Analisi di un processo chimico attraverso l'utilizzo di un controllore LQR e regolare LQG

Pellegrino Davide e Redini Nilo

Part I
Introduzione

In questa relazione vedremo come, dato un modello linearizzato in un intorno dell'equilibrio instabile di un sistema dinamico (nel caso in esame di un sistema chimico), sia possibile stabilizzare tale sistema (discretizzato) all'origine utilizzando due diverse strategie ottime di controllo: LQR ed LQG.

In generale, dato un modello continuo, può essere espresso nel tempo discreto in forma matriciale come:

$$\begin{cases} x(k+1) = A_d x(k) + B_d u(k) \\ y(k) = C_d x(k) + D_d u(k) \end{cases} \quad (1)$$

Dove $x(k)$ rappresenta il vettore degli stati interni al tempo discreto k-esimo, $u(k)$ il vettore degli ingressi del nostro sistema, mentre le quattro matrici con pedice d rappresentano la discretizzazione delle matrici nel tempo continuo.

Capita spesso di voler osservare in uscita l'andamento degli stati interni; in questo caso la matrice C sarà una matrice identità di dimensione pari al numero di stati interni e D una matrice nulla.

Nel seguito supporemo che quanto appena descritto sia esattamente il caso in esame, sarà quindi sottointeso che la matrice C sia la matrice identità di dimensione opportuna e che D sia univocamente nulla, permettendoci di studiare così l'andamento interno del sistema.

Sarà inoltre supposto che il sistema in esame abbia riferimento nullo, in altre parole che il sistema sia autonomo, e si evolva a partire da uno stato interno x_0 scelto arbitrariamente random.

In generale per procedere allo studio e alla stabilizzazione di sistema, dobbiamo prima verificare che questo sia osservabile e controllabile; se queste verifiche hanno esito positivo, allora potremmo passare alla sua stabilizzazione posizionando i poli di quest'ultimo in posizione ottima.

Il primo passo per la stabilizzazione all'origine di un sistema è, naturalmente, la sua discretizzazione; il metodo usato in questo caso di studio è quello di Tustin nello spazio delle varibili di stato, il quale ha una corrispondenza perfetta tra il piano continuo S ed il piano discreto Z.

Questo significa che le singolarità stabili nel piano S (quindi a parte reale negativa) vengono mappate all'interno del cerchio di raggio unitario nel piano Z e viceversa, mentre quelle instabili rimangono tali una volta effettuata la trasformazione quale che sia il senso di percorrenza (da S a Z e viceversa), cosa che invece può non accadere usando il metodo dei triangoli di Eulero.

I controllori che veranno modellizati sono il regolatore lineare quadratico (LQR) ed il regolatore lineare quadratico gaussiano (LQG); nel primo caso, in fase di modellazione, supporremo che non esista alcun tipo di disturbo e che siano perfettamente noti i parametri del sistema, mentre nel secondo oltre ad introdurre dei piccoli disturbi a media nulla sui canali di ingresso e di uscita, vedremo come è possibile ricostruire gli stati interni del sistema partendo dall'analisi delle sue uscite.

Part II
Caso di studio

Il sistema che verrà esaminato nel seguito è un reattore chimico avente due ingressi, due uscite e due stati interni; nello specifico il primo ingresso (u_1) rappresenta la concentrazione della sostanza in ingresso, il secondo (u_2) il calore rimosso dal reattore, il primo stato interno (x_1) la concentrazione del reagente ed infine il secondo (x_2) la temperatura del reattore.

Nel tempo continuo tale sistema è caratterizzato dai seguenti parametri:

```
par.V = 0.1; par.R = 8.314;
par.Ca0s = 0.9934;
par.Ta0s = 310.6193;
par.dH = -4.78*10^4;
par.k0 = 72*10^9;
par.E = 8.314*10^4;
par.cp = 0.239;
par.rho = 1000;
par.F = 0.1;
par.Trs = 395.3;
par.Cas = 0.57;
par.Qbar = 0.0167;
umax = [1;par.Qbar];

A = [-par.F/par.V-par.k0*exp(-par.E/(par.R*par.Trs)), ...-0.0108-0.2445i
-par.k0*exp(-par.E/(par.R*par.Trs))*(par.E/(par.R*par.Trs^2))*par.Cas;
-par.dH/(par.rho*par.cp)*par.k0*exp(-par.E/(par.R*par.Trs)), ...
-par.F/par.V-par.dH/(par.rho*par.cp)*par.k0*exp(-par.E/(par.R*par.Trs))*
(par.E/(par.R*par.Trs^2))*par.Cas];

B = [par.F/par.V, 0; 0, 1/(par.rho*par.cp*par.V)];

C=eye(2);

D=zeros(2);
```

Quindi dalle seguenti matrici:

$$A = \begin{matrix} -1.7428 & -0.0271 \\ 148.5626 & 4.4191 \end{matrix}$$

$$B = \begin{matrix} 1 & 0 \\ 0 & 0.0418 \end{matrix}$$

$$C = \begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix}$$

$$D = \begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix}$$

È stato inoltre posto un vincolo sulla dimensione del segnale in ingresso, quest'ultimo infatti non deve mai superare un valore massimo u_{max} definito nelle specifiche.

