باسمه تعالی

نگاهی به کاربرد نرم افزار Matlab
در
کنترل مدرن

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

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

۱

مقدمه

۲

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=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=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=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 = (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= canon(SYS,'TYPE')
[CSYS,T]= 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 می‌گردد.

الف - فرم MODAL :

دستور CSYS= 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 ازاء هر زوج قطب موهومی یک بلوک ۲×۲ 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:

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

ب - فرم COMPANION :

دستور CSYS = canon(SYS,'companion') یک تحققCOMPANION از SYS ارائه می دهد که در آن ضرایب معادله مشخصه سیستم روی ستون منتهی الیه سمت راست ماتریس A قرار می گیرد. اگر معادله مشخصه سیستم به صورت زیر باشد :

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

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

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

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

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 می آید. در تحقق MODAL ماتریس P که مجموعه بردارهای ویژه ماتریس A است محاسبه می شود و سپس با معادلات زیر تبدیل تشابهی $T=P^{-1}$ به سیستم اعمال شده و فرم کانونیکال بدست می آید:

$$\dot{x}_c = P^{-1}APx_c + P^{-1}Bu$$
$$y = CPx + Du$$

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 می آید.

- **مثال:** سیستم زیر مفروض است:

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

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 برنامهٔ 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 تتحقق کانونیکال و T تبدیل تشابهی مربوطه است. تحقق کانونیکال به صورت زیر خواهد بود:

$$\dot{x}_c = A_c x + B_c u = \begin{bmatrix} 0.6715 & 2.571 & 0 \\ -2.571 & 0.6715 & 0 \\ 0 & 0 & 6.6571 \end{bmatrix} x_c + \begin{bmatrix} 0.2776 \\ -1.2267 \\ 1.93 \end{bmatrix} u$$

$$y = C_c x + D_c u = \begin{bmatrix} 0.473 & -0.2081 & 0.3178 \end{bmatrix} x_c$$

- **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 معادلهٔ 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 منحصربفرد معادلهٔ ریکاتی زیر را محاسبه می‌کند:

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

به نحوی که همهٔ مقاذیر ویژهٔ ماتریس $A-BB^TX$ سمت چپ محور موهومی قرار گیرند. ماتریس X متقارن[1] است و پاسخ پایدارساز[2] معادلهٔ Ric(X)=0 نامیده می‌شود. این دستور موارد زیر را نیز نتیجه می‌دهد:

۱- بردار L که شامل مقادیر ویژهٔ ماتریس $A-BB^TX$ است.

۲- ماتریس بهرهٔ $G=B^TX$

[1] Symmetric
[2] Stabilizing

۳- ماندهٔ نسبی[3] که به صورت زیر محاسبه می‌شود:

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

دستور `[X,L,G,rr]=care(A,B,Q,R,S,E)` معادلهٔ ریکاتی عمومی‌تری را حل می‌کند:

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

ماتریس بهره در اینحالت $G = R^{-1}(B^T XE + S^T)$ بوده و نیز مقادیر ویژهٔ حلقه بسته در بردار $L = eig(A - B*G, E)$ قرار می‌گیرند.

دو نگارش دیگر دستور care برای کمک به استفاده از آن در کاربردهای مختلف نظیر طراحی کنترل کنندهٔ H_∞ در نظر گرفته شده‌اند[4].

مثال: با فرض

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

معادلهٔ ریکاتی

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

با استفاده از برنامهٔ 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 معادلهٔ ریکاتی را به فرم زیر نتیجه می‌دهد:

```
X=
    0.5895     1.8216
    1.8216     8.8188
```

مقایسهٔ مقادیر ویژهٔ ماتریس A با ماتریس A-B*g نشان می‌دهد که این پاسخ پایدارساز است:

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

- **آلگوریتم:**

دستور care از ماتریس هامیلتونین، هامیلتونین توسعه‌یافته یا آلگوریتم QZ برای حل معادلهٔ ریکاتی استفاده می‌کند[5].

- **محدودیتها:**

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

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

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

---

[3] Relative residual

[4] برای توضیحات کاملتر در این مورد به help نرم‌افزار Matlab مراجعه شود.

