بسم الله الرحمن الرحیم

طراحی کنترل کننده برای سرو موتور با استفاده از روش زیگلر – نیکولز و بهینه سازی ضرایب در MATLAB

تهیه کننده :

داود شقاقی

Davood.shaghaghi@gmail.com

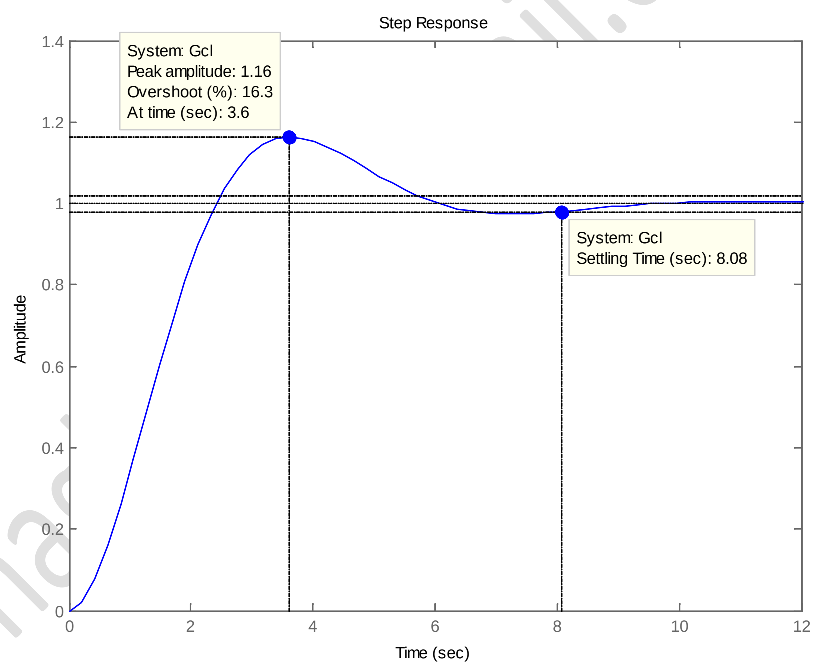
بهار ۱۳۸۸

در این مقاله، سعی بر آن است که یک کنترل کننده کلاسیک برای تابع تبدیل سرو موتور طراحی شود. ابتدا این کار با استفاده از روش زیگلر – نیکولز انجام می شود، مشاهده می شود که پاسخ مطلوبی بدست نمی آید، بنابراین با انتقال سیستم و تابع تبدیل به محیط sisotool سعی داریم که پاسخ را مطلوب و ضرایب را بهینه سازیم .یک تابع تبدیل معمول برای servo motor ها به صورت زیر است :

$$G(s)=\frac{1}{s(s+1)}$$

پاسخ سیستم حلقه بسته را بدست می آوریم:

```
>> s=tf('s');
>> G=1/s/(s+1)

Transfer function:
   1
-------
s^2 + s

>> Gcl=feedback(G,1)

Transfer function:
     1
-----------
s^2 + s + 1

>> step(Gcl)
```



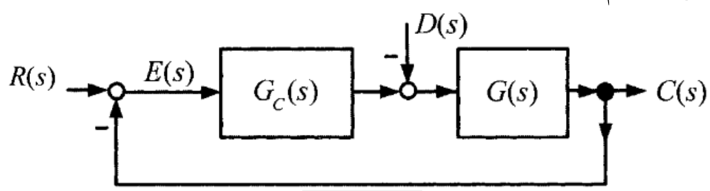
شکل 1

ملاحظه می شود که زمان نشست سیستم با توجه به مطلوبات مسئله ، بالاست.

حال می خواهیم یک کنترلر PID (یا PI) طراحی کنیم که شرایط زیر را ارضا کند:

- سیستم تحت هر شرایط پایدار باشد.
- زمان استقرار (نشست) سیستم حداکثر ۱ ثانیه باشد.
- حداکثر فراجهش سیستم ۱۶/۵٪ باشد.
- خطای حالت ماندگار به ورودی شیب کمتر از ۰/۲ باشد.

سیستم کنترل حلقه بسته را به صورت زیر در نظر می گیریم:



شکل 2

با فرض D(s)=0، برای کنترلر PID داریم :

$$C(s)=k_p(1+\frac{1}{T_i s}+T_d s)$$
$$=k_p(\frac{T_iT_d s^2+T_i s+1}{T_i s})$$
$$G(s)=\frac{1}{s(s+1)}$$
$$\Rightarrow G(s)C(s)=\frac{k_p(T_iT_d s^2+T_i s+1)}{T_i s^2(s+1)}$$
$$Kv=\frac{1}{\lim\limits_{s\to 0} sG(s)C(s)}=\frac{1}{\infty}=0$$

به دلیل وجود انتگرال گیر در plant ، K_v دارای شرایط مطلوبیست .

برای اینکه سیستم تحت هر شرایطی پایدار باشد ، با تمامی جملات ستون اول جدول راث ، مثبت باشند:

$$G(s)C(s)=\frac{k_p(T_iT_d s^2+T_i s+1)}{T_i s^2(s+1)}$$

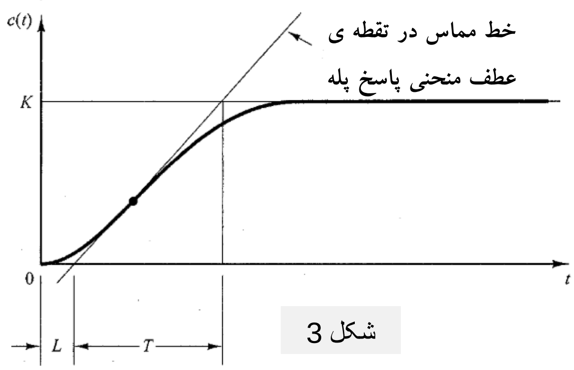
$$Gcl=\frac{G(s)C(s)}{1+G(s)C(s)}=\frac{\dfrac{k_p(T_iT_d s^2+T_i s+1)}{T_i s^2(s+1)}}{1+\dfrac{k_p(T_iT_d s^2+T_i s+1)}{T_i s^2(s+1)}}=\frac{k_p(T_iT_d s^2+T_i s+1)}{T_i s^3+(T_i+k_pT_iT_d)s^2+(k_pT_i s)s+k_p}$$

$$\begin{array}{llll}
s^3 & T_i & k_pT_i & \\
s^2 & T_i+k_pT_iT_d & k_p & \Rightarrow T_i+k_pT_iT_d>0 \Rightarrow 1+k_pT_d>0 \quad \text{همواره صحیح} \\
s & \dfrac{(T_i+k_pT_iT_d)(k_pT_i)-k_pT_i}{T_i+k_pT_iT_d} & & \Rightarrow T_i+k_pT_iT_d-1>0 \Rightarrow T_i(1+k_pT_d)>1 \\
1 & k_p & & \boxed{T_i>\dfrac{1}{1+k_pT_d}}
\end{array}$$

ابتدا می خواهیم پارامترهای PID را بصورت دستی محاسبه کنیم. در واقع این حل دستی ،یک حدس یا تخمین اولیه از حدود مقدار پارامترها است و سپس با استفاده از Matlab ، یک کنترلر با ضرایب بهینه طراحی خواهیم کرد.

چون plant درجه دو است، روش اول زیگلر–نیکولس برای آن مناسب است.

