

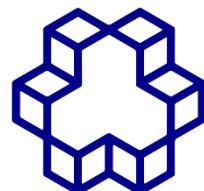
به نام خدا



کرد پوشه‌نی اپاک

دانشگاه صنعتی خواجه نصیرالدین طوسی

دانشکده برق



دانشگاه صنعتی خواجه نصیرالدین طوسی

شناسایی سیستم

گزارش تمرین شماره دوم

[علیرضا قاسمی]

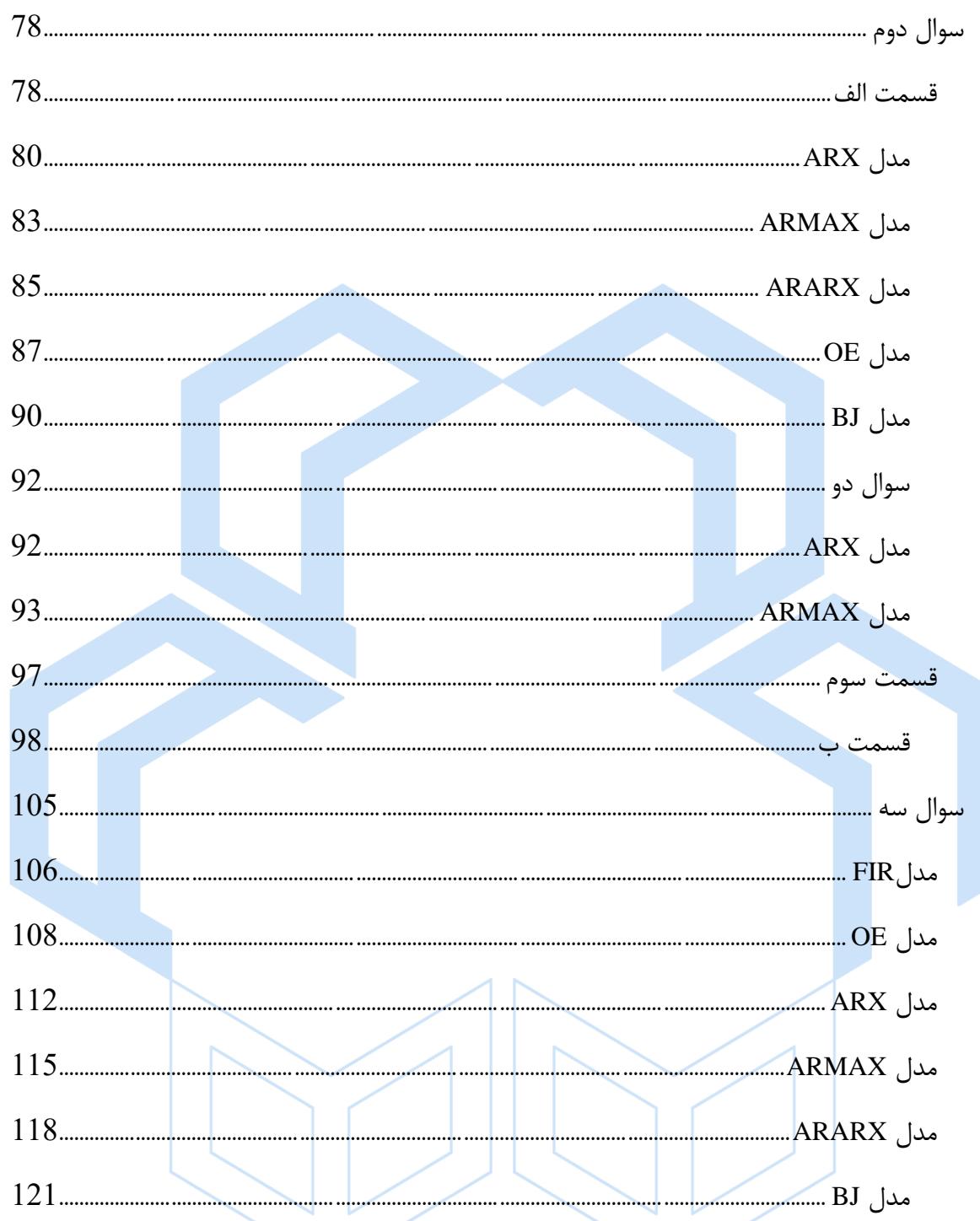
[40110214]

دکتر مهدی علیاری شوره دلی

## فهرست مطالب

عنوان	شماره صفحه
بخش ۱: سوالات تحلیلی	5
سوال اول	5
سوال دوم	7
سوال سه	9
سوال چهارم	10
بخش ۲: سوالات شبیه سازی	12
سوال اول	12
به ازای ورودی اول	16
ARX	17
ARMAX	17
صفر و قطب های سیستم و سیستم های شناسایی شده	18
برای ورودی دوم	23
ARX	23
ARMAX	23
poles and zeros	24
Errors	25
برای ورودی سوم	30
برای ورودی چهارم	35
ARX	35
ARMAX	35
poles and zeros	36
Errors	37

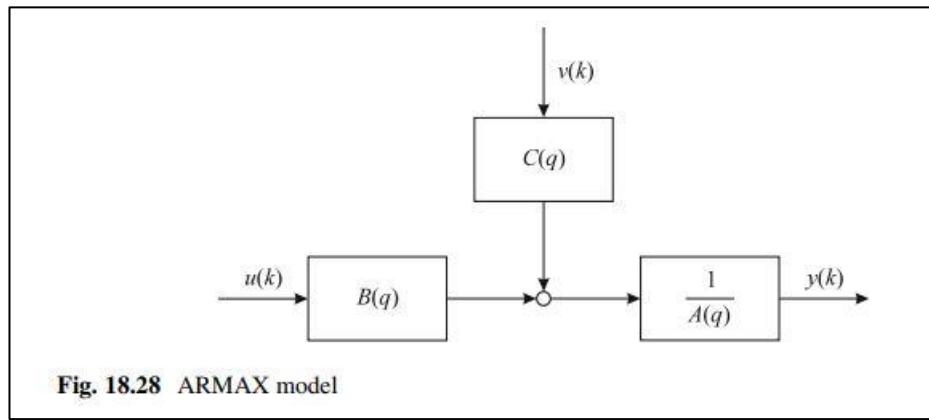
37	..... Plot
41	..... سوال دوم
50	..... سوال سوم
51	..... برای نویز کم
52	..... ARX
52	..... ARMAX
52	..... ARARX
52	..... OE
53	..... BJ
53	..... Figure and result
60	..... برای نویز متوسط
61	..... ARX
61	..... ARMAX
61	..... ARARX
61	..... OE
62	..... BJ
62	..... Errors
63	..... Figure and result
69	..... AIC
69	..... برای نویز زیاد
70	..... ARX
70	..... ARMAX
70	..... ARARX
71	..... OE
71	..... BJ
71	..... Errors
72	..... Figure and result
78	..... AIC



## بخش ۱: سوالات تحلیلی

### سوال اول

طبق شکل داخل کتاب ARMAX(Autoregressive Moving Average with Exogenous Input) برای الگوریتم Nelles ، دیاگرام بلوکی آن به صورت زیر است :



روابط مدل ARMAX به فرم زیر است :

$$Ay(t) = Bu(t) + Cn(t)$$

با توجه به اینکه  $C$  پارامتر رگرسور نویز است نمیتوانیم از روش LS استفاده کنیم برای حل این مشکل می بایست از بهترین تخمین نویز که در دسترس است استفاده کنیم. بهترین تخمینی که برای نویز می توان در نظر گرفت برای با آخرین خطای شناسایی است. در واقع ما می خواهیم که یک مرحله از تخمین حداقل مربعات بازگشتی استفاده کنیم و خطای به دست آمده را برابر با نویز رنگی سیستم در نظر بگیریم. این خطا برابر با خطا پیش بینی است برای همین خطای پیش بینی PEM را باید بدست اوریم که به

صورت زیر خواهد بود :

$$\begin{aligned}\varepsilon(t) &= y(t) - \hat{y}(t-1) \\ \varepsilon(t) &= y(t) - \phi^T(t-1)\hat{\theta}(t-1)\end{aligned}$$

در این حالت انگار برای نویز سفید یک رویتگر قرار داده ایم. و مقدار آن را با مقدار خطا پیش بینی رویت میکنیم در رابطه با رگرسورها نیز میتوان نوشت:

$$\begin{aligned}A(q)y(t) &= B(q)u(t) + C(q)e(t) \rightarrow \\ y(t) &= -a_1y(t-1) + \dots + b_1u(t) + \dots + b_my(t-m+1) + e(t) + \dots \\ &\quad + c_{n_c}e(t-n_c)\end{aligned}$$

$$\begin{aligned}\rightarrow \varepsilon(t) &= (1 - C(q^{-1}))\varepsilon(t) + A(q^{-1})y(t) - B(q^{-1})U(t) \\ \varepsilon(t) &= -c_1\varepsilon(t-1) - \dots - c_{nc}\varepsilon(t-nc) + a_1y(t-1) + \dots + a_{na}y(t-na) \\ &\quad - b_1u(t-1) - \dots - b_{nb}u(t-nb)\end{aligned}$$

در این حالت، یک نویز سفید وجود دارد ولی با توجه به جایگاه این نوع نویز در فرمول ، تخمین را بایاس دار نمیکند، بر تخمین بی اثر است. زیرا در فرمول بالا نویز دیگر رگرسور پارامتر های شناسایی نیست بنابراین، برای سایر جملات نیز داریم:

$$\begin{aligned}\hat{\theta}(t-1) &= [a_1 \quad \dots \quad a_n \quad b_1 \quad \dots \quad b_m \quad c_1 \quad \dots \quad c_{n_c}] \\ \phi^T(t-1) &= \\ [-y(t-1) \quad \dots \quad -y(t-n) \quad u(t) \quad \dots \quad u(t-m+1) \quad \varepsilon(t-1) \quad \dots \quad \varepsilon(t-n_c)]\end{aligned}$$

به صورت زیر میتوان این الگوریتم را توضیح داد:

1. مقدار دهی اولیه:  $\hat{\theta}(0)$  و  $p(0)$  با توجه به داشتن قبلی ساختار یک حدسی از ساختار سیستم میزنیم و مقادیر  $na, nb, nc$  را در نظر میگیریم و برای دیگر مقادیر اولیه مقدار دلخواه در بازه قابل قبول و منطقی قرار میدهیم

2. در گام های زمانی  $t=n$ ، خروجی را محاسبه میکنیم.

3. مقدار  $U$ ،  $e$  و  $u$  را از  $\phi(n)$  به دست میاوریم.

$$\phi(n) = [-y(n-1) \quad U(n-1) \quad e(n-1)]^T \quad -1$$

4. الگوریتم RLS را برای  $p(n)$  و  $\hat{\theta}(n-1)$  اجرا میکنیم.

$$e(n) = y(n) - \phi(n)^T \hat{\theta}(n-1)$$

$$P(n) = \frac{1}{\lambda} \left[ P(n-1) - \frac{P(n-1)\phi(n)\phi(n)^T P(n)}{\lambda + \phi(n)^T P(n-1)\phi(n)} \right] \quad -2$$

$$K = P(n)\phi(n)$$

$$\hat{\theta}(n) = \hat{\theta}(n-1) + Ke(n)$$

5. دوباره الگوریتم را اجرا میکنیم و نتایج حاصل الگوریتم RLS را جایگزین نتایج قبلی میکنیم.

6. قرار میدهیم  $t=n+1$  و به مرحله 2 میرویم.

این الگوریتم همان الگوریتم ELS است که در آن به جای تخمین حداقل مربعات از حداقل مربعات بازگشتی استفاده میکنیم.

## سؤال دوم

(الف)

سیستم بدون نویز در فضای حالت را میتوان در حالت کلی به شکل زیر در نظر گرفت

$$x[K+1] = Ax[K] + Bu[K]$$

$$y[K] = Cx[K] + Du[K]$$

میتوان معادلات فضای حالت ارائه شده در بالا را با در نظر گرفتن چندین جمله از آن به شکل زیر نشان

:داد:

$$Y_h = \Gamma_i X + H_t U_h$$

$$Y_h = \begin{bmatrix} y[k] & y[k+1] & \dots & y[k+j-1] \\ y[k+1] & y[k+2] & \dots & y[k+1] \\ \dots & \dots & \dots & \dots \\ y[k+i-1] & y[k+i] & \dots & y[k+j+i-2] \end{bmatrix}$$

$$X = [x[k] \quad x[k+1] \quad \dots \quad x[k+j-1]]$$

$$U_h = \begin{bmatrix} u[k] & u[k+1] & \dots & u[k+j-1] \\ u[k+1] & u[k+2] & \dots & u[k+1] \\ \dots & \dots & \dots & \dots \\ u[k+i-1] & u[k+i] & \dots & u[k+j+i-2] \end{bmatrix}$$

$$\Gamma_i = \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^{i-1} \end{bmatrix}$$

$$H_i = \begin{bmatrix} D & 0 & 0 & \dots & 0 \\ CB & D & 0 & \dots & 0 \\ CAB & CB & D & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ CA^{i-2}B & CA^{i-3}B & CA^{i-4}B & \dots & D \end{bmatrix}$$

از رابطه بالا داریم:

$$Y_h U_h^\perp = \Gamma_i X U_h^\perp$$

داریم:

$$\text{rank}(\Gamma_i X U_h^\perp) = \text{rank}(X U_h^\perp) - \dim(\text{span}_{\text{col}}(X U_h^\perp) \cap \text{span}_{\text{row}}(\Gamma_i)^\perp)$$

اگر ماتریس  $i$  دارای رنک کامل باشد، معادله بالا جواب دارد و میتوان  $x$  را به دست اورد.

از رابطه بالا داریم:

$$\text{rank}(Y_h U_h^\perp) = \text{rank}(X) = n$$

با اعمال ورودی ضربه داریم:

$$Y_{h1} = \Gamma_i X_1 + H_t U_{h1}$$

$$Y_{h2} = \Gamma_i X_2 + H_t U_{h2}$$

$$H_1 = \begin{bmatrix} Y_{41} \\ U_{41} \end{bmatrix}, H_2 = \begin{bmatrix} Y_{A_2} \\ U_{12} \end{bmatrix}$$

$$Y_{h1} = \begin{bmatrix} y[k] & y[k+1] & \dots & y[k+j-1] \\ y[k+1] & y[k+2] & \dots & y[k+j] \\ y[k+2] & y[k+3] & \dots & y[k+j+1] \\ \vdots & \vdots & & \vdots \\ y[k+i-1] & y[k+i] & \dots & y[k+j+i-2] \end{bmatrix}$$

$$Y_{h2} = \begin{bmatrix} y[k+1] & y[k+i+1] & \dots & y[k+i+j-1] \\ y[k+1+1] & [(k+i+2)] & \dots & y[k+i+1] \\ y[k+i+2] & y[k+i+3] & \dots & y[k+i+j+1] \\ \vdots & \vdots & & \vdots \\ y[k+2i-1] & y[k+2] & \dots & y[k+2i+j-2] \end{bmatrix}$$

در صورتی که داشته باشیم:

$$X_2 = [x[k+i] \quad x[k+i+1] \quad \dots \quad x[k+i+j-1]]$$

میتوان نتیجه گرفت که:

$$\text{span}_{\text{row}}(x_2) = \text{span}_{\text{row}}(H_1) \cap \text{span}_{\text{row}}(H_2)$$

در این حالت بین دو فضای  $(H_1)$  و  $(H_2)$  اشتراک ایجاد شده است و در این اشتراک جوابی برای  $x$  وجود دارد. اگر پاسخ ضربه سیستم موجود باشد و فرض عدم وجود نویز برقرار باشد، میتوان با استفاده از روش LS، دینامیک های سیستم و پارامتر های فضای حالت را به دست اورد.

(ب)

در حالت کلی معادله فضای حالت سیستم همراه با نویز فرایند و نویز اندازه گیری به صورت زیر است

$$x(k+1) = Ax(k) + Bu(k) + Gw(k)$$

$$y(k) = Cx(k) + Du(k) + v(k)$$

حال اگر دینامیک های سیستم را باز کنیم داریم

$$x(k) = A^k x(0) + \sum_{i=0}^{k-1} A^{k-i-1} Bu(i) + \sum_{i=0}^{k-1} A^{k-i-1} Gw(i)$$

در این حالت اگر بخواهیم پارامتر  $A^{k-i-1}G$  را محاسبه کنیم با توجه به اینکه رگرسور این پارامتر نویز است به مشکل بر میخوریم و باید از روش های مانند ARARX و ARMAX که برای حل این مشکل طراحی شده اند استفاده کنیم.

(ج)

در فضای حالت، اگر ماتریس  $D$  غیر صفر باشد به این معنی است که سیستم ناسره است. در سیستم های ناسره درجه صورت بیشتر از درجه مخرج است و به همین خاطر در سیستم های ناسره در فرکانس های پایین سخت شناسایی میشوند. در این حالت به علت پیش فازی یا پس فازی، تخمین دقیق پارامتر هایی مانند ثابت زمانی و بهره با مشکلاتی روبرو شود. همچنین، اگر سیستم رویت پذیر نباشد، نمی توان آنها پارامتر های مربوط به بخش رویت ناپذیر ان را یافت. و در واقع تنها میتوانیم دینامیک های رویت پذیر سیستم ها را شناسایی کنیم. و برای دینامیک های رویت ناپذیر شاید بتوانیم یک تخمین غیر دقیق داشته باشیم اما نمیتوانیم با روش هایی که خوانده ایم شناسایی انجام دهیم.

### سوال سه

در سیستم هایی با قطب روی محور  $\omega_z$ ، اگر سیستم توسط یک سیگنال سینوسی هم فرکانس با قطب های موهومی سیستم، به مدت زیادی تحریک شود، سبب ناپایداری سیستم می شود. شناسایی سیستم های ناپایدار کار بسیار سختی است و مطلوب نیست اما اگر برای لحظه ای کوتاه برای سیستم با قطب موهومی چنین شرایطی برقرار شود، دیگر مشکلی برای سیستم پیش نمی آید. بنابراین در شناسایی چنین سیستم هایی، ورودی نباید از نوع سینوسی باشد. لذا استفاده از سیگنال شناسایی مانند نویز یا PRBS مشکلی برای سیستم به لحاظ پایداری ایجاد نمی کند و همچنین طیف گسترده ای از فرکانس ها

را تحریک می کند. لذا با استفاده از این طیف سیگنال های شناسایی و انتخاب مدل مناسب از خانواده مدل های خطی می توان سیستم را به خوبی شناسایی کرد.

بنابراین اگر ورودی طیف وسیعی از فرکانس ها در برگیرد، مشکل ساز نخواهد بود. اما اگر ورودی هم فرکانس با فرکانسی باشد که دیاگرام بود پیک زده است، پدیده تشدید اتفاق می افتد و سیستم ناپایدار میشود و امکان شناسایی سیستم دیگر وجود نخواهد داشت.

در سیستم های انتگرالی به علت وجود قطب در مبدأ، ترم انتگرال گیر درتابع تبدیل ظاهر می شود که سبب می شود از ورودی به نوعی انتگرال گیری انجام شود. حال اگر سیگنال ورودی ما جهت شناسایی دارای مقدار DC باشد یا به عبارت دقیق تر ، تبدیل فوریه سیگنال ورودی در  $\omega = 0$  ، مقداری مخالف صفر داشته باشد، با گذشت زمان طولانی، مقدار خروجی زیادتر شده و باعث ناپایداری سیستم میشود . به همین دلیل برای شناسایی چنین سیستم هایی باید توجه داشت که مقدار DC سیگنال شناسایی حتما صفر باشد. لذا می توان نتیجه گرفت که سیگنالی مانند PRBS که دارای مقدار DC است برای شناسایی این نوع سیستم ها مناسب نیست ولی اگر از سیگنالی مانند نویز سفید با میانگین صفر به عنوان سیگنال شناسایی استفاده شود، با انتخاب مدل مناسب از خانواده مدل های خطی می توان سیستم را به خوبی شناسایی کرد.

#### سوال چهارم

تابع همبستگی یک معیار ارزیابی ارتباط بین دو متغیر یا دنباله از داده با یک دیگر است. این تابع یک معیاری است برای اینکه نشان دهد چقدر دو متغیر یا دنباله از دادهها با یکدیگر هماهنگ و متناسب هستند. به بیان دیگر نشان دهنده اندازه تغییرات همزمان دو سیگنال یا شیفت های زمانی انها با یکدیگر هستند تابع همبستگی را میتوان از رابطه زیر به دست اورد:

$$C(h) = \frac{\sum_{i=1}^n (Z(x_i) - \bar{Z})(Z(x_i + h) - \bar{Z})}{\sum_{i=1}^n (Z(x_i) - \bar{Z})^2}$$

در برخی برخی مسئله ها، ترکیب تابع همبستگی با روش کمترین مربعات (COR-LS) به عنوان یک تابع هزینه جدید می تواند به دقت و کارایی مدل کمک کند برای این کار تابع همبستگی را با یک ضریب  $\lambda$  به تابع هزینه مسئله LS اضافه میکنیم و با استفاده از پارامتر  $\lambda$  میتوانیم تاثیر هر یک از این جملات را در تابع هزینه نهایی کنترل کنیم.

$$\text{cost} = \sum_{i=1}^n (y_i - (mx_i + b))^2 + \lambda \sum_{h=1}^k (C(h) - \hat{C}(h))^2$$

مزیت‌های استفاده از این تابع هزینه نسبت به روش تخمین مجموع مربعات :

1. COR-LS در مواردی که نیاز به پیچیدگی بیشتر برای مدل و مدلسازی ارتباطات غیر خطی

بین داده‌ها باشد باعث می‌شود که نسبت به روش LS متداول نتایج بهتری از خود نشان دهد

2. اگر داده‌ها الگوهای زمانی داشته باشند، تابع همبستگی می‌تواند اطلاعات بیشتری از ارتباطات زمانی بین داده‌ها را نشان دهد. این اطلاعات زمانی ممکن است در تخمین پارامترهای مدل کمترین مربعات (LS) در نظر گرفته نشوند.

3. COR-LS اگر داده‌های ورودی شامل نویز باشد استفاده از این روش باعث بهبود نتایج روش LS معمولی می‌شود

روابط ریاضی این مدل را می‌توان به شکل زیر خلاصه کرد:

$$\begin{aligned} y(k) &= b_1 u(k-1) + \dots + b_m u(k-m) - a_1 y(k-1) - \dots - a_m y(k-m) \\ \rightarrow u(k-\kappa) y(k) &= b_1 u(k-\kappa) u(k-1) + \dots + b_m u(k-\kappa) u(k-m) \\ &\quad - a_1 u(k-\kappa) y(k-1) - \dots - a_m u(k-\kappa) y(k-m) \end{aligned}$$

داریم:

$$\begin{aligned} \sum_{k=\kappa+1}^N u(k-\kappa) y(k) &= \\ b_1 \sum_{k=\kappa+1}^N u(k-\kappa) u(k-1) + \dots + b_m \sum_{k=\kappa+1}^N u(k-\kappa) u(k-m) \\ - a_1 \sum_{k=\kappa+1}^N u(k-\kappa) y(k-1) - \dots - a_m \sum_{k=\kappa+1}^N u(k-\kappa) y(k-m) \end{aligned}$$

$$\begin{aligned} \text{corr}_{uy}(\kappa) &= b_1 \text{corr}_{uu}(\kappa-1) + \dots + b_m \text{corr}_{uu}(\kappa-m) \\ &\quad - a_1 \text{corr}_{uy}(\kappa-1) - \dots - a_m \text{corr}_{uy}(\kappa-m) \end{aligned}$$

به بیان دیگر:

$$U_{corr} = \begin{bmatrix} \text{corr}_{uu}(0) & \cdots \text{corr}_{uu}(1-m) & -\text{corr}_{uy}(0) & \cdots & -\text{corr}_{uy}(1-m) \\ \text{corr}_{uu}(1) & \cdots \text{corr}_{uu}(2-m) & -\text{corr}_{uy}(1) & \cdots & -\text{corr}_{uy}(2-m) \\ \vdots & \vdots & \vdots & & \vdots \\ \text{corr}_{uu}(l-1) & \cdots \text{corr}_{uu}(l-m) & -\text{corr}_{uy}(l-1) & \cdots & -\text{corr}_{uy}(l-m) \end{bmatrix}$$

$$\underline{y}_{corr} = \begin{bmatrix} \text{corr}_{uy}(1) \\ \text{corr}_{uy}(2) \\ \vdots \\ \text{corr}_{uy}(l) \end{bmatrix}$$

## بخش ۲: سوالات شبیه سازی

### سوال اول

در ابتدا مطابق خواسته سوال تابع *binrand* را میسازیم این تابع در ورودی خود پارامتر های  $T$  برابر با بردار زمانی  $N$  تعداد پالس های ورودی است.  $t_{min}$  کمترین عرض پالس است و  $t_0$  زمان شروع پالس است و در نهایت نیز *dist* نوع توزیع چگالی احتمال ورودی را مشخص میکند که یا توزیع نرمال یا توزیع یکنواخت باشد. این تابع در واقع ورودی PRBS ای را تولید میکند که شرط های زیر را دارا میباشد

عرض پالس های تولید شده باید بزرگتر از  $t_{min}$  باشد

مجموع عرض پالس ها برابر با مقدار  $T$  باید باشد

به دلیل افزایش غنا داده های تولیدی نباید هیچ دو پالس متوالی دارای مقدار یکسان باشند و همچنین اولین پالس نباید برابر با صفر باشد زیرا در این صورت اطلاعات مفیدی در خروجی بدست نمیاید و باید

یکی از مقادیر صفر یا منفی یک باشد

کد نوشته شده برای این تابع به صورت زیر است.

```
function [u,d]=binrand(T,N,tmin,t0,dist)
if (strcmp(dist,'normal')==1)
    m=randn(1,N-1);
elseif strcmp(dist,'uniform')==1
    m=rand(1,N-1);
else
    disp('Bad Command')
end
```

```

m_normalized=(m-min(m)) / (max(m)-min(m));
n=(((T(end)-t0+1) / (N-1))-tmin)*m_normalized+tmin;
d=round(n);
d=[d,T(end)-t0+1-sum(d)];
u=zeros(1,t0-1);
r=round(rand(1,1))+1;
for i=1:length(d)
    switch r
        case 1
            u=[u,ones(1,d(i))];
            r=round(rand(1,1))+2;
        case 2
            u=[u,-ones(1,d(i))];
            r=round(rand(1,1))+2;
            r=1;
        case 3
            u=[u,0*ones(1,d(i))];
            r=round(rand(1,1))+1;
        otherwise
            end
    end
end
u=u';
end

```

خلاصه عملکرد تابع

در ابتداء با توجه به نوع توزیع خواسته شده به تعداد  $N-1$  عدد تصادفی با توزیع گفته شده تولید

میکنیم

```

if (strcmp(dist,'normal')==1)
    m=randn(1,N-1);
elseif strcmp(dist,'uniform')==1
    m=rand(1,N-1);
else
    disp('Bad Command')
end

```

سپس اعداد تصادفی تولید شده را نرمالیزه میکنیم و به بازه صفر تا یک انتقال میدهیم

```
m_normalized=(m-min(m)) / (max(m)-min(m));
```

و با استفاده از دستور زیر اعداد بدست امده در مرحله قبلی را به بازه

$(tmin, ((T(end)-t0+1) / (N-1)))$

بالا انتقال میدهیم

```
n=(((T(end)-t0+1)/(N-1))-tmin)*m_normalized+tmin;
```

سپس اعداد بدست امده را به نزدیک ترین عدد صحیح گرد میکنیم که در نتیجه  $N-I$  عرض پالس با مشخصات خواسته شده تولید کرده ایم و عرض پالس اخر را به گونه ای تنظیم میکنیم که مجموع عرض پالس ها دقیقا برابر با مقدار خواسته شده باشد

```
d=[d,T(end)-t0+1-sum(d)];  
u=zeros(1,t0-1);  
r=round(rand(1,1))+1;  
for i=1:length(d)  
    switch r  
        case 1  
            u=[u,ones(1,d(i))];  
            r=round(rand(1,1))+2;  
        case 2  
            u=[u,-ones(1,d(i))];  
            r=round(rand(1,1))+2;  
            r=1;  
        case 3  
            u=[u,0*ones(1,d(i))];  
            r=round(rand(1,1))+1;  
    otherwise  
        end  
    end
```

و در اخر به دلیل اینکه ورودی های سیگنال به شکل قابل قبول برای تولباکس متلب باشد از ورودی بدست امده ترانهاده میگیریم

حال چهار سیگنال ورودی خواسته شده در صورت سوال را با استفاده از کد زیر ساخته میشود و به عنوان ورودی شناسایی از انها استفاده میشود در ادامه برای استفاده از هر یک ورودی ها مقدار متغیر  $K$  را متناسب با شماره ورودی تغییر میدهیم

```

%% Input Data
T=800;
U1 = binrand(1:T, 10, 40, 1, 'normal');
U2 = binrand(1:T, 30, 10, 1, 'normal');
U3 = normrnd(0, 1, 1,T);
U4 = unifrnd(-1, 1, 1,T);

k =1
if k == 1
    U = U1;
elseif k == 2
    U = U2;
elseif k == 3
    U = U3';
else
    U = U4';
end

```

با استفاده از ساختار مشخص شده در صورت سوال برای مدل های ARX,ARMAX به شناسایی

سیستم میپردازیم

برای طولانی نشدن گزارش کار توضیحات تکراری مربوط به همه ورودی ها یک بار ذکر شده است

همانگونه که مشاهده میشود در همه ورودی ها ARX توانسته سیستم را با دقت 100 درصد

شناسایی کند و مقدار خطاهای  $MSE, FPE$  برابر با صفر یا مقدار خیلی کمی میباشد

در همه ورودی ها ARMAX توانسته سیستم را با دقت 100 درصد شناسایی کند و مقدار خطاهای

$MSE, FPE$  برابر با صفر یا مقدار خیلی کمی میباشد

صفر ها بدست امده از هر دو روش ARX,ARMAX در همه روش ها برابر با صفر های سیستم اصلی

میباشد

در شکل های زیر نمودار های خروجی مدل اصلی و مدل شناسایی شده نشان داده شده است که مدل شناسایی شده در همه ورودی ها به صورت کامل به مدل اصلی فیت شده است

در شکل های زیر نمودار های اتوکورولیشن خطأ و کراس کورولیشن خطأ و ورودی و خطأ و خروجی رسم شده است. در حالت ایده ال باید همه نمودار ها داخل ابی رنگ مشخص شده قرار گرفته باشند

فقط نمودار اتوکورولیشن خطابا خودش در زمان صفر باید مقدار یک داشته باشد و بجز آن در بقیه زمانها باید مقدار کمی داشته باشد و داخل باند اطمینان قرار بگیرند این نشان دهنده این است که خطابا خودش وابسته است و به نمودارهای شیفت یافته خود وابسته نیست

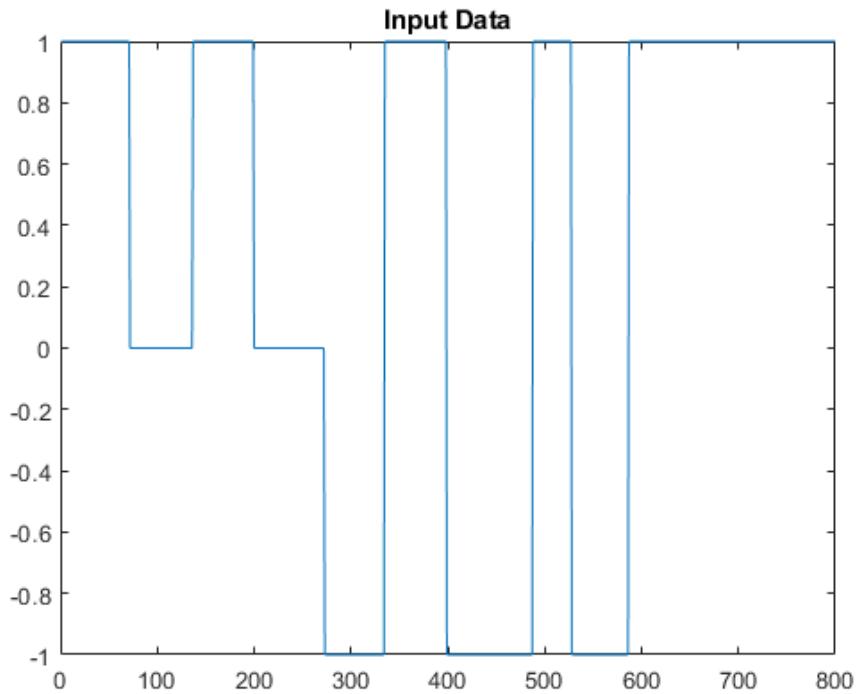
اگر مقادیر این داده ها درون بازه مشخص شده قرار نگرفته باشد نشان دهنده این است که هنوز اطلاعاتی درون داده ها وجود دارد که ساختار مدل پیشنهاد داده شده در صورت سوال و همچنین داده های ورودی نتوانسته این اطلاعات را استخراج کند

با فرض بدون نویز بودن سیستم هر چهار ورودی به خوبی توانسته اند سیستم را شناسایی کنند این نشان دهنده این موضوع است که ورودی ها و ساختارهای انتخاب شده برای شناسایی سیستم مناسب میباشند

اما از نظر نمودارهای اتوکورولیشن و کراش کورولیشن ورودی های سوم و چهارم نسبت به ورودی های اول حالت بهتری داشته اند و بیشتر داخل باند اطمینان قرار گرفته اند این نشان دهنده این موضوع است که درجه  $pe$  بودن انها نسبت به ورودی های اول و دوم بیشتر است و بین ورودی های اول و دوم ورودی دوم به نسبت سیگنال بهتری میباشد زیاد دارای پالس های بیشتر و با عرض پالس کمتر است

از نظر ساختار مدل ARX نسبت به مدل ARMAX دینامیک های بیشتری در نمودارهای کورولیشن در خود جای داده است و نتایج بهتری داشته است و در واقع بهتر توانسته سیستم را شناسایی کنم

به ازای ورودی اول



## ARX

```
estimated_y_1_arx =
Discrete-time ARX model: A(z)y(t) = B(z)u(t) + e(t)
  A(z) = 1 - 1.72 z^-1 + 0.9 z^-2
  B(z) = 0.48 z^-1 - 0.48 z^-2
Sample time: 0.1 seconds
```

### Parameterization:

Polynomial orders: na=2 nb=2 nk=1  
Number of free coefficients: 4  
Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.

### Status:

Estimated using ARX on time domain data "data\_1".  
Fit to estimation data: 100% (prediction focus)  
FPE: 1.353e-32, MSE: 1.333e-32

## ARMAX

```
estimated_y_1_armax =
Discrete-time ARMAX model: A(z)y(t) = B(z)u(t) + C(z)e(t)
  A(z) = 1 - 1.72 z^-1 + 0.9 z^-2
  B(z) = 0.48 z^-1 - 0.48 z^-2
  C(z) = 1 + 0.8211 z^-1
Sample time: 0.1 seconds
```

Parameterization:

Polynomial orders: na=2 nb=2 nc=1 nk=1

Number of free coefficients: 5

Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.

Status:

Estimated using ARMAX on time domain data "data\_1".

