

باسمه تعالی

نگاهی به کاربرد نرم افزار Matlab در کنترل مدرن

(ضمیمه کتاب "اصول کنترل مدرن" تألیف دکتر علی خاکی صدیق)

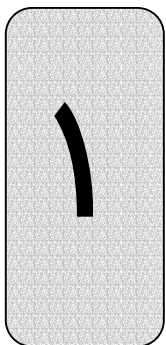
تهیه کننده: علی مرادی امانی

بهار ۱۳۸۲

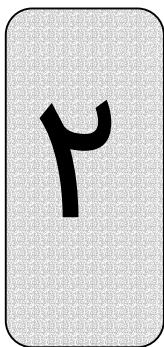
فهرست

۱- مقدمه	
۲- بررسی دستورهای نرم افزار Matlab مرتبط با بحث کنترل مدرن	
۱-۲- acker دستور	
۲-۲- canon دستور	
۳-۲- care دستور	
۴-۲- cdf2rdf دستور	
۵-۲- ctrb دستور	
۶-۲- Ctrbf دستور	
۷-۲- diag دستور	
۸-۲- eig دستور	
۹-۲- estim دستور	
۱۰-۲- exam دستور	
۱۱-۲- Gram دستور	
۱۲-۲- initial دستور	
۱۳-۲- kalman دستور	
۱۴-۲- Lyap دستور	
۱۵-۲- null دستور	
۱۶-۲- obsv دستور	
۱۷-۲- obsvf دستور	
۱۸-۲- place دستور	
۱۹-۲- ss دستور	
۲۰-۲- ssdata دستور	
۲۱-۲- ss2ss دستور	
۲۲-۲- ss2tf دستور	
۲۳-۲- tf2ss دستور	
۳- بررسی برنامه های مربوط به مثالهای متن کتاب	
۱-۳- رسم پاسخهای سیستم غیرخطی (مثال ۳-۵)	
۲-۳- طراحی فیدبک حالت با دستور acker (مثال ۸-۶)	
۳-۳- فیدبک حالت با کنترل انتگرال (مثال ۹-۶)	
۴-۳- اثر تغییر فیدبک حالت بر رفتار دینامیکی سیستم (مثال ۱۰-۶)	
۵-۳- طراحی رؤیتگر حالت (مثال ۱-۷)	
۶-۳- رؤیتگر مرتبه کاهش یافته (مثال ۲-۷)	
۷-۳- کنترل فیدبک حالت با رؤیتگر (مثال ۳-۷)	

	۳-۸- کنترل فیدبک حالت با ر‌ؤیتگر مرتبه کاهش یافته (مثال ۷-۴)
	۳-۹- طراحی کنترل کننده بهینه به صورت فیدبک حالت (مثال ۸-۱)
	۳-۱۰- کنترل بهینه و حساب تغییرات (مثال ۸-۲)
	۳-۱۱- کنترل بهینه و درجه پایداری (مثال ۸-۳)
	۳-۱۲- فیلتر کالمن (مثال ۸-۴)
	۳-۱۳- کنترل کننده فیدبک خروجی با فیلتر کالمن (مثال ۸-۵)



مقدمه



بررسی دستوره‌ای نرم افزار Matlab مرتبط با بحث کنترل مدرن

۲-۱- دستور acker:

هدف: طراحی جایاب قطب برای سیستمهای تک ورودی

• نگارش دستورالعمل:

$$K = \text{acker}(A, B, P)$$

• توضیحات:

سیستم تک ورودی زیر مفروض است:

$$\dot{x} = Ax + Bu$$

می خواهیم بردار فیدبک حالت K را چنان طراحی کنیم که قطبهای سیستم حلقه بسته در محلهای تعیین شده در بردار P قرار گیرند. دستور $K = \text{acker}(A, B, P)$ با استفاده از فرمول آکرمن بردار بهره فیدبک (K) را چنان مشخص می کند که قانون فیدبک $u = -Kx$ ، قطبهای سیستم حلقه بسته را در محلهای تعیین شده در بردار P قرار دهد. به عبارت دیگر مقادیر ویژه ماتریس $A - BK$ در محل اعضای بردار P قرار می گیرند. به عنوان نمونه، در تکه برنامه زیر با طراحی فیدبک حالت K ، قطبهای سیستم ناپایدار به $P = [-2, -3]$ منتقل می شوند.

$$A = [2, 0, -1, -1]$$

$$B = [1; 1]$$

$$P = [-2, -3]$$

$$K = \text{acker}(A, B, P)$$

علاوه بر این چنانچه سیستم تک خروجی باشد میتوان از دستور acker برای محاسبه بهره رویتگر لیونبرگر نیز استفاده کرد. اگر $y = Cx$ معادله خروجی سیستم باشد، دستور زیر بردار L را به عنوان بهره رویتگر محاسبه خواهد کرد:

$$L = (\text{acker}(A', C', P))'$$

و قطبهای رویتگر در محلهای مشخص شده توسط بردار P قرار می گیرند.

• محدودیتها:

۱. دستور acker فقط در سیستمهای تک ورودی قابل استفاده است.
۲. (A, B) باید کنترل پذیر باشد.
۳. این روش از قابلیت اطمینان بالایی برخوردار نیست و چنانچه درجه سیستم بیش از ۵ بوده یا کنترل پذیری سیستم ضعیف باشد دچار اشکال خواهد شد.

• دستورات مرتبط:

دستور place عمل جایابی قطب را در سیستمهای چند ورودی انجام می دهد.

۲-۲- دستور canon:

هدف: محاسبه تحقق های کانونیکال در فضای حالت

• نگارش دستورالعمل:

$$CSYS = \text{canon}(SYS, 'TYPE')$$

$$[CSYS, T] = \text{canon}(SYS, 'TYPE')$$

- توضیحات: این دستور، یک تحقق کانونیکال در فضای حالت برای سیستم LTI پیوسته یا گسسته زمان که با SYS مشخص شده است ارائه می دهد. دو نوع تحقق کانونیکال در این دستورالعمل در نظر گرفته شده است که توسط متغیر $TYPE$ انتخاب می گردد.

الف - فرم MODAL :

دستور $CSYS = \text{canon}(SYS, 'modal')$ یک تحقق MODAL از سیستم SYS را بدست آورده و در CSYS می‌ریزد. در این تحقق، مقادیر ویژه حقیقی SYS روی قطرهای ماتریس A قرار می‌گیرند و به اِزاء هر زوج قطب موهومی یک بلوک 2×2 در ماتریس A ظاهر می‌شود. به عنوان مثال برای سیستمی با مقادیر ویژه $(\lambda_1, \sigma \pm j\omega, \lambda_2)$ ، ماتریس A در تحقق فرم MODAL به صورت زیر خواهد بود:

$$A = \begin{bmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \sigma & \omega & 0 \\ 0 & -\omega & \sigma & 0 \\ 0 & 0 & 0 & \lambda_2 \end{bmatrix}$$

ب - فرم COMPANION :

دستور $CSYS = \text{canon}(SYS, 'companion')$ یک تحقق COMPANION از SYS ارائه می‌دهد که در آن ضرایب معادله مشخصه سیستم روی ستون منتهی الیه سمت راست ماتریس A قرار می‌گیرد. اگر معادله مشخصه سیستم به صورت زیر باشد :

$$P(s) = s^n + a_1 s^{n-1} + \dots + a_{n-1} s + a_n$$

آنگاه ماتریس A در تحقق COMPANION به صورت زیر خواهد بود:

$$A = \begin{bmatrix} 0 & 0 & \dots & 0 & 0 & -a_n \\ 1 & 0 & \dots & 0 & 0 & -a_{n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 1 & 0 & -a_2 \\ 0 & 0 & 0 & 0 & 1 & -a_1 \end{bmatrix}$$

اگر SYS به صورت مدل فضای حالت وجود داشته باشد آنگاه دستور $[CSYS, T] = \text{canon}(SYS, 'TYPE')$ ماتریس تبدیل T را که توسط آن حالت‌های سیستم اصلی به حالت‌های سیستم کانونیکال مرتبط می‌شوند نیز ارائه می‌دهد: $x_c = T.x$

چنانچه سیستم SYS به فرم فضای حالت نباشد، ماتریس T تهی خواهد بود. روند انجام عملیات در این دستور به این صورت است که ابتدا چنانچه سیستم SYS به فرم حالت تبدیل یا قطب و صفر داده شده باشد با استفاده از دستور SS فرم فضای حالت آن بدست می‌آید. در تحقق MODAL ماتریس P که مجموعه بردارهای ویژه ماتریس A است محاسبه می‌شود و سپس با معادلات زیر تبدیل تشابهی $T = P^{-1}$ به سیستم اعمال شده و فرم کانونیکال بدست می‌آید:

$$\dot{x}_c = P^{-1} A P x_c + P^{-1} B u$$

$$y = C P x + D u$$

به این ترتیب ماتریس تبدیل حالت T که از نتایج اجرای دستورالعمل است همان P^{-1} خواهد بود. تحقق فرم COMPANION نیز با استفاده از تبدیل تشابهی از روی ماتریس کنترل پذیری بدست می‌آید.

• مثال: سیستم زیر مفروض است:

$$\dot{x} = \begin{bmatrix} 1 & 2 & 0 \\ -1 & 5 & 6 \\ 3 & 0.5 & 2 \end{bmatrix} x + \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} u$$

$$y = [1 \quad 0 \quad 0] x$$

مقادیر ویژه ماتریس A عبارتند از :

$$\lambda_{1,2} = 0.6715 \pm j2.571$$

$$\lambda_3 = 6.6571$$

برای بدست آوردن تحقق کانونیکال MODAL سیستم تکه برنامه زیر را داریم:

```
A=[1 2 0;-1 5 6;3 0.5 2];
B=[1;1;1];
C=[1 0 0];
D=0;
SYS=ss(A,B,C,D);
[CSYS,T]=canon(SYS,'modal');
[Ac,Bc,Cc,Dc]=ssdata(SYS);
```

که در آن Ac , Bc , Cc , Dc ماتریسهای مربوط به تحقق کانونیکال و T تبدیل تشابهی مربوطه است. تحقق کانونیکال به صورت زیر خواهد بود:

$$\begin{aligned} \dot{x}_c &= A_c x_c + B_c u = \begin{bmatrix} 0.6715 & 2.571 & 0 \\ -2.571 & 0.6715 & 0 \\ 0 & 0 & 6.6571 \end{bmatrix} x_c + \begin{bmatrix} 0.2776 \\ -1.2267 \\ 1.93 \end{bmatrix} u \\ y &= C_c x_c + D_c u = [0.473 \quad -0.2081 \quad 0.3178] x_c \end{aligned}$$

• محدودیتها:

۱. این دستور زمانی پاسخ صحیح می‌دهد که ماتریس A مقدار ویژه تکراری نداشته باشد.

۲. در تحقق COMPANION سیستم باید از اولین ورودی کنترل پذیر باشد.

۳. The companion form is often poorly conditioned for most state-space computation avoid using it when possible.

• دستورات مرتبط:

دستور ss2ss برای انجام تبدیل تشابهی به کار می‌رود.

۲-۳ دستور care:

• هدف: حل معادله ریکاتی جبری زمان-پیوسته

• نگارش:

```
[X,L,G,rr]=care(A,B,Q)
[X,L,G,rr]=care(A,B,Q,R,S,E)
[X,L,G,report]=care(A,B,Q,...,'report')
[X1,X2,L,report]=care(A,B,Q,...,'implicit')
```

• توضیحات:

دستور $[X,L,G,rr]=care(A,B,Q)$ پاسخ منحصر بفرد معادله ریکاتی زیر را محاسبه می‌کند:

$$Ric(X) = A^T X + XA - XBB^T X + Q = 0$$

به نحوی که همه مقادیر ویژه ماتریس $A-BB^T X$ سمت چپ محور موهومی قرار گیرند. ماتریس X متقارن^۱ است و پاسخ پایدار ساز^۲ معادله $Ric(X)=0$ نامیده می‌شود. این دستور موارد زیر را نیز نتیجه می‌دهد:

۱- بردار L که شامل مقادیر ویژه ماتریس $A-BB^T X$ است.

۲- ماتریس بهره $G=B^T X$

¹ Symmetric

² Stabilizing

۳- مانده نسبی^۳ که به صورت زیر محاسبه می‌شود:

$$rr = \frac{\|Ric(X)\|_F}{\|X\|_F}$$

دستور $[X, L, G, rr] = \text{care}(A, B, Q, R, S, E)$ معادله ریکاتی عمومی تری را حل می‌کند:

$$Ric(X) = A^T X E + E^T X A - (E^T X B + S) R^{-1} (B^T X E + S^T) + Q = 0$$

ماتریس بهره در این حالت $G = R^{-1} (B^T X E + S^T)$ بوده و نیز مقادیر ویژه حلقه بسته در بردار $L = \text{eig}(A - B * G, E)$ قرار می‌گیرند.

دو نگارش دیگر دستور care برای کمک به استفاده از آن در کاربردهای مختلف نظیر طراحی کنترل کننده H_∞ در نظر گرفته شده‌اند.^۴

مثال: با فرض

$$A = \begin{bmatrix} -3 & 2 \\ 1 & 1 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, C = [1 \quad -1], r = 3$$

معادله ریکاتی

$$A^T X + X A - X B R^{-1} B^T X + C^T C = 0$$

با استفاده از برنامه زیر قابل حل است:

```
A=[ -3 2; 1 1];
B=[ 0; 1];
C=[ 1 -1];
R=3;
[X, L, g]=care(A, B, C'*C, r);
```

این برنامه پاسخ معادله ریکاتی را به فرم زیر نتیجه می‌دهد:

$$X = \begin{bmatrix} 0.5895 & 1.8216 \\ 1.8216 & 8.8188 \end{bmatrix}$$

مقایسه مقادیر ویژه ماتریس A با ماتریس A-B*g نشان می‌دهد که این پاسخ پایدار ساز است:

```
> [eig(A) eig(A-B*g)]
ans =
    -3.4495    -3.5026
     1.4495    -1.4370
```

• آلوگوریتم:

دستور care از ماتریس هامیلتونین، هامیلتونین توسعه یافته یا آلوگوریتم QZ برای حل معادله ریکاتی استفاده می‌کند.^۵

• محدودیتها:

زوج (A,B) باید پایداری پذیر باشد. بعلاوه، ماتریس هامیلتونین هیچ مقدار ویژه‌ای روی محور موهومی نداشته باشد. برای این منظور تحقق یکی از شرایط زیر کافیت:

۱- زوج (Q,A) آشکاری پذیر باشد در حالتی که: $S=0, R>0$

$$2- \begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} > 0$$

^۳ Relative residual

^۴ برای توضیحات کاملتر در این مورد به help نرم افزار Matlab مراجعه شود.

^۵ این آلوگوریتمها در مرجع زیر معرفی شده‌اند:

Arnold, W.F., III and A.J. Laub, "Generalized Eigenproblem Algorithms and Software for Algebraic Riccati Equation", Proc. IEEE, 72(1984), pp. 1746-1754

- دستورهای مرتبط:

dare: حل معادلات ریکاتی زمان-گسسته

lyap: حل معادلات لیاپانوف زمان-پیوسته

۲-۴- دستور cdf2rdf:

- هدف: تبدیل ماتریس قطری با عناصر مختلط به ماتریس قطری با بلوکهای حقیقی.

