دانشگاه صنعتی اصفهان
دانشکده مهندسی مکانیک

پروژه درس طراحی سیستم های کنترلی

شبیه‌سازی و کنترل سیستم دو درجه آزادی آونگ معکوس

استاد راهنما
دکتر محمدجعفر صدیق

محقق
علی نصر ۸۸۲۶۳۹۳

خرداد ۹۲

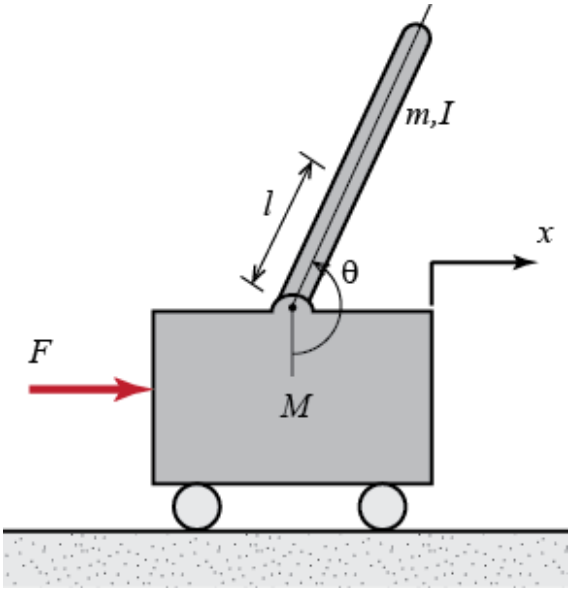
فهرست

فصل اول: مقدمه

این مسئله از یک آونگ معکوس متصل به یک گاری تشکیل‌شده است. سیستم آونگ وارونه یک مثال معمول در کتاب های کنترل و منابع تحقیقاتی است. محبوبیت این مسئله از این جا نشأت می‌شود که این آونگ ناپایدار است و بدون کنترل و حرکت گاری سقوط می‌کند. علاوه بر این دینامیک این سیستم غیرخطی است.
هدف این سیستم کنترلی، پایدارسازی آونگ معکوس به‌وسیله اعمال نیرویی به گاری که به آن متصل است. یک مثال در دنیای واقعی که به طور مستقیم به این سیستم آونگ معکوس مربوط می‌شود مسئله کنترل یک موشک در فاز برخاستن آن است.

مشخصات مسئله

در این حالت آونگ به در فضای ۲ بعدی در نظر گرفته‌شده است و فقط در صفحه عمودی حرکت می‌کند. برای این سیستم نیروی کنترلی F که باعث حرکت افقی گاری می‌شود یک ورودی سیستم است و زاویه آونگ θ و موقعیت افقی گاری x خروجی سیستم می‌باشند. همچنین فرض شده است که پاندول و چرخ های گاری در طول مسئله کاملاً صلب‌اند و همچنین هیچ‌گونه اصطکاکی در مفصل پاندول با گاری در نظر گرفته نشده است.



شکل ۱ - تصویر ساده‌شده مسئله آونگ معکوس

برای مثال فرض می‌کنیم مسئله‌داری مشخصات زیر است:

نماد	مقدار	توضیحات
M	$1\ \mathrm{kg}$	جرم گاری
m	$0.1\ \mathrm{kg}$	جرم آونگ
l	$1\ \mathrm{m}$	فاصله لولا تا مرکز جرم آونگ
I	0	ممان اینرسی آونگ
g	$9.8\ \mathrm{m}/s^2$	شتاب گرانش
F		نیروی وارد شونده به گاری
x		مختصات موقعیت گاری
θ		زاویه آونگ نسبت به خط عمود

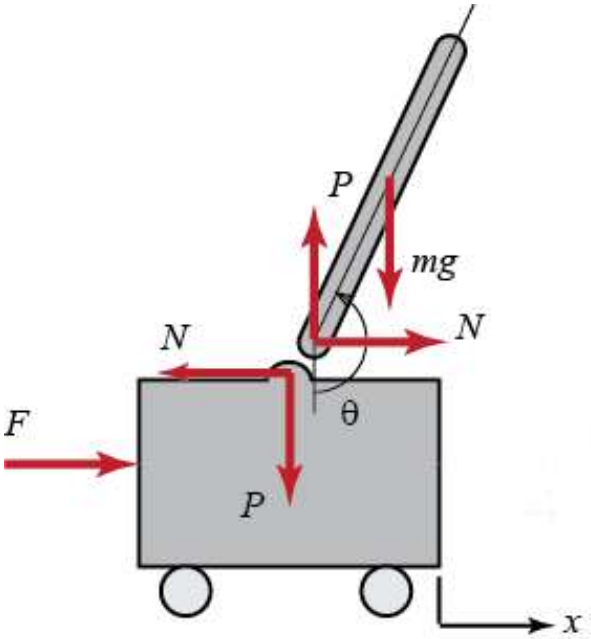
جدول ۱ - مشخصات مسئله

فصل دوم: مدل‌سازی سیستم

در این فصل قصد داریم معادلات سیستم را به‌دست آوریم و با خطی سازی آنها تابع تبدیل و معادلات فضـای حالـت را از آن‌هـا استخراج نماییم.

معادلات سیستم system equations

در شکل زیر نمودار جسم آزاد از دو المان آونگ معکوس نمایش داده‌شده است:



شکل ۲ - نمودار جسم آزاد هر دو المان سیستم آونگ معکوس

پس از جمع نیروهای افقی نمودار جسم آزاد گاری به معادله‌ای به‌صورت زیر دست می‌یابیم:

$$M\ddot{x} + N = F$$

توجه داشته باشید که همچنین می‌توان نیروهای در جهت عمودی وارد به گاری را می‌توان جمع کرد اما هیچ اطلاعـات مفیـدی به‌دست نمی‌آید.

با جمع نیروهای وارد بر آونگ در جهت افقی برای واکنش به نیروی N عبارت زیر به‌دست می‌آید:

$$N = m\ddot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta$$

اگر این معادله را با معادله قبل ترکیب کنیم و نیروی N را حذف کنیم یکی از معادله حالت حاکم برای این سیستم بـه‌صـورت زیر به‌دست می‌آید:

$$(M+m)\ddot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta = F$$

برای به‌دست آوردن معادله دوم حالت برای این سیستم می‌توان از مجموع نیروهای عمودی وارد بـر آونـگ اسـتفاده کـرد. حـل سیستم در طول محور آونگ محاسبات ریاضی را تا حد زیادی ساده می‌کند و معادله زیر به‌دست می‌آید:

$$P\sin\theta + N\cos\theta - mg\sin\theta = ml\ddot{\theta} + m\ddot{x}\cos\theta$$

برای حذف نیروهای داخلی N و P از معادله بالا می‌توان معادله مربوط به دوران آونگ را حول مرکز جرم آن نوشت تا معادله زیر به‌دست آید:

$$-Pl\sin\theta - Nl\cos\theta = I\ddot{\theta}$$

با ترکیب این دو عبارت آخر می‌توان به معادله دوم حالت را به‌دست آورد و نتیجه به‌صورت زیر است:

$$\left(I + ml^2\right)\ddot{\theta} - mgl\sin\theta = -ml\ddot{x}\cos\theta$$

از آنجایی که تجزیه، تحلیل و تکنیک های طراحی سیستم کنترلی به طور معمول برای سیستم های خطی است این مجموعه از معادلات نیاز دارد که خطی شود. به‌صورت خاص، معادلات حول موقعیت پایدار و عمودی آونگ یعنی $\theta = \pi$ خطی می‌کنیم و فرض می‌کنیم که سیستم در یک همسایگی کوچکی از موقعیت پایداری باقی بماند. این فرض در شرایطی تقریباً منطقی است که برای سیستم کنترلی ساخته‌شده زاویه انحراف آونگ بیش از ۲۰ درجه از موقعیت عمودی آونگ نباشد. درصورتی‌که زاویه انحراف از موقعیت تعادل آونگ را ϕ فرض کنیم؛ زاویه آونگ به‌صورت $\theta = \phi + \pi$ می‌شود و باز هم فرض کنیم که انحراف ϕ مقداری کوچک است می‌توانیم با تقریب های زیر توابع غیرخطی معادلات سیستم را خطی کنیم.

$$\cos\theta = \cos\left(\phi + \pi\right) \approx -1$$

$$\sin\theta = \sin\left(\phi + \pi\right) \approx -\phi$$

$$\dot{\theta}^2 = \dot{\phi}^2 \approx 0$$

پس از استفاده از جایگذاری معادلات فوق در معادله حالت سیستم غیرخطی، دو معادله خطی حرکت به‌دست می‌آید. توجه داشته باشید که u به‌جای نیروی ورودی F در معادلات جایگزین شده است.

$$\left(I + ml^2\right)\ddot{\phi} - mgl\phi = ml\ddot{x}$$

$$\left(M + m\right)\ddot{x} - ml\ddot{\phi} = u$$

تابع تبدیل Transfer Function

برای به‌دست آوردن توابع انتقال از معادله خطی شده سیستم ابتدا باید تبدیل لاپلاس معادلات سیستم را با فرض شرایط اولیه صفر به‌دست آوریم. نتیجه تبدیل لاپلاس به‌صورت زیر نشان داده‌شده است.

$$\left(I + ml^2\right)\Phi\left(s\right)s^2 - mgl\Phi\left(s\right) = mlX\left(s\right)s^2$$

$$\left(M + m\right)X\left(s\right)s^2 - ml\Phi\left(s\right)s^2 = U\left(s\right)$$

همان طور که می‌دانید تابع تبدیل نشان‌دهنده رابطه بین ورودی خروجی در یک زمان است. برای پیدا کردن اولین تابع تبدیل برای خروجی $\Phi\left(s\right)$ و ورودی $U\left(s\right)$ نیاز است که از معادلات فوق $X\left(s\right)$ را حذف کنیم. حل معادله اول برای $X\left(s\right)$ به این صورت است که:

$$X\left(s\right) = \left[\frac{I + ml^2}{ml} - \frac{g}{s^2}\right]\Phi\left(s\right)$$

سپس $X\left(s\right)$ را در معادله دوم قرار داده تا معادله زیر به‌دست آید:

$$\left(M + m\right)\left[\frac{I + ml^2}{ml} - \frac{g}{s^2}\right]\Phi\left(s\right)s^2 - ml\Phi\left(s\right)s^2 = U\left(s\right)$$

با مرتب کردن معادله فوق تابع تبدیل به‌صورت زیر به‌دست می‌آید:

$$\frac{\Phi(s)}{U(s)} = \frac{\dfrac{ml}{\left[(M+m)\left(I+ml^2\right)-(ml)^2\right]}s^2}{s^4 - \dfrac{(M+m)mgl}{\left[(M+m)\left(I+ml^2\right)-(ml)^2\right]}s^2}$$

از تابع تبدیل بالا می‌توان مشاهده کرد که ۲ قطب و ۲ صفر دقیقاً در مبدأ قرار دارند و این دو می‌توانند لغو شوند و تابع تبدیل در نهایت به شکل زیر می‌شود:

$$P_{pend}(s) = \frac{\Phi(s)}{U(s)} = \frac{\dfrac{ml}{\left[(M+m)\left(I+ml^2\right)-(ml)^2\right]}}{s^2 - \dfrac{(M+m)mgl}{\left[(M+m)\left(I+ml^2\right)-(ml)^2\right]}} \quad \left[\frac{rad}{N}\right]$$

تابع تبدیل دوم که موقعیت گاری را به‌عنوان خروجی در نظر گرفته‌شده را می‌توان به روش مشابه استخراج کرد و نتیجه به‌صورت زیر است:

$$P_{cart}(s) = \frac{X(s)}{U(s)} = \frac{\dfrac{\left(I+ml^2\right)}{\left[(M+m)\left(I+ml^2\right)-(ml)^2\right]}s^2 - \dfrac{mgl}{\left[(M+m)\left(I+ml^2\right)-(ml)^2\right]}}{s^4 - \dfrac{(M+m)mgl}{\left[(M+m)\left(I+ml^2\right)-(ml)^2\right]}s^2} \quad \left[\frac{m}{N}\right]$$

معادلات فضای حالت State-Space

معادلات خطی حرکت نمایش داده‌شده در قسمت قبل را اگر به‌صورت یک سری معادلات دیفرانسیل مرتبه اول مرتب شوند می‌توان به فرم فضای حالت نشان شود. از آنجایی که معادلات خطی هستند آن‌ها را می‌توان به شکل ماتریس های استاندارد زیر نشان داد.

$$\begin{bmatrix}\dot{x}\\ \ddot{x}\\ \dot{\Phi}\\ \ddot{\Phi}\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0\\ 0 & 0 & \dfrac{m^2gl^2}{I(M+m)+Mml^2} & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & \dfrac{mgl(M+m)}{I(M+m)+Mml^2} & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\\ \Phi\\ \dot{\Phi}\end{bmatrix} + \begin{bmatrix}0\\ \dfrac{I+ml^2}{I(M+m)+Mml^2}\\ 0\\ \dfrac{ml}{I(M+m)+Mml^2}\end{bmatrix}u$$

$$y = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\\ \Phi\\ \dot{\Phi}\end{bmatrix} + \begin{bmatrix}0\\ 0\end{bmatrix}u$$

ماتریس C به این دلیل ۲ ردیفه است که هر دو متغیر موقعیت گاری و زاویه آونگ به‌عنوان خروجی سیستم است. به طور خاص، موقعیت گاری المان اول خروجی y و زاویه انحراف آونگ از موضع تعادل المان دوم خروجی y است.

فصل سوم: نمایش در نرم‌افزار MATLAB

در این فصل قصد داریم تابع تبدیل و معادلات فضای حالت را در نرم‌افزار MATLAB نمایش دهیم.

تابع تبدیل Transfer Function

می‌توان توابع انتقال بالا را برای سیستم آونگ معکوس در داخل نرم‌افزار قدرتمند MATLAB با استفاده از دستورات نمـایش داد.

```
M=1;
m=0.1;
I=0;
g=9.8;
l=1;
q=(M+m)*(I+m*l^2)-(m*l)^2;
s=tf('s');
P_cart=(((I+m*l^2)/q)*s^2-(m*g*l/q))/(s^4-((M+m)*m*g*l)*s^2/q);
P_pend=(m*l/q)/(s^2-((M + m)*m*g*l)/q);
sys_tf=[P_cart;P_pend];
inputs={'u'};
outputs={'x';'phi'};
set(sys_tf,'InputName',inputs)
set(sys_tf,'OutputName',outputs)
sys_tf
```

دستور ۱ - نمایش تابع تبدیل در نرم‌افزار MATLAB

اجرای دستور فوق در پنجره فرمان نرم‌افزار MATLAB خروجی زیر را نمایش می‌دهد:

```
Transfer function from input "u" to output...
       0.1 s^2 - 0.98
 x:  -------------------
     0.1 s^4 - 1.078 s^2

            1
 phi:  -----------
       s^2 - 10.78
```

دستور ۲ - نتیجه دستورات اراده شده قبلی

معادلات فضای حالت State-Space

همچنین می‌توانیم سیستم را به‌وسیله معادلات حالت نمایش دهیم. دستورات زیر معادلات فضای حالت را بـرای سیسـتم آونـگ معکوس در داخل نرم‌افزار قدرتمند MATLAB نمایش می‌دهند.

```
M=1;
m=0.1;
I=0;
g=9.8;
l=1;
p=I*(M+m)+M*m*l^2;
```

```
A=[0,1,0,0;0,0,(m^2*g*l^2)/p,0;0,0,0,1;0,0,m*g*l*(M+m)/p,0];
B=[0;(I+m*l^2)/p;0;m*l/p];
C=[1,0,0,0;0,0,1,0];
D=[0;0];
states={'x','x_dot','phi','phi_dot'};
inputs={'u'};
outputs={'x';'phi'};
sys_ss=ss(A,B,C,D,'statename',states,'inputname',inputs,'outputname',output
s)
```

دستور ۳ - نمایش معادلات فضای حالات در نرم‌افزار MATLAB

اجرای دستور فوق در پنجره فرمان نرم‌افزار MATLAB خروجی زیر را نمایش می‌دهد:

```
a =
                x    x_dot      phi  phi_dot
   x            0        1        0        0
   x_dot        0        0     0.98        0
   phi          0        0        0        1
   phi_dot      0        0    10.78        0

b =
             u
   x         0
   x_dot     1
   phi       0
   phi_dot   1

c =
              x    x_dot      phi  phi_dot
   x          1        0        0        0
   phi        0        0        1        0

d =
         u
   x     0
   phi   0

Continuous-time model.
```

دستور ۴ - نتیجه دستورات اراده شده قبلی

مدل فضای حالت فوق را می‌توان به‌صورت تابع تبدیل با استفاده از دستور tf همان طور که در زیر نشان داده‌شده است تبدیل کرد. در مقابل، مدل تابع تبدیل را می‌توان به فرم فضای حالت با استفاده از دستور ss تبدیل کرد.

```
sys_tf = tf(sys_ss)
```

دستور ۵ - تبدیل مدل فضای حالت به‌صورت تابع تبدیل

اجرای دستور فوق در پنجره فرمان نرم‌افزار MATLAB خروجی زیر را نمایش می‌دهد:

```
Transfer function from input "u" to output...
          s^2 + 2.22e-015 s - 9.8
 x:  --------------------------------
     s^4 - 4.441e-016 s^3 - 10.78 s^2

                    1
 phi:  --------------------------
       s^2 - 4.441e-016 s - 10.78
```

دستور ۶ - نتیجه دستورات اراده شده قبلی

با بررسی نتیجه فوق، توجه داشته باشید عباراتی با ضرایب بسیار کوچک در نتیجه وجود دارد. این عبارات در واقع صفر هستند و نشان‌دهنده جمع شدن خطای گرد کردن الگوریتم نرم‌افزار MATLAB می‌باشند. اگر این عبارات را صفر در نظر بگیریم به همان تابع تبدیل به‌دست آمده در قسمت های قبل می‌رسیم و کاملاً مطابقت دارد.