با توجه به این روش،ابتدا باید مقادیر L و T را از پاسخ پله (مطابق شکل۳) تعیین کنیم و سپس از جدول۱ پارامترهای کنترلر را بدست آوریم:



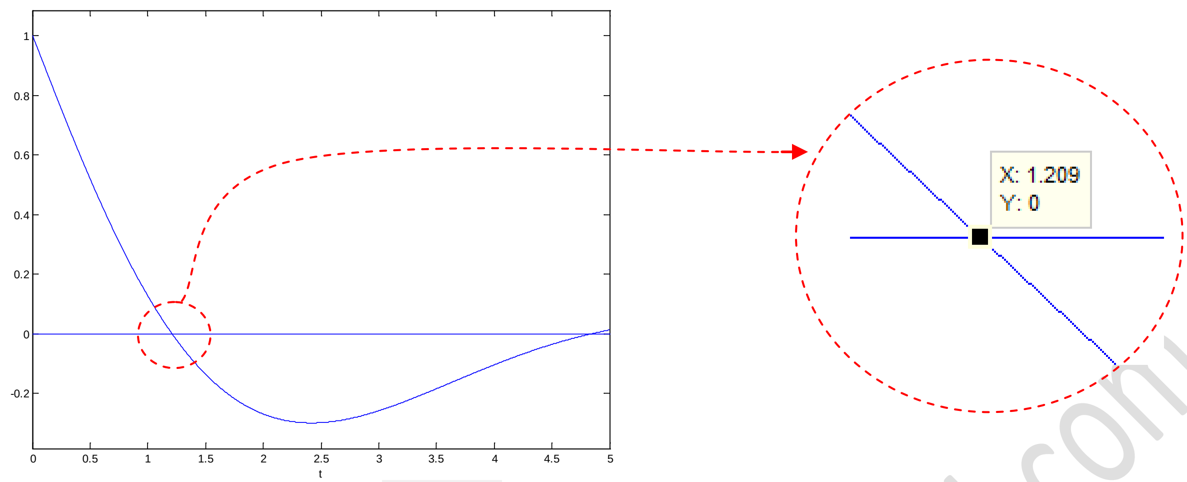
شکل 3

نوع کنترلر	K_p	T_i	T_d
P	$\frac{T}{L}$	∞	0
PI	$0.9\frac{T}{L}$	$\frac{L}{0.3}$	0
PID	$1.2\frac{T}{L}$	$2L$	$0.5L$

جدول 1

برای اینکار باید مراحل بدست آوردن پاسخ پله را به صورت symbolic انجام دهیم تا در نهایت معادله ی منحنی را داشته باشیم:

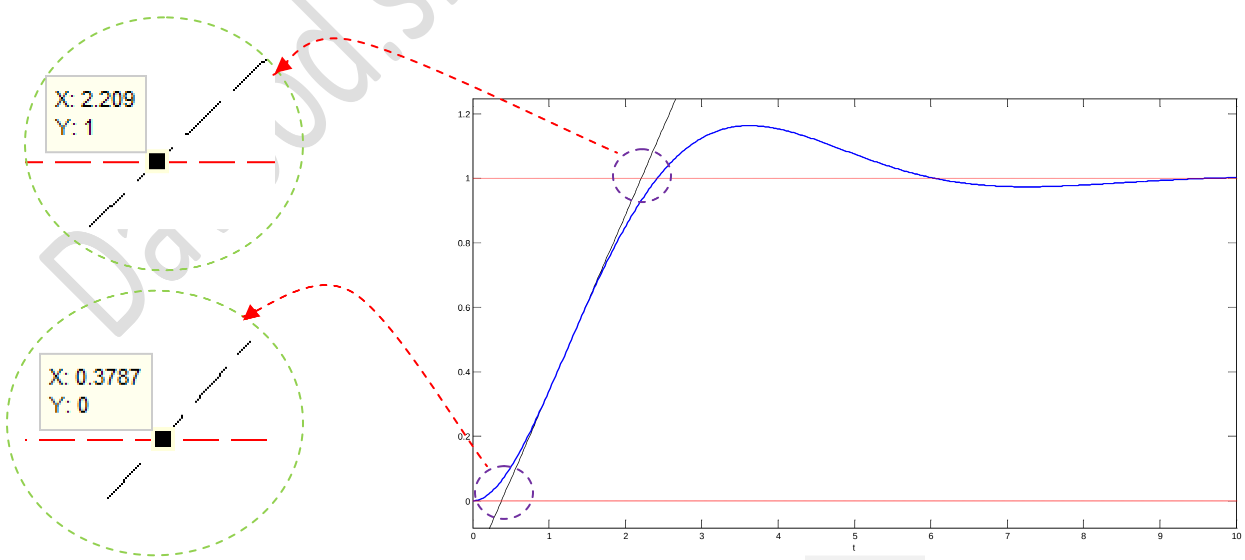
```
>> syms s
>> G=1/s/(s+1);      %%plant transfer function
>> Gcl=G/(G+1);      %%close loop transfer function
>> C=Gcl/s;          %%step response in Laplace domain
>> Ct=ilaplace(C);   %%step response in time domain
>> pretty(simplify(Ct))

                 /                          /   1/2    \ \
                 |                   1/2    | 3     t  | |
                 |    /   1/2    \   3    sin| ------  | |
         /   t \ |    | 3     t  |           \     2   / |
  1 - exp| - - | | cos| ------  | + ------------------ |
         \   2 / \    \     2    /            3          /
>>
>> d2=diff(Ct,2);     %%2th difference of C(t)
>> ezplot(d2,[0 10]) %%plot 2th difference of C(t)
>> line([0 10],[0 0])
```



شکل 4 مشتق دوم منحنی پاسخ پله

تا اینجا نقطه ی عطف منحنی بدست آمد . حال باید مقدار شیب منحنی و مقدار منحنی را در این نقطه بدست بیاوریم و سپس معادله ی خط مماس را بدست آوریم:

```
>> d1=diff(Ct);  %%first difference of C(t)
>> slope=subs(d1,1.209);  %%slope of curve in t=sol
>> y0=subs(Ct,1.209);  %% value of curve in t=sol
>> syms t
>> y=slope*(t-1.209)+y0;  %%Tangent line at inflection point
>> ezplot(y,[0 10])       %%plot of tangent line
>> hold on;
>> ezplot(Ct,[0 10])      %%plot of step response
>> line([0 10],[0 0])
```



شکل 5

مقادیر L و T به صورت زیر محاسبه می شوند :

$$L = .3787\ s$$
$$T = 2.209 - .3787 = 1.8303$$

با توجه به جدول پارامترهای کنترلر PI به صورت زیر بدست می آید :