Il primo passo per la stabilizzazione di un sistema instabile consiste nel controllare che ciò sia effettivamente possibile.

Nello specifico si deve verificare che il rango della matrice di controllabilità $[B, AB]$ (cioè costruita dalla sequenza di Krilov) sia massimo; nel caso in esame che sia uguale a 2, che è esattamente il grado della matrice A.

Oltre che controllabile il sistema deve risultare pure osservabile, caratteristica che sarà utile nel proseguo del progetto; questo si ottiene controllando che il rango della matrice di osservabilità $[C; CA]$ sia anch'esso uguale a 2.

Verificato ciò possiamo procedere con la discretizzazione del sistema; a tale scopo è stata creata un'apposita funzione ($DiscreteTustin$) la quale prende in ingresso sia le quattro matrici che identificano il sistema continuo (descritto in forma di variabili di stato) sia il tempo di campionamento, resituendo le corrispettive matrici nel tempo discreto.

1 Discretizazzione

Supponiamo di avere un sistema continuo descritto nello spazio di stato:

$$\begin{cases} \dot{x}=Ax+Bu \\ y=Cx+Du \end{cases} \quad (1.1)$$

Vogliamo determinare il corrispettivo sistema nel tempo discreto, ricavandosi le quattro matrici che lo descrivono, come riportato nella formula 1.

La soluzione del sistema continuo sarà del tipo:

$$x(t) = e^{A(t-t_0)}x(t_0) + \int_{t_o}^{t} e^{A(t-\tau)}Bu(\tau)d\tau \, (1.2)$$

Dove il primo addendo è la risposta omogenea, mentre il secondo quella particolare.

Facendo l'ipotesi che $u(\tau)$ sia costante a tratti e facendo corrispondere:

$$t_0 \to kT, t \to (k+1)T \, (1.3)$$

con T tempo di campionamento, abbiamo che:

$$x((k+1)T) = e^{AT}x(kT) + \int_{kT}^{(k+1)T} e^{A((k+1)T-\tau)}Bu(kT)d\tau \, (1.4)$$

Ottenendo dunque:

$$A_d = e^{AT} \, (1.5)$$

$$B_{d=}B\int_{kT}^{(k+1)T} e^{A((k+1)T-\tau)}d\tau \, (1.6)$$

Ponendo adesso $\sigma = (k+1)T - \tau$ ed, implicitamente, $d\sigma = -d\tau$ otteniamo:

$$-\int_T^0 e^{A\sigma} d\sigma = \int_0^T e^{A\sigma} d\sigma \, (1.7)$$

quindi:

$$B_d = B \int_0^T e^{A\sigma} d\sigma \, (1.8)$$

Per lo sviluppo di Taylor:

$$e^{A\sigma} = I + A\sigma + \frac{A^2\sigma^2}{2!} + \frac{A^3\sigma^3}{3!} + ... \, (1.9)$$

Integrando adesso tale espressione, e moltiplicando per A otteniamo:

$$A \int_0^T e^{A\sigma} d\sigma = AT + \frac{A^2T^2}{2!} + \frac{A^3T^3}{3!} + ... = e^{AT} - I \, (1.10)$$

Quindi, ritornando alle 1.8 abbiamo:

$$AB_d = (e^{AT} - I)B \, (1.11)$$

Supponendo adesso che A sia invertibile abbiamo finalmente che:

$$B_d = A^{-1}(e^{AT} - I)B \, (1.12)$$

Per quanto riguarda le matrici C e D, esse rimangono invariate nel discreto; se invece A non fosse invertibile, per trovare B_d bisogna necessariamente svolgere il calcolo integrale.

La funzione appena citata è stata implementata nel seguente modo:

```
function [Ad,Bd,Cd,Dd]=DiscreteTustin(A,B,C,D,Tc)
    Ad=expm(A*Tc);
    Bd=inv(A)*((Ad-eye(2))*B);
    Cd=C;
    Dd=D;
end
```

Come possiamo osservare le matrici C_de D_d vengono mantenute come nel caso continuo per i motivi già esposti.

A questo punto possiamo procedere con la stabilizzazione del sistema, il primo metodo che vedremo nel prossimo paragrafo sarà il regolare LQR.