[5] این آلگوریتمها 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]=cdf2rdf(V,D)`
- **توضیحات:**

چنانچه مقادیر ویژه ماتریس X موهومی باشند، ماتریسهای V و D که از اجرای دستور `[V,D]=eig(X)` بدست می‌آیند 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 ۲×۲ حقیقی متناظر خواهد شد. البته در این تبدیل ماتریس V تیز تغییر خواهد کرد و دیگرV 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]=eig(X)
   V =
      1.0000 0.4002-0.0191i    0.4002+0.191i
       0           0.6479            0.6479
       0           0+0.6479i         0-0.6479i
   D =
       1           0           0
       0           4+5i        0
       0           0           4-5i
```

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

```
»[V,D]=cdf2rdf(V,D)
   V =
       1     0.4002      -0.0191
       0     0.6479       0
       0     0            0.6479
   D =
       1     0     0
       0     4     5
       0     -5    4
```

۲-۵- دستور ctrb :

- **هدف:** محاسبه ماتریس کنترل پذیری
- **نگارش دستورالعمل:**

```
Co=ctrb(A,B)
Co=ctrb(SYS)
Co=ctrb(SYS.a,SYS.b)
```

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

$$Co = \begin{bmatrix} B & AB & A^2B & \cdots & A^{n-1}B \end{bmatrix}$$

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:

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

برنامهٔ زیر رتبهٔ ماتریس کنترل‌پذیری را محاسبه می‌کند:

```
»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 نمی‌باشد. تعداد حالتهای کنترل ناپذیر سیستم به صورت زیر محاسبه می‌شود:

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

و لذا سیستم یک حالت کنترل ناپذیر دارد.

- **محدودیتها:**

محاسبه Co در پاره‌ای 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:

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

اما اگر $\delta^2 < eps$ باشد که در آن eps وضوح 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 حالتهای کنترل پذیر و کنترل ناپذیر سیستم

۲-۶- دستور Ctrbf:

- **هدف:** تهیه فرم پلکانی کنترل پذیر (جداسازی حالتهای کنترل پذیر و کنترل ناپذیر)
- **نگارش دستورالعمل:**

```
[Abar,Bbar,Cbar,T,K]=ctrbf(A,B,C)
[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 می‌تواند حالتهای کنترل ناپذیر سیستم را از حالتهای کنترل پذیر جدا کند. ماتریس تبدیل unitary T است[1] و داریم :

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

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

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

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

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

یعنی حالتهای کنترل ناپذیر در تعیین تابع تبدیل سیستم اثری ندارد.

دستور `[Abar,Bbar,Cbar,T,K]=ctrbf(A,B,C)` سیستم فضای حالت نمایش داده شده با A، B و C را به سیستمی با $\overline{A}, \overline{B}, \overline{C}$ 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)`[2] تعداد حالتهای A_c را نشان می دهد.

عنصر `tol` در دستور`ctrbf(A,B,C,tol)` مقدار تلرانس مجاز در محاسبه زیر فضاهای کنترل پذیر/کنترل ناپذیر را نشان می دهد. اگر مقدار این تلرانس 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]=ctrbf(A,B,C)
```

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

```
Abar=
      -3   0
      -3   2
Bbar=
        0           0
        1.4142      -1.4142
Cbar=
       -0.7071    0.7071
        0.7071    0.7071
```

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

```
T=
    -0.7071      0.7071
     0.7071      0.7071
K=
     1    0
```

---

[1] ماتریس Unitay T است اگر تنها و تنها اگر :

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

[2] دستور `SUM(K)` مجموع عناصر بردار K را محاسبه می‌کند.

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=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+abs(k) است و عناصر بردار V روی K امین قطر آن قراردارند. K=0 بیانگر قطر اصلی، K>0 مربوط به قطرهای بالای قطر اصلی و K<0 مربوط به قطرهای زیر آن است.

دستور `X=diag(v)` در واقع همان دستور قبل به ازای K=0 است. دستور `V=diag(x,k)` که در آن x یک ماتریس است ، قطر kام x را در بردارV قرار می دهد. وضعیت علامت K مانند دستور`X=diag(v,k)` می باشد. `V=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
      -8
» V2=diag(Y,1)
   V2=
       2
      -1
```

۲-۸- دستور eig :

- **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*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 می‌شود. 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*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 می‌گرداند.

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 سازی[1] 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 کردن[2] وجود داتشه 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]=eig(A,B,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 cholesky 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 می‌شوند. 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
```

۲-۹- دستور estim:

- **هدف:** تعیین معادلات دینامیکی رویتگر با دریافت معادلات حالت سیستم و بهره رویتگر.
- **نگارش:**

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

- **توضیحات:**

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

X=Ax+Bw
Y=Cx+Dw

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

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

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:

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

بردارهای اندیس sensors و known به ترتیب مشخص می کند که کدامیک از خروجیها قابل اندازه‌گیری هستتند (y) و کدامیک از ورودیها مقدار غیر تصادفی دارند (u). رویتگر est از ورودیهای u و خروجیهای y و تخمین حالتهای سیستم استفاده می کند.