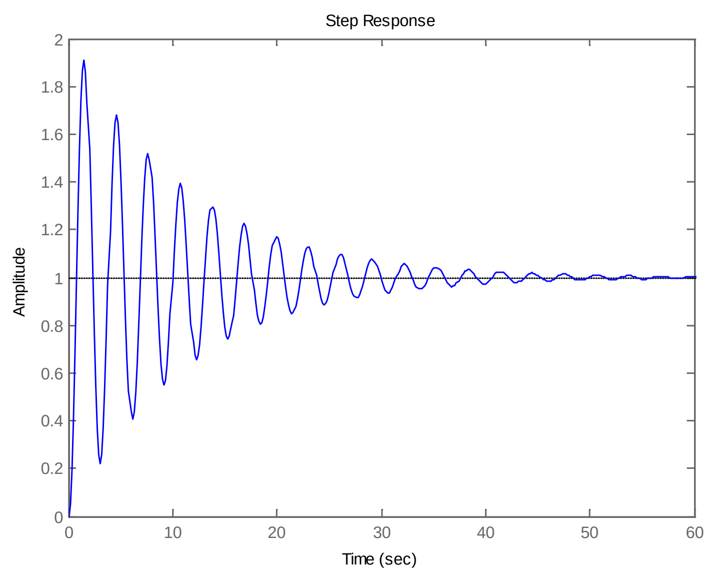
$$k_p = .9\frac{T}{L} = .9\frac{1.8303}{.3787} = 4.3498$$
$$T_i = \frac{L}{.3} = \frac{.3787}{.3} = 1.2623$$
$$T_d = 0$$

و کنترلر PI به صورت زیر خواهد بود :

$$G_c(s) = 4.3498(1 + \frac{1}{1.2623\ s})$$

پاسخ پله ی سیستم با کنترلر را بدست می آوریم :

```
>> clear s
>> s=tf('s');
>> Gc=4.3498*(1+1/1.2623/s);
>> G=1/s/(s+1);
>> Gcl=feedback(Gc*G,1)

Transfer function:
      4.35 s + 3.446
-------------------------
s^3 + s^2 + 4.35 s + 3.446

>> step(Gcl)
```



شکل 6

اکنون باید در محیط sisotool ، قطب های کنترلر را آنقدر جابجا کنیم تا به حول و حوش پاسخ مطلوب برسیم . اگر با این کار نتوانیم به جواب مطلوب برسیم ،این کنترلر برای رسیدن به شرایط مسئله مطلوب نیست و باید به سراغ PID برویم.

```
>> sisotool(G,Gc)
```

با جابجایی قطب کنترلر و تغییر گین آن ، به این نتیجه می رسیم که این کنترلر مناسب نیست.

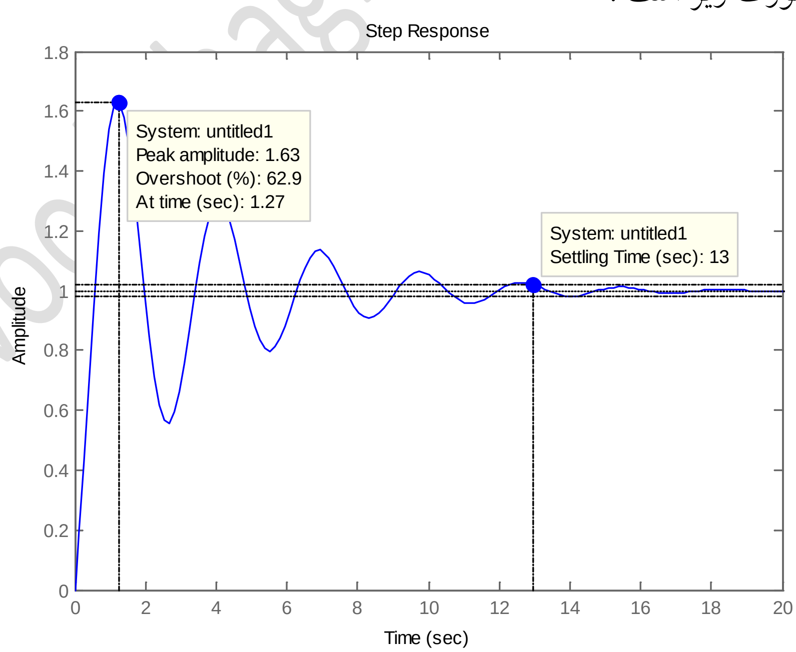
با توجه به جدول ، پارامترهای کنترلر PID به صورت زیر محاسبه می شود:

$$k_p = 1.2\frac{T}{L} = 1.2\frac{1.8303}{.3787} = 5.7997$$
$$T_i = 2L = 2(.3787) = .7574$$
$$T_d = .5L = .1893$$

و PID به فرم زیر خواهد بود :

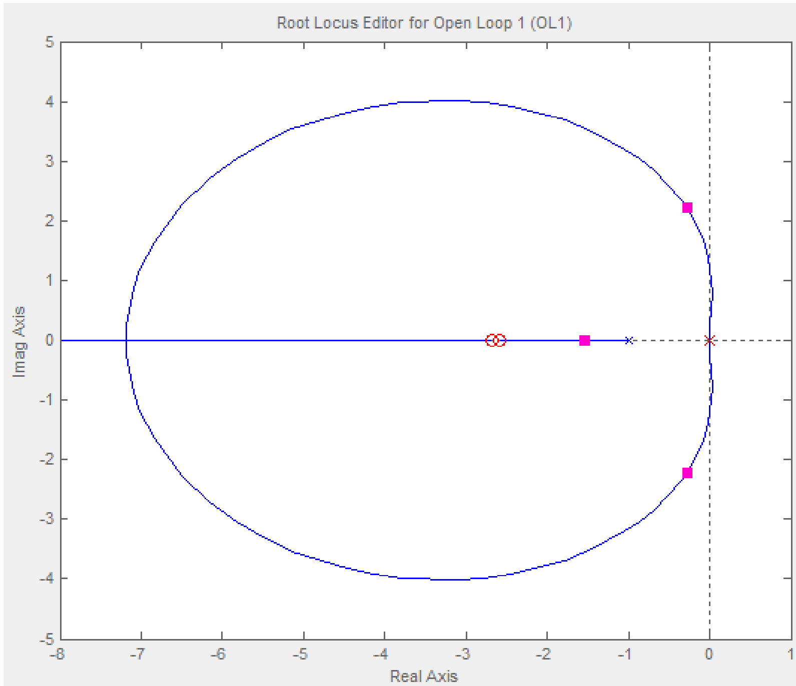
$$G_c(s) = 5.7997(1 + \frac{1}{.7574\,s} + .1893\,s)$$

پاسخ پله ی اولیه به صورت زیر است :



شکل 7

مکان ریشه های plant و کنترلر اولیه :