Fit to estimation data: 100% (prediction focus)

FPE: 1.529e-32, MSE: 1.51e-32

### صفرو قطب های سیستم و سیستم های شناسایی شده

zeros\_sys =

1

poles\_sys =

0.8600 + 0.4005i

0.8600 - 0.4005i

zeros\_and\_poles\_arx\_1

zeros\_arx\_1 =

1.0000

poles\_arx\_1 =

0.8600 + 0.4005i

0.8600 - 0.4005i

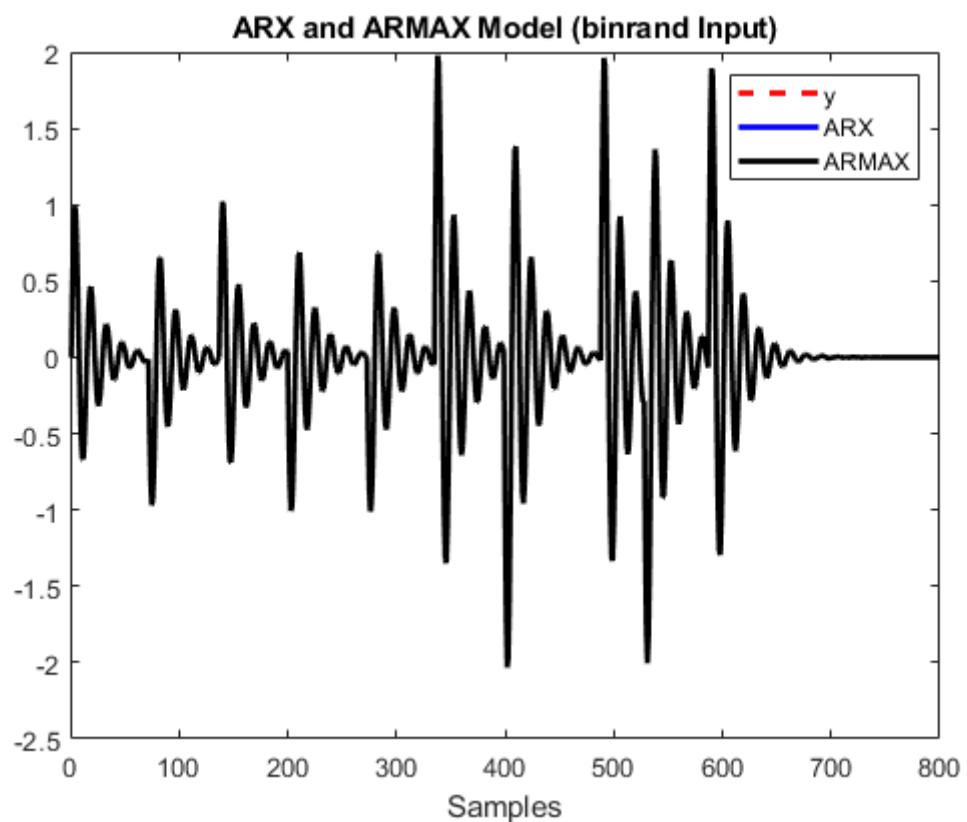
```
zeros_and_poles_armax_1
```

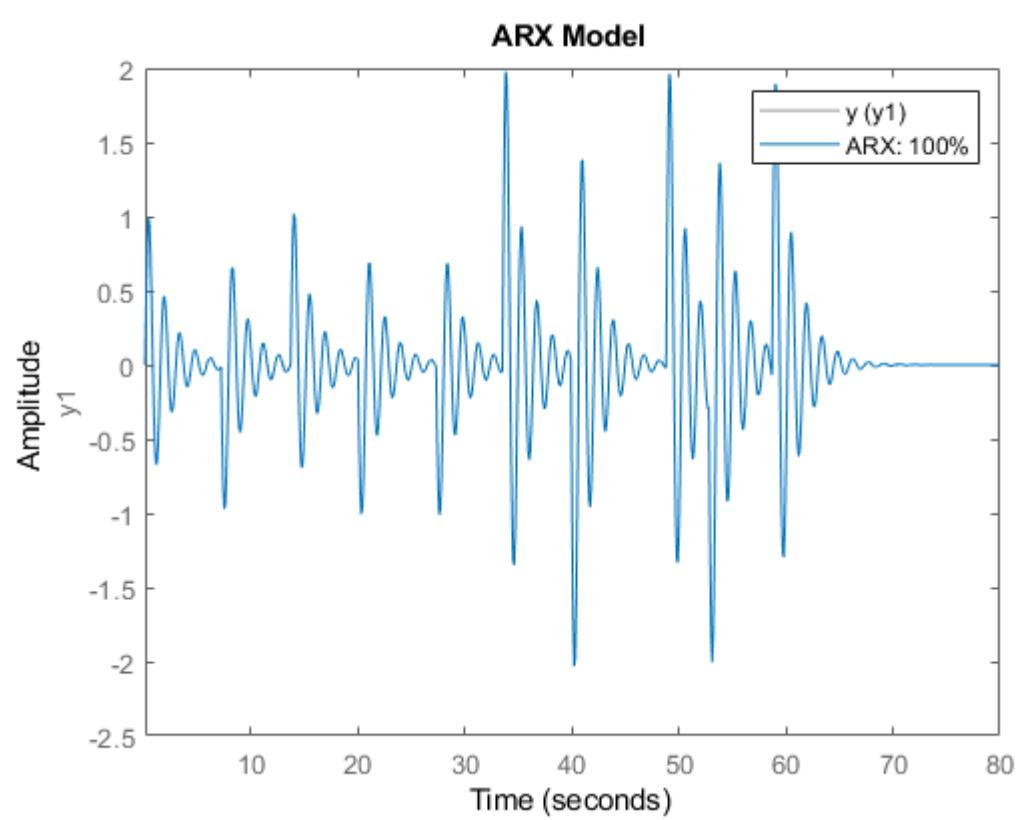
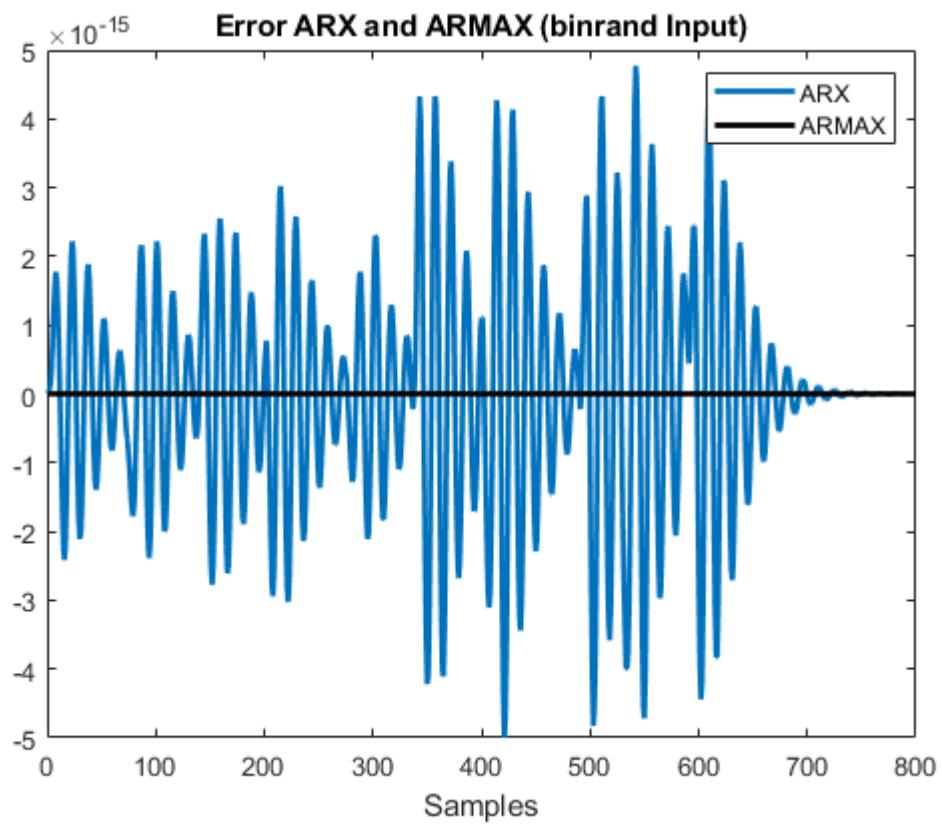
```
zeros_armax_1 =
```

```
1
```

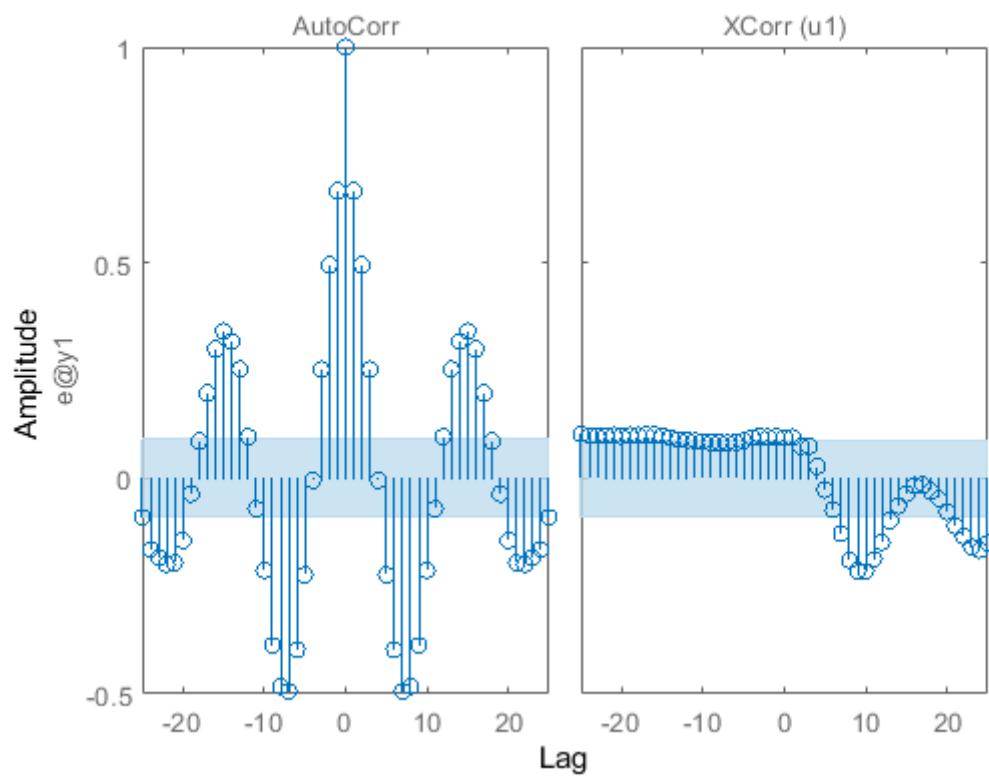
```
poles_armax_1 =
```

```
0.8600 + 0.4005i  
0.8600 - 0.4005i
```

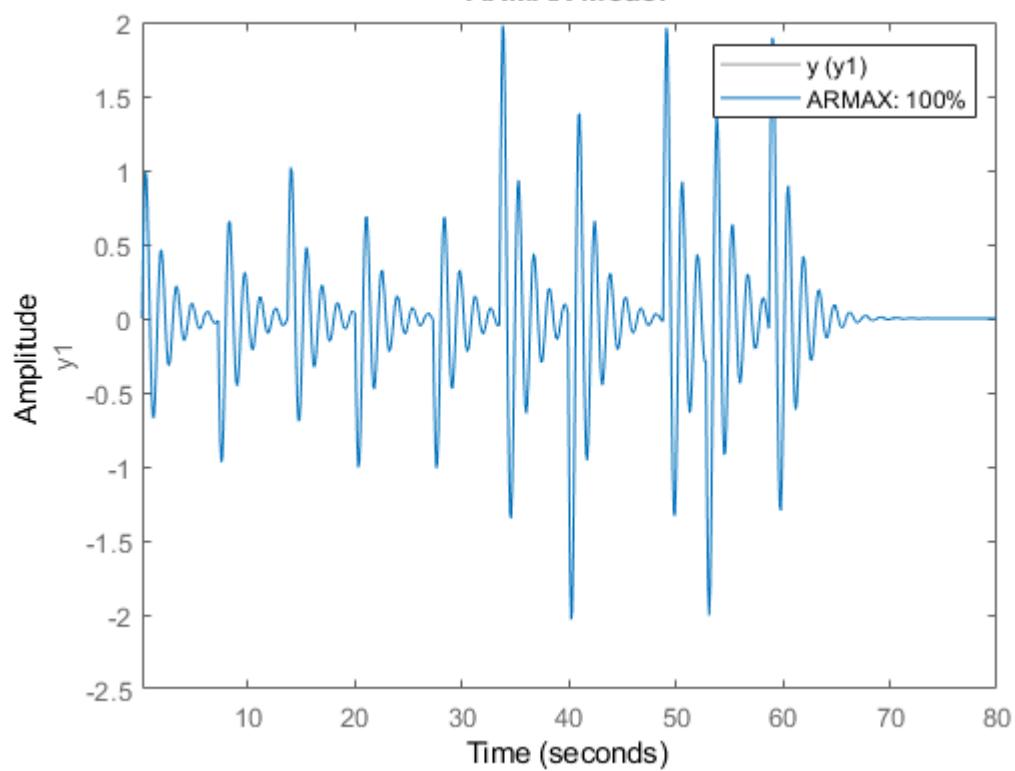




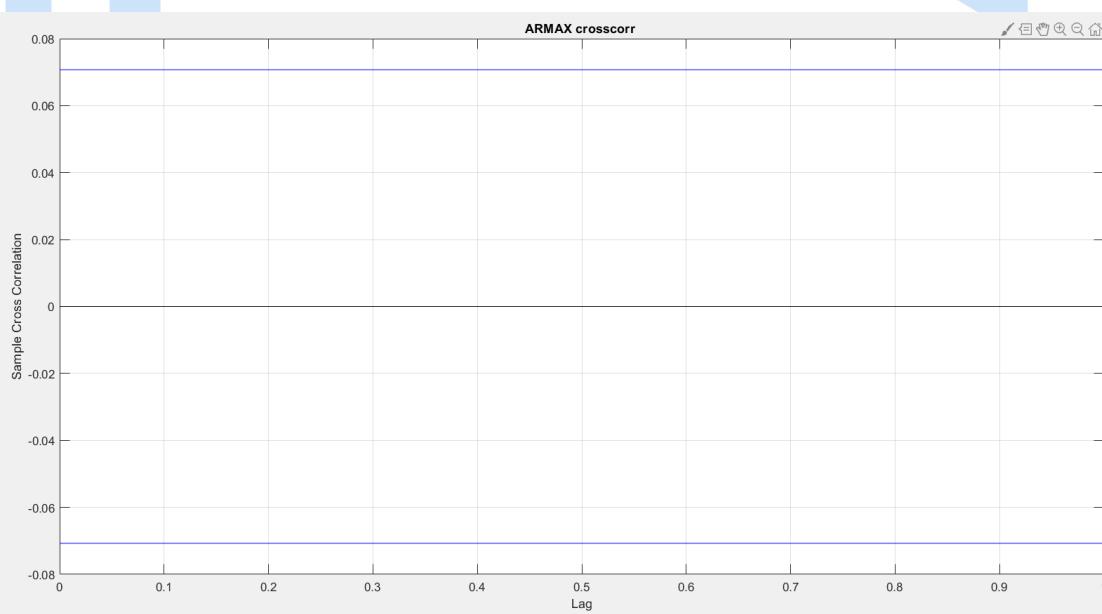
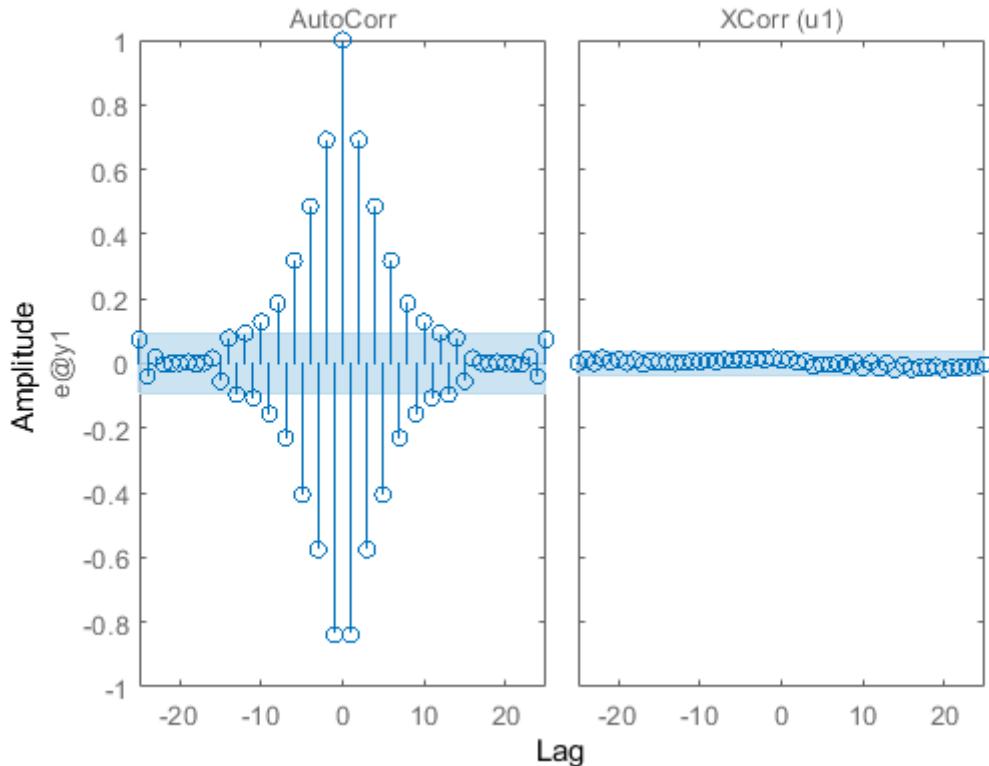
### ARX Model



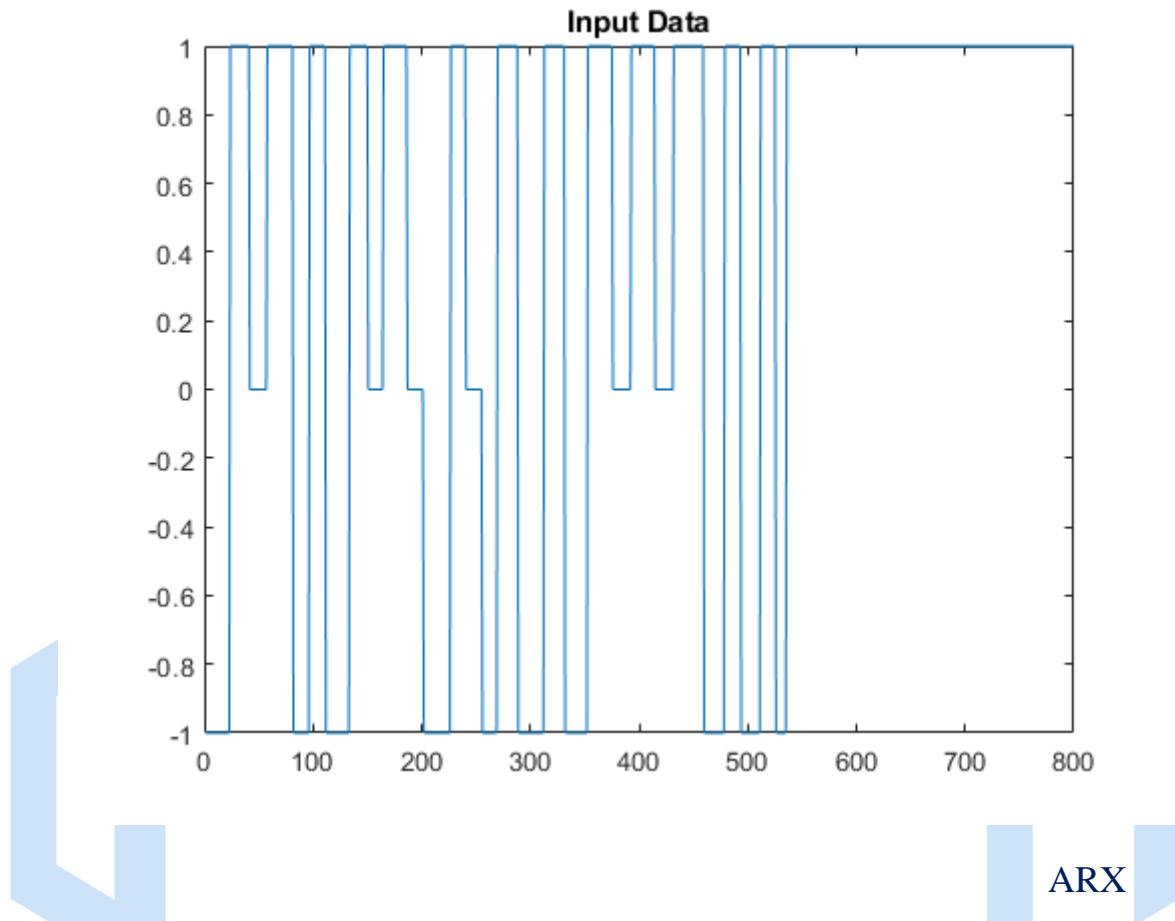
### ARMAX Model



### ARMAX Model



## برای ورودی دوم



Status:  
Estimated using ARX on time domain data "data\_1".  
Fit to estimation data: 100% (prediction focus)  
FPE: 5.856e-32, MSE: 5.769e-32

## ARMAX

```
estimated_y_1_armax =  
Discrete-time ARMAX model:  $A(z)y(t) = B(z)u(t) + C(z)e(t)$   
 $A(z) = 1 - 1.72 z^{-1} + 0.9 z^{-2}$ 
```

```
B(z) = 0.48 z^-1 - 0.48 z^-2
```

```
C(z) = 1 + 0.8008 z^-1
```

```
Sample time: 0.1 seconds
```

```
Parameterization:
```

```
Polynomial orders: na=2 nb=2 nc=1 nk=1
```

```
Number of free coefficients: 5
```

```
Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.
```

```
Status:
```

```
Estimated using ARMAX on time domain data "data_1".
```

```
Fit to estimation data: 100% (prediction focus)
```

```
FPE: 3.905e-32, MSE: 3.857e-32
```

```
zeros_and_poles_sys
```

```
zeros_sys =
```

```
1
```

```
poles_sys =
```

```
0.8600 + 0.4005i  
0.8600 - 0.4005i
```

```
zeros_and_poles_arx_1
```

```
zeros_arx_1 =
```

```
1.0000
```

```
poles_arx_1 =
```

```
0.8600 + 0.4005i  
0.8600 - 0.4005i
```

```
zeros_and_poles_armax_1
```

```
zeros_armax_1 =
```

```
1
```

```
poles_armax_1 =
```

```
0.8600 + 0.4005i  
0.8600 - 0.4005i
```

poles and zeros

## Errors

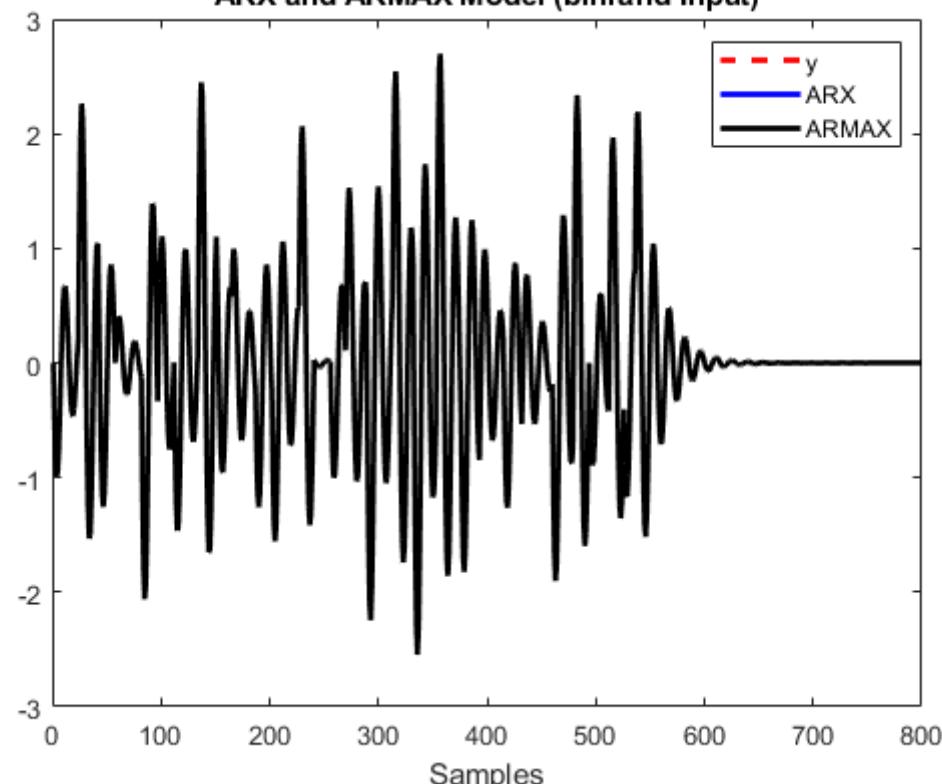
```
error_arx_1 =
```

```
1.0813e-16
```

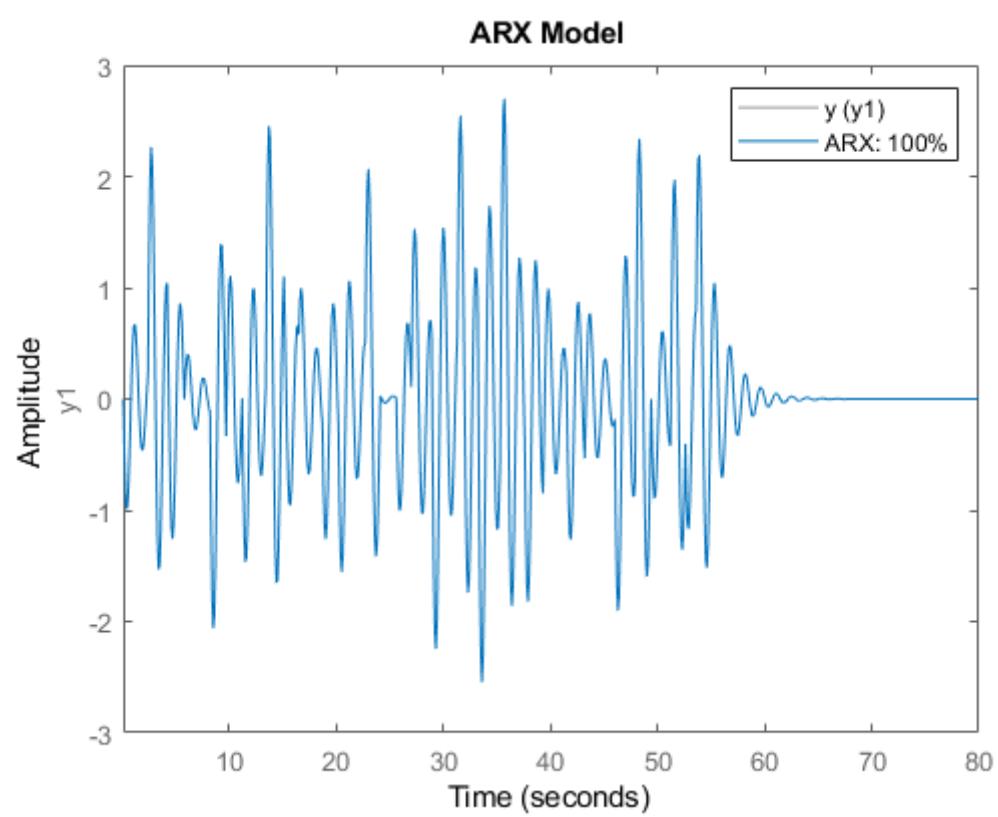
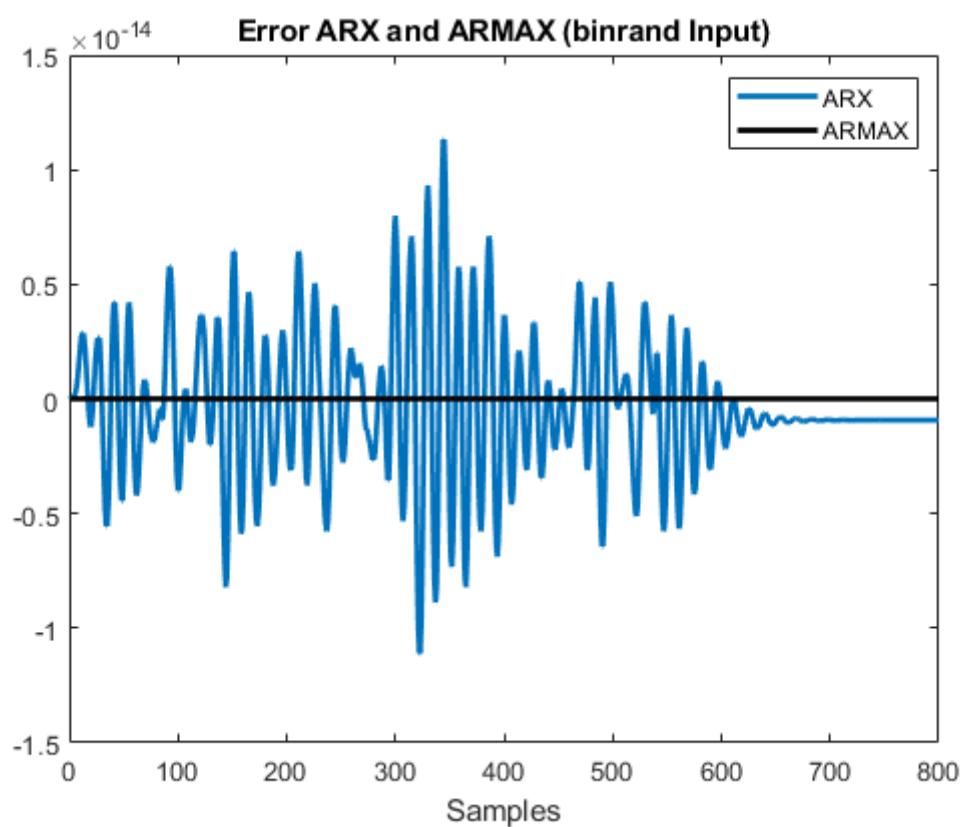
```
error_armax_1 =
```

```
0
```

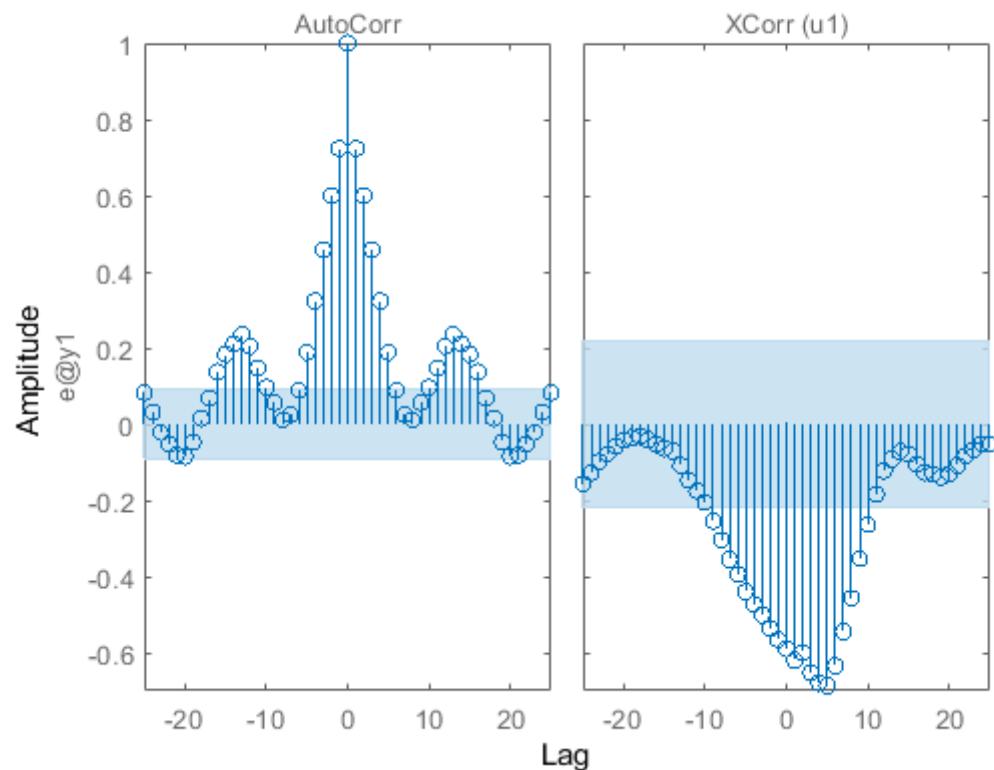
**ARX and ARMAX Model (binrand Input)**



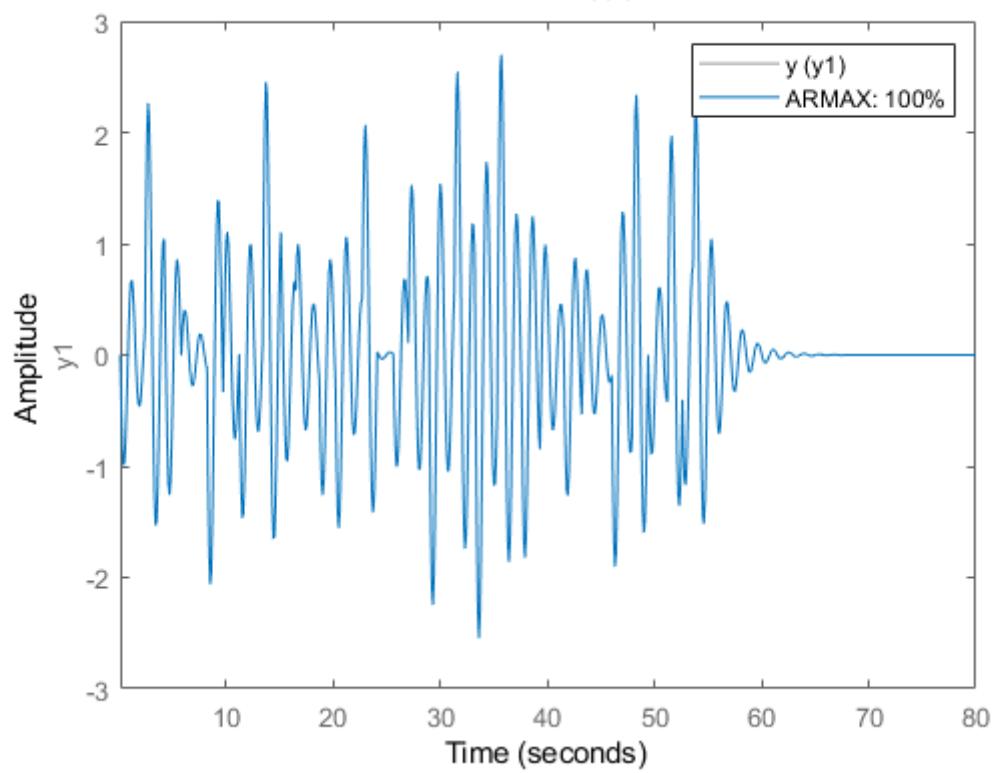
Samples



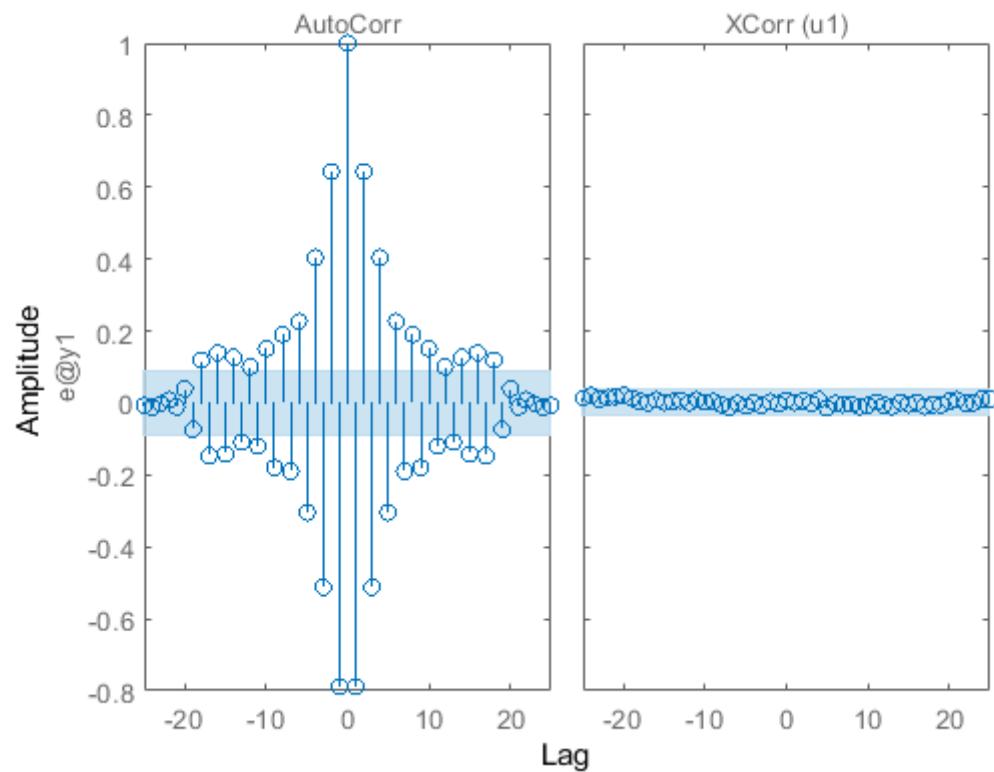
### ARX Model



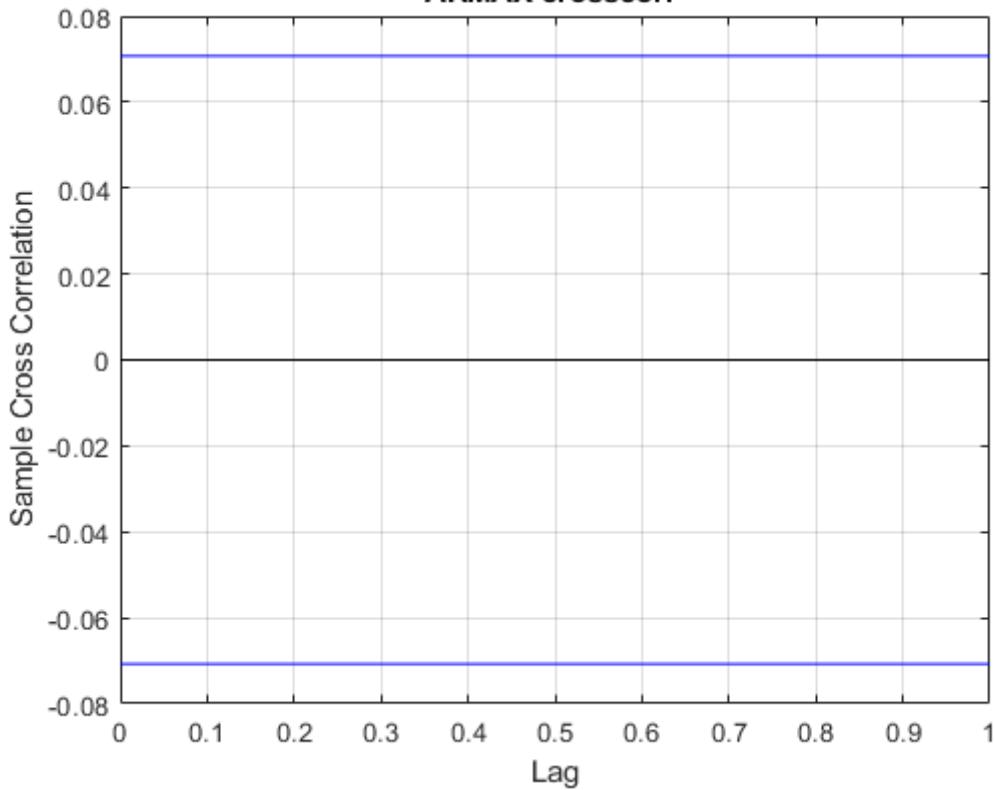
### ARMAX Model

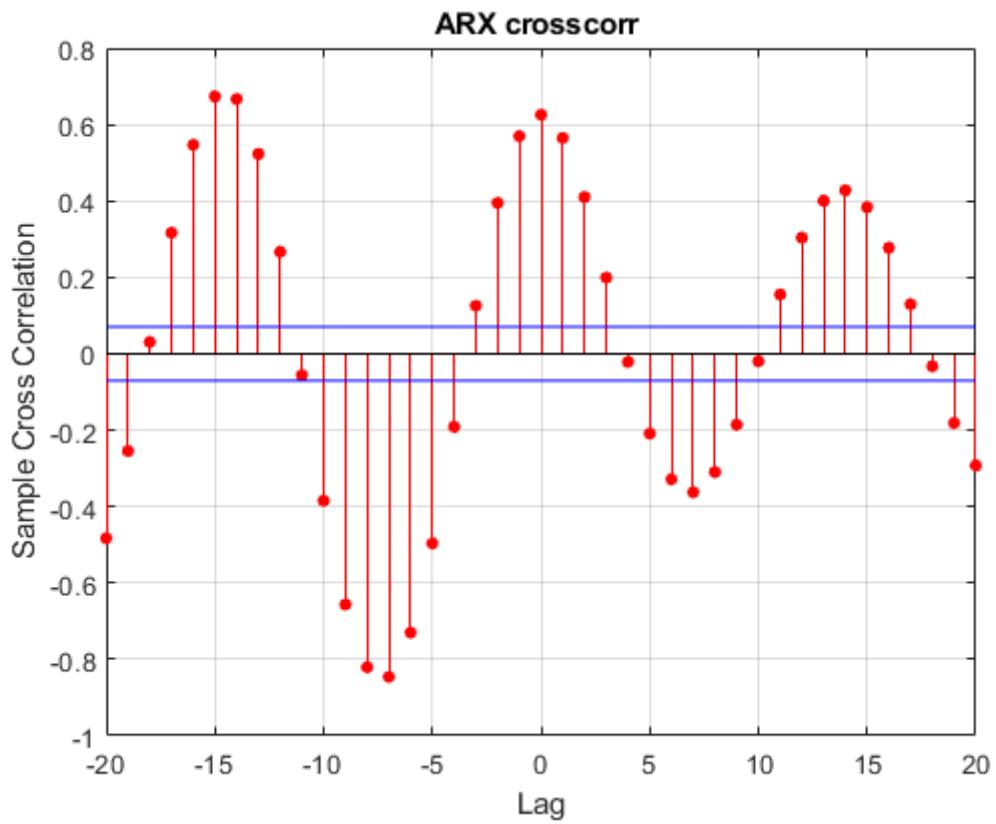


### ARMAX Model



### ARMAX crosscorr

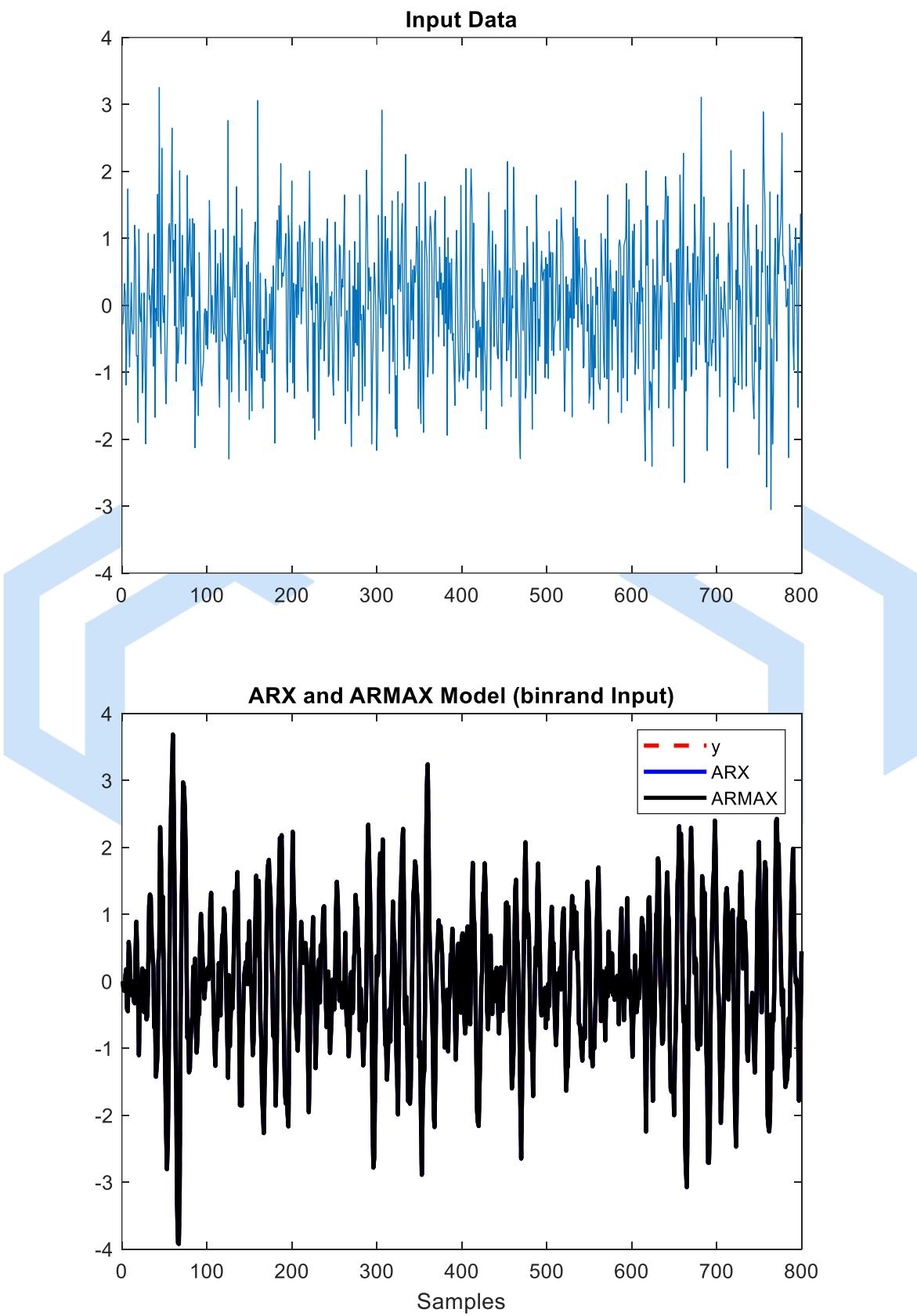


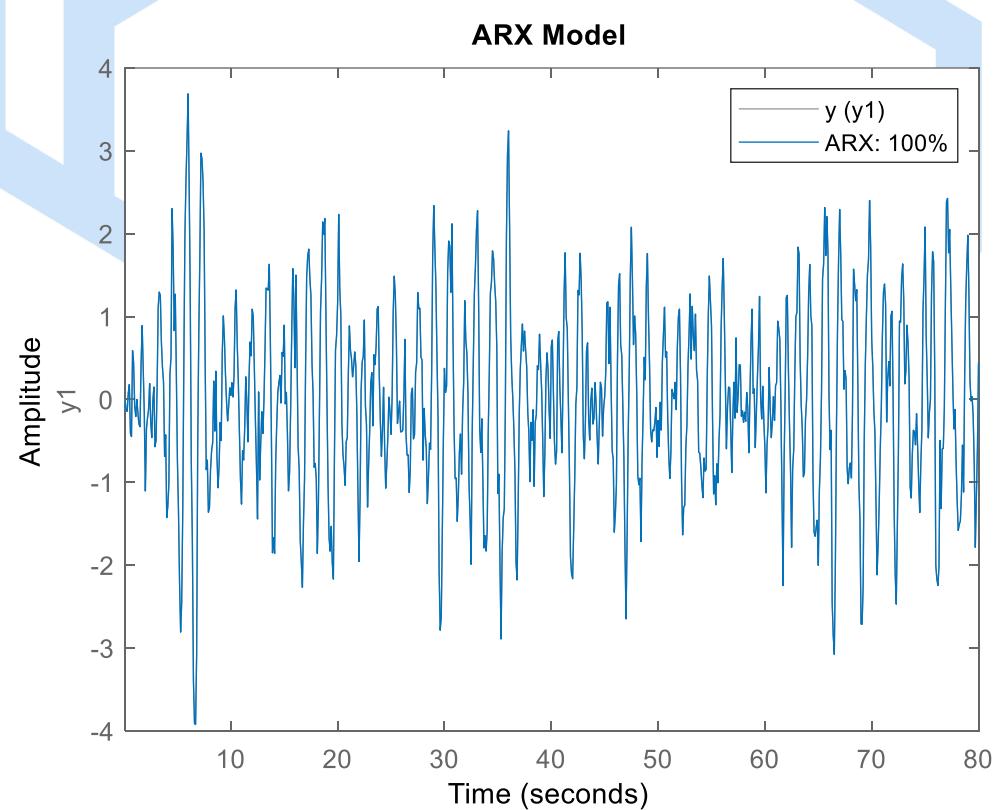
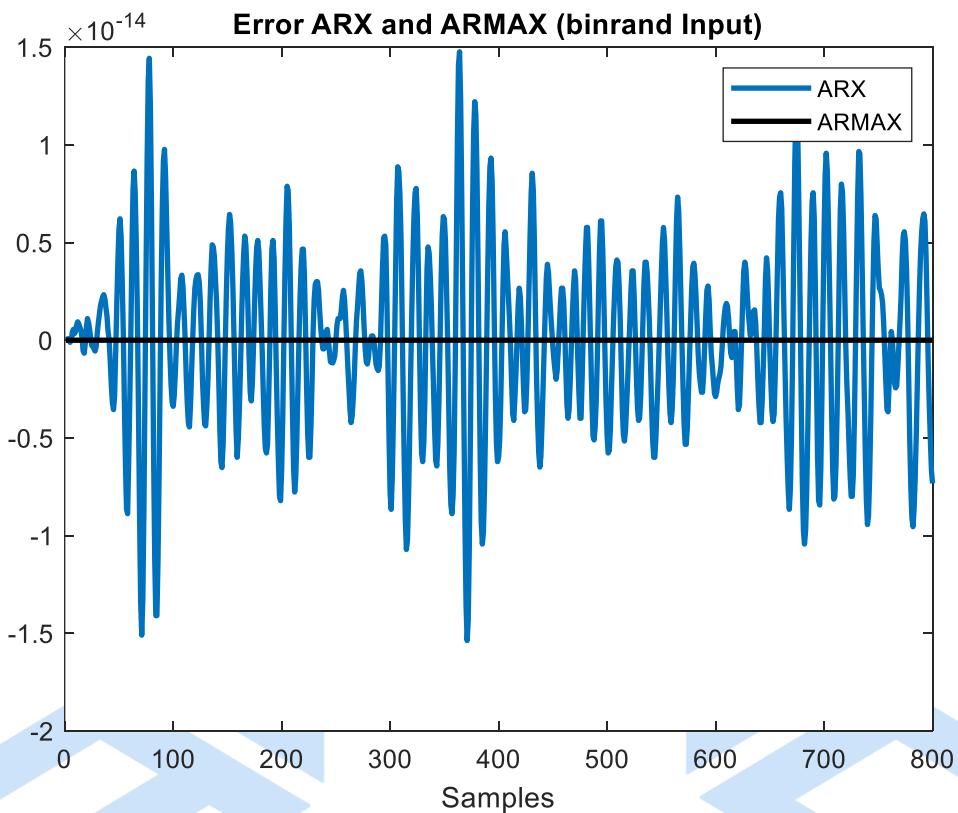