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

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

- **دستورهای مرتبط:**

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

۲-۱۰- دستور exam:

- **هدف:** محاسبه e^X در حالتی‌که 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:

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

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

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

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=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:

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

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

$$W_c = \int_0^\infty e^{A\tau} B.B^T e^{A^T\tau} d\tau$$

$$W_o = \int e^{A^T\tau} C^T.Ce^{A\tau} d\tau$$

و برای سیستمهای گسسته زمان خواهیم داشت:

$$W_c = \sum_{k=0}^{\infty} A^k BB^T (A^T)^K$$

$$W_o = \sum (A^T)^K C^T CA^k$$

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:

در سیستمهای پیوسته: $AW_c + W_c A^T + BB^T = 0$

در سیستمهای گسسته: $AW_c A^T - W_c + BB^T = 0$

گرامیان رویت پذیری نیز در معادلات لیاپانوف زیر صادق است:

در سیستمهای پیوسته: $A^T W_o + W_o A + C^T C = 0$

در سیستمهای گسسته: $A^T W_o A - W_o + C^T C = 0$

بنابراین از دستورات lyap و dlyap نیز می‌توان برای محاسبهٔ 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 حالتهای 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 حالتهای سیستم محاسبه می کند. این تابع به مدلهای پیوسته و گسسته زمان قابل اعمال است. به جز در حالت `[y,t,x]=initial(sys,xo)` در سایر موارد اجرای دستور `initial` با رسم پاسخهای 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 منحنیهای شبیه سازی به طور خودکار چنان تعیین می شود که عملکرد پاسخها را در حالت گذرا کاملاً نشان دهد.

نتیجه دستور `initial(sys,xo,t)` مانند حالت قبل است با این تفاوت که در اینحالت بازه زمانی مطلوب برای محاسبه شرایط سیستم نیز مشخص شده است. می‌توان 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 اینحالت علاوه بر بازه زمانی شبیه سازی (صفر تا T_{final})، گامهای شبیه سازی نیز مشخص شده است . در سیستمهای گسسته ، dt باید با زمان نمونه برداری هماهنگ باشد. در سیستمهای پیوسته dt باید آنقدر کوچک اختیار شود تا رفتار حالت گذرا به خوبی مشخص گردد.

برای رسم همزمان پاسخهای 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 زمانهای شبیه سازی (t) و ماتریس مسیرهای حالت سیستم (x) می‌شود. در اینحالت هیچ منحنی رسم نمی شود. تعداد سطرهای ماتریس y به تعداد زمانهای شبیه سازی (ابعاد t) و تعداد

ستونهای آن به تعداد خروجیهای سیستم است. ماتریس x نیز به تعداد `length(t)` سطر و به تعداد حالتهای 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 برنامهٔ 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 رویتگر (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، رویتگر 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 حالتهای سیستم است چنانکه کواریانس خطای حالت ماندگار کمینه گردد:

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

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 می‌شود.

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]

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

- **محدودیت ها:**

با این فرض که:

$$\overline{Q} = GQG^T$$
$$\overline{R} = R + HN + N^T H^T + HQH^T$$
$$\overline{N} = G(QH^T + N)$$

سیستم و نویز باید شرایط زیر را ارضا کنند:

۱. (C,A) آشکاری‌پذیر باشد(detectable)

۲. $\overline{Q} - \overline{N}\overline{R}^{-1}\overline{N}^T \geq 0$ و R>0

۳. $(A - \overline{N}\overline{R}^{-1}C, \overline{Q} - \overline{N}\overline{R}^{-1}\overline{N}^T)$ 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 لزوما" مربعی نیستند ولی ابعاد آنها سازگار است.