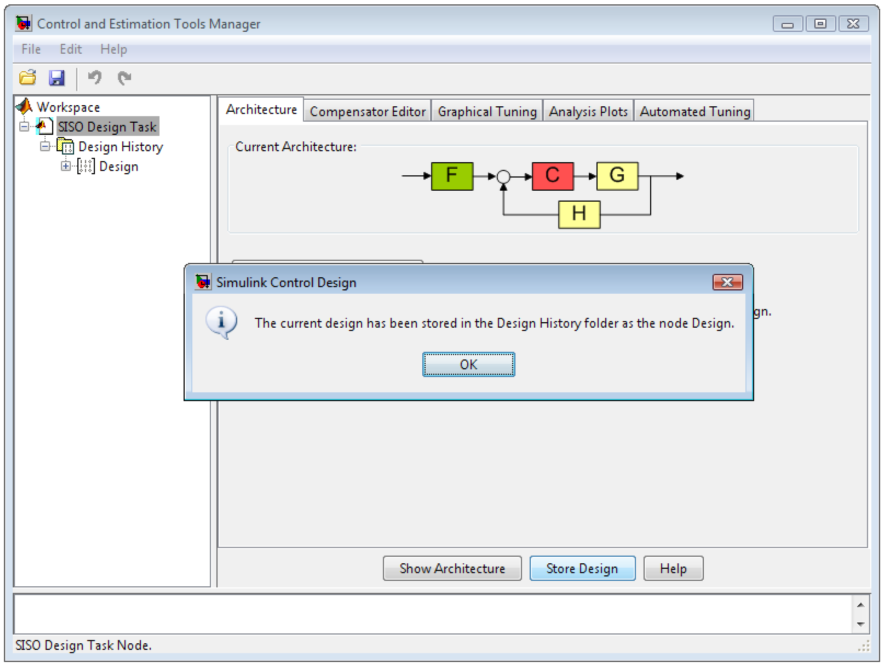


شکل 8

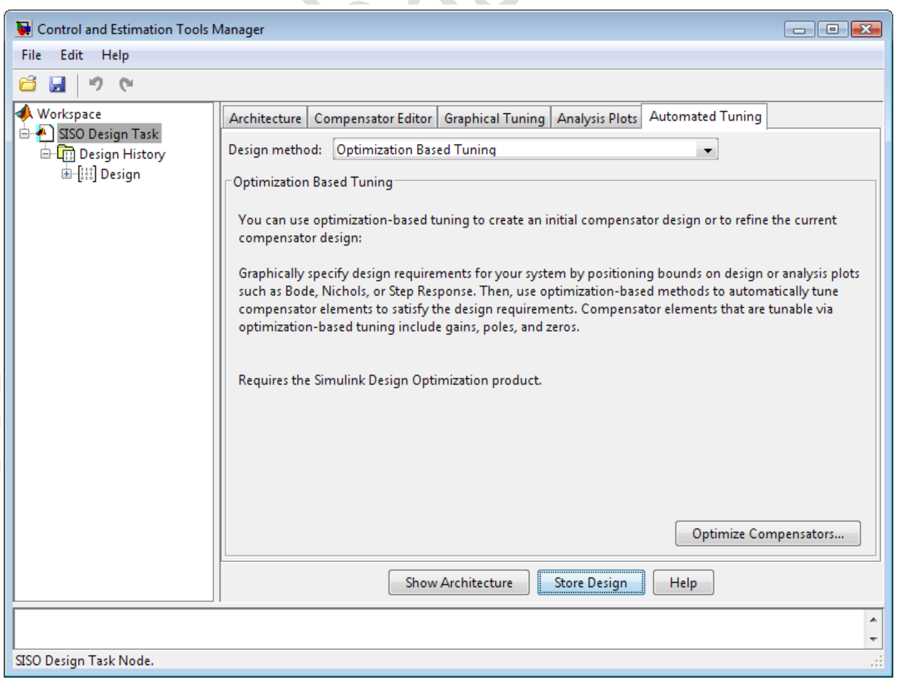
اکنون با استفاده از sisotool ، کنترلر طراحی شده را بهینه می کنیم.این پروسه شامل تغییر پارامترهای PID تا رسیدن به شرایط مطلوب می باشد.

```
>> Gc=5.7997*(1+1/.7574/s+.1893*s);
>> sisotool(G,Gc)
```

ابتدا مطابق شکل (۸) در پایین پنجره ی Control and Estimation Tools Manager ، با انتخاب Store Design ، کنترلر اولیه را ذخیره می کنیم.از تب Automated tuning ،نوع طراحی (Design Method)را Optimization Based Tuning انتخاب می کنیم.سپس ...Optimize Compensators را کلیک میکنیم،شکل (۹).

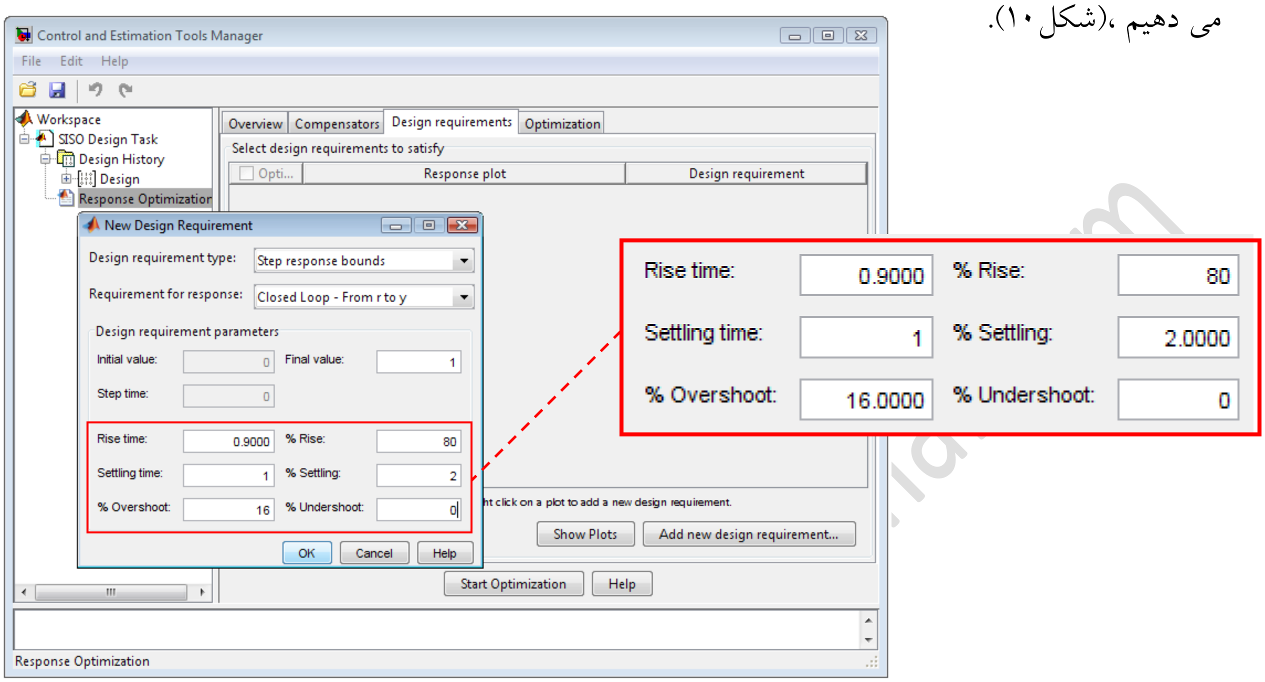

Control and Estimation Tools Manager
File Edit Help
Workspace
SISO Design Task
Design History
Design
Architecture
Compensator Editor
Graphical Tuning
Analysis Plots
Automated Tuning
Current Architecture:
F
C
G
H
Simulink Control Design
The current design has been stored in the Design History folder as the node Design.
OK
Show Architecture
Store Design
Help
SISO Design Task Node.

شکل 8


Control and Estimation Tools Manager
File Edit Help
Workspace
SISO Design Task
Design History
Design
Architecture
Compensator Editor
Graphical Tuning
Analysis Plots
Automated Tuning
Design method: Optimization Based Tuning
Optimization Based Tuning
You can use optimization-based tuning to create an initial compensator design or to refine the current compensator design:
Graphically specify design requirements for your system by positioning bounds on design or analysis plots such as Bode, Nichols, or Step Response. Then, use optimization-based methods to automatically tune compensator elements to satisfy the design requirements. Compensator elements that are tunable via optimization-based tuning include gains, poles, and zeros.
Requires the Simulink Design Optimization product.
Optimize Compensators...
Show Architecture
Store Design
Help
SISO Design Task Node.