در شکل های بالا نمودار های خروجی مدل اصلی و مدل شناسایی شده نشان داده شده است که مدل شناسایی شده بر مدل اصلی فیت شده است

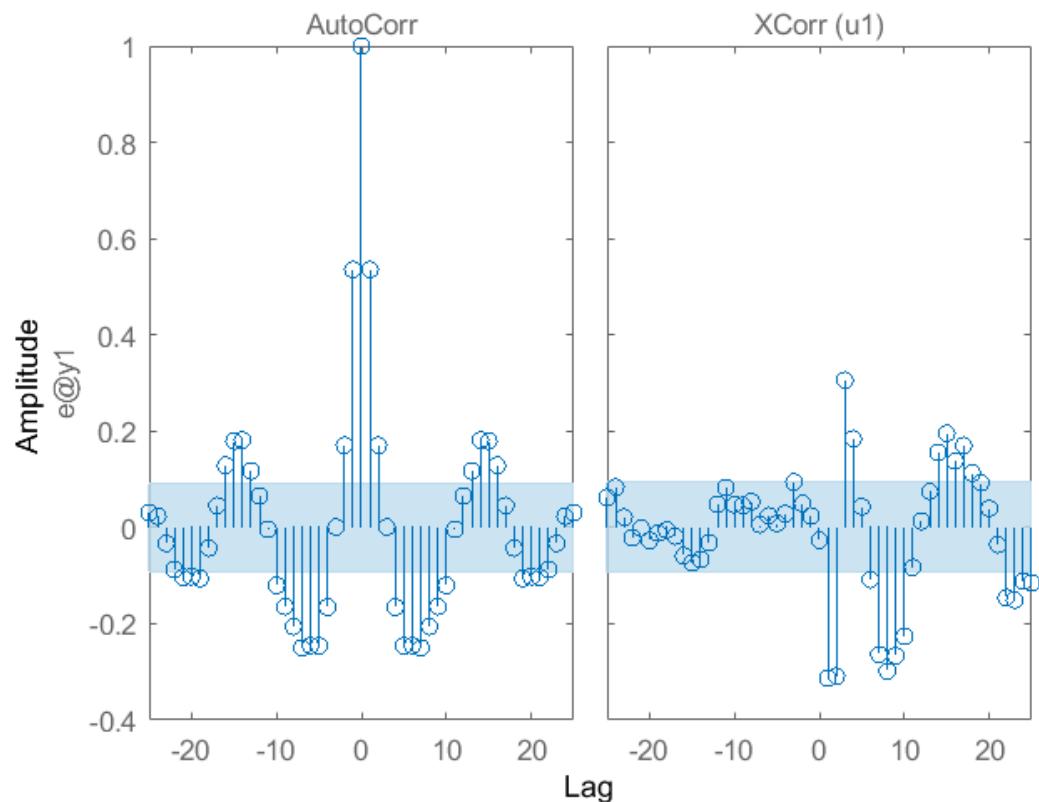
در شکل های بالا نمودار های اتوکورولیشن خطأ و کراس کورولیشن خطأ و ورودی و خطأ و خروجی رسم شده است. مقادیر این داده ها درون بازه مشخص شده قرار نگرفته است و این نشان دهنده این است که هنوز اطلاعاتی درون داده ها وجود دارد که ساختار مدل پیشنهاد داده شده در صورت سوال و همچنین داده های ورودی نتوانسته این اطلاعات را استخراج کند

برای ورودی سوم

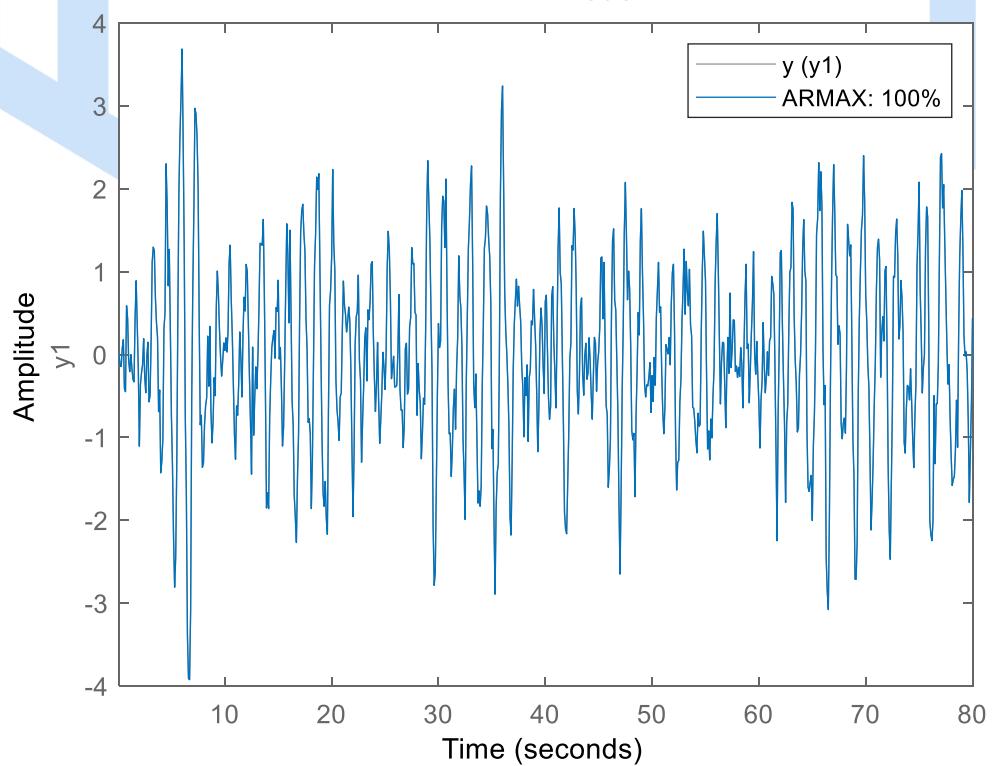




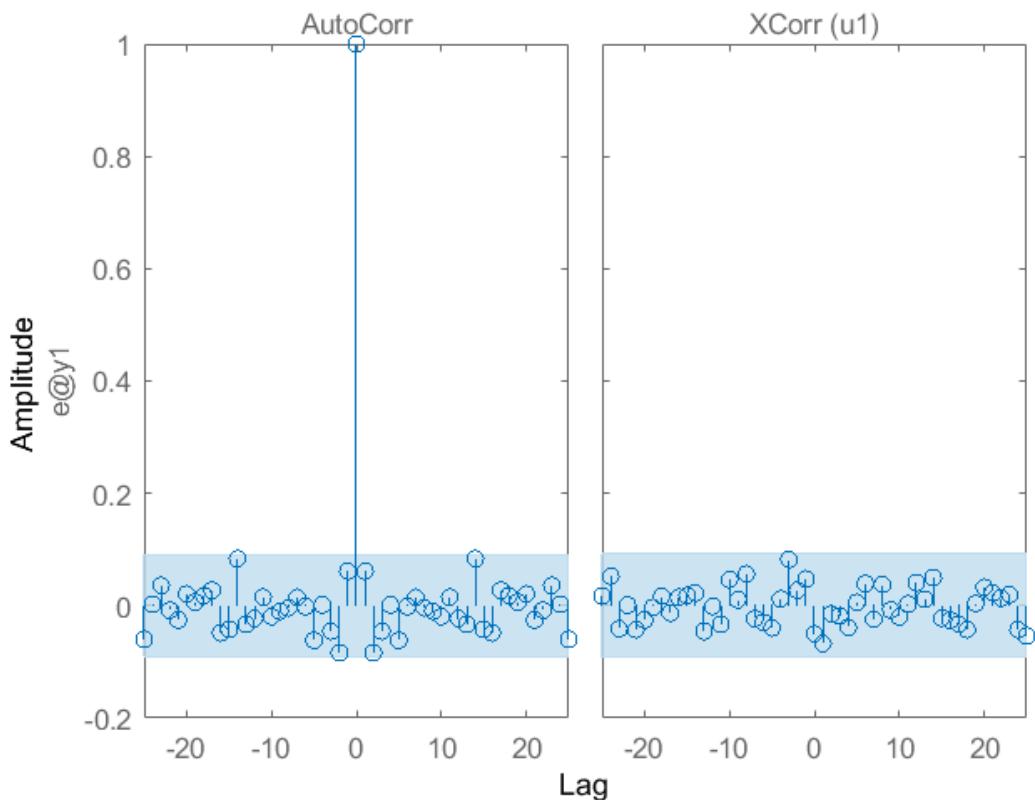
### ARX Model



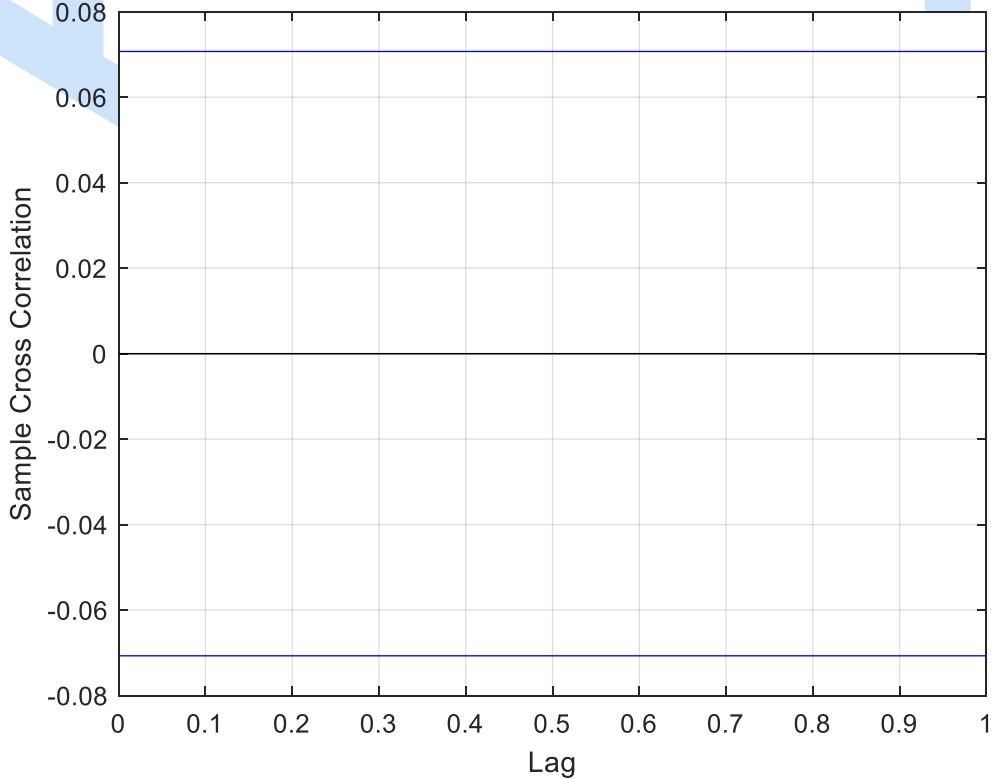
### ARMAX Model



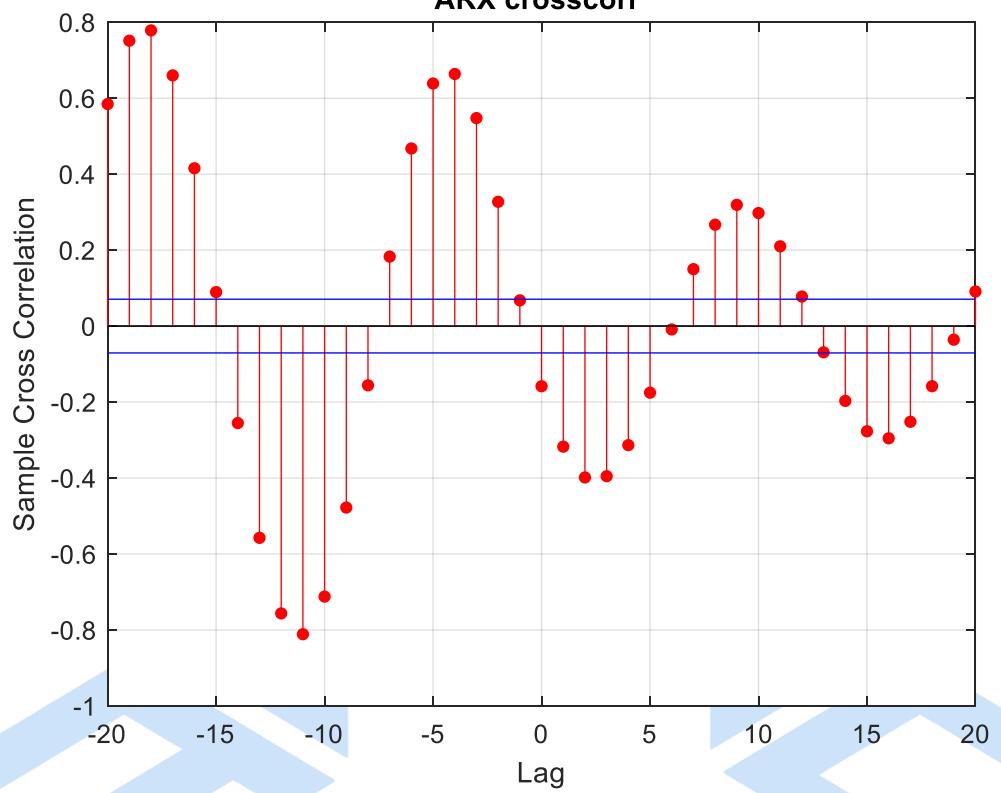
### ARMAX Model



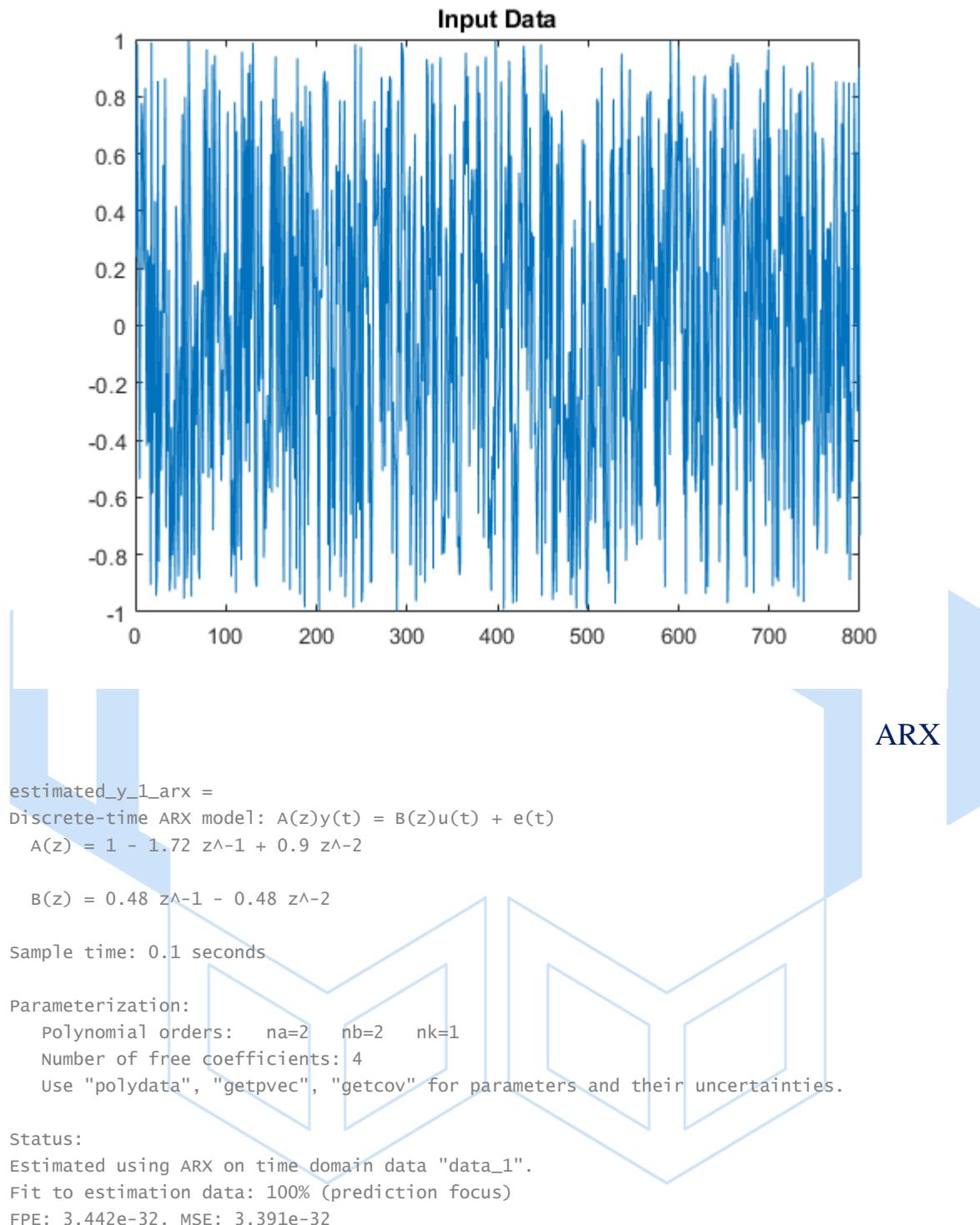
### ARMAX crosscorr



**ARX crosscorr**



## برای ورودی چهارم



## ARMAX

```
estimated_y_1_armax =  

Discrete-time ARMAX model:  $A(z)y(t) = B(z)u(t) + C(z)e(t)$   

 $A(z) = 1 - 1.72 z^{-1} + 0.9 z^{-2}$   

 $B(z) = 0.48 z^{-1} - 0.48 z^{-2}$ 
```

```
C(z) = 1 + 0.2826 z^-1
```

```
Sample time: 0.1 seconds
```

```
Parameterization:
```

```
Polynomial orders: na=2 nb=2 nc=1 nk=1
```

```
Number of free coefficients: 5
```

```
Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.
```

```
Status:
```

```
Estimated using ARMAX on time domain data "data_1".
```

```
Fit to estimation data: 100% (prediction focus)
```

```
FPE: 1.107e-32, MSE: 1.093e-32
```

```
zeros_and_poles_sys
```

```
zeros_sys =
```

```
1
```

```
poles_sys =
```

```
0.8600 + 0.4005i  
0.8600 - 0.4005i
```

```
zeros_and_poles_arx_1
```

```
zeros_arx_1 =
```

```
1.0000
```

```
poles_arx_1 =
```

```
0.8600 + 0.4005i  
0.8600 - 0.4005i
```

```
zeros_and_poles_armax_1
```

```
zeros_armax_1 =
```

```
1
```

```
poles_armax_1 =
```

```
0.8600 + 0.4005i  
0.8600 - 0.4005i
```

poles and zeros

Errors

```
error_arx_1 =
```

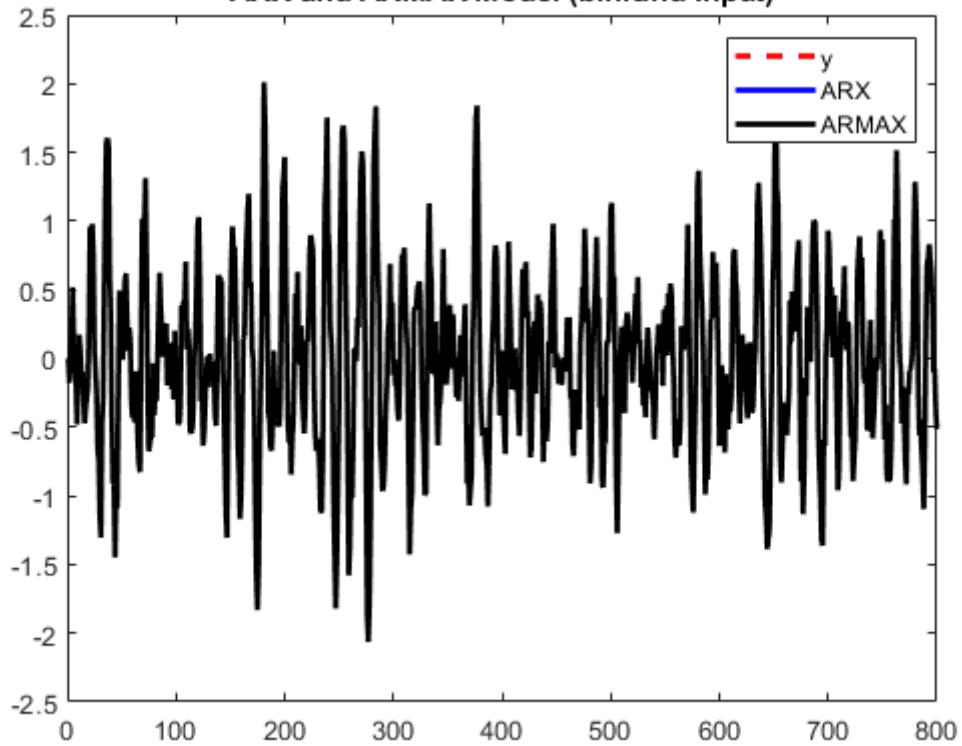
```
7.0702e-17
```

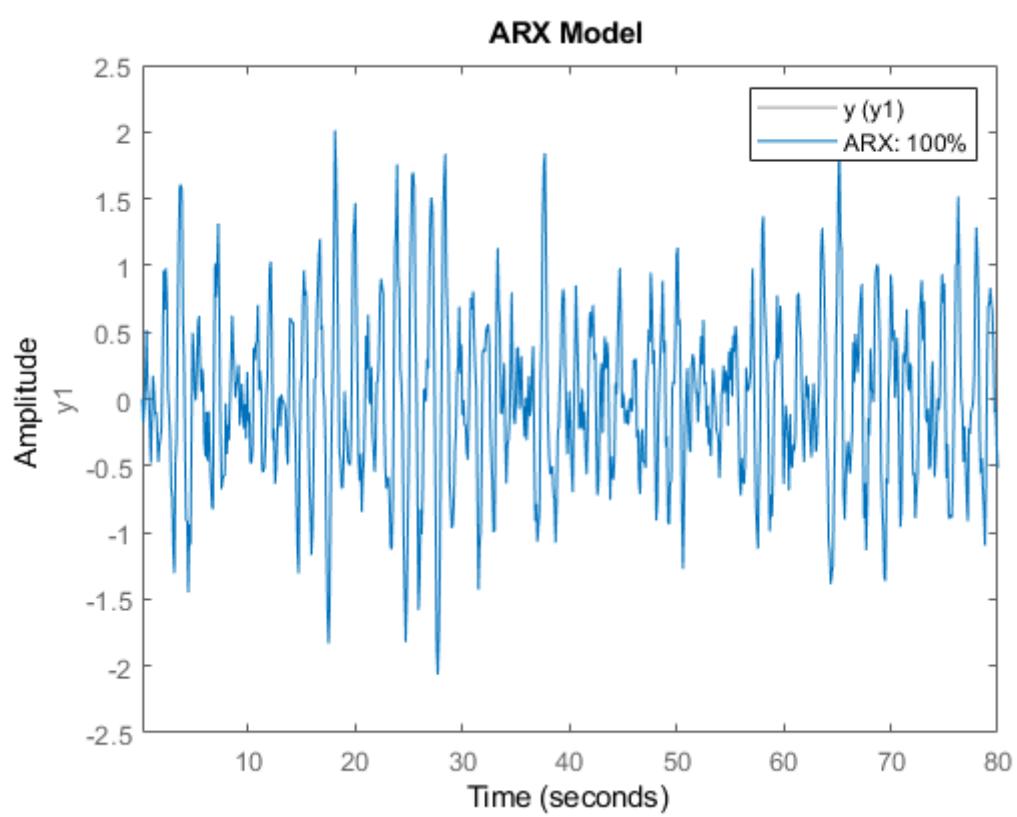
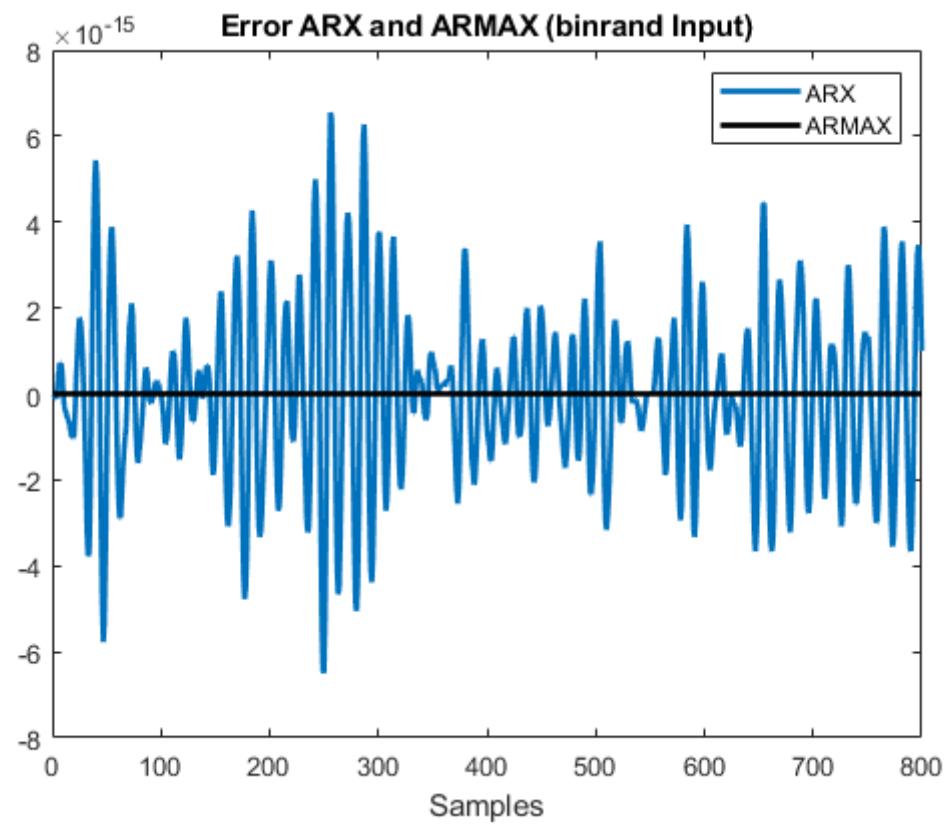
```
error_armax_1 =
```

```
0
```

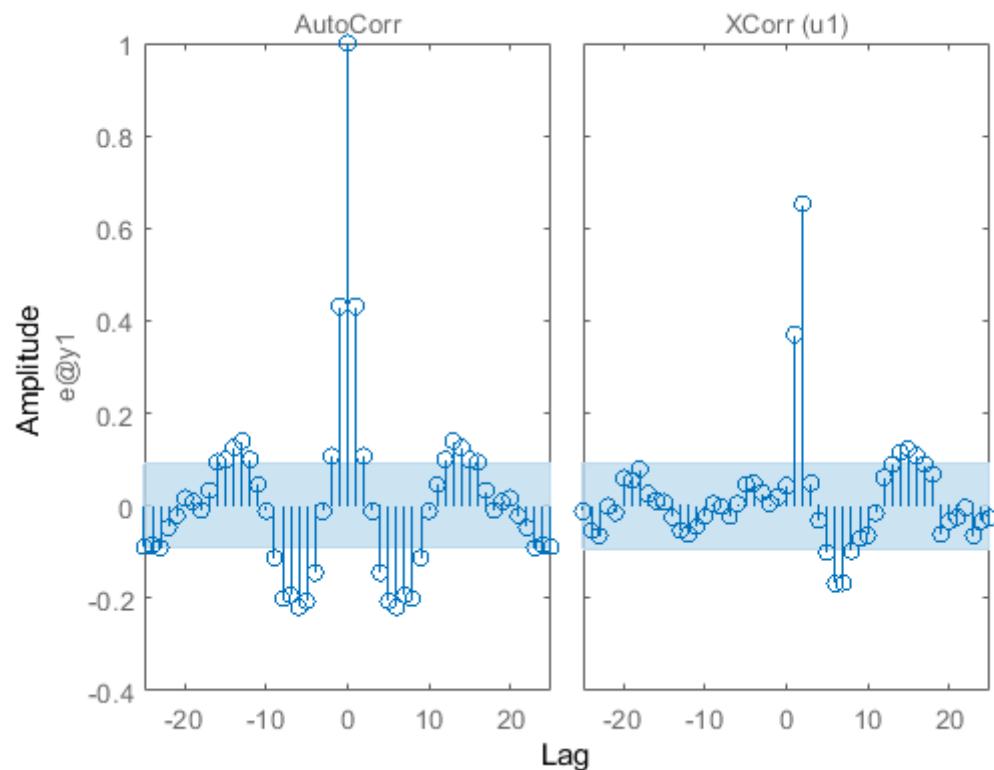
Plot

**ARX and ARMAX Model (binrand Input)**

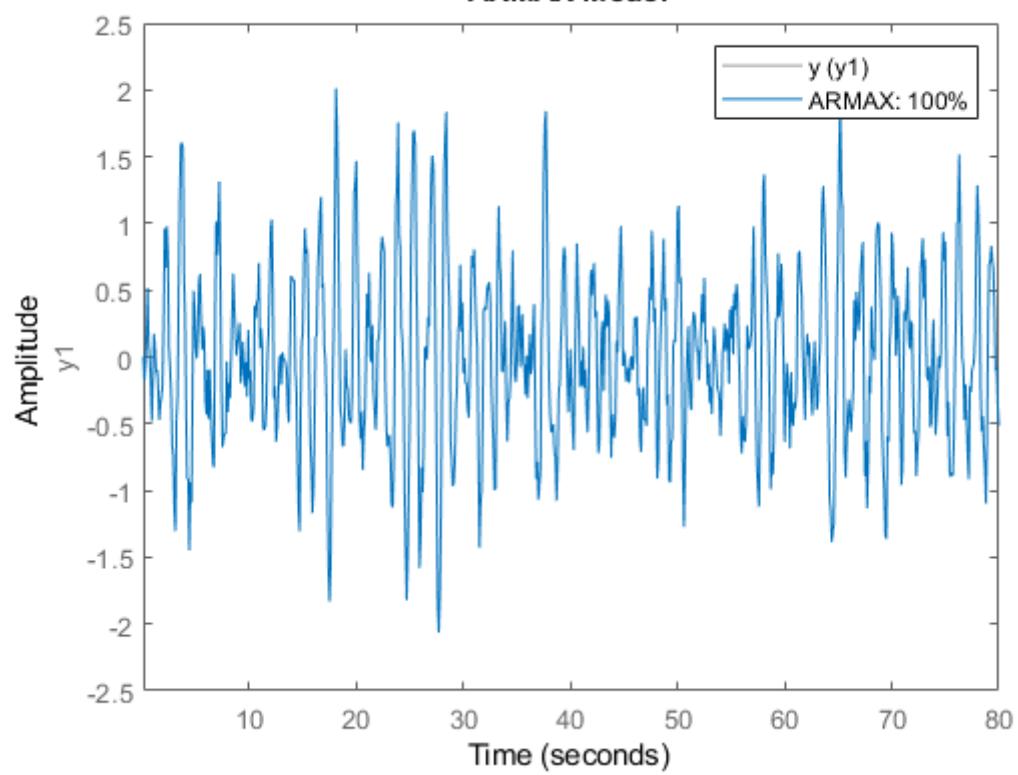




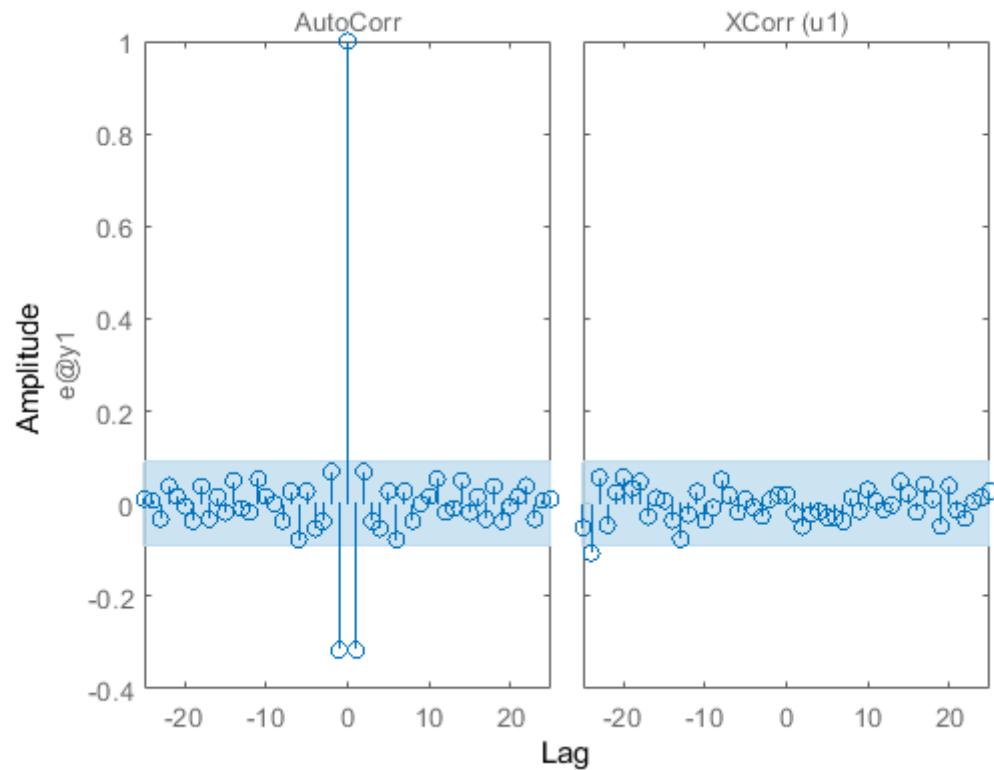
### ARX Model



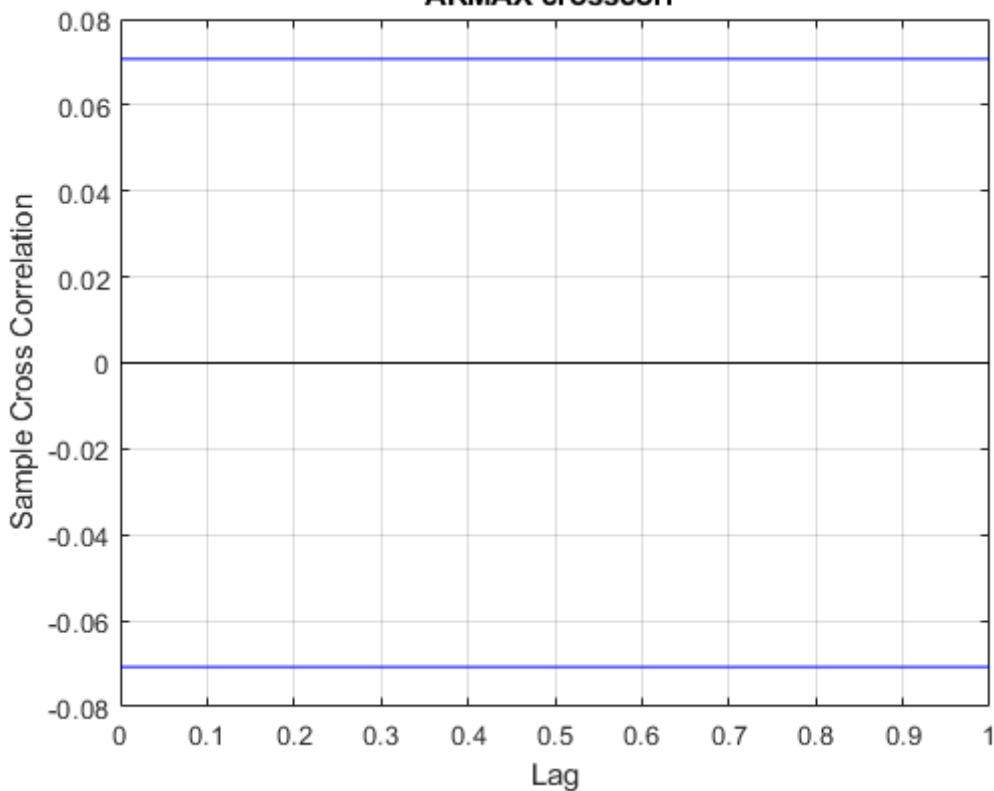
### ARMAX Model

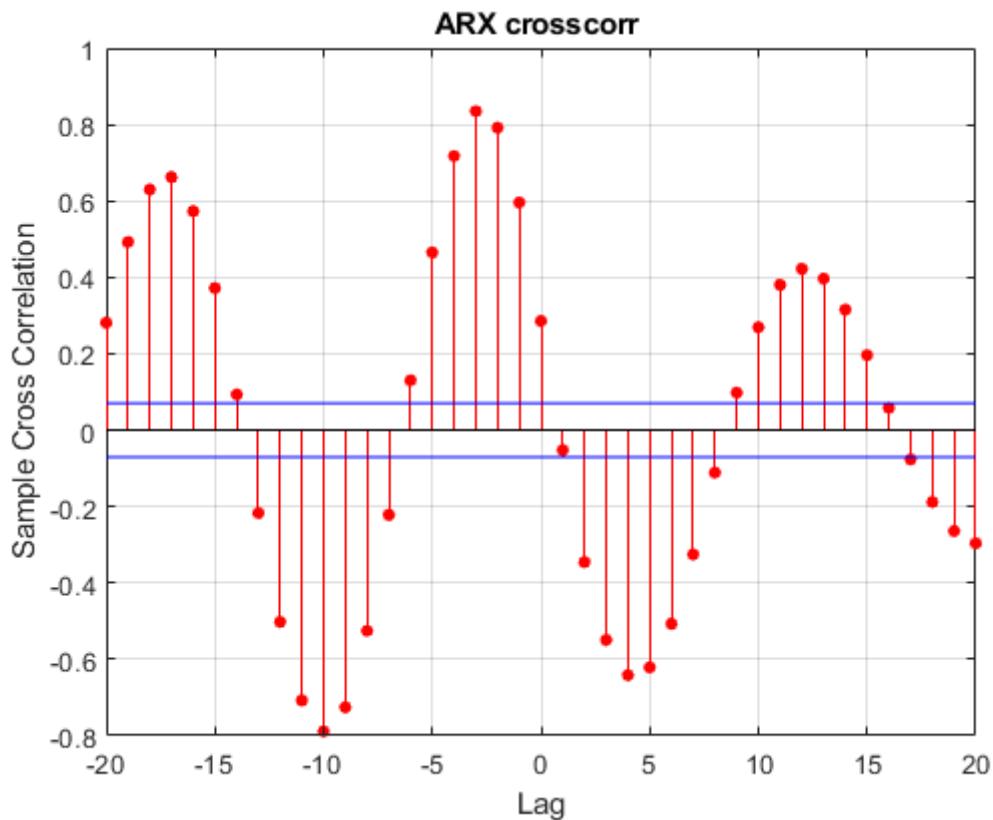


### ARMAX Model



### ARMAX crosscorr





### سوال دوم

در بخش قبلی به طراحی مدل ARX بدون استفاده از جعبه ابزار متلب پرداخته بوده ایم در اینجا تنها به پیاده سازی روش گرادیان نزولی میپردازیم.