2 Linear Quadratic Regulator (LQR)

Nel'ambito del controllo ottimo un regolatore LQR è definito come un compensatore dinamico, ottenuto minimizzando un indice di costo $J(x,u)$, dipendente dallo stato $x(t)$ e dal controllo $u(t)$. In generale un obiettivo nel realizzare un controllore è quello di rendere "piccolo" sia lo stato interno $x(t)$ del sistema per regolarlo verso l'origine, sia $u(t)$ per economizzare l'uso degli attuatori, in generale questi due obiettivi sono contrastanti.

In particolare dato un sistema dinamico da controlloare scritto in forma delle variabili di stato:

$$\begin{cases} x(k+1)=Ax(k)+Bu(k) \\ x(0)=definita \end{cases} \quad (2.1)$$

Si cerca la sequenza ottima di ingressi $U = \{u(0)....u(T-1)\}$, con T orizzonte temporale, che porti il sistema dallo stato da $x(0)$ all'origine minimizzando l'inidice di prestazione (prima accennato) definito come:

$$J(x,u) = \frac{1}{2}\sum_{k=0}^{T} x^T(k)Qx(k) + u^T(k)Ru(k) + x^T(T)Nx(T) \, (2.2)$$

I fattori Q ed R all'interno dell'espressione sopra, servono a pesare le componenti del sistema in modo da poter ottimizzare quest'ultimo privilegiando il consumo energetico, dando quindi maggior peso (importanza) agli ingressi del sistema, o l'equilibrio degli stati.

La matrice Q (simmetrica semi-definita positiva) misura sostanzialmente la deviazione dello stato rispetto al valore desiderato (x=0), la R (definita positiva) l'intensità dell'ingresso di controllo, mentre la N (semi-definita positiva) misura la deviazione dello stato finale rispetto allo zero.

Le due matrici Q ed R si dimensionano valutando il sistema in esame, scegliendo quali parametri vogliamo minimizzare; ogni elemento sulla diagonale della matrice Q rappresenta il peso su ogni stato interno, mentre gli altri valori rappresentano i pesi sui termini misti; stessa considerazione può essere fatta sulla matrice R considerando gli ingressi invece degli stati interni.

A questo punto quello che vogliamo determinare è il controllore proporzionale $K(k)$ tale che soddifisfi l'equazione sopra e che, una volta chiuso in reazione il sistema,lo stabilizzi (se instabile, ma è spesso questo il caso) posizionando i poli in posizione ottima; naturalmente assumendo che il sistema sia osservabile e controllabile.

Dalla teoria dei sistemi sappiamo che retroazionando negativamente un sistema autonomo avente le matrici C e D come prima definite abbiamo che:

$$u(k) = -K(k)x(k) \, (2.3)$$

Si può inoltre dimostrare che per questo tipo di problema $K(k)$ è dimensionato come:

$$K(k) = [R + B^T P(k+1)B]^{-1} B^T P(k+1)A \, (2.4)$$

Dove $P(k+1)$ risolve l'equazione discreta di Riccati (DRE) definita come:

$$\begin{cases} P(k)=A^T[P(k+1)-P(k+1)BR^{-1}B^TP(k+1)]A+Q \\ P(T)=N \end{cases} \quad (2.5)$$

Possiamo quindi dire che $K(k)$ rappresenta il controllore (proporzionale) ottimale per il controllo di un particolare sistema in esame.

Sostituendo l'equazione 2.3 nella 2.1 dopo semplici passaggi algebrici quello che otteniamo è l'espressione ad anello chiuso nella forma delle equazioni di stato:

$$\begin{cases} x(k+1)=(A-BK(k))x(k) \\ x(0)=definita \end{cases} \quad (2.6)$$

2.1 Implementazione algoritmo LQR

Tornando all'analisi del nostro problema, vediamo come possiamo simulare l'implementazione del regolatore LQR in matlab.

Dopo aver vericato la controllabilita ed osservabilità del nostro sistema, quello che dobbiamo fare prima di ricavare il K ottimo, è definire le due matrici Q ed R.

Dato il vincolo u_{max} posto in fase di progetto, la matrice Q è stata definita come una matrice identità di dimensione due, mentre la matrice R è stata definita come segue:

$$\begin{matrix} 1 & 0 \\ 0 & 19 \end{matrix}$$

In questo modo retroazionando il sistema è empiricamente verificata la disequazione $u(k) \leqslant u_{max} \forall k$

La definizione di K si ottiene invocando la funzione matlab *dlqr*, il cui scopo è quello di calcolare e restituire, oltre alla matrice K, la soluzione all'equazione di Riccati (P) ed gli autovalori in anello chiuso del sistema retroazionato con K; dando in ingresso un sistema discreto e le due matrici Q ed R.