- **محدودیت ها:**

اگر α_1 ، α_2، ... و α_n مقادیر ویژه A و β_1، β_2، ... و β_n مقادیر ویژه B باشند، آنگاه معادله لیاپانوف پیوسته پاسخ منحصر بفرد دارد اگر برای هر زوج (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 uniqe
```

- **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[1] 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 ماتریسp*n باشد، دستور obsv(A,C) ماتریس رویت پذیری زیر را نتیجه خواهد داد:

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

واضح است که با دستور rank[obsv(A,C)] می توان رویت‌پذیری کامل سیستم را بررسی کرد.

اگر sys مدل فضای حالت یک سیستم باشد ، دستور obsv(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:

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

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]=obsvf(A,B,C)
[Abar,Bbar,Cbar,T,k]=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 اینحالت، تبدیل تشابهی T وجود دارد که unitary است و حالتهای رویت‌پذیر و رویت‌ناپذیر سیستم را از هم جدا می کند.

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

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

$$\overline{A} = \begin{bmatrix} A_{no} & A_{12} \\ 0 & A_0 \end{bmatrix}, \overline{B} = \begin{bmatrix} B_{no} \\ B_o \end{bmatrix}, \overline{C} = \begin{bmatrix} 0 & C_o \end{bmatrix}$$

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

دستور [Abar,Bbar,Cbar,T,k]=obsvf(A,B,C) ماتریس های A و B و C را به نحوی که قبلا" گفته شد تولید می کند. T نیز تبدیل تشابهی برای جدا کردن حالتهای کنترل‌پذیر و کنترل‌ناپذیر است. k یک بردار n عضوی است (n 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 sum(k) تعداد حالتهای A_o را نشان خواهد داد. با دستور obsvf(A,B,C,tol) می توان تلرانس 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=place(A,B,P)
[K,prec,message]=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[1]. 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 بردارP قرار دهد. واضح است که در سیستمهای چند ورودی ماتریسK 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[2].

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:

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

اگر D=0 باشد کافیست 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 درltisys تولید می کند. این ویژگی می تواند یکی از مو.ارد ذکر شده در جدول زیر باشد:

نام ویژگی	معرفی ویژگی	نوع ویژگی

[1] قطبهای سیستم حلقه بسته همان مقادیر ویژه A-BK هستند.

[2] 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 نمونه‌برداری	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 وOutputGroup امکان گروه بندی ورودیها و خروجیهای سیستم را به برنامه نویس می دهد. به عنوان مثال در یک سیستم ۵ ورودی، برنامه نویس می‌تواند 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 دستورالعملهای ذکر شده، نحوه مقدار مقدار دهی به این ویژگیها مشخص خواهد شد.

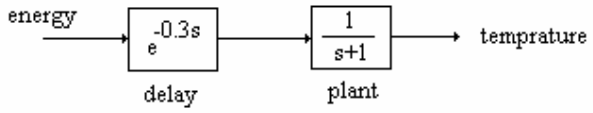
دستور `sys_ss=ss(sys)` مدل sys، تولید شده توسط دستورات `tf` و `zpk` را به مدل `sys_ss` در فضای حالت تبدیل می کند. به این عملیات تحقق فضای حالت گفته می شود.

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 ر ا 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 بهplant ، 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]= ssdata (SYS) سیستمهای زمان پیوسته

  [A,B,C,D,Ts]=ssdata(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[1] 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

۳

بررسی برنامه های مربوط به مثالهای متن کتاب

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

در مثال ۵-۳ برای مدل خطی شدهٔ آونگ معکوس، فیدبک حالت ke=[-1.91 -4.262 -45.49 -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 برروی سیستم غیرخطی آزمود. به این منظور نیازمند حل معادلات دیفرانسیل غیرخطی هستیم. اینکار در نرم افزار Matlab معمولاً با دستورهای ODE[1] انجام می‌گیرد. اما در این مثال به تفصیل، شیوهٔ حل این نوع معادلات بررسی شده است. دو فایل به نامهای EXP5_3 و ode به صورت زیر برای حل این مثال درنظر 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 ض۱-۱- فایل EXP5_3

[1] 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 ض۱-۱- (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 ض۱-۲- 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 ض۱-۲- (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 اولیهٔ پارامترها، معادلات حالت غیرخطی سیستم در یک حلقهٔ For به صورت گام به گام (با طول گام ۰/۰۱) حل می شوند و نتیجهٔ نهایی که تغییرات زاویهٔ 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 اولیهٔ متفاوت برای θ[1] انجام گرفته است و نتیجه در شکل ۵-۵ رسم گردیده است.

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

در مثال ۶-۷ برای سیستم ناپایدار هواپیما کنترل کنندهٔ فیدبک حالت به نحوی طراحی گردید که قطبهای سیستم حلقه بسته در [-0.9+1.9i -0.9-1.9i -1.8+0.6i -1.8-.6i] قرار گرفته و سیستم پایدار شود. برنامهٔ 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 ض۱-۳- برنامهٔ EXP6_8.m

[1] $\theta = 5°, \theta = 20°, \theta = 33.4°$

```
%----- 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];

%----- 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 Ploes
   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 برنامهٔ 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. شبیه‌سازیها در دو بخش رگولاسیون و ردیابی صورت گرفته است. در هردو بخش معادلات حالت سیستم با استفاده از

یک حلقهٔ 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 برنامهٔ فوق هستند.