فصل چهارم: تجزیه‌وتحلیل سیستم

از مسئله آونگ معکوس توابع تبدیل حلقه باز زیر مشتق شده‌اند.

$$P_{pend}(s)=\frac{\Phi(s)}{U(s)}=\frac{\frac{ml}{\left[(M+m)\left(I+ml^2\right)-(ml)^2\right]}}{s^2-\frac{(M+m)mgl}{\left[(M+m)\left(I+ml^2\right)-(ml)^2\right]}} \quad \left[\frac{rad}{N}\right]$$

$$P_{cart}(s)=\frac{X(s)}{U(s)}=\frac{\frac{\left(I+ml^2\right)}{\left[(M+m)\left(I+ml^2\right)-(ml)^2\right]}s^2-\frac{mgl}{\left[(M+m)\left(I+ml^2\right)-(ml)^2\right]}}{s^4-\frac{(M+m)mgl}{\left[(M+m)\left(I+ml^2\right)-(ml)^2\right]}s^2} \quad \left[\frac{m}{N}\right]$$

از فصول قبل به یاد دارید که توابع تبدیل فوق زمانی معتبر هستند که مقدار زاویه انحراف آونگ از حالت عمودی ϕ کمتر از ۲۰ درجه باشد و همچنین زاویه مطلق آونگ θ برابر $\pi+\phi$ است.

با توجه به پاسخ سیستم آونگ معکوس به یک ضربه‌ای که به گاری اعمال می‌شود و برابر $1\ \text{N.sec}$ است، الزامات طراحی برای آونگ عبارت‌اند از:

- زمان نشست برای زاویه θ کمتر از $5\ \text{sec}$ باشد.
- زاویه انحراف آونگ ϕ هیچ‌گاه بیشتر از $0.5\ \text{rad}$ نباشد.

علاوه بر این، شرایط مورد نیاز برای ورودی پله $0.2\ \text{m}$ به موقعیت گاری به‌صورت زیر است:

- زمان نشست برای θ و x کمتر از $5\ \text{sec}$ باشد.
- زمان صعود برای x کمتر از $0.5\ \text{sec}$ باشد.
- زاویه انحراف آونگ از $20\ \text{deg}$ یا $0.35\ \text{rad}$ بیشتر نباشد.

پاسخ ضربه سیستم حلقه باز

ابتدا به پاسخ حلقه باز سیستم آونگ معکوس توجه می‌کنیم. یک m-file جدید در نرم‌افزار MATLAB ایجاد کرده و از دستور زیر استفاده می‌کنیم تا مدل سیستم را شبیه‌سازی کنیم.

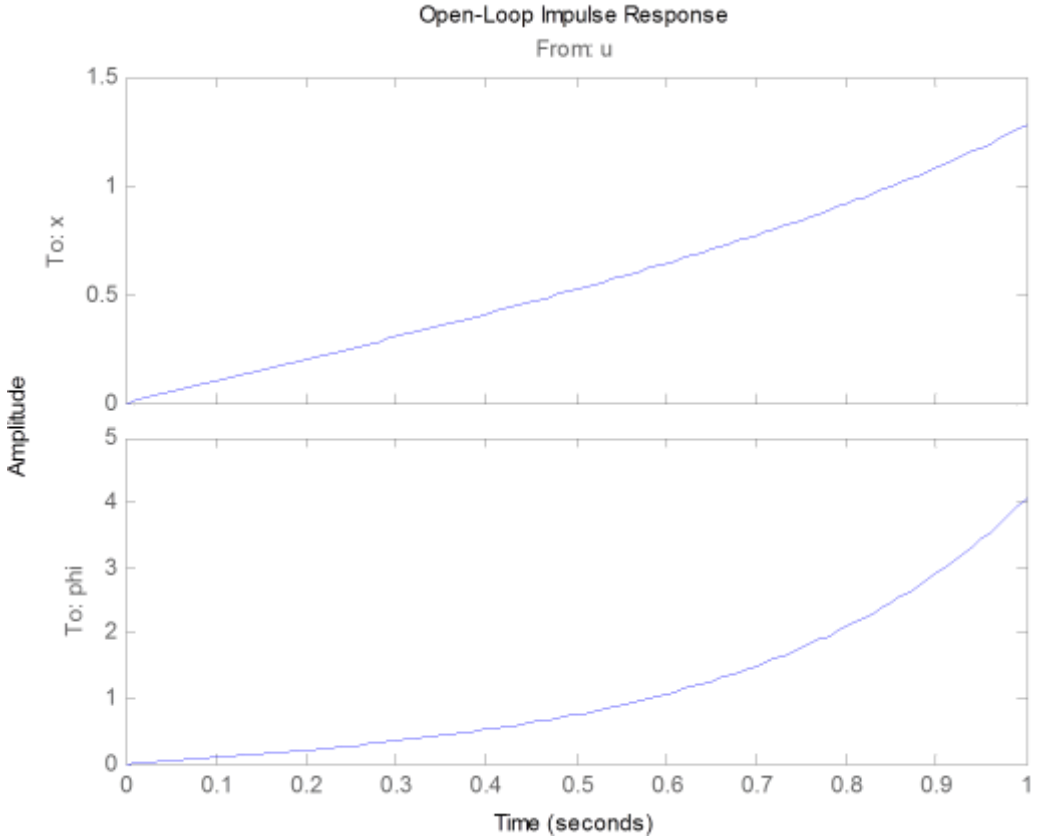
```
M=1;
m=0.1;
I=0;
g=9.8;
l=1;
q=(M+m)*(I+m*l^2)-(m*l)^2;
s=tf('s');
P_cart=(((I+m*l^2)/q)*s^2-(m*g*l/q))/(s^4-((M+m)*m*g*l)*s^2/q);
P_pend=(m*l/q)/(s^2-((M + m)*m*g*l)/q);
sys_tf=[P_cart;P_pend];
```

```
inputs={'u'};
outputs={'x';'phi'};
set(sys_tf,'InputName',inputs)
set(sys_tf,'OutputName',outputs)
```

حال می‌توانیم پاسخ سیستم حلقه باز به ورودی ضربه را بررسی کنیم. به طور خاص، به بررسی چگونگی پاسخ به یک نیروی ضربه به گاری را با دستور زیر در MATLAB بررسی می‌کنیم.

```
t=0:0.01:1;
impulse(sys_tf,t);
title('Open-Loop Impulse Response')
```

با اضافه کردن دستور فوق به دستورات قبلی و اجرا کردن آن‌ها در نرم‌افزار به نمودار زیر دست می‌یابیم.



با بررسی این دو نمودار می‌توان به این نتیجه رسید که پاسخ سیستم به طور کامل غیرقابل‌قبول است. در واقع، این سیستم به‌صورت حلقه باز ناپایدار است. بااین‌وجود افزایش زاویه آونگ به‌اندازه 4 rad است که مدل‌سازی فقط برای زاویه‌های کوچک ϕ انجام‌شده و پاسخ برای مقادیر زیاد آن دقیق نیست. همچنین از نمودار مربوط به موقعیت گاری می‌توان به این نتیجه رسید که موقعیت آن در اثر ضربه بدون تغییر به بینهایت می‌رود.

قطب یک سیستم می‌تواند در مورد پاسخ زمانی یک سیستم اطلاعات خوبی دهد. ازآنجاکه این سیستم دو خروجی و یک ورودی دارد با ۲ تابع تبدیل توضیح داده می‌شود. به طور کلی، تمام توابع تبدیل در سیستم های چند ورودی و چند خروجی MIMO دارای قطب یکسان‌اند (صفر های متفاوت هستند) مگر اینکه صفری باعث لغو قطب گردیده باشد. به طور خاص با استفاده از دستور zpkdata نرم‌افزار MATLAB به استخراج قطب و صفر سیستم می‌پردازیم. پارامتر 'v' نشان داده‌شده در دستور زیر مقادیر قطب و صفر را به‌صورت نمودار نشان می‌دهد و نه به‌صورت آرایه‌ای.

```
[zeros poles] = zpkdata(P_pend,'v')
```

که نتیجه دستور فوق به‌صورت زیر است:

```
zeros =

    3.1305
```

```
   -3.1305


poles =

         0
         0
    3.2833
   -3.2833
```

و همین ترتیب، صفر و قطب سیستم برای تابع تبدیلی که خروجی آن موقعیت گاری است از دستور زیر استفاده می‌کنیم:

```
[zeros poles] = zpkdata(P_cart,'v')
```

که نتیجه دستور فوق به‌صورت زیر است:

```
zeros =

    3.1305
   -3.1305


poles =

         0
         0
    3.2833
   -3.2833
```

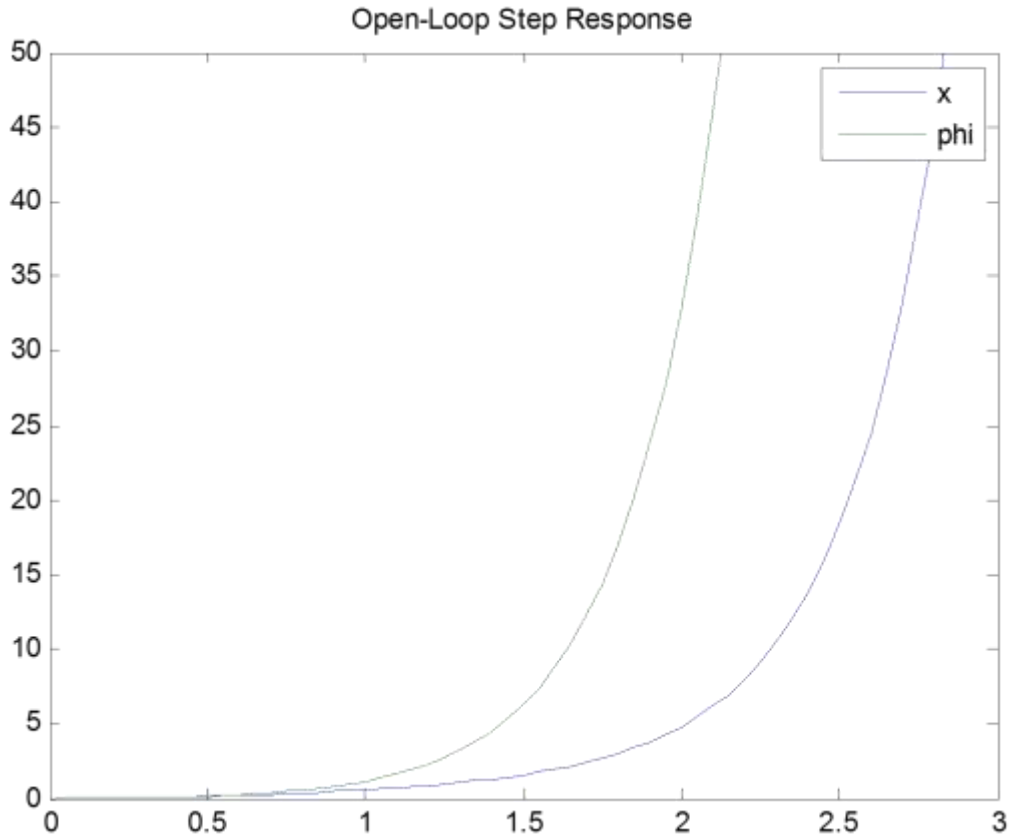
همان طور که پیش‌بینی‌شده بود قطب هر دو تابع تبدیل یکسان است. از قطب ۳,۲۸۳۳ به دلیل داشتن مقدار حقیقی مثبت، متوجه می‌شویم سیستم ناپایدار است. به‌عبارت‌دیگر، این قطب در سمت راست صفحه مختلط s-plane قرار دارد. این موضوع دقیقاً روند ناپایداری را در نمودار های فوق توضیح می‌دهد.

پاسخ پله سیستم حلقه باز

ازآنجاکه سیستم دارای یک قطب با مقدار حقیقی مثبت است پاسخ به ورودی پله آن نیز ناپایدار است. به‌وسیله دستور lsim می‌توان پاسخ سیستم LTI را به ورودی دلخواه شبیه‌سازی کرد. در این مورد، ورودی پله را به‌اندازه 1 Newton استفاده شده است.

```
t = 0:0.05:10;
u = ones(size(t));
[y,t] = lsim(sys_tf,u,t);
plot(t,y)
title('Open-Loop Step Response')
axis([0 3 0 50])
legend('x','phi')
```

اضافه کردن دستور فوق به m-file و اجزای آن نمودار زیر را نتیجه می‌دهد.



همچنین می‌توانید برخی از ویژگی های پاسخ را با استفاده از دستور lsiminfo تشخیص داد.

```
step_info = lsiminfo(y,t);
cart_info = step_info(1)
pend_info = step_info(2)
```

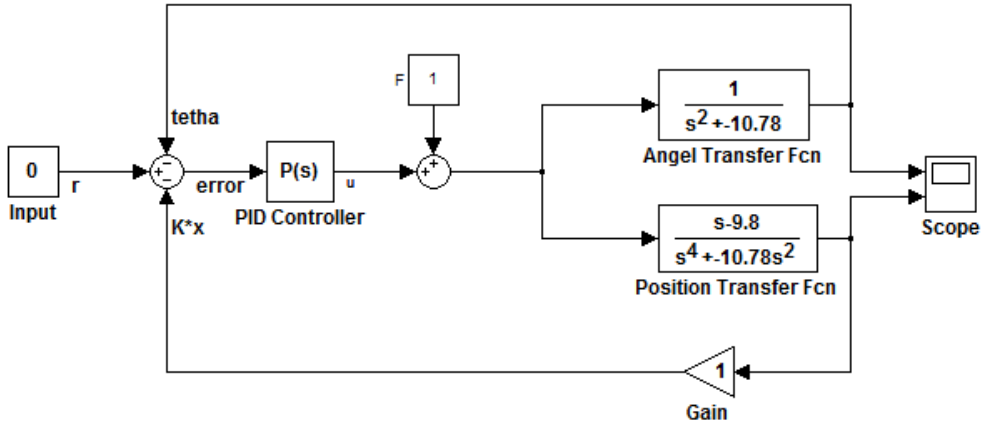
که نتیجه دستور فوق به‌صورت زیر است.

```
cart_info =

    SettlingTime: 9.9934
             Min: 0
         MinTime: 0
             Max: 7.6579e+011
         MaxTime: 10


pend_info =

    SettlingTime: 9.9934
             Min: 0
         MinTime: 0
             Max: 8.4237e+012
         MaxTime: 10
```

نتایج فوق تأیید کننده آن است که سیستم به ورودی پله ناپایدار است. از تجزیه‌وتحلیل های فوق این‌گونه استنباط می‌شود که به کنترلی برای بهبود پاسخ سیستم نیاز است.

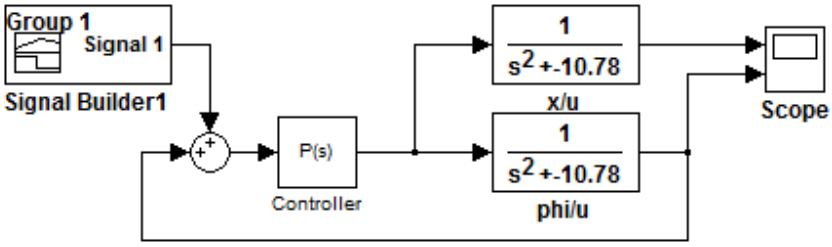
فصل پنجم: طراحی کنترل‌کننده

در سؤال مطرح‌شده از طرف دکتر محمدجعفر صدیق خواسته‌شده کنترلری برای سیستم زیر طراحی شود تا سیستم پایدار شود و پاسخ مطلوبی داشته باشد. علاوه بر این جمع‌کردن سیگنال های ϕ و x به‌کلی غلط است زیرا این دو سیگنال از جنس های متفاوت زاویه و موقعیت بر حسب رادیان و متر هستند علاوه بر این سیستم چند ورودی و چند خروجی است و امکان طراحی کنترل گر بر مبنای کنترل کلاسیک نیست زیرا زمانی که شروع به طراحی می‌کنیم نمی‌دانیم که کدام خروجی را برای محاسبه تابع تبدیل حلقه بسته ملاک قرار دهیم با این‌وجودبرای این سیستم می‌توان یک کنترل PID طراحی می‌کنیم. البته در این بخش از جبرانساز استفاده نشده و فقط با روش صفر–قطب حذف شونده سیستم را پایدار کرده و نتیجه را بررسی می‌کنیم.