A questo punto si può procedere semplicemente retroaziondo negativamente il sistema con il K trovato, semplicemente facendo uso della funzione $feedback$ del matlab.

Il codice appena descritto si può riassumere nel seguente modo:

```
function [sys,KLQR]=lqrControl(Ad,Bd,Cd,Dd,Tc)
        x0_lqr = randn(2,1);
        dts=ss(Ad,Bd,Cd,Dd,Tc);

        %% Definisco alcuni parametri
        weight=19;
        %peso sulle variabili di stato
        Q=eye(2);
        %peso sugli ingressi
        R=[1 0; 0 weight];

        %% DLQR
        [K,S,E]=dlqr(Ad,Bd,Q,R);
```

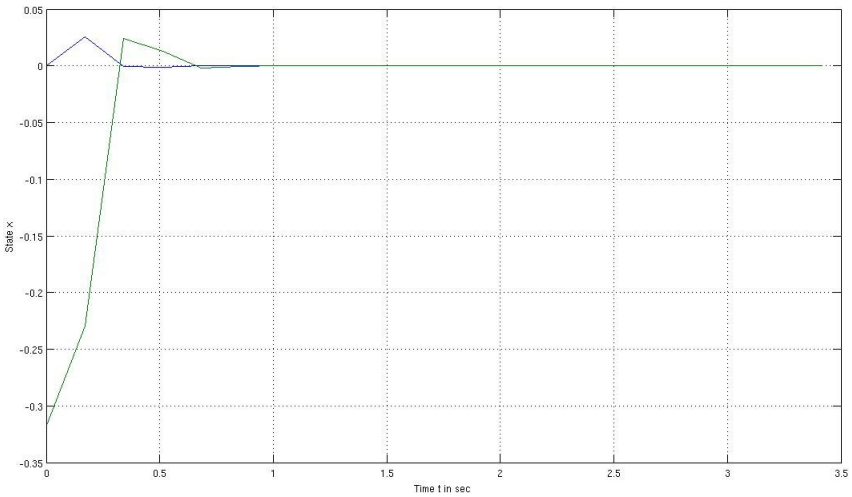
```
    % retroaziono il sistema con il K ottimo ricavato
    dts_loop=feedback(dts,K,-1);

    %simulazione e controllo specifica
    [y,t]=initial(dts_loop,x0_lqr);
    res=states(y,umax,K);

    y_lqr=y;
    t_lqr=t;
    cond_lqr=res;
end
```

Come si nota, prima di terminare la funzione, viene eseguita la simulazione dell'evoluzione libera del sistema da un x_0 random tramite la funzione matlab *initial*; quest'ultima ritorna sia l'uscita del sistema, sia gli stati interni attraversati, sia il tempo di esecuzione ed, avendo posto la matrice C come la matrice identità e la D composta da soli zeri ed essendo il riferimento nullo, il vettore degli stati interni coinciderà con quello delle uscite del sistema.

Fatto questo, tramite la funzione *states* (illustrata in seguito), si controlla che il vincolo imposto su *umax* venga sempre rispettato, quindi si ritornano i vettori uscita y, il tempo di simulazione t ed il risultato della funzione *states*.

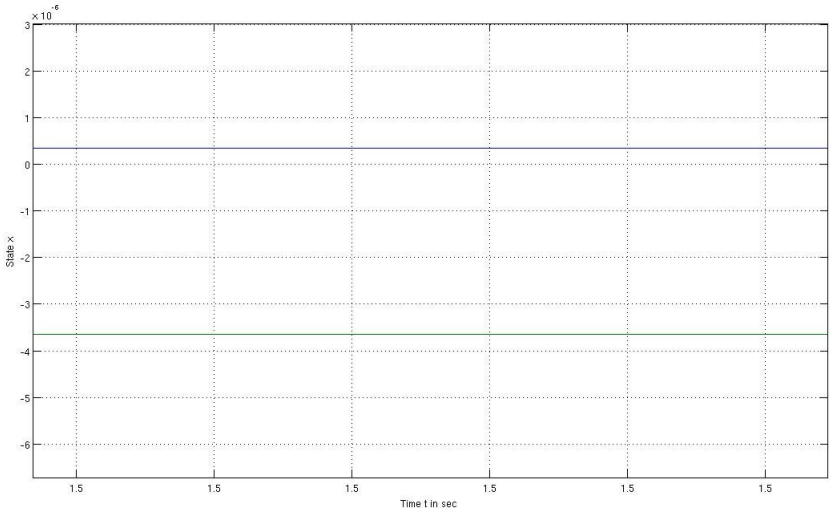
Se visualizziamo a schermo la variabile E, contenente la posizione dei poli ad anello chiuso, vediamo che un polo si trova in posizione $-0.0108 + 0.2445i$ mentre l'altro il posizione $-0.0108 - 0.2445i$; dunque, come ci aspettavamo, sono situati all'interno della circonferenza di raggio unitario nell'intorno dell'origine.