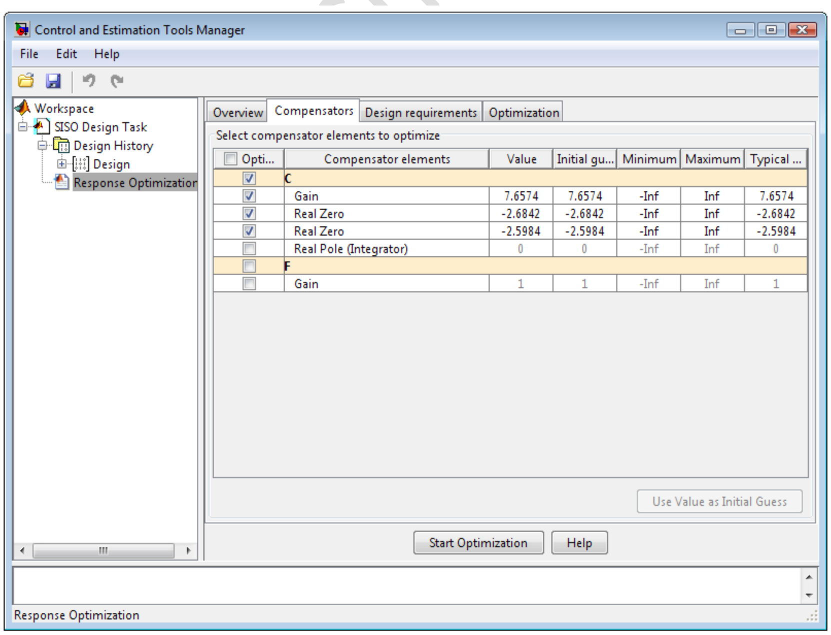
شکل 9

در این قسمت باید شرایط مطلوب مسئله را وارد کنیم. این کار را از تب Design requirements انجام می دهیم ،(شکل ۱۰).



شکل10

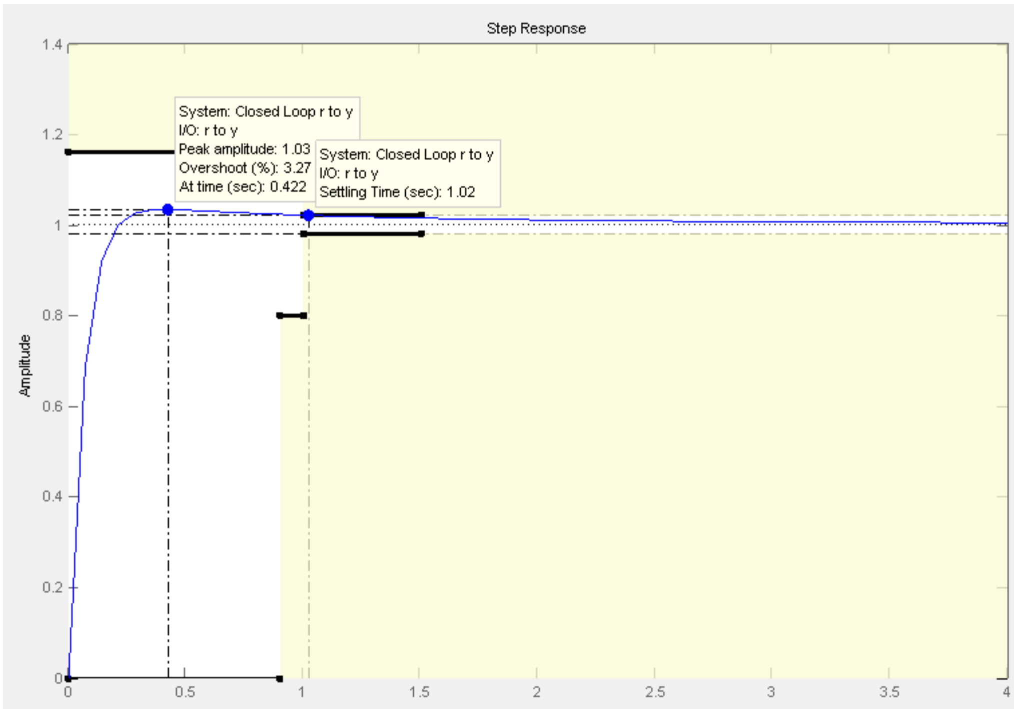
در مرحله ی بعد ، باید پارامترهای tuning را انتخاب کنیم.در این قسمت متغیرهایی که Matlab اجازه دارد در آنها تغییر به وجود آورد تا به جواب مطلوب برسد را انتخاب می کنیم . این کار را از طریق تب Compensators انجام می دهیم.با انتخاب check mark برای C پارامترهای کنترلر انتخاب می شود.



شکل11

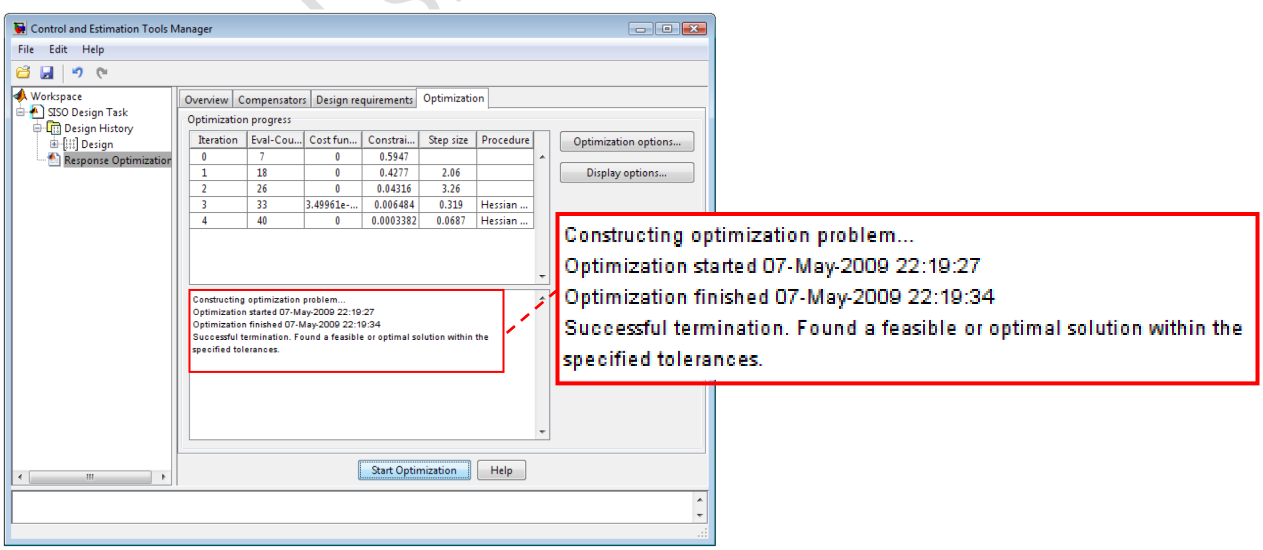
حال با انتخاب گزینه ی start Optimization ، بهینه سازی کنترلر انجام خواهد شد.Matlab به صورت اتوماتیک،بهترین کنترلری که در شرایط مسئله صدق کند را پیدا خواهد کرد.