طراحی کنترل‌کننده با K=0 مربوط به دینامیک سیستم

با $k=0$ سیستم به‌صورت زیر ساده می‌شود.



پس تابع تبدیل ورودی نیرو به خروجی زاویه آونگ با استفاده از دستور زیر به‌دست می‌آید.

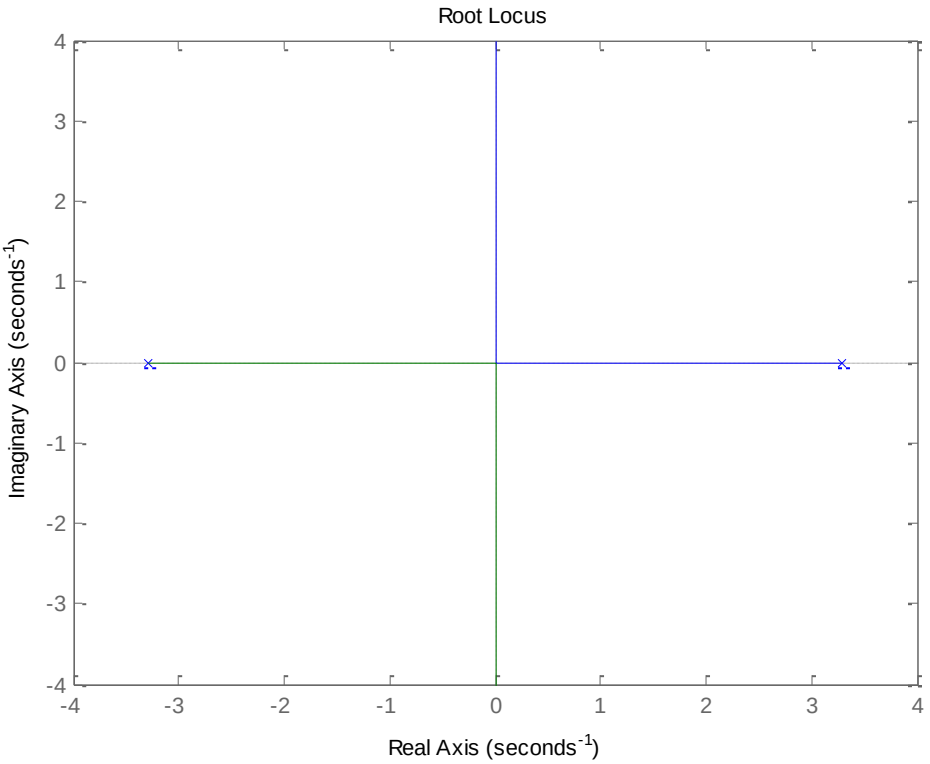
```
sys=tf([1],[1 0 -10.78]);
inputs={'u'};
outputs={'K*x+phi'};
set(sys,'InputName',inputs)
set(sys,'OutputName',outputs)
sys
rlocus(sys)
```

که نتیجه دستور فوق به‌صورت زیر است.

```
Transfer function from input "u" to output "K*x+phi":
```

```
      1
-----------
s^2 - 10.78
```

و نمودار مکان هندسی ریشه‌ها به‌صورت زیر است.



و با استفاده از دستور زیر مقادیر قطب و صفر به‌دست می‌آید.

```
[zeros poles]=zpkdata(sys,'v')
```

که نتیجه دستور فوق به‌صورت زیر است.

```
zeros =

   Empty matrix: 0-by-1


poles =

    3.2833
   -3.2833
```

همان طور که مشاهده می‌کنید ۱ قطب سمت راست محور اعداد حقیقی قرار دارد و سیستم را ناپایدار می‌سازد.

و در کنترلر یک قطب در ۴- قرار می‌دهیم و یک صفر در ۳,۲۸۳۳ قرار می‌دهیم.

```
sysc=zpk([(10.78)^0.5],[-4],1);
tf(sysc)
```

که نتیجه دستور فوق به‌صورت زیر است.

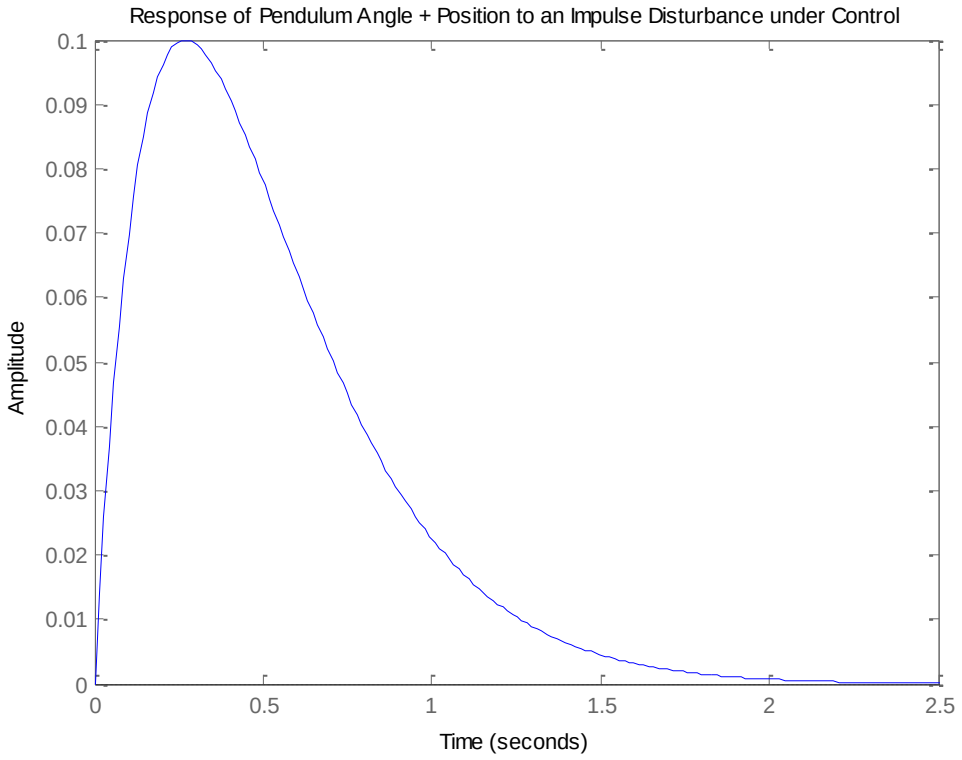
```
Transfer function:
s - 3.283
---------
  s + 4
```

و به‌صورت زیر کارایی سیستم را بررسی می‌کنیم.

```
openloop=zpk([],[-3.2833,-4],1);
rlocus(openloop);
syst=tf(openloop)
```

```
T=syst/(1+syst);
impulse(T)
title('Response of Pendulum Angle + Position to an Impulse Disturbance
under Control');
```

و نتیجه به‌صورت زیر است.



البته می‌توان با افزایش ضریب کنترلر آن را بهبود بخشید.

طراحی کنترل‌کننده با K=1 مربوط به دینامیک سیستم

تابع تبدیل بخش های این آونگ معکوس در فصل قبل به‌دست آمده و به‌صورت زیر است.

$$P_{pend}(s)=\frac{\Phi(s)}{U(s)}=\frac{\frac{ml}{\left[(M+m)\left(I+ml^2\right)-(ml)^2\right]}}{s^2-\frac{(M+m)mgl}{\left[(M+m)\left(I+ml^2\right)-(ml)^2\right]}} \quad \left[\frac{rad}{N}\right]$$

$$P_{cart}(s)=\frac{X(s)}{U(s)}=\frac{\frac{\left(I+ml^2\right)}{\left[(M+m)\left(I+ml^2\right)-(ml)^2\right]}s^2-\frac{mgl}{\left[(M+m)\left(I+ml^2\right)-(ml)^2\right]}}{s^4-\frac{(M+m)mgl}{\left[(M+m)\left(I+ml^2\right)-(ml)^2\right]}s^2} \quad \left[\frac{m}{N}\right]$$

لذا تابع تبدیل موقعیت گاری به زاویه آونگ به‌صورت زیر است.

$$X(s)=\left[\frac{I+ml^2}{ml}-\frac{g}{s^2}\right]\Phi(s)\rightarrow\frac{\Phi(s)}{X(s)}=\left[\frac{I+ml^2}{ml}-\frac{g}{s^2}\right]=\frac{s^2-9.8}{s^2}$$

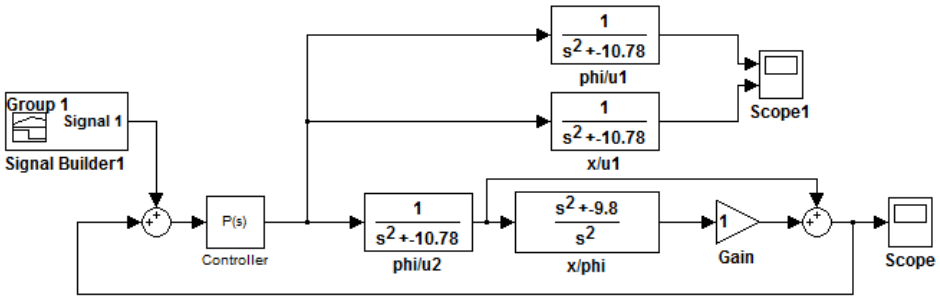
که با استفاده از نرم‌افزار MATLAB به‌صورت زیر نمایش داده می‌شود.

```
sys=tf([1 0 0],[1 0 -9.8]);
inputs={'phi'};
outputs={'x'};
set(sys,'InputName',inputs)
set(sys,'OutputName',outputs)
sys
```

نتیجه دستور فوق به‌صورت زیر است.

```
Transfer function from input "phi" to output "x":
   s^2
---------
s^2 - 9.8
```

پس شکل سیستم کنترلی به‌صورت زیر بازسازی می‌شود



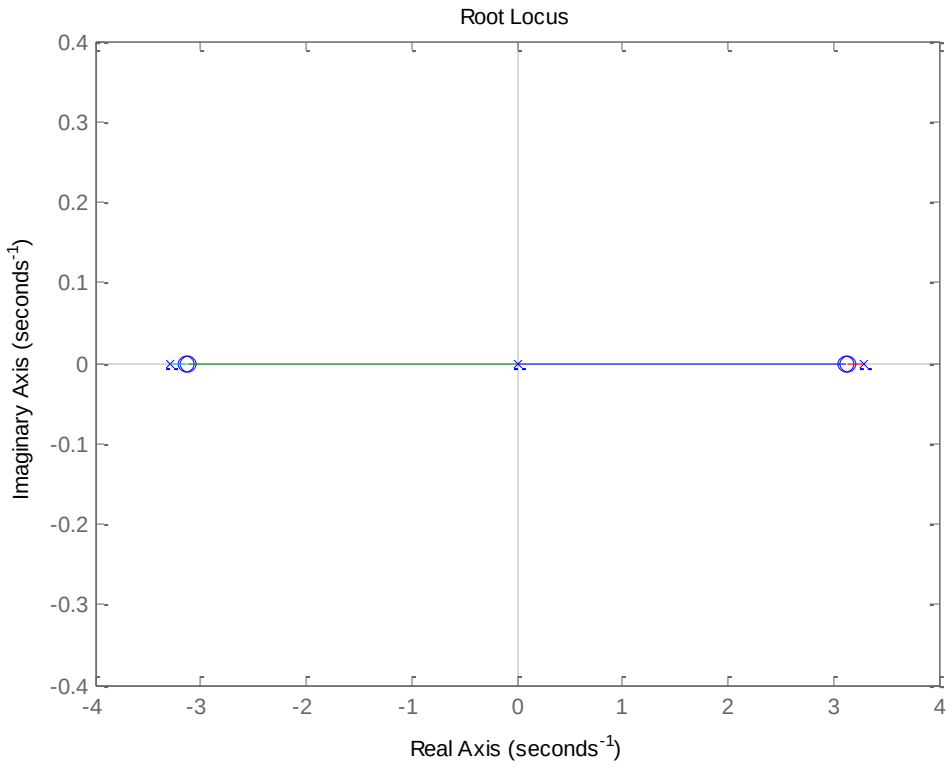
پس تابع تبدیل ورودی نیرو به خروجی زاویه آونگ به‌علاوه موقعیت گاری با استفاده از دستور زیر به‌دست می‌آید.

```
K=1;
sys1=tf([1 0 -9.8],[1 0 0]);
sys2=tf([1],[1 0 -10.78]);
sys=K*sys1*sys2+sys1;
inputs={'u'};
outputs={'K*x+phi'};
set(sys,'InputName',inputs)
set(sys,'OutputName',outputs)
sys
rlocus(sys)
```

که نتیجه دستور فوق به‌صورت زیر است.

```
Transfer function from input "u" to output "K*x+phi":
s^6 - 19.58 s^4 + 95.84 s^2
---------------------------
      s^6 - 10.78 s^4
```

و نمودار مکان هندسی ریشه‌ها به‌صورت زیر است.



و با استفاده از دستور زیر مقادیر قطب و صفر به‌دست می‌آید.

```
[zeros poles]=zpkdata(sys,'v')
```

که نتیجه دستور فوق به‌صورت زیر است.

```
zeros =

         0
         0
   -3.1305
   -3.1273
    3.1305
    3.1273


poles =

         0
         0
         0
         0
    3.2833
   -3.2833
```

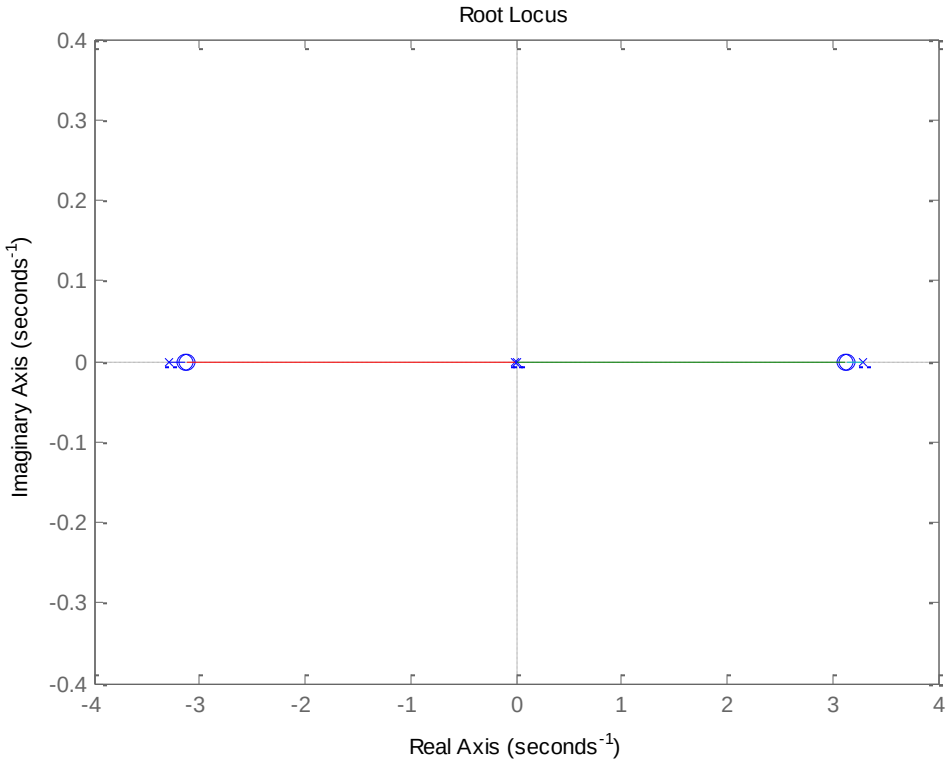
همان طور که مشاهده می‌کنید ۲ صفر و ۱ قطب سمت راست محور اعداد حقیقی قرار دارد و سیستم را ناپایدار می‌سازد و به دلیل وجود صفر امکان طراحی کنترلر نیست. و سیستم ساده‌شده به‌صورت زیر است.

```
sys=zpk([-3.1305,-3.1273,3.1305,3.1273],[0,0,3.2833,-3.2833],1);
tf(sys)
rlocus(sys)
```

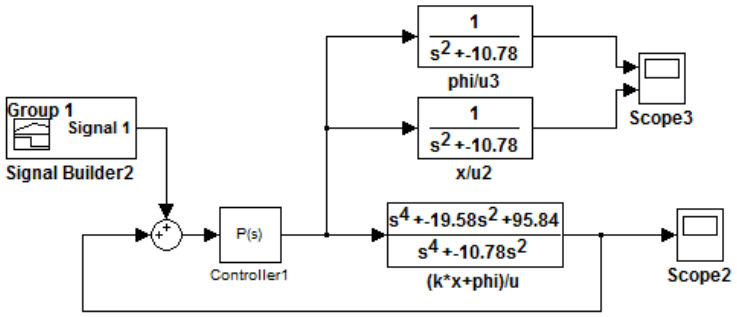
که نتیجه آن به‌صورت زیر است.

```
Transfer function:
s^4 - 19.58 s^2 + 95.84
-----------------------------------------
            s^4 - 10.78 s^2
```

و نمودار مکان هندسی ریشه‌ها به‌صورت زیر است.



