Modern Control Paper report
S.Mehdi Moosaviun
ID : 99413136
Spring 2024

پروژه کنترل مدرن،گزارش بخش شبیه سازی مقاله

استاد: دکتر بلندی

تدریس یار: سرکارخانم قنادیان

نویسنده : سید مهدی موسویون – ۹۹۴۱۳۱۳۶

بهار۱۴۰۳

فهرست مطالب

لینک دانلود مقاله

LQR hybrid approach control of a robotic arm two degrees of freedom

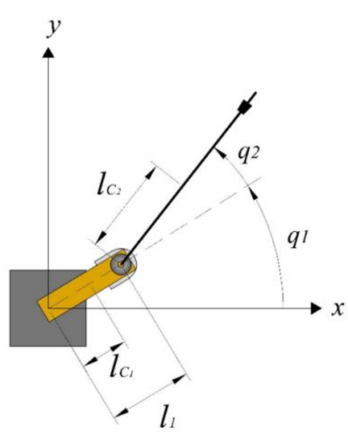
۱- خلاصه ای از مقاله

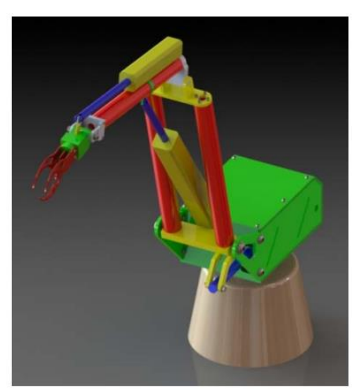
این مقاله یک سیستم کنترل رگولاتور مربعی خطی (LQR) را برای بازوی روباتیک با دو درجه آزادی پیشنهاد می‌دهد. هدف اصلی حفظ موقعیت عمود بازو در برابر اختلالات با استفاده از نظریه‌های کنترل بهینه کلاسیک و اصول LQR است.

مطالعه با مدل‌سازی ریاضی بازوی روباتیک آغاز می‌شود تا معادلات دینامیکی حرکت و کنترل آن استخراج شود. سپس معادلات غیرخطی سیستم برای تسهیل طراحی کنترل، خطی‌سازی شده و برای شبیه‌سازی رفتار بازوی روباتیک تحت شرایط مختلف با استفاده از MATLAB Simulink به کار گرفته می‌شوند. اجزای کلیدی LQR شامل نمایش حالت-فضا، تابع هزینه و معادله Riccati است. الگوریتم پیاده‌سازی LQR به صورت گام به گام ارائه شده که شامل مقداردهی اولیه، حل معادله Riccati و بهبود تدریجی قانون کنترل است.

نتایج شبیه‌سازی نشان می‌دهند که کنترل‌کننده LQR به طور مؤثری بیش‌ازحد را کاهش داده و زمان نشست را بهبود می‌بخشد. معیارهای عملکرد کلیدی مانند زمان صعود (۰.۳۵ ثانیه)، زمان اوج (۰.۴۵ ثانیه)، زمان نشست (از ۲.۵ به ۱.۵ ثانیه بهبود یافته) و خطای حالت پایدار (کاهش یافته به مقدار ناچیز) بهبود قابل توجهی را نشان می‌دهند.

مقاله همچنین یک رویکرد ترکیبی جدید برای کنترل بهینه سیستم‌های هیبریدی ارائه می‌دهد. این رویکرد از ساختار واریاسیونی سیستم‌های مکانیکی غیرخطی استفاده می‌کند و یک مسئله بهینه‌سازی چند هدفه را فرموله می‌کند. الگوریتمی برای محاسبه جفت بهینه کنترل و مسیر حالت ارائه شده است. نتایج نشان می‌دهند که با استفاده از کنترل هیبریدی LQR می‌توان پاسخ سیستم را بهبود بخشید، از جمله حذف فراجهش و کاهش زمان نشست از حدود ۲۰ ثانیه به ۵ ثانیه در حالت گسسته بهبود بخشید.





۲- کمیت ها، دینامیک و سیستم اولیه

مقاله LQR Hybrid Approach Control of a Robotic Arm Two Degrees of Freedom به ارائه سیستمی برای کنترل بهینه یک بازوی روباتیک با دو درجه آزادی با استفاده از روش کنترلر خطی مربعی (LQR) اشاره دارد. در این مقاله به مدل‌سازی ریاضی بازوی روباتیک به منظور حفظ وضعیت عمودی بازو در حضور اغتشاشات پرداخته شده است.سیستمی که این مقاله به آن پرداخته شده یک سیستم غیرخطی بوده که در ادامه مقاله آن را خطی سازی میکند. مقاله یک الگوریتم مفهومی برای محاسبه بهینه زوج کنترل بهینه ارائه می‌دهد که شامل حل معادله Riccati در بازه‌های زمانی مشخص و بهبود نتایج با استفاده از تکنیک‌های بهینه‌سازی محدب است. در ادامه، به ایجاد سیستم معادلات خطی شده و شبیه‌سازی آن با استفاده از Simulink در نرم‌افزار MATLAB میپردازد و نتایج این شبیه سازی نمایان میکند که شبیه‌سازی پاسخ سیستم در زمان گسسته با استفاده از کنترل هیبریدی LQR، می‌توان پاسخ سیستم را با حذف افزایشی بیش از حد بهبود بخشید و سیگنال به حالت پایدار خود سریع‌تر می‌رسد.

در این مقاله با استفاده از پارامترهای طراحی شده در Solidwork، مدل خطی سیستم ارائه شده و سپس شبیه‌سازی‌های مختلف برای ارزیابی عملکرد سیستم انجام شده است.

کمیت‌های ورودی سیستم

۱. طول لینک‌ها:

طول لینک اول (l1): 0.80278 متر

طول لینک دوم (l2): ذکر نشده است

۲. جرم لینک‌ها:

جرم کل لینک اول (m1): 0.062402 کیلوگرم

جرم کل لینک دوم (m2): 0.054284 کیلوگرم

۳. فاصله تا مرکز جرم لینک‌ها:

فاصله تا مرکز جرم لینک اول (r1): 0.025912 متر

فاصله تا مرکز جرم لینک دوم (r2): 0.31406 متر

- m_1= total mass of the first link [kg]
- l_1= length of the first link [m]
- l_{c1}= distance to the center of mass of the first link [m]
- I_{c1}= moment of inertia of the first link relative to the axis passing through its center of mass [k gm^2]
- m_2= total mass of the second link [kg]
- l_{c2}= distance to the center of mass of the second link [m]
- I_{c2}= moment of inertia of the second link relative to the axis passing through its center of mass link [m]
- g= acceleration of gravity [m/s^2]

$$m_{11} = \theta_1 + \theta_2 + 2\theta_3 \cos(q_2)$$
$$m_{12} = m_{21} = \theta_2 + \theta_3 \cos(q_2)$$
$$m_{22} = \theta_2$$
$$c_{11} = -\theta_3 \sin(q_2) q_2$$
$$c_{11} = -\theta_3 \sin(q_2) q_1 - \theta_3 \sin(q_2) q_2$$
$$c_{21} = \theta_3 \sin(q_2) q_1$$
$$c_{22} = 0$$
$$g_1 = \theta_4 g \cos(q_1) + \theta_5 g \cos(q_1 + q_2)$$
$$g_2 = \theta_5 g \cos(q_1 + q_2)$$
$$\det M = m_{11} m_{22} - m_{21} m_{12}$$
$$T_1 = \tau - c_{11}\dot{q}_1 - c_{12}\dot{q}_2 - g_1$$
$$T_2 = 0 - c_{21}\dot{q}_1 - c_{22}\dot{q}_2 - g_2$$

۴. شتاب گرانش (g): 9.81 متر بر مجذور ثانیه

۵. ممان اینرسی لینک‌ها:

ممان اینرسی لینک اول (I1): ذکر نشده است

ممان اینرسی لینک دوم (I2): ذکر نشده است

کمیت‌های کنترلی سیستم

$$\theta_1 = m_1 l_{c1}^2 + m_2 l_1^2 + l_{c1}$$
$$\theta_2 = m_2 l_{c2}^2 + l_{c2}$$
$$\theta_3 = m_2 l_1 l_{c2}$$
$$\theta_4 = m_1 l_{c1} + m_2 l_1$$

۱. گشتاورهای اعمال‌شده به اتصالات بازوی رباتیک (τ): که به عنوان ورودی‌های کنترلی عمل می‌کنند.

۲. زاویه‌های جابه‌جایی (θ1 θ2): که وضعیت مکانی اتصالات بازوی رباتیک را نشان می‌دهند.

۳. سرعت‌های زاویه‌ای (ω1 ω2): که سرعت چرخش اتصالات بازو را مشخص می‌کنند.

۴. ماتریس اینرسی (M(q)): که دینامیک سیستم را توصیف می‌کند.

۵. ماتریس نیروهای کوریولیس و مرکزیت (C(q, q̇)): که نیروهای غیر خطی مربوط به حرکت را محاسبه می‌کند.

۶. گشتاور گرانشی (g(q)): که تأثیر نیروی گرانش را نشان می‌دهد.

کمیت های خروجی

زمان خیزش (Rise Time): 0.35 ثانیه

زمان اوج (Peak Time): 0.45 ثانیه

زمان نشست (Settling Time): 2.5 ثانیه (برای برخی تست‌ها ۱.۵ ثانیه)

مقدار حالت پایدار (Steady State Value): 0.019

اوج بیشینه (Overshoot): در سیستم بدون کنترل LQR حدود ۲۴٪ و با کنترل LQR بهبود یافته و حذف شده است.

این نتایج در نمودارهای مربوط به رفتار دو لینک بازوی رباتیک که در نرم‌افزارهای Solidwork و Matlab مدل‌سازی و شبیه‌سازی شده‌اند، نشان داده شده است .

مدل دینامیکی بازوی رباتیک با دو درجه آزادی با استفاده از روش اویلر-لاگرانژ به صورت زیر بیان می‌شود.

$$M(q)\ddot{q} + C(\dot{q}, q)\dot{q} + g(q) = \tau$$

در این معادله:

q وکتور موقعیت بازو است.

$\dot{q}$ وکتور سرعت بازو است.

$\ddot{q}$ وکتور شتاب بازو است.

$M(q)$ ماتریس اینرسی است.

$C(\dot{q},q)$ ماتریس نیروهای کوریولیس و مرکزگرا است.

$g(q)$ گشتاور گرانشی است.

τ گشتاور اعمال شده به بازو است.

$$M(q) = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix}$$

$$C(q, \dot{q}) = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

$$g(q) = \binom{g_1}{g_2}$$

$$M((q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \tau$$

$$\begin{bmatrix} \mathrm{I2 + m2l1lc2cos(q2)} & \mathrm{I1 + I2 + m2l12 + 2m2l1lc2cos(q2)} \\ \mathrm{I2} & \mathrm{I2 + m2l1lc2cos(q2)} \end{bmatrix} = M(q)$$

$$\begin{bmatrix} \mathrm{-m2l1lc2sin(q2)(\dot{q}1 + \dot{q}2)} & \mathrm{-m2l1lc2sin(q2)\dot{q}2} \\ 0 & \mathrm{m2l1lc2sin(q2)\dot{q}1} \end{bmatrix} = C(q\,\dot{}, q)$$

$$\begin{bmatrix} \mathrm{m1glc1cos(q1) + m2g\big(l1cos(q1) + lc2cos(q1 + q2)\big)} \\ \mathrm{m2glc2cos(q1 + q2)} \end{bmatrix} = \mathrm{g(q)}$$

- I_1و I_2 ممان اینرسی لینک‌های اول و دوم هستند.
- m_1و m_2 جرم لینک‌های اول و دوم هستند.
- l_1 طول لینک اول است.
- l_{c1} و l_{c2} فاصله تا مرکز جرم لینک‌های اول و دوم هستند.
- G شتاب گرانش است.

۳- معادله سیستم حالت

۳-۱ تئوری

Above

$\dot{x}_1 = AX + BU$

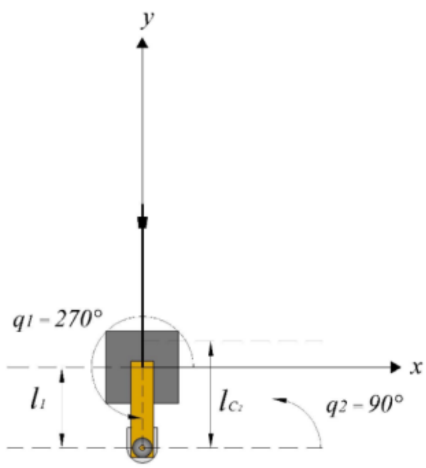
$$\begin{bmatrix}\dot{x}\\ \ddot{x}\\ \dot{\Phi}\\ \ddot{\Phi}\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0\\ 0 & 108.69545 & -24.959145 & 0\\ 0 & 0 & 0 & 1\\ 0 & -112.387 & 85.0663 & 0\end{bmatrix}\begin{bmatrix}\dot{x}\\ \ddot{x}\\ \dot{\Phi}\\ \ddot{\Phi}\end{bmatrix} + \begin{bmatrix}0\\ 13.930735\\ 0\\ -20.58139\end{bmatrix} u$$

Half

$\dot{x}_1 = AX + BU$

$$\begin{bmatrix}\dot{x}\\ \ddot{x}\\ \dot{\Phi}\\ \ddot{\Phi}\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0\\ 0 & -108.69545 & 24.959145 & 0\\ 0 & 0 & 0 & 1\\ 0 & 105.039 & 35.148095 & 0\end{bmatrix}\begin{bmatrix}\dot{x}\\ \ddot{x}\\ \dot{\Phi}\\ \ddot{\Phi}\end{bmatrix} + \begin{bmatrix}0\\ 13.9307\\ 0\\ -7.281015\end{bmatrix} u$$

تعریف متغیرها:



- θ_1 زاویه لینک اول نسبت به بدنه
- θ_2 زاویه لینک دوم نسبت به بدنه
- ω_1 سرعت زاویه‌ای لینک اول
- ω_2 سرعت زاویه‌ای لینک دوم
- τ_1 گشتاور اعمال شده به لینک اول
- τ_2 گشتاور اعمال شده به لینک دوم

بردارهای حالت و ورودی:

بردار حالت x شامل زاویه‌ها و سرعت‌های زاویه‌ای لینک‌ها است.

$$x = [\theta_1\ \omega_1\ \theta_2\ \omega_2]^T$$

بردار ورودی u شامل گشتاورهای اعمال شده به لینک‌ها است.

$$u = [\tau_1 \, \tau_2]^T$$

بردار خروجی y وضعیت و سرعت لینک‌ها را نشان می‌دهد.