شکل۱۲ ، روند این پروسه را نشانمی دهد:



شکل12

پنجره ی شکل۱۳ نشان دهنده ی پایان بهینه سازی است و بیانگر این است کنترلر مناسب یافت شده است :

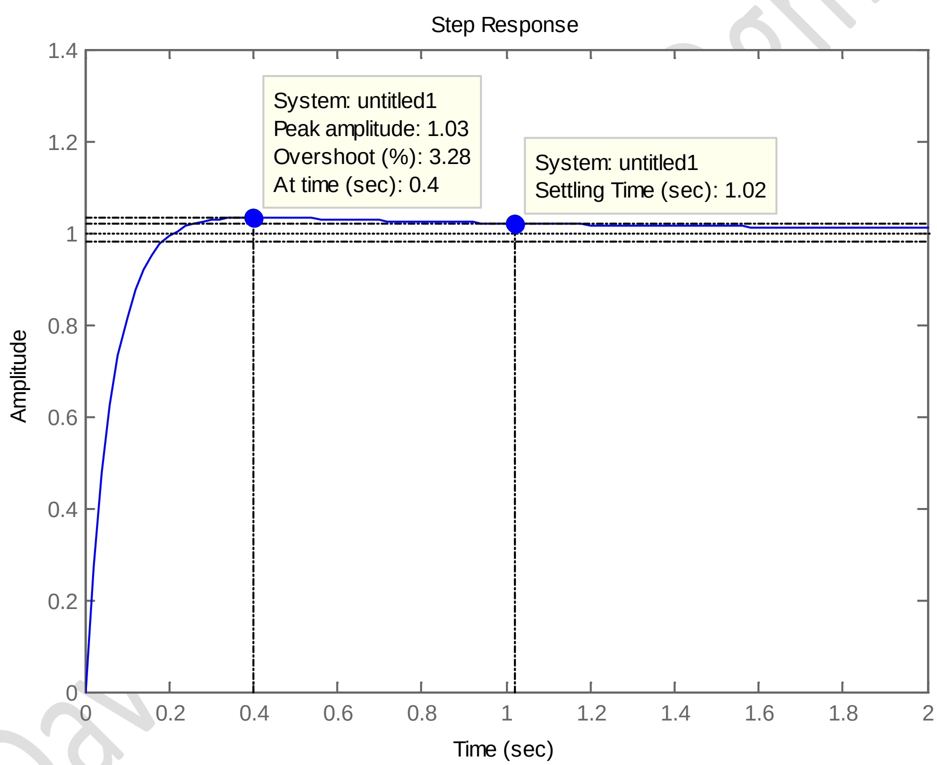


شکل13

کنترار طراحی شده دستی به صورت زیر بهینه شد :

```
>> Gc

Zero/pole/gain:
15.8177 (s+1.335) (s+0.3625)
----------------------------
             s
```

برای اطمینان از شرایط مطلوب ، پاسخ پله ی plant و کنترلر را بدست می آوریم:



شکل14

شرط اول را بررسی می کنیم :

$$Gc=\frac{15.8177(s+1.335)(s+.3625)}{s}$$

$$=\underbrace{9.3182}_{k_p}(1+\frac{1}{\underbrace{3.5077}_{T_i}s}+\underbrace{0.5891}_{T_d}s)$$

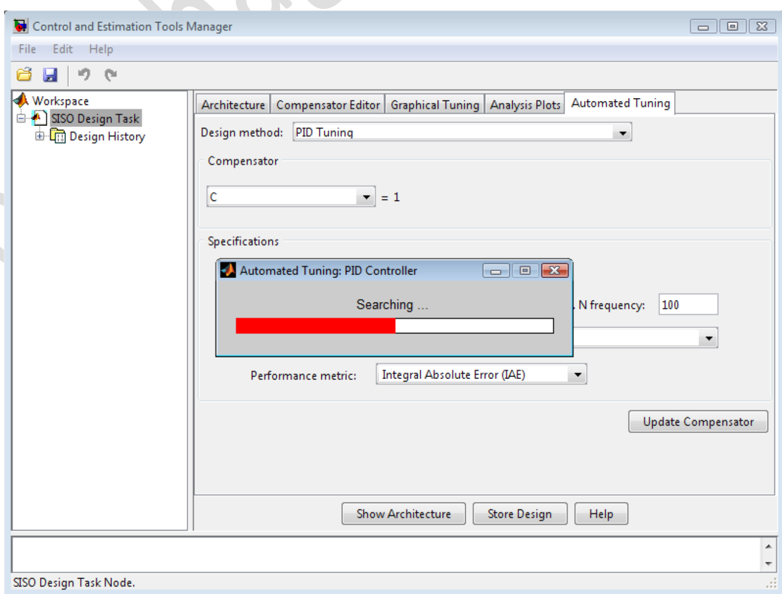
$$T_i=3.5077>\frac{1}{1+k_pT_d}=\frac{1}{1+9.3182\,(.5891)}=0.1541 \quad \surd$$

ملاحظه می شود که با کنترلر فوق ، شرایط مسئله برآورده شده است .

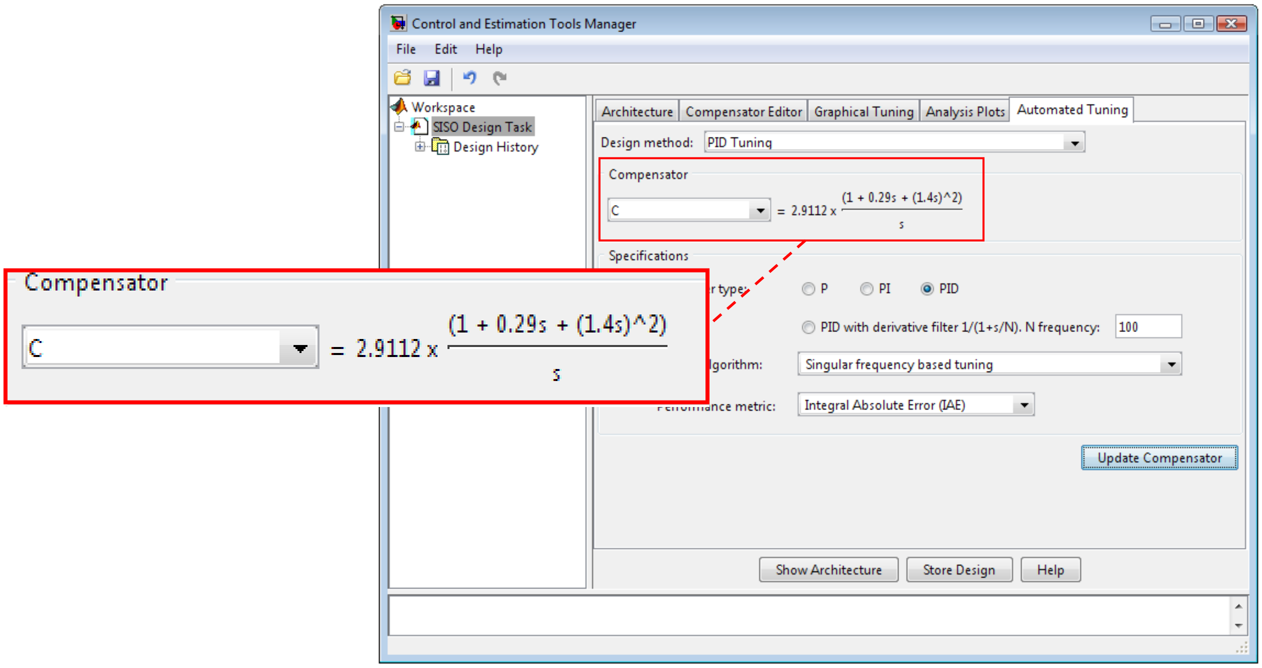
تا اینجا روند کار به این صورت شد که ابتدا کنترلری به صورت دستی طراحی شد و سپس توسط Matlab بهینه شد.حال می خواهیم طراحی اولیه را هم با Matlabانجام دهیم.

```
>> s=tf('s');
>> G=1/s/(s+1);
>> sisotool(G)
```

از تب Automated tuning ، این بار روش طراحی را PID Tuning انتخاب می کنیم و سپس Update Compensatorرا کلیک می کنیم.در این حین یک کنترلر PID طراحی می شود.