همان طور که مشاهده می‌کنید باز هم یک صفر سمت راست دارد و سیستم کنترل ناپذیر است ولی می‌توان در کنترلر یک قطب در آن نقطه قرار داد تا اثر آن را خنثی کند. به‌هرحال سیستم ساده‌شده به‌صورت زیر است.



و در کنترلر یک قطب در ۳,۱۲۷۳ قرار می‌دهیم و یک صفر در ۳,۲۸۳۳ قرار می‌دهیم.

```
sysc=zpk([3.2833],[ 3.1305,3.1273],0.05);
tf(sysc)
```

که نتیجه دستور فوق به‌صورت زیر است.

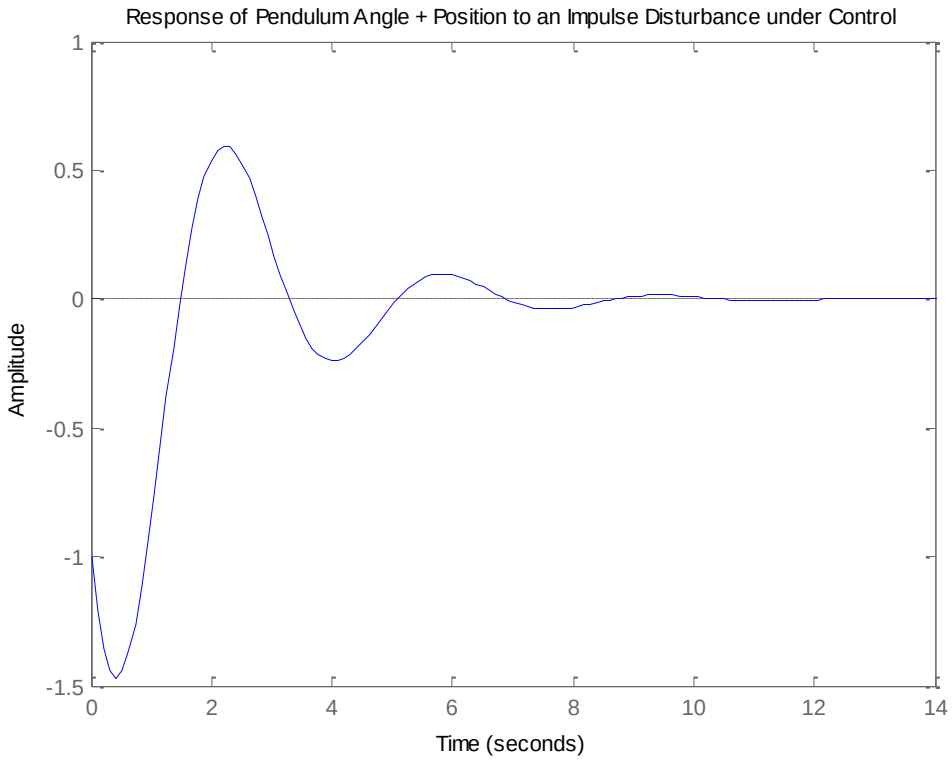
```
Transfer function:
  0.05 s - 0.1642
--------------------
s^2 - 6.258 s + 9.79
```

و به‌صورت زیر کارایی سیستم را بررسی می‌کنیم.

```
sys=zpk([0,0,-3.1273,3.1273],[-3.2833,-3.1305,3.2833],1);
sysc=zpk([3.2833],[3.1305],1);
openloop=zpk([0,0,-3.1273],[-3.2833,-3.1305],1);
rlocus(openloop);
syst=tf(openloop)
T=syst/(1+syst);
impulse(T)
```

```
title('Response of Pendulum Angle + Position to an Impulse Disturbance
under Control');
```

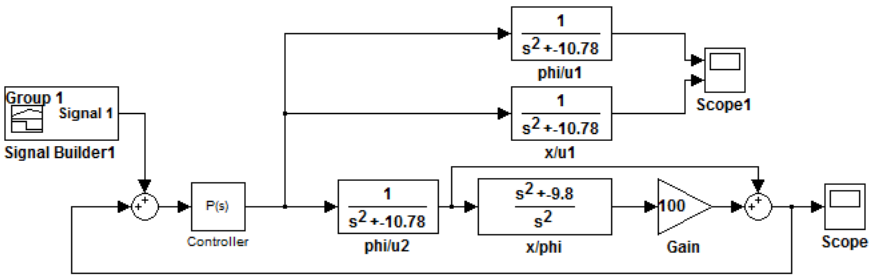
و نتیجه به‌صورت زیر است.



البته می‌توان با افزایش ضریب کنترلر آن را بهبود بخشید.

طراحی کنترل‌کننده با K=100 مربوط به دینامیک سیستم

شکل سیستم کنترلی به‌صورت زیر بازسازی می‌شود.



پس تابع تبدیل ورودی نیرو به خروجی زاویه آونگ به‌علاوه موقعیت گاری با استفاده از دستور زیر به‌دست می‌آید.

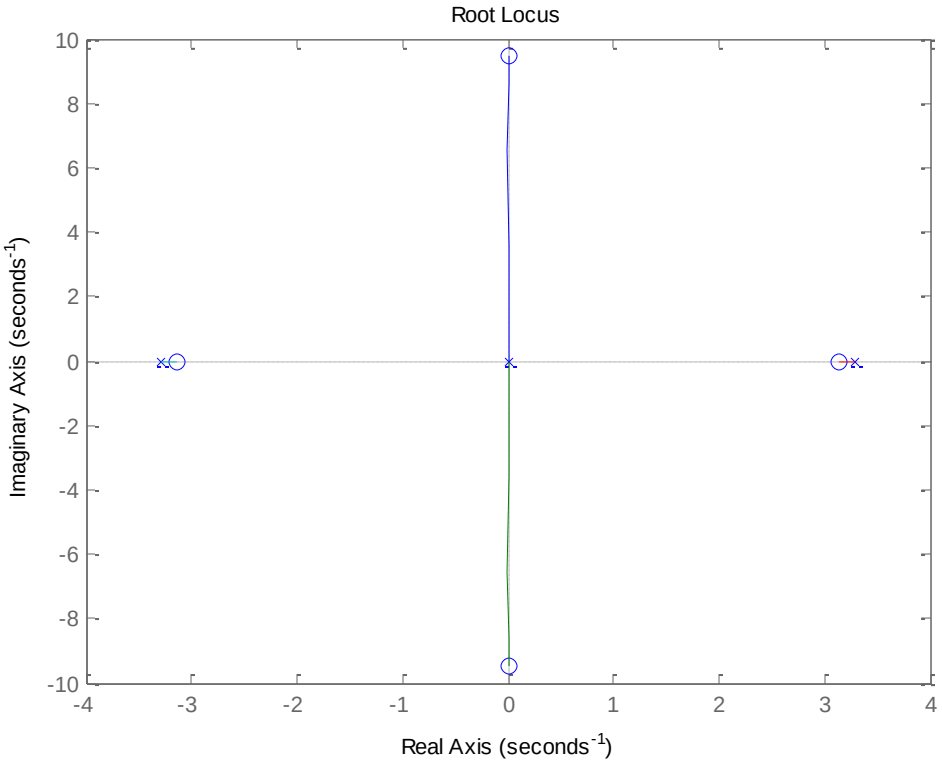
```
K=100;
sys1=tf([1 0 -9.8],[1 0 0]);
sys2=tf([1],[1 0 -10.78]);
sys=K*sys1*sys2+sys1;
inputs={'u'};
outputs={'K*x+phi'};
set(sys,'InputName',inputs)
set(sys,'OutputName',outputs)
sys
```

```
rlocus(sys)
```

که نتیجه دستور فوق به‌صورت زیر است.

```
Transfer function from input "u" to output "K*x+phi":
s^6 + 79.42 s^4 - 874.4 s^2
---------------------------
      s^6 - 10.78 s^4
```

و نمودار مکان هندسی ریشه‌ها به‌صورت زیر است.



و با استفاده از دستور زیر مقادیر قطب و صفر به‌دست می‌آید.

```
[zeros poles]=zpkdata(sys,'v')
```

که نتیجه دستور فوق به‌صورت زیر است.

```
zeros =

         0
         0
   0.0000 + 9.4456i
   0.0000 - 9.4456i
  -3.1305
   3.1305


poles =

          0
          0
          0
          0
    3.2833
   -3.2833
```

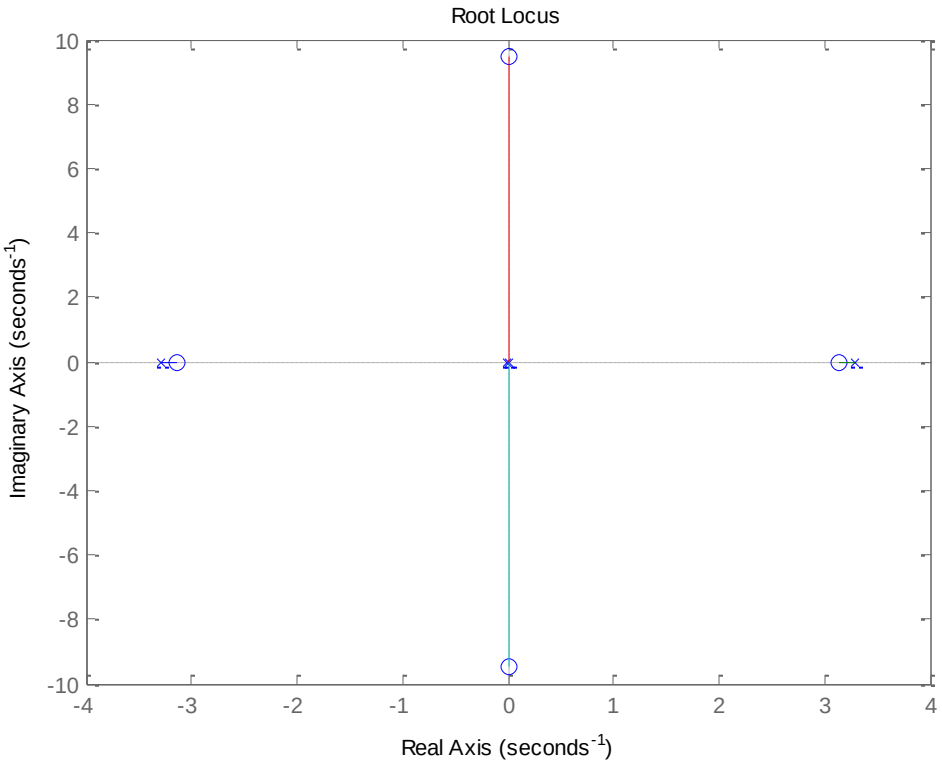
همان طور که مشاهده می‌کنید 1 صفر و ۱ قطب سمت راست محور اعداد حقیقی قرار دارد و سیستم را ناپایدار می‌سازد و به دلیل وجود صفر امکان طراحی کنترلر نیست. و سیستم ساده‌شده به‌صورت زیر است.

```
sys=zpk([9.4456i,-9.4456i,-3.1305,3.1305],[0,0,3.2833,-3.2833],1);
tf(sys)
rlocus(sys)
```

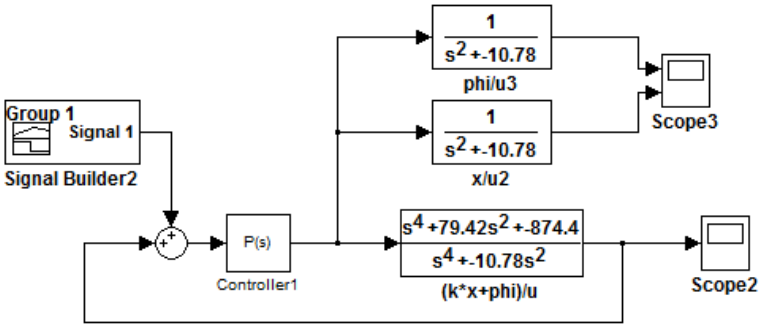
که نتیجه آن به‌صورت زیر است.

```
Transfer function:
s^4 + 79.42 s^2 - 874.4
-----------------------
    s^4 - 10.78 s^2
```

و نمودار مکان هندسی ریشه‌ها به‌صورت زیر است.



همان طور که مشاهده می‌کنید باز هم یک صفر سمت راست دارد و سیستم کنترل ناپذیر است ولی می‌توان در کنترلر یک قطب در آن نقطه قرار داد تا اثر آن را خنثی کند. به‌هرحال سیستم ساده‌شده به‌صورت زیر است.



و در کنترلر یک قطب در ۳,۱۳۰۵ قرار می‌دهیم و یک صفر در ۳,۲۸۳۳قرار می‌دهیم.

```
sysc=zpk([3.2833],[3.1305],1);
tf(sysc)
```

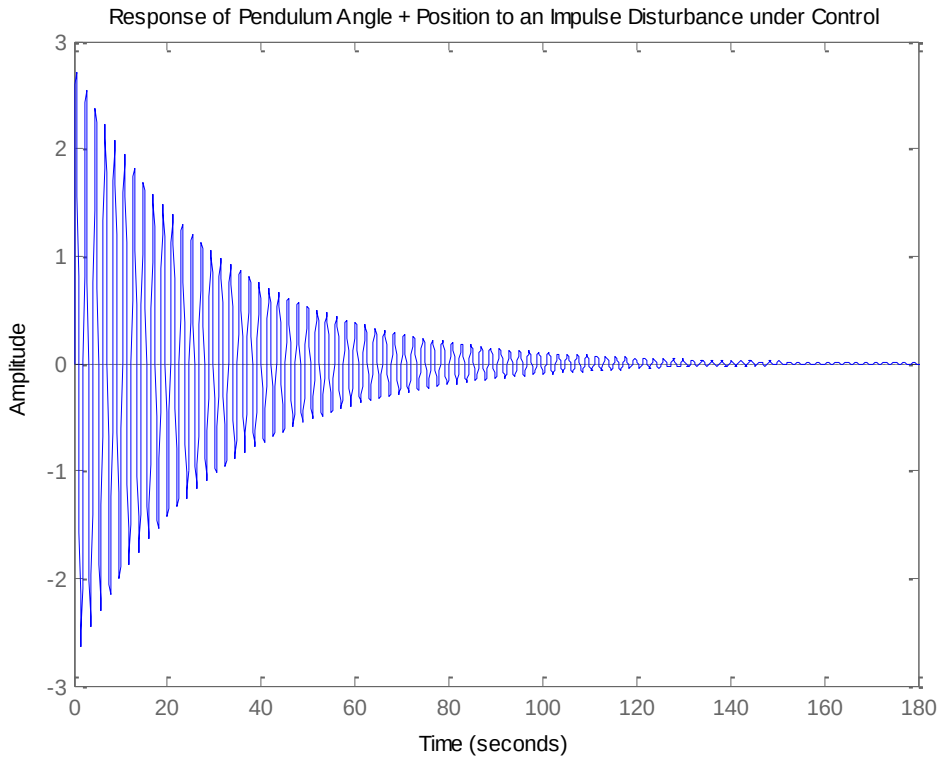
که نتیجه دستور فوق به‌صورت زیر است.

```
Transfer function:
s - 3.283
---------
s - 3.131
```

و به‌صورت زیر کارایی سیستم را بررسی می‌کنیم.

```
openloop=zpk([9.4456i,-9.4456i,-3.1305],[0,0,-3.2833],0.12);
rlocus(openloop);
syst=tf(openloop)
T=syst/(1+syst);
impulse(T)
title('Response of Pendulum Angle + Position to an Impulse Disturbance
under Control');
```

و نتیجه به‌صورت زیر است.



البته می‌توان با تغییر کنترلر آن را بهبود بخشید و یا حتی از جبران ساز های پیشرو و پسرو استفاده کنیم. به علت سنگین بودن پروژه از بهبود این سیستم صرف‌نظر می‌کنیم.

فصل ششم: روش فضای حالت برای طراحی کنترل

از مسئله اصلی، معادلات دینامیک سیستم آونگ معکوس به فرم فضای حالت به‌صورت زیر شرح داده‌شده است.