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

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

معادلات فضای حالت:

معادلات فضای حالت برای سیستم خطی شده به صورت زیر بیان می‌شوند:

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

$$y = Cx + Du$$

خطی‌سازی حول نقاط تعادل:

در این مقاله، دو مجموعه معادلات خطی برای دو حالت مختلف سیستم وجود دارد: حالت بالا و حالت میانی.

۱. **حالت بالا:**

در این حالت، بازوهای رباتیک در موقعیتی قرار دارند که لینک‌ها در وضعیت بالایی نسبت به بدنه هستند. معادلات خطی شده در این حالت با خطی‌سازی دینامیک سیستم حول نقطه تعادل در این موقعیت بدست آمده‌اند. این معادلات رفتار سیستم را در نزدیکی این نقطه تعادل توصیف می‌کنند و برای طراحی کنترل‌کننده‌های مناسب برای این وضعیت استفاده می‌شوند.

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\Phi} \\ \ddot{\Phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 108.69545 & -24.959145 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -112.387 & 85.0663 & 0 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\Phi} \\ \ddot{\Phi} \end{bmatrix} + \begin{bmatrix} 0 \\ 13.930735 \\ 0 \\ -20.58139 \end{bmatrix} u$$

۲. **حالت میانی:**

در این حالت، بازوهای رباتیک در موقعیتی قرار دارند که لینک‌ها در وضعیت میانی نسبت به بدنه هستند. معادلات خطی شده در این حالت نیز با خطی‌سازی دینامیک سیستم حول نقطه تعادل در این موقعیت بدست آمده‌اند. این معادلات رفتار سیستم را در نزدیکی این نقطه تعادل توصیف می‌کنند و برای طراحی کنترل‌کننده‌های مناسب برای این وضعیت استفاده می‌شوند.

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\Phi} \\ \ddot{\Phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -108.69545 & 24.959145 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 105.039 & 35.148095 & 0 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\Phi} \\ \ddot{\Phi} \end{bmatrix} + \begin{bmatrix} 0 \\ 13.9307 \\ 0 \\ -7.281015 \end{bmatrix} u$$

تفاوت خطی‌سازی در دو حالت:

- **حالت بالا :**در این حالت، سیستم به طور کلی پایداری بیشتری دارد و معادلات خطی شده ساده‌تر هستند. تأثیرات غیرخطی و تعاملات بین لینک‌ها در این حالت کمتر است.
- **حالت میانی :**در این حالت، سیستم پایداری کمتری دارد و معادلات خطی شده پیچیده‌تر هستند. تأثیرات غیرخطی و تعاملات بین لینک‌ها در این حالت بیشتر است و باید در طراحی کنترل‌کننده‌ها مدنظر قرار گیرند.

روش خطی‌سازی:

معادلات خطی شده در این مقاله با استفاده از روش تقریب خطی (approximate linearization) بدست آمده‌اند. این روش شامل محاسبه مشتقات جزئی معادلات دینامیکی غیرخطی نسبت به متغیرهای حالت و ورودی حول نقاط تعادل می‌باشد. با این خطی‌سازی، مدل خطی بدست آمده نشان‌دهنده رفتار دینامیکی سیستم در نزدیکی نقاط تعادل است و برای طراحی کنترل‌کننده‌های خطی مانند LQR مناسب است.

۲-۳ شبیه سازی در متلب

در این مسئله، سیستم با استفاده از متغیرهای زیر برای نمایش فضای حالت تعریف شده است:

$\theta_1,\theta_2,\theta_3,\theta_4,\theta_5$ این متغیرها پارامترهایی هستند که در مدل فضای حالت سیستم استفاده می‌شوند. هر کدام از این متغیرها یک مقدار ثابت را نمایان می‌کنند.

سیستم دو حالت "Up" و "Middle" دارد. برای هر یک از این حالات، ماتریس‌های A و B داده شده‌اند که معادله‌ی فضای حالت سیستم را مشخص می‌کنند. با استفاده از توابع ss درMATLAB ، سیستم خطی معادل با این ماتریس‌ها ساخته شده است.

```
sys_up = ss(A_up, B_up, eye(4), zeros(4,1));
```

در این بخش، سیستم معادل با حالت "Up" ساخته شده است. ماتریس Aup و بردار Bup به عنوان ورودی داده شده‌اند. خروجی sys_up یک سیستم خطی است که توسط MATLAB ساخته شده و مشخصات آن از جمله ماتریس حالت و ماتریس ورودی-خروجی نمایش داده می‌شود.

```
System for Up position:
sys_up =

  A =
            x1       x2       x3       x4
   x1        0        1        0        0
   x2        0        0    108.7   -112.4
   x3        0        0        0        1
   x4        0   -24.96    85.07        0

  B =
            u1
   x1        0
   x2    13.93
   x3        0
   x4   -20.58

  C =
        x1  x2  x3  x4
   y1    1   0   0   0
   y2    0   1   0   0
   y3    0   0   1   0
   y4    0   0   0   1

  D =
        u1
   y1    0
   y2    0
   y3    0
   y4    0

Continuous-time state-space model.
```

```
sys_mid = ss(A_mid, B_mid, eye(4), zeros(4,1));
```

در این بخش نیز سیستم معادل با حالت "Middle" ساخته شده است، با استفاده از ماتریس‌های Amid و Bmid مانند حالت قبل، خروجی sys_mid نشان دهنده‌ی سیستم خطی معادل است که MATLAB آن را ایجاد کرده است.

```
System for Middle position:
sys_mid =

  A =
             x1       x2       x3       x4
   x1         0        1        0        0
   x2         0        0   -108.7      105
   x3         0        0        0        1
   x4         0    24.96    35.15        0

  B =
             u1
   x1         0
   x2     13.93
   x3         0
   x4    -7.281

  C =
         x1   x2   x3   x4
   y1     1    0    0    0
   y2     0    1    0    0
   y3     0    0    1    0
   y4     0    0    0    1

  D =
         u1
   y1     0
   y2     0
   y3     0
   y4     0

Continuous-time state-space model.
```

```
disp('Eigenvalues for Up position:');
eig(A_up)


disp('Eigenvalues for Middle position:');
eig(A_mid)
```

این بخش مقادیر ویژه ماتریس Aup و Bup را نمایش می‌دهد. مقادیرویژه در تحلیل سیستم‌های دینامیکی مهم هستند زیرا خصوصیات اصلی سیستم را مشخص می‌کنند.

```
Eigenvalues for Up position:
ans = 4×1
         0
  -54.2234
   53.2845
    0.9390
Eigenvalues for Middle position:
ans = 4×1
         0
  -52.0476
   51.0261
    1.0215
```

۴- کنترل‌پذیری و رویت‌پذیری

کنترلپذیری و رویتپذیری نمایش فضای حالت بدست آمده را بررسی کنید؟ آیا امکان طراحی فیدبک حالت و تخمینگر حالت برای این نمایش وجود دارد؟ اگر نمایش فضای حالت بدست آمده کنترلناپذیر است آن را به زیرسیستمهای کنترلپذیر و کنترلناپذیر تفکیک کنید. همچنین اگر نمایش فضای حالت بدست آمده رویتناپذیر است آن را به زیرسیستمهای رویتپذیر و رویتناپذیر تفکیک کنید.

در این بخش از تحلیل، ما می‌خواهیم کنترل‌پذیری و رویت‌پذیری سیستم را بررسی کنیم. این تحلیل به ما اطلاعاتی ارائه می‌دهد که برای طراحی کنترل‌گرها و تخمین‌گرهای حالت بسیار حیاتی هستند. با استفاده از نمایش فضای حالت برای دو حالت مختلف "Up" و "Middle" سیستم، قصد داریم تا بررسی کنیم که آیا این حالات قابلیت کنترل و تخمین دقیق وضعیت حالت‌های سیستم را دارند یا خیر.

```
Mc_up = ctrb(A_up, B_up);
rank_Mc_up = rank(Mc_up);
disp(['Rank of controllability matrix (Up): ', num2str(rank_Mc_up)]);
if rank_Mc_up == size(A_up, 1)
   disp('The system is controllable (Up)');
else
   disp('The system is not controllable (Up)');
end
```

Mc_up ماتریس کنترل‌پذیری برای حالت"Up" ، که توسط تابع ctrb محاسبه شده است.

rank_Mc_up رتبه ماتریس کنترل‌پذیری برای حالت"Up" میباشد.

```
Controllability and Observability Analysis:
Up position:

Rank of controllability matrix (Up): 4
The system is controllable (Up)
```

```
Mo_up = obsv(A_up, eye(4));
rank_Mo_up = rank(Mo_up);
disp(['Rank of observability matrix (Up): ', num2str(rank_Mo_up)]);
if rank_Mo_up == size(A_up, 1)
   disp('The system is observable (Up)');
else
   disp('The system is not observable (Up)');
end
```

Mo_up   ماتریس رویت‌پذیری برای حالت"Up" ، که توسط تابع obsv محاسبه شده است .

rank_Mo_up   رتبه ماتریس رویت‌پذیری برای حالت"Up" میباشد.

```
Rank of observability matrix (Up): 4
The system is observable (Up)
```

پس به این صورت سیستم ما در حالت بالا هم رویت پذیر و هم کنترل پذیر میباشد، برای حالت میانی نیز به همین ترتیب اما این بار برای Middle تکرار میکنیم که نتایج در زیر آورده شده است.

```
Middle position:

Rank of controllability matrix (Middle): 4
The system is controllable (Middle)

Rank of observability matrix (Middle): 4
The system is observable (Middle)
```

تحلیل کنترل‌پذیری و رویت‌پذیری:

اگر سیستم کنترل‌پذیر باشد، این به معنای این است که می‌توان با استفاده از ورودی‌های موجود به طور کامل و بدون مشکلات، وضعیت حالت‌های سیستم را کنترل کرد. اگر سیستم رویت‌پذیر باشد، این به معنای این است که وضعیت حالت‌های سیستم با استفاده از خروجی‌های موجود به درستی قابل تخمین است.

تفکیک بین زیرسیستم‌های کنترل‌پذیر و کنترل‌ناپذیر، و رویت‌پذیر و رویت‌ناپذیر:

اگر سیستم کنترل‌پذیر باشد اما رویت‌ناپذیر باشد، بخشی از حالت‌های سیستم که از طریق خروجی‌های موجود قابل تخمین نیستند. اگر سیستم رویت‌پذیر باشد اما کنترل‌ناپذیر باشد، بخشی از حالت‌های سیستم که با ورودی‌های موجود قابل کنترل نیستند.

امکان طراحی فیدبک حالت و تخمین‌گر حالت:

اگر سیستم همزمان کنترل‌پذیر و رویت‌پذیر باشد، امکان طراحی کنترل‌گرها و تخمین‌گرهای حالت وجود دارد که به صورت موثر و دقیق از ورودی‌ها و خروجی‌های سیستم استفاده کنند.

۵- سیستم مینیمال

در صورتی که نمایش فضای حالت بدست آمده در قسمت قبل مینیمال نیست، یک نمایش فضای حالت مینیمال برای سیستم بدست آورید؟

در این بخش از تحلیل، ما به بررسی نمایش فضای حالت مینیمال برای سیستم‌هایی که در قسمت‌های قبلی بررسی شده‌اند، می‌پردازیم. هدف از این تحلیل، به دست آوردن یک نمایش فضای حالت ساده‌تر و مینیمال برای سیستم‌ها است که معمولاً اطلاعات کافی و کامل‌تری را درباره خصوصیات دینامیکی و عملکرد سیستم‌ها فراهم می‌آورد. این نمایش‌های فضای حالت اساسی هستند برای طراحی کنترل‌گرها و تخمین‌گرهای حالت که هدف آن‌ها بهینه‌سازی عملکرد و ایمنی سیستم است.