- نگارش دستورالعمل: $[V, D] = \text{cdf2rdf}(V, D)$

- توضیحات:

چنانچه مقادیر ویژه ماتریس X موهومی باشند، ماتریسهای V و D که از اجرای دستور $[V, D] = \text{eig}(X)$ بدست می آیند دارای عناصر موهومی خواهند بود. دستور cdf2rdf ماتریس قطری با عناصر موهومی D را به یک ماتریس قطری-بلوکی با بلوکهای حقیقی تبدیل می کند. هر زوج از مقادیر ویژه موهومی طبق آنچه در بخش ۲-۶-۳ گفته شد با یک بلوک 2×2 حقیقی متناظر خواهد شد. البته در این تبدیل ماتریس V نیز تغییر خواهد کرد و دیگر V شامل بردارهای ویژه سیستم نخواهد بود؛ اما تغییرات V به نحوی است که کماکان رابطه $X*V = V*D$ برقرار باشد.

- مثال: ماتریس X را به صورت زیر در نظر می گیریم:

$$X = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 0 & -5 & 4 \end{bmatrix}$$

که یک جفت مقدار ویژه موهومی دارد:

» $[V, D] = \text{eig}(X)$

$$V = \begin{bmatrix} 1.0000 & 0.4002-0.0191i & 0.4002+0.191i \\ 0 & 0.6479 & 0.6479 \\ 0 & 0+0.6479i & 0-0.6479i \end{bmatrix}$$

$$D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 4+5i & 0 \\ 0 & 0 & 4-5i \end{bmatrix}$$

تبدیل این نمایش به فرم قطری بلوکی حقیقی، نتیجه زیر را حاصل خواهد کرد:

» $[V, D] = \text{cdf2rdf}(V, D)$

$$V = \begin{bmatrix} 1 & 0.4002 & -0.0191 \\ 0 & 0.6479 & 0 \\ 0 & 0 & 0.6479 \end{bmatrix}$$

$$D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 4 & 5 \\ 0 & -5 & 4 \end{bmatrix}$$

۲-۵- دستور ctrb:

- هدف: محاسبه ماتریس کنترل پذیری

- نگارش دستورالعمل:

$\text{Co} = \text{ctrb}(A, B)$

$\text{Co} = \text{ctrb}(\text{SYS})$

$\text{Co} = \text{ctrb}(\text{SYS.a}, \text{SYS.b})$

- توضیحات: چنانچه معادله حالت سیستم به فرم $\dot{x} = Ax + Bu$ باشد که در آن $A_{n \times n}$ و $B_{n \times m}$ است، ماتریس کنترل پذیری سیستم (که با دستور ctrb محاسبه می شود) به صورت زیر خواهد بود:

$$Co = [B \quad AB \quad A^2B \quad \dots \quad A^{n-1}B]$$

واضح است که سیستم زمانی کنترل پذیر کامل است که رتبه ماتریس Co کامل باشد.

- مثال: ماتریسهای A و B را به صورت زیر در نظر می گیریم:

$$A = \begin{bmatrix} 1 & 1 \\ 4 & -2 \end{bmatrix}, B = \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}$$

برنامه زیر رتبه ماتریس کنترل پذیری را محاسبه می کند:

```
»A=[1 1;4 -2];
»B=[1 -1;1 -1];
»Co=ctrb(A,B)
Co=
    1    -1     2    -2
    1    -1     2    -2
»rank(Co)
ans=
    1
```

چون رتبه Co کامل نیست، لذا سیستم کنترل پذیر کامل نمی باشد. تعداد حالت های کنترل ناپذیر سیستم به صورت زیر محاسبه می شود:

```
»unco=length(A)-rank(Co)
ans=
    1
```

و لذا سیستم یک حالت کنترل ناپذیر دارد.

- محدودیتها:

محاسبه Co در پاره ای موارد دچار خطا می شود. این موضوع را با یک مثال ساده بررسی می کنیم:

$$A = \begin{bmatrix} 1 & \delta \\ 0 & 1 \end{bmatrix}, B = \begin{bmatrix} 1 \\ \delta \end{bmatrix}$$

اگر $\delta \neq 0$ باشد این زوج کنترل پذیر خواهد بود:

$$Co = [B \quad AB] = \begin{bmatrix} 1 & 1+\delta^2 \\ \delta & \delta \end{bmatrix}$$

اما اگر $\delta^2 < \epsilon$ باشد که در آن ϵ ضریب نسبی محاسبات کامپیوتر است آنگاه رایانه δ^2 را صفر در نظر خواهد گرفت و سیستم را کنترل ناپذیر نشان خواهد داد در حالی که سیستم کنترل پذیر است. در این شرایط بهتر است از دستور ctrbf استفاده شود.

- دستورات مرتبط:

obsv: محاسبه ماتریس رویت پذیری سیستم

ctrbf: جداسازی حالت های کنترل پذیر و کنترل ناپذیر سیستم

۲-۶- دستور Ctrbf:

- هدف: تهیه فرم پلکانی کنترل پذیر (جداسازی حالت های کنترل پذیر و کنترل ناپذیر)

[Abar, Bbar, Cbar, T, K]=ctrbf(A, B, C)

- نگارش دستورالعمل:

[Abar, Bbar, Cbar, r, k]=ctrbf(A, B, C, tol)

• توضیحات:

اگر رتبه ماتریس کنترل پذیری زوج (A,B) کوچکتر از n (n مرتبه A است) باشد، آنگاه یک تبدیل تشابهی مانند T وجود دارد که می تواند حالت های کنترل ناپذیر سیستم را از حالت های کنترل پذیر جدا کند. ماتریس تبدیل T unitary است^۱ و داریم :

$$\bar{A} = TAT^T, \bar{B} = TB, \bar{C} = CT^T$$

این تبدیل ماتریس A را به یک ماتریس پلکانی چنان تبدیل می کند که بخش مربوط به حالت های کنترل ناپذیر (A_{uc}) در گوشه سمت چپ بالای ماتریس A قرار گیرد:

$$\bar{A} = \begin{bmatrix} A_{uc} & 0 \\ A_{21} & A_c \end{bmatrix}, \bar{B} = \begin{bmatrix} 0 \\ B_c \end{bmatrix}, \bar{C} = [C_{uc} \quad C_c]$$

در این حالت زوج (A_c, B_c) کنترل پذیر بوده ، تمام مقادیر ویژه A_{uc} کنترل ناپذیر هستند و خواهیم داشت:

$$C_c(sI - A)^{-1}B_c = C(sI - A_c)^{-1}B_c$$

یعنی حالت های کنترل ناپذیر در تعیین تابع تبدیل سیستم اثری ندارد.

دستور $[Abar, Bbar, Cbar, T, K] = ctrb(A, B, C)$ سیستم فضای حالت نمایش داده شده با A, B و C را به سیستمی با $\bar{A}, \bar{B}, \bar{C}$ فوق الذکر تبدیل می کند. T ماتریس تبدیل تشابهی و K یک بردار به طول n است که n ابعاد ماتریس A می باشد. هر عضو K نشان می دهد که در هر بار تکرار برای محاسبه ماتریس تبدیل تشابهی چند حالت کنترل پذیر استخراج شده است .

بنابراین تعداد اعضای غیر صفر بردار K نشان می دهد برای محاسبه ماتریس تبدیل T چند تکرار لازم است. بعلاوه $SUM(K)$ ^۲ تعداد حالت های A_c را نشان می دهد.

عنصر tol در دستور $ctrb(A, B, C, tol)$ مقدار تolerانس مجاز در محاسبه زیر فضاهای کنترل پذیر/کنترل ناپذیر را نشان می دهد. اگر مقدار این تolerانس تعیین نشده باشد، مقدار $10^{-n} * norm(A, 1) * eps$ به عنوان پیش فرض در نظر گرفته می شود.

• مثال: سیستم زیر را در نظر بگیرید :

$$A = \begin{bmatrix} 1 & 1 \\ 4 & -2 \end{bmatrix}, B = \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}, C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

با اعمال دستور

$$[Abar, Bbar, Cbar, T, K] = ctrb(A, B, C)$$

حالت های کنترل پذیر و کنترل ناپذیر سیستم از هم جدا شده و سیستم جدید را به صورت زیر خواهیم داشت:

Abar=

$$\begin{bmatrix} -3 & 0 \\ -3 & 2 \end{bmatrix}$$

Bbar=

$$\begin{bmatrix} 0 & 0 \\ 1.4142 & -1.4142 \end{bmatrix}$$

Cbar=

$$\begin{bmatrix} -0.7071 & 0.7071 \\ 0.7071 & 0.7071 \end{bmatrix}$$

ماتریس تبدیل تشابهی T و بردار K نیز به صورت زیر خواهند بود:

T=

$$\begin{bmatrix} -0.7071 & 0.7071 \\ 0.7071 & 0.7071 \end{bmatrix}$$

K=

$$\begin{bmatrix} 1 & 0 \end{bmatrix}$$

^۱ ماتریس Unitay T است اگر تنها و تنها اگر :

$$T.T^T = T^T.T = I$$

^۲ دستور SUM(K) مجموع عناصر بردار K را محاسبه می کند.

سیستم جدید نشان می دهد که حالت کنترل پذیر سیستم در ۳- و حالت کنترل پذیر آن در ۲+ قرار گرفته است.

- دستورالعملهای مرتبط:

ctrb: ماتریس کنترل پذیری را محاسبه می کند.

minreal: حذف قطب و صفر انجام داده و تحقق می نیمال را نتیجه می دهد.

۲-۷- دستور diag

- هدف:

۱. قراردادن عناصر یک بردار روی قطر اصلی یک ماتریس یا روی قطرهای موازی قطر اصلی

۲. قراردادن عناصر روی قطر اصلی یک ماتریس یا قطرهای موازی با آن در یک بردار.

- نگارش دستورالعمل:

```
X=diag(v,k)
X=diag(v)
V=diag(x,k)
V=diag(x)
```

- توضیحات:

دستور $X=\text{diag}(v,k)$ که در آن V یک بردار و n عضوی و k یک عدد صحیح است، ماتریس X را تولید می کند که ابعاد آن $n+\text{abs}(k)$ است و عناصر بردار V روی K امین قطر آن قرار دارند. $K=0$ بیانگر قطر اصلی، $K>0$ مربوط به قطرهای بالای قطر اصلی و $K<0$ مربوط به قطرهای زیر آن است.

دستور $X=\text{diag}(v)$ در واقع همان دستور قبل به ازای $K=0$ است. دستور $V=\text{diag}(x,k)$ که در آن x یک ماتریس است، قطر k ام x را در بردار V قرار می دهد. وضعیت علامت K مانند دستور $X=\text{diag}(v,k)$ می باشد. $V=\text{diag}(x)$ نیز عناصر روی قطر اصلی ماتریس x را در بردار k قرار می دهد.

- مثال: بردار V را به صورت زیر در نظر می گیریم:

```
» V=[-1 -2 -3];
» X=diag(V)
X=
    -1     0     0
     0    -2     0
     0     0    -3
» X=diag(v,1)
X=
     0    -1     0     0
     0     0    -2     0
     0     0     0    -3
     0     0     0     0
» Y=[1 2 3;-3 -2 -1;-5 -6 -8];
» V1=diag(Y)
V1=
     1
    -2
    -5
    -8
» V2=diag(Y,1)
V2=
     2
    -1
```

۸-۲- eig - دستور :

- هدف: یافتن مقادیر و بردارهای ویژه
- نگارش دستورالعمل:

```
d=eig(A)
d=eig(A,B)
[V,D]=eig(A)
[V,D]=eig(A,'no balance')
[V,D]=eig(A,B)
[V,D]=eig(A,B,flag)
```

- توضیحات : در آغاز به مرور پاره ای تعاریف مقدماتی می پردازیم. مساله مقادیر ویژه در واقع حل معادله زیر است:

$$AX = \lambda X$$

که در آن A یک ماتریس $n \times n$ ، X یک بردار n عضوی و λ یک اسکالر است. مقادیری از λ که معادله فوق را برآورده می کنند ، مقادیر ویژه ماتریس A هستند. تعداد مقادیر ویژه برابر با تعداد ابعاد ماتریس A (یعنی n) است. بردار X نیز به نام بردار ویژه سمت راست ماتریس A شناخته می شود. در نرم افزار Matlab دستور eig برای محاسبه مقادیر و بردارهای ویژه در نظر گرفته شده است.

در مساله مقادیر ویژه تعمیم یافته ، هدف یافتن مقادیری از λ و x است که معادله زیر را برآورده سازند:

$$AX = \lambda BX$$

که در آن A و B هر دو ماتریسهای $n \times n$ و معلوم هستند. اگر B ویژه نباشد آنگاه معادله فوق را می توان به فرم زیر تبدیل کرد:

$$B^{-1}AX = \lambda X$$

که در واقع همان یافتن مقادیر و بردارهای ویژه معمولی برای ماتریس $B^{-1}A$ می باشد . اما اگر B ویژه باشد استفاده از الگوریتم QZ ضروری است.

دستور $d=eig(A)$ مقادیر ویژه ماتریس A را بر می گرداند.

دستور $d=eig(A,B)$ که در آن A و B دو ماتریس مربعی هستند، مقادیر ویژه تعمیم یافته را محاسبه می کند.

$[V,D]=eig(A)$ مقادیر ویژه ماتریس A را در روی ماتریس قطری D و بردارهای ویژه آن را در ماتریس V قرار می دهد ، بنابراین خواهیم داشت: $A*V=V*D$ که V ماتریس بردارهای ویژه راست A است. ماتریس W ، ماتریس بردارهای چپ A نامیده می شود اگر $W^T*A=D*W^T$. برای محاسبه W دستورات زیر لازم است:

```
» [W,D]=eig(A. ');
» W=conj(W);
```

دستور $[V,D]=eig(a, 'no balance')$ مقادیر و بردارهای ویژه A را بدون انجام فرآیند متعادل سازی¹ اولیه محاسبه می کند. به طور معمول، انجام عمل متعادل سازی شرایط ماتریس ورودی را بهبود بخشیده، منجر به محاسبه صحیح تر مقادیر و بردارهای ویژه می شود. اما چنانچه عناصر ماتریس به حدی کوچک باشند که امکان بروز خطای گرد کردن² وجود داشته باشد عمل متعادل سازی ممکن است آنها را به گونه ای مقیاس دهی کند که در مقایسه با سایر عناصر ماتریس، مقادیر قابل ملاحظه ای بگیرند و در نتیجه در محاسبه بردارهای ویژه خطا بوجود خواهد آمد. بنابراین بهتر است در مواردی که عنصری از ماتریس A خیلی کوچک است از حالت 'no balance' برای محاسبه مقادیر و بردارهای ویژه استفاده شود.