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\Phi} \\ \ddot{\Phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0.98 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 10.78 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \Phi \\ \dot{\Phi} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} u$$

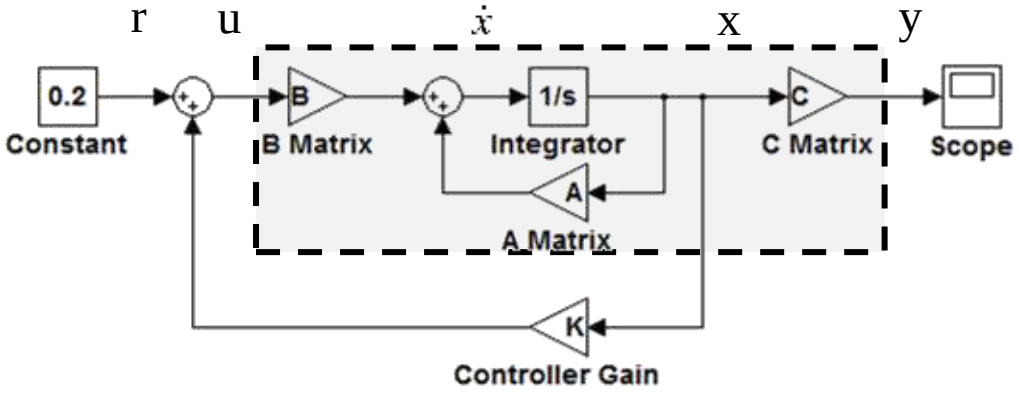
$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \Phi \\ \dot{\Phi} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u$$

معیار های طراحی برای انتقال موقعیت گاری x به‌اندازه 0.2 m به‌صورت تابع پله به‌صورت زیر است:

- زمان نشست برای x و θ کمتر از 5 sec باشد.
- زمان صعود برای x کمتر از 0.5 sec باشد.
- زاویه انحراف آونگ ϕ نباید بیشتر از 20 deg یا 0.35 rad از حالت عمودی آن باشد.
- خطای حالت ماندگار کمتر از 2% برای x و θ باشد.

در این بخش قصد داریم تا با طراحی کنترلر مناسب آونگ را به‌صورت عمودی نگه‌داریم درصورتی‌که موقعیت گاری به‌اندازه 0.2 m به سمت راست حرکت کرده است. رویکرد طراحی در فضای حالت به این صورت است که به خوبی برای سیستم های چند خروجی کنترلر طراحی می‌کند.

این مسئله را می‌توان با استفاده از بازخورد حالت کامل حل کرد. طرح کلی این نوع سیستم کنترل در شکل زیر نشان داده‌شده است و $\tilde{K}$ ماتریس ضرایب کنترلی است. توجه داشته باشید که تمام متغیر های حالت را به‌جای استفاده از خروجی سیستم بازخورد دادیم.



قطب های حلقه باز Open-loop poles

در این مسئله r نشان‌دهنده ورودی پله به موقعیت گاری است. متغیر حالت تشکیل‌شده از موقعیت و سرعت گاری و زاویه و سرعت زاویه آونگ است. خروجی y نشان‌دهنده موقعیت گاری و زاویه آونگ است. قصد داریم یک کنترلی طراحی کنیم که درصورتی‌که یک ورودی پله به سیستم وارد شد و آونگ تغییر زاویه داد در نهایت کنترلر آن را به حالت عمودی برگرداند و گاری به موقعیت جدید برود.

اولین گام برای طراحی کنترلر از بازخورد حالت کامل تعیین قطب های حلقه باز سیستم است. با واردکردن دستور زیر در یک m-file می‌توان قطب های حلقه باز را از مقادیر ویژه‌ماتریس A به‌دست آورد.

```
M=1;
m=0.1;
I=0;
g=9.8;
l=1;
p=I*(M+m)+M*m*l^2;
A=[0,1,0,0;0,0,(m^2*g*l^2)/p,0;0,0,0,1;0,0,m*g*l*(M+m)/p,0];
B=[0;(I+m*l^2)/p;0;m*l/p];
C=[1,0,0,0;0,0,1,0];
D=[0;0];
states={'x','x_dot','phi','phi_dot'};
inputs={'u'};
outputs={'x';'phi'};
sys_ss=ss(A,B,C,D,'statename',states,'inputname',inputs,'outputname',output
s);
poles = eig(A)
```

که نتیجه دستور فوق به‌صورت زیر می‌شود.

```
poles =

         0
         0
    3.2833
   -3.2833
```

همان طور که ملاحظه می‌کنید قطب ۳,۲۸۳۳ در نیمه سمت راست صفحه قرار دارد و نشان‌دهنده این است که سیستم حلقه باز ناپایدار است.

روش مقررات درجه دوم خطی (LQR)

مرحله بعدی در روند طراحی پیدا کردن بردار ضرایب کنترلی $\underset{\sim}{K}$ با فرض اندازه‌گیری کامل متغیر های حالت است. پیـدا کـردن بردار ضرایب کنترلی به روش های مختلفی قابل انجام است. درصورتی‌که محل قطب حلقه باز مطلوب مشخص باشد می‌توان از دستورات place و یا acker در نرم‌افزار MATLAB استفاده کرد. روشی دیگر این است که از دستور lqr استفاده شود که ضرایب کنترلی بهینه‌شده مطلوبی را با توجه به سیستم خطی، تابع درجه دو، و مراجع برابر صفر ارائه می‌دهد.

قبل از طراحی کنترلر به بررسی کنترل‌پذیری سیستم پرداخته می‌شود. وجود کنترل‌پذیری نشان‌دهنده آن است کـه متغیـر هـای حالت را در هر مکانی و در هر زمانی که می‌خواهیم ببریم. برای سیستم کنترل‌پذیر کامل باید ماتریس کنترل‌پـذیری دارای رتبـه n باشد به این معنی که دارای تعداد ردیف (یا ستون) مستقل باشد. ماتریس کنترل‌پذیری به‌صورت زیر به‌دست می‌آید. که تعـداد n مربوط به تعداد متغیرهای حالت است.

$$C=\left[A|AB|A^2B|...|A^{n-1}B\right]$$

از آن جایی که ماتریس کنترل‌پذیری برای سیستم ما 4×4 است باید رتبه ماتریس ۴ باشد. با استفاده از دستور ctrb نرم‌افـزار MATLAB می‌توان ماتریس کنترل‌پذیری سیستم را استخراج نمود و با استفاده از دستور rank می‌توان مرتبـه مـاتریس را بررسی کرد.

```
co=ctrb(sys_ss)
controllability=rank(co)
```

با اضافه کردن دستور فوق به دستورات قبلی در پنجره فرمان نرم‌افزار MATLAB خروجی زیر تولید می‌شود.

```
co =

         0    1.0000         0    0.9800
    1.0000         0    0.9800         0
```

```
         0    1.0000         0   10.7800
    1.0000         0   10.7800         0


controllability =

     4
```

بنابراین متوجه شدیم که سیستم کنترل‌پذیر است و در نتیجه ما باید قادر به طراحی یک کنترلر برای دستیابی به سیستم پایدار هستیم. به طور خاص با استفاده از روش LQR برای تعیین ماتریس ضرایب کنترلی $\underset{\sim}{K}$ استفاده می‌کنیم. تابع lqr نرم‌افزار MATLAB به کاربر اجازه می‌دهد که دو پارامتر R و Q را انتخاب کنیم که متعادل‌کننده اهمیت کنترل u و خطا e (انحراف از ۰) است که به ترتیب در تابع بهینه‌سازی است. ساده‌ترین حالت آن است که فرض کنیم $R=1$ و $Q=C'C$ باشد که نشان‌دهنده این است که کنترل زاویه آونگ و موقعیت گاری به یک میزان اهمیت دارد. برای مشاهده ساختار Q می‌توان از دستور زیر در نرم‌افزار MATLAB استفاده کرد.

```
Q=C'*C
```

که نتیجه استفاده از دستور فوق به‌صورت زیر است.

```
Q =

     1     0     0     0
     0     0     0     0
     0     0     1     0
     0     0     0     0
```

المان ردیف اول و ستون اول نشان‌دهنده وزن کنترل موقعیت گاری و المان ردیف سوم و ستون سوم نشان‌دهنده وزن کنترل زاویه آونگ در طراحی کنترلر بهینه است. و در نهایت ماتریس R بیان‌کننده اهمیت ارزش نسبی R و Q نسبت به هم است و ارزش مطلق آنها نیست. حال با استفاده از دو ماتریس فوق به پیدا کردن تعیین ماتریس ضرایب کنترلی $\underset{\sim}{K}$ می‌پردازیم.

```
Q=C'*C;
R=1;
K=lqr(A,B,Q,R)
```

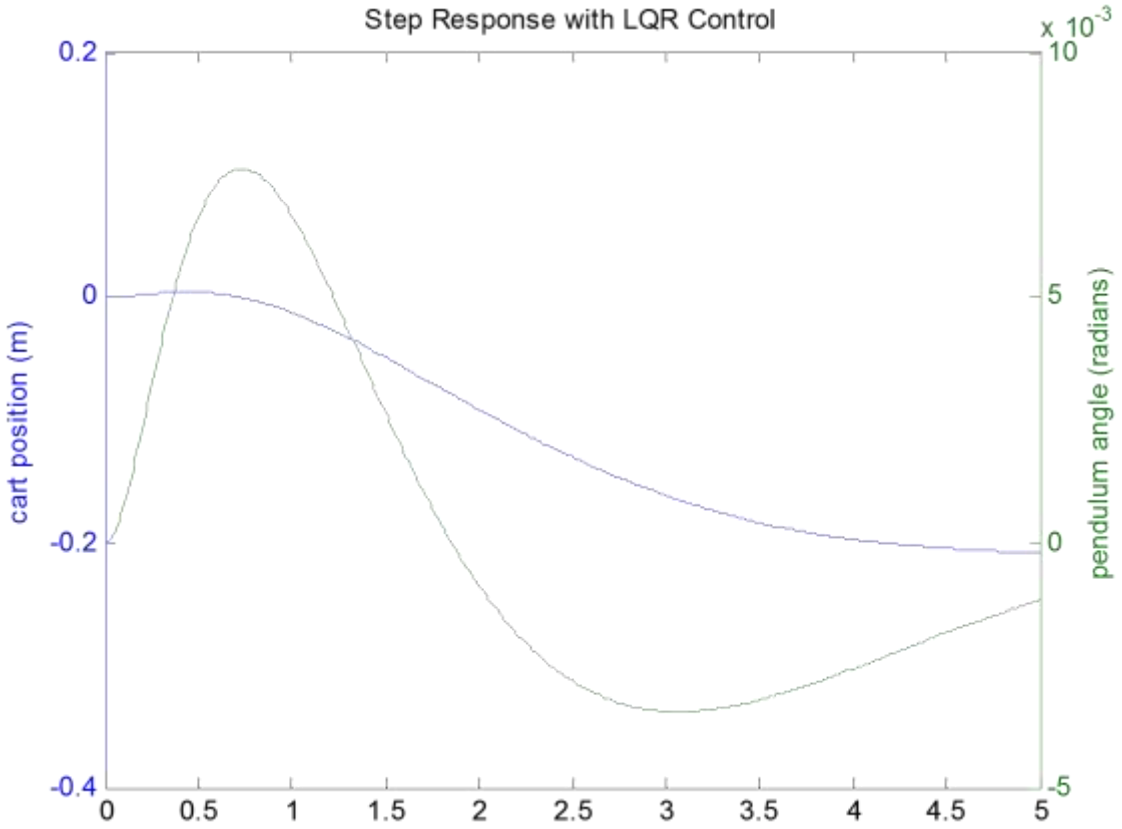
با اضافه کردن دستور فوق به دستورات قبلی در پنجره فرمان نرم‌افزار MATLAB می‌توان را به‌صورت زیر به‌دست آورد

```
K =

   -1.0000   -2.1000   32.3890   10.0232
```

نمودار پاسخ نیز با استفاده از دستورات زیر می‌توان استخراج شود.

```
Ac=[(A-B*K)];
Bc=[B];
Cc=[C];
Dc=[D];
states={'x','x_dot','phi','phi_dot'};
inputs={'r'};
outputs={'x';'phi'};
sys_cl=ss(Ac,Bc,Cc,Dc,'statename',states,'inputname',inputs,'outputname',ou
tputs);
t = 0:0.01:5;
r =0.2*ones(size(t));
[y,t,x]=lsim(sys_cl,r,t);
[AX,H1,H2]=plotyy(t,y(:,1),t,y(:,2),'plot');
set(get(AX(1),'Ylabel'),'String','cart position (m)')
set(get(AX(2),'Ylabel'),'String','pendulum angle (radians)')
title('Step Response with LQR Control')
```

که نتیجه دستور فوق نمودار زیر است.



منحنی سبز رنگ و سمت راست نشان‌دهنده زاویه آونگ بر حسب رادیان و منحنی آبی‌رنگ و سمت چپ نشـان‌دهنـده موقعیـت گاری بر حسب متر است. همان طوری که می‌بینید این منحنی‌ها رضایت‌بخش نیست. حداکثر فرا جهش زاویه آونگ و موقعیـت گاری در نمودار به نظر مناسب می‌رسد ولی زمان نشست نیاز دارد تا بهبود یابد و همچنین زمان صعود گاری بایـد کـاهش یابـد. همچنین موقعیت گاری در جهت مخالف حرکت کرده است. به مقدار خطای ماندگار در بخش های بعدی پرداخته مـی‌شـود و در حال حاضر بر روی زمان صعود و زمان نشست تمرکز می‌گردد.

با مراجعه به دستورات فوق و تغییر در ماتریس Q به پیدا کردن پاسخ بهتری به‌صورت آزمایشی می‌پـردازیم. بـا افـزایش مقـادیر آرایه‌های و به کاهش زمان نشست، زمان نشست و تغییرات زاویه آونگ منجر می‌شود. که در واقع این تغییرات وزن خطـا را در بخش بهینه‌سازی دستور lqr افزایش می‌دهد.

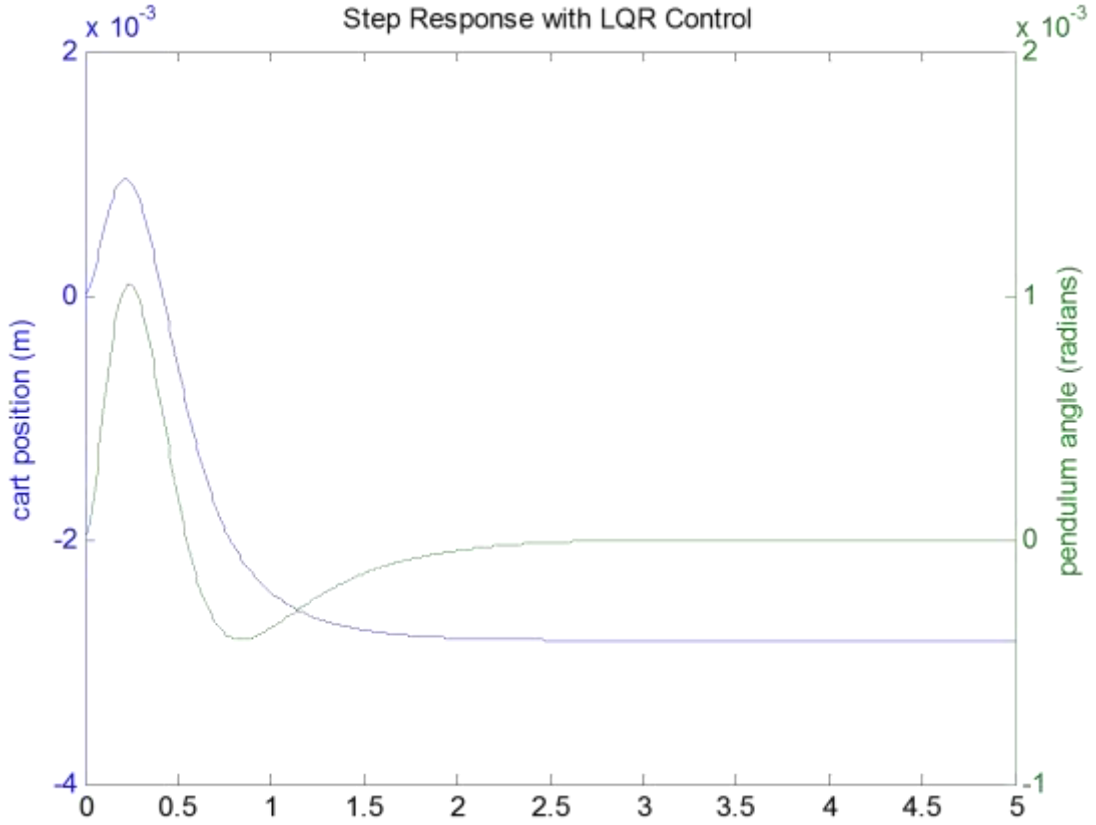
```
Q=C'*C;
Q(1,1)=5000;
Q(3,3)=100;
Q
R=1;
K=lqr(A,B,Q,R)
Ac=[(A-B*K)];
Bc=[B];
Cc=[C];
Dc=[D];
states={'x','x_dot','phi','phi_dot'};
inputs={'r'};
outputs={'x';'phi'};
sys_cl=ss(Ac,Bc,Cc,Dc,'statename',states,'inputname',inputs,'outputname',ou
tputs);
t=0:0.01:5;
r=0.2*ones(size(t));
[y,t,x]=lsim(sys_cl,r,t);
[AX,H1,H2]=plotyy(t,y(:,1),t,y(:,2),'plot');
set(get(AX(1),'Ylabel'),'String','cart position (m)')
```

```
set(get(AX(2),'Ylabel'),'String','pendulum angle (radians)')
title('Step Response with LQR Control')
```

با اضافه کردن دستورات فوق به m-file قبلی در نرم‌افزار MATLAB به نتیجه زیر می‌رسیم.

```
Q =

        5000            0            0            0
           0            0            0            0
           0            0          100            0
           0            0            0            0

K =

  -70.7107  -57.1791  237.3410   75.4345
```

و نمودار جدید به‌صورت زیر می‌گردد.