در اینجا از تابع `minimal_realization` استفاده می‌کنیم تا نمایش فضای حالت مینیمال را برای سیستم‌های "Up" و "Middle" به دست آوریم. این تحلیل شامل محاسبه ماتریس‌های A_min، B_min و C_min است که ساختار ساده‌تر و کاربردی‌تری از سیستم را نشان می‌دهند.

```
C_up = eye(4);
[A_min_up, B_min_up, C_min_up] = minimal_realization(A_up, B_up, C_up);

C_mid = eye(4);
[A_min_mid, B_min_mid, C_min_mid] = minimal_realization(A_mid, B_mid, C_mid);
```

این بخش از کد ماتریس‌های C را برای دو حالت "بالا" و "وسط" به صورت ماتریس واحد ۴*۴ تعریف می‌کند. سپس با استفاده از تابع minimal_realization ، نمایش فضای حالت مینیمال را برای هر دو حالت محاسبه می‌کند. این تابع ماتریس‌های A ، B و C اولیه را گرفته و ماتریس‌های مینیمال شده A_min ، B_min و C_min را برمی‌گرداند. این فرآیند برای کاهش پیچیدگی سیستم و حذف حالت‌های غیرقابل کنترل یا غیرقابل مشاهده انجام می‌شود.

```
Minimal Realization for Up position:

A_min_up:
   17.3368 -109.9351   -7.6333  -96.2077
  -23.3995  -18.2759  -11.6135  -97.8676
    0.0001    0.0001    0.4203    3.3717
    0.0000    0.0000    0.0647    0.5188

B_min_up:
  -16.5549
   18.5186
   -0.7773
    0.2288
```

```
C_min_up:
   -0.0052    0.0385    0.9916   -0.1236
   -0.9910   -0.1337    0.0011    0.0088
    0.0089    0.0009    0.1237    0.9923
    0.1336   -0.9903    0.0385   -0.0051



Minimal Realization for Middle position:

A_min_mid:
  -50.1129   81.9814  -61.0193   95.6491
    2.3865   49.0901   -6.2201   12.1109
   -0.0004   -0.0090    0.9757   -1.4236
    0.0000    0.0003   -0.0323    0.0471
B_min_mid:
   15.6940
   -0.8224
   -0.3106
   -0.0556
C_min_mid:
   -0.0164    0.0358   -0.8241   -0.5650
    0.9094    0.4159   -0.0055    0.0079
    0.0085    0.0044    0.5655   -0.8247
   -0.4156    0.9087    0.0322    0.0227
```

```
disp('Eigenvalues for Minimal Up position:');
eig(A_min_up)
disp('Eigenvalues for Minimal Middle position:');
eig(A_min_mid)
```

این بخش از کد مقادیر ویژه ماتریس‌های A_min را برای هر دو حالت محاسبه و نمایش می‌دهد. مقادیر ویژه اطلاعات مهمی در مورد پایداری و رفتار دینامیکی سیستم ارائه می‌دهند. مقادیر ویژه با قسمت حقیقی منفی نشان‌دهنده پایداری سیستم هستند، در حالی که مقادیر ویژه با قسمت حقیقی مثبت نشان‌دهنده ناپایداری هستند. همچنین، مقادیر ویژه مختلط نشان‌دهنده رفتار نوسانی سیستم هستند.

```
Eigenvalues for Minimal Up position:
ans = 4×1
   53.2845
  -54.2234
    0.9390
    0.0000

Eigenvalues for Minimal Middle position:
ans = 4×1
  -52.0476
   51.0261
    1.0215
    0.0000
```

```
sys_min_up = ss(A_min_up, B_min_up, C_min_up, 0);
sys_min_mid = ss(A_min_mid, B_min_mid, C_min_mid, 0);


disp('Minimal System for Up position:');
sys_min_up
disp('Minimal System for Middle position:');
sys_min_mid
```

در این قسمت نهایی، کد با استفاده از ماتریس‌های مینیمال شده، سیستم‌های فضای حالت مینیمال را برای هر دو حالت ایجاد می‌کند. این سیستم‌ها با استفاده از تابع ss ساخته می‌شوند و سپس نمایش داده می‌شوند. ماتریس D در این سیستم‌ها صفر در نظر گرفته شده است. این نمایش نهایی، یک توصیف کامل از سیستم‌های مینیمال شده را ارائه می‌دهد که شامل تمام اطلاعات لازم برای تحلیل‌های بیشتر و طراحی کنترل‌کننده است. این سیستم‌های مینیمال می‌توانند برای شبیه‌سازی، تحلیل پایداری، و طراحی کنترل‌کننده استفاده شوند.

```
Minimal System for Up position:
sys_min_up =

  A =
                x1         x2         x3         x4
   x1        17.34     -109.9     -7.633     -96.21
   x2        -23.4     -18.28     -11.61     -97.87
   x3    0.0001237  0.0001017     0.4203      3.372
   x4    1.903e-05  1.565e-05    0.06467     0.5188

  B =
              u1
   x1     -16.55
   x2      18.52
   x3    -0.7773
   x4     0.2288

  C =
                x1         x2         x3         x4
   y1    -0.005186    0.03851     0.9916    -0.1236
   y2       -0.991    -0.1337   0.001112   0.008831
   y3     0.008865  0.0008599     0.1237     0.9923
   y4       0.1336    -0.9903    0.03852  -0.005138

  D =
       u1
   y1   0
   y2   0
   y3   0
   y4   0

Continuous-time state-space model.
```

```
Minimal System for Middle position:
sys_min_mid =

  A =
                x1          x2          x3          x4
   x1       -50.11       81.98      -61.02       95.65
   x2        2.387       49.09       -6.22       12.11
   x3   -0.0004487   -0.009035      0.9757      -1.424
   x4    1.484e-05   0.0002989    -0.03228     0.04709

  B =
              u1
   x1      15.69
   x2    -0.8224
   x3    -0.3106
   x4   -0.05561

  C =
               x1         x2         x3         x4
   y1    -0.01644    0.03577    -0.8241     -0.565
   y2      0.9094     0.4159  -0.005468   0.007854
   y3    0.008499   0.004427     0.5655    -0.8247
   y4     -0.4156     0.9087    0.03219    0.02267

  D =
       u1
   y1   0
   y2   0
   y3   0
   y4   0

Continuous-time state-space model.
```

با استفاده از نمایش فضای حالت مینیمال، ما اطلاعات بیشتری درباره ساختار دینامیکی و خصوصیات سیستم‌ها به دست آورده‌ایم. ماتریس‌های A_min که شامل مقادیر ویژه مهمی هستند، امکان می‌دهند تا به طور دقیق‌تر و با دانش بیشتری روی تحلیل و طراحی سیستم‌های کنترلی متمرکز شویم.

با تحلیل و به دست آوردن نمایش فضای حالت مینیمال برای سیستم‌های "Up" و "Middle"، ما به یک پایه قوی‌تر برای تحلیل دقیق‌تر و طراحی بهینه‌تر کنترل‌گرها و تخمین‌گرهای حالت دست یافته‌ایم. این اطلاعات برای بهبود عملکرد و اعتماد به سیستم‌های پیچیده و مهندسی از اهمیت بالایی برخوردار هستند.

۶- فیدبک حالت

برای نمایش فضای حالت بدست آمده برای سیستم، فیدبک حالت را چنان طراحی کنید تا قطبهای سیستم حلقه بسته در مکانهای دلخواهی در سمت چپ محور jω قرار بگیرند. پاسخ پله و متغیرهای حالت سیستم را رسم کنید. قطبها و صفرهای سیستم حلقه بسته و حلقه باز را با یکدیگر مقایسه کنید .توجه: این مرحله را برای جانمایی قطبهای دور و نزدیک انجام دهید و سیگنال کنترلی و بهره فیدبک حالت بدست آمده در هر دو حالت را با یکدیگر مقایسه کنید.

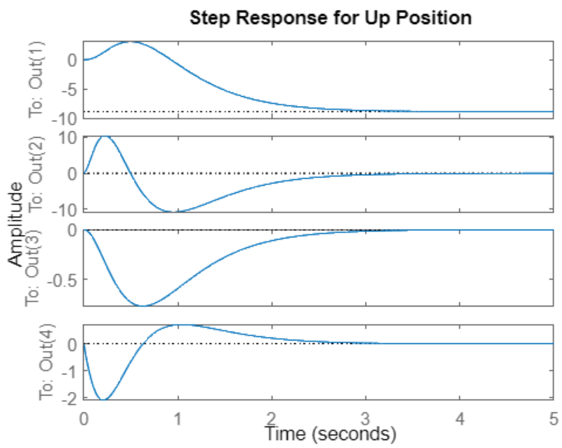
این سؤال به طراحی فیدبک حالت برای سیستم در دو موقعیت "بالا" و "وسط" می‌پردازد. هدف اصلی، جایگذاری قطب‌های سیستم حلقه بسته در مکان‌های مطلوب در سمت چپ محور jω است. این کار برای بهبود پایداری و عملکرد سیستم انجام می‌شود. سپس، پاسخ پله و متغیرهای حالت سیستم رسم می‌شوند تا رفتار دینامیکی سیستم را نشان دهند. در نهایت، قطب‌ها و صفرهای سیستم حلقه بسته و حلقه باز مقایسه می‌شوند تا تأثیر فیدبک حالت بر دینامیک سیستم مشخص شود. این فرآیند برای دو حالت قطب‌های دور و نزدیک انجام می‌شود تا تأثیر محل قطب‌ها بر سیگنال کنترلی و بهره فیدبک حالت بررسی شود.

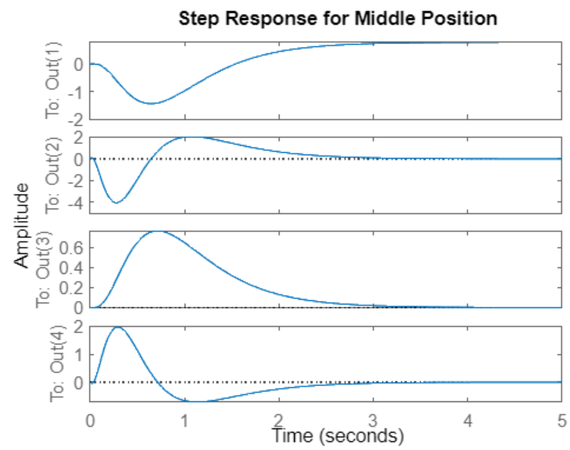
```
% Pole placement for Up position
p_up = [-2 -4 -6 -8];
K_up = place(A_up, B_up, p_up);
A_fb_up = A_up - B_up * K_up;
sys_fb_up = ss(A_fb_up, B_up, eye(4), zeros(4,1));

% Pole placement for Middle position
p_mid = [-2 -4 -6 -8];
K_mid = place(A_mid, B_mid, p_mid);
A_fb_mid = A_mid - B_mid * K_mid;
sys_fb_mid = ss(A_fb_mid, B_mid, eye(4), zeros(4,1));
```

این قسمت از کد به طراحی فیدبک حالت برای هر دو موقعیت می‌پردازد. ابتدا مکان‌های مطلوب برای قطب‌های سیستم حلقه بسته تعیین می‌شود (در اینجا[8- 6- 4- 2-]). سپس با استفاده از تابعplace ، ماتریس بهره فیدبک حالت K محاسبه می‌شود. ماتریس A جدید برای سیستم حلقه بسته با کم کردن حاصلضرب B و K از A اصلی به دست می‌آید. در نهایت، سیستم‌های فضای حالت جدید برای حالت حلقه بسته ایجاد می‌شوند.

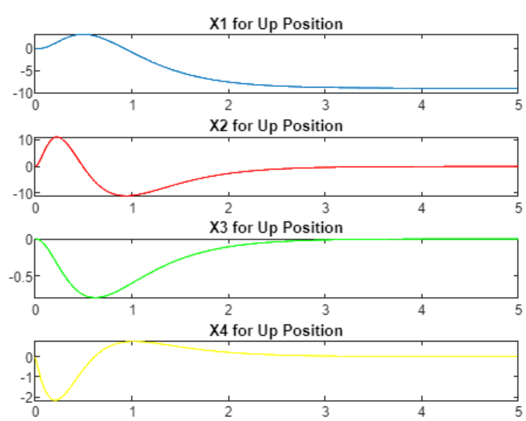
```
% Open loop poles
open_loop_pole_up = pole(sys_up);
open_loop_pole_mid = pole(sys_mid);
% Closed loop poles
close_loop_pole_up = pole(sys_fb_up);
close_loop_pole_mid = pole(sys_fb_mid);
```

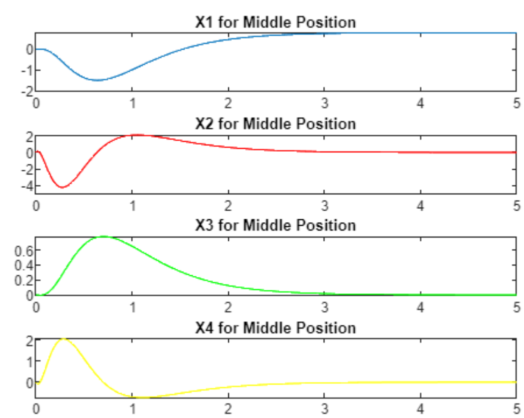
این بخش از کد قطب‌های سیستم‌های حلقه باز و حلقه بسته را برای هر دو موقعیت محاسبه می‌کند. این قطب‌ها برای مقایسه و ارزیابی تأثیر فیدبک حالت بر دینامیک سیستم استفاده خواهند شد.






```
% State variables over time
t = 0:0.01:5;
x_up = zeros(4, length(t));
x_mid = zeros(4, length(t));


% Simulate state responses
for i = 2:length(t)
    x_up(:, i) = x_up(:, i-1) + 0.01 * (A_fb_up * x_up(:, i-1) + B_up);
    x_mid(:, i) = x_mid(:, i-1) + 0.01 * (A_fb_mid * x_mid(:, i-1) + B_mid);
end
```

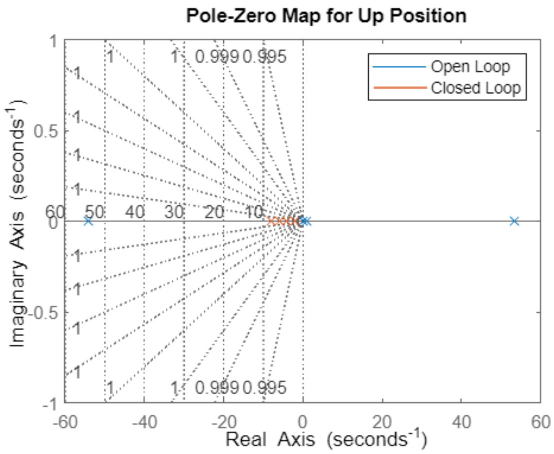


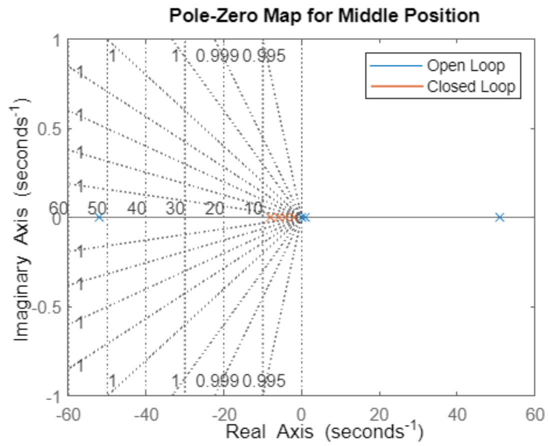


```
Open Loop Poles for Up Position:
         0
  -54.2234
   53.2845
    0.9390
Closed Loop Poles for Up Position:
   -8.0000
   -6.0000
   -4.0000
   -2.0000

Open Loop Poles for Middle Position:
         0
  -52.0476
   51.0261
    1.0215
Closed Loop Poles for Middle Position:
   -8.0000
   -6.0000
   -4.0000
   -2.0000
```

```
figure;
  pzmap(sys_up);
    hold on;
  pzmap(sys_fb_up);
    title('Pole-Zero Map for Up Position');
    legend('Open Loop', 'Closed Loop');
    grid on

figure;
  pzmap(sys_mid);
    hold on;
  pzmap(sys_fb_mid);
    title('Pole-Zero Map for Middle Position');
    legend('Open Loop', 'Closed Loop');
    grid on
```

این بخش از کد به ترسیم نقشه‌های قطب-صفر برای سیستم‌ها در دو موقعیت "بالا" و "وسط" می‌پردازد. با استفاده از تابع pzmap، نقشه قطب-صفر برای سیستم حلقه باز (sys_up یا sys_mid) رسم می‌شود. در ادامه، نقشه قطب-صفر برای سیستم حلقه بسته (sys_fb_up یا sys_fb_mid) روی همان نمودار رسم می‌شود. این نمودارها امکان مقایسه بصری موقعیت قطب‌ها و صفرها را در حالت‌های حلقه باز و حلقه بسته برای هر دو موقعیت فراهم می‌کنند، که برای تحلیل تأثیر فیدبک حالت بر دینامیک سیستم بسیار مفید است.