دستور $[V,D]=eig(A,B)$ ماتریس قطری D شامل مقادیر ویژه تعمیم یافته و ماتریس V که ستونهای آن بردارهای ویژه تعمیم یافته هستند را تولید می کند. بنابراین خواهیم داشت:

$$A*V=B*V*D$$

¹ Balancing

² Roundoff error

دستور $[V,D]=\text{eig}(A,B,\text{flag})$ الگوریتم مورد استفاده در محاسبه مقادیر و بردارهای ویژه را نیز مشخص می کند. flag می تواند یکی از موارد زیر را اختیار کند:

‘chol’: محاسبه مقادیر ویژه تعمیم یافته را با استفاده از فاکتورگیری choleky از B انجام می دهد. چنانچه A متقارن (Hermation) و B متقارن معین مثبت باشد این روش به صورت پیش فرض انتخاب خواهد شد.

‘qz’: از الگوریتم QZ بهره می گیرد و برای A و B نامتقارن نیز قابل اعمال است.

• مثال:

ماتریس زیر را در نظر می گیریم :

$$B = \begin{bmatrix} 3 & -2 & -0.9 & 2*eps \\ -2 & 4 & 1 & -eps \\ -eps/4 & eps/2 & -1 & 0 \\ -0.5 & -0.5 & 0.1 & 1 \end{bmatrix}$$

این ماتریس دارای عناصری است که در محاسبات دچار خطای گردکردن می شوند. این مثال به خوبی نشان می دهد که استفاده از حالت no balance برای محاسبه صحیح بردارهای ویژه ضروری است . با اجرای دستورات:

```
» [VB,DB]=eig(B);
» B*VB-VB*DB
```

خواهیم داشت :

ans =

```

0      -0.0000      0.0000      0
0.0000  -0.0000  -0.0000  -0.0000
-0.0000  -0.0000  -0.0000      0.0000
0      0.0000      0      1.4420
```

که برابر با ماتریس صفر نیست در حالی که با اجرای دستورات :

```
» [VN,DN]=eig(B,'nobalance');
» B*VN-VN*DN
```

ماتریس زیر بدست می آید که به صفر بسیار نزدیکتر است:

ans =

```

1.0e-015 *
0.4441  -0.2220      0.1471  -0.4996
0      0.0555  -0.3695      0.0278
-0.0172  -0.0015      0.0066      0
0      -0.2220  -0.1110      0.0555
```

۹-۲- دستور estim:

• هدف: تعیین معادلات دینامیکی رویتر با دریافت معادلات حالت سیستم و بهره رویتر.

• نگارش:

```
est=estim(sys,L)
est=estim(sys,L,sensors,known)
```

• توضیحات:

دستور $\text{est}=\text{estim}(\text{sys},L)$ با دریافت سیستم sys و بهره رویتر L ، معادلات حالت رویتر (est) را نشان می دهد. همه ورودیهای سیستم (W) تصادفی و همه خروجیهای Y قابل اندازه گیری فرض می شوند. رویتر est در فرم فضای حالت ارائه می شود. اگر سیستم اصلی (sys) دارای معادلات زیر باشد:

$$\begin{aligned} X &= Ax + Bw \\ Y &= Cx + Dw \end{aligned}$$

آنگاه دستور `estim` تخمینهای خروجی و حالت سیستم $(\hat{x}, \hat{y})$ را با معادلات زیر تولید می کند:

$$\begin{aligned} \dot{\hat{x}} &= A\hat{x} + L(y - C\hat{x}) \\ \begin{bmatrix} \hat{y} \\ \hat{x} \end{bmatrix} &= \begin{bmatrix} C \\ I \end{bmatrix} \hat{x} \end{aligned}$$

دستور `est=estim(sys,l,sensors,known)` برای سیستمهای عمومی تر به کار می رود. در اینجا فرض می شود که سیستم `sys` دارای ورودی های معلوم `u` و ورودیهای تصادفی `w` باشد. همچنین خروجی های سیستم نیز به دو گروه قابل اندازه گیری (`y`) و غیر قابل اندازه گیری (`z`) تقسیم می شوند. معادلات سیستم به صورت زیر خواهد بود:

$$\begin{aligned} \dot{x} &= Ax + B_1 w + B_2 u \\ \begin{bmatrix} z \\ y \end{bmatrix} &= \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} x + \begin{bmatrix} D_{11} \\ D_{21} \end{bmatrix} w + \begin{bmatrix} D_{12} \\ D_{22} \end{bmatrix} u \end{aligned}$$

بردارهای اندیس `sensors` و `known` به ترتیب مشخص می کند که کدامیک از خروجیها قابل اندازه گیری هستند (`y`) و کدامیک از ورودیها مقدار غیر تصادفی دارند (`u`). روینگر `est` از ورودیهای `u` و خروجیهای `y` و تخمین حالتیهای سیستم استفاده می کند.

$$\begin{aligned} \dot{\hat{x}} &= A\hat{x} + B_2 u + L(y - C_2 \hat{x} - D_{22} u) \\ \begin{bmatrix} \hat{y} \\ \hat{x} \end{bmatrix} &= \begin{bmatrix} C_2 \\ I \end{bmatrix} \hat{x} + \begin{bmatrix} D_{22} \\ 0 \end{bmatrix} u \end{aligned}$$

شایان ذکر است که طراحی بهره روینگر (`L`) با دستورات `place` یا `kalman` امکانپذیر است. از سوی دیگر به منظور تضمین صحت تخمین لازم است که دینامیکهای روینگر از دینامیکهای سیستم اصلی سریعتر باشند.

• دستورهای مرتبط:

دستور `reg` معادلات یک رگولاتور را با دریافت فیدبک حالت و بهره روینگر ارائه می دهد.

۲-۱۰- دستور `exam`:

- هدف: محاسبه e^X در حالتی که X یک ماتریس است.
- نگارش: `Y=exam(X)`
- توضیحات: در محاسبه e^X هنگامی که X یک ماتریس است دو روش عمده وجود دارد: روش اول استفاده از سری تیلور است:

$$e^X = I + X + \frac{X^2}{2!} + \frac{X^3}{3!} + \dots + \frac{X^K}{K!} + \dots$$

روش دوم اعمال تبدیل تشابهی روی X و قطرسازی آن است.

$$D = P^{-1} X P \rightarrow X = P D P^{-1} \rightarrow e^X = P e^D P^{-1}$$

به این ترتیب با دستورات زیر می توان e^X را محاسبه کرد.

```
» [V,D]=eig(X);
» exp_x=V*diag(exp(diag(D)))/V;
```

چنانچه X مقادیر ویژه تکراری نداشته باشد از روش سوم برای محاسبه e^X استفاده می شود. روش سوم محاسبه e^X از راه تقریب `Pade` است.

- **دستورات مرتبط:**

- expm1: محاسبه e^X از روش تقریب Pade
- expm2: محاسبه e^X از روش سری تیلور
- expm3: محاسبه e^X از روش مقادیر و بردارهای ویژه
- logm: لگاریتم ماتریس A را محاسبه می کند.
- sqrtm: دستور $X = \text{sqrtm}(A)$ ماتریس X را چنان بدست می آورد که $X^*X = A$
- funm: فرم ماتریسی توابع مختلف را اعمال می نماید.

۲-۱۱- دستور Gram :

- **هدف:** محاسبه گرامیانهای کنترل پذیری و رویت پذیری.
- **نگارش:**

```
Wc=gram(sys,'c')
Wo=gram(sys,'o')
```

- **توضیحات:**

دستور gram گرامیانهای کنترل پذیری و رویت پذیری سیستم را محاسبه می کند. از این گرامیانها در بررسی کنترل پذیری و رویت پذیری سیستم و نیز در کاهش مرتبه مدلها استفاده می شود. اگر معادلات فضای سیستم به صورت زیر باشد:

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

گرامیانهای کنترل پذیری و رویت پذیری به صورت زیر تعریف می شوند:

$$\begin{aligned}W_c &= \int_0^{\infty} e^{A\tau} B B^T e^{A^T \tau} d\tau \\ W_o &= \int_0^{\infty} e^{A^T \tau} C^T C e^{A\tau} d\tau\end{aligned}$$

و برای سیستمهای گسسته زمان خواهیم داشت:

$$\begin{aligned}W_c &= \sum_{k=0}^{\infty} A^k B B^T (A^T)^k \\ W_o &= \sum_{k=0}^{\infty} (A^T)^k C^T C A^k\end{aligned}$$

گرامیان کنترل پذیری مثبت معین است اگر و تنها اگر (A,B) کنترل پذیر باشد، گرامیان رویت پذیری نیز مثبت معین است اگر و تنها اگر (C,A) رویت پذیر باشد. دستور gram به سیستمهای گسسته زمان نیز قابل اعمال است. گرامیان کنترل پذیری را از حل معادله لیاپانوف زیر می توان محاسبه کرد:

$$AW_c + W_c A^T + BB^T = 0 \quad \text{در سیستمهای پیوسته}$$

$$AW_c A^T - W_c + BB^T = 0 \quad \text{در سیستمهای گسسته}$$

گرامیان رویت پذیری نیز در معادلات لیاپانوف زیر صادق است:

$$A^T W_o + W_o A + C^T C = 0 \quad \text{در سیستمهای پیوسته}$$

$$A^T W_o A - W_o + C^T C = 0 \quad \text{در سیستمهای گسسته}$$

بنابراین از دستورات lyap و dlyap نیز می توان برای محاسبه گرامیانها استفاده کرد.

- محدودیتها: ماتریس A باید پایدار باشد.
- دستورات مرتبط:
 ctrb: محاسبه ماتریس کنترل پذیری
 obsv: محاسبه ماتریس رویت پذیری
 balreal: محاسبه تحقق متعادل شده با استفاده از گرامیان ها
 lyap, dlyap: حل معادله لیاپانوف در حالت های پیوسته و گسسته زمان.

۲-۱۲- دستور initial:

- هدف: محاسبه پاسخ سیستم به شرایط اولیه (پاسخ ورودی صفر)
- نگارش:

```
initial(sys,x0)
initial(sys,x0,t)
initial(sys1,sys2,...,sysN,x0)
initial(sys1,sys2,...,sysN,x0,t)
initial(sys1,'PlotStyle1',...,sysN,'PlotStyleN',x0)
[y,x,t]=initial(sys,x0)
```

- توضیحات: دستور initial پاسخ سیستمی را که هیچ ورودی خارجی به آن اعمال نمی شود به ازای شرایط اولیه در حالت های سیستم محاسبه می کند. این تابع به مدلهای پیوسته و گسسته زمان قابل اعمال است. به جز در حالت $[y,t,x]=initial(sys,x0)$ در سایر موارد اجرای دستور initial با رسم پاسخ های شرایط اولیه سیستم همراه است.

دستور $initial(sys,x0)$ پاسخ سیستم sys (که معادلات آن به فرم فضای حالت است) را به شرایط اولیه $x0$ رسم می کند. سیستم sys می تواند پیوسته یا گسسته، SISO یا MIMO، با ورودی یا بدون ورودی باشد. بازه زمانی نمایش منحنی های شبیه سازی به طور خودکار چنان تعیین می شود که عملکرد پاسخها را در حالت گذرا کاملاً نشان دهد.

نتیجه دستور $initial(sys,x0,t)$ مانند حالت قبل است با این تفاوت که در این حالت بازه زمانی مطلوب برای محاسبه شرایط سیستم نیز مشخص شده است. می توان با انتخاب $t=T_{final}$ لحظه انتهای شبیه سازی را مشخص کرد یا t را به صورت بردار زیر تعریف نمود:

```
t=0:dt:tfinal
مثال: t=0:0.1:10;
```

در این حالت علاوه بر بازه زمانی شبیه سازی (صفر تا T_{final})، گام های شبیه سازی نیز مشخص شده است. در سیستم های گسسته، dt باید با زمان نمونه برداری هماهنگ باشد. در سیستم های پیوسته dt باید آنقدر کوچک اختیار شود تا رفتار حالت گذرا به خوبی مشخص گردد.

برای رسم همزمان پاسخ های چند سیستم LTI به یک شرایط اولیه، دستورات زیر به کار می رود:

```
initial(sys1,sys2,...,sysN,x0)
initial(sys1,sys2,...,sysN,x0,t)
```

اجرای دستورات:

```
[y,t,x]=initial(sys,x0)
[y,t,x]=initial(sys,x0,t)
```

منجر به تولید ماتریس خروجی شبیه (y)، بردار زمان های شبیه سازی (t) و ماتریس مسیرهای حالت سیستم (x) می شود. در این حالت هیچ منحنی رسم نمی شود. تعداد سطرهای ماتریس y به تعداد زمان های شبیه سازی (ابعاد t) و تعداد

ستونهای آن به تعداد خروجیهای سیستم است. ماتریس x نیز به تعداد $\text{length}(t)$ سطر و به تعداد حالتیهای سیستم ستون دارد.

- مثال: معادلات حالت سیستم را به فرم زیر در نظر می گیریم :

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -0.5572 & -0.7814 \\ 0.7814 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$y = \begin{bmatrix} 1.9691 & 6.4493 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

پاسخ سیستم با شرایط اولیه $x_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ توسط برنامه زیر قابل حصول است:

```
A = [-0.5572 -0.7814; 0.7814 0];
C = [1.9691 6.4493];
x0 = [1 ; 0.5];
t1=0:0.1:10;
t2=0:1:10;
sys = ss(A, [], C, []);

subplot(2,2,1);initial(sys,x0);
subplot(2,2,2);initial(sys,x0,t1);
subplot(2,2,3);initial(sys,x0,6.5);
subplot(2,2,4);initial(sys,x0,t2);
```

- دستورات مرتبط :

impulse: رسم پاسخ ضربه سیستم
lsim: رسم پاسخ سیستم به هر ورودی دلخواه
step: رسم پاسخ پله سیستم

۲-۱۳- دستور kalman :

- هدف: طراحی رویگر (فیلتر) کالمن برای سیستمهای پیوسته و گسسته زمان
- نگارش:

```
[kest, l, p]=kalman(sys, Qn, Rn, Nn)
[kest, l, p, m, z]=kalman(sys, Qn, Rn, Nn)
[kest, l, p]=kalman(sys, Qn, Rn, Nn, Sensors, Known)
```

(مورد آخر فقط برای سیستمهای گسسته زمان صادق است.)

• توضیحات:

دستور kalman با دریافت مدل فضای حالت سیستم و نیز کواریانس نویز سیستم و نویز اندازه گیری، رویترگر کالمن را طراحی می کند.

سیستم با معادلات حالت زیر در نظر گرفته می شود:

$$\dot{x} = Ax + Bu + Gw$$

$$y_v = Cx + Du + Hw + v$$

به ترتیب نویزهای سفید اندازه گیری و فرایند هستند و داریم v و w ورودی معین و u که در آن

$$E(w) = E(v) = 0$$

$$E(ww^T) = Q, E(vv^T) = R, E(wv^T) = N$$

هدف یافتن تخمین $\hat{x}$ از حالتی سیستم است چنانکه کواریانس خطای حالت ماندگار کمینه گردد:

$$P = \lim_{t \rightarrow \infty} (\{x - \hat{x}\} \{x - \hat{x}\}^T)$$

پاسخ بهینه، فیلتر کالمن است که معادلات آن به صورت زیر می باشد:

$$\dot{\hat{x}} = A\hat{x} + Bu + L(y_v - C\hat{x} - Du)$$

$$\begin{bmatrix} \hat{y} \\ \hat{x} \end{bmatrix} = \begin{bmatrix} C \\ I \end{bmatrix} \hat{x} + \begin{bmatrix} D \\ 0 \end{bmatrix} u$$

که در آن L بهره فیلتر بوده و از حل یک معادله ریکاتی جبری حاصل می شود.