حتی با افزایش بیشتر مقادیر ماتریس Q به بهبود سیستم بیشتر کمک می‌کند و خطای حالت ماندگار را کاهش می‌دهد. ولی نیاز به نیروی کنترلی بیشتری می‌طلبد. در واقع هزینه ساخت کنترلر، انرژی بیشتر و عملگر بزرگ‌تر را نیاز دارد.

روش استقرار قطب مقاوم place

استقرار قطب مبتنی بر روش استقرار قطب مقاوم یکی از روش های مرسومی است که به ماکزیمم کردن حد پایداری می‌انجامد. هرچند فرمان place در نرم‌افزار MATLAB می‌تواند برای هر دو نوع سیستم یک ورودی و سیستم چند ورودی بکار رود ولی لازمه‌اش این است که چندگانگی قطب در قطب های حلقه بسته بیش از مرتبه B نباشد. ماتریس J ماتریسی متشکل از قطب های حلقه بسته مطلوب مانند زیر است.

$$J = [\mu_1 \,|\, \mu_2 \,|\, ... \,|\, \mu_n]$$

که μ_i قطب های حلقه بسته مطلوب است.

همان طور که گفته شد معیار های طراحی برای انتقال موقعیت گاری x به‌اندازه $0.2\ \text{m}$ به‌صورت تابع پله به‌صورت زیر است:

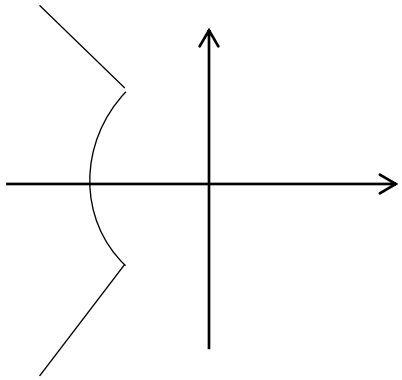
- زمان نشست برای x و θ کمتر از $5\ \text{sec}$ باشد.
- زمان صعود برای x کمتر از $0.5\ \text{sec}$ باشد.
- زاویه انحراف آونگ ϕ نباید بیشتر از $20\ \text{deg}$ یا $0.35\ \text{rad}$ از حالت عمودی آن باشد.
- خطای حالت ماندگار کمتر از 2% برای x و θ باشد.

که با محاسبات زیر می‌توان محدوده قطب های مطلوب را به‌دست آورد.

$$\left.\begin{aligned} t_s &= \frac{4.6}{\xi\omega_n} \xrightarrow{t_s<5} \xi\omega_n > 0.92 \\ t_r &= \frac{1.8}{\omega_n} \xrightarrow{t_r<0.5} \omega_n > 3.6 \end{aligned}\right\} \rightarrow \begin{cases} \omega_n > 3.6 \\ \xi > 0.26 \end{cases}$$

$$e_{ss} < 0.2\%$$

البته بهتر است $0.5 > \xi > 0.7$ باشد تا پاسخ سیستم سریع‌تر و با زمان نشست کمتر باشد لذا محدوده قطب هـای قابـل‌قبـول به‌صورت زیر می‌گردد.



با استفاده از دستورات زیر می‌توانیم بردار ضرایب کنترلی $\underset{\sim}{K}$ را به‌دست آوریم.

```
M=1;
m=0.1;
I=0;
g=9.8;
l=1;
p=I*(M+m)+M*m*l^2;
A=[0,1,0,0;0,0,(m^2*g*l^2)/p,0;0,0,0,1;0,0,m*g*l*(M+m)/p,0];
B=[0;(I+m*l^2)/p;0;m*l/p];
C=[1,0,0,0;0,0,1,0];
D=[0;0];
J=[-3.9 -5 -4.4 -4];
K=place(A,B,J)
```

که نتیجه آن ماتریس زیر است.

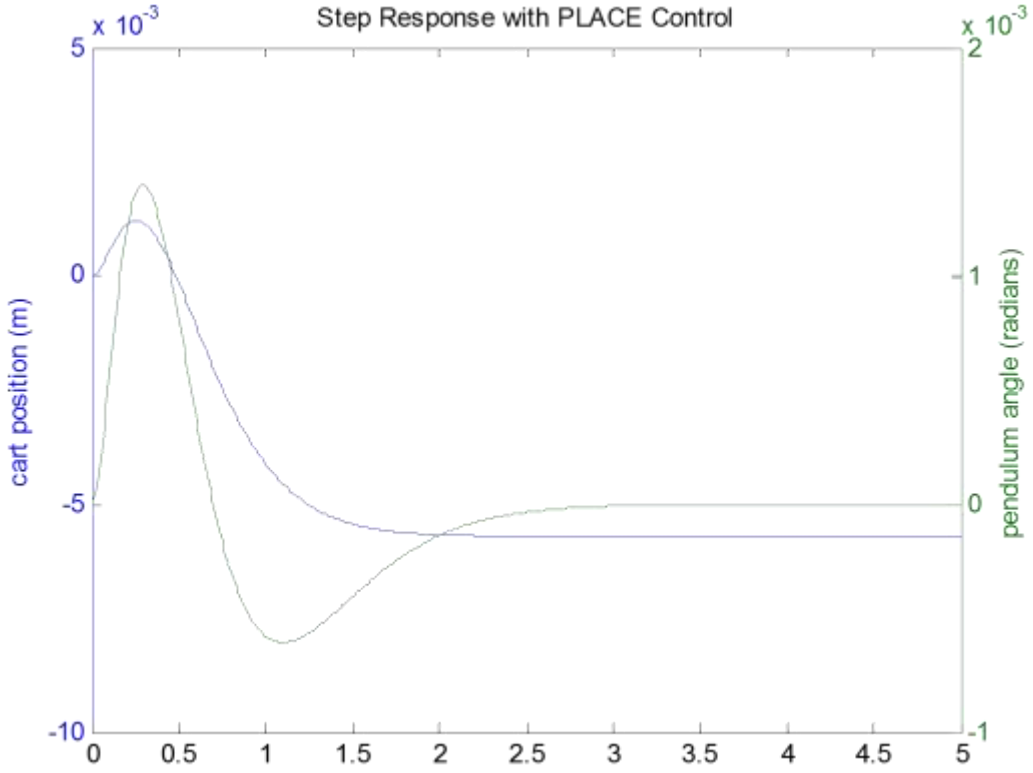
```
K =

  -35.0204  -32.6980  157.6604   49.9980
```

و پاسخ سیستم با استفاده از دستور زیر به‌دست می‌آید.

```
Ac=[(A-B*K)];
```

```
Bc=[B];
Cc=[C];
Dc=[D];
states={'x','x_dot','phi','phi_dot'};
inputs={'r'};
outputs={'x';'phi'};
sys_cl=ss(Ac,Bc,Cc,Dc,'statename',states,'inputname',inputs,'outputname',ou
tputs);
t=0:0.01:5;
r=0.2*ones(size(t));
[y,t,x]=lsim(sys_cl,r,t);
[AX,H1,H2]=plotyy(t,y(:,1),t,y(:,2),'plot');
set(get(AX(1),'Ylabel'),'String','cart position (m)')
set(get(AX(2),'Ylabel'),'String','pendulum angle (radians)')
title('Step Response with PLACE Control')
```

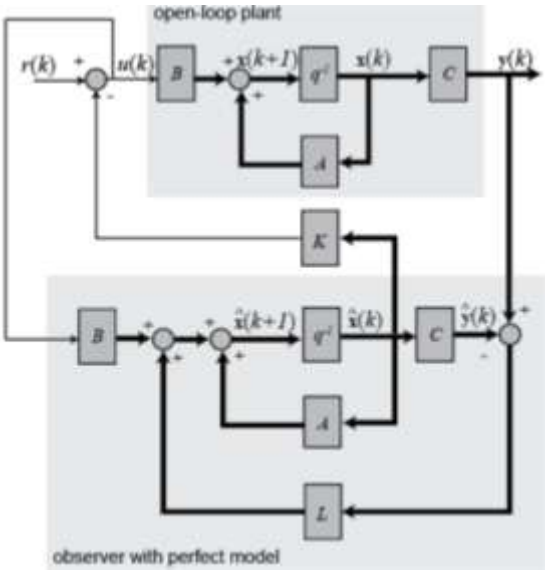
که این دستور نمودار زیر را ارائه می‌دهد.



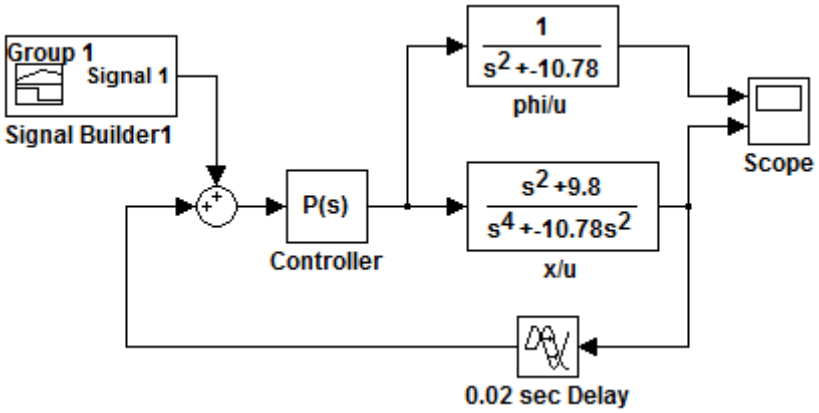
همان طور که از نمودار مشهود است ملاحظات طراحی به خوبی رعایت شده است.

فصل هفتم: طراحی کنترل دیجیتال

قصد داریم سیستم را به‌صورت دیاگرام زیر کنترل کنیم.



طراحی کنترلر برای سیستم زیر با زمان نمونه‌برداری ۰,۰۰۵ ثانیه به این صورت است. که ابتدا تابع تأخیر را تقریب می‌زنیم. سپس تابع تبدیل سیستم و تأخیر را پیداکرده و آن را به فضای حالت برده و شروع به طراحی کنترلر و مشاهده گر می‌کنیم.



تقریب پادهٔ برای دینامیک تأخیر به‌صورت زیر است.

$$e^{-Ts} = \frac{1-0.5Ts}{1+0.5Ts} \xrightarrow{T=0.02} \frac{1-0.01s}{1+0.01s}$$

حال تابع تبدیل سیستم را به‌دست می‌آوریم.

```
M=1;
m=0.1;
I=0;
g=9.8;
l=1;
p=I*(M+m)+M*m*l^2;
A=[0,1,0,0;0,0,(m^2*g*l^2)/p,0;0,0,0,1;0,0,m*g*l*(M+m)/p,0];
B=[0;(I+m*l^2)/p;0;m*l/p];
C=[1,0,0,0;0,0,1,0];
```

```
D=[0;0];
states={'x','x_dot','phi','phi_dot'};
inputs={'u'};
outputs={'x';'phi'};
sys_ss=ss(A,B,C,D,'statename',states,'inputname',inputs,'outputname',output
s);
[b,a]=ss2tf(A,B,C,D)
```

که نتیجه استفاده از دستور فوق به‌صورت زیر است.

```
b =

         0    0.0000    1.0000    0.0000   -9.8000
         0    0.0000    1.0000         0         0


a =

    1.0000   -0.0000  -10.7800         0         0
```

که به‌صورت زیر ساده می‌شود.

$$\frac{X(s)}{U(s)} = \frac{s^2 - 9.8}{s^4 - 10.78s^2}$$

$$\frac{\phi(s)}{U(s)} = \frac{s^2}{s^4 - 10.78s^2}$$

حال دینامیک تأخیر را در تابع تبدیل مربوط به موقعیت گاری ضرب می‌کنیم و سیستم را ساده می‌کنیم.

```
xsysdelay=tf([-0.01,1],[0.01,1]);
xsysplant=tf([1,0,-9.8],[1,0,-10.78,0,0]);
xsys=tf(xsysplant*xsysdelay)
```

که نتیجه استفاده از دستور فوق به‌صورت زیر است.

```
Transfer function:
    -0.01 s^3 + s^2 + 0.098 s - 9.8
-------------------------------------
0.01 s^5 + s^4 - 0.1078 s^3 - 10.78 s^2
```

البته تابع تبدیل زاویه آونگ به نیرو نیز به‌صورت زیر تبدیل می‌شود.

```
phisysd=tf([0.01,1],[0.01,1]);
phisysplant=tf([1,0,0],[1,0,-10.78,0,0]);
xsys=tf(phisysplant*phisysd)
```

که نتیجه دستور فوق به‌صورت زیر است.

```
Transfer function:
           0.01 s^3 + s^2
-------------------------------------
0.01 s^5 + s^4 - 0.1078 s^3 - 10.78 s^2
```

با استفاده از دستور زیر می‌توان تابع تبدیل را به فضای حالت تبدیل کرد البته به دلیل اینکه فقط اندازه‌گیری موقعیت وجـود دارد ردیف دوم ماتریس ضرایب صورت صفر است.

```
b=[0,0,-0.01,1,0.098,-9.8;0,0,0,0,0,0];
a=[0.1,1,-0.1078,-10.78,0,0];
[A,B,C,D]=tf2ss(b,a)
```

که نتیجه دستور فوق به‌صورت زیر است.

```
A =

    0.0000   10.7800         0         0
    1.0000         0         0         0
         0    1.0000         0         0
```

```
          0          0    1.0000          0

B =

     1
     0
     0
     0

C =

   -0.0100    1.0000   98.0000   -9.8000
         0          0         0         0

D =

     0
     0
```

و اگر به‌صورتی دستی متغیر زاویه آونگ و مشتق زاویه آونگ را به فرمول فوق اضافه کنیم نتیجه به‌صورت زیر است:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\Phi} \\ \ddot{\Phi} \end{bmatrix} = \begin{bmatrix} 0 & 10.78 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \Phi \\ \dot{\Phi} \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} u$$

$$y = \begin{bmatrix} -0.01 & 1 & 98 & -9.8 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \Phi \\ \dot{\Phi} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u$$

حال این سیستم را به یک سیستم گسسته تبدیل می‌کنیم.

```
A =[0,10.78,0,0;1,0,0,0;0,1,0,0;0,0,1,0];
B =[1;0;0;0];
C =[-0.01,1,98,-9.8;0,0,0,0];
D=[0;0];
states={'x' 'x_dot' 'phi' 'phi_dot'};
inputs={'u'};
outputs={'x';'phi'};
sys_ss=ss(A,B,C,D,'statename',states,'inputname',inputs,'outputname',output
s);
Ts=5/1000;
sys_d=c2d(sys_ss,Ts,'zoh')
```

که نتیجه دستور فوق در نرم‌افزار به‌صورت زیر است.

```
a =
                     x       x_dot         phi     phi_dot
   x                 1      0.0539           0           0
x_dot          0.005           1           0           0
   phi        1.25e-005       0.005           1           0
phi_dot   2.083e-008   1.25e-005       0.005           1

b =
                     u
   x             0.005
x_dot      1.25e-005
   phi       2.083e-008
phi_dot   2.604e-011
```

```
c =
              x    x_dot      phi  phi_dot
   x      -0.01        1       98     -9.8
   phi        0        0        0        0

d =
         u
   x     0
   phi   0

Sampling time (seconds): 0.005
Discrete-time model.
```

کنترل‌پذیری و مشاهده‌پذیری از طریق دستور زیر چک می‌شود.

```
co = ctrb(sys_d);
ob = obsv(sys_d);
controllability = rank(co)
observability = rank(ob)
```

طبق نتیجه زیر این سیستم کنترل‌پذیر و مشاهده‌پذیر است.

```
controllability =

     4


observability =

     4
```

حال با استفاده از دستور زیر به طراحی کنترلر می‌پردازیم.