۷- طراحی ردیاب استاتیک

برای سیستم مورد بررسی یک ردیاب استاتیک طراحی کنید. عملکرد این ردیاب را در برابر حضور اغتشاشهای مختلف و عدم قطعیت در مدل سیستم ارزیابی کنید.

- نکته ای که وجود دارد این است که از این جا به بعد برای عدم تکرار و سادگی کار تنها **حالت بالایی** محاسبه میشود.

این سؤال به طراحی یک ردیاب استاتیک برای سیستم مورد بررسی و ارزیابی عملکرد آن در برابر اغتشاشات مختلف و عدم قطعیت در مدل سیستم می‌پردازد. هدف اصلی، ایجاد یک سیستم کنترلی است که بتواند ورودی مرجع را به خوبی دنبال کند، حتی در حضور عوامل مختل‌کننده خارجی و عدم دقت در مدل‌سازی سیستم.

روند حل این مسئله شامل چند مرحله اصلی است:

۱. طراحی ردیاب استاتیک با استفاده از فیدبک حالت

۲. ارزیابی عملکرد سیستم در حضور اغتشاشات

۳. بررسی تأثیر عدم قطعیت مدل بر عملکرد سیستم

۷-۱ طراحی ردیاب استاتیک

در این قسمت ابتدا باید معکوس تابع تبدیل به عنوان ضریب P محاسبه کنیم.

$$P = inv(C * inv(-A\_fb) * B)$$

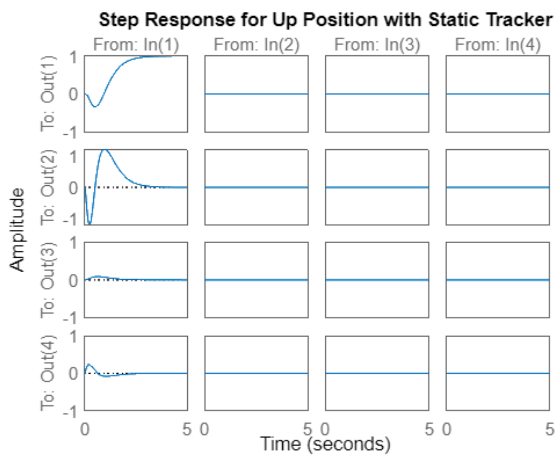
```
A_up = [0 1 0 0;
       0 0 108.69545 -112.387;
       0 0 0 1;
       0 -24.959145 85.0663 0];
B_up = [0; 13.930735; 0; -20.58139];
C_up = eye(4);
D_up = zeros(4, 1);
% Desired pole locations for Up position
p_up = [-2 -4 -6 -8];
K_up = place(A_up, B_up, p_up);
A_fb_up = A_up - B_up * K_up;

% Static state feedback tracker design
P_up = pinv(C_up * inv(-A_fb_up) * B_up)
B_st_up = B_up * P_up

% Adjust D_up to have the same number of columns as B_st_up
D_up = zeros(size(C_up, 1), size(B_st_up, 2));

% State-space system with state feedback
sys_fb_st_up = ss(A_fb_up, B_st_up, C_up, D_up);
```

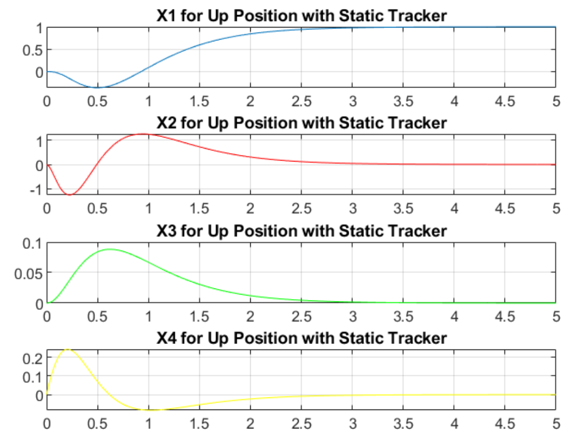
این بخش از کد به تعریف ماتریس‌های سیستم و طراحی ردیاب استاتیک می‌پردازد. ابتدا ماتریس‌های A ، B، C و D برای موقعیت "بالا" تعریف می‌شوند. سپس، با استفاده از تابع place ، ماتریس بهره فیدبک حالت K محاسبه می‌شود. ماتریس A جدید برای سیستم حلقه بسته (A_fb_up) محاسبه می‌شود. در ادامه، ماتریس P برای ردیاب استاتیک با استفاده از معکوس مور-پنروز محاسبه می‌شود و ماتریس B جدید (B_st_up) برای سیستم با ردیاب استاتیک به دست می‌آید. در نهایت، سیستم فضای حالت جدید با فیدبک حالت و ردیاب استاتیک ایجاد می‌شود.

```
P_up = 1×4
   -0.1122         0         0         0

B_st_up = 4×4
         0         0         0         0
   -1.5632         0         0         0
         0         0         0         0
    2.3094         0         0         0
```

```
% Simulation parameters
t = 0:0.01:5;
x_st_up = zeros(4, length(t));
% Simulate state responses with static tracker
for i = 2:length(t)
   x_st_up(:, i) = x_st_up(:, i-1) + 0.01 * (A_fb_up * x_st_up(:, i-1) + B_st_up(:, 1));
end
```

این بخش از کد به شبیه‌سازی پاسخ سیستم با ردیاب استاتیک می‌پردازد. ابتدا پارامترهای شبیه‌سازی تعریف می‌شوند و سپس با استفاده از یک حلقهfor ، پاسخ حالت سیستم محاسبه می‌شود. در نهایت، پاسخ پله سیستم با استفاده از تابع step رسم می‌شود.

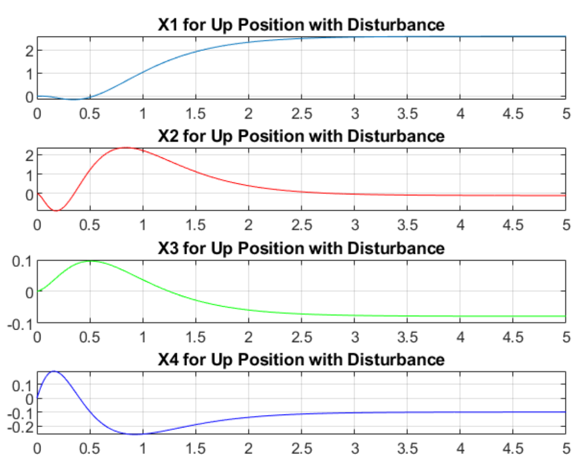




۷-۲ عملکرد سیستم در حضور اغتشاشات

```
% Evaluate performance in presence of disturbances
% Adding disturbances to the system
disturbance = [0.1; 0.05; 0.1; 0.05];

x_dist_up = zeros(4, length(t));

for i = 2:length(t)
   x_dist_up(:, i) = x_dist_up(:, i-1) + 0.01 * (A_fb_up * x_dist_up(:, i-1) + B_st_up(:, 1) + disturbance);
end
```

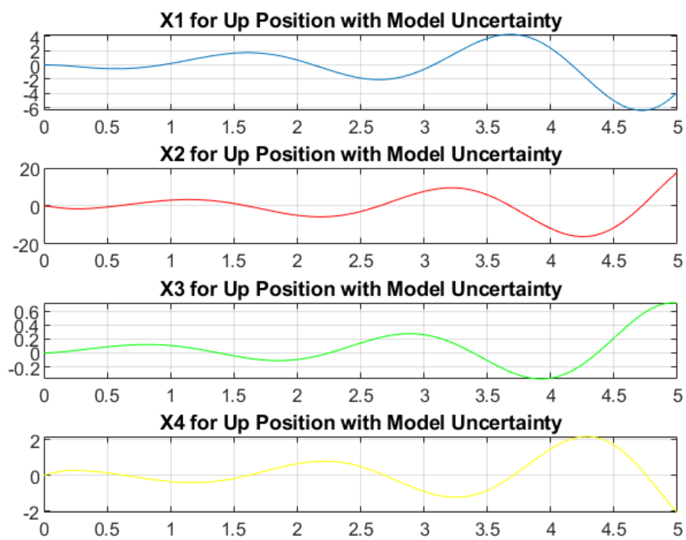
این بخش از کد به ارزیابی عملکرد سیستم در حضور اغتشاشات می‌پردازد. یک بردار اغتشاش نمونه تعریف می‌شود و پاسخ سیستم در حضور این اغتشاش محاسبه می‌شود.



۳-۷ عدم قطعیت مدل بر عملکرد سیستم

```
% Evaluate performance in presence of model uncertainty
% Adding model uncertainty to the system
A_uncertain_up = A_fb_up + 0.05 * randn(size(A_fb_up));

x_uncertain_up = zeros(4, length(t));

for i = 2:length(t)
    x_uncertain_up(:, i) = x_uncertain_up(:, i-1) + 0.01 * (A_uncertain_up * x_uncertain_up(:, i-1) + B_st_up(:,
1));
end
```

این قسمت از کد به بررسی تأثیر عدم قطعیت مدل بر عملکرد سیستم می‌پردازد. یک ماتریس A جدید با اضافه کردن نویز تصادفی به ماتریس A اصلی ایجاد می‌شود و پاسخ سیستم با این ماتریس نامطمئن محاسبه می‌شود.



در نهایت، نتایج نشان می‌دهد که:

۱. ردیاب استاتیک طراحی شده قادر است سیستم را به خوبی کنترل کند و پاسخ پله مناسبی را ایجاد کند.

۲. در حضور اغتشاشات، سیستم همچنان عملکرد نسبتاً خوبی دارد، اما انحرافاتی از حالت ایده‌آل مشاهده می‌شود.

۳. عدم قطعیت در مدل سیستم باعث تغییراتی در پاسخ سیستم می‌شود، اما سیستم همچنان پایدار باقی می‌ماند.

۸- ردیاب انتگرالی

برای سیستم مورد بررسی یک ردیاب انتگرالی طراحی کنید. عملکرد این ردیاب را در برابر حضور اغتشاشهای مختلف و عدم قطعیت در مدل سیستم ارزیابی کنید.

برای طراحی ردیاب انتگرالی دو شرط باید چک شود. اول آنکه ماتریس کنترل پذیر باشد سپس ماتریس $\begin{bmatrix} A & B \\ 0 & -C \end{bmatrix}$ Full Rank باشد.

همانطور که مشاهده می شود ماتریس کنترل پذیر است، اما شرط دوم برقرار نیست .در نتیجه امکان طراحی ردیاب انتگرالی وجود ندارد.

```
% Check controllability of the original system
Mc_up = ctrb(A_up, B_up);
if rank(Mc_up) == size(A_up, 1)
    disp('Original system is controllable');
else
    disp('Original system is not controllable');
    return; % Exit the script if the system is not controllable
end

% Check integral controllability
n = size(A_up, 1); % Number of states
m = size(C_up, 1); % Number of outputs

augmented_matrix = [B_up, A_up; zeros(m, 1), C_up];
if rank(augmented_matrix) == n + m
    disp('System is integrally controllable');
else
    disp('System is not integrally controllable');
    return; % Exit the script if the system is not integrally controllable
end
```

```
Original system is controllable

System is not integrally controllable
```

پس امکان طراحی ردیاب انتگرالی وجود ندارد.

- در غیر این صورت کد پاسخ به این سوال در فایل متلب آورده شده است که میتوانید مشاهده کنید.

۹- طراحی تخمینگر مرتبه کامل

برای نمایش فضای حالت بدست آمده برای سیستم، یک تخمینگر مرتبه کامل طراحی کنید. ملاک انتخاب قطبهای تخمینگر حالت چیست؟ متغیرهای حالت اصلی و تخمین زده شده سیستم و خطای تخمین را رسم کنید.

این سؤال به طراحی یک تخمینگر مرتبه کامل برای سیستم مورد بررسی می‌پردازد. هدف اصلی، تخمین متغیرهای حالت سیستم با استفاده از اطلاعات خروجی است. سؤال همچنین به بررسی ملاک انتخاب قطب‌های تخمینگر حالت و مقایسه عملکرد تخمینگرهای کند و سریع می‌پردازد.