Infine il risultato ottenuto implementando il regolatore LQR è il seguente:



Il vincolo imposto è rispettato per quanto prima detto, vedremo più avanti come è stato implementato tale controllo, come possiamo inoltre vedere entrambe le variabili di stato convergono a zero in un tempo ragionevole.

Nella successiva figura possiamo vedere inoltre con che ordine di grandezza i due stati interni convergono a zero per $t = 1.5$.


×10^-6
3
2
1
0
-1
-2
-3
-4
-5
-6
State x
1.5
1.5
1.5
1.5
1.5
1.5
Time t in sec

3 Linear Quadratic Gaussian (LQG)

Il secondo tipo di controllore che prenderemo in esame sarà l'LQG o controllore lineare quadratico gaussiano.

Questo viene utilizzato nei sistemi reali, nei quali sono presenti rumori sia in ingresso che in uscita, che ne alterano l'evoluzione ed inoltre quando non si hanno abbastanza informazioni riguardo gli stati interni del processo, come spesso accade.

La costruzione di un controllore di questo tipo consiste nella progettazione di un LQR, secondo le procedure viste nel paragrafo precedente, e di un tipo di osservatore particolare chiamato filtro di Kalman opportunamente connessi tra di loro.

Se consideriamo il nostro sistema MIMO affetto da rumore, possiamo scrivere il modello matematico in forma matriciale come segue:

$$\begin{cases} x(k+1) = Ax(k) + Bu(k) + w(k) & (3.1) \\ y(k) = Cx(k) + v(k) & (3.2) \end{cases}$$

dove w_k e v_k sono rispettivamente il rumore di processo ed il rumore di misura del sistema.

Questi, per il corretto funzionamento del filtro, devono essere processi di rumore bianco con distribuzione normale (ipotesi sotto cui il filtro di Kalman è ottimo); quindi

$$w(k) \in \aleph(0, Q_k)$$
$$v(k) \in \aleph(0, R_k)$$

con R_k e Q_k le matrici di autocorrelazione.

Il filtro di Kalman è un filtro di tipo ricorsivo che valuta, a partire da una serie di misure soggette a disturbi esterni, lo stato di un sistema dinamico; sotto le precedenti ipotesi è dimostrato essere un filtro ottimo.

Lo stato stimato viene così calcolato

$$\hat{x}_k = \hat{x}_k^- + k_k(y_k - C_k\hat{x}_k^- \quad (3.3)$$

dove $\hat{x}_k^-$ è per definizione lo stato stimato prima di leggere l'uscita y_k, k_k rappresenta il guadagno di Kalman e viene ottenuto tramite

$$k_k = P_k^- C_k^T (C_k P_k^- C_k^T + R_k)^{-1} \quad (3.4)$$

con C_k matrice del processo, R_k matrice di autocorrelazione del rumore e P_k^- definita come

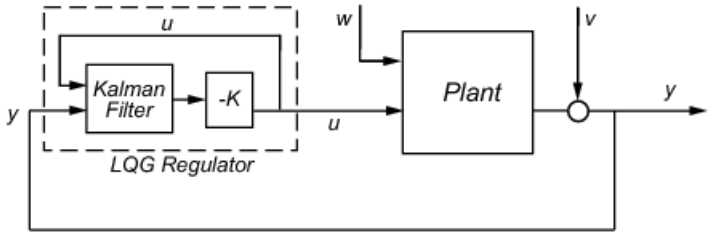
$$P_k^- = E\left[e_k^- * e_k^{-^T}\right] \quad (3.5)$$

considerando e_k^- l'errore di stima previsto, così definito $e_k^- \triangleq x_k - \hat{x}_k^-$ otteniamo

$$P_k^- = E\left[\left(x_k - \hat{x}_k^-\right) * \left(x_k - \hat{x}_k^-\right)^T\right] \quad (3.6)$$

Una volta ricavato lo stato stimato questo vieno portato in ingresso all'LQR che calcolerà il K proporzionale da utilizzare per controllare il sistema.

Il montaggio finale sarà del tipo:



3.1 Implementazione algoritmo LQG

Adesso vediamo una possibile implementazione di quanto detto con un caso di studio in Matlab.

Prendendo come riferimento il processo utilizzato nel capitolo precedente, e la sua relativa discretizzazione, si sono create le due matrici di autocorrelazione del rumore di processo w_k e del rumore di misura v_k.