دستور $[kest, l, p] = kalman(sys, Qn, Rn, Nn)$ مدل فضای حالت فیلتر کالمن (kest) را نتیجه می دهد. Rn, Qn و Nn کواریانس نویز هستند که در روابط بالا با Q, R و N مشخص شده اند. sys نیز مدل سیستم است که باید به صورت زیر وارد شود:

$$A, [B \ G], C, [D \ H]$$

رویترگر حاصل (kest) دارای ورودیهای $[u; y_v]$ و خروجیهای $\begin{bmatrix} \hat{y} \\ \hat{x} \end{bmatrix}$ می باشد. این دستور علاوه بر معادله رویترگر (kest)، مقدار بهره رویترگر L و نیز ماتریس کواریانس خطای ماندگار P را نیز ارائه می دهد.

• محدودیت ها:

با این فرض که:

$$\bar{Q} = GQG^T$$

$$\bar{R} = R + HN + N^T H^T + HQH^T$$

$$\bar{N} = G(QH^T + N)$$

سیستم و نویز باید شرایط زیر را ارضا کنند:

$$1. (C, A) \text{ آشکاری پذیر باشد (detectable)}$$

$$2. R > 0 \text{ و } \bar{Q} - \bar{N}\bar{R}^{-1}\bar{N}^T \geq 0$$

$$3. (A - \bar{N}\bar{R}^{-1}C, \bar{Q} - \bar{N}\bar{R}^{-1}\bar{N}^T) \text{ هیچ حالت کنترل ناپذیری روی محور موهومی نداشته باشند.}$$

- **دستورات مرتبط:**

care: حل معادله ریکاتی جبری پیوسته

estim: تهیه معادلات رویتگر از روی معادلات سیستم و بهره رویتگر

lqr: طراحی یک رگولاتور بیدبک حالت LQ

۲-۱۴- دستور Lyap :

هدف: حل معادله لیاپانوف پیوسته - زمان

- **نگارش:**

X=lyap(A,Q)
X=lyap(A,B,C)

- **توضیحات:** دستور X=Lyap(A, Q) معادله لیاپانوف زیر را حل می کند:

$$AX+XA^T+Q=0$$

که در آن A و Q ماتریسهای مربعی هستند. ماتریس X متقارن است اگر Q متقارن باشد.

دستور X=lyap(A, B, C) معادله لیاپانوف تعمیم یافته را حل می کند.

$$AX+XB+C=0$$

که در آن A و B و C لزوماً مربعی نیستند ولی ابعاد آنها سازگار است.

- **محدودیت ها:**

اگر $\alpha_1, \alpha_2, \dots, \alpha_n$ مقادیر ویژه A و $\beta_1, \beta_2, \dots, \beta_n$ مقادیر ویژه B باشند، آنگاه معادله لیاپانوف پیوسته پاسخ منحصر بفرد دارد اگر برای هر زوج دلخواه (i,j) داشته باشیم:

$$\alpha_i + \beta_j \neq 0$$

چنانچه این شرایط برقرار نباشد دستور lyap تولید پیغام خطا می کند:

Solution does not exits or is not unique

- **دستورات مرتبط:**

دستور dlyap معادله لیاپانوف گسسته را حل می کند.

۲-۱۵- دستور null :

هدف: تعیین فضای تهی یک ماتریس

- **نگارش:** B=null(A)

- **توضیحات:** دستور B=null(A) یک پایه متعامد یکه برای فضای تهی ماتریس A ارائه می کند.

واضح است که: $B^T B = I$ و تعداد ستونهای B، ابعاد فضای تهی^۱ A را می دهد.

» A=[1 2 3;1 2 3;1 2 3]

A =

1	2	3
1	2	3
1	2	3

» null(A)

ans =

¹ Nullity

```

-0.1690    -0.9487
 0.8452         0
-0.5071    0.3162
» null(A, 'r')

```

```
ans =
```

```

-2    -3
 1     0
 0     1

```

۲-۱۶- دستور obsv:

- هدف: محاسبه ماتریس رویت پذیری
- نگارش:

```
Ob=obsv(A,C)
Ob=obsv(sys)
```

- توضیحات: دستور obsv ماتریس رویت پذیری یک سیستم با نمایش فضای حالت را محاسبه می کند. اگر A یک ماتریس n*n و C یک ماتریس p*n باشد، دستور obsv(A,C) ماتریس رویت پذیری زیر را نتیجه خواهد داد:

$$Ob = \begin{bmatrix} C & CA & CA^2 & \dots & CA^{n-1} \end{bmatrix}^T$$

واضح است که با دستور $\text{rank}[\text{obsv}(A, C)]$ می توان رویت پذیری کامل سیستم را بررسی کرد. اگر sys مدل فضای حالت یک سیستم باشد، دستور $\text{obsv}(\text{sys})$ نیز ماتریس رویت پذیری سیستم را محاسبه خواهد کرد.

این دستور با دستور زیر معادل است:

```
Ob=obsv(sys.a, sys.b)
```

- مثال: سیستم زیر مفروض است:

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

که در آن :

$$A = \begin{bmatrix} 1 & 1 \\ 4 & -2 \end{bmatrix}, B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

تکه برنامه زیر نشان می دهد که این سیستم رویت پذیر کامل است.

```

» A=[1 1;4 -2];
» B=eye(2);
» C=[1 0];
» D=0;
» rank(obsv(A,C))

```

```
ans =
```

```
2
```

- **دستورهای مرتبط:**

obsvf: حالت‌های رویت پذیر و رویت ناپذیر سیستم را از هم جدا می کند.

۲-۱۷- دستور obsvf :

- **هدف:** جدا کردن حالت‌های رویت ناپذیر و رویت پذیر سیستم

- **نگارش:**

$$[Abar, Bbar, Cbar, T, k] = \text{obsvf}(A, B, C)$$

$$[Abar, Bbar, Cbar, T, k] = \text{obsvf}(A, B, C, tol)$$

- **توضیحات:** معادلات حالت سیستم را به صورت زیر در نظر می گیریم:

$$\dot{x} = A_{n \times n} x + Bu$$

$$y = C_{p \times n} x + Du$$

اگر رتبه ماتریس رویت‌پذیری سیستم r باشد (و $r < n$) آنگاه سیستم رویت پذیر کامل نیست. در این حالت، تبدیل تشابهی T وجود دارد که unitary است و حالت‌های رویت‌پذیر و رویت‌ناپذیر سیستم را از هم جدا می کند.

$$\bar{A} = TAT^T, \bar{B} = TB, \bar{C} = CT^T$$

با اعمال این تبدیل، حالت‌های رویت‌ناپذیر سیستم در گوشه بالا-چپ ماتریس A قرار می گیرند:

$$\bar{A} = \begin{bmatrix} A_{no} & A_{12} \\ 0 & A_o \end{bmatrix}, \bar{B} = \begin{bmatrix} B_{no} \\ B_o \end{bmatrix}, \bar{C} = [0 \quad C_o]$$

به این ترتیب زوج $[Abar, Bbar, Cbar, k] = \text{obsvf}(A, B, C)$ رویت‌پذیر کامل است و مقادیر ویژه A_{no} حالت‌های رویت ناپذیر سیستم هستند.

دستور $[Abar, Bbar, Cbar, T, k] = \text{obsvf}(A, B, C)$ ماتریس‌های A و B و C را به نحوی که قبلاً گفته شد تولید می کند. T نیز تبدیل تشابهی برای جدا کردن حالت‌های کنترل‌پذیر و کنترل‌ناپذیر است. k یک بردار n عضوی است (ابعاد A است) و هر عضو آن نشان می دهد که در هر مرتبه تکرار برای محاسبه ماتریس نگاشت T ، چند حالت رویت‌پذیر جدا شده است.

در نتیجه $\text{sum}(k)$ تعداد حالت‌های A_o را نشان خواهد داد. با دستور $\text{obsvf}(A, B, C, tol)$ می توان تolerانس انجام محاسبات و تعیین فضا‌های رویت پذیر و رویت ناپذیر را نیز تعیین کرد.

- **دستورات مرتبط:**

obsv : محاسبه ماتریس رویت پذیری

۲-۱۸- دستور place :

- **هدف:** طراحی جایاب قطب برای سیستم های چند ورودی

- **نگارش:**

$$K = \text{place}(A, B, P)$$

$$[K, prec, message] = \text{place}(A, B, P)$$

- **توضیحات:**

سیستم چند ورودی (یا تک ورودی) زیر مفروض است :

$$\dot{x} = Ax + Bu$$

فرض می شود که بردار P حاوی محل قطبهای مطلوب برای سیستم حلقه بسته باشد.^۱ دستور $K=place(A,B,P)$ ماتریس فیدبک حالت K را چنان طراحی می کند که قانون $u=-Kx$ قطبهای سیستم حلقه بسته را در محلهای مشخص شده در بردار P قرار دهد. واضح است که در سیستمهای چند ورودی ماتریس K منحصر بفرد نیست. دستور $place$ از الگوریتمهایی استفاده می کند که حساسیت سیستم حلقه بسته را در برابر اغتشاشات در A و B کاهش می دهد.^۲ دستور $[K, prec, message]=place(A,B,P)$ علاوه بر تعیین K ، $prec$ را بر می گرداند که نشان می دهد که قطبهای سیستم حلقه بسته تا چه اندازه به محلهای تعیین شده در P نزدیک هستند. $prec$ تعداد قطبهایی را که دقیقاً در محلهای تعیین شده در P قرار گرفته اند نشان می دهد. بعلاوه اگر یکی از قطبهای سیستم حلقه بسته بیش از ۱۰٪ از محلهای مطلوب دور باشد، یک پیغام هشدار در $message$ صادر میشود. با استفاده از دستور $place$ می توان ماتریس بهره رویتگر را به صورت زیر محاسبه کرد:

$$L=place(A',C',P)'$$

• دستورات مرتبط:

$acker$: جایابی قطب با استفاده از فرمول آکرمن برای سیستمهای تک ورودی

۲-۱۹- دستور ss:

- هدف: تعریف مدل فضای حالت یا تبدیل یک مدل LTI به فضای حالت
- نگارش:

```
sys=ss(A,B,C,D);
sys=ss(d);
sys=ss(A,B,C,D,ltisys);
sys=ss(A,B,C,D,'property1',value1,...,'propertyN',valueN);
sys_ss=ss(sys)
sys_ss=ss(sys,'minimal')
```

- توضیحات: دستور ss برای نمایش سیستم به فرم فضای حالت یا تبدیل یک مدل تابع تبدیل یا مدل صفر-قطب-بهره به مدل فضای حالت به کار می رود.

دستور $SYS=ss(A,B,C,D)$ مدل فضای حالت برای سیستم زمان پیوسته زیر تولید می کند:

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

اگر $D=0$ باشد کافیت مقدار آن به صورت اسکالر صفر (بدون توجه به ابعاد آن) وارد شود.

دستور $sys=ss(d)$ سیستم با ماتریس بهره ثابت d را نمایش می دهد و معادل است با:

```
sys=ss([],[],[],d);
```

دستور $sys=ss(A,B,C,D,ltisys)$ یک مدل فضای حالت با توجه به ویژگی مشخص شده در $ltisys$ تولید می کند. این ویژگی می تواند یکی از موارد ذکر شده در جدول زیر باشد:

نام ویژگی	معرفی ویژگی	نوع ویژگی
-----------	-------------	-----------

^۱ قطبهای سیستم حلقه بسته همان مقادیر ویژه $A-BK$ هستند.

^۲ این الگوریتمها از مرجع زیر اقتباس شده اند:

Kautsky, J. and N.K. Nichols, "Robust Pole Assignment in Linear State Feedback", Int.J.Control, 41 (1985), pp.1129-1155

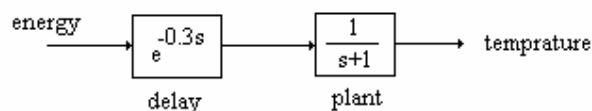
IoDelayMatrix	تأخیرهای ورودی/خروجی	ماتریس
InputDelay	تأخیر در ورودی	بردار
InputGroup	گروه کانالهای ورودی	Cell array
InputName	اسامی کانالهای ورودی	Cell vector of string
Notes	یادداشتها برای معرفی مدل	Text
OutputDelay	تأخیرهای خروجی	بردار
OutputGroup	گروه کانالهای خروجی	Cell array
OutputName	اسامی کانالهای خروجی	Cell vector of strings
T_s	زمان نمونه برداری	اسکالر
UserData	اطلاعات اضافی	دلخواه

هر یک از این ویژگیها به اختصار توضیح داده می شود:

۱. زمان نمونه برداری T_s (برحسب ثانیه) برای تعریف مدل فضای حالت سیستمهای زمان-گسسته به کار می رود. $T_s=0$ سیستم زمان-پیوسته و $T_s=-1$ معرف سیستم زمان گسسته با زمان نمونه برداری نامعلوم است.
 ۲. ویژگی `InputDelay`، `OutputDelay` و `IoDelayMatrix` امکان وارد کردن تاخیر در مدلهای فضای حالت را ایجاد می کند و مقادیر پیش فرض این ویژگیها صفر است (بدون تاخیر)
 ۳. ویژگی های `InputName` و `OutputName` امکان نامگذاری تک تک ورودیها و خروجیها را به کاربر می دهد.
 ۴. `InputGroup` و `OutputGroup` امکان گروه بندی ورودیها و خروجیهای سیستم را به برنامه نویس می دهد. به عنوان مثال در یک سیستم ۵ ورودی، برنامه نویس می تواند ۳ ورودی را در گروه `control` و ۲ ورودی را در گروه `noise` قرار دهد.
 ۵. ویژگیهای `Notes` و `User data` نیز برای تعریف موارد اضافی توسط برنامه نویس در نظر گرفته شده اند. `Notes` تنها یک `text` بوده و معمولاً حاوی توضیحاتی درباره سیستم است. در ویژگی `User data` برنامه نویس می تواند داده های مورد نظر خود را تعریف کند.
- در مثالهایی که در انتهای این دستورات عملهای ذکر شده، نحوه مقدار مقدار دهی به این ویژگیها مشخص خواهد شد.
- دستور `sys_ss=ss(sys)` مدل `sys`، تولید شده توسط دستورات `tf` و `zpk` را به مدل `sys_ss` در فضای حالت تبدیل می کند. به این عملیات تحقق فضای حالت گفته می شود.
- دستور `sys_ss=ss(sys, 'minimal')` یک تحقق فضای حالت از سیستم `sys` را چنان ارائه می دهد که در آن هیچ حالت رویت ناپذیر یا کنترل ناپذیری موجود نباشد.
- این دستور با دستور زیر معادل است:

```
sys_ss=mineral(ss(sys))
```

مثال ۱: سیستم تک ورودی/تک خروجی تاخیر دار زیر را در نظر می گیریم:



دستورات زیر `plant` را در فضای حالت معرفی می کند :

```
» sys=tf(1,[1 1]);
» sys_ss=ss(sys);
```

برای افزودن تاخیر به `plant`، ویژگی `Input Delay` را مقدار دهی می کنیم. اینکار با دستور `set` انجام می شود:

```
» set(sys_ss, 'InputDelay', 0.3);
```

برای نامگذاری ورودی و خروجی سیستم و نیز افزودن یک یادداشت به آن دستور زیر را به کار می بریم :

```
» set(sys_ss, 'InputName', 'energy', 'Outputname', 'temprature', 'Notes', 'A  
Sample Heater Mode')
```

به این ترتیب sys_ss در واقع مدل فضای حالت برای سیستم تاخیر دار است و رسم پاسخ پله آن به خوبی ۰/۳ ثانیه زمان مرده را در ابتدای پاسخ مشخص خواهد کرد.