ملاک انتخاب قطب‌های تخمینگر حالت :قطب‌های تخمینگر حالت باید به گونه‌ای انتخاب شوند که(تخمینگر پایدار باشد (قطب‌ها در سمت چپ محور موهومی قرار گیرند) سرعت همگرایی تخمین به مقدار واقعی مناسب باشد (معمولاً ۲ تا ۵ برابر سریع‌تر از دینامیک سیستم اصلی) نویز و اغتشاشات سیستم را به خوبی فیلتر کند. در طراحی رویتگر هرچه قطب ها پایدارتر طراحی شوند به همان اندازه تخمین سریع تر رخ خواهد داد حال با این دانش یک بار با قطب های نزدیک و یک بار با قطب های دور طراحی را انجام میدهیم.

```
K = place(A_up, B_up, [-1, -2, -3, -4]);
A_stable = A_up - B_up * K;
```

این قسمت به طراحی فیدبک حالت برای پایدارسازی سیستم می‌پردازد. تابع place برای محاسبه ماتریس بهره فیدبک حالت K استفاده می‌شود. قطب‌های مطلوب [-۱, -۲, -۳, -۴] انتخاب شده‌اند که همگی در سمت چپ محور موهومی قرار دارند و پایداری سیستم را تضمین می‌کنند A_stable ماتریس حالت جدید برای سیستم حلقه بسته است. این ماتریس نشان می‌دهد که دینامیک سیستم چگونه پس از اعمال فیدبک حالت تغییر می‌کند.

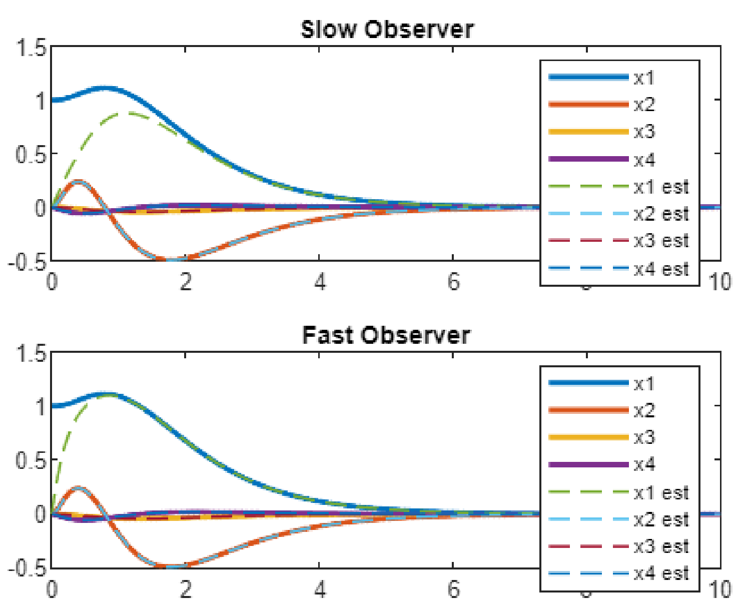
```
P_slow = [-1.5, -2.5, -3.5, -4.5];
L_slow = place(A_stable', C_up', P_slow)'

P_fast = [-5, -6, -7, -8];
L_fast = place(A_stable', C_up', P_fast)'
```

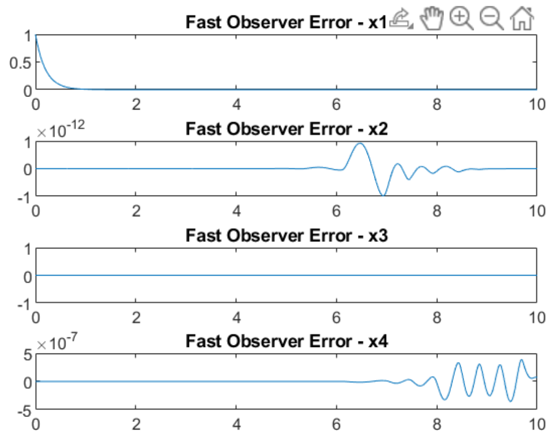
```
L_slow = 4×4
    1.5000    1.0000         0         0
    0.0977  -15.2958  177.8041 -117.6637
         0         0    3.5000    1.0000
   -0.1443    1.3326  -17.0354   12.2958

L_fast = 4×4
    5.0000    1.0000         0         0
    0.0977  -11.7958  177.8041 -117.6637
         0         0    7.0000    1.0000
   -0.1443    1.3326  -17.0354   15.7958
```

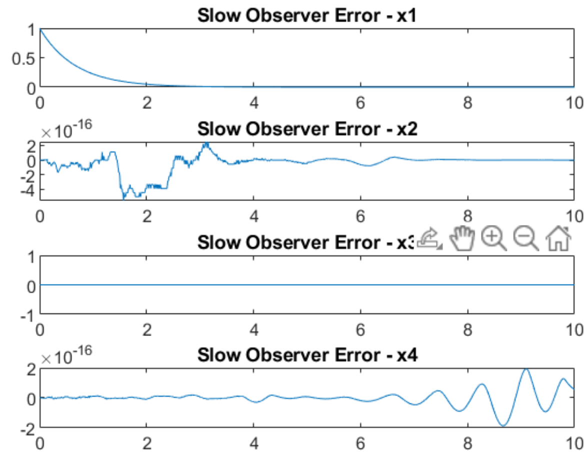
این بخش به طراحی دو تخمینگر حالت می‌پردازد: یکی کند و دیگری سریع P_slow و P_fast قطب‌های مطلوب برای تخمینگرهای کند و سریع هستند. قطب‌های تخمینگر کند نزدیک به قطب‌های سیستم اصلی هستند، در حالی که قطب‌های تخمینگر سریع دورتر از محور موهومی قرار دارند. این انتخاب باعث می‌شود که تخمینگر سریع، سریع‌تر به مقدار واقعی همگرا شود، اما ممکن است حساسیت بیشتری به نویز داشته باشد.L_slow و L_fast ماتریس‌های بهره تخمینگر هستند که با استفاده از تابع place محاسبه می‌شوند. توجه کنید که از ترانهاده A_stable و C_up استفاده شده، زیرا معادلات تخمینگر به صورت دوگان با معادلات سیستم اصلی هستند .

```
t = 0:0.01:10;
x0 = [1; 0; 0; 0];
x0_est = [0; 0; 0; 0];

[t, x_slow] = ode45(@(t, x) observer_dynamics(t, x, A_stable, B_up, C_up, L_slow, K), t, [x0; x0_est]);

[t, x_fast] = ode45(@(t, x) observer_dynamics(t, x, A_stable, B_up, C_up, L_fast, K), t, [x0; x0_est]);
```

این قسمت به شبیه‌سازی سیستم و تخمینگرها می‌پردازد. بازه زمانی شبیه‌سازی از ۰ تا ۱۰ ثانیه با گام‌های ۰.۰۱ ثانیه تعریف شده است . x0شرایط اولیه سیستم واقعی و x0_est شرایط اولیه تخمینگر را مشخص می‌کند. توجه کنید که شرایط اولیه تخمینگر با شرایط اولیه واقعی متفاوت است، که این امر باعث می‌شود بتوانیم عملکرد تخمینگر را در همگرا شدن به مقادیر واقعی بررسی کنیم. تابع ode45 یک حل‌کننده عددی برای معادلات دیفرانسیل است که در اینجا برای حل همزمان معادلات سیستم و تخمینگر استفاده می‌شود. تابع observer_dynamics معادلات دینامیکی سیستم و تخمینگر را تعریف می‌کند.







نمودارهای متغیرهای حالت واقعی و تخمین زده شده را برای هر دو تخمینگر کند و سریع رسم می‌کند. در هر نمودار، خطوط پیوسته ضخیم‌تر نشان‌دهنده متغیرهای حالت واقعی و خطوط خط‌چین نازک‌تر نشان‌دهنده متغیرهای حالت تخمین زده شده هستند. می‌توانید ببینید که چگونه تخمین‌ها به مرور زمان به مقادیر واقعی نزدیک می‌شوند و تفاوت در سرعت همگرایی بین تخمینگر کند و سریع را مشاهده کنید. قسمت بعد خطای تخمین را برای تخمینگر کند رسم می‌کند. خطای تخمین برای هر متغیر حالت به صورت جداگانه نمایش داده می‌شود. این نمودارها نشان می‌دهند که چگونه خطای تخمین برای هر متغیر حالت به مرور زمان کاهش می‌یابد. شما می‌توانید سرعت همگرایی و هرگونه نوسان یا اورشوت در فرآیند تخمین را مشاهده کنید. بخش آخر نیز خطای تخمین را برای تخمینگر سریع رسم می‌کند. مشابه قسمت قبل، خطای تخمین برای هر متغیر حالت به صورت جداگانه نمایش داده می‌شود. با مقایسه این نمودارها با نمودارهای تخمینگر کند، می‌توانید تفاوت در سرعت همگرایی و رفتار گذرای دو تخمینگر را مشاهده کنید. انتظار می‌رود که تخمینگر سریع، خطای تخمین را سریع‌تر به صفر برساند، اما ممکن است در ابتدا نوسانات بیشتری داشته باشد.

۱۰ – طراحی تخمینگر کاهش مرتبه

برای نمایش فضای حالت بدست آمده برای سیستم، یک تخمینگر کاهش مرتبه طراحی کنید. متغیرهای حالت اصلی و تخمین زده شده سیستم و خطای تخمین را رسم کنید.

برای طراحی این رویتگر، ابتدا باید بررسی کنیم که آیا ماتریس (C) به فرم $[I_q \quad 0]$ است یا خیر. به وضوح، این شرط برقرار نیست؛ بنابراین، لازم است از تبدیل‌هایی استفاده کنیم تا سیستم مطلوب را به دست آوریم.

تبدیل مورد نظر، همان‌طور که در شرح درس گفته شد، شامل استفاده از ماتریس (C) و یک ماتریس دلخواه دیگر برای ساخت یک ماتریس مرتبه کامل است که مرتبه آن برابر با (n) باشد. این فرآیند به ما کمک می‌کند تا سیستم جدیدی را تعریف کنیم که دارای خصوصیات مورد نظر برای طراحی رویتگر است.

ابتدا یک ماتریس دلخواه (T) را انتخاب می‌کنیم که همراه با (C) یک ماتریس مرتبه کامل تشکیل دهد. این ماتریس باید به گونه‌ای انتخاب شود که ماتریس ترکیبی [C ; T] دارای رتبه کامل باشد و تمام خصوصیات مشاهده‌پذیری سیستم را حفظ کند. برای انجام این تبدیل‌ها، ماتریس تبدیل (P) را محاسبه می‌کنیم که ماتریس‌های سیستم اصلی را به سیستم جدید تبدیل می‌کند. این ماتریس تبدیل به ما کمک می‌کند تا حالت‌های جدید و ماتریس‌های جدید سیستم را بدست آوریم. با استفاده از سیستم جدید، رویتگر مورد نظر را طراحی می‌کنیم. رویتگر باید به گونه‌ای طراحی شود که حالت‌های سیستم را به طور دقیق تخمین بزند و خطای تخمین را به حداقل برساند. بهره‌های رویتگر (L) با توجه به قطب‌های مورد نظر سیستم انتخاب می‌شوند تا سرعت و دقت تخمین افزایش یابد.

```
% Check the rank of P
disp('Rank of P:');
disp(rank(P));
```

این قسمت رتبه ماتریس P را محاسبه و نمایش می‌دهد. رتبه P باید برابر با تعداد حالت‌های سیستم باشد تا تبدیل سیستم امکان‌پذیر باشد. در این مورد، چون P یک ماتریس واحد ۴*۴ است، رتبه آن ۴ خواهد بود.

```
Rank of P:
     4
```

```
% Transform the system
Abar = A;
Bbar = B;
Cbar = C;

% Partition the transformed system
A11 = Abar(1, 1);
A12 = Abar(1, 2:end);
A21 = Abar(2:end, 1);
A22 = Abar(2:end, 2:end);
B1 = Bbar(1);
B2 = Bbar(2:end);
```

این قسمت سیستم تبدیل شده را به زیرماتریس‌های مورد نیاز برای طراحی تخمینگر کاهش مرتبه تقسیم می‌کند . A11، A12، A21و A22 زیرماتریس‌های A هستند، در حالی که B1 و B2 بخش‌های تقسیم شده ماتریس B هستند. این تقسیم‌بندی برای ساخت معادلات تخمینگر ضروری است.

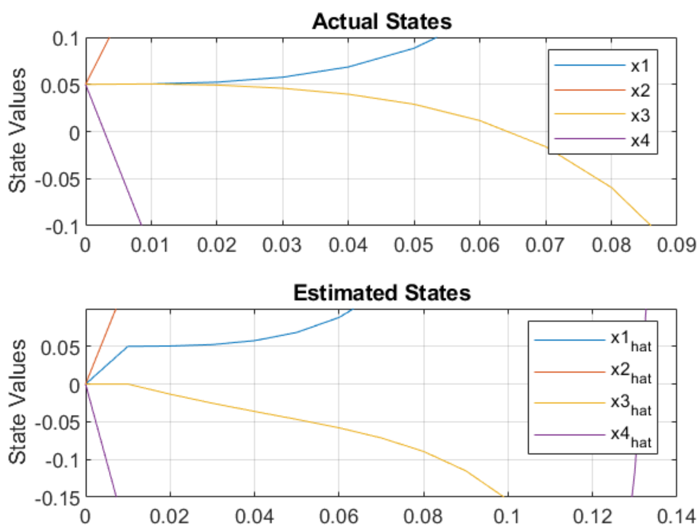
```
% Ensure A22 rank is non-zero
if rank(A22) == 0
    error('Rank of A22 is zero, cannot proceed with observer design.');
end

% Desired poles for the reduced-order observer
P_desire = [-5 -6 -7];

% Calculate the observer gain
K = place(A22', A12', P_desire)';
L = K;

disp('Observer gain L:');
disp(L);
```