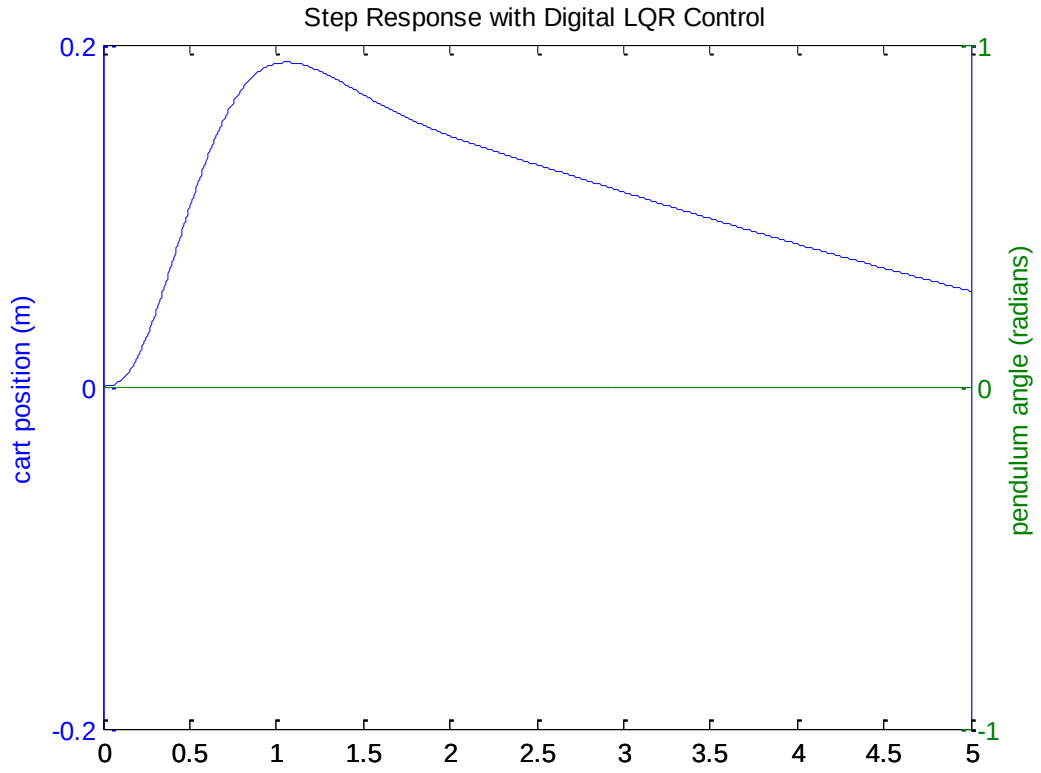
```
A = sys_d.a;
B = sys_d.b;
C = sys_d.c;
D = sys_d.d;
Q = C'*C
R = 1;
[K] = dlqr(A,B,Q,R)
Ac = [(A-B*K)];
Bc = [B];
Cc = [C];
Dc = [D];
states = {'x' 'x_dot' 'phi' 'phi_dot'};
inputs = {'r'};
outputs = {'x'; 'phi'};
sys_cl=ss(Ac,Bc,Cc,Dc,Ts,'statename',states,'inputname',inputs,'outputname'
,outputs);
t = 0:0.005:5;
r =0.2*ones(size(t));
[y,t,x]=lsim(sys_cl,r,t);
[AX,H1,H2] = plotyy(t,y(:,1),t,y(:,2),'plot');
set(get(AX(1),'Ylabel'),'String','cart position (m)')
set(get(AX(2),'Ylabel'),'String','pendulum angle (radians)')
title('Step Response with Digital LQR Control')
```

که نتیجه به‌صورت زیر است

```
Q =

  1.0e+003 *

    0.0000   -0.0000   -0.0010    0.0001
   -0.0000    0.0010    0.0980   -0.0098
   -0.0010    0.0980    9.6040   -0.9604
```

```
    0.0001   -0.0098   -0.9604    0.0960


K =

   10.7264   58.0607  100.0813    9.5372
```

و نمودار پاسخ به‌صورت زیر است.



طراحی مشاهده گر

قطب‌ها به‌صورت زیر محاسبه می‌گردد.

```
poles = eig(A-B*K)
```

و نتیجه به‌صورت زیر است

```
poles =

   0.9735
   0.9865 + 0.0163i
   0.9865 - 0.0163i
   0.9995
```

حال با استفاده از این قطب‌ها به محاسبه L از روش Place می‌پردازیم.

```
P = [-0.2 -0.21 -0.22 -0.23];
L = place(A',C',P)'
```

نتیجه به‌صورت زیر است.

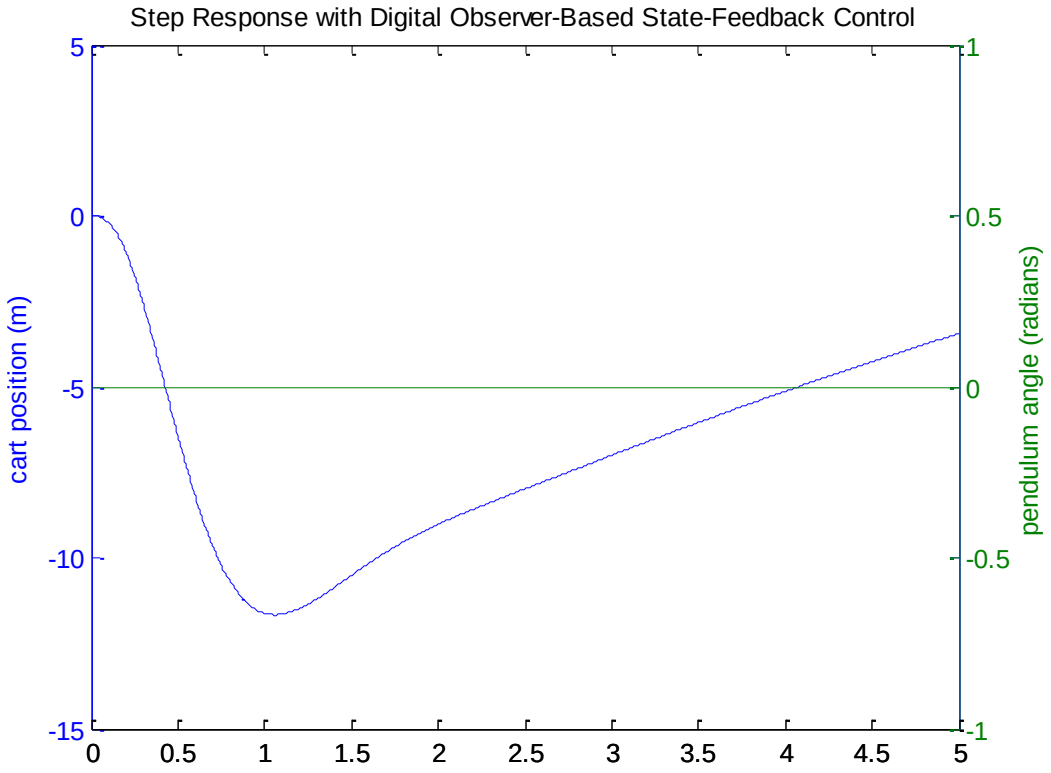
```
L =

  1.0e+006 *

    0.0019         0
    0.0163         0
    0.1631         0
```

```
    1.6322          0
```

حال یک مشاهده گر طراحی می‌کنیم. روش طراحی و دستور های مربوطه به‌صورت زیر است.

```
Nbar = -61.55;
Ace = [(A-B*K)  (B*K);
       zeros(size(A))  (A-L*C)];
Bce = [B*Nbar;
       zeros(size(B))];
Cce = [Cc zeros(size(Cc))];
Dce = [0;0];

states = {'x' 'x_dot' 'phi' 'phi_dot' 'e1' 'e2' 'e3' 'e4'};
inputs = {'r'};
outputs = {'x'; 'phi'};

sys_est_cl =
ss(Ace,Bce,Cce,Dce,Ts,'statename',states,'inputname',inputs,'outputname',ou
tputs);

t = 0:0.005:5;
r = 0.2*ones(size(t));
[y,t,x]=lsim(sys_est_cl,r,t);
[AX,H1,H2] = plotyy(t,y(:,1),t,y(:,2),'plot');
set(get(AX(1),'Ylabel'),'String','cart position (m)')
set(get(AX(2),'Ylabel'),'String','pendulum angle (radians)')
title('Step Response with Digital Observer-Based State-Feedback Control')
```

که پاسخ به تابع پله به‌صورت زیر است.



فصل نهم: کنترل غیرخطی

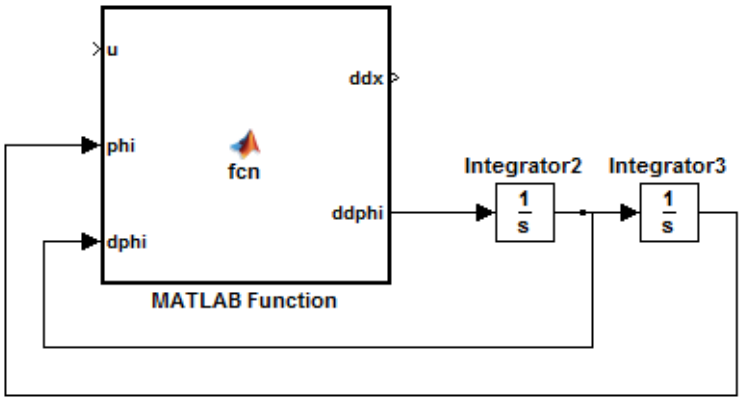
با توجه به استخراج معادلات در فصل اول و بدون خطی سازی معادلات حاکم بر سیستم به صورت زیر در می‌آیند.

$$\left.\begin{aligned}(M+m)\ddot{x}+ml\ddot{\theta}\cos\theta-ml\dot{\theta}^2\sin\theta=F\\(I+ml^2)\ddot{\theta}-mgl\sin\theta=-ml\ddot{x}\cos\theta\end{aligned}\right\}\rightarrow\begin{aligned}1.1\ddot{x}+0.1\ddot{\theta}\cos\theta-0.1\dot{\theta}^2\sin\theta=F\\0.1\ddot{\theta}-0.98\sin\theta=-0.1\ddot{x}\cos\theta\end{aligned}$$

که با ساده‌سازی داریم:

$$\left.\begin{aligned}11\ddot{x}+\ddot{\theta}\cos\theta-\dot{\theta}^2\sin\theta-10F=0\\\ddot{\theta}-9.8\sin\theta+\ddot{x}\cos\theta=0\end{aligned}\right\}\rightarrow\begin{aligned}\ddot{\theta}&=9.8\sin\theta-\frac{\dot{\theta}^2\sin\theta+10F-9.8\sin\theta\cos\theta}{11-\cos^2\theta}\cos\theta\\\ddot{x}&=\frac{\dot{\theta}^2\sin\theta+10F-9.8\sin\theta\cos\theta}{11-\cos^2\theta}\end{aligned}$$

با ساختن یک MATALAB FUNCTION با معماری زیر می‌توان با دادن مقدار اولیه ۰٫۱ رادیان به انتگرال گیر دوم سیستم شرایط را در خروجی شبیه‌سازی کرد:



با استفاده بلاک function و المان های مشتق گیر برای شبیه‌سازی سیستم غیرخطی به‌صورت فوق از دستور زیر در بلاک استفاده می‌کنیم.

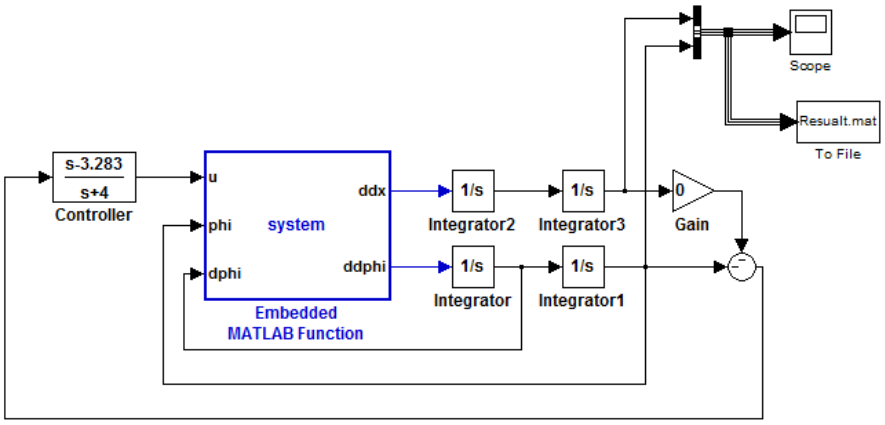
```
function [ddx,ddphi] = fcn(u,phi,dphi)
teta=phi+pi;
dteta=dphi;
ddx=((dteta^2*sin(teta)+10*u-9.8*sin(teta)*cos(teta))/(11-
cos(teta)*cos(teta)));
ddteta=9.8*sin(teta)-((dteta^2*sin(teta)+10*u-9.8*sin(teta)*cos(teta))/(11-
cos(teta)*cos(teta)))*cos(teta);
ddphi=ddteta;
```

پاسخ سیستم غیرخطی با کنترل کلاسیک

از کنترل‌کننده‌های طراحی‌شده در فصل پنجم را یک بار دیگر استفاده می‌کنیم و نتایج را با همان سیستم ولی به‌صورت غیرخطی صحت سنجی می‌کنیم.

کنترل‌کننده با K=0 مربوط به دینامیک سیستم

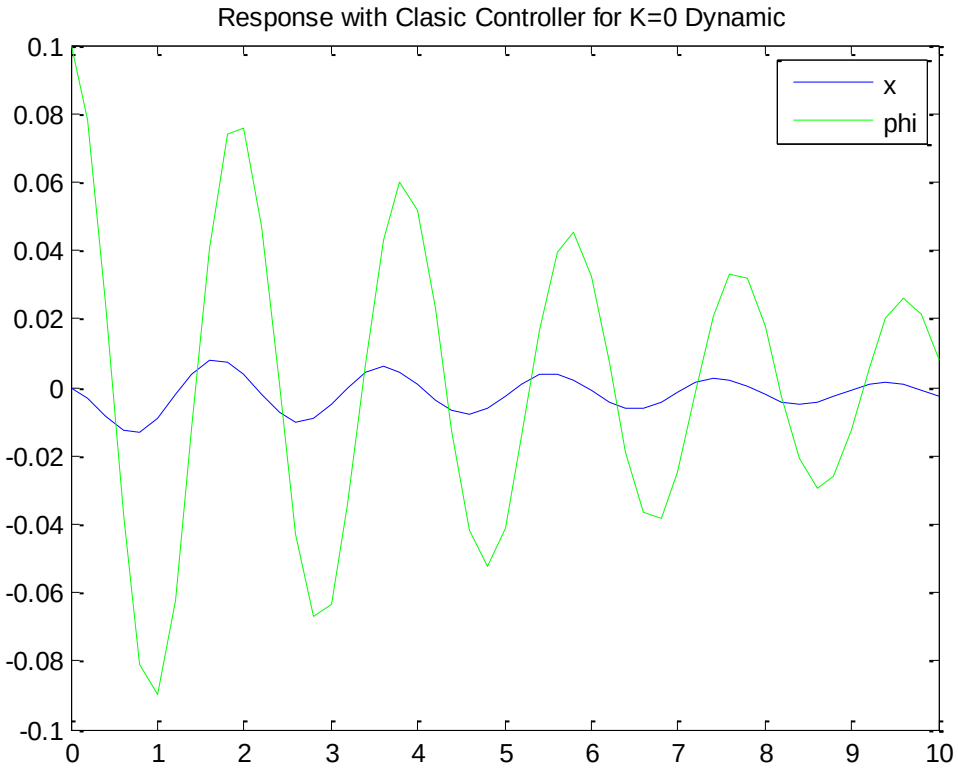
سیستم را به‌صورت زیر شبیه‌سازی می‌کنیم.



همان طور که در شکل فوق می‌بینید از یک المان To File استفاده شده است که اطلاعات زا ذخیره کند و پس از اجرای برنامه از طریق یک m-file همان اطلاعات را فراخوانی کنیم و نمودار مربوطه را رسم کنیم. این فراخوانی با دستورات زیر انجام می‌شود.

```
load Resualt
nt=max(size(ans));
nd=1;
x=ans(1,:);
y1=ans(2,:);
y2=ans(3,:);
plot(x,y1,'b',x,y2,'g');
title('Response with Clasic Controller for K=0 Dynamic')
legend('x','phi')
```

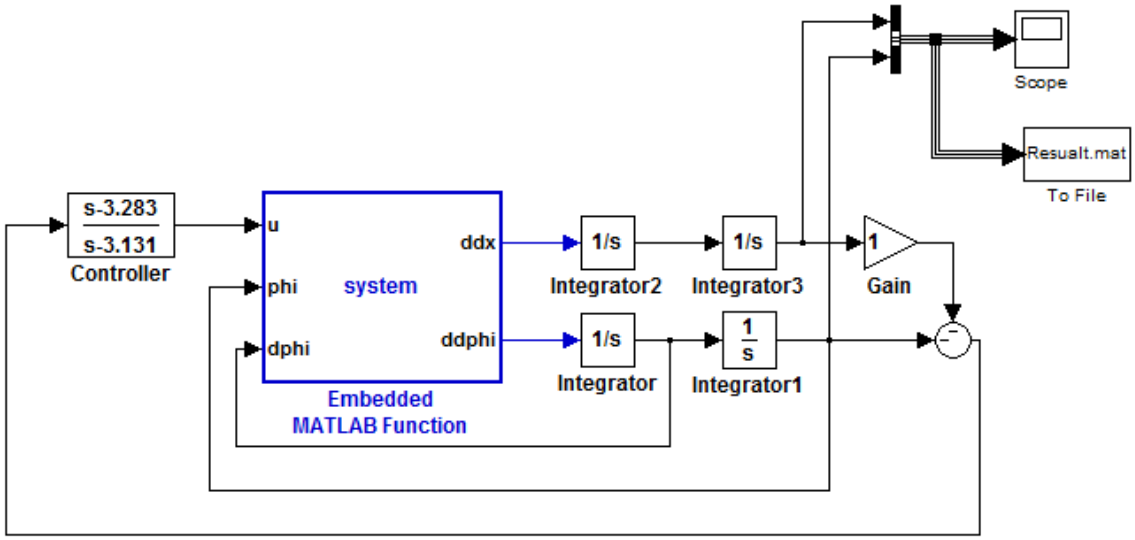
که پاسخ سیستم در ۱۰ ثانیه به‌صورت زیر است.