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

در بخش ۶-۴-۲ نحوهٔ افزودن کنترل انتگرال به منظور ردیابی بدون خطای ورودیهای پله مورد بررسی قرار گرفت. برنامهٔ زیر مربوط به مثال ۶-۹ می‌باشد.

```
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 ض۱-۴- فیدبک حالت با کنترل انتگرال (مثال ۶-۹)

```
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 ض۱-۴- (ادامه)

ابتدا با استفاده از رابطهٔ ۶-۴-۴۵ ماتریسهای $\overline{A}$ و $\overline{B}$ معرفی شده و کنترل‌پذیری انتگرال سیستم به کمک دستور `ctrb` بررسی گردیده است. سپس از دستور `acker` برای طراحی فیدبک حالت اسفاده شده، ماتریسهای K_1 و K_2طبق رابطهٔ ۶-۴-۴۷ جداسازی شده‌اند.

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

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

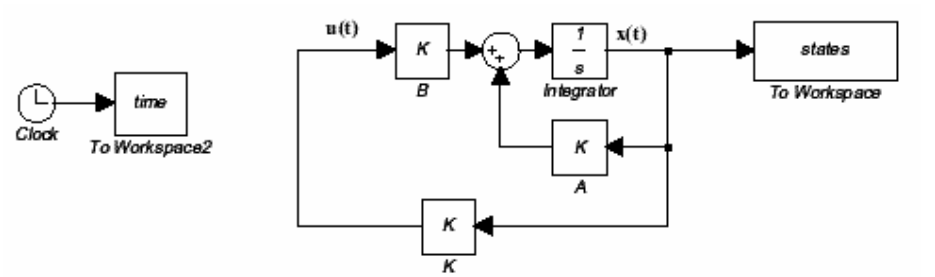
شکل ض۱-۵- اثر تغییر فیدبک حالت بررفتار دینامیکی سیستم (مثال ۶-۱۰)

```
%---- 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 محاسبهٔ فضای تهی استفاده شده است. اما نکتهٔ جالب در این برنامه استفاده از تکنیک "محاسبات در Matlab ، شبیه‌سازی با Simulink " استفاده شده است. دستور `sim` برای همین منظور مورد استفاده قرار گرفته است. ذیلاً به بررسی نحوهٔ استفاده از این دستور می‌پردازیم:

همانطور که می‌دانیم Simulink یک محیط گرافیکی برای شبیه‌سازی سیستمهای دینامیکی است. در این محیط بلوکهای متعددی نظیر: نمایش فضای حالت، نمایش تابع تبدیل، بهرهٔ 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 پارامتراست 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 می شوند[1]. یک روش مقداردهی به این پارامترها از درون محیط Matlab است. چنانچه به یک ماتریس در محیط Matlab مقداری 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 برنامهٔ خودمان بازمی‌گردیم. دستور `sim('exp610',[0 8])` فایل `exp610.mdl` که یک فایل در محیط Simulink است را برای اجرا فراخوانی می‌کند و بازهٔ زمانی اجرای آن را ۸ ثانیه قرار می‌دهد. فایل `exp610.mdl` در شکل ض۱-۶ آمده است.



شکل ض۱-۶- فایل `exp610.mdl`

هر یک از بلوکهای مشخص شده با اسامی A، B و K دارای یک پارامتر به نام Gain Matrix هستند[2] 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 برنامهٔ شکل ض۱-۵ 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. بعلاوه برای دادن مقدار اولیه به حالتهای سیستم در فایل `exp610.mdl` پارامتر initial condition در بلوک integrator درنظر گرفته شده است. نکتهٔ قابل توجه این است که دو فایل مطرح شده در شکلهای ض۱-۵ و ض۱-۶ باید در یک زیرشاخه ذخیره شوند.

۳-۵- طراحی رؤیتگرحالت (مثال ۷-۱)

نحوهٔ طراحی رؤیتگر مرتبهٔ کامل در بخش ۷-۱ مورد بررسی قرار گرفت. برنامهٔ زیر مربوط به مثال ۱-۷ می‌باشد.

---

۱ نمایش فضای حالت سیستم به صورت زیر فرض می‌شود:

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

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

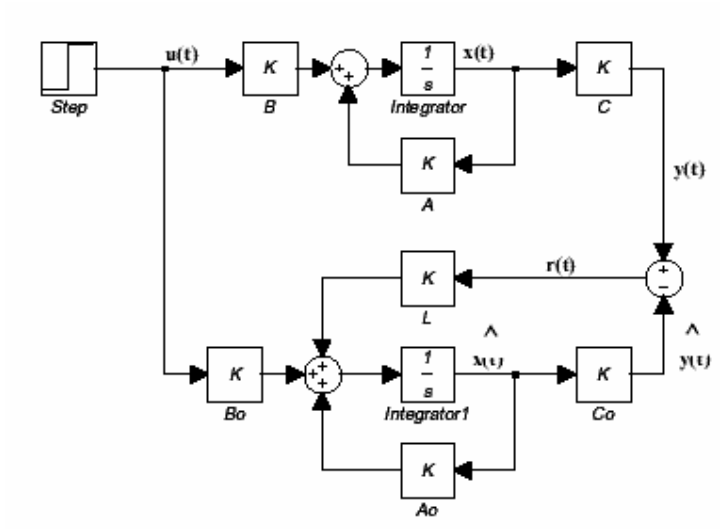
```
     clc
     clear all