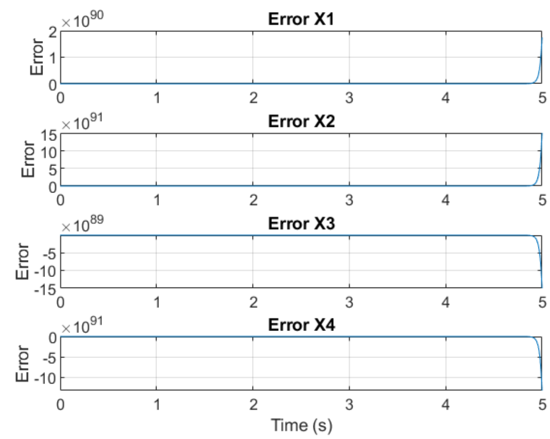
این بخش رتبه ماتریس A22 را بررسی می‌کند. اگر رتبه A22 صفر باشد، طراحی تخمینگر امکان‌پذیر نیست و برنامه با یک پیام خطا متوقف می‌شود. این بررسی برای اطمینان از امکان‌پذیر بودن طراحی تخمینگر ضروری است. این بخش قطب‌های مطلوب برای تخمینگر کاهش مرتبه را تعیین می‌کند و سپس با استفاده از تابع place ، بهره تخمینگر (L) را محاسبه می‌کند. قطب‌های انتخاب شده (-۵، -۶ و -۷) سرعت همگرایی تخمینگر را تعیین می‌کنند. بهره محاسبه شده برای اطمینان از پایداری و عملکرد مطلوب تخمینگر استفاده می‌شود.

```
Observer gain L:
   18.0000
   -0.2038
  -26.8659
```

```
% Initial conditions
xn = [0.05; 0.05; 0.05; 0.05];
tspan = 0:0.01:5;
ct = length(tspan);
z = zeros(3, ct);
y = zeros(4, ct);

% Simulate the system
for k = 2:ct
    y(:, k) = Cbar * xn(:, k-1);
    xn(:, k) = xn(:, k-1) + 0.01 * (Abar * xn(:, k-1) + Bbar);
    z(:, k) = z(:, k-1) + 0.01 * ((A22 - L * A12) * z(:, k-1) + ((A22 - L * A12) * L + A21) * y(1, k-1) + (B2 - L *
B1));
end

% Calculate the estimated state
x_hat = [y(1, :); z];
```

این قسمت شرایط اولیه برای شبیه‌سازی را تعیین می‌کند xn شرایط اولیه برای حالت‌های سیستم است این حلقه سیستم و تخمینگر را شبیه‌سازی می‌کند. در هر تکرار، خروجی سیستم(y) ، حالت‌های واقعی سیستم (xn) و حالت‌های تخمین زده شده (z) به‌روزرسانی می‌شوند. روش اویلر برای حل معادلات دیفرانسیل استفاده شده است. این شبیه‌سازی امکان مقایسه عملکرد سیستم واقعی و تخمینگر را فراهم می‌کند. این خط حالت‌های تخمین زده شده کامل (x_hat) را محاسبه می‌کند. حالت اول مستقیماً از خروجی سیستم گرفته می‌شود، در حالی که سه حالت دیگر از تخمینگر کاهش مرتبه (z) به دست می‌آیند.





نمودار بالایی حالت‌های واقعی سیستم را نشان می‌دهد، در حالی که نمودار پایینی حالت‌های تخمین زده شده را نمایش می‌دهد.

۱۱- رگولاتور با تخمینگر مرتبه کامل

سیستم حلقه بسته را با طراحی فیدبک حالتهای تخمین زده شده چنان طراحی کنید که قطبهای سیستم حلقه بسته سمت چپ محور jω بوده و پاسخ پله به لحاظ فراجهش و زمان نشست رفتار قابل قبولی داشته باشد. پاسخ پله و متغیرهای حالت سیستم را رسم کنید. در واقع فرض کنید، حالتهای سیستم در دسترس نبوده و باید از حالتهای تخمین زده شده برای فیدبک حالت استفاده شود.

این قسمت به طراحی یک سیستم کنترل حلقه بسته با استفاده از فیدبک حالت‌های تخمین زده شده می‌پردازد. هدف این است که قطب‌های سیستم حلقه بسته در سمت چپ محور jω قرار گیرند و پاسخ پله سیستم از نظر

فراجهش و زمان نشست رفتار مطلوبی داشته باشد. طبق خواسته این کار را انجام میدهیم اما به این موضوع توجه داشته باشید که برای بهتر و سریع تر بودن تخمینگر نیاز داریم تا قطب‌های تخمینگر دورتر از قطب‌های رگولاتور باشند تا سیستم بتواند با دقت مناسبی تخمین بزند حال نتیجه را بررسی میکنیم.

```
% Design state feedback controller
K = place(A, B, [-5, -6, -7, -8]);
L = place(A', C', [-50, -60, -70, -80])';
```

این بخش به طراحی کنترل‌کننده فیدبک حالت و رؤیتگر می‌پردازد. تابع place برای تعیین بهره‌های K (برای کنترل‌کننده) و L (برای رؤیتگر) استفاده می‌شود. قطب‌های انتخاب شده برای کنترل‌کننده ([-5, -6, -7, -8]) و رؤیتگر ([-50, -60, -70, -80]) باعث می‌شوند که سیستم حلقه بسته پایدار باشد و رؤیتگر سریع‌تر از سیستم اصلی همگرا شود.

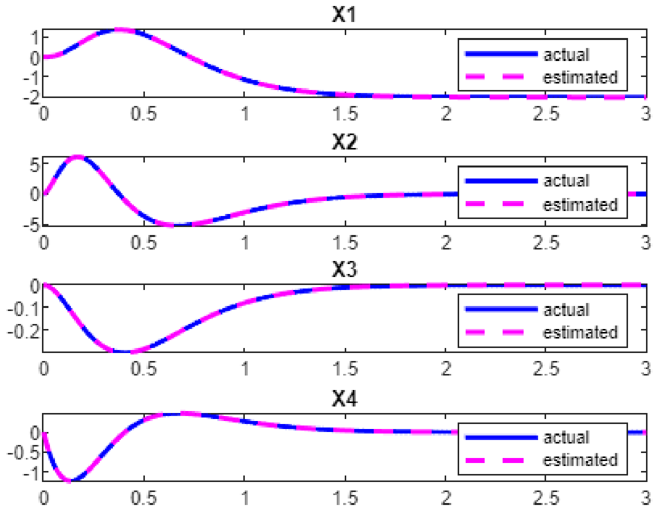
```
K = 1×4
   -0.4909    1.2013  -10.3522   -0.4502

L = 4×4
   50.0000    1.0000         0         0
         0   60.0000  108.7000 -112.4000
         0         0   70.0000    1.0000
         0  -24.9600   85.0700   80.0000
```

```
% Augmented system matrices for observer-based state feedback
A_aug = [A - B*K, zeros(4); L*C, A - L*C - B*K];
B_aug = [B; B];
C_aug = [zeros(1,4), C(1,:)];
D_aug = 0;
```

این قسمت ماتریس‌های سیستم افزوده را برای کنترل فیدبک حالت مبتنی بر رؤیتگر ایجاد می‌کند. سیستم افزوده شامل دینامیک سیستم اصلی و رؤیتگر است. A_aug ماتریس حالت افزوده، B_aug ماتریس ورودی افزوده، C_aug ماتریس خروجی افزوده و D_aug ماتریس انتقال مستقیم افزوده هستند.

```
% Simulate the augmented system
sys_aug = ss(A_aug, B_aug, C_aug, D_aug);
r = ones(size(tspan));
[y_aug, ~, x_aug] = lsim(sys_aug, r, tspan, x0);
```

این قسمت سیستم افزوده را شبیه‌سازی می‌کند. ابتدا یک مدل فضای حالت (sys_aug) از سیستم افزوده ایجاد می‌شود. سپس با استفاده از تابع lsim، پاسخ سیستم به یک ورودی پله واحد (r) شبیه‌سازی می‌شود y_aug خروجی سیستم و x_aug حالت‌های سیستم در طول زمان هستند.



این بخش پایانی نتایج شبیه‌سازی را در چهار زیرنمودار نمایش می‌دهد. هر زیرنمودار یکی از حالت‌های سیستم را نشان می‌دهد، که در آن مقدار واقعی (آبی) و مقدار تخمین زده شده (صورتی خط‌چین) با هم مقایسه شده‌اند.

این کد یک سیستم کنترل حلقه بسته را با استفاده از فیدبک حالت‌های تخمین زده شده طراحی و شبیه‌سازی می‌کند. نتایج نشان می‌دهند که چگونه حالت‌های تخمین زده شده به حالت‌های واقعی سیستم نزدیک می‌شوند، که نشان‌دهنده عملکرد مؤثر رؤیتگر است. همچنین، پاسخ سیستم به ورودی پله را می‌توان از این نمودارها مشاهده کرد، که می‌تواند برای ارزیابی عملکرد سیستم کنترل از نظر فراجهش و زمان نشست استفاده شود.

۱۲- طراحی LQR

بهره فیدبک حالت بهینه سیستم را برای حداقل سازی تابع هزینه زیر برای مقادیر مختلف ماتریس R و ماتریس Q بدست آورید.

$$J = \int (x^T Qx + u^T Ru)dt$$

یک بار ماتریس Q را ثابت فرض کنید و ماتریس R تغییر دهید. نتایج شبیهسازی را با یکدیگر مقایسه کنید. بار دیگر ماتریس R را ثابت در نظر بگیرید و ماتریس Q را تغییر دهید. نتایج شبیهسازی را با یکدیگر مقایسه کنید.

در این سوال، هدف طراحی یک کنترل‌کننده بهینه با استفاده از روش LQR (Linear Quadratic Regulator) برای حداقل‌سازی تابع هزینه J است. تابع هزینه به صورت زیر تعریف شده است:

$$J = \int (x^T Qx + u^T Ru)dt$$

که در آن Q ماتریس وزن‌دهی به حالات سیستم و R ماتریس وزن‌دهی به ورودی کنترل است. دو حالت در نظر گرفته شده است:

۱. تغییر ماتریس R با ماتریس Q ثابت.
۲. تغییر ماتریس Q با ماتریس R ثابت.

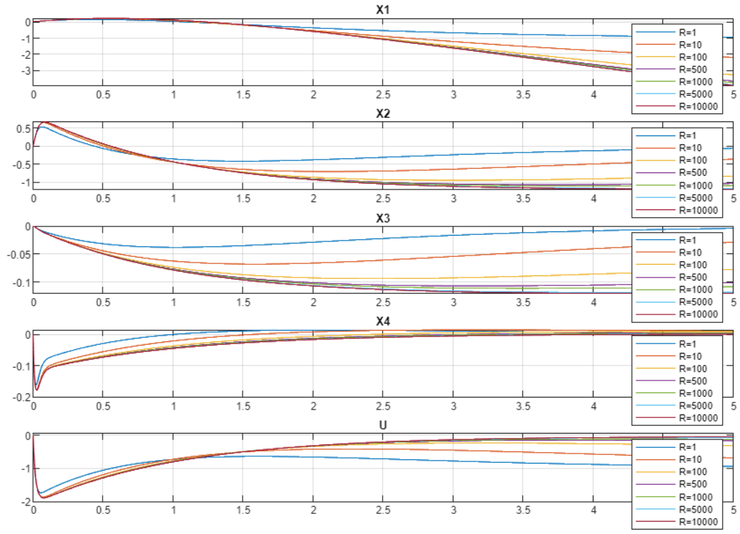
۱۲-۱ Q ثابت و R متغیر

```
% Case 1: Varying R with constant Q
Q = eye(4);
R_values = [1, 10, 100, 500, 1000, 5000, 10000];


figure('Position', [100, 100, 1200, 800]);


for i = 1:length(R_values)
    R = R_values(i);
    [K,~,P] = lqr(A, B, Q, R);
    sys = ss(A-B*K, B, C, D);
    [y,~,x] = lsim(sys, ones(size(t)), t, x0);
    u = -K * x';
```

این بخش به بررسی اثر تغییرات ماتریس R (وزن‌دهی به ورودی کنترلی) در حالی که ماتریس Q (وزن‌دهی به حالت‌ها) ثابت است، می‌پردازد. برای هر مقدار R، کنترل‌کننده LQR طراحی شده و سیستم حلقه بسته شبیه‌سازی می‌شود. نتایج برای هر حالت و ورودی کنترلی رسم می‌شوند.

تابع lqr برای محاسبه بهره فیدبک حالت بهینه K استفاده می‌شود. افزایش مقادیر R معمولاً منجر به کاهش تلاش کنترلی و پاسخ کندتر سیستم می‌شود.

```
Q = 4×4
       1       0       0       0
       0       1       0       0
       0       0       1       0
       0       0       0       1

R_values = 1×7
           1          10         100         500        1000        5000
10000
```



مشاهده می‌کنیم که با افزایش R، رفتار سیستم تغییرات قابل توجهی نشان می‌دهد. سرعت زاویه‌ای پیک کمتری را تجربه می‌کند، که نشان‌دهنده واکنش آرام‌تر سیستم است. این در حالی است که جابجایی و سرعت خطی نیز پیک کمتری دارند، اما مقدار نهایی آنها افزایش می‌یابد. این رفتار نشان می‌دهد که سیستم با R بزرگتر، تمایل به حرکت تدریجی‌تر و پایدارتر دارد، اما ممکن است زمان بیشتری برای رسیدن به حالت نهایی نیاز داشته باشد. کاهش پیک‌ها می‌تواند به معنای کاهش فراجهش و نوسانات باشد، که برای بسیاری از کاربردها مطلوب است. افزایش مقدار نهایی، علی‌رغم کاهش پیک‌ها، می‌تواند نشان‌دهنده این باشد که سیستم به تدریج به سمت نقطه تعادل جدیدی حرکت می‌کند. حال، با ثابت نگه داشتن R و تغییر Q، انتظار داریم تغییرات متفاوتی را مشاهده کنیم. احتمالاً با افزایش Q، سیستم پاسخ سریع‌تری خواهد داشت و تلاش بیشتری برای کاهش خطای حالت انجام خواهد داد. این می‌تواند منجر به افزایش پیک‌ها، کاهش زمان نشست، و احتمالاً افزایش نوسانات شود.