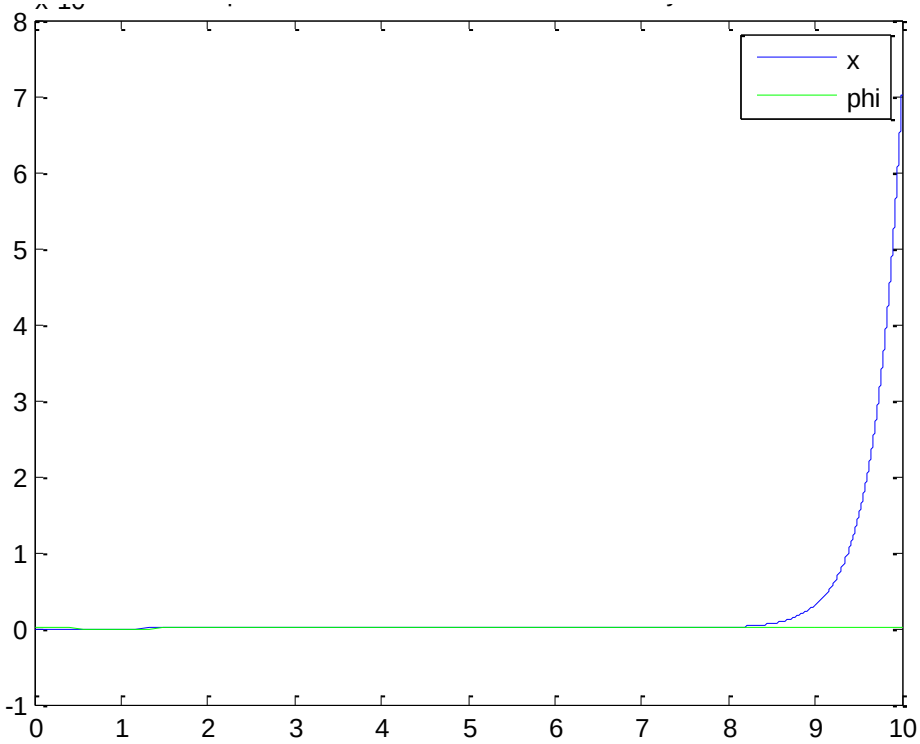
همان طور که گفته شد با طراحی یک جبران ساز نیز می‌توانیم پاسخ سیستم را بهبود بخشید.

کنترل‌کننده با K=1 مربوط به دینامیک سیستم

این سیستم به‌صورت زیر شبیه‌سازی شده است.



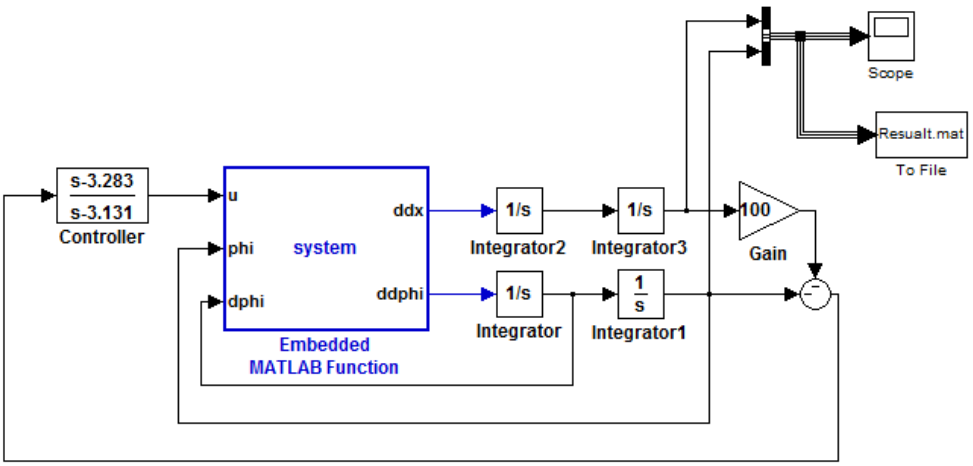
پاسخ به‌صورت زیر خواهد بود.



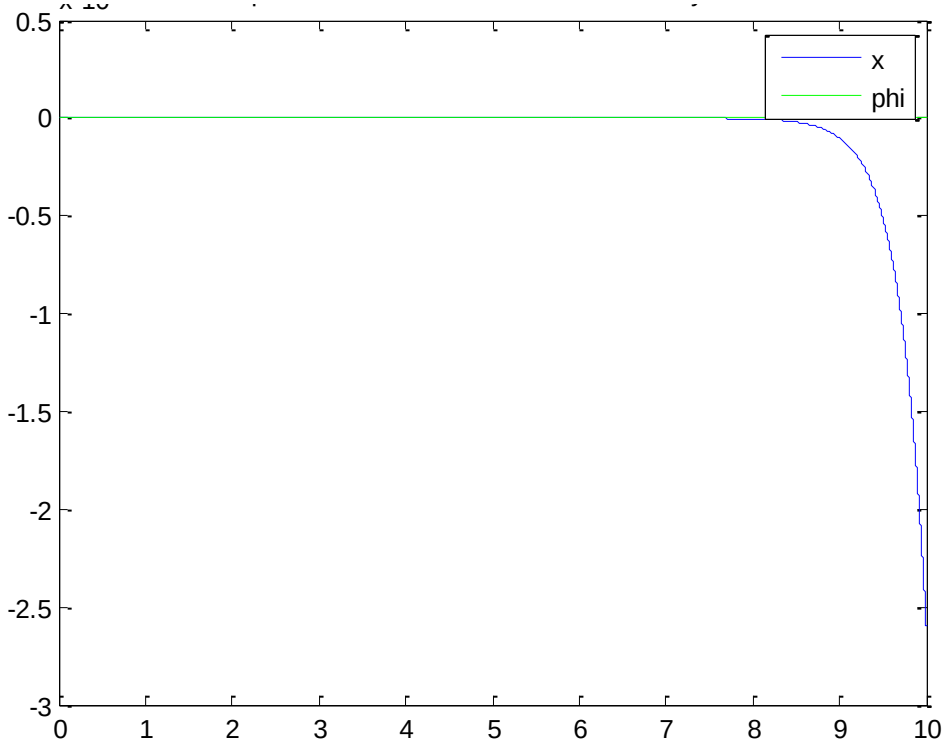
همان طور که متوجه شدید زاویه آونگ توسط کنترل‌کننده تصحیح شده است ولی موقعیت گاری پس از گذشت زمان کمی ناپایدار شده و افزایش می‌یابد و این به دلیل عدم طراحی کنترل‌کننده دقیق است و این‌گونه طراحی کاملاً وابسته به دینامیک سیستم و با تغییر جزئی آن کنترل‌کننده پاسخ مناسبی نمی‌دهد حتی باعث تخریب عملکرد سیستم می‌شود.

کنترل‌کننده با K=100 مربوط به دینامیک سیستم

این بخش نیز به‌صورت زیر در نرم‌افزار MATLAB شبیه‌سازی می‌شود.



پاسخ این سیستم جدید نیز به‌صورت زیر است.

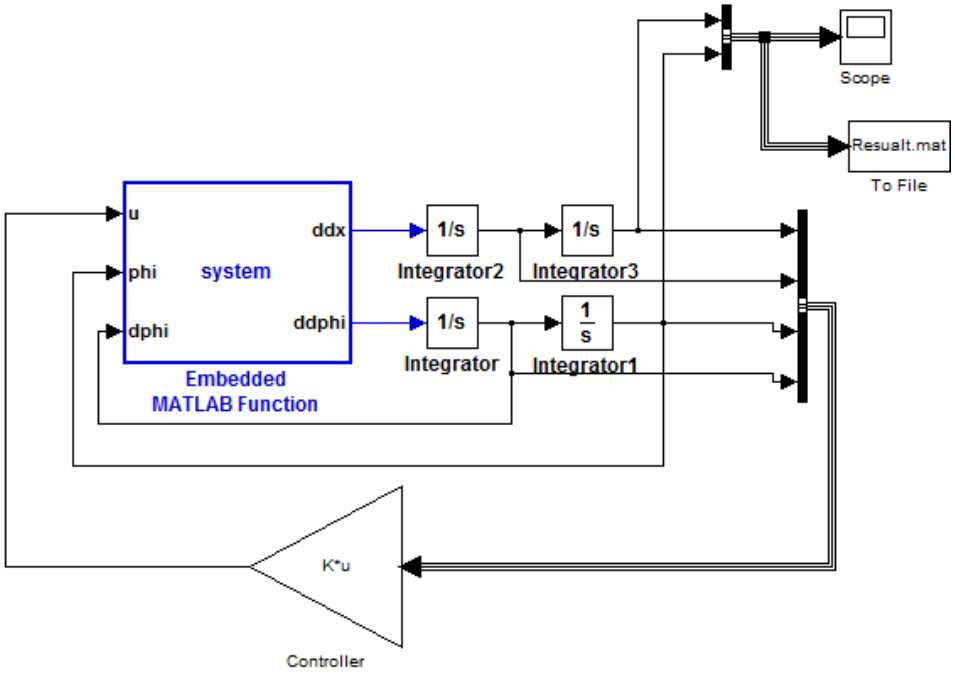


دقیقاً توضیحات قسمت قبل نیز برای این بخش صادق است.

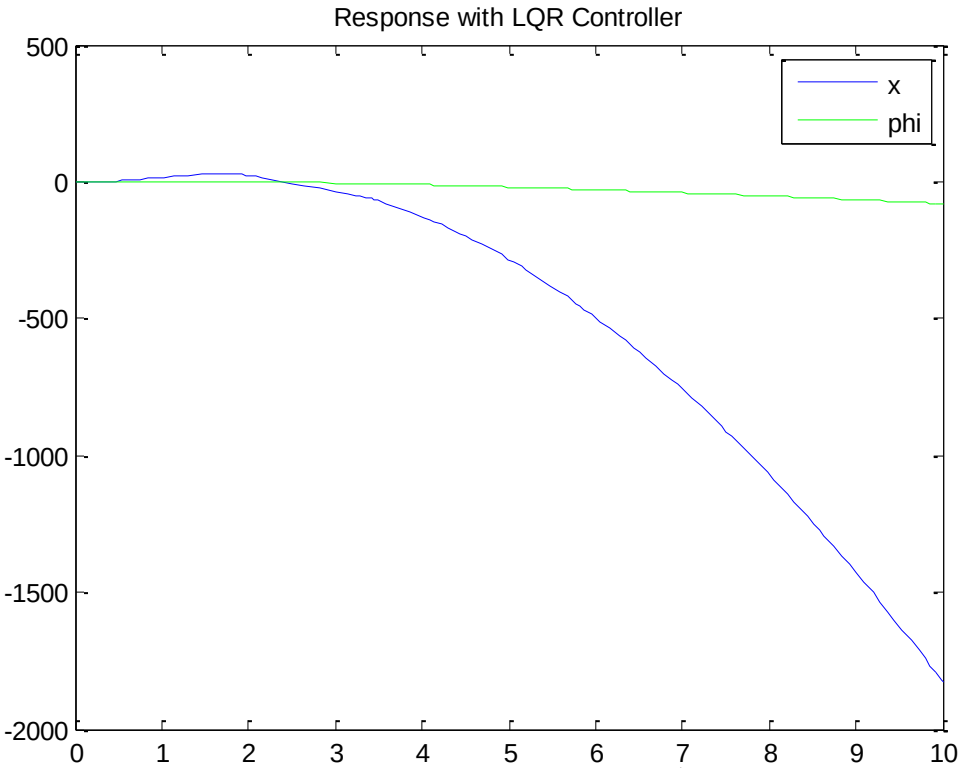
پاسخ سیستم غیرخطی با کنترل‌کننده مدرن

کنترل‌کننده طراحی‌شده با روش مقررات درجه دوم خطی (LQR)

سیستم به‌صورت زیر در نرم‌افزار MATLAB پیاده‌سازی شده است.



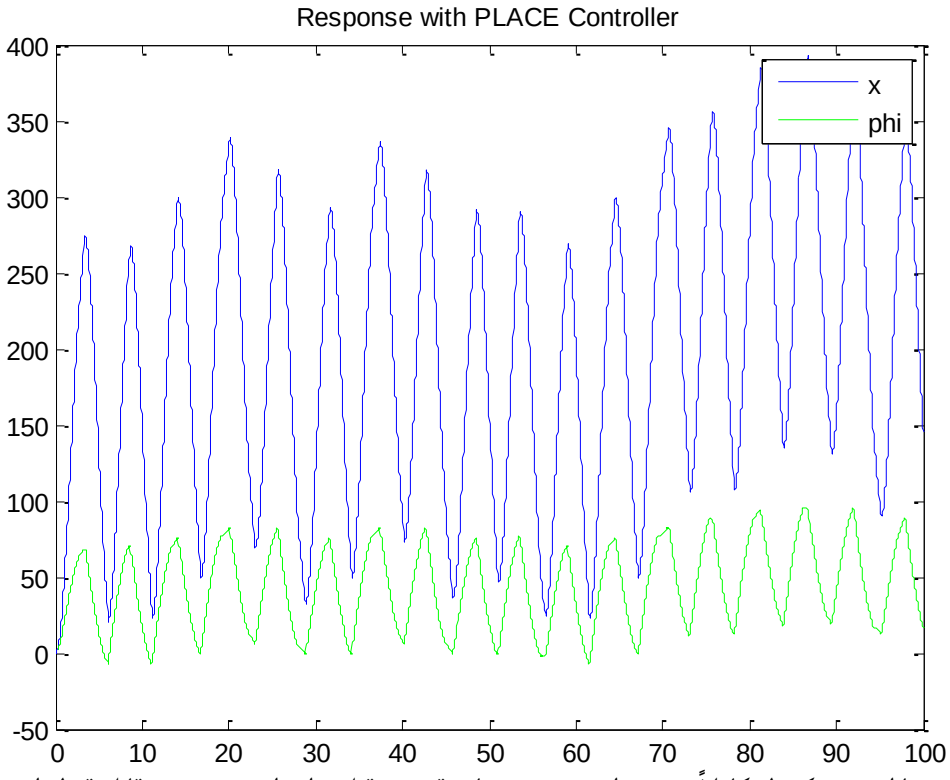
و پاسخ به‌صورت زیر است.



همان طور که متوجه شدید این پاسخ کاملاً نامطلوب است و تصدیق‌کننده آن است که سیستم غیرخطی با روش‌هایی کـه بـرای سیستم های LTI کنترلر طراحی می‌شود سازگار نیست.

کنترل‌کننده طراحی‌شده با روش استقرار قطب مقاوم place

این بخش نیز مانند بخش قبل شبیه‌سازی می‌شود فقط ماتریس K متفاوت است و در فصل ششم بیان شـده اسـت. پاسـخ ایـن سیستم به‌صورت زیر است.



ناپایداری سیستم با این نوع کنترلر کاملاً مشهود است و توضیحات قسمت قبل برای این بخش نیز قابل قبول است.

فصل دهم: جمع‌بندی

در این فصل با بررسی و مقایسه‌ی فصول گذشته و سؤالات مسئله به نتیجه‌گیری اجمالی از آن چه گذشت می‌پردازیم.
نتایج حاصل از حل این مسئله باعث شد که با چالش های مختلف برای کنترل یک سیستم mimo از جمله برای طراحی نوع کنترلر و معماری های آن و مقدار خطای ماندگار و امکان پایدار کردن سیستم در هر نوع طراحی و همچنین تأثیر خطاهای دینامیک داخلی اجزا اندازه‌گیری بر روی کنترل سیستم از جمله سنسورها و پردازش کننده‌ها برخوردیم. همچنین با گسسته سازی باعث شد که شبیه‌سازی سیستم خود را باحالت واقعی منطبق تر کنیم.
در طراحی کنترلر برای قسمت های اول مسئله مشاهده شد با آن که با اعمال ورودی مناسب می‌توان هر دو خروجی را کنترل کرد اما مقداری خطای ماندگار برای یکی از خروجی‌ها داشتیم که غیرقابل اصلاح کلی بود.
تأثیر گسسته سازی در دینامیک سیستم را مورد بررسی قراردادیم.
تأثیر خطی سازی در معادلات را با شبیه‌سازی معادلات غیرخطی و مقایسه آن دو با یکدیگر مشاهده نمودیم و نامناسب بودن این نوع کنترل گر عملاً اثبات شد.
همچنین کنترل‌پذیری و مشاهده‌پذیری سیستم با داشتن ماتریس های مربوطه گویای صحت امکان کنترل سیستم بودند.