Per fare ciò viene utilizzata la funzione "*Noise()*", da noi creata, che per ogni matrice di autocorrelazione si genera 20000 valori di rumore con distribuzione normale, calcola la covarianza tra due valori, di volta in volta esegue la somma ed infine ne calcola la media aritmetica; se T tendesse all'infinito avremmo esattamente le matrici di covarianza per la legge dei grandi numeri.

Possiamo quindi dire che questo algoritmo ci garantisce di avere delle matrici che rispecchiano bene l'andamento della variabili aleatorie di cui il rumore è composto.

```
function [Qn,Rn]=Noise(Tc)
        cv=zeros(2);
        T=20000;
        wx1=randn(2,1);
        for i=0:T
                wx=randn(2,1)*0.5;
                cv=cv+cov(wx,wx1);
                wx1=wx;
        end
        cv=cv./T+1;
        Qn=cv;
        cv=zeros(2);
        wx1=randn(2,1);
        for i=0:T
                wx=randn(2,1)*0.001;
```

```
            cv=cv+cov(wx,wx1);
            wx1=wx;
        end
        cv=cv./T+1;
        Rn=cv;
end
```

Una volta costruite Q_n ed R_n viene utilizzata la funzione di Matlab "*kalman()*" la quale genera un sistema KEST che servirà successivamente per calcolare la stima dello stato.

Come spiegato precedentemente nella costruzione di un LQG avremo bisogno anche di un controllore LQR per il calcolo del k proporzionale per la stabilizzazione del processo; questo si effettua tramite la funzione "*dlqr()*" che restituisce la costante desiderata.

Ottenuto sia il filtro di Kalman sia il controllore LQR non resta che creare il regolatore LQG, sempre affidandosi alle funzioni preesistenti, da montare poi in retroazione al sistema.

```
function [KLQG]=lqgControl(Ad,Bd,Cd,Dd,Tc,umax)

        R = [0.01 0;0 200]; %peso ingressi
        Q=[1e-4 0;0 1]; %peso stati interni
        %N se omesso è zero, non c'è correlazione tra i rumori.
        G = Bd;
        H = Dd;
        Pdts=ss(Ad,[Bd G],Cd,[Dd H],Tc);
        [Qn,Rn]=Noise(Tc);
        KEST = kalman(Pdts,Qn,Rn);
        K=dlqr(Ad,Bd,Q,R);
        lqg_reg = lqgreg(KEST, K, 'current')
        x = (randn(2,1)*2)';
        x_r= (randn(2,1)*2)';
        N=1;
        for i=0:Tc:10
            t(N)=i;
            wx=randn(2,1)*0.5;
            nx=randn(2,1)*0.001;
            y=(Cd*x(N,:)')'+nx';
            %registro le uscite per la plot
            out(N,:)=y;
            %regolatore lqg
            x_r(N+1,:)=lqg_reg.a*x_r(N,:)'+lqg_reg.b*y';
            u=lqg_reg.c*x_r(N,:)'+lqg_reg.d*y';
            if u>umax
                disp('The condition imposed is
                    not satisfied for LQG regulator');
            end
            x(N+1,:)=(Ad*x(N,:)'+Bd*(u+wx))';
            N=N+1;
        end
        plot(t,out);
        KLQG=K;
end
```

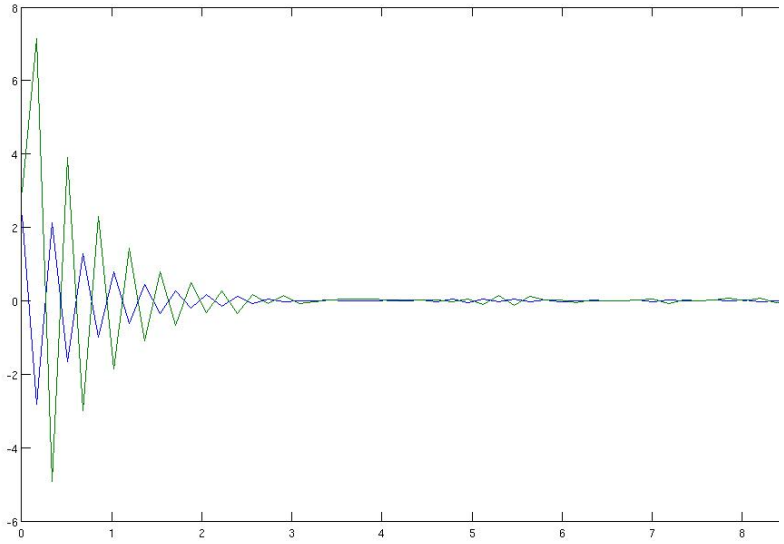
Infine, costruito il controllore si effettua una simulazione del sistema partendo da uno stato iniziale casuale per verificare che i vincoli imposti in fase di

progetto vengano rispettati.