در ابتدا تابع هزینه را برابر با مقدار زیر در نظر میگیریم.

$$J = \frac{1}{2} \epsilon^2$$

برای کاهش تابع هزینه باید در خلاف جهت گرادیان حرکت کنیم پس گرادیان آن را نسبت به هر یک از پارامتر ها محاسبه میکنیم به طور مثال

$$b_i(k) = b_i(k-1) - \rho \left( \frac{\partial J}{\partial b_i} \right)$$

در این مسئله باید مقدار  $\rho$  عدد مثبت کوچکی باشد تا شرط همگرایی برقرار باشد اگر این مقدار بزرگ باشد الگوریتم همگرا نمیشود مقادیر اولیه تنا را بین منفی 2 و 2 لحاظ کردیم. نرخ یادگیری را کوچک و 0.0001 لحاظ کردیم تا همگرایی اتفاق افتد

Recall that we have  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ , convex and differentiable, want to solve

$$\min_{x \in \mathbb{R}^n} f(x),$$

i.e., find  $x^*$  such that  $f(x^*) = \min f(x)$

**Gradient descent:** choose initial  $x^{(0)} \in \mathbb{R}^n$ , repeat:

$$x^{(k)} = x^{(k-1)} - t_k \cdot \nabla f(x^{(k-1)}), \quad k = 1, 2, 3, \dots$$

| Stop at some point

حال به پیاده سازی این الگوریتم میپردازیم

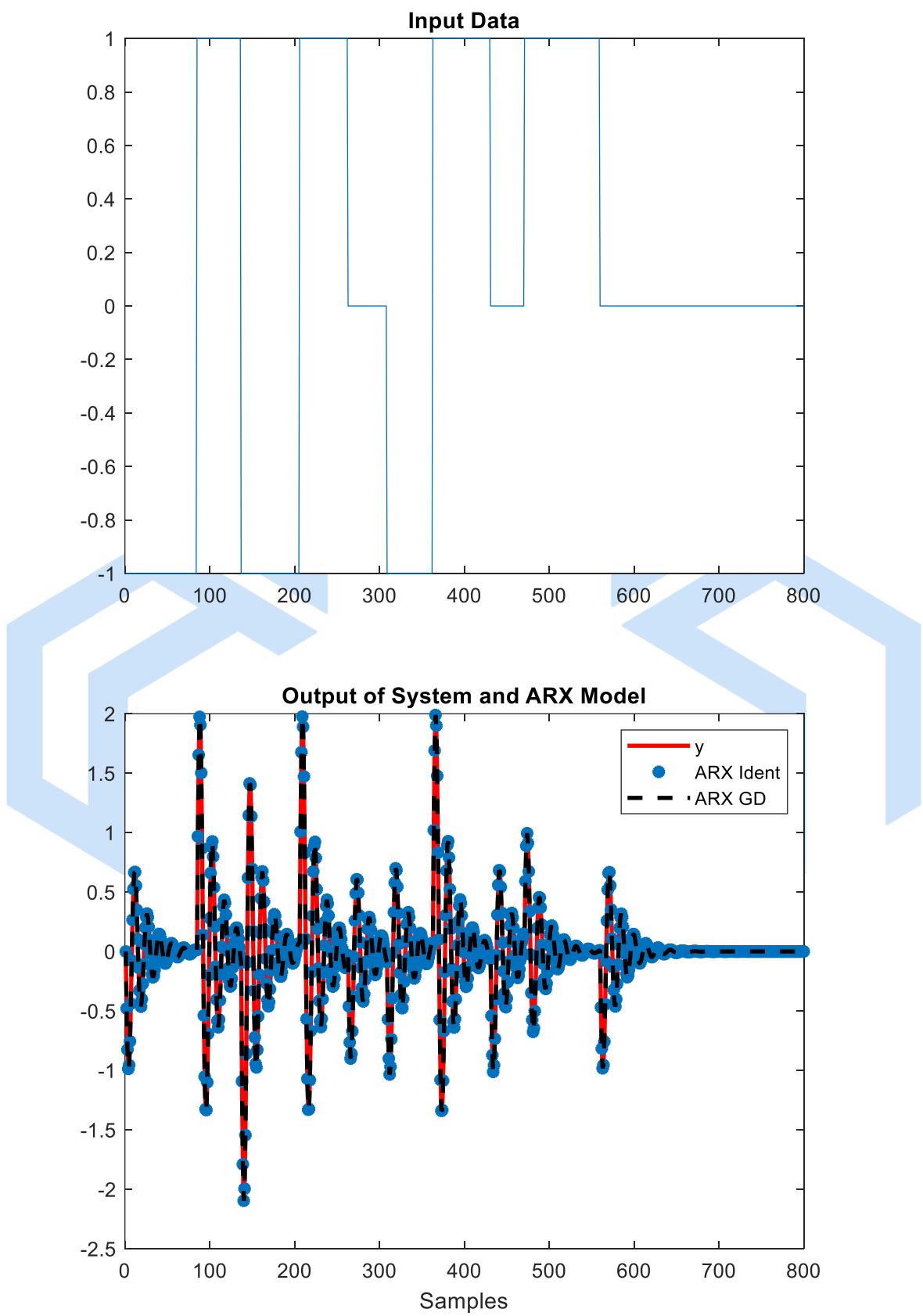
#### 8.8 ARX Gradient Descent Algorithm

```
X = GenData(na,nb,nk,U,y);
theta=randi([-2 2],1,na+nb); % initial theta : random theta beetwin [-2 2]
Gradient=ones(1,nb+na);
Step= 0.0001;
while norm(Gradient)>1e-4
    Gradient = -2.* (y'-(theta*X'))*X';
    theta=theta-Step*Gradient;
end
den_arx_Gradient=[1 theta(1:nb)];
num_arx_Gradient=[zeros(1,nk) theta(nb+1:nb+na)];
sys_arx_Gradient=filt(num_arx_Gradient,den_arx_Gradient,Ts);
y_arx_Gradient=lsim(sys_arx_Gradient,U);
```

نتایج خروجی به ازای ورودی های مختلف به صورت زیر است. در همه ورودی ها نتایج بدست امده از

روش گرادیان نزولی و روش جعبه ابزار متلب بسیار نزدیک به یک دیگر میباشند

به ازای ورودی اول



```
zeros_arx_ident =
```

```
1.0000
```

```
zeros_arx_Gradient =
```

```
1.0000
```

```
poles_sys =
```

```
0.8600 + 0.4005i
```

```
0.8600 - 0.4005i
```

```
poles_arx_ident =
```

```
0.8600 + 0.4005i
```

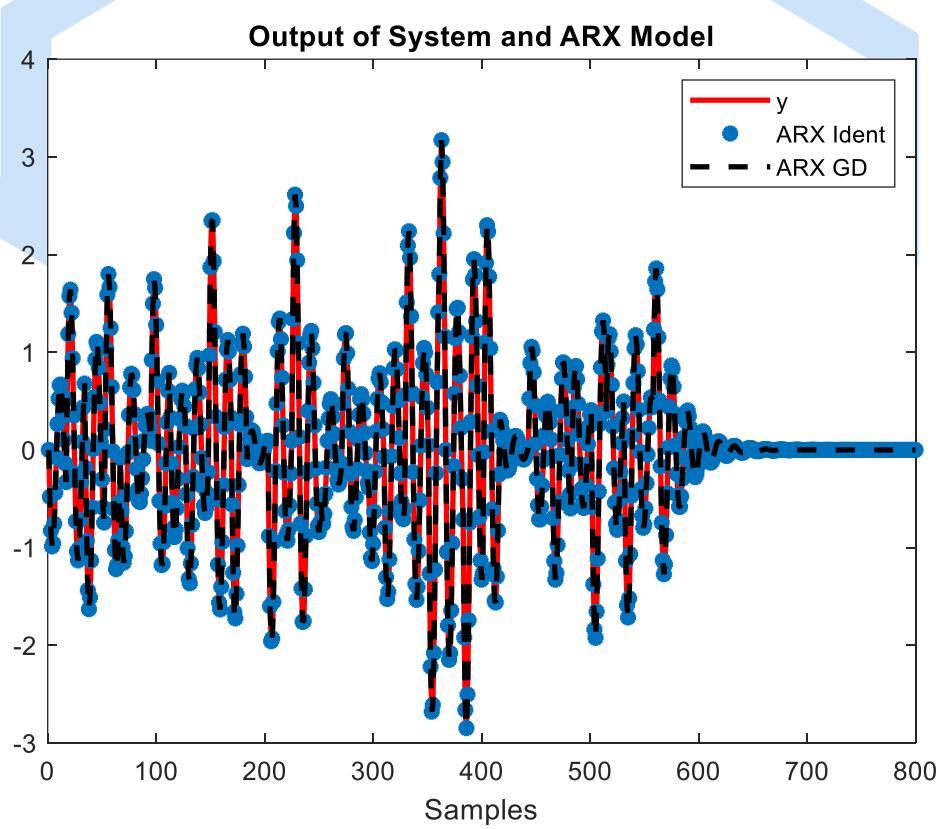
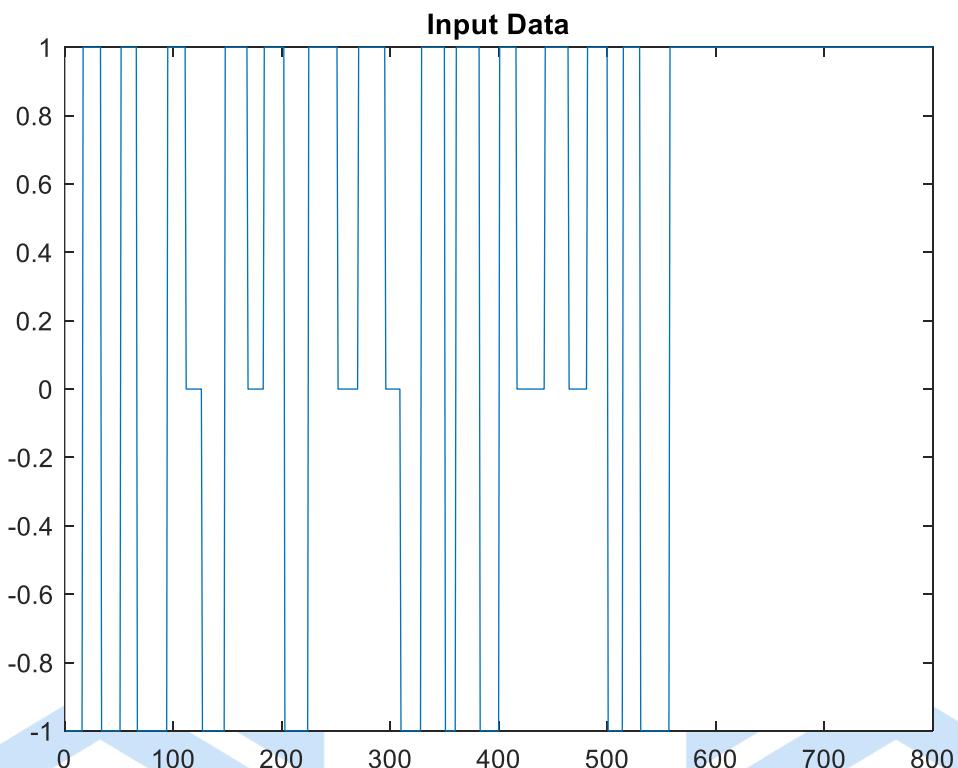
```
0.8600 - 0.4005i
```

```
poles_arx_Gradient =
```

```
0.8600 + 0.4005i
```

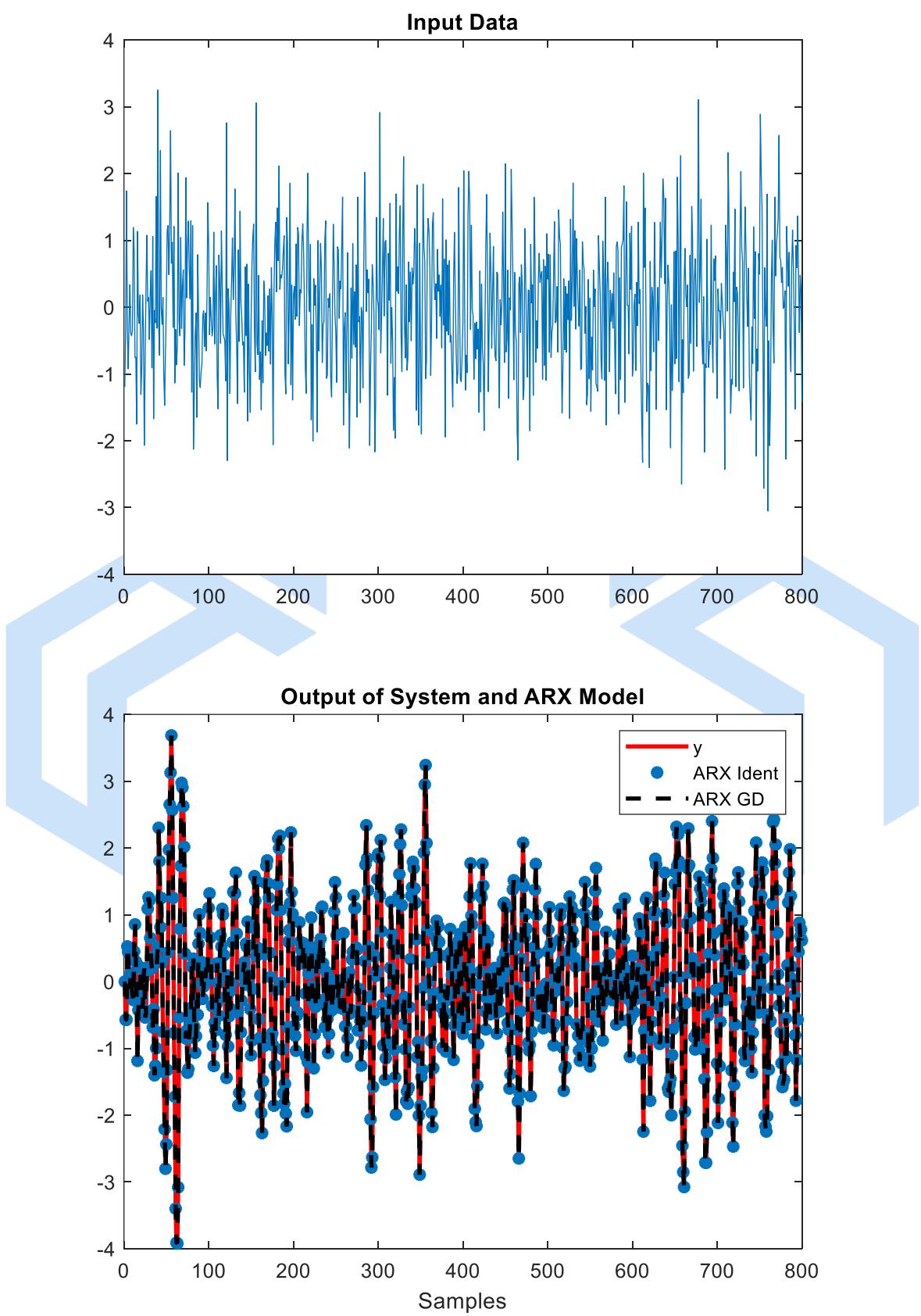
```
0.8600 - 0.4005i
```

به ازای ورودی دوم



```
zeros_arx_ident =  
1.0000  
  
zeros_arx_Gradient =  
1.0000  
  
poles_sys =  
0.8600 + 0.4005i  
0.8600 - 0.4005i  
  
poles_arx_ident =  
0.8600 + 0.4005i  
0.8600 - 0.4005i  
  
poles_arx_Gradient =  
0.8600 + 0.4005i  
0.8600 - 0.4005i
```

به ازای ورودی سوم



```
zeros_arx_ident =
```

```
1.0000
```

```
zeros_arx_Gradient =
```

```
1.0000
```

```
poles_sys =
```

```
0.8600 + 0.4005i
```

```
0.8600 - 0.4005i
```

```
poles_arx_ident =
```

```
0.8600 + 0.4005i
```

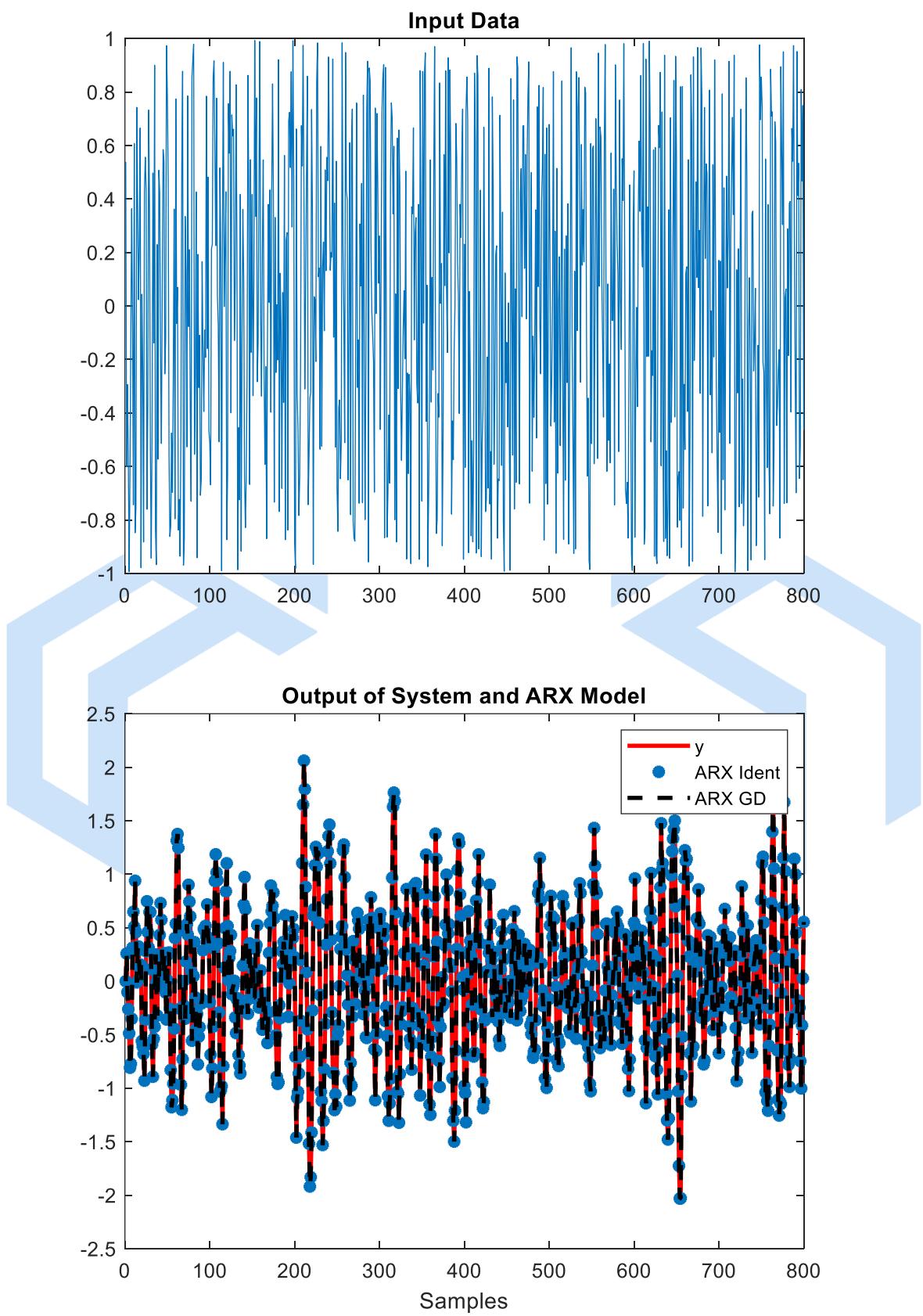
```
0.8600 - 0.4005i
```

```
poles_arx_Gradient =
```

```
0.8600 + 0.4005i
```

```
0.8600 - 0.4005i
```

به ازای ورودی چهارم



```

zeros_sys =
1

zeros_arx_ident =
1.0000

zeros_arx_Gradient =
1.0000

poles_sys =
0.8600 + 0.4005i
0.8600 - 0.4005i

poles_arx_ident =
0.8600 + 0.4005i
0.8600 - 0.4005i

poles_arx_Gradient =
0.8600 + 0.4005i
0.8600 - 0.4005i

```

### سوال سوم

برای حلقه بسته کردن سیستم از روابط میسون استفاده میکنیم بدین صورت که در ابتدا تابع تبدیل حلقه بسته از ورودی به خروجی را حساب میکنیم سپس با استفاده از قاعده جمع اثار تابع تبدیل نویز به خروجی را حساب میکنیم و سپس این دو تابع تبدیل را با یک دیگر جمع میکنیم

$$G_1(z) = \frac{G(z)}{1 + G(z)}$$

$$G_2(z) = \frac{1}{1 + G(z)}$$

$$y = \frac{G(z)}{1 + G(z)} u + \frac{1}{1 + G(z)} v$$

با توجه به اندازه سیگنال ورودی و مدل های انتخاب شده و ساختار های انها مقادیر زیر را با ازمون و خطاب برای واریانس نویز بدست اورده ایم

```
%% noise
sigma1 = 0.01;
sigma2 = 0.05;
sigma3 = 0.10;

sigma = sigma3;
v=random('norm',0,sigma,length(U),1);
```

به طور کلی با افزایش کوواریانس نویز دقت سیستم شناسایی شده کمتر میشود از بین مدل های مختلف مدل ARX با اضافه کردن نویز عملکرد بسیار ضعیفی از خود نشان میدهد و بعد از آن مدل های ARMAX,ARARX عملکرد متسطی داشته اند و مدل های OE,BJ عملکرد بهتری نسبت به بقیه مدل ها داشته اند.

برای نویز کم

به ازای نویز کم مقدار صفر سیستم را در همه حالت ها به خوبی تشخیص داده شده است اما مقادیر قطب ها اشتباہ تشخیص داده شده اند

در بین همه مدل ها OE بهتر توانسته قطب ها و صفر سیستم را تشخیص بدهد و ARX در شناسایی قطب ها ضعیف تر عملکرده است

درصد فیت شدن مدل ها نیز بالا 90 درصد میباشد و همچنین در بیشتر مدل ها خروجی نمودار های کورولیشن به نسبت مناسب میباشد و بیشتر دینامیک های سیستم را به خوبی توانسته اند شناسایی کنند

و مقادیر AIC نیز کم و مناسب میباشند

ARX

`zeros_sys_arx =`

0.9995

`poles_sys_arx =`

$0.5963 + 0.1393i$   
 $0.5963 - 0.1393i$

`zeros_sys_armax =`

0.9997

`poles_sys_armax =`

$0.6160 + 0.1822i$   
 $0.6160 - 0.1822i$

`zeros_sys_ararx =`

0.9998

`poles_sys_ararx =`

$0.6164 + 0.1825i$   
 $0.6164 - 0.1825i$

`zeros_sys_oe =`

0.9997

`poles_sys_oe =`

$0.6200 + 0.1883i$   
 $0.6200 - 0.1883i$

ARMAX

ARARX

OE

BJ

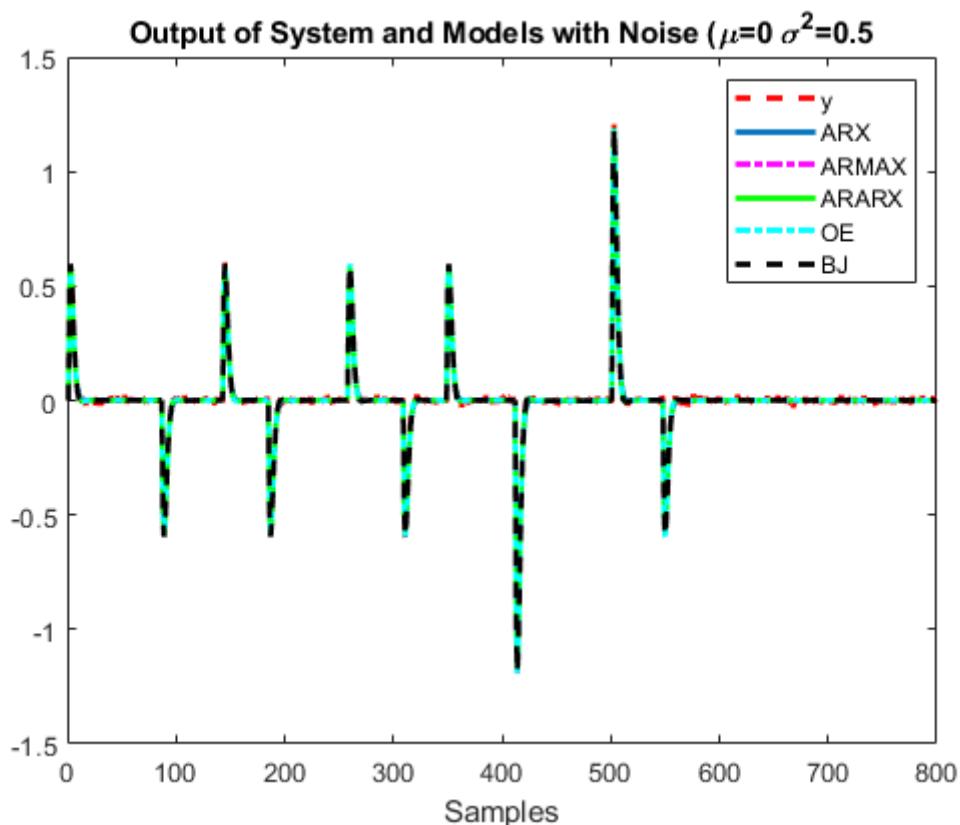
zeros\_sys\_bj =

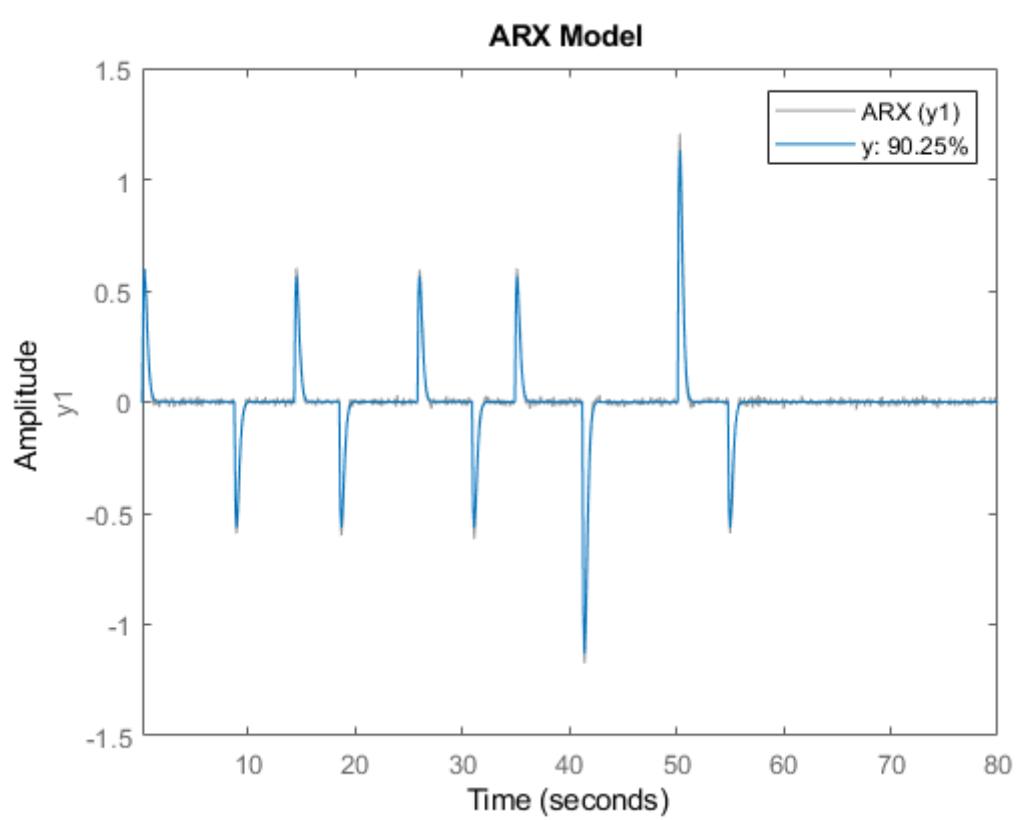
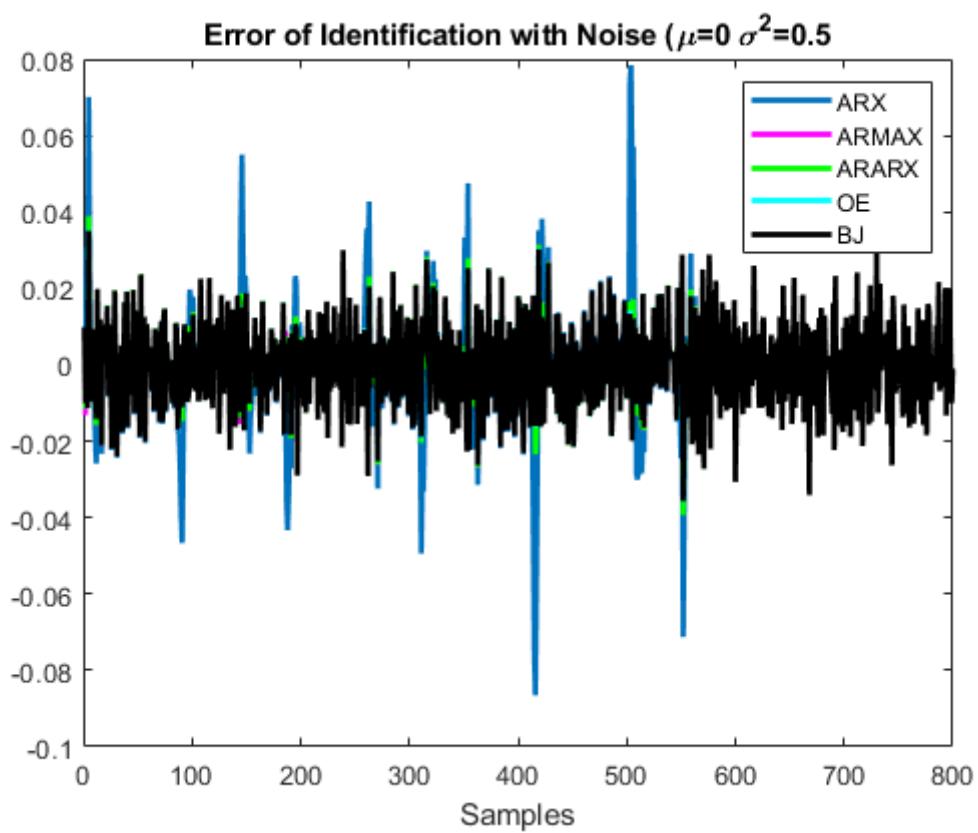
0.9997