%~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.

%-- PLOTING 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 ض۱-۷- برنامهٔ طراحی رؤیتگر (مثال ۷-۱)

در این برنامه ابتدا با استفاده از دستور `rank(obsv(A,C))` رتبهٔ ماتریس رؤیت‌پذیری سیستم بررسی می‌گردد. برای طراحی بهرهٔ رؤیتگر نیز از دستور `acker` استفاده شده است. شبیه‌سازیها در فایل `observer.mdl` (شکل ض۱-۸) 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 `output`، `states` و `time` به محیط Matlab بازگردانده می‌شود و در آنجا با استفاده از دستور `plot` منحنی تغییرات حالتها و خروجی سیستم را می‌توان رسم کرد.



شکل ض۱-۸- برنامهٔ `observer.mdl` برای شبیه‌سازی مثال ۱-۷

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

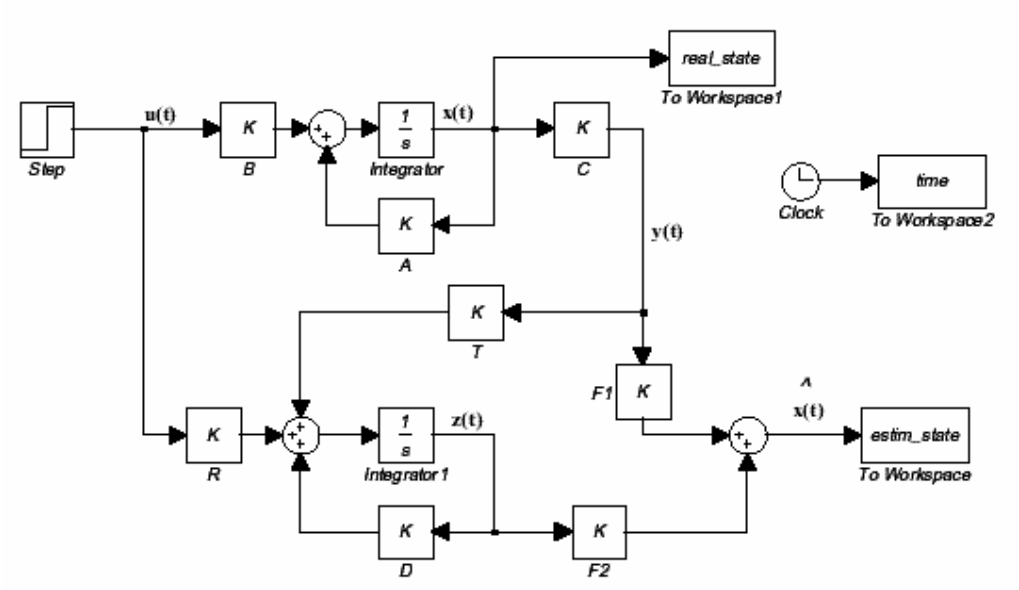
در بخش ۷-۲ نشان داده شد که در شرایطی که سیستم رؤیت‌پذیر کامل نیست چگونه می‌توان رؤیتگری برای شناسایی حالتهای رؤیت‌پذیر سیستم طراحی کرد. برنامهٔ زیر مربوط به مثال ۷-۲ برای بررسی همین موضوع می‌باشد.

```
% 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 ض۱-۹- 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)');
```

شکل ض ۱-۹- (ادامه)



شکل ض۱-۱۰- فایل `reduced_observer.mdl` **مورد استفاده در برنامهٔ شکل ض۱-۹**

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

پس از طراحی جداگانهٔ فیدبک حالت و رؤیتگر، در بخش ۷-۴ نحوهٔ تلفیق ایندو و بدست آوردن کنترل کننده بررسی گردید. برنامهٔ زیر مربوط به بررسی این موضوع در مثال ۷-۳ می‌باشد.