چنانچه در هر لحظه برنامه نویس بخواهد از مقدار فعلی یکی از ویژگیهای تنظیم شده سیستم آگاه شود دستور get مورد استفاده قرار می گیرد. به عنوان مثال در سیستم قبل :

```
» get(sys_ss, 'InputDelay')  
ans=  
0.3000
```

این دسترسی به صورت زیر نیز امکانپذیر است:

```
» sys_ss.InputDelay  
ans=  
0.3000
```

مثال ۲ :

برنامه زیر علاوه بر قراردادن مدل فضای حالت سیستم در متغیر system، حالتها و خروجیهای سیستم را نیز نامگذاری می کند.

```
A=[0 1; -1 -2];  
B=[1;1];  
C=[1 0; 0 2];  
D=0;  
system=ss(A,B,C,D, 'statename', {'position' 'velocity'}, ...  
'Outputname', {'out1'; 'out2'});
```

۲-۲۰- دستور ssdata :

- هدف : دسترسی به ماتریسهای A ، B ، C و D فضای حالت از روی مدل سیستم
- نگارش : $[A, B, C, D] = \text{ssdata}(\text{SYS})$ سیستمهای زمان پیوسته
- $[A, B, C, D, Ts] = \text{ssdata}(\text{SYS})$ سیستمهای زمان گسسته

توضیحات :

چنانچه مدل سیستم به فرم تابع تبدیل یا قطب / صفر/بهره باشد ، این دستور ابتدا آن را به مدل فضای حالت تبدیل مرده ، سپس A و B و C و D را استخراج می کند.

• دستورهای مرتبط :

SS : تعریف مدل فضای حالت

۲-۲۱- دستور ss2ss :

- هدف: تبدیل محورهای حالت^۱ در نمایش فضای حالت (اعمال تبدیل تشابهی)

¹ State Coordinates

• نگارش: $SYST = ss2ss(SYS, T)$

• توضیحات: سیستم SYS با نمایش فضای حالت زیر مفروض است:

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

دستور ss2ss تبدیل تشابهی $\bar{x} = Tx$ را روی بردار x اعمال می کند و سیستم SYST با مدل فضای حالت زیر را نتیجه می دهد.

$$\dot{\bar{x}} = TAT^{-1}\bar{x} + TBu$$

$$y = CT^{-1}\bar{x} + Du$$

دستور $SYST = ss2ss(SYS, T)$ با دریافت سیستم SYS و تبدیل تشابهی T سیستم معادل SYST را محاسبه می کند. ماتریس T باید وارون پذیر باشد.

• دستورهای مرتبط:

canon: تحقق کانونیکال در فضای حالت

۲-۲۲- دستور ss2tf :

• هدف: تبدیل مدل فضای حالت سیستم به تابع تبدیل

• نگارش: $[Num, Den] = ss2tf(A, B, C, D, iu)$

• توضیحات: تابع تبدیل سیستم

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

را نسبت به iu امین ورودی محاسبه می کند. محاسبه تابع تبدیل طبق رابطه زیر است:

$$H(s) = \frac{Num(s)}{Den(s)} = C(sI - A)^{-1}B + D$$

بردار Den حاوی ضرایب مخرج تابع تبدیل بر حسب توانهای نزولی s است. تعداد سطرهای ماتریس Num برابر با تعداد خروجیهای سیستم است و هر سطر به صورت تابع تبدیل خروجی متناظر است.

• دستورهای مرتبط:

tf2ss: تبدیل تابع تبدیل سیستم به مدل فضای حالت

ss2ss: انجام تبدیل تشابهی روی سیستم

۲-۲۳- دستور tf2ss :

• هدف: بدست آوردن مدل فضای حالت از روی تابع تبدیل سیستم

• نگارش: $[A, B, C, D] = tf2ss(Num, Den)$

• توضیحات:

تابع تبدیل سیستم را به صورت مدل فضای حالت زیر تبدیل می کند.

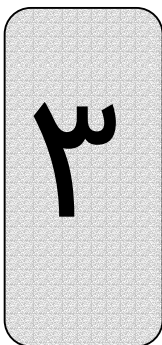
$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

مدل فضای حالت به فرم کانونیکال کنترلر می باشد.

• دستوره‌ای مرتبط:

ss2tf : تبدیل مدل فضای حالت سیستم به تابع تبدیل



بررسی برنامه های مربوط به مثالهای
متن کتاب

۳-۱- رسم پاسخهای سیستم غیرخطی (مثال ۵-۳)

در مثال ۳-۵ برای مدل خطی شده آونگ معکوس، فیدبک حالت

$ke = [-1.91 \quad -4.262 \quad -45.49 \quad -15.64]$ طراحی گردید تا قطبهای سیستم حلقه بسته در $[-1, -2, -2.5, -3]$ قرار گیرند. اما برای بررسی عملکرد این فیدبک حالت باید آن را بر روی سیستم غیرخطی آزمود. به این منظور نیازمند حل معادلات دیفرانسیل غیرخطی هستیم. اینکار در نرم افزار Matlab معمولاً با دستورهایی^۱ ODE انجام می‌گیرد. اما در این مثال به تفصیل، شیوه حل این نوع معادلات بررسی شده است. دو فایل به نامهای EXP5_3 و ode به صورت زیر برای حل این مثال در نظر گرفته شده‌اند.

```
%-- Chapter 5, Example 3

clc
clear all;

%----- PARAMETERS DEFINITION -----
m=.1;      % Kg - mass of pendulum
M=1;      % Kg - mass of cart
I=.06;     % Kg.m^2 - moment of inertia
g=9.81;    % m/sec^2
L=.35;     % meter (length of pendulum is 2*L)

m1_bar=m*L/(m+M);
m2_bar=m*L/(I+m*L^2);

%----- Simulation Time=5 sec
t=0.01:.01:5;
[rt,ct]=size(t);
par=[m,M,I,g,L,m1_bar,m2_bar];

%----- 1. teta(0)=5 deg -----
teta_0=5*(pi/180);      % teta(0) in radian
x(:,1)=[0;0;teta_0;0];  % initial state
teta_5=ode(par,ct,x(:,1)); % calling user-defined
function: ode
plot(t,teta_5,'b-');

%----- 2. teta(0)=20 deg -----
teta_0=20*(pi/180);      % teta(0) in radian
x(:,1)=[0;0;teta_0;0];  % initial state
teta_5=ode(par,ct,x(:,1)); % calling user-defined
function: ode
hold on;plot(t,teta_5,'b--');
```

شکل ض ۱-۱- فایل EXP5_3

^۱ ode23, ode45, ode113, ode15s, ode23s, ode23t, ode23tb

```
%~~~~~ 4. teta(0)=33.4 deg ~~~~~~
teta_0=33.4*(pi/180);      % teta(0) in radian
x(:,1)=[0;0;teta_0;0];    % initial state
teta_5=ode(par,ct,x(:,1)); % calling user-defined
function: ode
hold on;plot(t,teta_5,'b:');

legend('initial angle= 5 deg','initial angle= 20
deg','initial angle= 33.4 deg',3);
```

شکل ض ۱-۱- (ادامه) فایل EXP5_3.m

```
function [teta]=ode(par_in,ct,x);

x(:,1)=x;    % initial state

%~~~~~ DEFINITION OF PARAMETERS ~~~~~

m=par_in(1);
M=par_in(2);
I=par_in(3);
g=par_in(4);
L=par_in(5);
m1_bar=par_in(6);
m2_bar=par_in(7);

mb=m1_bar*m2_bar;    %parameter definition only
for convenience!

for k=2:ct

%~~~~~ STATE FEEDBACK ~~~~~
ke=[-1.91 -4.262 -45.49 -15.64];
u=-ke*x(:,k-1);

%~~~~~ NONLINEAR DYNAMIC EQUATION ~~~~~
x1_dot=x(2,k-1);
x2_dot1=-mb*g*cos(x(3,k-1))*sin(x(3,k-1))/(1-
mb*cos(x(3,k-1))^2);
x2_dot2=(m1_bar*sin(x(3,k-1))*x(4,k-1)^2)/(1-
mb*cos(x(3,k-1))^2);
x2_dot3=u/((m+M)*(1-mb*cos(x(3,k-1))^2));
x2_dot=x2_dot1+x2_dot2+x2_dot3;

x3_dot=x(4,k-1);

x4_dot1=g*m2_bar*sin(x(3,k-1))/(1-mb*cos(x(3,k-1))^2);
x4_dot2=-(mb*sin(x(3,k-1))*x(4,k-1)^2)/(1-mb*cos(x(3,k-1))^2);
x4_dot3=-m2_bar*cos(x(3,k-1))*u/((m+M)*(1-mb*cos(x(3,k-1))^2));
```

شکل ض ۱-۲- فایل ode.m

```

x4_dot=x4_dot1+x4_dot2+x4_dot3;

x(1,k)=x(1,k-1)+0.01*x1_dot;
x(2,k)=x(2,k-1)+0.01*x2_dot;
x(3,k)=x(3,k-1)+0.01*x3_dot;
x(4,k)=x(4,k-1)+0.01*x4_dot;
end

teta=x(3,:)*180/pi; % changing x3 from radian to
                    degree

```

شکل ض ۱-۲- (ادامه)

دو فایل EXP5_3.m و ode.m باید در یک زیرشاخه ذخیره شوند. فایل اصلی برنامه EXP5_3.m است و فایل ode.m از درون آن فراخوانی می‌شود. در فایل ode.m پس از مقداردهی اولیه پارامترها، معادلات حالت غیرخطی سیستم در یک حلقه For به صورت گام به گام (با طول گام ۰/۰۱) حل می‌شوند و نتیجه نهایی که تغییرات زاویه θ است برحسب درجه در متغیر teta قرار می‌گیرد و به فایل EXP5_3.m فرستاده شده، در آنجا رسم می‌گردد. این عملیات برای سه مقدار اولیه متفاوت برای θ^1 انجام گرفته است و نتیجه در شکل ۵-۵ رسم گردیده است.

۲-۳- طراحی فیدبک حالت با دستور acker (مثال ۶-۸)

در مثال ۶-۷ برای سیستم ناپایدار هواپیما کنترل کننده فیدبک حالت به نحوی طراحی گردید که قطبهای سیستم حلقه بسته در $[-1.8-0.6i \ -1.8+0.6i \ -0.9-1.9i \ -0.9+1.9i]$ قرار گرفته و سیستم پایدار شود. برنامه مورد استفاده در این طراحی به صورت زیر است:

```

clear all

%----- PARAMETERS DEFINITION -----
a11=-0.0507;
a12=-3.861;
a21=-0.00117;
a22=-0.5164;
a31=-0.000129;
a32=1.4168;
a33=-0.4932;
b1=0;
b2=-0.0717;
b3=-1.645;
g=9.81;

```

شکل ض ۱-۳- برنامه EXP6_8.m

^۱ $\theta = 5^\circ, \theta = 20^\circ, \theta = 33.4^\circ$

```

%----- OPEN-LOOP SYSTEM:  $\dot{X}=AX+BU$  -----
A=[a11 a12 0 -g;a21 a22 1 0;a31 a31 a33 0;0 1 0 0];
B=[b1;b2;b3;0];

%----- OPEN-LOOP ANALYSIS -----
openloop_eigenvalues=eig(A)
controlability_rank=rank(ctrb(A,B))

%----- STATE FEEDBACK DESIGN -----

P=[-0.9+1.9i -0.9-1.9i -1.8+0.6i -1.8-.6i];
                                % Desired Closed-Loop Poles
K=acker(A,B,P);                % State-Feedback Gain

%-----
%----- SIMULATION OF REGULATION PROBLEM -----
%-----

t=.01:.01:10;                  % simulation time
[rt,ct]=size(t);

x(:,1)=[0;1;0;0]; % initial condition

for k=2:ct
    u(k)=-K*x(:,k-1);
    x(:,k)=x(:,k-1)+.01*(A*x(:,k-1)+B*u(k));
end

%----- PLOTTING STATES -----
subplot(2,2,1);plot(t,x(1,:));
title('Regulation Problem, state:X_1');
subplot(2,2,2);plot(t,x(2,:));
title('Regulation Problem, state:X_2');
subplot(2,2,3);plot(t,x(3,:));
title('Regulation Problem, state:X_3');
subplot(2,2,4);plot(t,x(4,:));
title('Regulation Problem, state:X_4');

%----- PLOTTING CONTROL SIGNAL -----

figure;plot(t,u);
title('Regulation Problem, Control Signal (u)');

%*****

%-----
%----- SIMULATION OF TRACKING PROBLEM -----
%-----

C=[1 0 0 0]; % selecting 'teta' as system output
D=0;

```

شکل ض ۱-۳- (ادامه)

```

CL_sys=ss(A-B*K,B,C-D*K,D);    % Closed-Loop System
CL_gain=dcgain(CL_sys);
                                % DC-Gain of closed-loop system

K_r=inv(CL_gain);    % control signal: u=-K*x+K_r*yd
yd=1;                % desired value for output (teta)

%--- SIMULATION ---

x(:,1)=[0;1;0;0];    % initial condition

for k=2:ct
    u(k)=-K*x(:,k-1)+K_r*yd;
    x(:,k)=x(:,k-1)+.01*(A*x(:,k-1)+B*u(k));
end

%----- PLOTTING STATES -----
figure;subplot(2,2,1);plot(t,x(1,:));
title('Tracking Problem, state X_1');
subplot(2,2,2);plot(t,x(2,:));
title('Tracking Problem, state:X_2');
subplot(2,2,3);plot(t,x(3,:));
title('Tracking Problem, state:X_3');
subplot(2,2,4);plot(t,x(4,:));
title('Tracking Problem, both:X_4 and Output');

%----- PLOTTING CONTROL SIGNAL -----
figure;plot(t,u);
title('Tracking Problem, Control Signal');

```

شکل ض ۱-۳- (ادامه)

نتایج اجرای دو دستورالعمل بخش OPEN-LOOP ANALYSIS برنامه فوق نشان می‌دهد که سیستم حلقه باز دارای یک قطب ناپایدار در 0.1255 بوده و کنترل‌پذیر کامل حالت می‌باشد:

openloop_eigenvalues =

```

0.1255
-0.3502
-0.2878
-0.5478

```

controlability_rank =

4

در بخش STATE FEEDBACK DESIGN با استفاده از دستور acker، فیدبک حالت طراحی شده است. شبیه‌سازها در دو بخش رگولاسیون و ردیابی صورت گرفته است. در هر دو بخش معادلات حالت سیستم با استفاده از

یک حلقه For حل شده‌اند. البته به جای این کار، روشهای بهتری نیز وجود دارد که در مثالهای بعد بررسی خواهد شد. شکل‌های ۶-۷ و ۶-۸ نتایج اجرای برنامه فوق هستند.