poles\_sys\_bj =

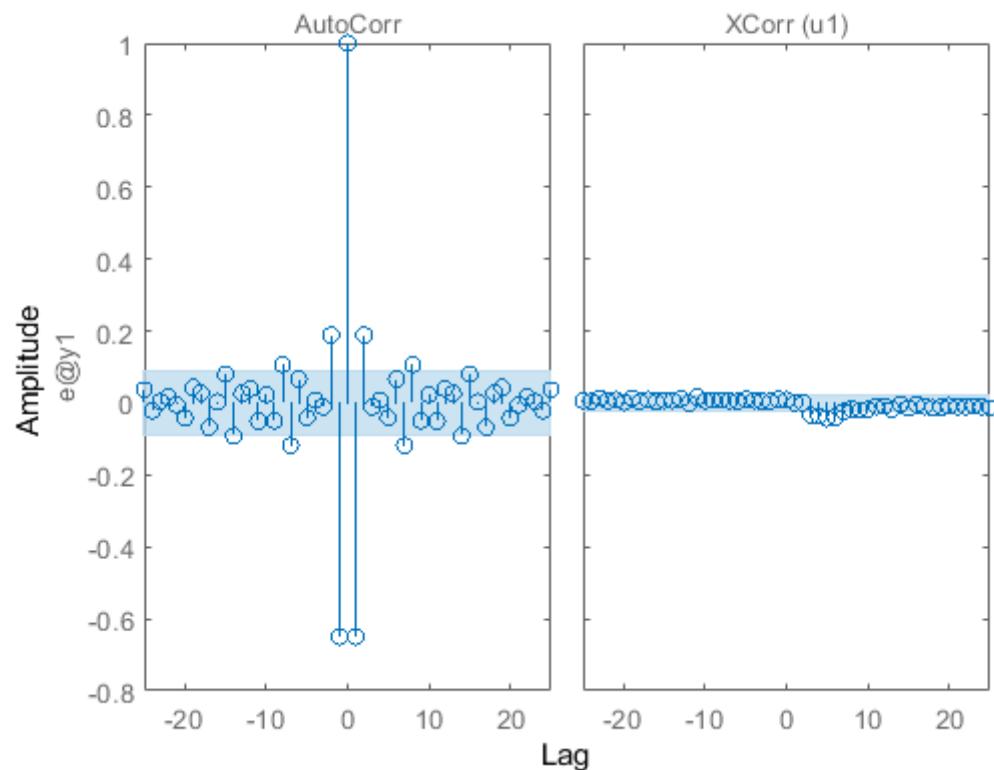
0.6199 + 0.1884i  
0.6199 - 0.1884i

Figure and result

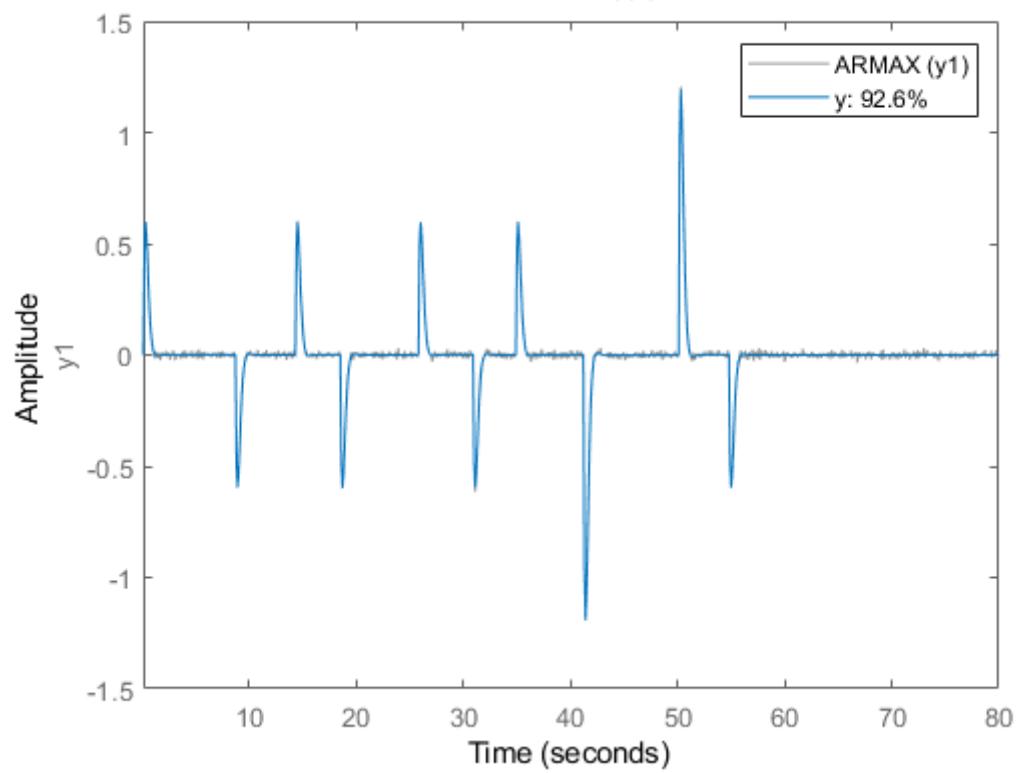




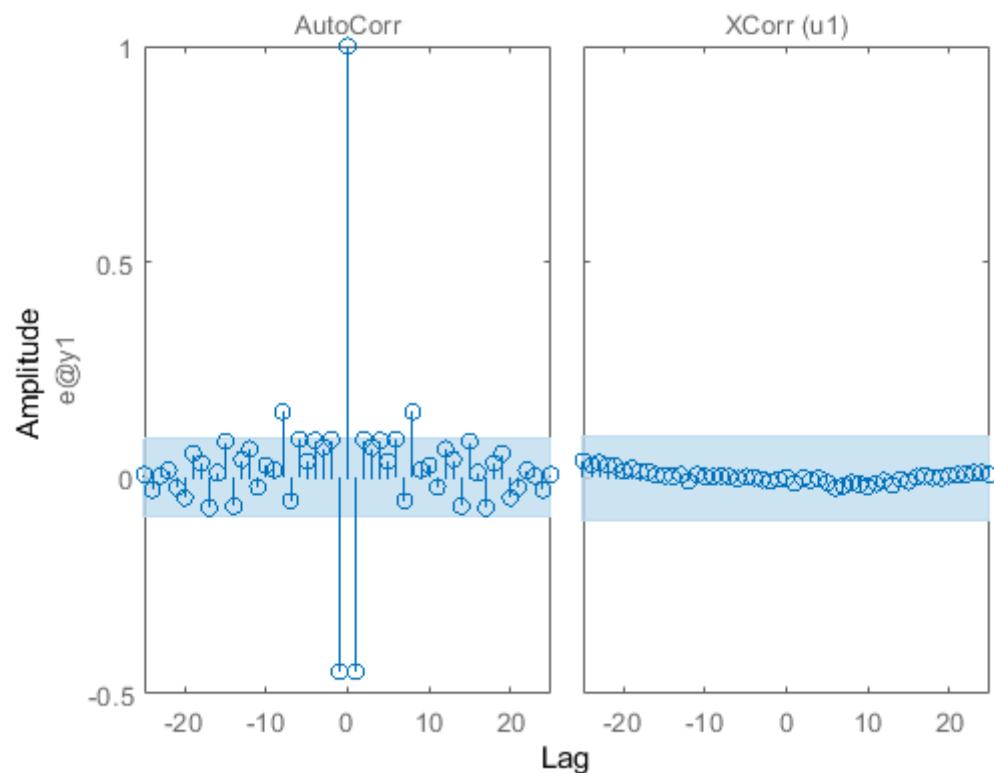
### ARX Model



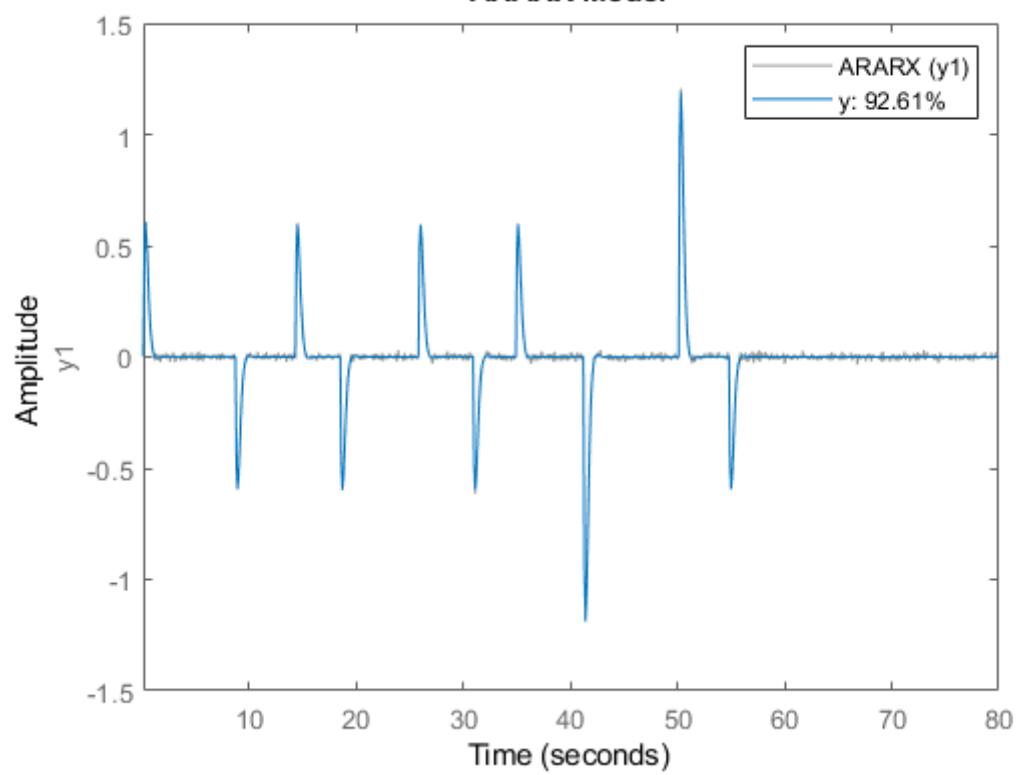
### ARMAX Model



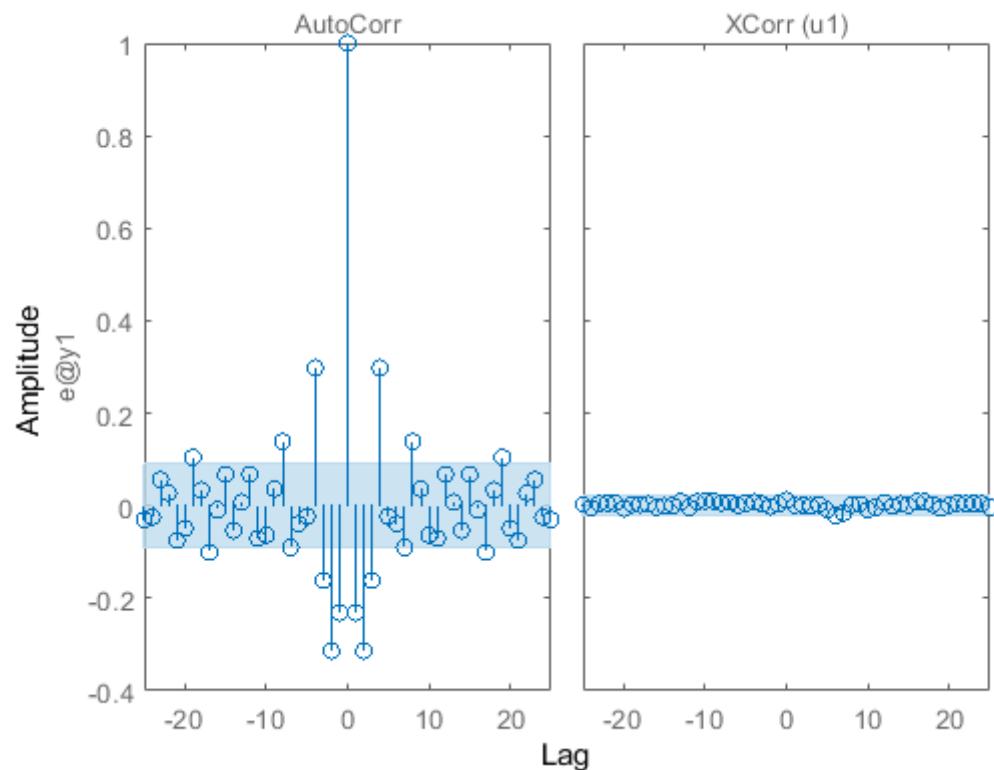
### ARMAX Model



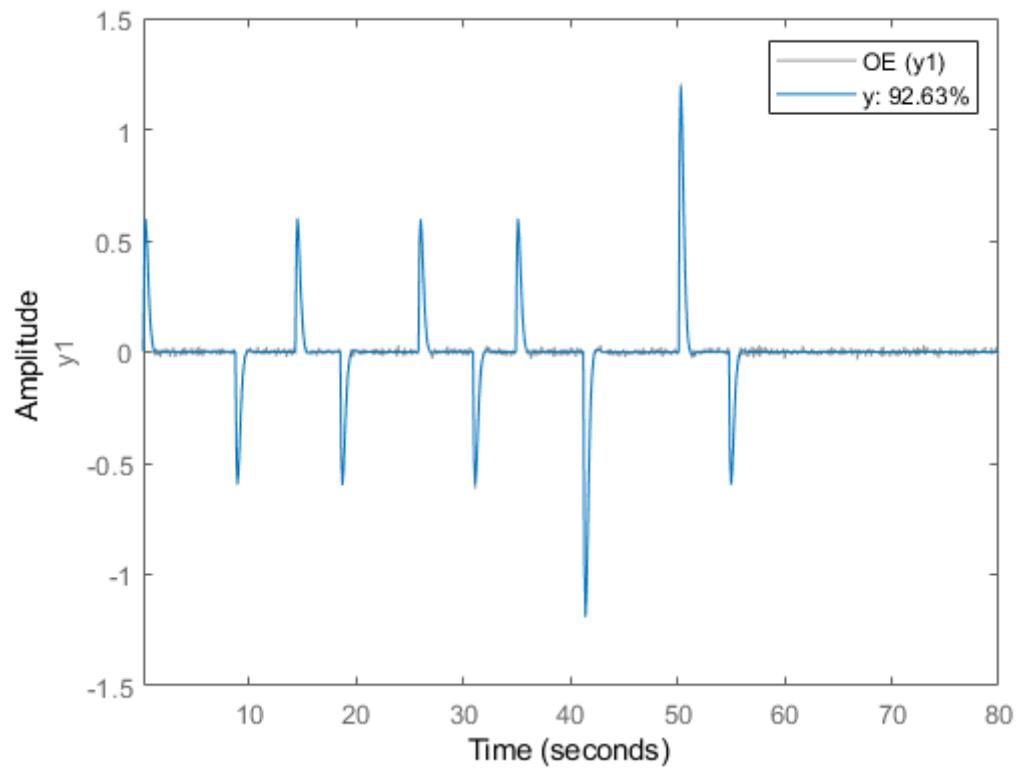
### ARARX Model



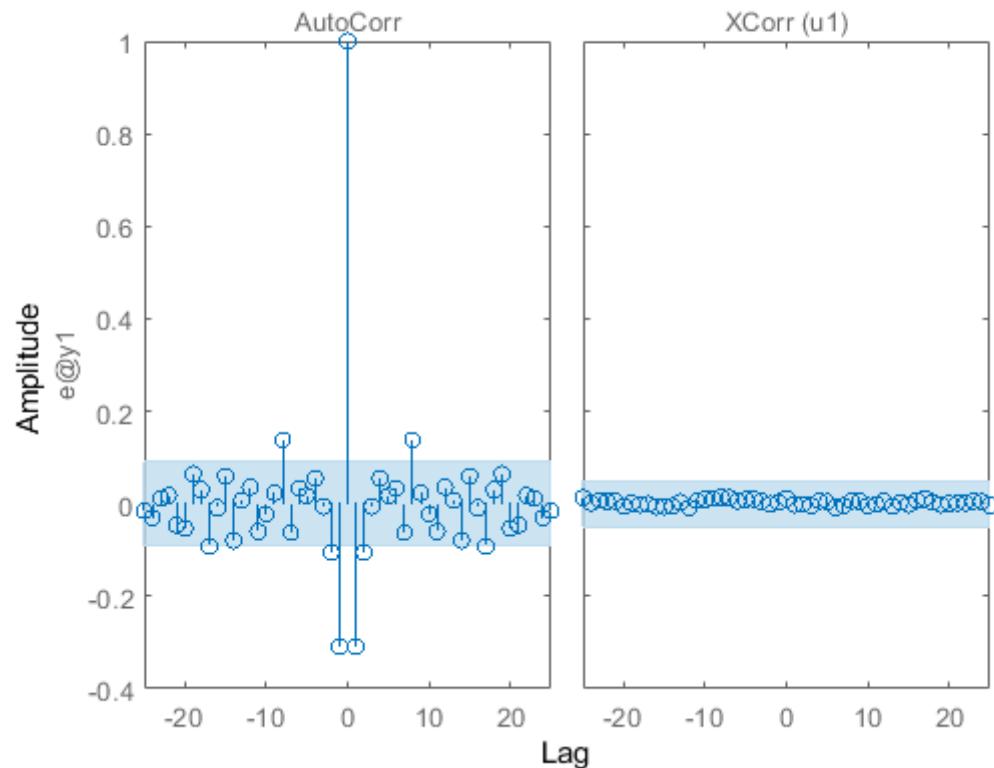
### ARARX Model



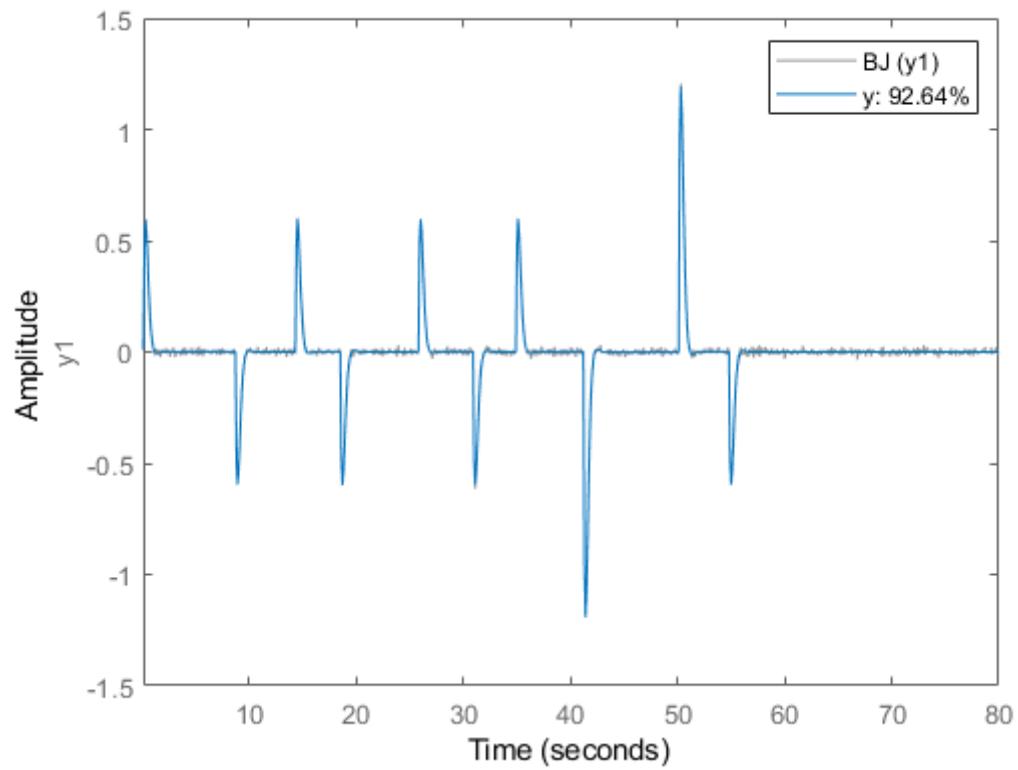
### OE Model



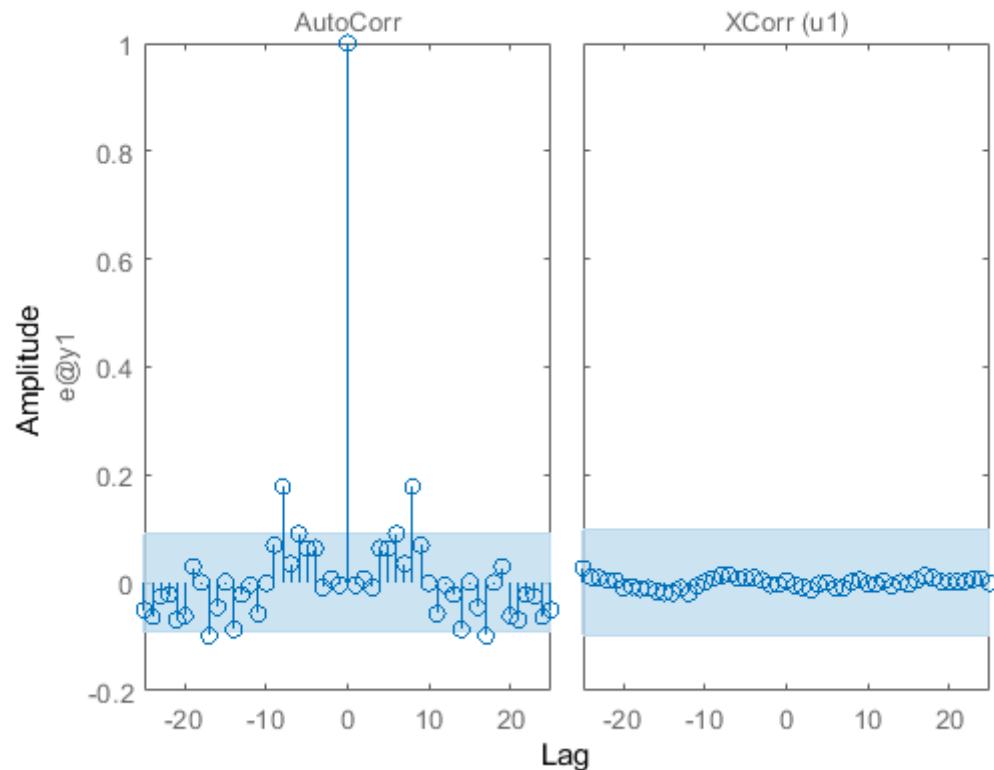
### OE Model



### BJ Model



### BJ Model



```
arx_aic =  
-7.6139
```

```
arimax_aic =
```

```
-8.5982
```

```
ararx_aic =
```

```
-8.6644
```

```
oe_aic =
```

```
-8.9456
```

```
bj_aic =
```

```
-9.1266
```

برای نویز متوسط

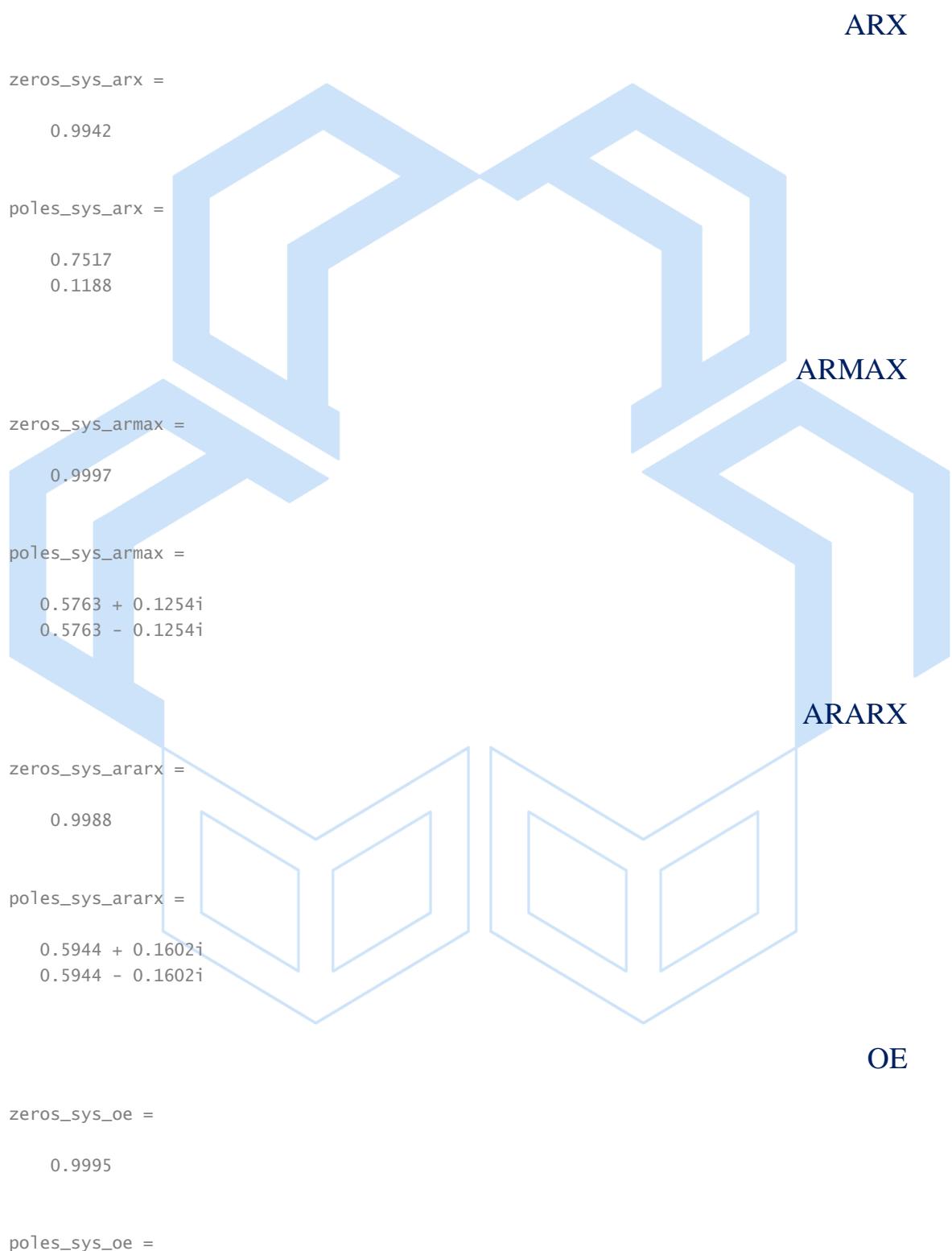
به ازای نویز متوسط مقدار صفر سیستم را در همه حالت ها به خوبی تشخیص داده شده است اما مقادیر قطب ها اشتباه تشخیص داده شده اند و در OE,BJ مقدار قطب ها نسبت به بقیه سیستم ها بهتر تخمین زده شده اند. و در مدل ARX در شناسایی قطب ها ضعیف ترین عملکرد است

درصد فیت شدن مدل ها بجز ARX نیز بالا 70 درصد میباشد و همچنین در بیشتر مدل ها خروجی نمودار های کورولیشن به نسبت مناسب میباشد و بیشتر دینامیک های سیستم را به خوبی توانسته اند شناسایی کنند اما به نسبت به حالت قبلی دینامیک های بیشتری از مقدار استانه عبور کرده اند

و مقادیر AIC نیز نسبت به حالت قبلی بزرگتر شده اند

به ترتیب تخمین ARX عملکرد ضعیفی داشته اند. و بقیه مدل ها هم تقریباً عملکرد مشابه ای داشته

اند



$0.6236 + 0.1965i$   
 $0.6236 - 0.1965i$

BJ

`zeros_sys_bj =`

$0.9995$

`poles_sys_bj =`

$0.6261 + 0.1996i$   
 $0.6261 - 0.1996i$

`error_arx =`

$0.0031$

`error_armax =`

$0.0020$

`error_ararx =`

$0.0020$

`error_bj =`

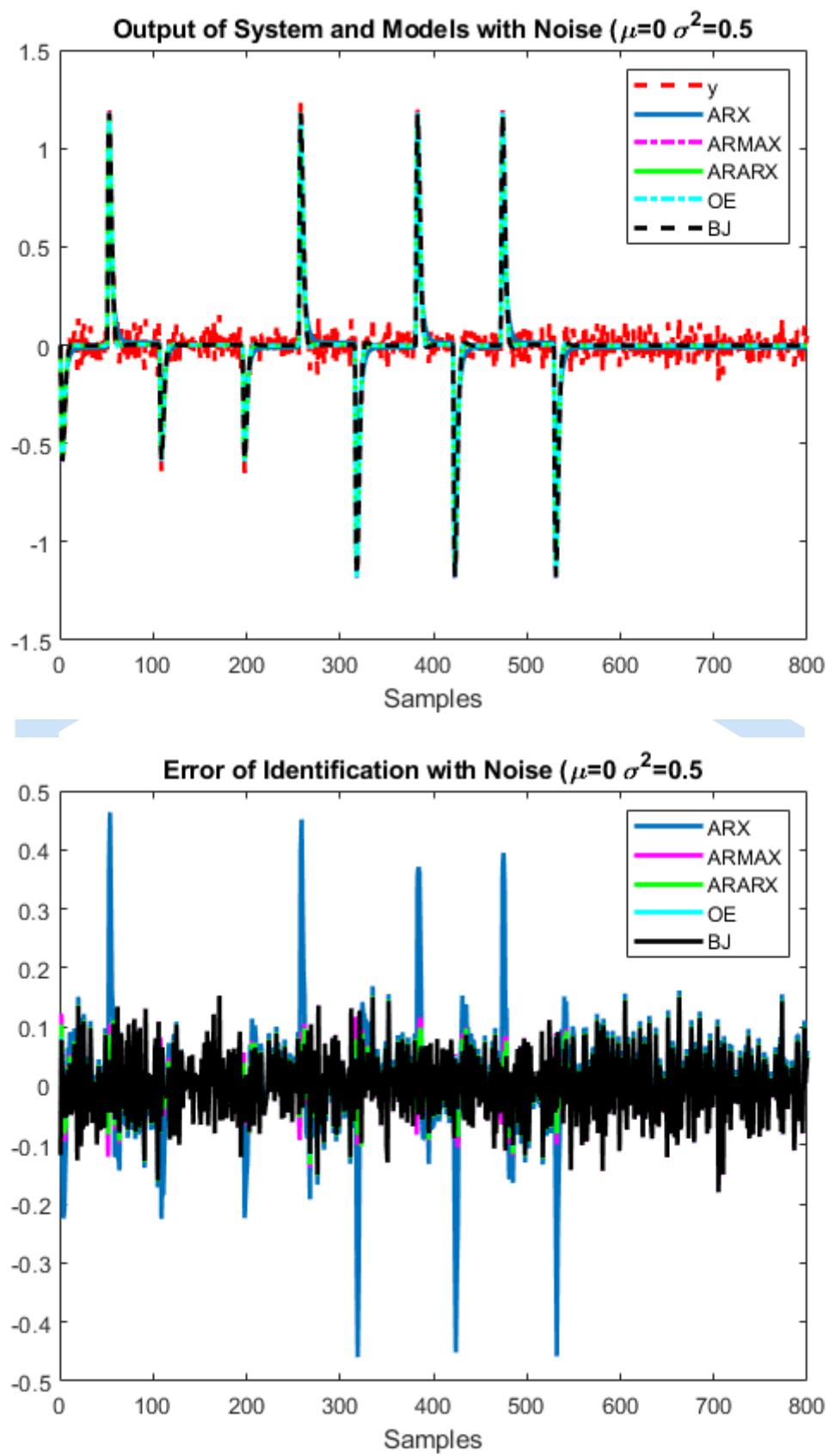
$0.0019$

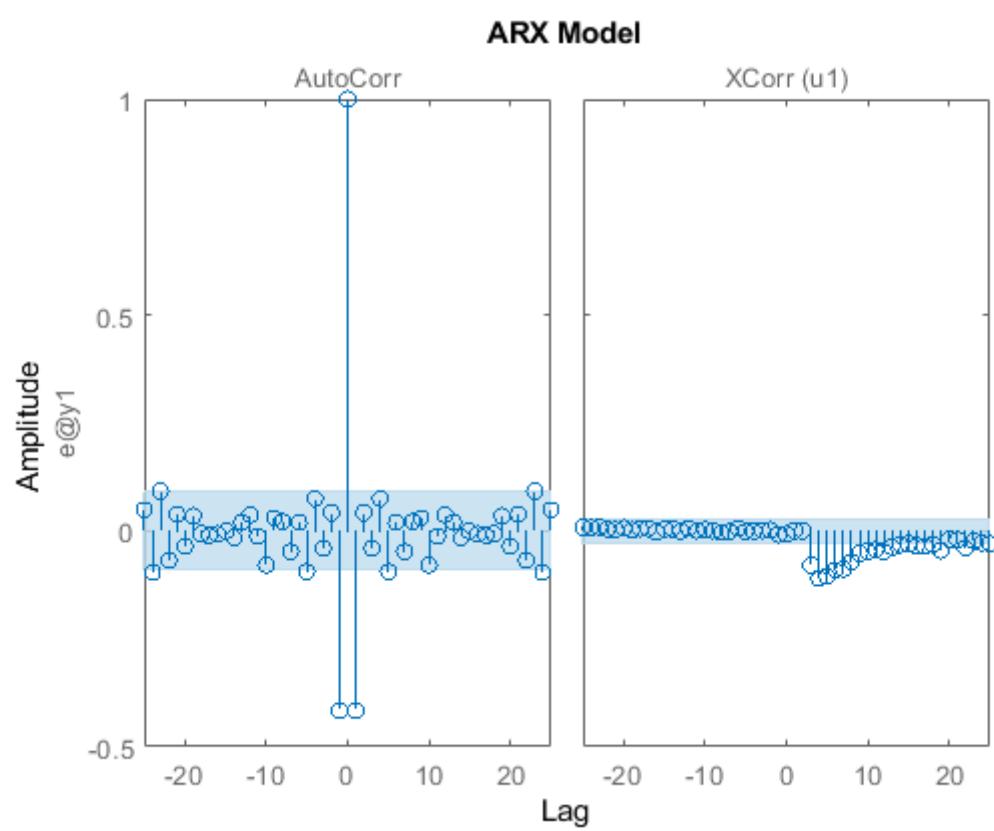
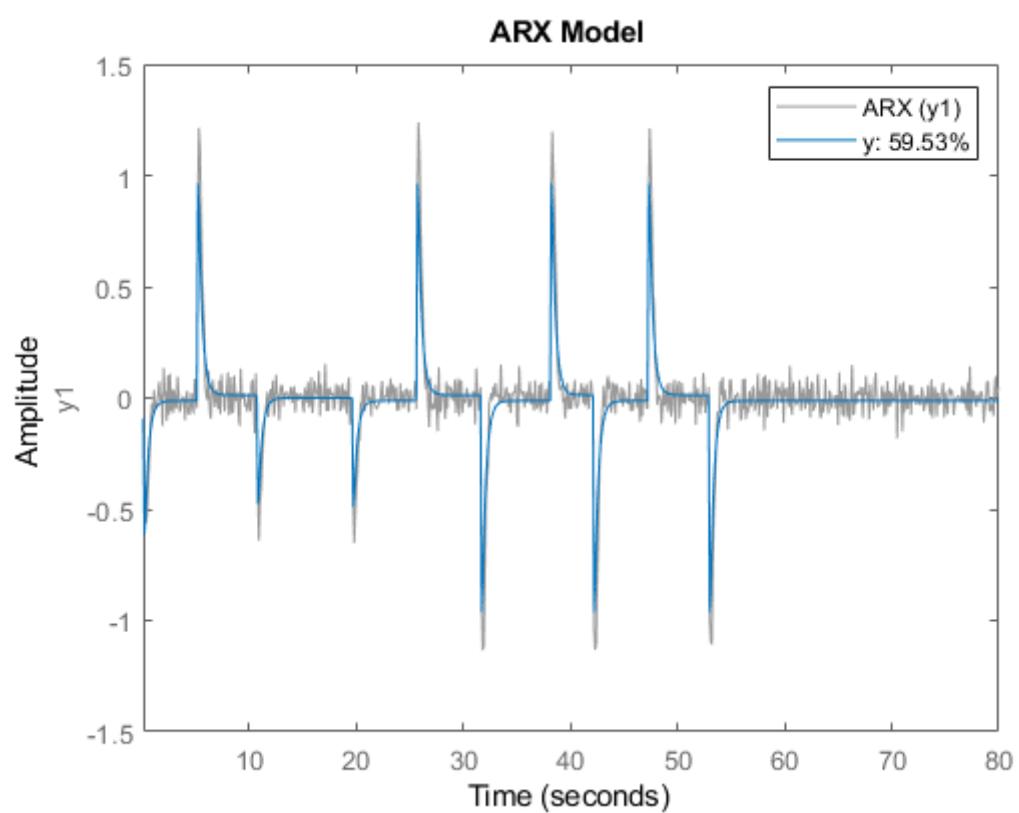
`error_oe =`

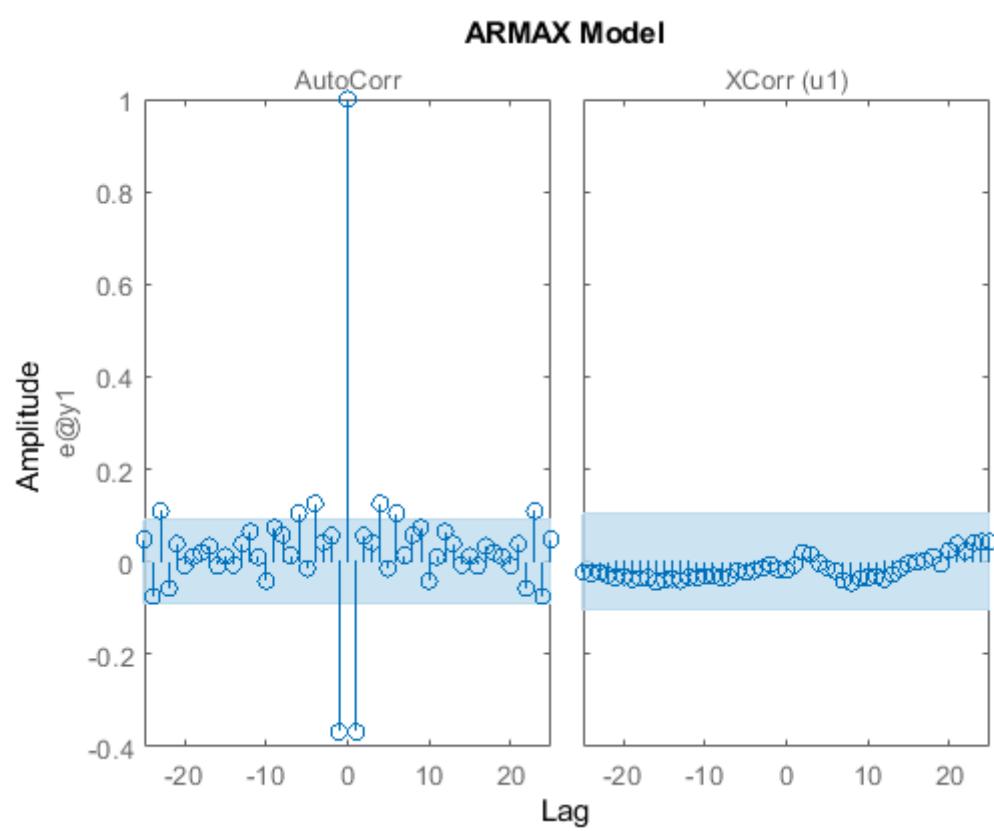
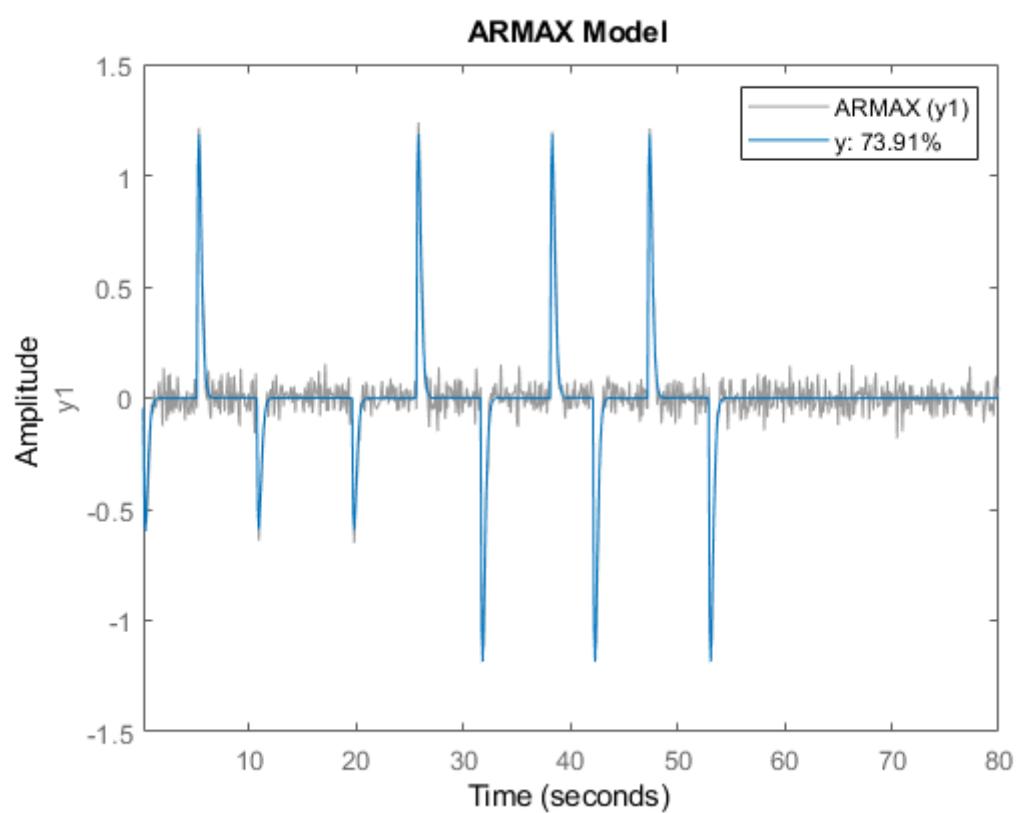
$0.0019$

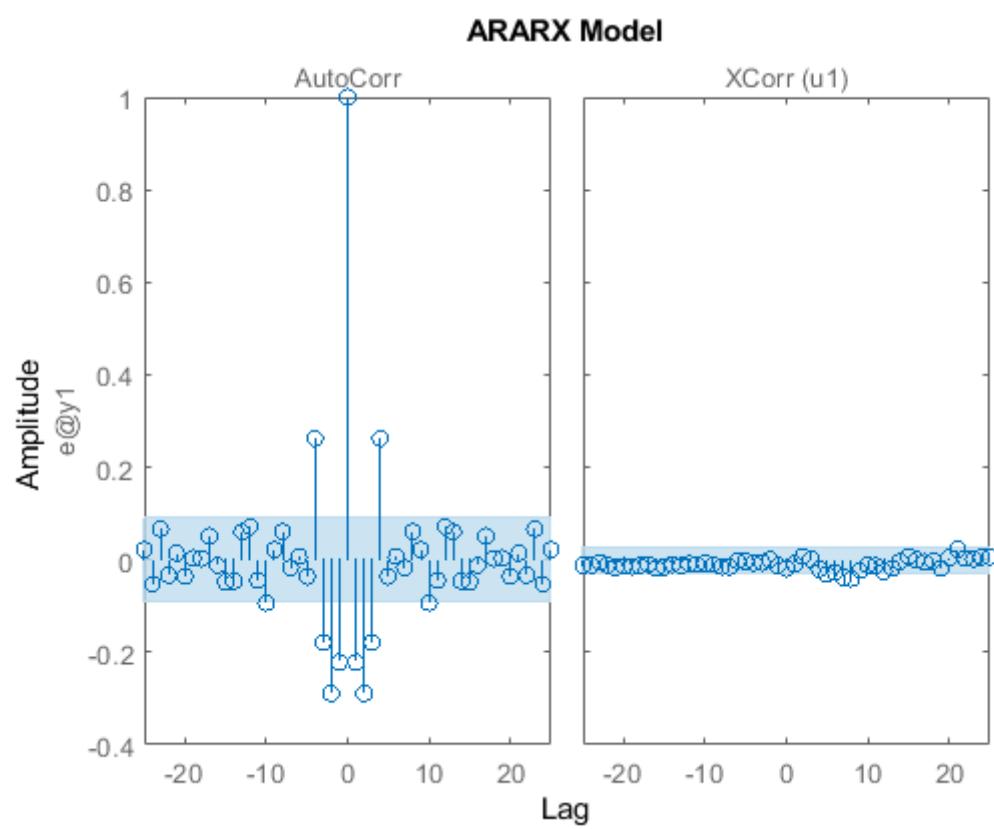
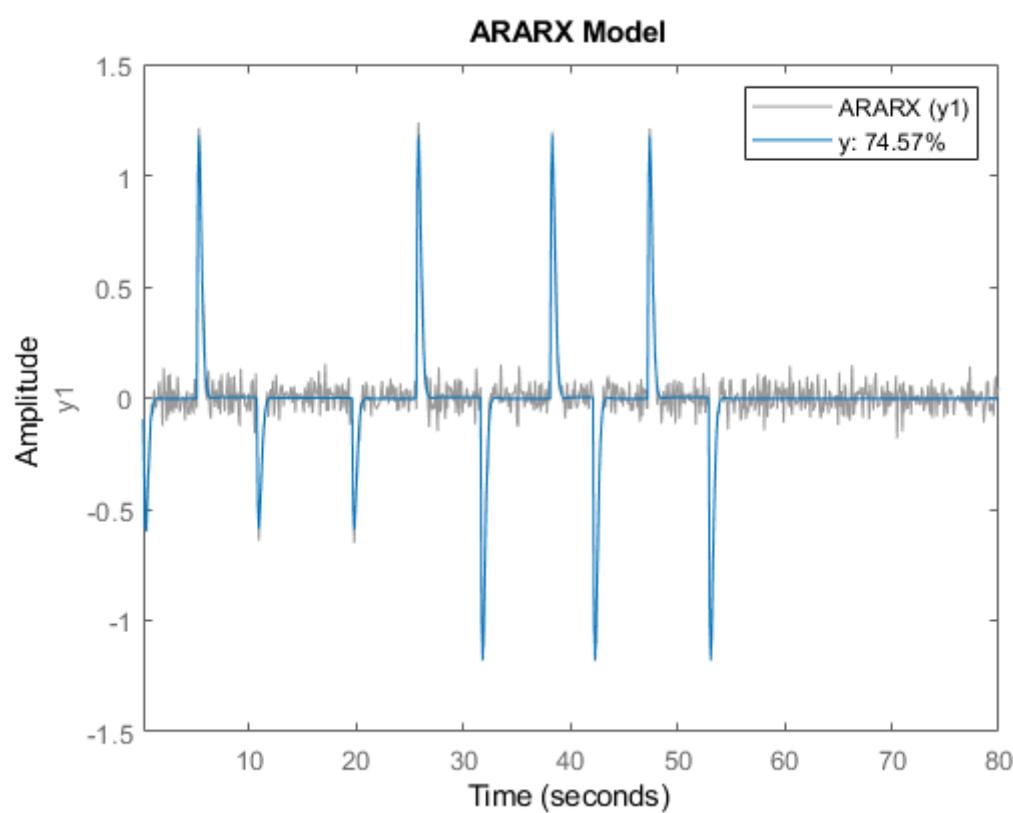
Errors