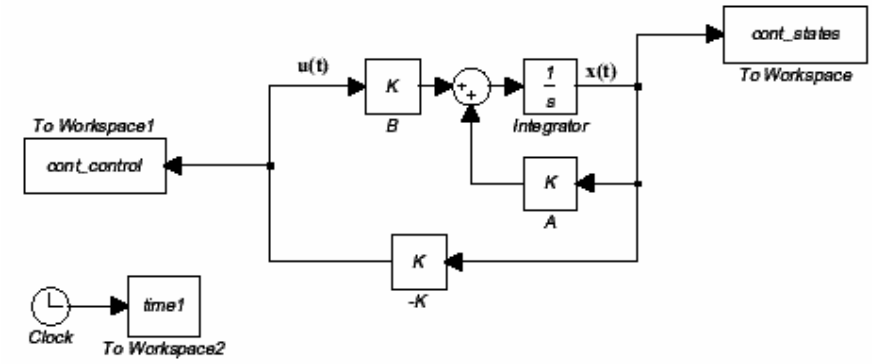
```
% Example 7-3, State Feedback+State Observer

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(obsv(A,C))
```

شکل ض۱-۱۱- کنترل فیدبک حالت با رؤیتگر (مثال ۷-۳)

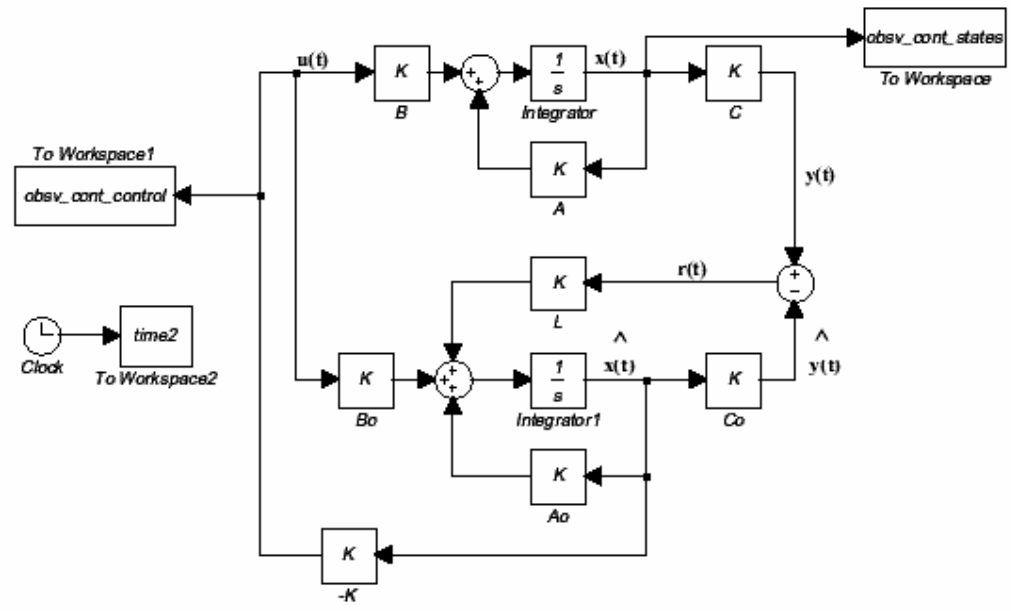
```
%-- SELECTING POLES OF CLOSED-LOOP SYSTEM AND OBSERVER --
 Pc=[-2,-2.5,-1,-1.5];      %closed-loop poles
 Po=3*[-2,-2.5,-1,-1.5];    %observer poles


%---- 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 ض۱-۱۱- (ادامه)

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



شکل ض۱-۱۲- فایل cont.mdl



شکل ض۱-۱۳- فایل `obsv_cont.mdl`

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

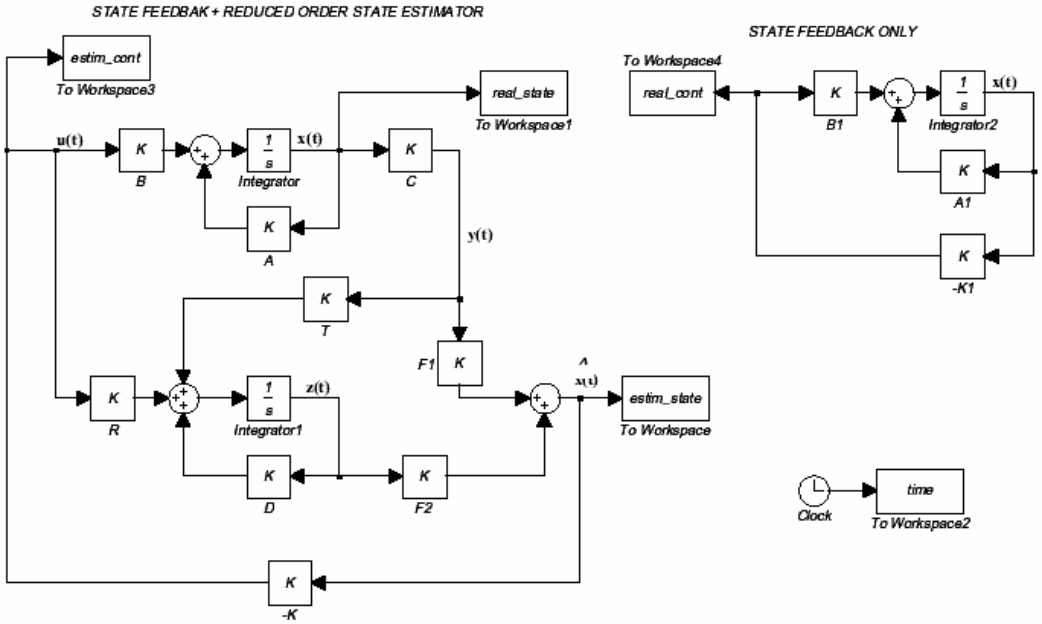
```
% Example 7-4: State Feedback + Reduced-Order Observer