۳-۳- فیدبک حالت با کنترل انتگرال (مثال ۹-۶)

در بخش ۲-۴-۶ نحوه افزودن کنترل انتگرال به منظور ردیابی بدون خطای ورودیهای پله مورد بررسی قرار گرفت. برنامه زیر مربوط به مثال ۹-۶ می باشد.

```
clc
clear all
%----- PARAMETERS DEFINITION -----
a11=-0.0507;
a12=-3.861;
a21=-0.00117;
a22=-0.5164;
a31=-0.000129;
a32=1.4168;
a33=-0.4932;
b1=0;
b2=-0.0717;
b3=-1.645;
g=9.81;

%----- OPEN-LOOP SYSTEM: X_dot=AX+BU, Y=CX ----
A=[a11 a12 0 -g;a21 a22 1 0;a31 a31 a33 0;0 1 0 0];
B=[b1;b2;b3;0];
C=[1 0 0 0];

%----- INTEGRAL CONTROLABILITY ANALYSIS -----
[rA,cA]=size(A);
[rB,cB]=size(B);
[rC,cC]=size(C);

A_bar=[A zeros(rA,1);-C zeros(rC,1)];
B_bar=[B;zeros(rC,cB)];
Integral_controlability_rank=rank(ctrb(A_bar,B_bar))

%----- STATE FEEDBACK DESIGN -----
P=[-1.6 -2 -0.9 -2.5 -1]; % Desired Closed-Loop Ploes
K=acker(A_bar,B_bar,P); % State-Feedback Gain
K1=K(1:rA);
K2=K(rA+1:rA+rC);

%-----
%----- SIMULATION OF REGULATION PROBLEM -----
%-----

t=.01:.01:10; % simulation time
[rt,ct]=size(t);

yd=1; % Desired output (desired value for 'teta')
w=[1;0;0;0]; %disturbance

x(:,1)=[0;0;0;0;0]; % initial condition of augmented
system
```

شکل ض ۱-۴- فیدبک حالت با کنترل انتگرال (مثال ۹-۶)

```

for k=2:ct
    u(k)=[-K1 -K2]*x(:,k-1);
    x(:,k)=x(:,k-1)+.01*(A_bar*x(:,k-1)+B_bar*u(k)+[zeros(rA,1);1]*yd+[w;0]);
end

%----- PLOTTING STATES -----
subplot(2,2,1);plot(t,x(1,:));
title('Tracking with Disturbance, state: X_1');
subplot(2,2,2);plot(t,x(2,:));
title('Tracking with Disturbance, state: X_2');
subplot(2,2,3);plot(t,x(3,:));
title('Tracking with Disturbance, state: X_3');
subplot(2,2,4);plot(t,x(4,:));
title('Tracking with Disturbance, state: X_4');

%----- PLOTTING CONTROL SIGNAL -----
figure;plot(t,u);
title('Tracking with Disturbance, Control Signal');

%----- PLOTTING OUTPUT SIGNAL -----
figure;plot(t,x(1,:));
title('Tracking with Disturbance, Output Signal');

```

شکل ض ۱-۴- (ادامه)

ابتدا با استفاده از رابطه ۴-۴-۴۵ ماتریسهای $\bar{A}$ و $\bar{B}$ معرفی شده و کنترل پذیری انتگرال سیستم به کمک دستور `ctrb` بررسی گردیده است. سپس از دستور `acker` برای طراحی فیدبک حالت استفاده شده، ماتریسهای K_1 و K_2 طبق رابطه ۴-۴-۴۷ جداسازی شده‌اند.

۴-۳- اثر تغییر فیدبک حالت بر رفتار دینامیکی سیستم (مثال ۶-۱۰)

در مثال ۱۰-۶ گفته شد که فیدبک حالت در سیستمهای چندمتغیره به منظور قرار دادن قطبها در نقاط معین، منحصریفر نیست و تغییر آن رفتار دینامیکی سیستم را تحت تأثیر قرار می‌دهد. برنامه زیر مربوط به این مثال بوده و حاصل اجرای آن شکل ۱۱-۶ می‌باشد.

```

clc
clear all

%---- DEFINITION OF MIMO SYSTEM ----

A=[0 1 0;0 -1.799 6.953;0 0.939 -1.660];
B=[0 0;-28.97 -6.453;-0.174 -0.230];

[rA,cA]=size(A);
[rB,cB]=size(B);

```

شکل ض ۱-۵- اثر تغییر فیدبک حالت بر رفتار دینامیکی سیستم (مثال ۶-۱۰)

```

%---- DESIRED POLES OF CLOSED-LOOP SYSTEM ----

P=[-2 -2.5 -1];
[rP,cP]=size(P);

N1=null([A-P(1,1)*eye(rA) B]);
N2=null([A-P(1,2)*eye(rA) B]);
N3=null([A-P(1,3)*eye(rA) B]);

%- SELECTING DESIRED PERFORMANCE BY SELECTING VECTORS --
%---- First selection: K1 ----
vec=[N1(:,1) N2(:,1) N3(:,2)];% SLOW MODES CANCELATION
Q=vec(4:5,:);
V=vec(1:3,:);

K=Q*inv(V); % STATE FEEDBACK GAIN
sim('exp610',[0 8]); % system simulation
X1=states;
T1=time;

%---- Second selection: K2 ----
vec=[N1(:,1) N2(:,1) N3(:,1)];
Q=vec(4:5,:);
V=vec(1:3,:);

K=Q*inv(V); % STATE FEEDBACK GAIN: K2
sim('exp610',[0 8]); % system simulation
X2=states;
T2=time;

%----- RESULTS PLOT -----
plot(T1,X1(:,1),'b',T2,X2(:,1),'r--');
legend('x_1 for K1','x_1 for K2');
title('State variation for x_1');
xlabel('Time (sec)');

figure;plot(T1,X1(:,2),'b',T2,X2(:,2),'r--');
legend('x_2 for K1','x_2 for K2');
title('State variation for x_2');
xlabel('Time (sec)');

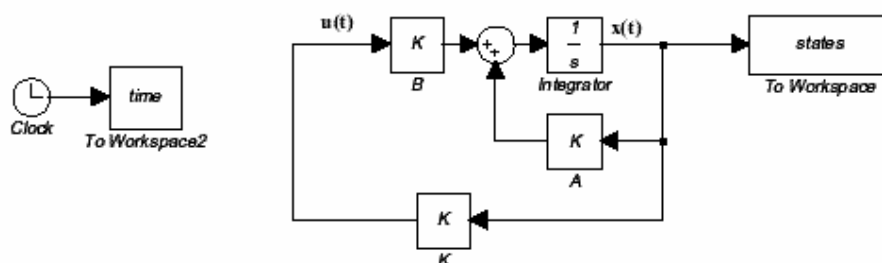
figure;plot(T1,X1(:,3),'b',T2,X2(:,3),'r--');
legend('x_3 for K1','x_3 for K2');
title('State variation for x_3');
xlabel('Time (sec)');

```

شکل ض ۱-۵- (ادامه)

از دستور `null` برای محاسبه فضای تهی استفاده شده است. اما نکته جالب در این برنامه استفاده از تکنیک "محاسبات در Matlab ، شبیه سازی با Simulink" استفاده شده است. دستور `sim` برای همین منظور مورد استفاده قرار گرفته است. ذیلاً به بررسی نحوه استفاده از این دستور می پردازیم:

همانطور که می‌دانیم Simulink یک محیط گرافیکی برای شبیه‌سازی سیستم‌های دینامیکی است. در این محیط بلوک‌های متعددی نظیر: نمایش فضای حالت، نمایش تابع تبدیل، بهره اسکالر، انتگرالگیر و ... در نظر گرفته شده است که طراح با اتصال آنها به یکدیگر سیستم خود را شبیه‌سازی می‌کند. هر یک از این بلوک‌ها دارای تعدادی پارامتر است که طراح قبل از شروع شبیه‌سازی باید آنها را تعیین کند. به عنوان مثال بلوک نمایش فضای حالت دارای چهار پارامتر به صورت ماتریسهای A ، B ، C و D است که توسط طراح مقداردهی می‌شوند.^۱ یک روش مقداردهی به این پارامترها از درون محیط Matlab است. چنانچه به یک ماتریس در محیط Matlab مقدار نسبت داده شود، این ماتریس در هر بلوک در محیط Simulink قابل فراخوانی خواهد بود فقط کافیست که در محل وارد کردن آن پارامتر، نام ماتریس نوشته شود. اکنون به برنامه خودمان بازمی‌گردیم. دستور `sim('exp610',[0 8])` فایل `exp610.mdl` که یک فایل در محیط Simulink است را برای اجرا فراخوانی می‌کند و بازه زمانی اجرای آن را ۸ ثانیه قرار می‌دهد. فایل `exp610.mdl` در شکل ض ۱-۶ آمده است.



شکل ض ۱-۶- فایل `exp610.mdl`

هر یک از بلوک‌های مشخص شده با اسامی A ، B و K دارای یک پارامتر به نام Gain Matrix هستند^۲ که باید به ترتیب نامهای A ، B و K در آنها نوشته شود. ماتریسهای A ، B و K قبل از اجرای دستور `sim('exp610',[0 8])` در برنامه شکل ض ۱-۵ تعریف شده‌اند. پس از اجرای این فایل در محیط Simulink متغیر state حاوی حالت‌های سیستم و نیز متغیر `time` که حاوی زمان‌های متناظر است به محیط Matlab بازگردانده می‌شود. علاوه برای دادن مقدار اولیه به حالت‌های سیستم در فایل `exp610.mdl` پارامتر initial condition در بلوک integrator در نظر گرفته شده است. نکته قابل توجه این است که دو فایل مطرح شده در شکل‌های ض ۱-۵ و ض ۱-۶ باید در یک زیرشاخه ذخیره شوند.

۳-۵- طراحی رُویتگر حالت (مثال ۷-۱)

نحوه طراحی رُویتگر مرتبه کامل در بخش ۷-۱ مورد بررسی قرار گرفت. برنامه زیر مربوط به مثال ۷-۱ می‌باشد.

^۱ نمایش فضای حالت سیستم به صورت زیر فرض می‌شود:

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

^۲ برای ورود به پنجره وارد کردن پارامترها، روی بلوک موردنظر دوبار کلیک کنید.

```

clc
clear all

%~DEFINITION OF PARAMETERS OF INVERTED PENDULUM SYSTEM~

m=.1;      % Kg - mass of pendulum
M=1;      % Kg - mass of cart
I=.06;     % Kg.m^2 - moment of inertia
g=9.81;    % m/sec^2
L=.35;     % meter (length of pendulum is 2*L)
m1_bar=m*L/(m+M);
m2_bar=m*L/(I+m*L^2);
mb=m1_bar*m2_bar;

%-- LINEARIZED DYNAMIC EQUATIONS OF INVERTED PENDULUM ---
A=[0 1 0 0;0 0 -mb*g/(1-mb) 0;0 0 0 1;0 0 m2_bar*g/(1-mb)
  0];
B=[0;1/((m+M)*(1-mb));0;-m2_bar/((m+M)*(1-mb))];
C=[1 0 0 0]; %selecting state 'x(t):displacement' as
               system output

%-- OBSERVABILITY CHECKING ----
observability_rank=rank(obsv(A,C))

%-- SELECTION OF OBSERVER POLES -----
P=4*[-0.7+3i,-0.7-3i,-0.8+0.3i,-0.8-0.3i];
%observer poles are selected 4 times faster than
%closed-loop poles of example 5-3

%-- FINDING OBSERVER GAIN USING Ackermann's Formula --
L_ext=acker(A',C',P);
L=L_ext'; % "L" is the Observer Gain

%-- OBSERVER SIMULATION using "SIMULINK" -----
[time,states,output]=sim('observer',[0 2]);
% Matrix "states" has 8 columns: first 4 column are
%real states and the next 4 columns are estimated
columns.

%-- PLOTTING GRAPH OF REAL & ESTIMATED STATES ----
subplot(2,2,1);plot(time,states(:,1),'b:',time,states(:,5)
  ),'b-');title('X_1');
subplot(2,2,2);plot(time,states(:,2),'b:',time,states(:,6)
  ),'b-');title('X_2');
legend('Real State','Estimated State')
subplot(2,2,3);plot(time,states(:,3),'b:',time,states(:,7)
  ),'b-');title('X_3');
subplot(2,2,4);plot(time,states(:,4),'b:',time,states(:,8)
  ),'b-');title('X_4');

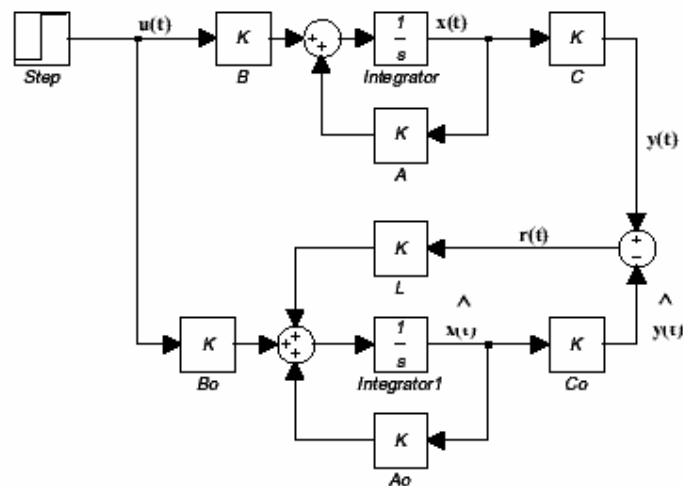
```

شکل ض ۱-۷- برنامه طراحی رُویتگر (مثال ۷-۱)

در این برنامه ابتدا با استفاده از دستور $\text{rank}(\text{obsv}(A, C))$ رتبه ماتریس رؤیت‌پذیری سیستم بررسی می‌گردد. برای طراحی بهره‌رؤیتگر نیز از دستور acker استفاده شده است. شبیه‌سازها در فایل `observer.mdl` (شکل ض ۸-۱) انجام گرفته که توسط دستور زیر فراخوانی شده است:

```
[time, states, output]=sim('observer', [0 2])
```

نتایج اجرای فایل `observer.mdl` در محیط Simulink به صورت ماتریسهای `time` و `states`، `output` به محیط Matlab بازگردانده می‌شود و در آنجا با استفاده از دستور `plot` منحنی تغییرات حالتها و خروجی سیستم را می‌توان رسم کرد.