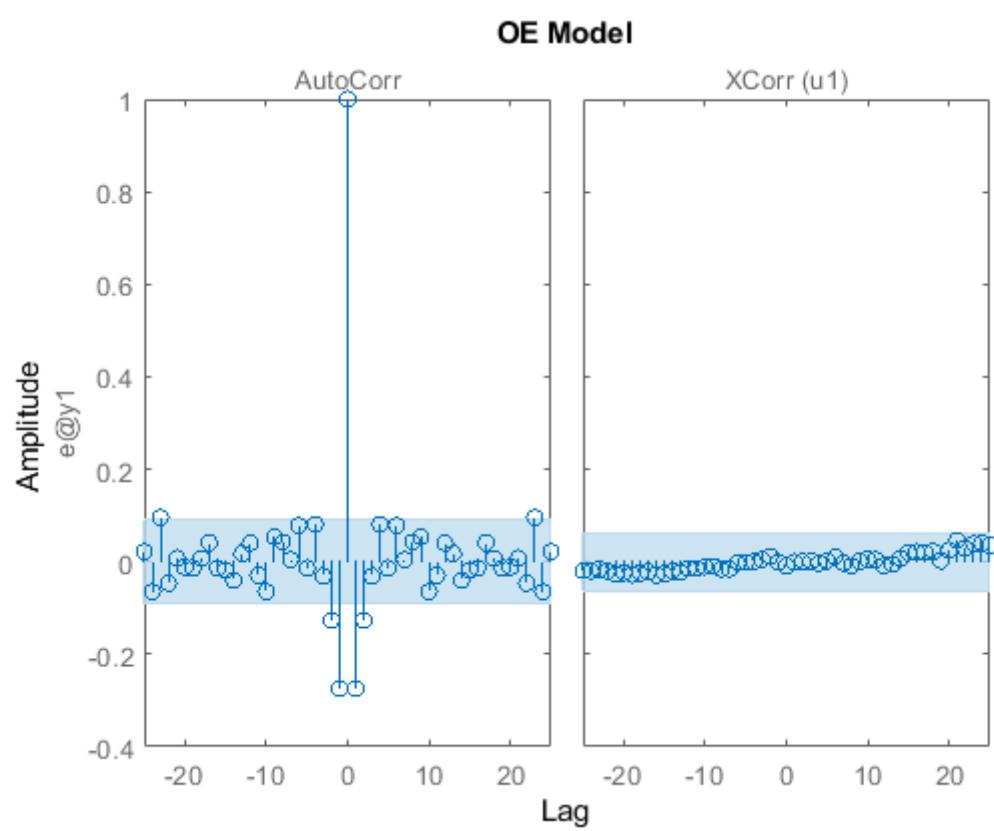
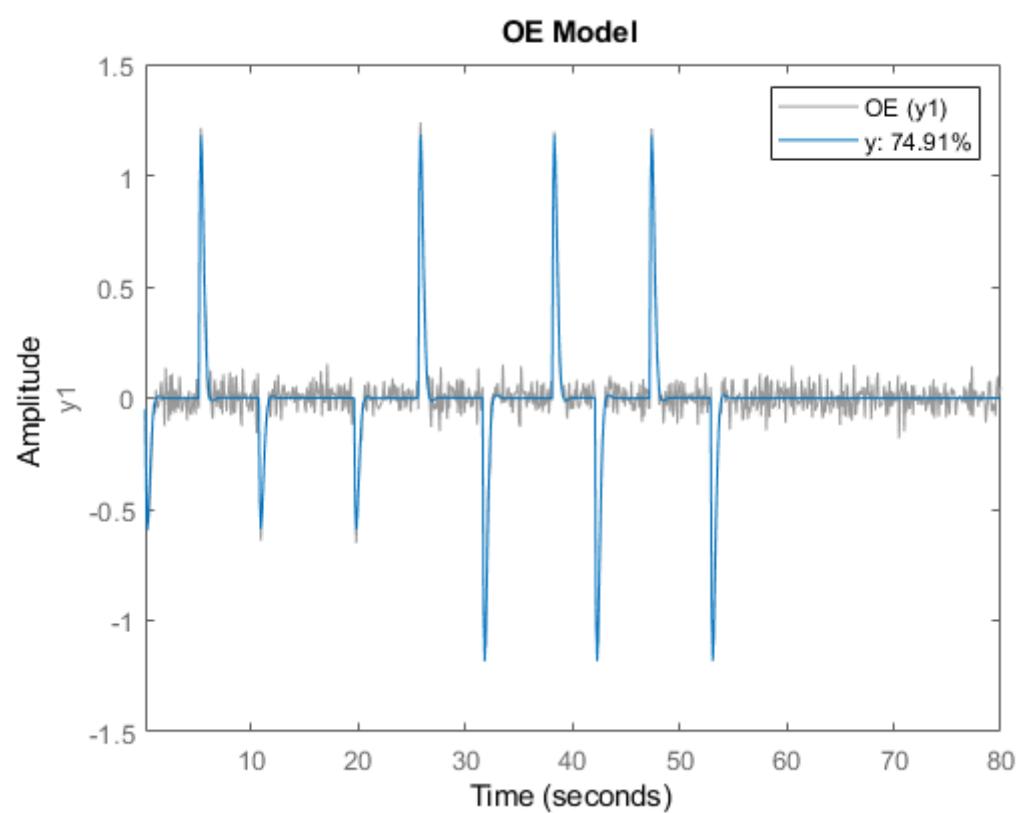
## Figure and result

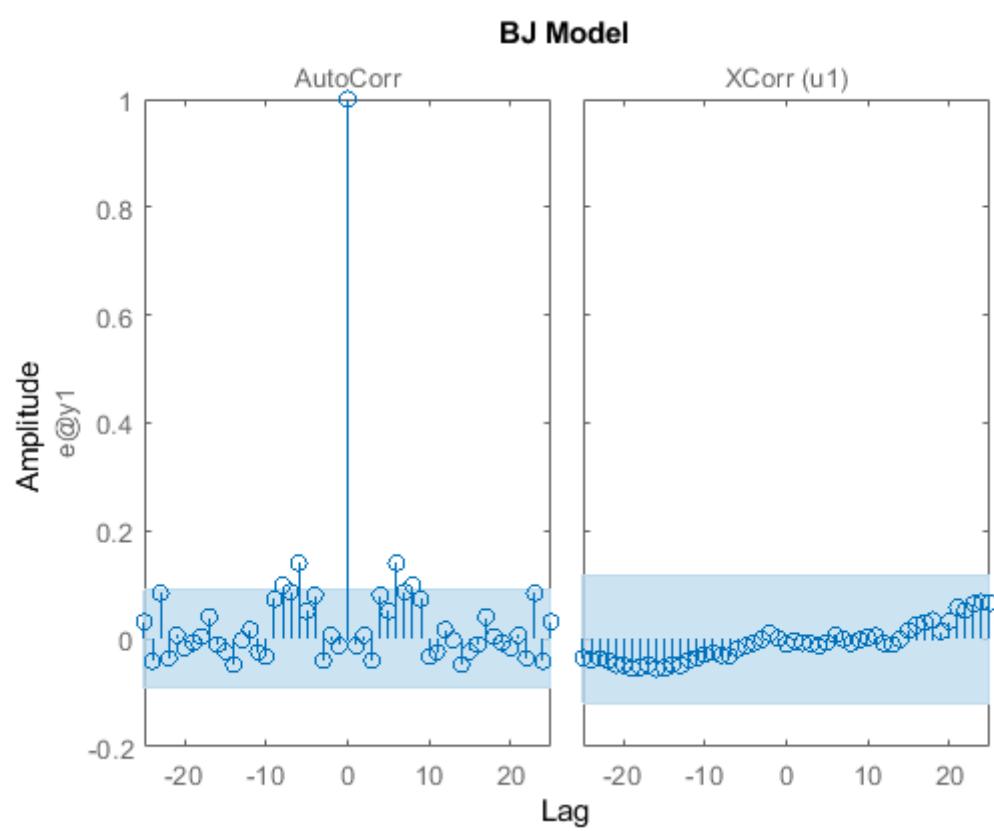
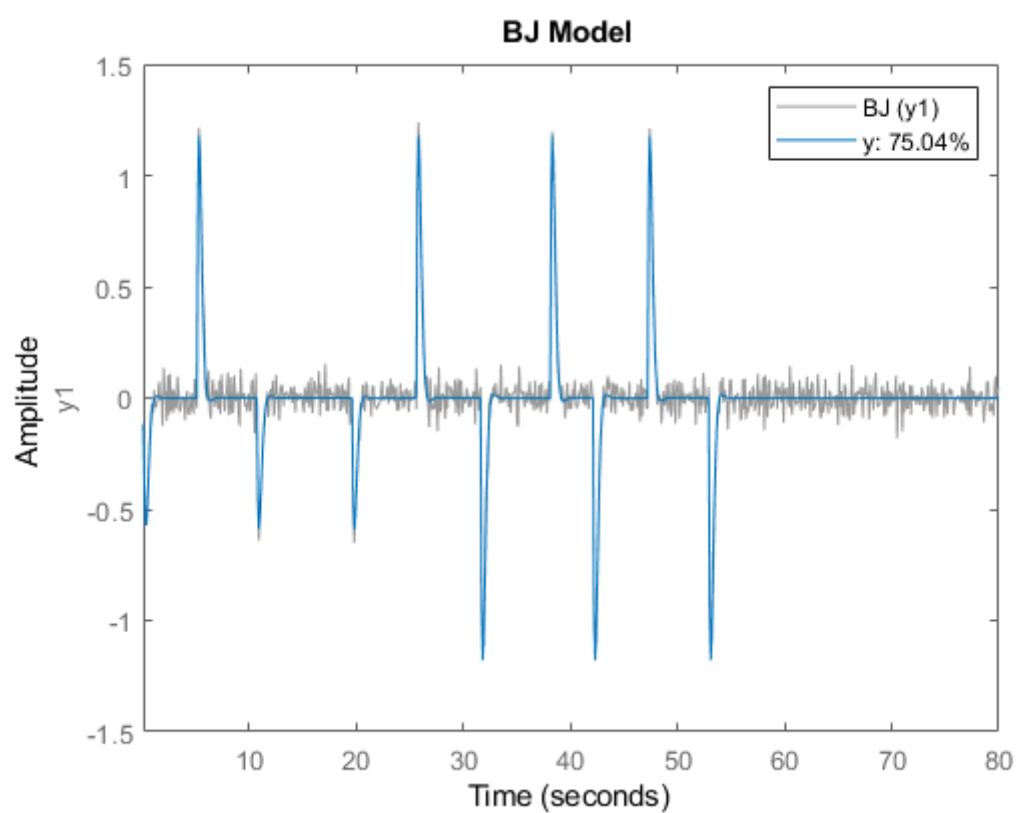






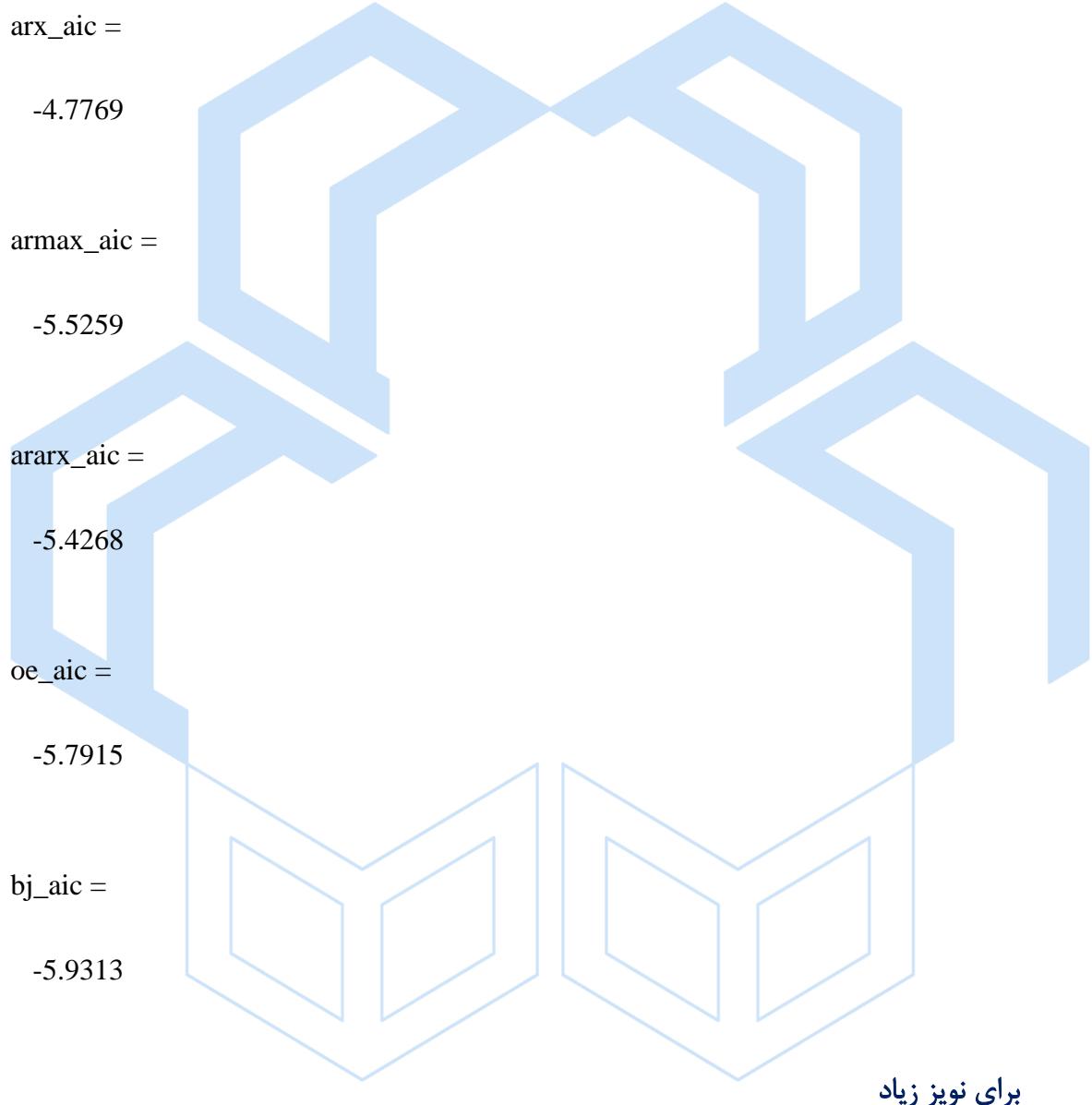






## AIC

```
arx_aic = aic( y_arx )
armax_aic = aic( y_armax )
ararx_aic = aic( y_ararx )
oe_aic = aic( y_oe )
bj_aic = aic( y_bj )
```

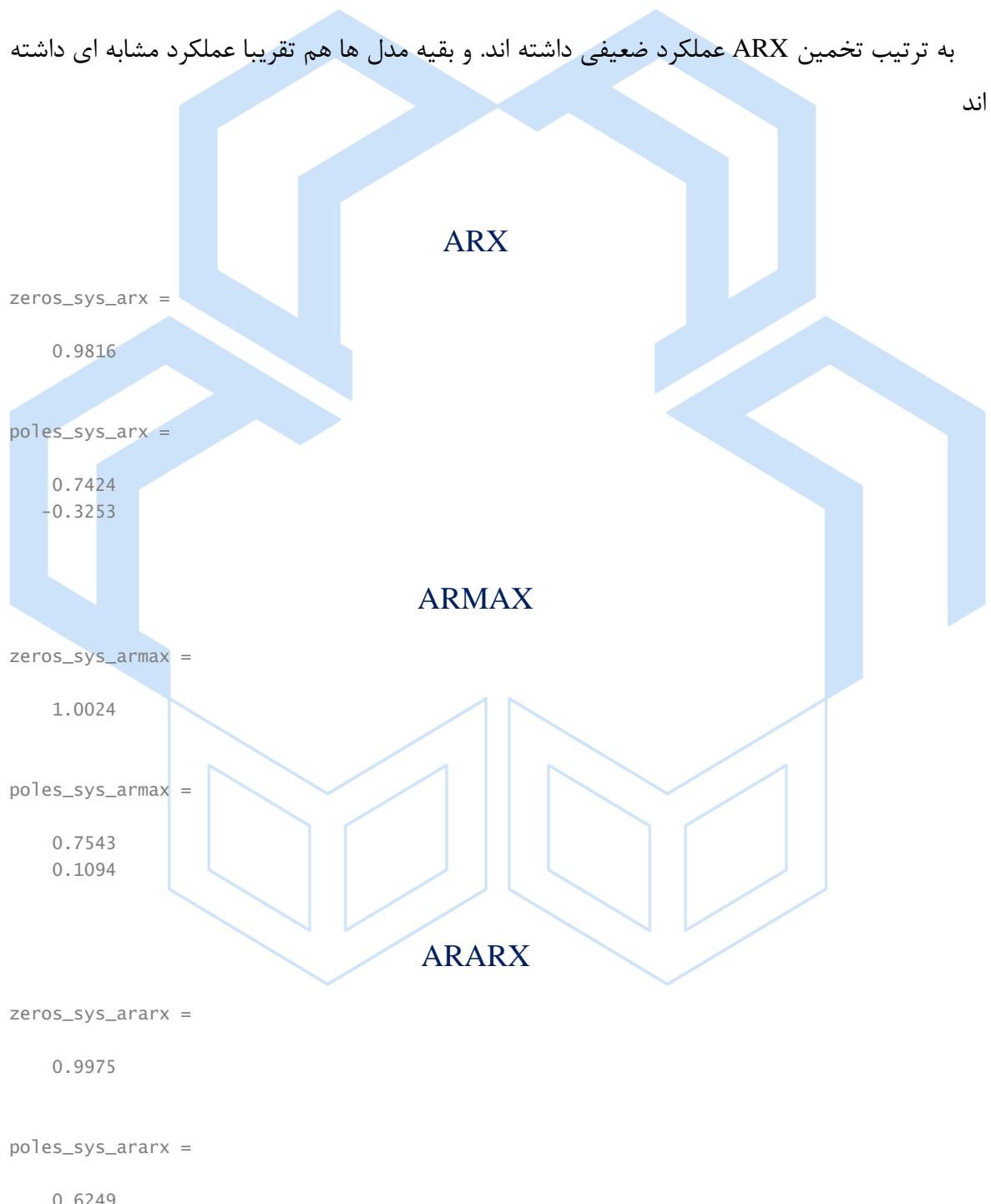


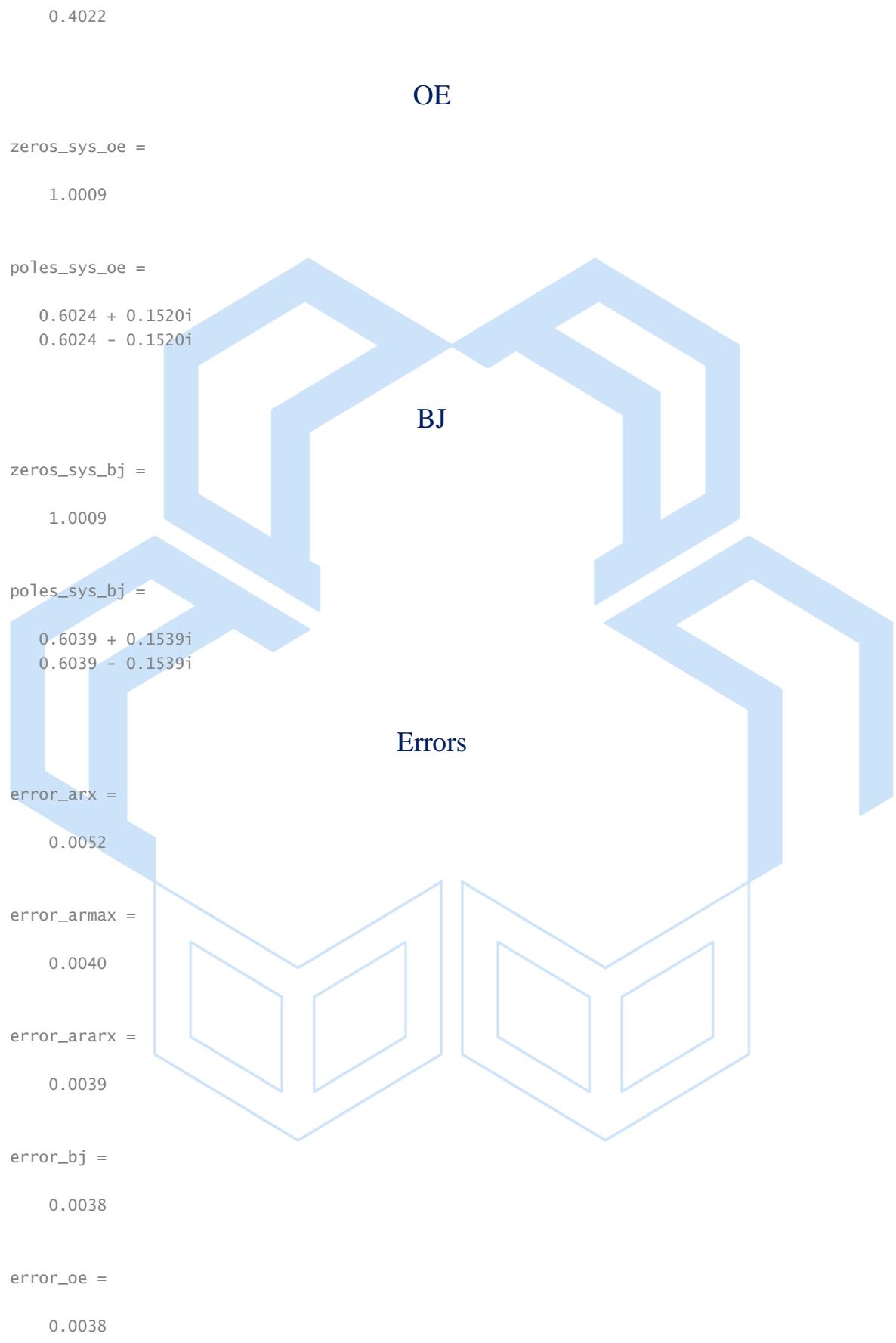
به ازای نویز زیاد مقدار صفر سیستم را در همه حالت ها به خوبی تشخیص داده شده است اما مقادیر قطب ها اشتباه تشخیص داده شده اند و مدل ARX در شناسایی قطب ها ضعیف ترین عملکرد است. همچنین در این حالت نسبت به دو حالت قبلی در شناسایی قطب ها عملکرد بدتری را شاهد بوده ایم

درصد فیت شدن مدل ها بجز ARX نیز بین 45 تا 48 درصد میباشد مدل ARX 29 درصد فیت شده است و همچنین در بیشتر مدل ها خروجی نمودار های کورولیشن به نسبت مناسب میباشد و بیشتر دینامیک های سیستم را به خوبی توانسته اند شناسایی کنند اما به نسبت به حالت قبلی دینامیک های بیشتری از مقدار استانه عبور کرده اند

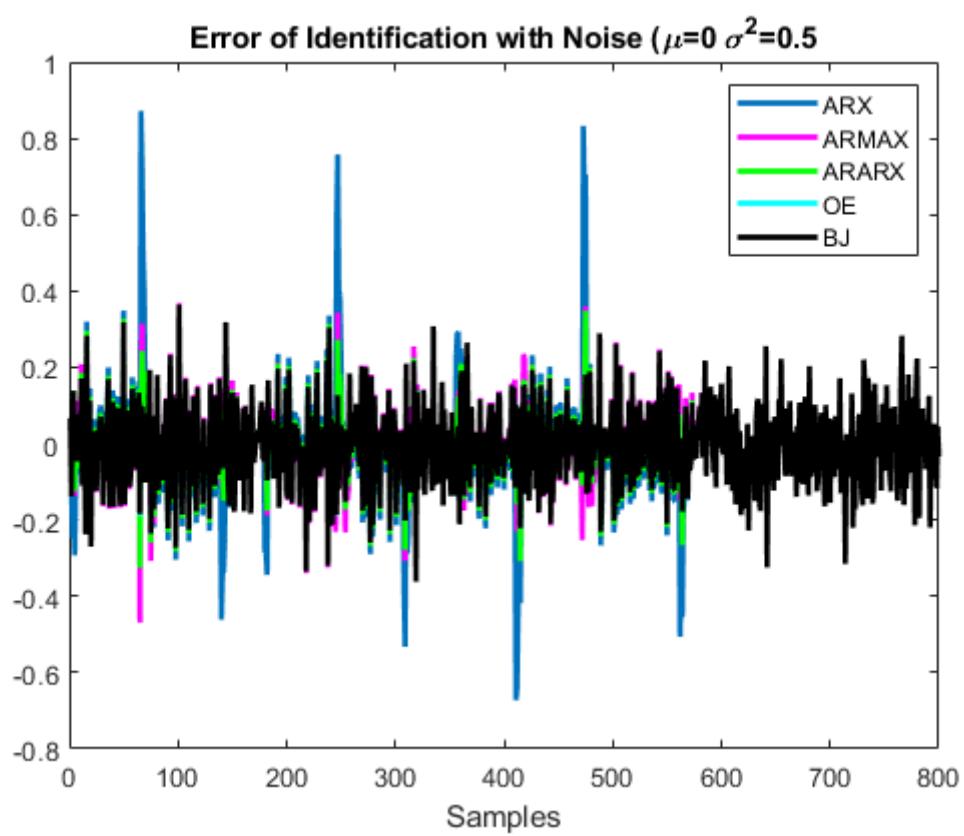
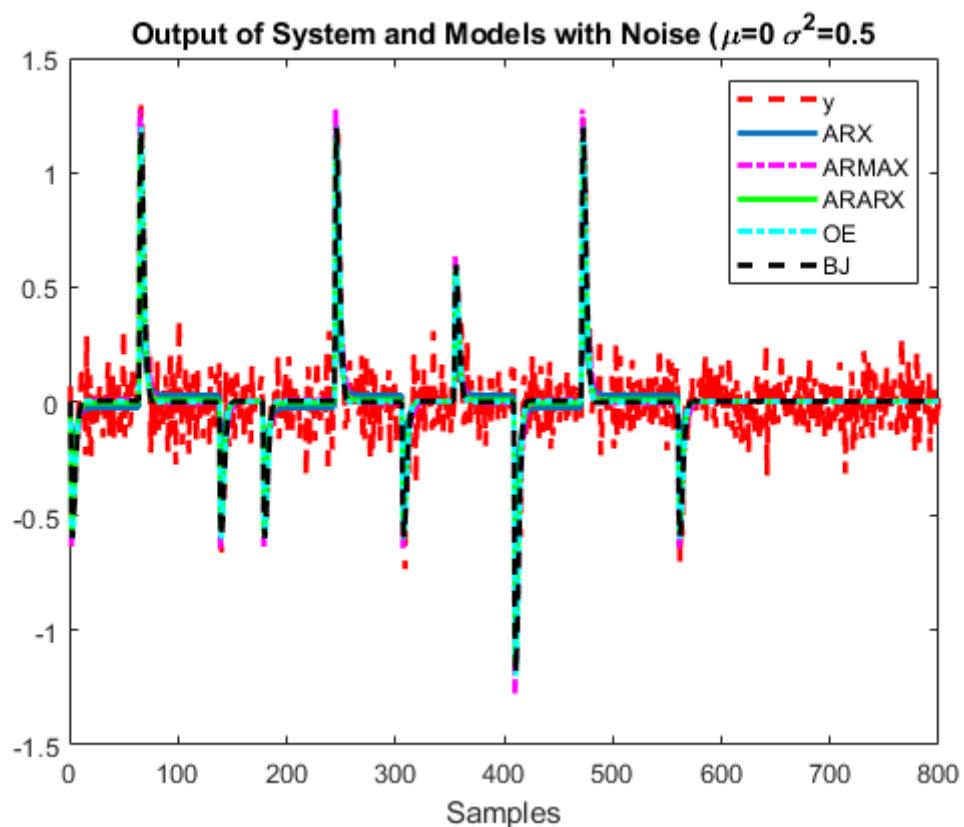
و مقادیر AIC نیز نسبت به حالت قبلی بزرگتر شده اند

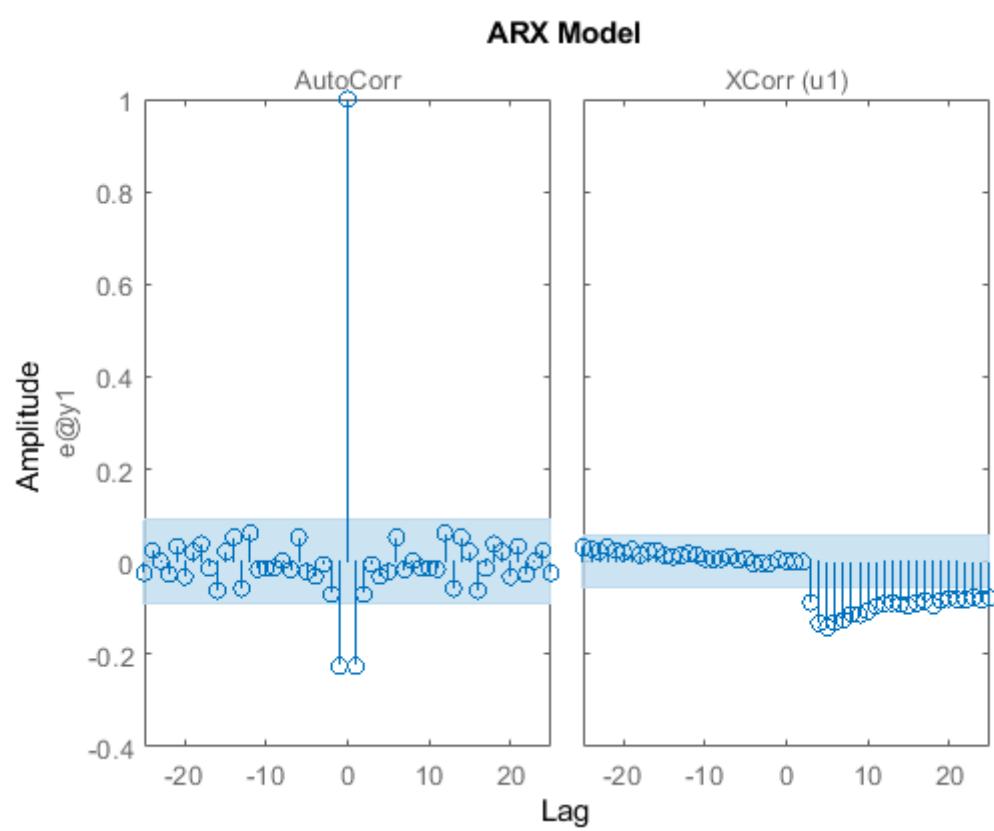
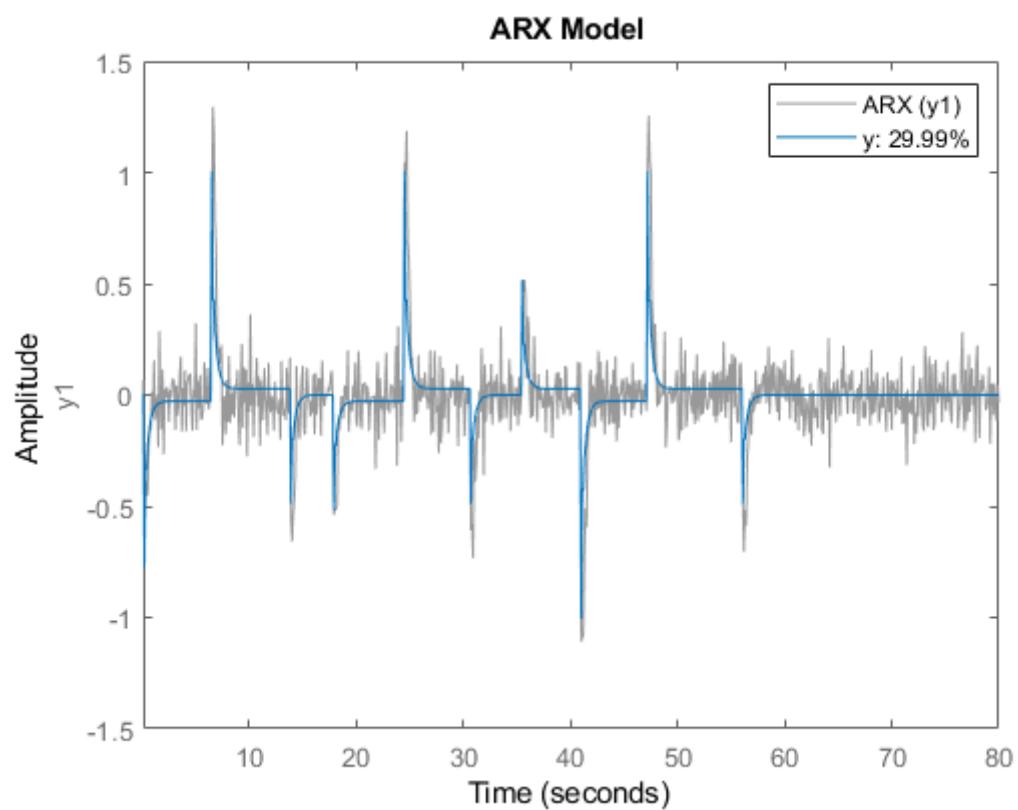
به ترتیب تخمین ARX عملکرد ضعیفی داشته اند. و بقیه مدل ها هم تقریباً عملکرد مشابه ای داشته اند

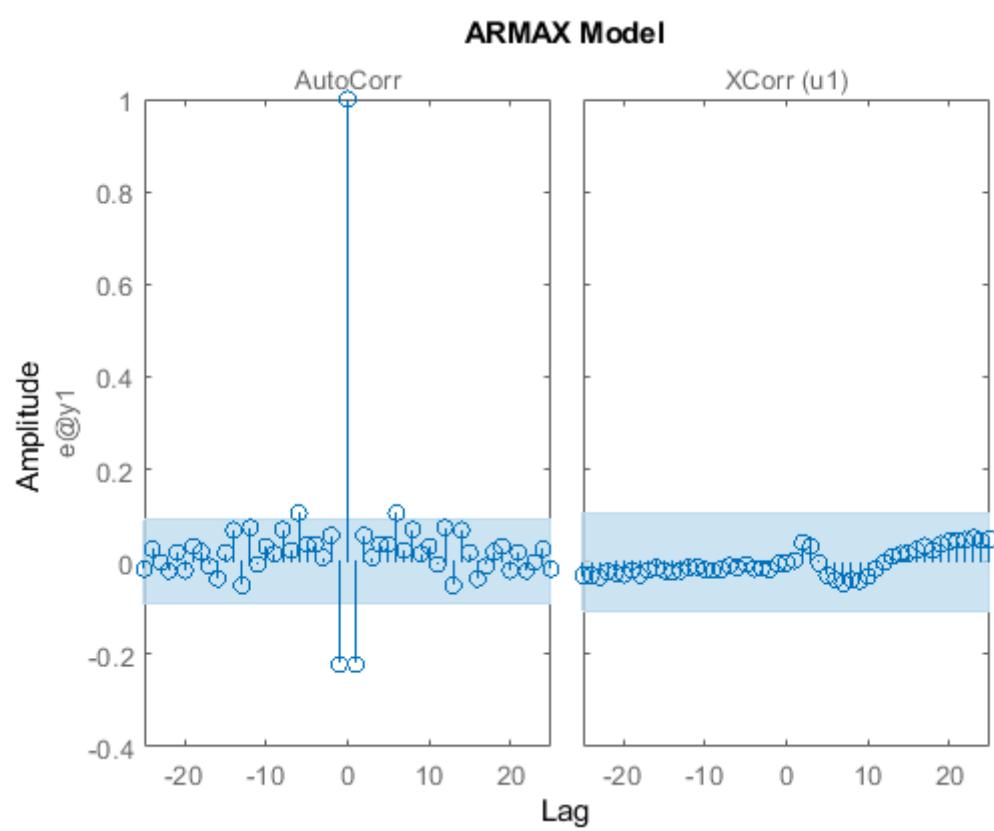
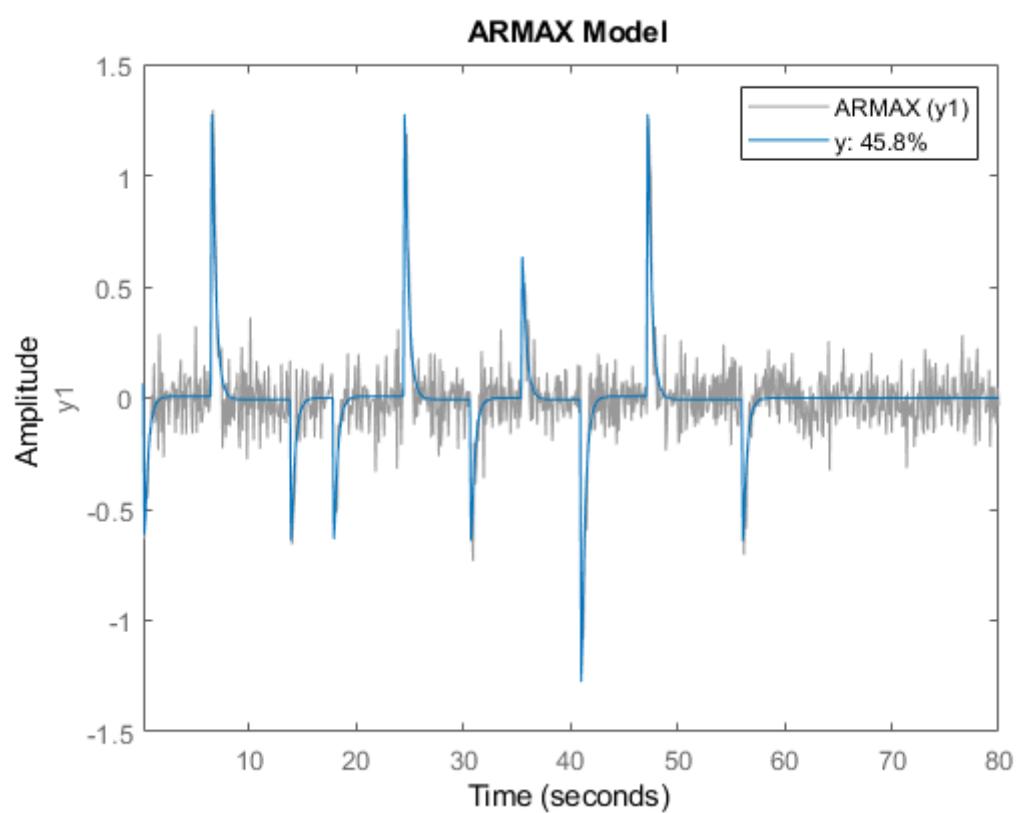