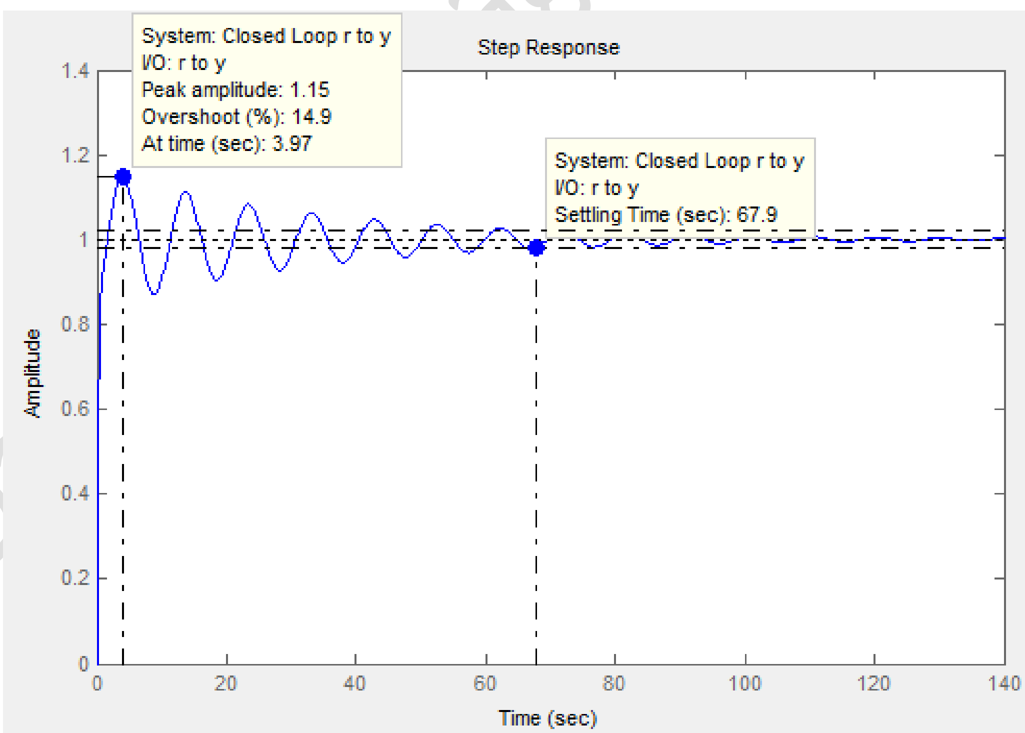


شکل15



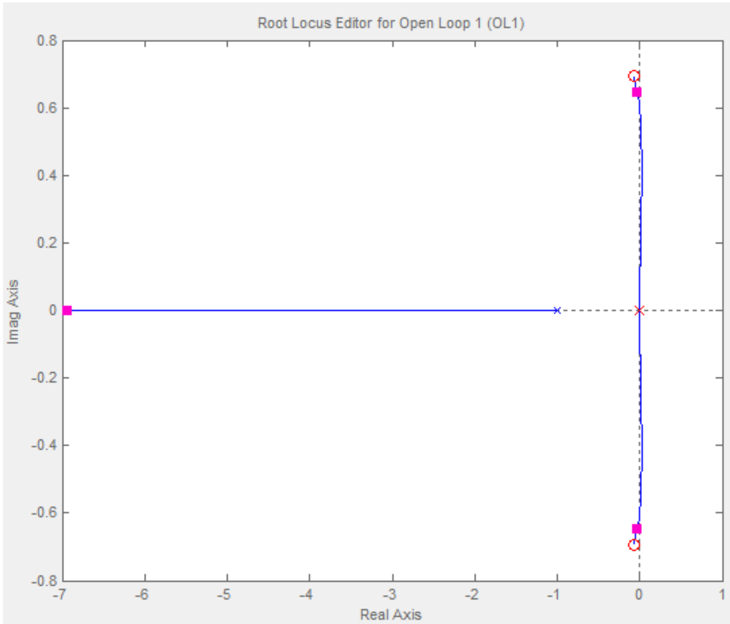
شکل 16

پاسخ پله ی سیستم با کنترلر اولیه به صورت زیر است :



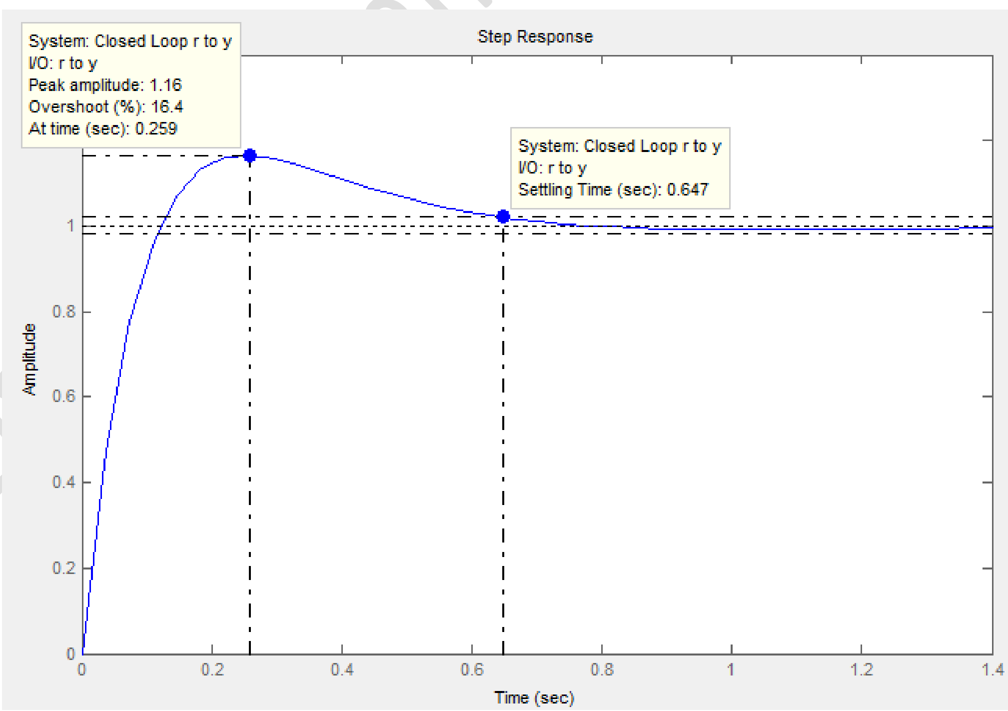
شکل 17

مکان ریشه ی کنترلر و plant :