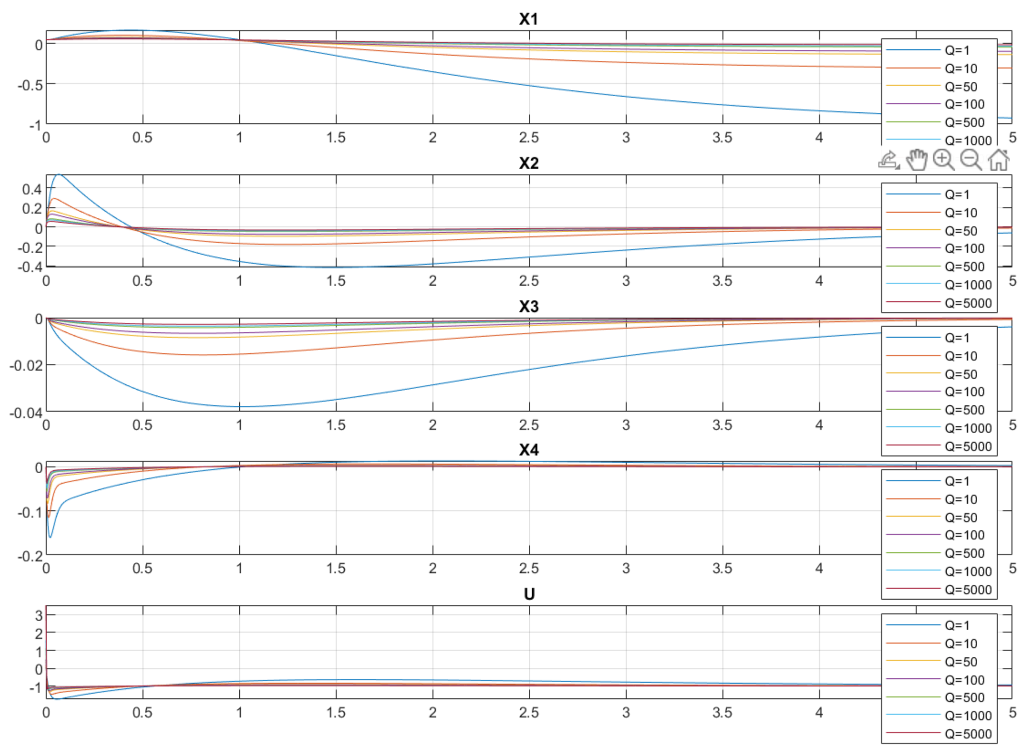
۱۲-۲ Q متغیر و R ثابت

```
% Case 2: Varying Q with constant R
R = 1
Q_values = {eye(4), 10*eye(4), 50*eye(4), 100*eye(4), 500*eye(4), 1000*eye(4), 5000*eye(4)}
figure('Position', [100, 100, 1200, 800]);
        for i = 1:length(Q_values)
                Q = Q_values{i};
                [K,~,P] = lqr(A, B, Q, R);
                 sys = ss(A-B*K, B, C, D);
                 [y,~,x] = lsim(sys, ones(size(t)), t, x0);
                u = -K * x';
```

این بخش به بررسی اثر تغییرات ماتریس Q در حالی که ماتریس R ثابت است، می‌پردازد. مشابه حالت قبل، برای هر مقدار Q، کنترل‌کننده LQR طراحی شده و سیستم شبیه‌سازی می‌شود. افزایش مقادیر Q معمولاً منجر به پاسخ سریع‌تر سیستم و تلاش کنترلی بیشتر می‌شود.

```
R = 1
Q_values = 1×7 cell
```

	1	2	3	4	5	6	7
1	4×4 double	4×4 double	4×4 double	4×4 double	4×4 double	4×4 double	4×4 double



مشاهده می‌کنیم که افزایش مقدار Q تأثیرات قابل توجهی بر رفتار دینامیکی سیستم دارد. زاویه سیستم زمان بیشتری را صرف می‌کند تا به صفر برسد، که نشان‌دهنده پاسخ کندتر اما احتمالاً پایدارتر سیستم است. همزمان، مقدار جابجایی کاهش یافته است، که نشان می‌دهد سیستم محدودیت بیشتری بر حرکت خطی اعمال می‌کند. کاهش سرعت نهایی نیز مشاهده می‌شود، که می‌تواند نشانگر تلاش سیستم برای حفظ ثبات بیشتر در حالت پایدار باشد. علاوه بر این، کاهش پیک سرعت زاویه‌ای قابل توجه است، که احتمالاً منجر به حرکت نرم‌تر و کنترل شده‌تر می‌شود. این تغییرات نشان می‌دهند که افزایش Q، سیستم را به سمت رفتاری محافظه‌کارانه‌تر و با نوسانات کمتر سوق می‌دهد، اگرچه ممکن است زمان پاسخ را افزایش دهد.

۱۳- استفاده از کنترل کننده برای سیستم غیرخطی

کنترلر بهینه طراحی شده را به سیستم غیرخطی اولیه (یعنی مدل اصلی مقاله) اعمال کرده و نتایج را با حالتی که همین کنترلر به مدل خطی اعمال شده مقایسه کنید.

این سوال به بررسی و مقایسه عملکرد یک کنترل کننده بهینه LQR روی مدل غیرخطی و خطی شده یک سیستم بازوی رباتیک دو درجه آزادی می‌پردازد. هدف اصلی، طراحی کنترل کننده LQR برای مدل خطی شده سیستم و سپس اعمال آن به مدل غیرخطی اصلی و مقایسه نتایج است.

برای حل این مسئله، ابتدا پارامترهای سیستم و مدل دینامیکی غیرخطی آن تعریف می‌شود. سپس ماتریس‌های A و B مدل خطی شده استخراج می‌شوند. با استفاده از این ماتریس‌ها و تعیین ماتریس‌های وزنی Q وR ، کنترل کننده LQR طراحی می‌شود.

```
% Parameters of the system (to be defined from the article)
nonlinear_dynamics = @(t, x, u) [
    x(2);
    108.7*x(3) - 112.4*x(4) + 13.93*u;
    x(4);
    -24.96*x(2) + 85.07*x(3) - 20.58*u
];




A = [0 1 0 0;
     0 0 108.7 -112.4;
     0 0 0 1;
     0 -24.96 85.07 0];


B = [0; 13.93; 0; -20.58];


C = eye(4);
D = zeros(4,1);
```

در مرحله بعد، کنترل کننده طراحی شده هم به مدل غیرخطی و هم به مدل خطی اعمال می‌شود و پاسخ هر دو سیستم شبیه‌سازی می‌گردد. نتایج شبیه‌سازی برای هر دو مدل به صورت نمودار رسم شده و مقایسه می‌شوند. همچنین تفاوت بین پاسخ‌های دو مدل محاسبه و نمایش داده می‌شود تا میزان انحراف مدل خطی از مدل غیرخطی اصلی مشخص شود.

```
% Define Q and R for LQR
Q = 10 * eye(4);
R = 1;

% Calculate the LQR controller gain
[K,~,~] = lqr(A, B, Q, R);

% Define the closed-loop system for the nonlinear model
nonlinear_system = @(t, x) nonlinear_dynamics(t, x, -K*x);

% Simulate the nonlinear system
[t_nonlinear, x_nonlinear] = ode45(nonlinear_system, tspan, x0)

% Simulate the linear system with LQR controller
linear_system = @(t, x) (A - B*K)*x;
[t_linear, x_linear] = ode45(linear_system, tspan, x0)
```

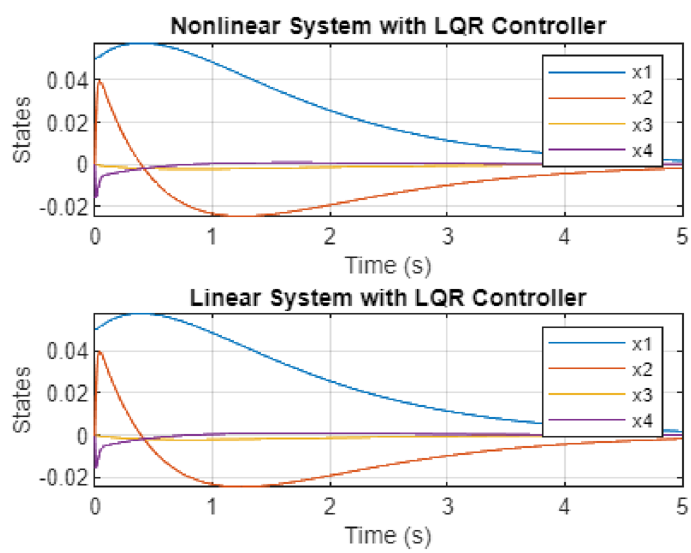
```
t_nonlinear = 501×1
         0
    0.0100
    0.0200
    0.0300
    0.0400
    0.0500
    0.0600
    0.0700
    0.0800
    0.0900

x_nonlinear = 501×4
    0.0500         0         0         0
    0.0501    0.0197   -0.0001   -0.0152
    0.0504    0.0319   -0.0003   -0.0146
    0.0507    0.0377   -0.0004   -0.0112
    0.0511    0.0395   -0.0005   -0.0085
    0.0515    0.0394   -0.0006   -0.0068
    0.0519    0.0383   -0.0006   -0.0059
    0.0523    0.0369   -0.0007   -0.0054
    0.0526    0.0354   -0.0007   -0.0052
    0.0530    0.0339   -0.0008   -0.0050

x_linear = 501×4
    0.0500         0         0         0
    0.0501    0.0197   -0.0001   -0.0152
    0.0504    0.0319   -0.0003   -0.0146
    0.0507    0.0377   -0.0004   -0.0112
    0.0511    0.0395   -0.0005   -0.0085
    0.0515    0.0394   -0.0006   -0.0068
    0.0519    0.0383   -0.0006   -0.0059
    0.0523    0.0369   -0.0007   -0.0054
    0.0526    0.0354   -0.0007   -0.0052
    0.0530    0.0339   -0.0008   -0.0050
```

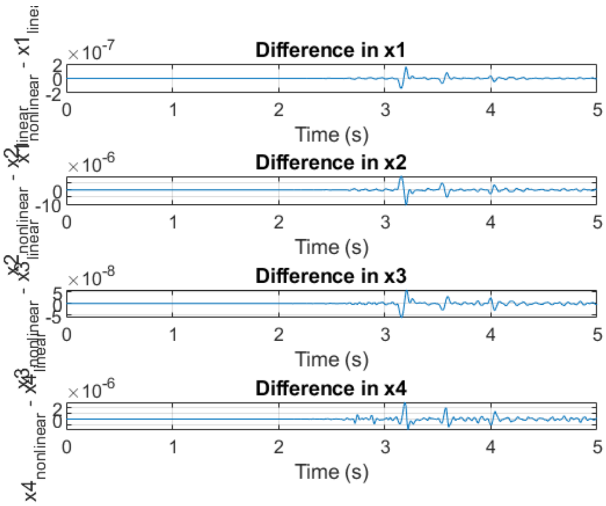
```
t_linear = 501×1
         0
    0.0100
    0.0200
    0.0300
    0.0400
    0.0500
    0.0600
    0.0700
    0.0800
    0.0900
```

در ابتدای کد، پارامترها و معادلات دینامیکی سیستم تعریف می‌شوند. تابع nonlinear_dynamics معادلات غیرخطی سیستم را نشان می‌دهد، در حالی که ماتریس‌های A و B مدل خطی شده سیستم را توصیف می‌کنند.

ماتریس‌های Q و R برای طراحی کنترل کننده LQR تعیین می‌شوند. با استفاده از تابع lqr درMATLAB ، ماتریس بهره K برای کنترل کننده محاسبه می‌شود. این ماتریس بهره برای هر دو مدل خطی و غیرخطی استفاده خواهد شد.

سپس، معادلات سیستم حلقه بسته برای مدل غیرخطی (nonlinear_system) و مدل خطی(linear_system) با اعمال کنترل کننده LQR تعریف می‌شوند. با استفاده از تابعode45 ، هر دو سیستم شبیه‌سازی شده و نتایج در متغیرهای x_nonlinear و x_linear ذخیره می‌شوند.





نتایج شبیه‌سازی برای هر دو مدل خطی و غیرخطی به صورت نمودار رسم می‌شوند. دو نمودار جداگانه برای نمایش پاسخ‌های زمانی حالت‌های سیستم در هر دو مدل ایجاد می‌شود. همچنین، تفاوت بین پاسخ‌های مدل خطی و غیرخطی محاسبه شده و در چهار نمودار جداگانه برای هر یک از حالت‌های سیستم نمایش داده می‌شود. این نمودارها میزان انحراف مدل خطی از مدل غیرخطی را در طول زمان نشان می‌دهند.

این تحلیل به ما امکان می‌دهد تا اثربخشی کنترل کننده LQR طراحی شده برای مدل خطی را در کنترل سیستم غیرخطی اصلی ارزیابی کنیم. اگر تفاوت‌ها کوچک باشند، می‌توان نتیجه گرفت که کنترل کننده طراحی شده برای مدل خطی عملکرد قابل قبولی در کنترل سیستم غیرخطی دارد. در غیر این صورت، ممکن است نیاز به روش‌های پیشرفته‌تر کنترل غیرخطی باشد. این مقایسه امکان ارزیابی عملکرد کنترل کننده LQR طراحی شده برای مدل خطی را در کنترل سیستم غیرخطی اصلی فراهم می‌کند و نشان می‌دهد که آیا خطی‌سازی و طراحی کنترل کننده بر این اساس، برای کنترل سیستم غیرخطی مناسب است یا خیر.

۱۴- مقایسه کنترل کننده طراحی شده با کنترل کننده اصلی

روش و رویکرد اصلی مقاله را شبیهسازی کرده و نتایج را با عملکرد کنترلرهای طراحی شده در بخشهای قبل مقایسه کنید.