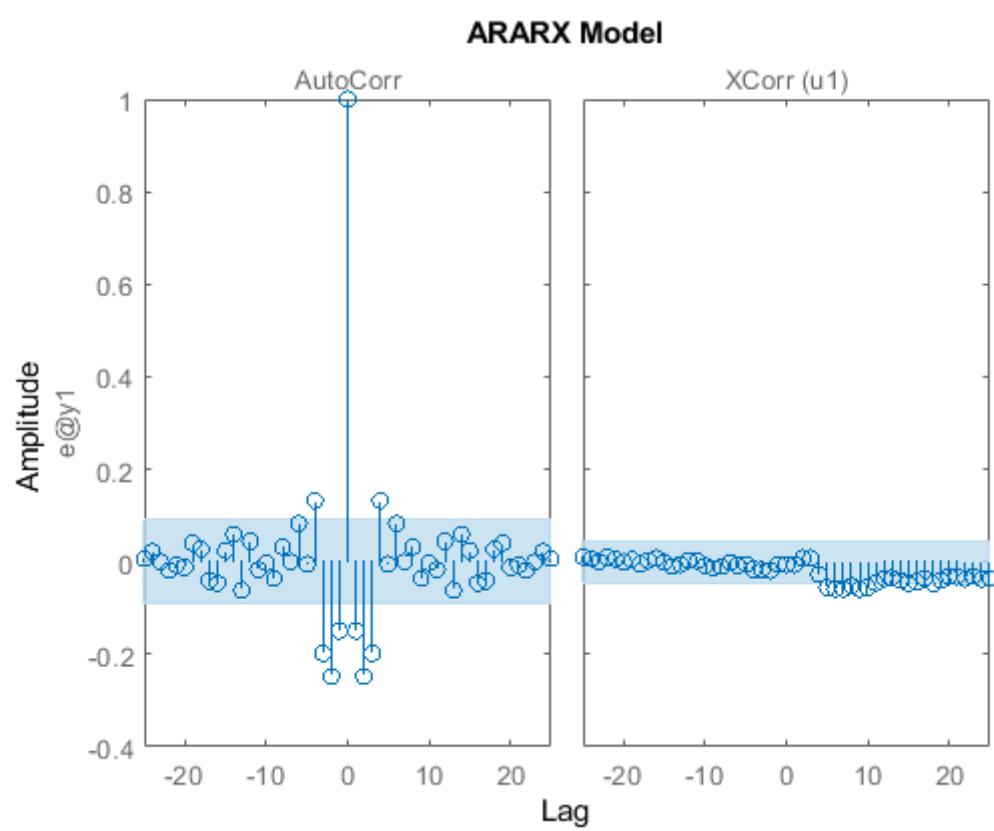
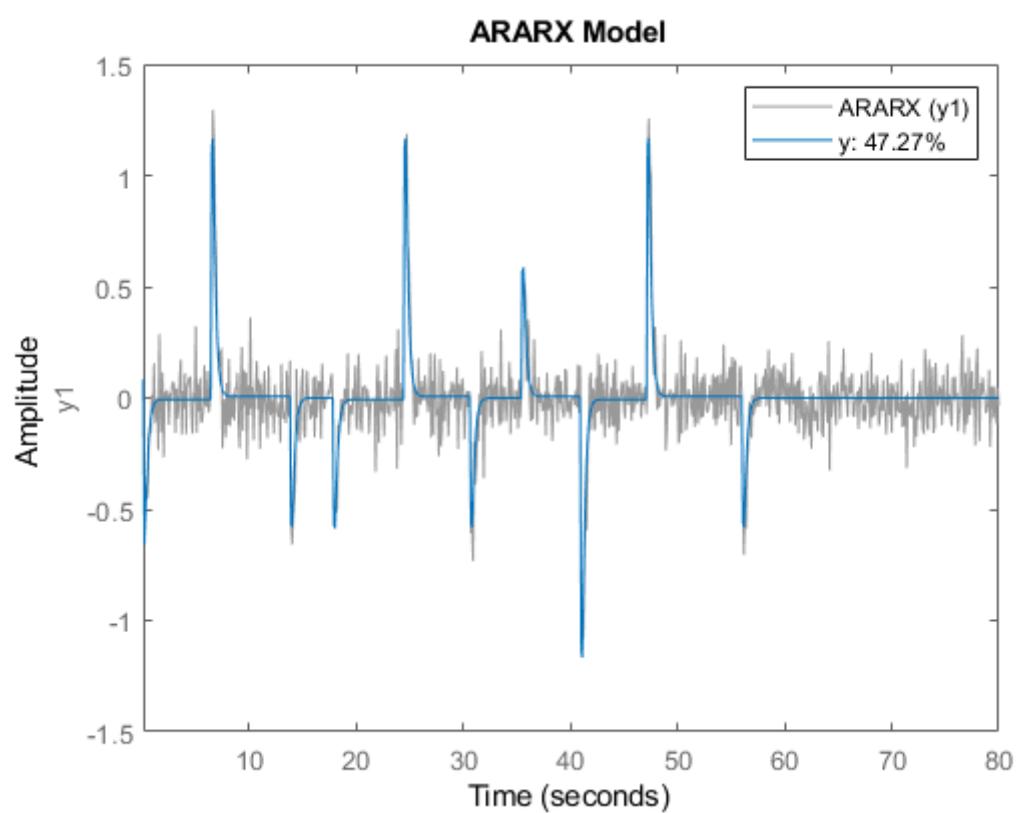


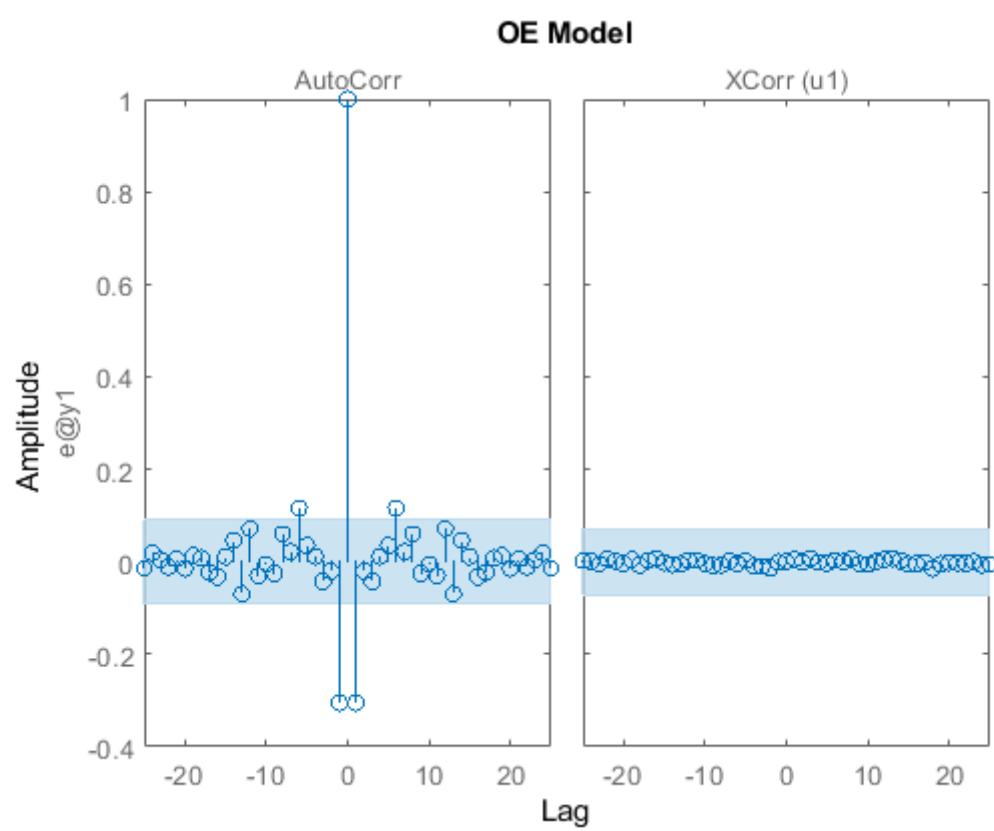
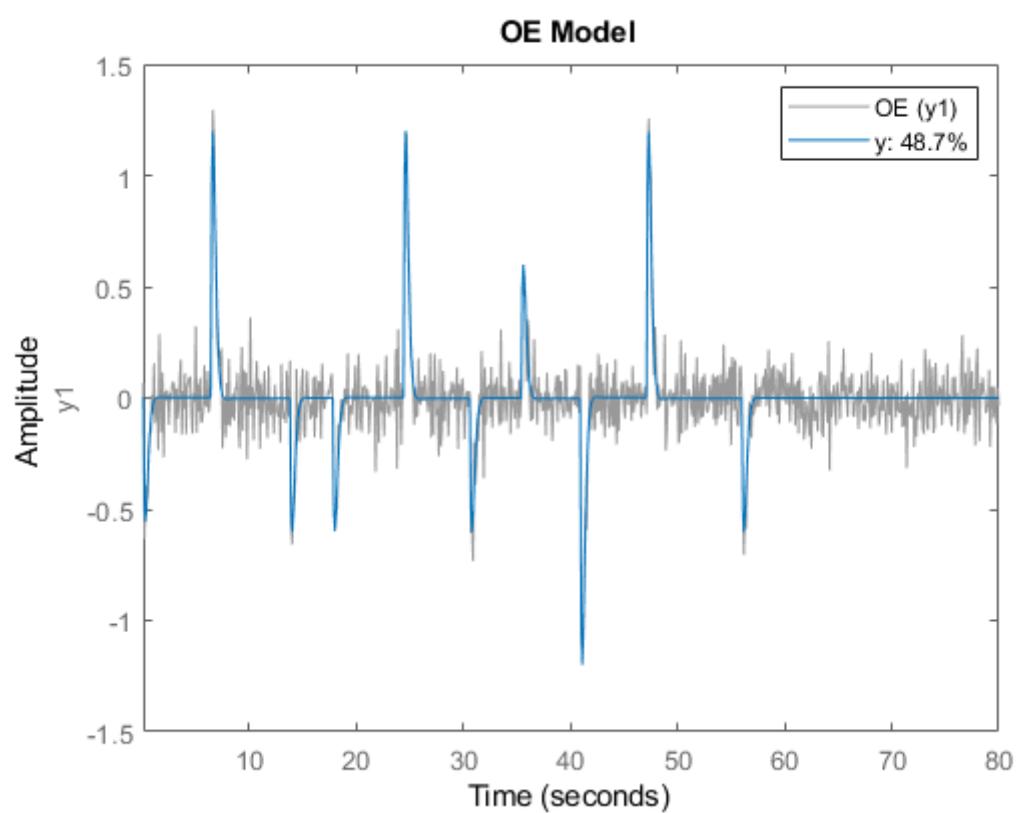
## Figure and result

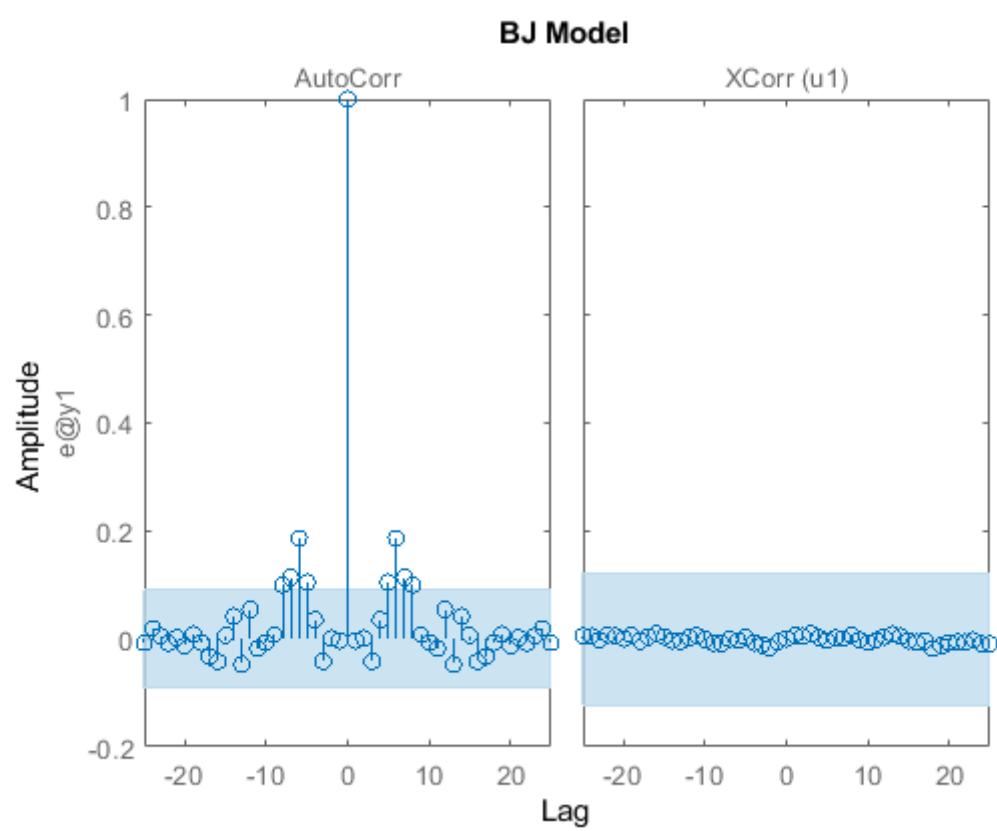
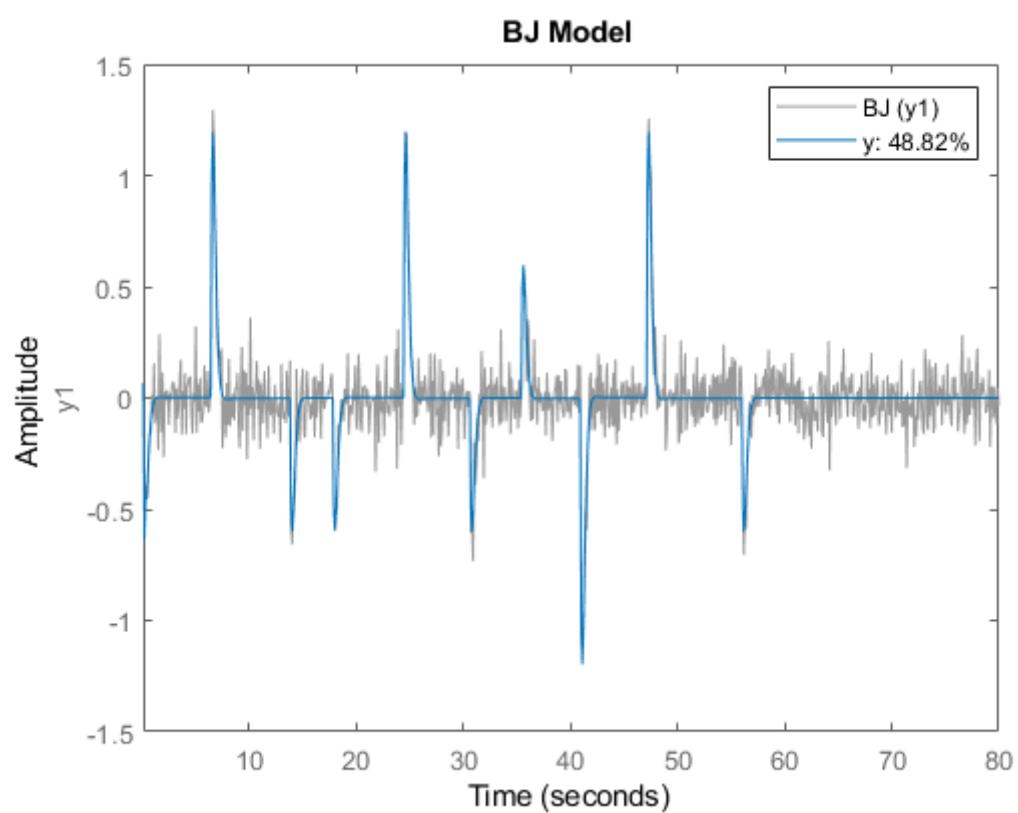












## AIC

```
arx_aic = aic( y_arx )
armax_aic = aic( y_armax )
ararx_aic = aic( y_ararx )
oe_aic = aic( y_oe )
bj_aic = aic( y_bj )
```

arx\_aic =

-3.8237

armax\_aic =

-4.3066

ararx\_aic =

-4.2158

oe\_aic =

-4.4337

bj\_aic =

-4.5455

## سوال دوم

### قسمت الف

در ابتدا طبق خواسته مسئله داده ها را به دو بخش تست و ترین تقسیم بندی میکنیم و 60 درصد ابتدایی داده ها را به عنوان ترین و 40 درصد باقی مانده را به عنوان تست انتخاب میکنیم. چون در صورت سوال گفته شده است که سیستم حلقه باز به طور کلی تاخیر ندارد و فقط یک تاخیر برای نمونه برداری به وجود آمده است پس ما نیز در این مسئله مقدار تاخیر را برابر با یک در نظر میگیریم اگر در صورت سوال این موضوع اشاره نشده بود باید با استفاده از دستور `deleyset` به محاسبه مقدار تاخیر بهینه میپرداختیم

حال برای اینکه مدل دینامیکی که حداقل 70 درصد دقت داشته باشد را پیدا کنیم یک حلقه for

مینویسیم

```

for i=1:15
    for j=1:15
        for k=1:1
            y_arx=arx(z1(1:600),[i j k]);
            [~, acc(i,j,k), ~]=compare(y_arx,z1(601:1000));
            if acc(i,j,k)>70
                acc2(i,j,k)=acc(i,j,k);
            end
        end
    end
end

```

با استفاده از این حلقه تمامی حالت های ممکن برای متغیر های na,nb,nc,nf,nd را در هر مدل با توجه به پارامتر های ان محاسبه میکنید و به ازای ترکیب های مختلفی که این متغیر ها میسازند به شناسایی سیستم میپردازیم و دقت متغیر ها را در یک ماتریس ذخیره میکنیم و در مرحله بعدی ان درایه های ماتریسی که دارای دقت بالاتر از 70 درصد هستند را در ماتریس جداگانه ای ذخیره میکنیم باید به این نکته توجه داشت که این کار حجم محاسباتی و زمان زیادی را میگیرد اما میتواند تمام ترکیب های موجود را محاسبه کند. بعد از محاسبه این ماتریس برای سادگی مدلی که پارامتر های ان مقدار کمتری داشته باشند را انتخاب میکنیم باید به این نکته توجه کنیم با این کار ممکن است در نمودار های اتوکورولیشن و کراس کورولیشن بعضی از دینامیک ها باقی بمانند اما با توجه به اصل سادگی ما مدلی که پارامتر های کوچک تری دارد را انتخاب کرده ایم و اگر بخواهیم ان دینامیک ها را نیز در مدل لحاظ کنیم باید مدلی که پارامترهای بزرگتری دارد را انتخاب کنیم البته باید این نکته را در نظر بگیریم که تغییرات دقت با تغییر مقدار پارامتر ها نموداری با شبیه خیلی کمی است و تقریباً دقت خروجی با افزایش پارامتر ها افزایش خیلی کمی دارد پس با توجه به اصل سادگی مدلی که پارامتر های کمتری دارد را باید انتخاب کنیم

همچنین میتوانستیم بجای این ماتریس به کمک سعی و خطا و با استفاده از این نکته که تمامی کسر ها باید سره باشند استفاده کنیم و درجه صورت را یکی از درجه مخرج کمتر در نظر بگیریم استفاده کنیم همچنین باید به این نکته توجه کنیم که درجه دینامیک های نویز باید از خود سیستم کمتر باشد

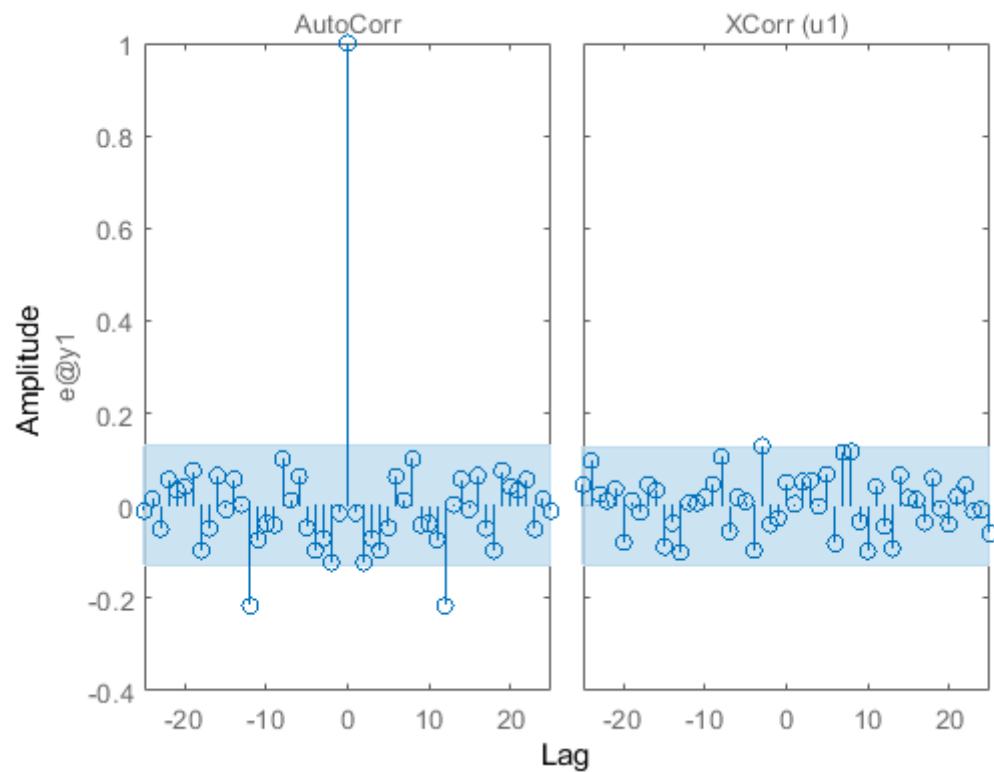
$$n_b = n_a - 1, n_c < n_d, n_a = n_f \\ n_d = n_a; n_d < n_f$$

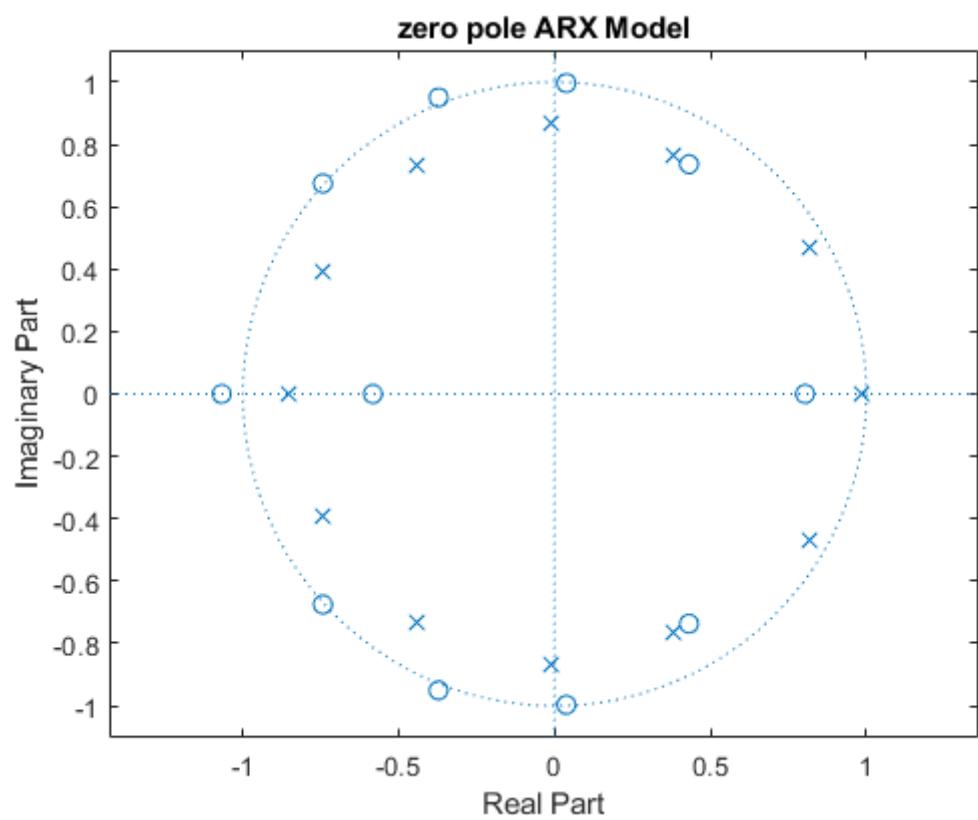
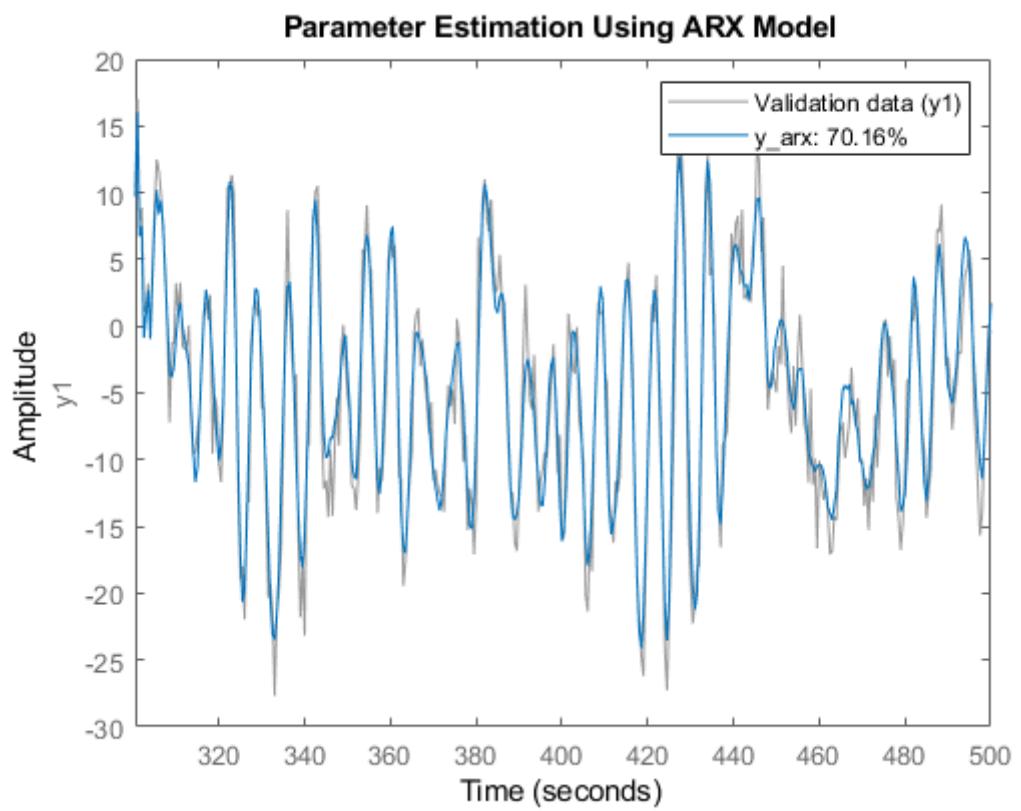
## ARX مدل

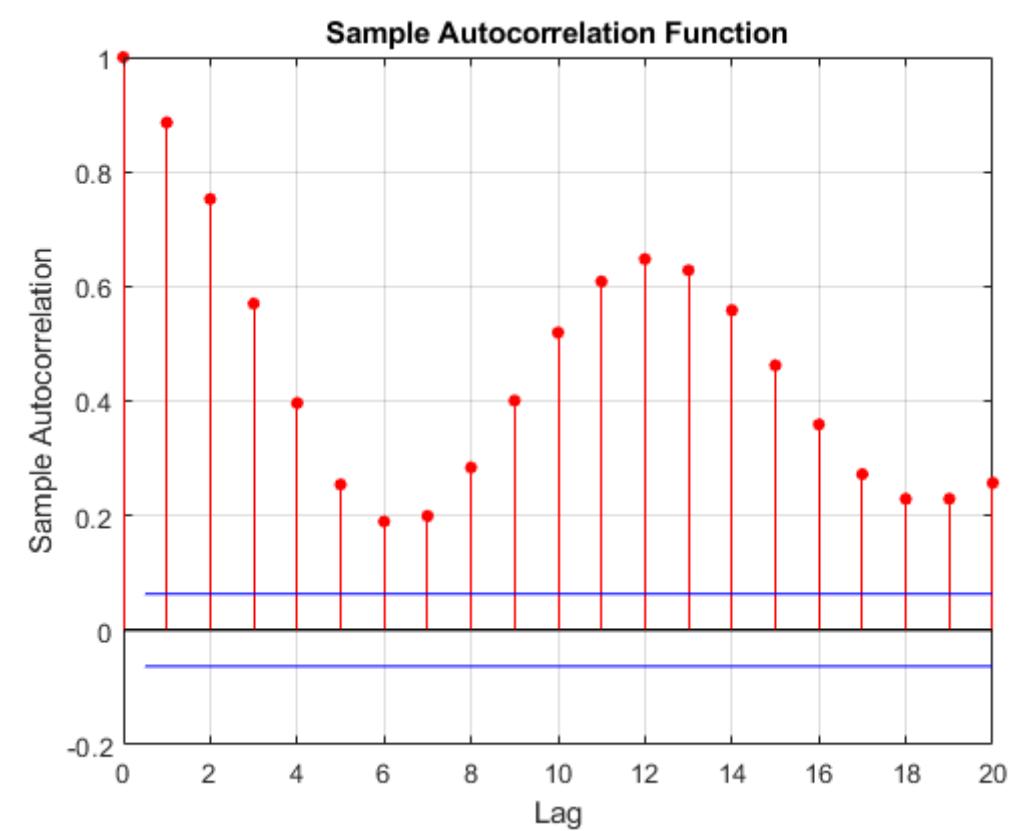
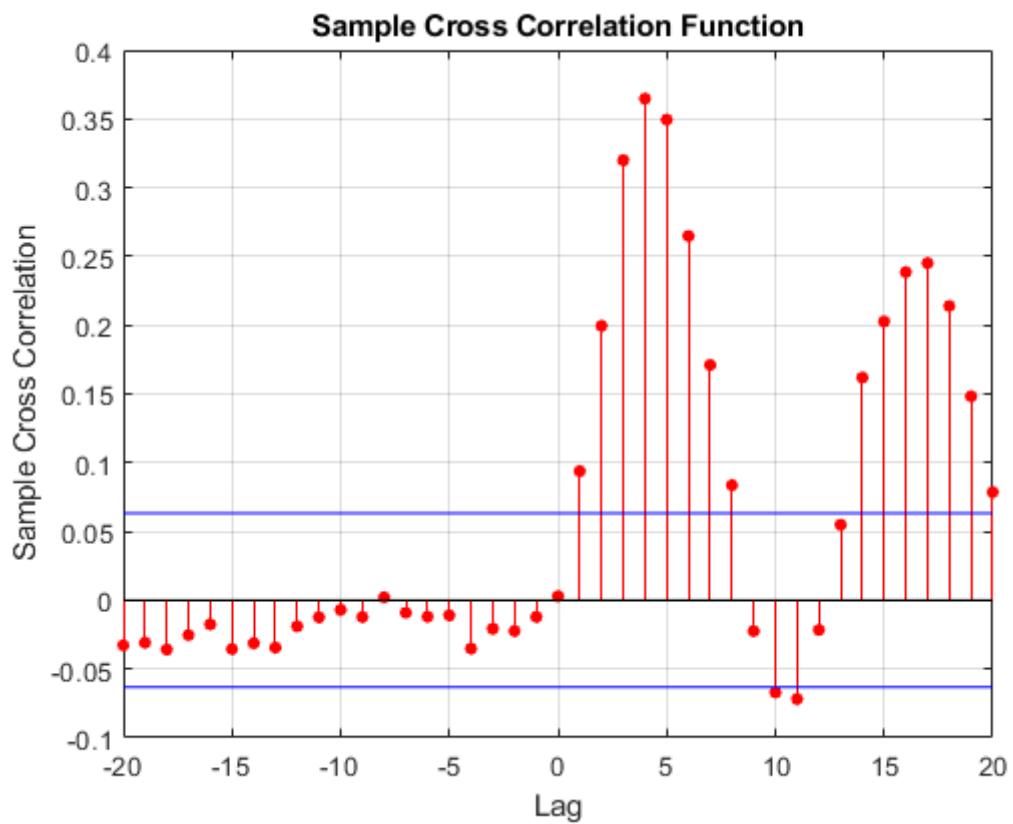
$$n_a = 12, n_b = 12, n_k = 1$$

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	70.1574	70.2233	70.0781	70.0659
13	0	0	0	0	0	0	0	0	0	70.1013	70.5636	70.6513	70.5431	70.5437
14	0	0	0	0	0	0	0	0	0	70.1046	70.6326	70.8009	70.6326	70.6346
15	0	0	0	0	0	0	0	0	0	70.0433	70.5684	70.8842	70.7129	70.7169

### Residue Correlation







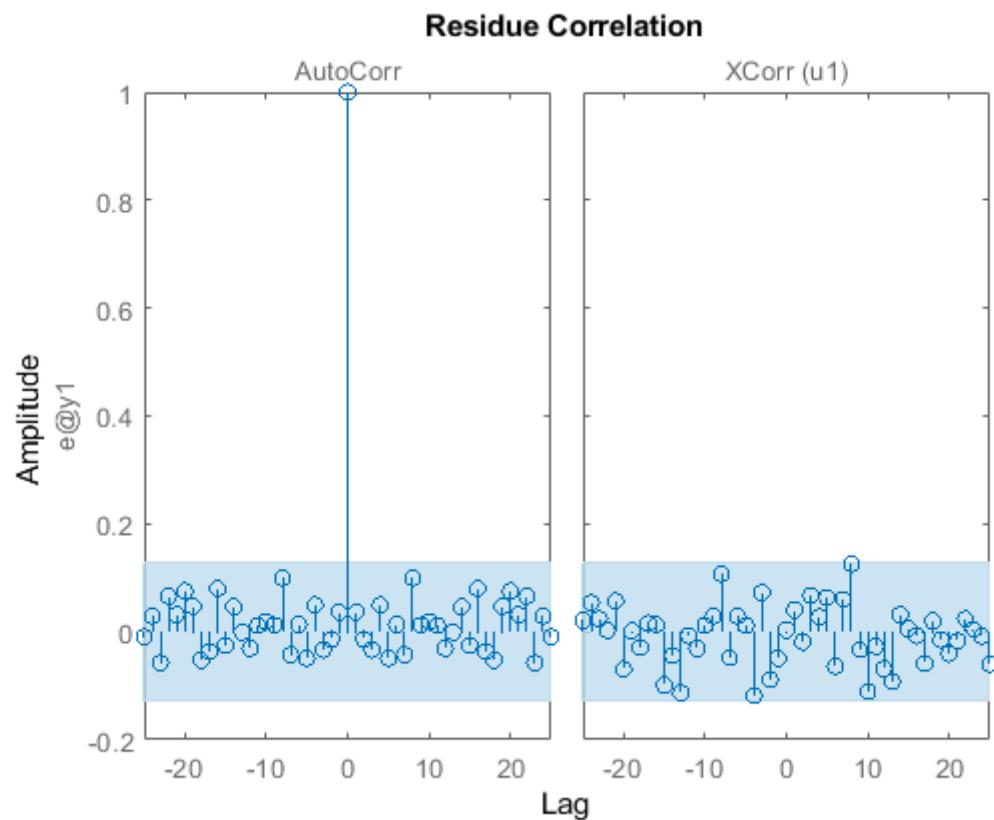
## ARMAX مدل

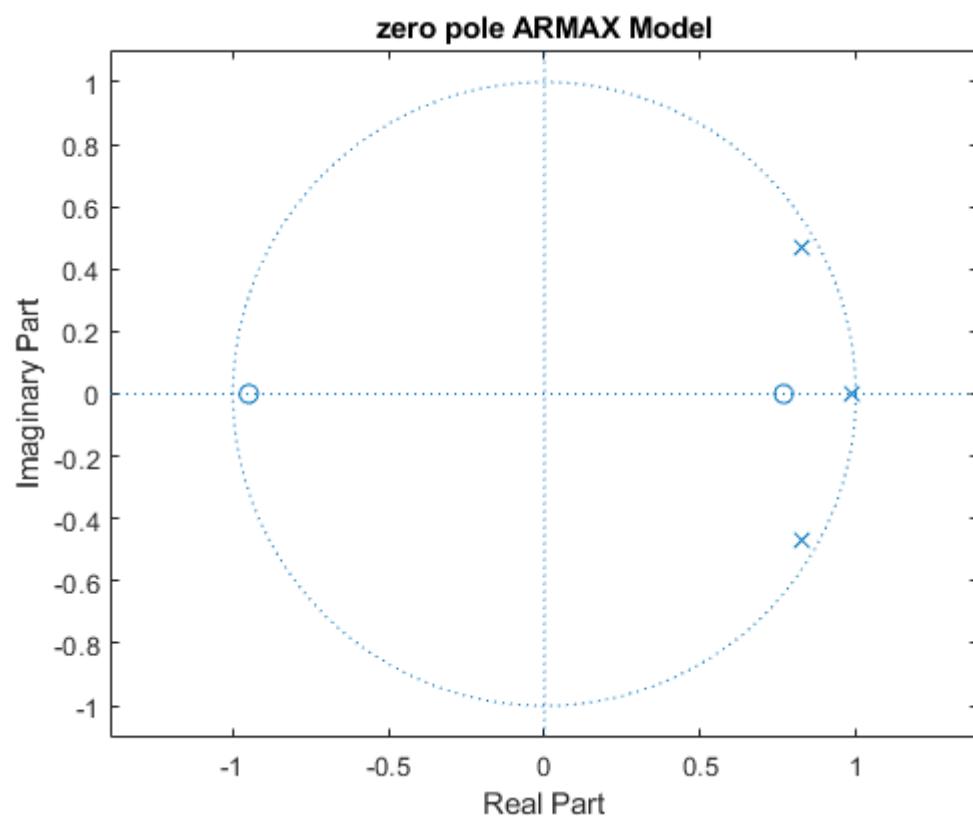
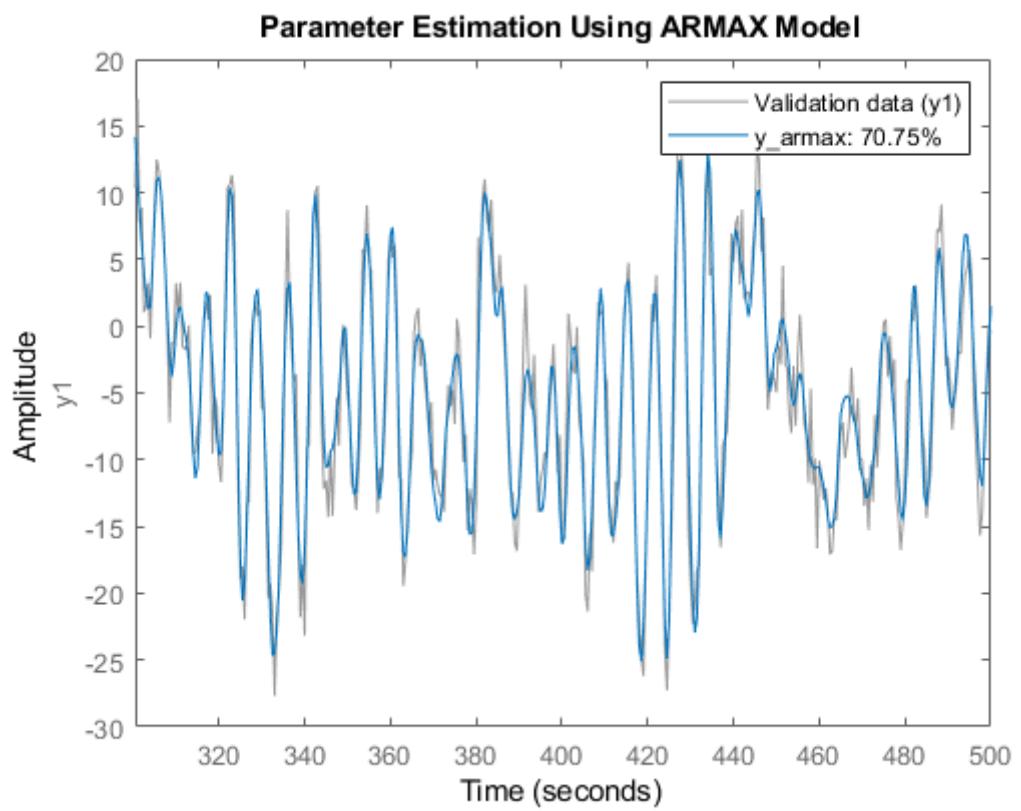
با توجه به این موضوع در این مسئله سه متغیر را در حلقه FOR قرار میدهیم پس ماتریس دقت ها  
ها از دو بعدی به سه بعدی تغییر پیدا میکند در اینجا مدلی که کمترین پارامتر ها را داشته باشد در نظر  
گرفته ایم

na=3;nb=3;nc=3;nk=1;

val(:,:,3) =

0	0	0	0	0
0	0	0	0	0
0	0	70.7464	70.9247	71.0709
0	70.0792	70.8526	70.9112	71.0566
0	70.6206	71.0390	71.1367	71.0275