Più precisamente ad ogni iterazione viene controllato che l'ingresso al sistema rientri entro un certo valore umax fissato in precedenza. Per garantire il rispetto di questa condizione sono state manipolate le matrici R e Q che rappresentano i pesi per la funzione di costo J(x,u).

Si è visto che dando maggior peso al controllo delle temperature del reattore chimico il nostro sistema raggiunge la stabilità nel rispetto delle specifiche per un totale di prove che supera il milione.

Come si può vedere dal grafico sottostante il sistema si stabilizza in circa 3 secondi, un risultato decisamente accettabile considerando la presenza di disturbi nel processo ma ovviamente peggiore rispetto al controllore LQR realizzato nel capitolo precedente, dato che in quest'ultimo caso ne erano presenti rumori, ne avevamo bisogno di un osservatore.



3.2 Visualizzazione dell'evoluzione del sistema e controllo sui vincoli

Una volta modellizzato il controllore e posto in retroazione al sistema, dobbiamo verificare che i due stati interni di quest'ultimo convergano a zero e che il vincolo posto in fase di specifica venga rispettato.

Per quanto riguarda questo ultimo passaggio, come abbiamo visto in precedenza, il controllo per il regolatore LQG è stato fatto durante la produzione del vettore uscita y; per quanto riguarda il regolatore LQR come già accennato abbiamo fatto uso di una particolare funzione chiamata *states*.

Come possiamo vedere nel codice sotto, l'implementazione di questo controllo è veramente semplice: viene preso il vettore x contenente tutti gli stati

attraversati dal nostro sistema, ognuno di questi viene moltiplicato per il coefficiente K (il motivo di tale operazione è gia stata spiegata precedentemente), e viene controllato infine che tale risultato sia minore, o al più uguale, ad u_{max}; se tale controllo risulta positivo un flag viene settato a 1, altrimenti a 0.

```
%% controllo degli stati di uscita
function [res]= states(x,umax,k)
        res=1;
        for i=1:size(x,1)
                xc=x(i,:);
                uc=k*xc';
                if(umax<uc)
                        res=0;
                        break;
                end
        end
end
```

Per quanto riguarda la stampa a schermo dei grafici inerenti all'evoluzione degli stati interni, la funzione adibita a tale scopo è altrettanto semplice.

Tale funzione prende in ingresso il vettore uscita y, il vettore dei tempi di simulazione t, il flag inerente al rispetto del vincolo u_{max} ed il tipo di contrllo usato.

Lo scopo della funzione è semplicemente quello di stampare a schermo, attraverso il metodo *plot*, due grafici: il primo rappresenta l'evoluzione degli stati interni del nostro sistema con regolatore LQR ed il secondo con regolatore LQG.