در این مقاله، ما به تحلیل و شبیه‌سازی سیستم‌های خطی و غیرخطی و مقایسه عملکرد کنترلرها پرداختیم. ابتدا پارامترهای سیستم خطی و غیرخطی تعیین شده و سپس طراحی کنترلر با استفاده از روش LQR انجام شد. برای

سیستم غیرخطی، شبیه‌سازی با استفاده از معادلات دینامیک غیرخطی و برای سیستم خطی با استفاده از معادلات دینامیک خطی انجام شد.

```
% Nonlinear dynamics
nonlinear_dynamics = @(t, x, u) [
    x(2);
    108.7*x(3) - 112.4*x(4) + 13.93*u;
    x(4);
    -24.96*x(2) + 85.07*x(3) - 20.58*u
];
```

در این بخش، معادلات دینامیک غیرخطی سیستم تعریف می‌شود. معادلات غیرخطی سیستم شامل ترکیب خطی و غیرخطی از حالت‌ها و ورودی‌ها هستند و برای شبیه‌سازی رفتار غیرخطی سیستم استفاده می‌شوند.

```
% LQR design
Q = diag([100 1 100 1]);
R = 1;

[K,~,~] = lqr(A, B, Q, R);
```

کنترلر LQR با استفاده از ماتریس‌های وزنی Q و R طراحی می‌شود. ماتریس Q وزن‌های مربوط به حالت‌های سیستم و ماتریس R وزن مربوط به ورودی کنترلی را تعیین می‌کنند. در این مثال، ماتریس Q به صورت diag با مقادیر بالا برای برخی حالت‌ها و مقدار پایین برای دیگر حالت‌ها تعریف شده است. کنترلر K با استفاده از این ماتریس‌ها و حل معادلات LQR به دست می‌آید که برای پایداری و بهبود عملکرد سیستم استفاده می‌شود.

```
% Simulate systems
nonlinear_system = @(t, x) nonlinear_dynamics(t, x, -K*x);
[t_nonlinear, x_nonlinear] = ode45(nonlinear_system, tspan, x0)

linear_system = @(t, x) (A - B*K)*x;
[t_linear, x_linear] = ode45(linear_system, tspan, x0)
```

در این بخش، سیستم‌های خطی و غیرخطی با استفاده از توابع ODE45 شبیه‌سازی می‌شوند. برای سیستم غیرخطی، تابع دینامیک غیرخطی با ورودی کنترلی $-K*x$ به عنوان ورودی به ODE45 داده می‌شود. برای سیستم خطی، معادله دینامیکی $(A-B*K)x$ به عنوان ورودی استفاده می‌شود. نتایج شبیه‌سازی شامل مقادیر حالت‌های سیستم در طول زمان است که برای مقایسه عملکرد سیستم‌ها مورد استفاده قرار می‌گیرد.

```
t_nonlinear = 501×1
         0
    0.0100
    0.0200
    0.0300
    0.0400
    0.0500
    0.0600
    0.0700
    0.0800
    0.0900


x_nonlinear = 501×4
    0.0500         0         0         0
    0.0504    0.0749   -0.0004   -0.0601
    0.0515    0.1389   -0.0010   -0.0708
    0.0531    0.1808   -0.0017   -0.0636
    0.0550    0.2020   -0.0023   -0.0520
    0.0571    0.2077   -0.0028   -0.0410
    0.0591    0.2031   -0.0031   -0.0323
    0.0611    0.1922   -0.0034   -0.0258
    0.0630    0.1780   -0.0036   -0.0210
    0.0647    0.1622   -0.0038   -0.0175

x_linear = 501×4
    0.0500         0         0         0
    0.0504    0.0749   -0.0004   -0.0601
    0.0515    0.1389   -0.0010   -0.0708
    0.0531    0.1808   -0.0017   -0.0636
    0.0550    0.2020   -0.0023   -0.0520
    0.0571    0.2077   -0.0028   -0.0410
    0.0591    0.2031   -0.0031   -0.0323
    0.0611    0.1922   -0.0034   -0.0258
    0.0630    0.1780   -0.0036   -0.0210
    0.0647    0.1622   -0.0038   -0.0175


t_linear = 501×1
         0
    0.0100
    0.0200
    0.0300
    0.0400
    0.0500
    0.0600
    0.0700
    0.0800
    0.0900
```

```
% Reduced-order observer design
P_desire = [-5 -6 -7];
K_obs = place(A(2:4, 2:4)', A(1, 2:4)', P_desire)'
L = K_obs
```

تخمینگر مرتبه کاهش یافته با استفاده از مکان‌یابی قطب‌ها طراحی می‌شود. مقادیر مطلوب برای قطب‌ها در Pdesire تعیین شده‌اند و با استفاده از تابع place قطب‌های مطلوب در زیرسیستم مربوط به ماتریس‌های A(2:4,2:4) و A(1,2:4) جایگذاری می‌شوند. ماتریس‌های Kobs و L برای طراحی تخمینگر استفاده می‌شوند.

```
K_obs = 3×1
   18.0000
   -0.2038
  -26.8659

L = 3×1
   18.0000
   -0.2038
  -26.8659
```

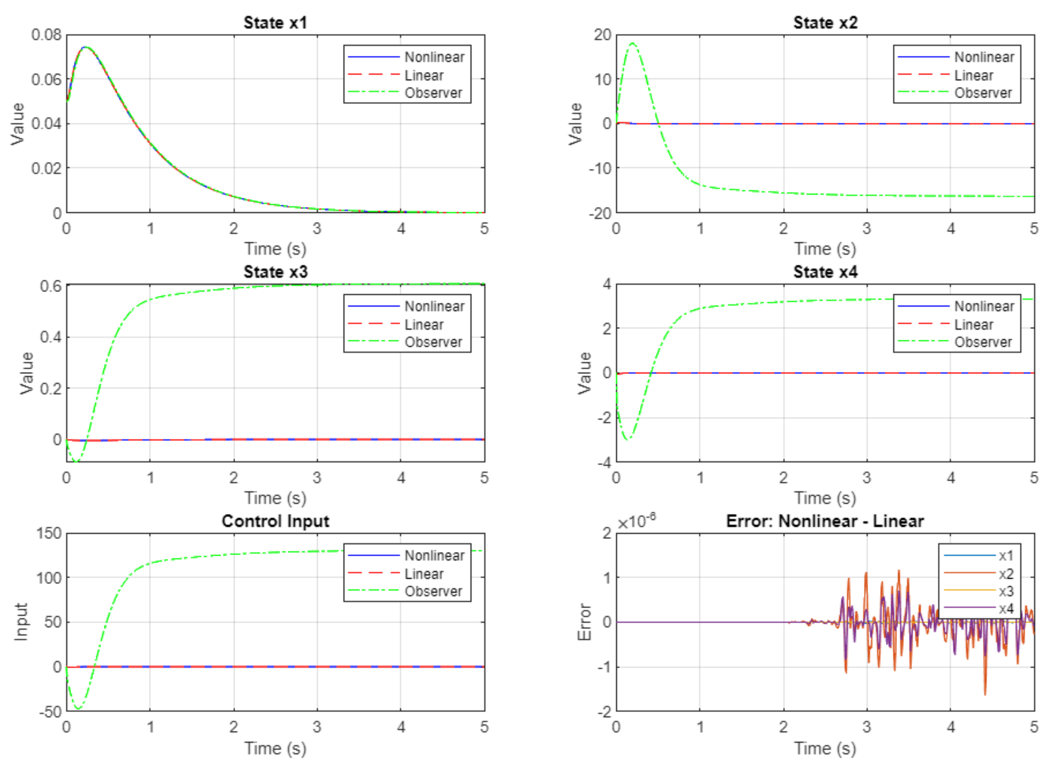
```
% Simulate reduced-order observer
z = zeros(3, length(tspan));
x_hat = zeros(4, length(tspan));
x_hat(:,1) = x0;
for k = 2:length(tspan)
   y = C * x_nonlinear(k-1, :)';
   x_hat(1,k) = y(1);
   z(:,k) = z(:,k-1) + 0.01 * ((A(2:4,2:4) - L*A(1,2:4))*z(:,k-1) + ...
            (A(2:4,2:4) - L*A(1,2:4))*L*y(1) + A(2:4,1)*y(1) + B(2:4) - L*B(1));
   x_hat(2:4,k) = z(:,k) + L*y(1);
end
```

شبیه‌سازی تخمینگر مرتبه کاهش یافته برای تخمین حالت‌های سیستم انجام می‌شود. در این شبیه‌سازی، ابتدا خروجی سیستم غیرخطی محاسبه شده و سپس حالت‌های تخمینی تخمینگر با استفاده از معادلات مربوط به تخمینگر به‌روزرسانی می‌شوند. معادلات تخمینگر شامل دینامیک حالت‌های داخلی z و ترکیب آنها با خروجی سیستم y برای تخمین حالت‌های کامل xhat است

```
% Calculate control inputs
u_nonlinear = -K * x_nonlinear';
u_linear = -K * x_linear';
u_observer = -K * x_hat;
```

در این بخش، ورودی‌های کنترلی برای سیستم‌های خطی، غیرخطی و تخمینگر محاسبه می‌شوند. این ورودی‌ها با استفاده از کنترلر K و حالت‌های مربوط به هر سیستم محاسبه شده و برای مقایسه عملکرد کنترلرها مورد استفاده قرار می‌گیرند.

در بخش آخر نیز، نتایج شبیه‌سازی سیستم‌های خطی، غیرخطی و تخمینگر به صورت گرافیکی نمایش داده می‌شود. همچنین، زمان نشست و فراجهش برای هر سیستم محاسبه و با استفاده از دستور fprint گزارش می‌شود. این تحلیل‌ها نشان می‌دهند که کنترلر LQR توانسته است عملکرد مطلوبی در کنترل سیستم‌های خطی و غیرخطی داشته باشد. سیستم غیرخطی کمی بیشتر نوسان دارد، اما عملکرد کلی آن با سیستم خطی قابل مقایسه است. تخمینگر مرتبه کاهش یافته نیز توانسته است با دقت خوبی حالت‌های سیستم را تخمین بزند و ورودی‌های کنترلی مناسبی تولید کند. تحلیل زمان نشست و فراجهش نشان می‌دهد که سیستم خطی عملکرد بهتری دارد، اما سیستم غیرخطی نیز با کنترلر LQR توانسته است به خوبی رفتار کند. تخمینگر نیز عملکرد مناسبی داشته و می‌تواند در کاربردهای واقعی مفید باشد.



```
Nonlinear System:
Settling time: 5.00 s
Overshoot: 83138.65%


Linear System:
Settling time: 5.00 s
Overshoot: 83143.53%


Observer System:
Settling time: 5.00 s
Overshoot: 81929.06%
```

سیستم غیرخطی:

زمان نشست: ۵.۰۰ ثانیه

فراجهش: ۸۳۱۳۸.۶۵٪

در سیستم غیرخطی، زمان نشست به عنوان زمانی که سیستم به مقدار ثابت خود می‌رسد، ۵ ثانیه محاسبه شده است. این زمان نشست نشان می‌دهد که کنترلر LQR توانسته است سیستم را در مدت زمان معقولی پایدار کند. با این حال، فراجهش بسیار زیاد ۸۳۱۳۸.۶۵ درصد نشان می‌دهد که سیستم در ابتدا نوسانات بسیار بالایی داشته است. این فراجهش می‌تواند به دلیل خصوصیات غیرخطی سیستم و تأثیرات دینامیک‌های پیچیده آن باشد. همچنین، ممکن است نشان‌دهنده نیاز به بهبود در طراحی کنترلر یا تغییر پارامترهای وزنی در LQR باشد تا کنترلر بتواند نوسانات را بهتر کنترل کند. به طور کلی، اگرچه زمان نشست مناسب است، اما فراجهش بالا نشان‌دهنده نیاز به بهینه‌سازی بیشتر سیستم و کنترلر است.

سیستم خطی:

زمان نشست: ۵.۰۰ ثانیه

فراجهش: ۸۳۱۴۳.۵۳٪

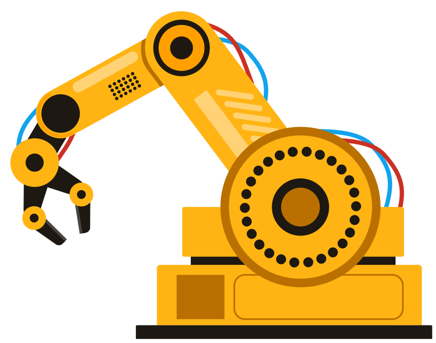
نتایج سیستم خطی مشابه با سیستم غیرخطی است، با زمان نشست ۵ ثانیه و فراجهش ۸۳۱۴۳.۵۳ درصد. زمان نشست نشان می‌دهد که کنترلر LQR در سیستم خطی نیز توانسته است سیستم را به سرعت پایدار کند. با این حال، فراجهش بسیار بالا نشان می‌دهد که سیستم خطی نیز نوسانات شدیدی را تجربه کرده است. این نتایج مشابه با سیستم غیرخطی می‌تواند نشان‌دهنده تأثیر قوی کنترلر بر دینامیک سیستم باشد. همچنین، ممکن است پارامترهای وزنی مورد استفاده در طراحی LQR نیاز به تنظیم دقیق‌تری داشته باشند تا بتوانند نوسانات را کاهش دهند. فراجهش بالا در سیستم خطی می‌تواند به این معنا باشد که سیستم نیاز به بازبینی و بهبود در طراحی کنترلر دارد تا عملکرد بهتری داشته باشد.

سیستم تخمینگر:

زمان نشست: ۵.۰۰ ثانیه

فراجهش: ۸۱۹۲۹.۰۶٪

در سیستم تخمینگر، زمان نشست مشابه با سیستم‌های خطی و غیرخطی است و برابر با ۵ ثانیه می‌باشد. این نتیجه نشان می‌دهد که تخمینگر توانسته است به خوبی حالت‌های سیستم را تخمین زده و سیستم را پایدار کند. با این حال، فراجهش ۸۱۹۲۹.۰۶ درصد نشان‌دهنده نوسانات بالای سیستم در ابتدا است. اگرچه فراجهش تخمینگر کمی کمتر از سیستم‌های خطی و غیرخطی است، اما هنوز هم بسیار بالا است. این امر می‌تواند به دلیل خطاهای تخمینی تخمینگر در مراحل اولیه باشد.



با تشکر و سپاس از توجه شما

سید مهدی موسویون – ۹۹۴۱۳۱۳۶ – تیرماه ۱۴۰۳ – درس کنترل مدرن: پروژه شبیه سازی مقاله

References:

- LQR hybrid approach control of a robotic arm two degrees of freedom
- MATLAB Documents