شکل ض ۸-۱- برنامه `observer.mdl` برای شبیه‌سازی مثال ۷-۱

۳-۶- رؤیتگر مرتبه کاهش یافته (مثال ۷-۲)

در بخش ۷-۲ نشان داده شد که در شرایطی که سیستم رؤیت‌پذیر کامل نیست چگونه می‌توان رؤیتگری برای شناسایی حالت‌های رؤیت‌پذیر سیستم طراحی کرد. برنامه زیر مربوط به مثال ۷-۲ برای بررسی همین موضوع می‌باشد.

```
% Example 7-2, Reduced-Order Observer

clc
clear all

%~ DEFINITION OF PARAMETERS OF INVERTED PENDULUM SYSTEM ~~

m=.1;      % Kg - mass of pendulum
M=1;       % Kg - mass of cart
I=.06;     % Kg.m^2 - moment of inertia
g=9.81;    % m/sec^2
L=.35;     % meter (length of pendulum is 2*L)
```

شکل ض ۹-۱- رؤیتگر مرتبه کاهش یافته (مثال ۷-۲)

```

m1_bar=m*L/(m+M);
m2_bar=m*L/(I+m*L^2);
mb=m1_bar*m2_bar;

%- LINEARIZED DYNAMIC EQUATIONS OF INVERTED PENDULUM --

A=[0 1 0 0;0 0 -mb*g/(1-mb) 0;0 0 0 1;0 0
   m2_bar*g/(1-mb) 0];
B=[0;1/((m+M)*(1-mb));0;-m2_bar/((m+M)*(1-mb))];
C=[1 0 0 0;0 0 1 0];
%Selecting state 'x(t)' & 'teta(t)' as measured
%outputs, Other states should be estimated!

%-- SELECTING OBSERVER POLES -----
D=[-10 0;0 -10];

%-- SELECTING "Q" AND FINDING "A_taw" ---
Q=[0,0,1,0;1,0,0,0;0,0,0,1;0,1,0,0];
A_taw=inv(Q)*A*Q;

A11_taw=A_taw(1:2,1:2);
A12_taw=A_taw(1:2,3:4);
A21_taw=A_taw(3:4,1:2);
A22_taw=A_taw(3:4,3:4);

%-- FINDING "R" & "T" --
L_taw=(D-A11_taw)*inv(A21_taw);
T=A12_taw+L_taw*A22_taw-D*L_taw;

L=[-10 1 0 0;0 0 -10 1];
R=L*B;

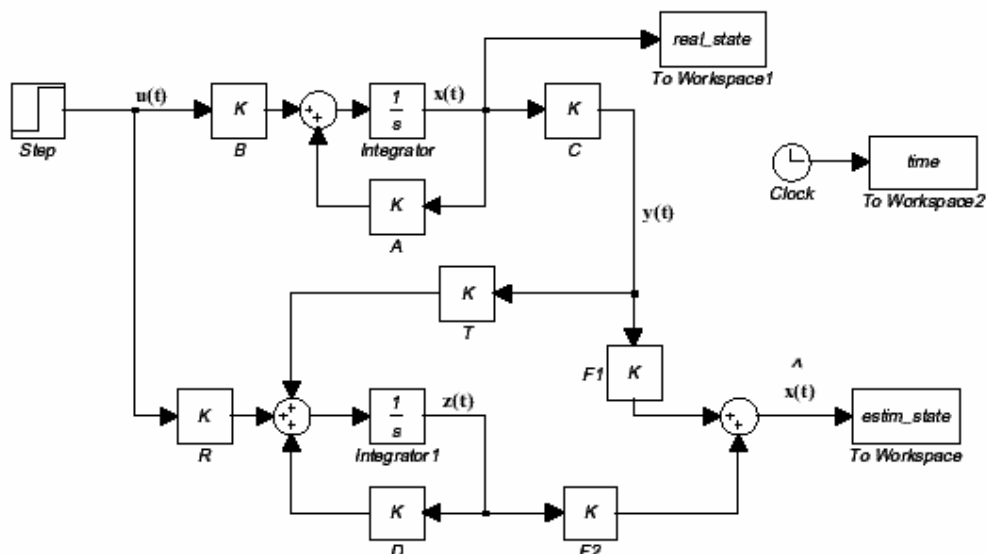
%-- FINDING "F1" & "F2" --
F=inv([C;L]);
F1=F(:,1:2);
F2=F(:,3:4);

%-- SIMULATION USING "SIMULINK" --
sim('reduced_observer',[0 1]);
% this simulation will return "time","real_states" &
% "estimated_states" to workspace.

%-- DRAWING GRAPHS ---
subplot(2,2,1);plot(time,real_state(:,1));
legend('Measured State');title('X_1');
subplot(2,2,2);plot(time,real_state(:,2),'b:',time,esti
m_state(:,2),'b-');
legend('Real State','Estimated State');title('X_2');
subplot(2,2,3);plot(time,real_state(:,3));legend('Measu
red State');title('X_3');xlabel('Time(sec)');
subplot(2,2,4);plot(time,real_state(:,4),'b:',time,esti
m_state(:,4),'b-');legend('Real State','Estimated
State');title('X_4');xlabel('Time(sec)');

```

شكل ض ١-٩- (ادامه)



شکل ض ۱۰-۱- فایل reduced_observer.mdl مورد استفاده در برنامه شکل ض ۹-۱

۷-۳- کنترل فیدبک حالت با رؤیتگر (مثال ۷-۳)

پس از طراحی جداگانه فیدبک حالت و رؤیتگر، در بخش ۷-۴ نحوه تلفیق ایندو و بدست آوردن کنترل کننده بررسی گردید. برنامه زیر مربوط به بررسی این موضوع در مثال ۷-۳ می باشد.

```
% Example 7-3, State Feedback+State Observer

clc
clear all
%----- PARAMETERS DEFINITION -----
a11=-0.0507;
a12=-3.861;
a21=-0.00117;
a22=-0.5164;
a31=-0.000129;
a32=1.4168;
a33=-0.4932;
b1=0;
b2=-0.0717;
b3=-1.645;
g=9.81;

%----- OPEN-LOOP SYSTEM: X_dot=AX+BU ----
A=[a11 a12 0 -g;a21 a22 1 0;a31 a31 a33 0;0 1 0 0];
B=[b1;b2;b3;0];
C=[1 0 0 0];

%----- CONTROLLABILITY & OBSERVABILITY CHECKING -----
controllability_rank=rank(ctrb(A,B))
observability_rank=rank(observ(A,C))
```

شکل ض ۱۱-۱- کنترل فیدبک حالت با رؤیتگر (مثال ۷-۳)

```

%-- SELECTING POLES OF CLOSED-LOOP SYSTEM AND OBSERVER --
Pc=[-2, -2.5, -1, -1.5];      %closed-loop poles
Po=3*[-2, -2.5, -1, -1.5];    %observer poles

%---- CONTROLLER AND OBSERVER GAIN ----
K=acker(A,B,Pc);
L=(acker(A',C',Po))';

%---- SIMULATION USING "SIMULINK" ----
sim('cont',[0 10]);           %FeedBack with Real states
sim('obsv_cont',[0 10]);      %FeedBack with Estimated states

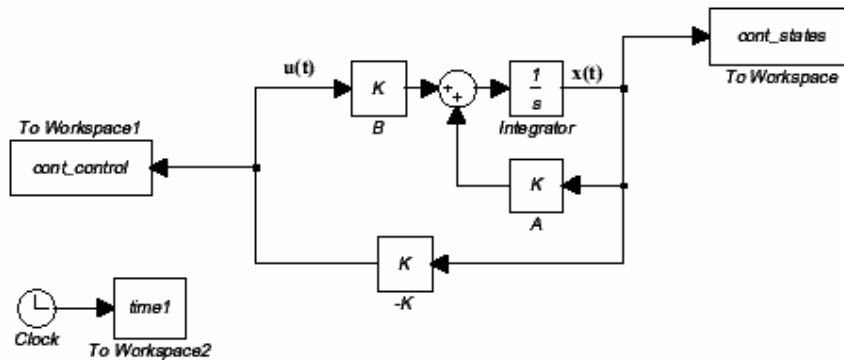
%---- DRAWING GRAPHS ----
plot(time1,cont_control,'b:',time2,obsv_cont_control,'b-');
xlabel('Time(sec)');
legend('Feedbak of real states','Feedbak of estiated states');
title('Control Signal');

figure;
for k=1:size(A)
    subplot(2,2,k);
    plot(time1,cont_states(:,k),'b:',time2,obsv_cont_states(:,k),'b-');
end
subplot(2,2,4);legend('Real State','Estimated State');
xlabel('Time(sec)');
subplot(2,2,3);xlabel('Time(sec)');

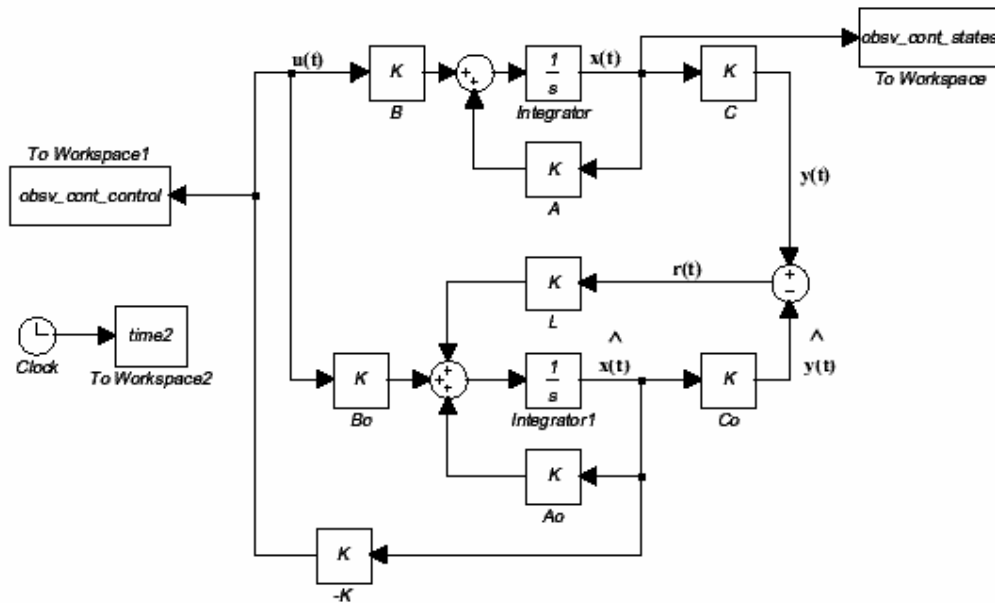
```

شکل ض ۱-۱۱- (ادامه)

فایل‌های cont.mdl و obsv_cont.mdl که در این برنامه مورد استفاده قرار گرفته‌اند به صورت زیر هستند:



شکل ض ۱-۱۲- فایل cont.mdl



شکل ض ۱-۱۳- فایل obsv_cont.mdl

شرایط اولیهٔ بلوک *integrator* در هر دو فایل فوق به صورت $[0.2, 0.1, -0.5, 0.5]$ در نظر گرفته شده است. این شرایط اولیه برای بلوک *integrator1* در فایل obsv_cont.mdl صفر در نظر گرفته شده است.

۳-۸- کنترل فیدبک حالت با رؤیتگر مرتبه کاهش یافته (مثال ۷-۴)

در مثال قبل از یک رؤیتگر کامل برای شناسایی حالت‌های سیستم استفاده گردید. برنامهٔ زیر که مربوط به مثال ۷-۴ است چگونگی کنترل فیدبک حالت را با رؤیتگر مرتبه کاهش یافته بررسی می‌کند.

```
% Example 7-4: State Feedback + Reduced-Order Observer

clc
clear all

%~~ DEFINITION OF PARAMETERS OF Longitudinal motion ~~

A=[-0.007  0.012  0 -9.81;-0.128 -0.54  1 0;0.064 0.96 -0.99
    0;0 0 1 0];
B=[0;-0.036;-12.61;0];
C=[1 0 0 0;0 1 0 0];
% Selecting state 'x(t)' & 'teta(t)' as measured outputs,
% Other states should be estimated!

%-- SELECTING OBSERVER POLES -----
D=[-10 0;0 -10];
```

شکل ض ۱-۱۴- کنترل فیدبک حالت با رؤیتگر مرتبه کاهش یافته (مثال ۷-۴)

```

%-- SELECTING "Q" AND FINDING "A_taw" ---
Q=[0,0,1,0;0,0,0,1;0,1,0,0;1,0,0,0];
A_taw=inv(Q)*A*Q;

A11_taw=A_taw(1:2,1:2);
A12_taw=A_taw(1:2,3:4);
A21_taw=A_taw(3:4,1:2);
A22_taw=A_taw(3:4,3:4);

%-- FINDING "R" & "T" --
L_taw=(D-A11_taw)*inv(A21_taw);
T=A12_taw+L_taw*A22_taw-D*L_taw;

L=[L_taw [0 1;1 0]];
R=L*B;

%-- FINDING "F1" & "F2" --
F=inv([C;L]);
F1=F(:,1:2);
F2=F(:,3:4);

K=[0.922 -6.5955 -2.35 -8.977];
% state feedback from example 6-8

%-- SIMULATION USING "SIMULINK" --
sim('reduced_obsv_cont',[0 10]);
% this simulation will return "time","real_states" &
% "estimated_states" to workspace.

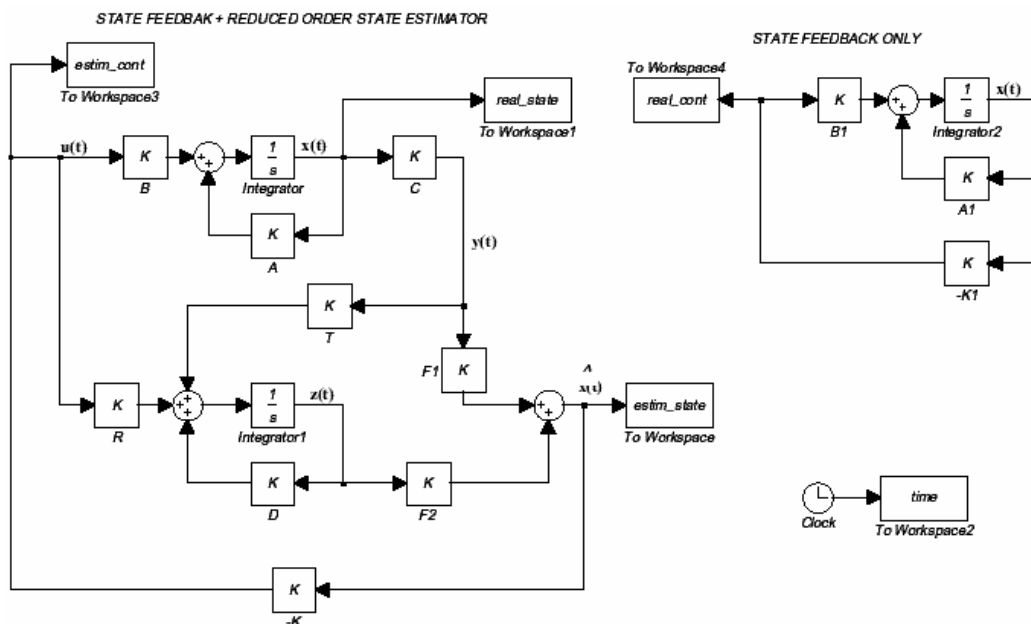
%-- DRAWING GRAPHS ---
subplot(2,2,1);plot(time,estim_state(:,1));
legend('Measured State');title('X_1');
subplot(2,2,2);plot(time,estim_state(:,2));
legend('Measured State');title('X_2');
subplot(2,2,3);plot(time,estim_state(:,3));
legend('Estimated State');title('X_3');
xlabel('Time(sec)');
subplot(2,2,4);plot(time,estim_state(:,4));
legend('Estimated State');title('X_4');
xlabel('Time(sec)');

[rt,ct]=size(time);
kk=(rt-mod(rt,4.5))/4.5;
figure;
plot(time(1:kk),real_cont(1:kk),'b:',time(1:kk),estim_cont(1:kk),'b-');xlabel('Time(sec)');
legend('With Real States','With Estimated States');
title('Control Signal')

```

شکل ض ۱-۱۴- (ادامه)

فایل reduced_obsv_cont.mdl که در برنامه فوق مورد استفاده قرار گرفته در شکل ض ۱-۱۵ آمده است. در این فایل عملکرد یک فیدبک حالت با رُویتگر مرتبه کاهش یافته با عملکرد فیدبک حالت در حضور همه حالت‌های سیستم مقایسه شده است.