 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 ض۱-۱۴- 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_con
t(1:kk),'b-');xlabel('Time(sec)');
legend('With Real States','With Estimated States');
title('Control Signal')
```

شکل ض۱-۱۴- (ادامه)

فایل `reduced_obsv_cont.mdl` که در برنامهٔ فوق مورد استفاده قرار گرفته در شکل ض۱-۱۵ آمده است. در این فایل عملکرد یک فیدبک حالت با رؤیتگر مرتبه کاهش یافته با عملکرد فیدبک حالت در حضور همهٔ حالتهای سیستم مقایسه شده است.



شکل ض۱-۱۵- فایل `reduced_obsv_cont.mdl` مورد استفاده در برنامهٔ شکل ض۱-۱۴

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

در فصل هشتم مروری بر بحث کنترل بهینه و طراحی کنترل کننده های بهینهٔ 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 برنامهٔ زیر مربوط به آن می‌باشد.

```
% 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 ض۱-۱۶- کنترل بهینهٔ 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--');
```

شکل ض۱-۱۶-(ادامه)

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

شکل ض۱-۱۶-(ادامه)

محاسبهٔ بردار فیدبک حالت در تابع Gain_cal.m انجام گرفته است. این تابع، که برنامهٔ آن در شکل ض۱-۱۷ آمده است، ماتریس A، بردار B و ماتریسهای وزنی R_1 و R_2 را دریافت کرده، با حل معادلهٔ ریکاتی مقدار بردار K را محاسبه می‌کند. برای حل معادلهٔ 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 ض۱-۱۷- تابع Gain_cal.m **به کاررفته در برنامهٔ شکل ض۱-۱۶**

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

در مثال ۸-۲ (که برنامهٔ زیر مربوط به همین مثال است) کنترل بهینه به نحوی انجام گرفته است که علاوه بر کمینه شدن شاخص عملکرد، حالتهای سیستم در لحظهٔ $t = 2$ 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 ض۱-۱۸- 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 ض۱-۱۸- (ادامه)

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

در مثال ۸-۳ کنترل کنندهٔ بهینه به نحوی طراحی گردید که سیستم حلقه-بستهٔ بهینه دارای درجهٔ پایداری ۲- باشد. برنامهٔ 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 ض۱-۱۹- کنترل بهینه و درجهٔ پایداری (مثال ۸-۳)

۳-۱۲- فیلتر کالمن (مثال ۸-۴)

برنامهٔ 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 ض۱-۲۰- فیلتر کالمن (مثال ۸-۴)

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

در مثال ۸-۵ کنترل کنندهٔ 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 ض۱-۲۱(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 ض۱-۲۲(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]');
```

شکل ض۱-۲۲(ب)- (ادامه)