{2\pi}{T}$ la pulsazione di campionamento (T è il periodo di campionamento), e sia ω_c la più alta componente spettrale del segnale tempo-continuo $x(t)$. Il segnale $x(t)$ è completamente ricostruibile a partire dal segnale campionato $x^*(t)$ se e solo se la pulsazione ω_s è maggiore del doppio della pulsazione ω_c :

$$\omega_s > 2\omega_c$$

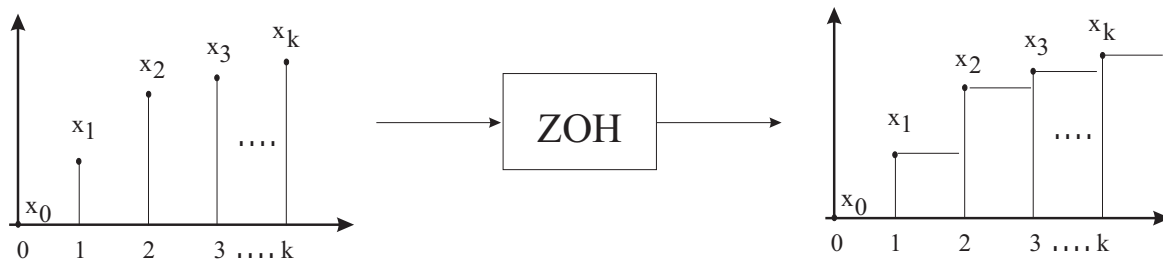
- Ricostruzione mediante filtro ideale

$$G_I(j\omega) = \begin{cases} T & -\frac{\omega_s}{2} \leq \omega \leq \frac{\omega_s}{2} \\ 0 & \text{altrove} \end{cases}$$



Il ricostruttore reale: lo Zero Order Hold (ZOH)

- Lo **ZOH** mantiene ad un valore costante i campioni su tutto il periodo di campionamento T .
- Da un punto di vista tecnologico, la funzione di **ZOH** è implementata da un **Sample and Hold**.



Caratteristica dello ZOH

- La caratteristica dello ZOH si ricava dalla funzione a gradino $s(t)$ calcolata all'istante $t = kT$ a cui si sottrae la stessa funzione calcolata in $t = kT - T$.

$$p(t) = s(t) - s(t - T)$$

$$\mathcal{L}[p(t)] = \mathcal{L}[s(t) - s(t - T)] = \frac{1 - e^{-sT}}{s}$$