شکل ض ۱-۱۵- فایل reduced_obsv_cont.mdl مورد استفاده در برنامه شکل ض ۱-۱۴

۹-۳- طراحی کنترل کننده بهینه به صورت فیدبک حالت (مثال ۸-۱)

در فصل هشتم مروری بر بحث کنترل بهینه و طراحی کنترل کننده های بهینه خطی صورت گرفت. در اولین مثال این فصل (مثال ۸-۱) یک طراحی فیدبک حالت برای کمینه سازی یک شاخص عملکرد درجه دو انجام شد که برنامه زیر مربوط به آن می باشد.

```
% Example 8-1: Optimal State Feedback

clc

%--- SYSTEM DEFINITION ---
A=[0 1;0 0];
B=[0;1];

%===== FIRST SELECTION FOR 'R1' & 'R2'=====
%- SELECTION OF "R1" & "R2" ACCORDING TO COST FUNCTION -
R1=[1 0;0 0];
R2=1;
K=gain_cal(A,B,R1,R2);

%--- SIMULATION ---
sim('optimal',[0 5]);state1=states;
control1=control;time1=time;
```

شکل ض ۱-۱۶- کنترل بهینه فیدبک حالت (مثال ۸-۱)

```

%===== SECOND SELECTION FOR 'R1' & 'R2'=====
%- SELECTION OF "R1" & "R2" ACCORDING TO COST FUNCTION -
R1=[10 0;0 0];
R2=1;
K=gain_cal(A,B,R1,R2);
%--- SIMULATION ---
sim('optimal',[0 5]);state2=states;
control2=control;time2=time;

%===== THIRD SELECTION FOR 'R1' & 'R2'=====
%- SELECTION OF "R1" & "R2" ACCORDING TO COST FUNCTION -
R1=[100 0;0 0];
R2=1;
K=gain_cal(A,B,R1,R2);
%--- SIMULATION ---
sim('optimal',[0 5]);state3=states;
control3=control;time3=time;

%===== FORTH SELECTION FOR 'R1' & 'R2'=====
%- SELECTION OF "R1" & "R2" ACCORDING TO COST FUNCTION -
R1=[10 0;0 0];
R2=5;
K=gain_cal(A,B,R1,R2);
%--- SIMULATION ---
sim('optimal',[0 5]);state4=states;
control4=control;time4=time;

%===== FIFTH SELECTION FOR 'R1' & 'R2'=====
%- SELECTION OF "R1" & "R2" ACCORDING TO COST FUNCTION -
R1=[10 0;0 0];
R2=25;
K=gain_cal(A,B,R1,R2);
%--- SIMULATION ---
sim('optimal',[0 5]);state5=states;
control5=control;time5=time;

##### RESULTS PLOTTING #####

subplot(2,1,1);plot(time1,state1(:,1),'b:',time2,state2(:,1),'b-',time3,state3(:,1),'b--');
legend('R1=1','R1=10','R1=100');

subplot(2,1,2);plot(time1,state1(:,2),'b:',time2,state2(:,2),'b-',time3,state3(:,2),'b--');
legend('R1=1','R1=10','R1=100');xlabel('Time(sec)');

figure;plot(time1,control1,'b:',time2,control2,'b-',time3,control3,'b--');
legend('R1=1','R1=10','R1=100');xlabel('Time(sec)');
title('Optimal Control Signal: Effect of "R1" Variation')

figure;plot(time2,control2,'b:',time4,control4,'b-',time5,control5,'b--');

```

شكل ض ١-١٦- (ادامه)

```
legend('R2=1','R2=5','R2=25',4);xlabel('Time(sec)');
title('Optimal Control Signal: Effect of "R2" Variation')
```

شکل ض ۱-۱۶- (ادامه)

محاسبه بردار فیدبک حالت در تابع Gain_cal.m انجام گرفته است. این تابع، که برنامه آن در شکل ض ۱-۱۷ آمده است، ماتریس A ، بردار B و ماتریسهای وزنی R_1 و R_2 را دریافت کرده، با حل معادله ریکاتی مقدار بردار K را محاسبه می کند. برای حل معادله ریکاتی پیوسته از دستور aresolv استفاده شده است.

```
function [K]=gain_cal(A,B,R1,R2)

%--- SOLVING RICCATI EQUATION (15-3-8) ---
R=B*inv(R2)*B';
Q=R1;
[P1,P2,LAMP,PERR,WELLPOSED,P] = aresolv(A,Q,R,'schur');
K=inv(R2)*B'*P; %state feedback
```

شکل ض ۱-۱۷- تابع Gain_cal.m به کاررفته در برنامه شکل ض ۱-۱۶

۳-۱۰- کنترل بهینه و حساب تغییرات (مثال ۸-۲)

در مثال ۸-۲ (که برنامه زیر مربوط به همین مثال است) کنترل بهینه به نحوی انجام گرفته است که علاوه بر کمینه شدن شاخص عملکرد، حالت های سیستم در لحظه $t = 2 \text{ sec}$ به صفر برسند.

```
% Example 8-2: Calculus of Variation

clc
clear all
%--- SYSTEM DEFINITION ---
A=[0 1;0 0];
B=[0;1];

%--- SELECTION OF "R1" & "R2" & "P" ACCORDING TO COST
FUNCTION ---
R1=[1000 0;0 0];
R2=20;
P1=[1 0;0 1];

%--- SYSTEM PERFORMANCE TIME AND BOUNDARY VALUES ---
t=.001:0.001:2;
[rt,ct]=size(t);

x(:,1)=[2;6];
P=P1; % final value for P(t) (it means P(t=2))
```

شکل ض ۱-۱۸- حساب تغییرات (مثال ۸-۲)

```

%--- SYSTEM SIMULATION USING EQ. (34-4-8)---

%~ IN FIRST FOR-LOOP, WE WILL FIND THE SEQUENCE OF
%~ FEEDBACK GAIN "K(t)" WHICH SHOULD APPLY TO SYSTEM IN
%~ ORDER TO "x(t=2)=0"

for k=1:ct-1
    P_dot=-(R1-P*B*inv(R2)*B'*P+P*A+A'*P);
    P=P-0.001*P_dot;
    %The sign '-' is because of backward iteration!

    K(ct-k,:)=inv(R2)*B'*P;    % Feedback gain
end

%~ IN SECOND FOR-LOOP, WE WILL APPLY CONTROL SEQUENCE
%~ TO THE SYSTEM
for k=1:ct-1
    u(k)=-K(k,:)*x(:,k);
    % Control signal
    x(:,k+1)=x(:,k)+0.001*(A*x(:,k)+B*u(k));
    % Applying control signal to system
end

%--- DRAWING GRAPHS ----

%Drawing states
plot(t,x(1,1:ct),'b:',t,x(2,1:ct),'b-');
xlabel('Time(sec)');legend('x_1(t)','x_2(t)');
title('System States');

%Drawing Control Signal
figure;plot(t(1:ct-1),u);xlabel('Time(sec)');
title('Control Signal');

%Drawing Feedback Gain
figure;
plot(t(1:ct-1),K(:,1),'b:',t(1:ct-1),K(:,2),'b-');
legend('K_1(t)','K_2(t)');
xlabel('Time [sec]');title('FeedBack Gain');

```

شکل ض ۱-۱۸- (ادامه)

۳-۱۱- کنترل بهینه و درجه پایداری (مثال ۸-۳)

در مثال ۸-۳ کنترل کننده بهینه به نحوی طراحی گردید که سیستم حلقه-بسته بهینه دارای درجه پایداری ۲- باشد. برنامه زیر مربوط به این مثال است.

```
% Example 8-3: Optimal Control & Degree of Stability

clc

%--- SYSTEM DEFINITION ---
A=[0 1;0 0];
B=[1 0;0 1];

%- SELECTION OF "R1" & "R2" ACCORDING TO COST FUNCTION -
R1=[1 1;1 1];
R2=[1 0;0 1];

%--- SOLVING RICCATI EQUATION (15-3-8) ---
R=B*inv(R2)*B';
Q=R1;
[P1,P2,LAMP,PERR,WELLPOSED,P_alfa] =
    aresolv(A+2*eye(2),Q,R,'schur');

K_alfa=inv(R2)*B'*P_alfa; %state feedback

%--- SIMULATION ---
sim('optimal',[0 2.5]);

plot(time,control);
title('Control Signal');xlabel('Time [sec]');

figure;
plot(time,states(:,1),'b-',time,states(:,2),'b-');
legend('x_1(t)','x_2(t)');
title('System States');xlabel('Time [sec]');

%--- CLOSED-LOOP POLES ---
closed_loop_poles=eig(A-B*K_alfa)
```

شکل ض ۱-۱۹- کنترل بهینه و درجه پایداری (مثال ۸-۳)

۳-۱۲- فیلتر کالمن (مثال ۸-۴)

برنامه زیر مربوط به طراحی یک فیلتر کالمن برای رؤیت حالت‌های سیستم در محیط نویزی مطابق مثال (۸-۴) می‌باشد. بعلاوه، عملکرد این فیلتر با رؤیتگر لیونبرگر مقایسه گردیده است. می‌دانیم که فیلتر کالمن یک رؤیتگر بهینه است.

```

% Example 8-4: Kalman Filter

clc
clear all

%--- SYSTEM DEFINITION ---
A=[0 1;0 0];
B=[0 1]';
C=[1 0;0 1];

%~~~ Part1: KALMAN FILTER DESIGN ~~~
%-----
%-SELECTION OF "R1" & "R2" ACCORDING TO COST FUNCTION -
R1=1;
R2=[2 0;0 1];

%--- SOLVING RICCATI EQUATION (8-3-15) ---
R=C'*inv(R2)*C;
Q=B*R1*B';
[P1,P2,LAMP,PERR,WELLPOSED,P] =
aresolv(A',Q,R,'schur');

L_Kalman=inv(R2)*C*P; %state feedback

%~~~ Part2: LUENBERGER OBSERVER DESIGN ~~~
P=[-2 -3]; %desired observer poles
L_t=place(A',C',P);
L_Luenberger=L_t';

%--- SIMULATION ---
sim('compare',[0 7]);

plot(time,state1(:,1),'b:',time,state1(:,2),'b--',
time,state1(:,3),'b-');
title('States variation: x_1');xlabel('Time [sec]');
legend('Real','Luenberger estimated','Kalman
estimated',2);

figure;
plot(time,state2(:,1),'b:',time,state2(:,2),'b--',
time,state2(:,3),'b-');
legend('Real','Luenberger estimated','Kalman
estimated',2);
title('States variation: x_2');xlabel('Time [sec]');

```

شکل ض ۱-۲۰- فیلتر کالمن (مثال ۸-۴)

۳-۱۳- کنترل کننده فیدبک خروجی با فیلتر کالمن (مثال ۸-۵)

در مثال ۸-۵ کنترل کننده فیدبک خروجی با فیلتر کالمن برای دو سیستم مطرح شده در مثالهای ۶-۱ و ۷-۳ طراحی گردید که برنامه های زیر مربوط به این مثال هستند.

```

% Example 8-5 (For system of example 6-1)

clc
clear all

%--- SYSTEM DEFINITION ---
A=[1 6 -3;-1 -1 1;-2 2 0];
B=[1 1 1]';
C=[0 1 -1];

%--- Part1: KALMAN FILTER DESIGN ---
%-----
%- SELECTION OF "R1" & "R2" ACCORDING TO COST FUNCTION -
R1=5;
R2=5;

%--- SOLVING RICCATI EQUATION (15-3-8) ---
R=C'*inv(R2)*C;
Q=B*R1*B';
[P1,P2,LAMP,PERR,WELLPOSED,P] =
aresolv(A',Q,R,'schur');

L_Kalman=inv(R2)*C*P; %state feedback

%--- Part2: LUENBERGER OBSERVER DESIGN ---
P=[-2 -3 -1]; %desired observer poles
K=acker(A,B,P);

%--- SIMULATION ---
sim('kalman_cont',[0 30]);
plot(time,output);
title('System Output');xlabel('Time [sec]');

figure;plot(time,control);
title('Control Signal');xlabel('Time [sec]');

```

شکل ض ۱-۲۱ (الف) کنترل فیدبک خروجی با فیلتر کالمن (مثال ۸-۵)

```

% Example 8-5 (For system of example 7-3)

clc
clear all

%----- PARAMETERS DEFINITION -----
a11=-0.0507;
a12=-3.861;
a21=-0.00117;
a22=-0.5164;
a31=-0.000129;

```

شکل ض ۱-۲۲ (ب) کنترل فیدبک خروجی با فیلتر کالمن (مثال ۸-۵)

```

a32=1.4168;
a33=-0.4932;
b1=0;
b2=-0.0717;
b3=-1.645;
g=9.81;

%----- OPEN-LOOP SYSTEM: X_dot=AX+BU -----
A=[a11 a12 0 -g;a21 a22 1 0;a31 a31 a33 0;0 1 0 0];
B=[b1;b2;b3;0];
C=[1 0 0 0];

%~~~ Part1: KALMAN FILTER DESIGN ~~~~
%-----
%-SELECTION OF "R1" & "R2" ACCORDING TO COST FUNCTION -
R1=5;
R2=5;

%--- SOLVING RICCATI EQUATION (15-3-8) ---
R=C'*inv(R2)*C;
Q=B*R1*B';
[P1,P2,LAMP,PERR,WELLPOSED,P] =
    aresolv(A',Q,R,'schur');

L_Kalman=inv(R2)*C*P; %state feedback

%~~~ Part2: LUENBERGER OBSERVER DESIGN ~~~
P=[-2 -2.5 -3 -4]; %desired observer poles
K=acker(A,B,P);

%--- SIMULATION ---
sim('kalman_cont',[0 12]);
plot(time,output);
title('System Output');xlabel('Time [sec]');

figure;plot(time,control);
title('Control Signal');xlabel('Time [sec]');

```

شكل ض ۱-۲۲ (ب) - (ادامه)