Part III
Codice

Il codice appena descritto è intermamente contenuto nella funzione ChSys():

```
function [] = ChSys()
      clc; clear; close all

      par.V = 0.1; par.R = 8.314;
      par.Ca0s = 0.9934;
      par.Ta0s = 310.6193;
      par.dH = -4.78*10^4;
      par.k0 = 72*10^9; par.E = 8.314*10^4;
      par.cp = 0.239; par.rho = 1000;
      par.F = 0.1; par.Trs = 395.3;
      par.Cas = 0.57; par.Qbar = 0.0167;

      umax = [1;par.Qbar];

      B=[par.F/par.V,0;0,1/(par.rho*par.cp*par.V)];
      A=[-par.F/par.V-par.k0*exp(-par.E/(par.R*par.Trs)),
      -par.k0*exp(-par.E/(par.R*par.Trs))*
      (par.E/(par.R*par.Trs^2))*par.Cas;
      -par.dH/(par.rho*par.cp)*par.k0*
      exp(-par.E/(par.R*par.Trs)),
      -par.F/par.V-par.dH/(par.rho*par.cp)*par.k0*
      exp(-par.E/(par.R*par.Trs))*
      (par.E/(par.R*par.Trs^2))*par.Cas];

      %% Imposto parametri e condizioni del sistema
      C=eye(2);
      D=zeros(2);

      %% Visualizzo a schermo la controllabilità
      %%ed osservabilità
      if (rank([B,A*B])==2)
            disp('The process can be controlled');
    end

      if (rank([C;C*A])==2)
            disp('The process can be observed');
      end

      %% Discretizzo il sistema
      Tc = 2*pi/(max(abs(eig(A)))*10);
```

```
        [Ad,Bd,Cd,Dd]=DiscreteTustin(A,B,C,D,Tc);

        % lQR
        [t_lqr,out_lqr,cond_lqr]=lqrControl(Ad,Bd,Cd,Dd,Tc,umax);

        %LQG
        [t_lqg,out_lqg,cond_lqg]=lqgControl(Ad,Bd,Cd,Dd,Tc,umax);

        PlotStates(out_lqr,t_lqr,cond_lqr,0);
        PlotStates(out_lqg,t_lqg,cond_lqg,1);
end

%% Discretizzazione
function [Ad,Bd,Cd,Dd]=DiscreteTustin(A,B,C,D,Tc)
        Ad=expm(A*Tc);
        Bd=inv(A)*((Ad-eye(2))*B);
        Cd=C;
        Dd=D;
end

function [Qn,Rn,CostW,CostV]=Noise()
        cv=zeros(2);
        T=20000;
        CostW=1e-2;
        CostV=0.001;
        wx1=randn(2,1)*CostW;
        for i=0:T
                wx=randn(2,1)*CostW;
                cv=cv+cov(wx,wx1);
                wx1=wx;
        end

        cv=cv./(T+1);
        Qn=cv;
        cv=zeros(2);

        wx1=randn(2,1)*CostV;
        for i=0:T
                wx=randn(2,1)*CostV;
                cv=cv+cov(wx,wx1);
                wx1=wx;
        end

        cv=cv./(T+1);
        Rn=cv;
end
```

```
%% LQG
function [time,sys_out,cond_lqg]=lqgControl(Ad,Bd,Cd,Dd,Tc,umax)
        cond=1;
        %peso ingressi
        R = [0.01  0;0  200];

        %peso stati interni
        Q = [1e-4  0;0  1];

        %N se omesso è zero, non c'è correlazione tra i rumori.

        Pdts=ss(Ad,[Bd eye(2)],Cd,[Dd Dd],Tc);
        [Qn,Rn,costW,costN]=Noise();

        KEST = kalman(Pdts,Qn,Rn);
        K=dlqr(Ad,Bd,Q,R);
        lqg_reg = lqgreg(KEST, K, 'current');

        x = (randn(2,1)*2)';
        x_r= (randn(2,1)*2)';
        N=1;

        for i=0:Tc:10
                t(N)=i;
                wx=randn(2,1)*costW;
                nx=randn(2,1)*costN;
                y=(Cd*x(N,:)')'+nx';

                %registro le uscite per la plot
                out(N,:)=y;

                %regolatore lqg
                x_r(N+1,:)=lqg_reg.a*x_r(N,:)'+lqg_reg.b*y';
                u=lqg_reg.c*x_r(N,:)'+lqg_reg.d*y';

                if u>umax
                        cond=0;
                end

                x(N+1,:)=(Ad*x(N,:)'+Bd*u+ wx)';
                N=N+1;
        end

        cond_lqg=cond;
        time=t;
```

```
        sys_out=out;
end

%% LQR
function [t_lqr,y_lqr,cond_lqr]=lqrControl(Ad,Bd,Cd,Dd,Tc,umax)
        x0_lqr = randn(2,1);
        dts=ss(Ad,Bd,Cd,Dd,Tc);

        %% Definisco alcuni parametri
        %peso sulle variabili di stato
        weight=19;

        %peso sugli ingressi
        Q=eye(2);

        R=[1 0; 0 weight];

        %% DLQR
        [K,S,E]=dlqr(Ad,Bd,Q,R);

        % retroaziono il sistema con il K ottimo ricavato
        dts_loop=feedback(dts,K,-1);

        %simulazione e controllo specifica
        [y,t]=initial(dts_loop,x0_lqr);

        res=states(y,umax,K);
        y_lqr=y;
        t_lqr=t;
        cond_lqr=res;
end

%% Stampa l'evoluzione libera del sistema
function PlotStates(y,t,res,type)
        if(res==1)
                figure (1)
                title('LQR and LQG controller');

                if type==0
                        disp('The condition imposed is
                                satisfied for LQR controller');

                        subplot(2,1,1),plot(t,y,'linewidth',1)
                end

                if type==1
```

```
                disp('The condition imposed is
                      satisfied for LQG controller');

                subplot(2,1,2),plot(t,y,'linewidth',1)
            end

            xlabel('Time t in sec'); ylabel('State x')
            grid on; box on;
        end

        if(res==0)
            disp('The condition imposed is not satisfied');
        end
end

%% controllo degli stati di uscita
function [res]= states(x,umax,k)
        res=1;
        for i=1:size(x,1)
            xc=x(i,:);
            uc=k*xc';
            if(umax<uc)
                res=0;
                break;
            end
        end
end
```