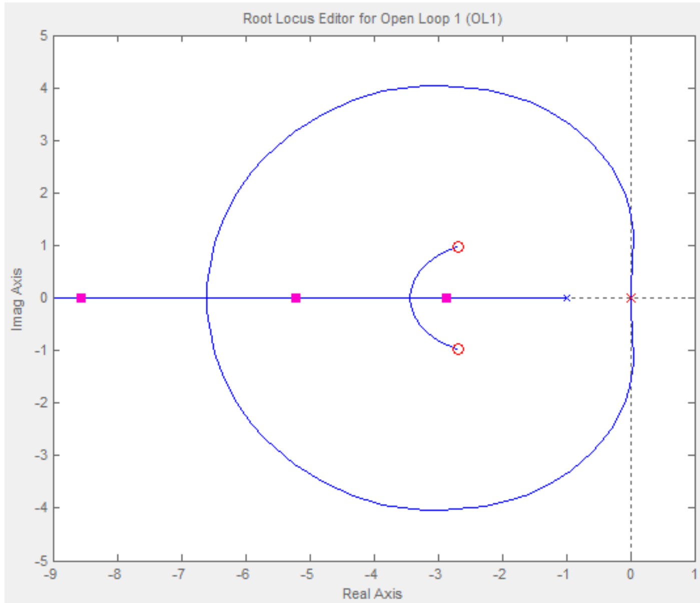
شکل 17

حال باید با تغییر مکان قطب ، پاسخ را به مقادیر دلخواه نزدیک کنیم.یک تغییر نمونه در مکان صفرها و مقدار گین به پاسخ زیر منجر می شود :



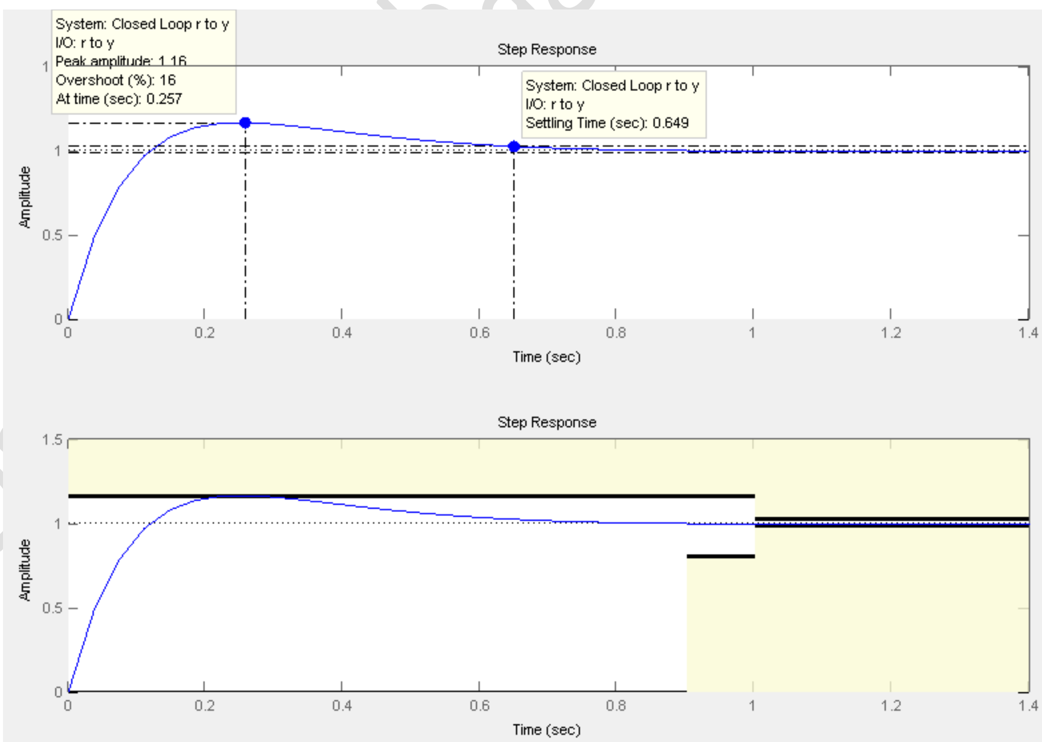
شکل 18

مکان ریشه ی کنترلر پس از تغییر محل صفرها و مقدار گین :

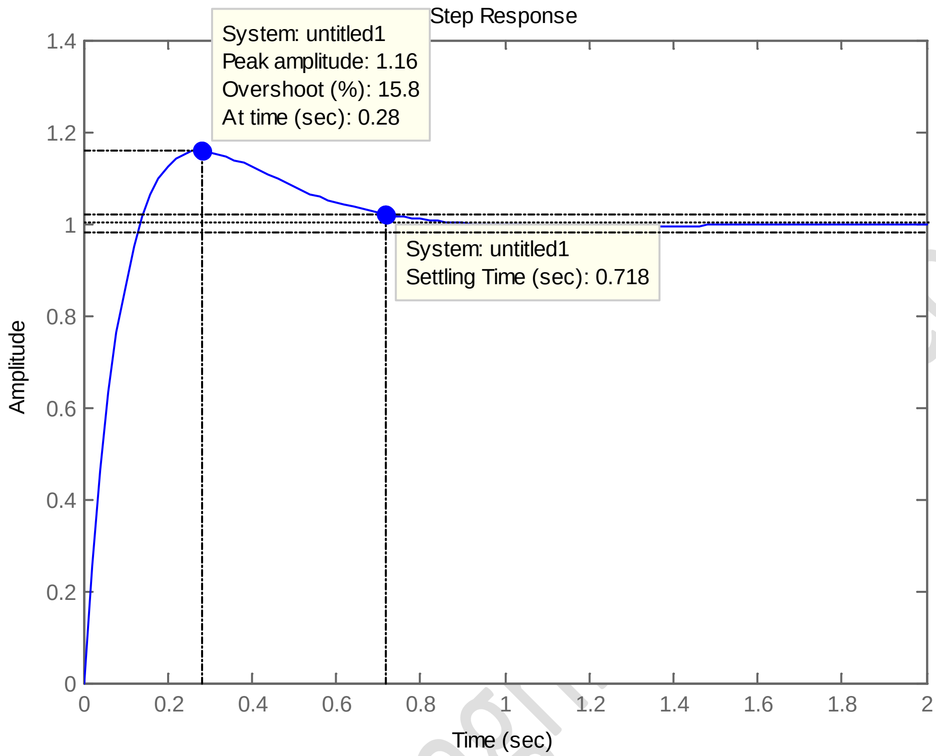


شکل 19

با تکرار مرحله بهینه سازی ، این بار برای این کنترلر ،پاسخ بهینه به صورت زیر بدست آمده است :



شکل 20



شکل 21

و کنترلر PID بهینه به صورت زیر بدست می آید :

```
>> C

Zero/pole/gain:
15.9795 (s^2 + 5.321s + 8.008)
-------------------------------
              s
```

شرط اول را بررسی می کنیم :

$$C=\frac{15.9795\,(s^2+5.321\,s+8.008)}{s}$$

$$=\underbrace{3.0031}_{k_p}(1+\frac{1}{\underbrace{.6645}_{T_i}s}+\underbrace{.1879}_{T_d}s)$$

$$T_i=.6645>\frac{1}{1+k_pT_d}=\frac{1}{1+3.0031\,(.1879)}=0.5643\quad\surd$$

ملاحظه می شود که با کنترلر فوق ، شرایط مسئله برآورده شده است .