## ARARX مدل

```
val (:,:,3) =
```

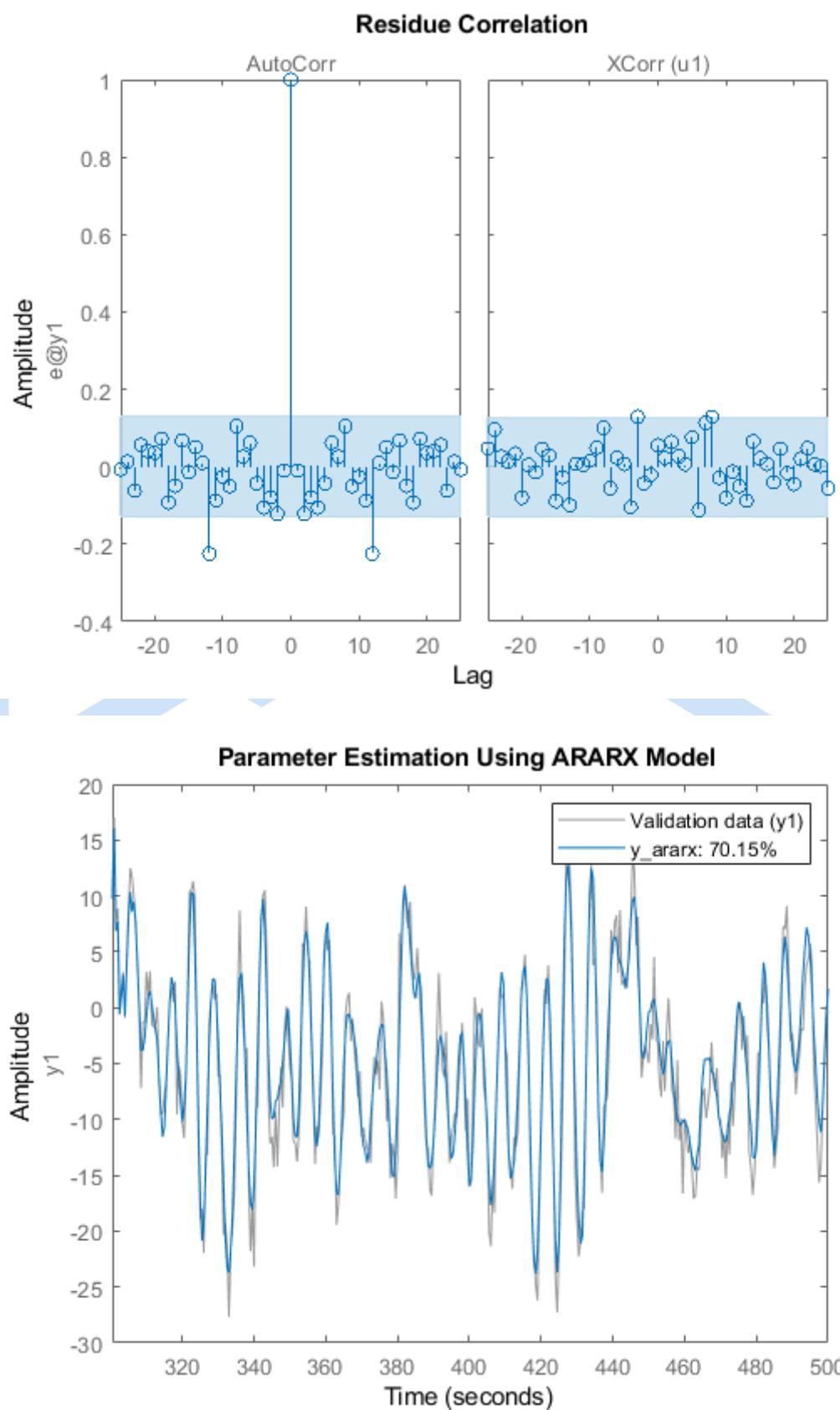
Columns 1 through 7

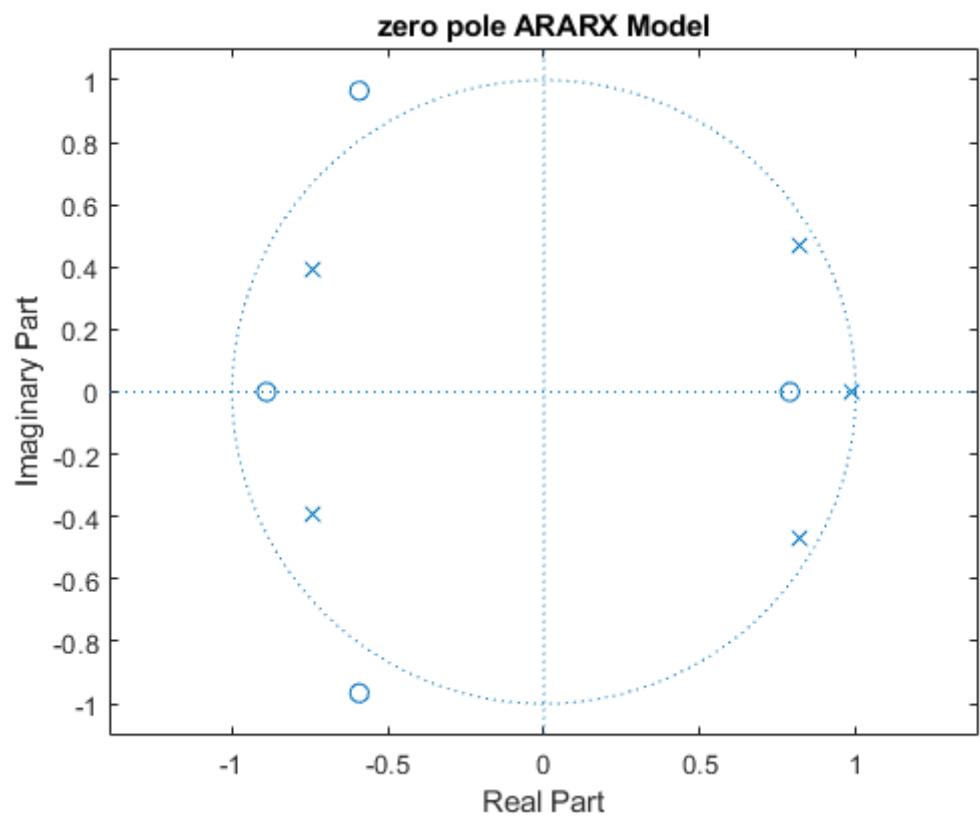
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Columns 8 through 10

0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	70.2145	70.4396
70.0411	70.6593	70.5915

```
%% parameter  
na=5;nb=5;nd=7;nk=1;
```

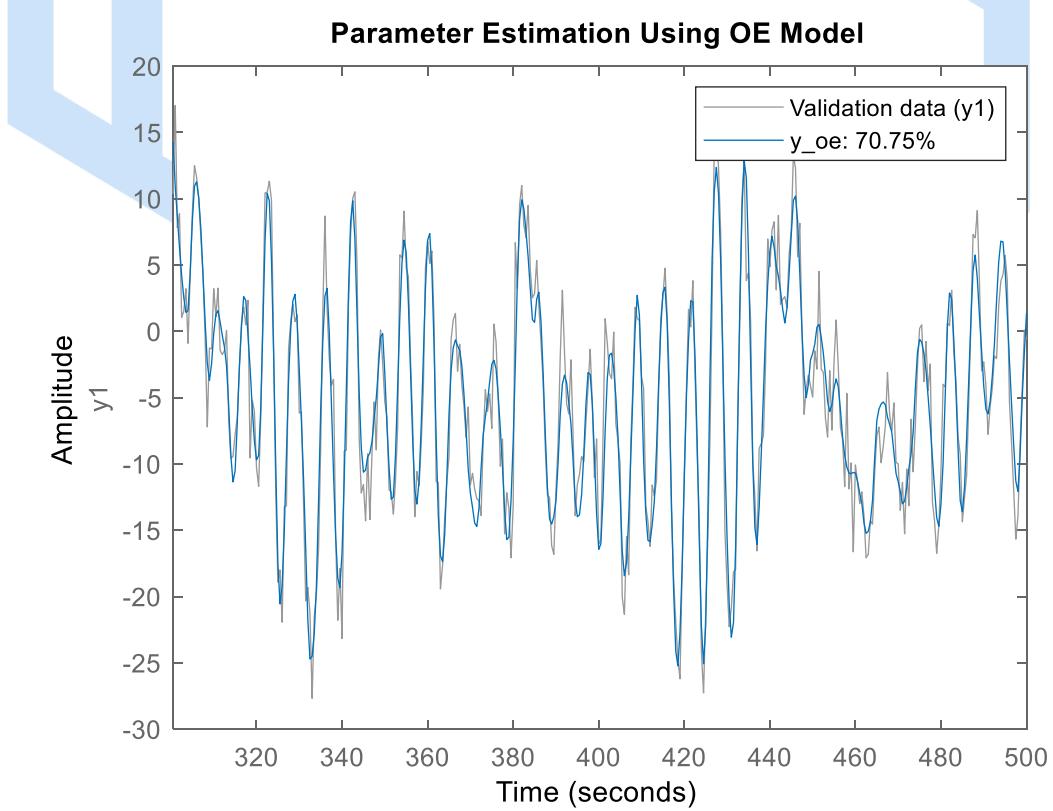
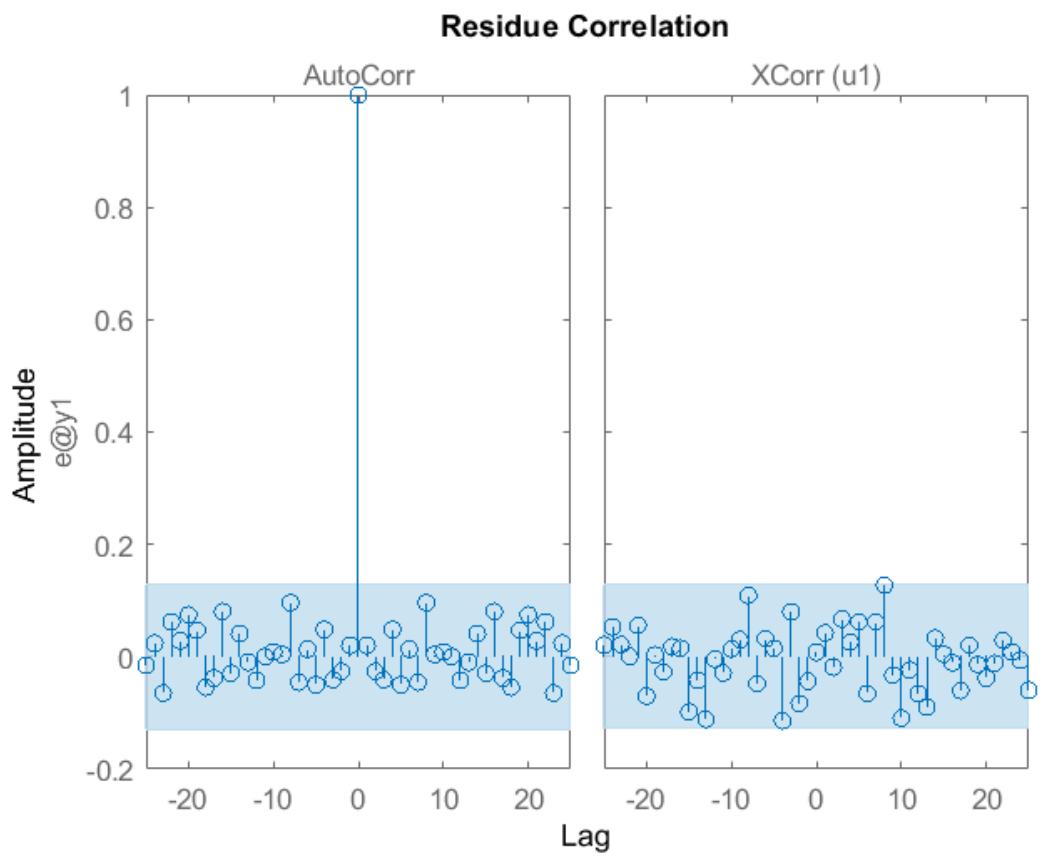




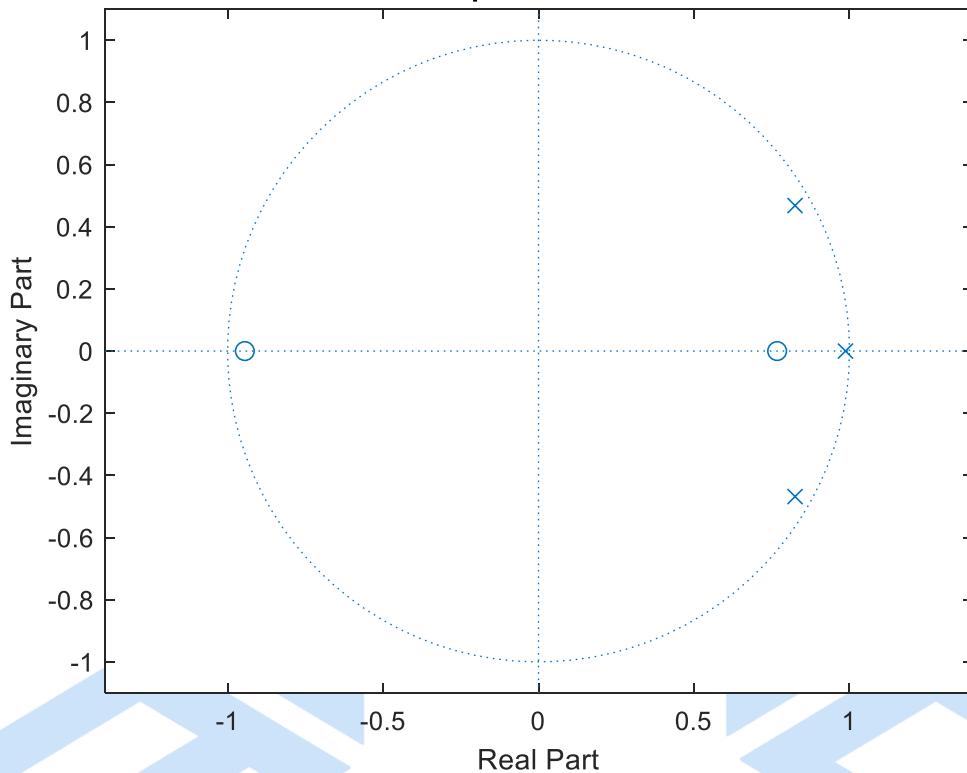
OE مدل

	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>
<b>1</b>	0	0	0	0	0
<b>2</b>	0	0	0	70.7799	70.9664
<b>3</b>	0	0	70.7493	0	0
<b>4</b>	0	0	70.9362	70.9404	0
<b>5</b>	0	0	71.0747	71.1377	71.3115
<b>6</b>					

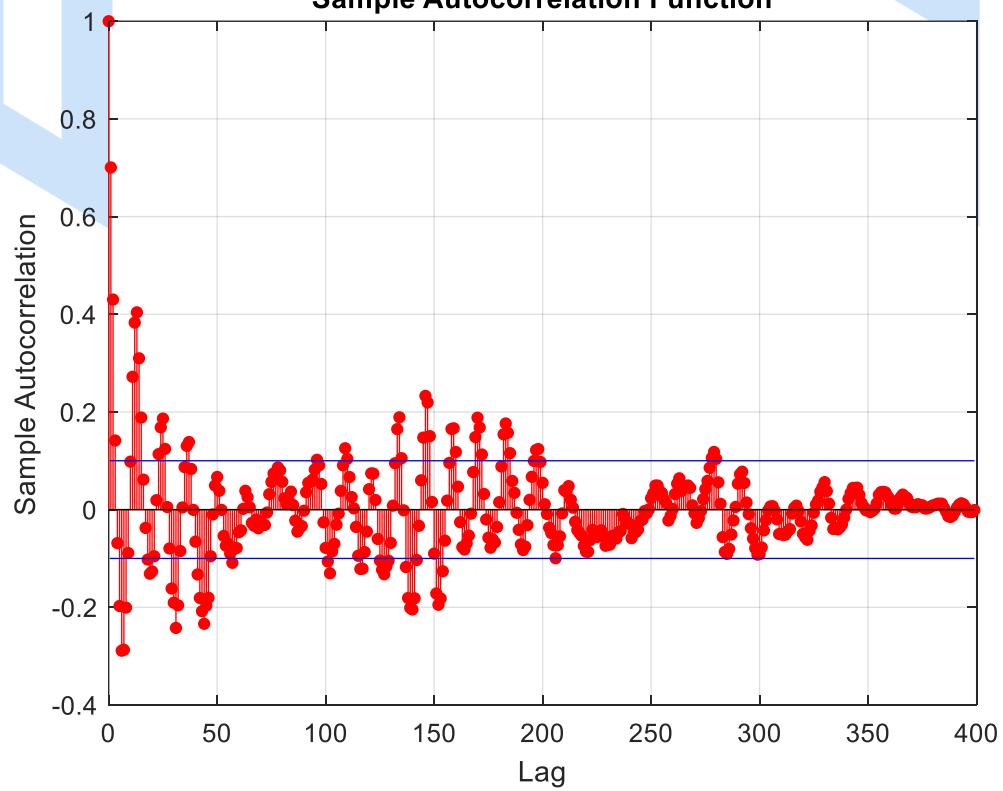
%% parameter  
nb=3; nf=3; nk=1;



**zero pole OE Model**



**Sample Autocorrelation Function**



## BJ مدل

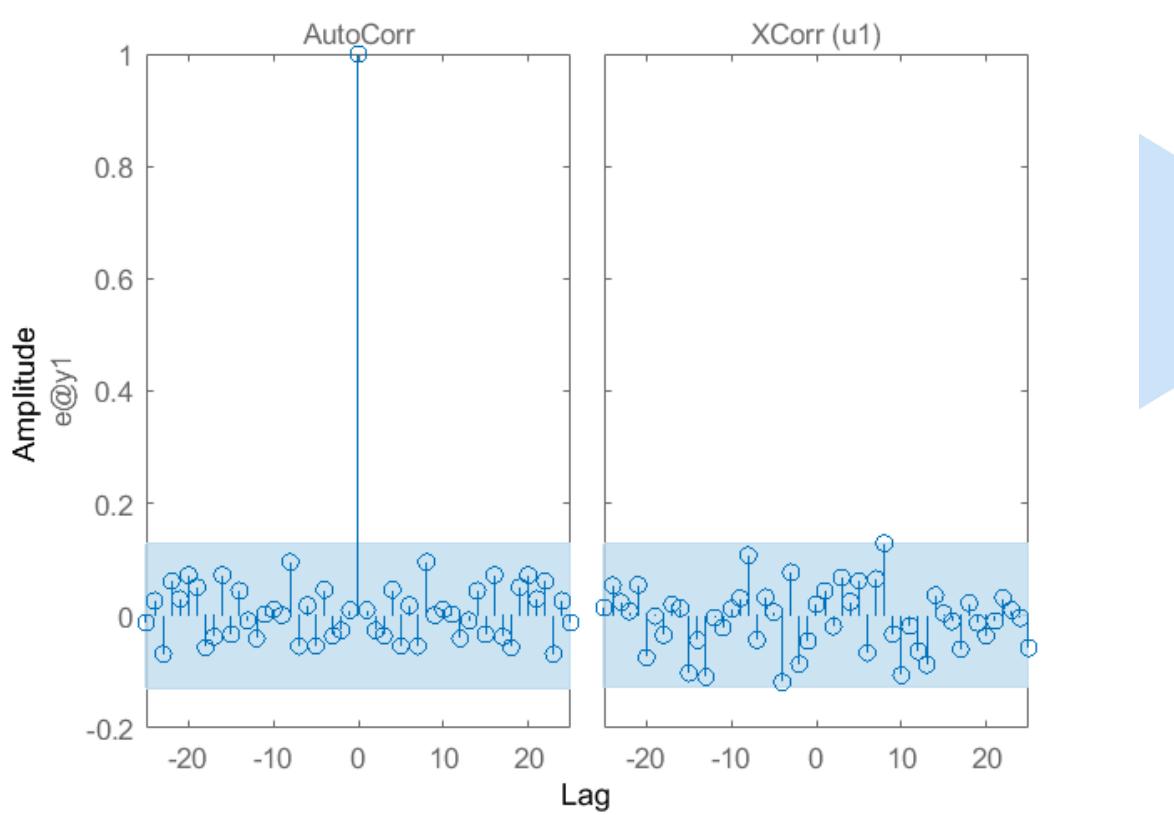
%% parameter

nb=3;nc=1;nd=0;nf=3;nk=1;

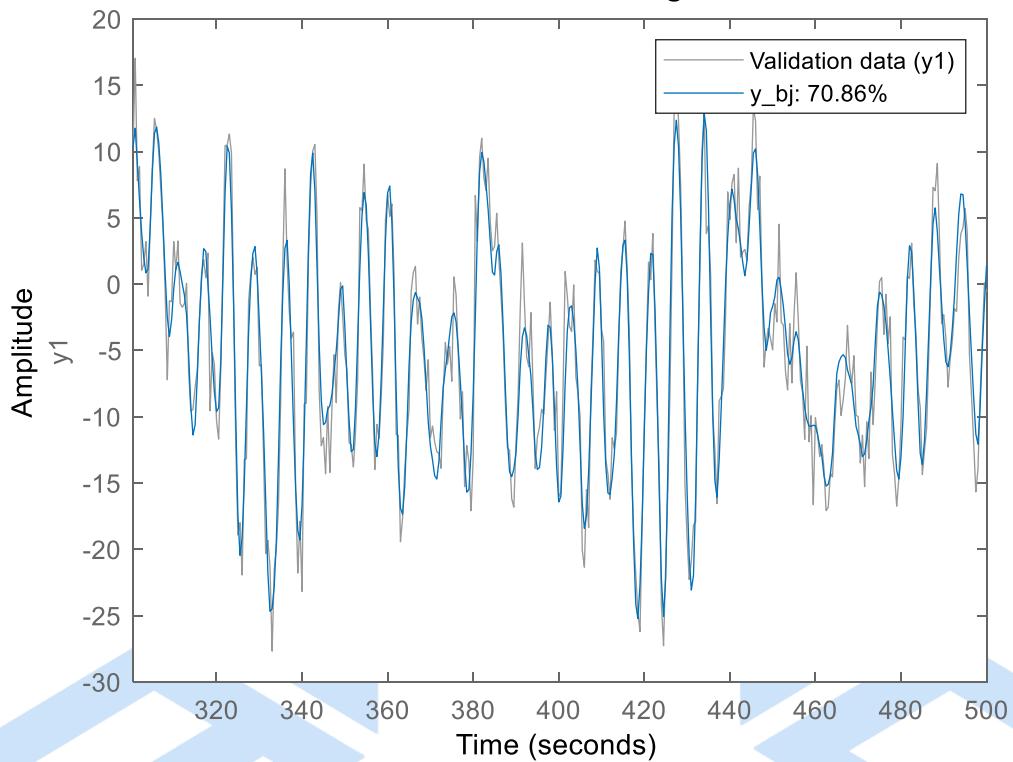
val(:,:,1,3) =

0	0	0	0	0
0	0	0	0	0
70.7496	71.0305	71.0700	71.1571	70.4897
71.0664	71.1086	71.1398	71.2284	71.2482
0	0	0	0	0

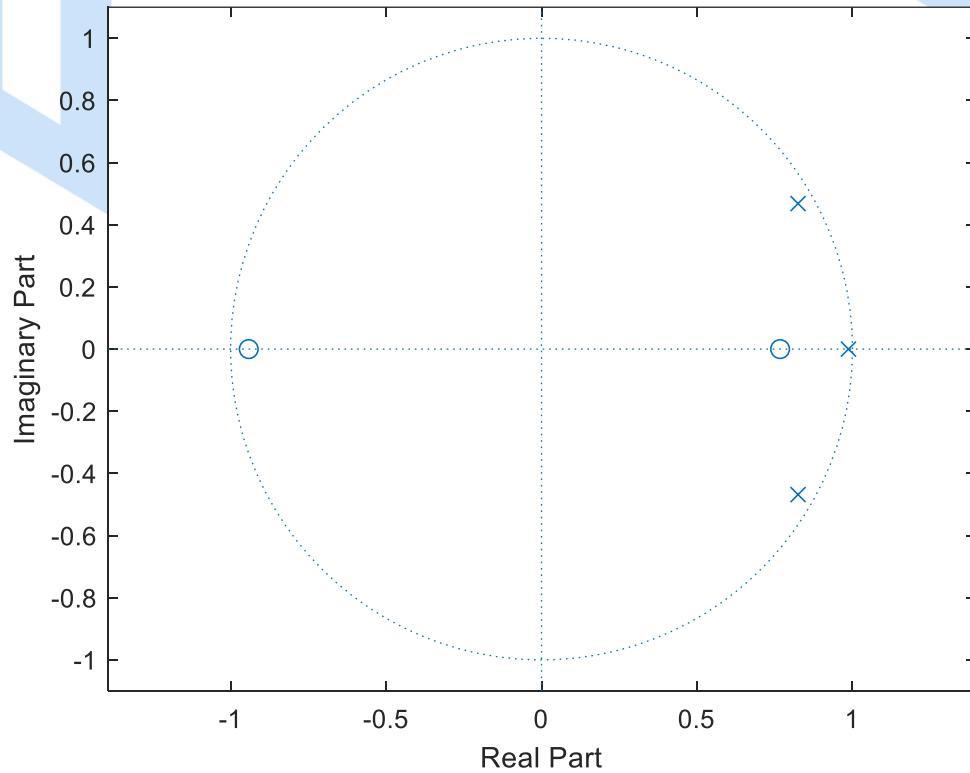
Residue Correlation



### Parameter Estimation Using BJ Model



### zero pole BJ Model



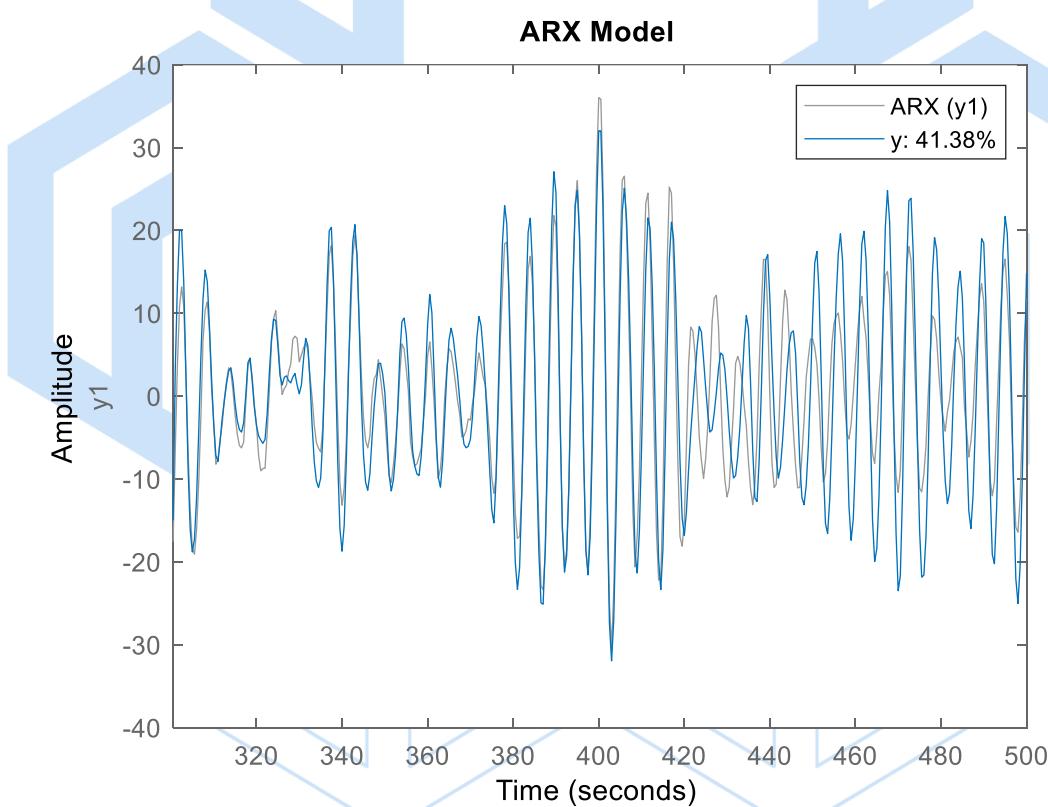
## سوال دو

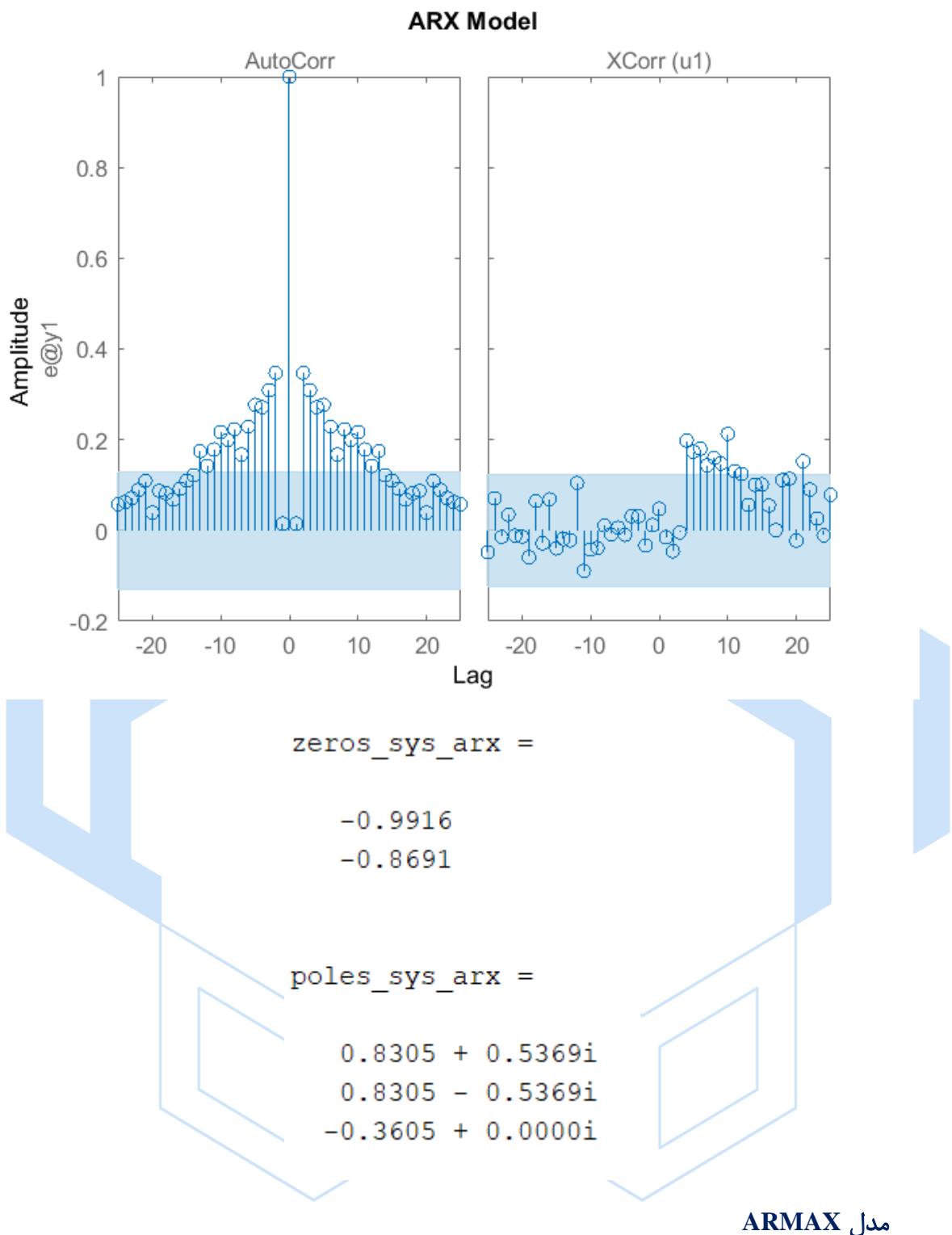
برای سیستم حلقه بسته ابتدا از مدل های ساده حلقه بسته برای شناسایی سیستم استفاده میکنیم  
پارامتر های زیر با استفاده از تکنیک سعی و خطأ که در بخش قبلی مراحل آن توضیح داده شده است  
بدست امده است

## ARX مدل

همانگونه که در نتایج قابل مشاهده است دقت مدل بدست امده خوب نمیباشد و نمودار های اتوکورولیشن و کراس کورولیشن مناسب نیست و دارای دینامیک های مدل نشده است.

```
%% parameter  
na=3; nb=3; nc=3; nk=1;  
Ts=0.5;
```

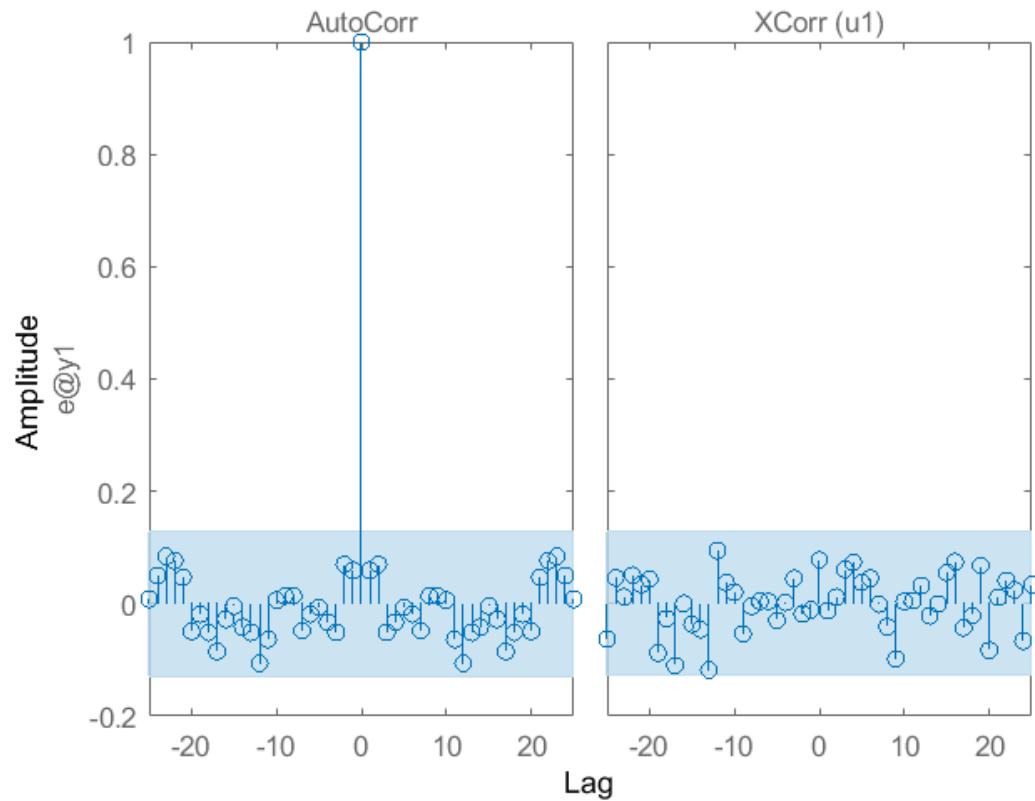




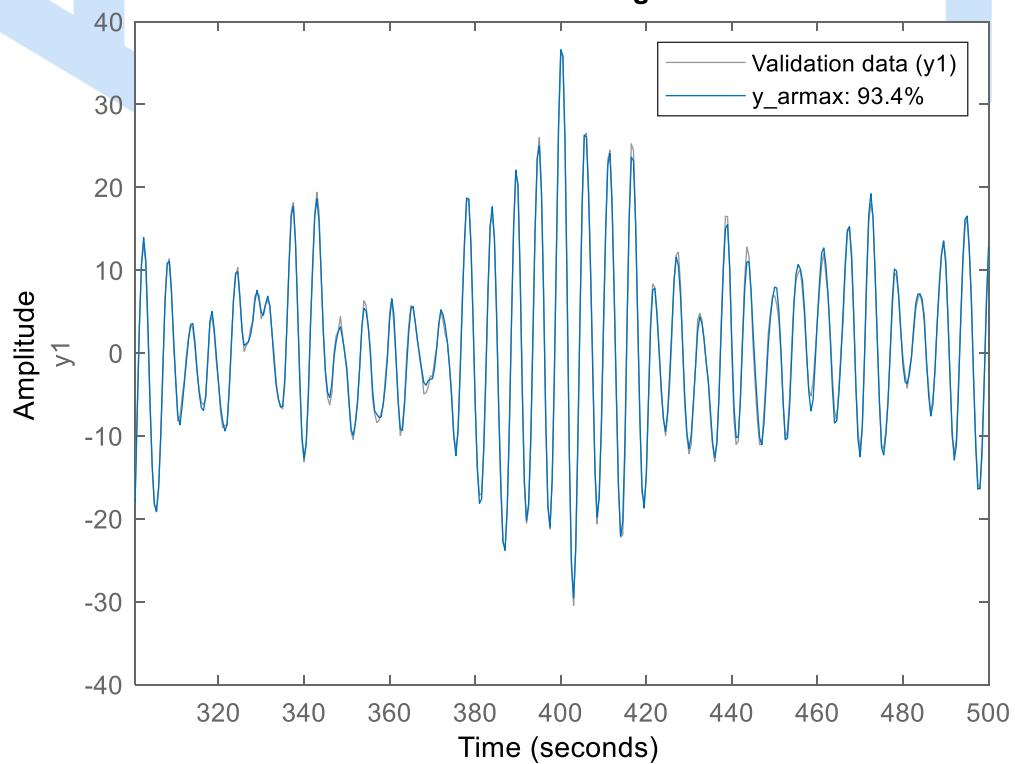
این مدل با پارامتر هایی مشابه با مدل ARX نتایج خیلی بهتری از خود نشان میدهد و با توجه به اصل سادگی این مدل را به عنوان مدل حلقه بسته در نظر میگیریم

```
%% parameter
na=3;nb=3;nc=3;nk=1;
Ts=0.5;
```

### Residue Correlation



### Parameter Estimation Using ARMAX Model



`zeros_sys_armax =`

`-1.0707`

`0.7730`

`poles_sys_armax =`

`0.8325 + 0.5482i`

`0.8325 - 0.5482i`

`0.9300 + 0.0000i`

حال که مدل حلقه بسته بدست امد به محاسبه پارامتر  $K$  میپردازیم

در بخش قبلی بهترین مدل بدست امده برای سیستم حلقه باز از نظر دقیق و سادگی مدل OE است  
که با توجه به قطب ها و صفر های بدست امده این مدل را میتوان به صورت زیر نوشت

$$\sum = \frac{B(z^{-1})}{F(z^{-1})}$$

$$B(z^{-1}) = 0.6661z^{-1} + 0.118z^{-2} - 0.4827z^{-3}$$

$$F(z^{-1}) = 1 - 2.637z^{-1} + 2.528z^{-2} - 0.888z^{-3}$$

با توجه به ساده سازی دیاگرام های بلوكی تابع تبدیل حلقه بسته به شکل

$$G_{cl} = \frac{\Sigma}{1 + K\Sigma}$$

اگر مقدار مدل شناسایی شده را در عبارت بالا جایگزاري کنیم داریم

$$G_{cl} = \frac{B}{F + KB}$$

با جایگزاري مقادير  $B, F$  در عبارت بالا داریم

$$G_{cl} = \frac{0.6661z^{-1} + 0.118z^{-2} - 0.4827z^{-3}}{1 - 2.637z^{-1} + 2.528z^{-2} - 0.888z^{-3} + K(0.6661z^{-1} + 0.118z^{-2} - 0.4827z^{-3})}$$

مشاهده شد که بهترین تابع تبدیل حلقه بسته سیستم با استفاده از مدل ARMAX بدست امده بود که با توجه به صفرها و قطب‌های آن مدل آن به شکل زیر است

$$Y(t) = \frac{B'(z^{(-1)})}{A'(z^{(-1)})} \times U(t) + \frac{C'(z^{(-1)})}{A'(z^{(-1)})} \times n(t)$$

این عبارت برابر با تابع تبدیل خروجی سیستم است که از جمع اثارات ورودی و نویز بدست امده است

تابع تبدیل ورودی به خروجی سیستم حلقه بسته برابر است با

$$Y(t) = \frac{B'(z^{(-1)})}{A'(z^{(-1)})} \times U(t)$$

$$Y(t) \simeq \frac{0.6315z^{(-1)} + 0.188z^{(-2)} - 0.5226z^{(-3)}}{1 - 2.595z^{(-1)} + 2.542z^{(-2)} - 0.924z^{(-3)}} U(t)$$

حال تابع تبدیل حلقه بسته بدست امده از روش ARMAX را با تابع تبدیل بدست امده از روش جبر بلوکی را با یک دیگر مساوی قرار میدهیم

$$Y = \frac{B}{F + KB} = \frac{B'}{A'} = \frac{0.6315z^{-1} + 0.188z^{-2} - 0.5226z^{-3}}{1 - 2.595z^{-1} + 2.542z^{-2} - 0.924z^{-3}}$$

با تقریب نسبتاً قابل قبولی دو عبارت  $B' = B$  با یک دیگر برابر هستند پس از انجایی که صورت دو کسر بالا با یکدیگر برابر هستند پس باید مخرج‌های انها با یک دیگر برابر باشند

$$F + KB = A' \Rightarrow K = \frac{A' - F}{B} = \frac{0.042z^{(-1)} + 0.014z^{(-2)} - 0.036z^{(-3)}}{0.6661z^{(-1)} + 0.118z^{(-2)} - 0.4827z^{(-3)}}$$

از جملاتی که مرتبه توان انها کمتر است میتوانیم با تقریب صرف نظر کنیم و تنها جمله‌ای با بزرگترین توان را در نظر میگیریم

پس داریم

$$K = \frac{0.042z^{(-1)}}{0.6661z^{(-1)}} = 0.063$$

### قسمت سوم

در درس کنترل صنعتی تلاش میشود تا مدل های مرتبه بالا را با مدل های مرتبه یک و دو تخمین زد با استفاده از این کار یک مدل ساده از سیستم در دست داریم این مدل ممکن است از دقت بالایی از نظر شناسایی سیستم برخوردار نباشد اما برای اهداف کنترلی مدل مناسبی است و به راحتی میتوانیم کنترل کننده های PID برای این مدل ها تعیین کنیم.

سیستم های مرتبه اول بدون تاخیر یا مدل دو پارامتری

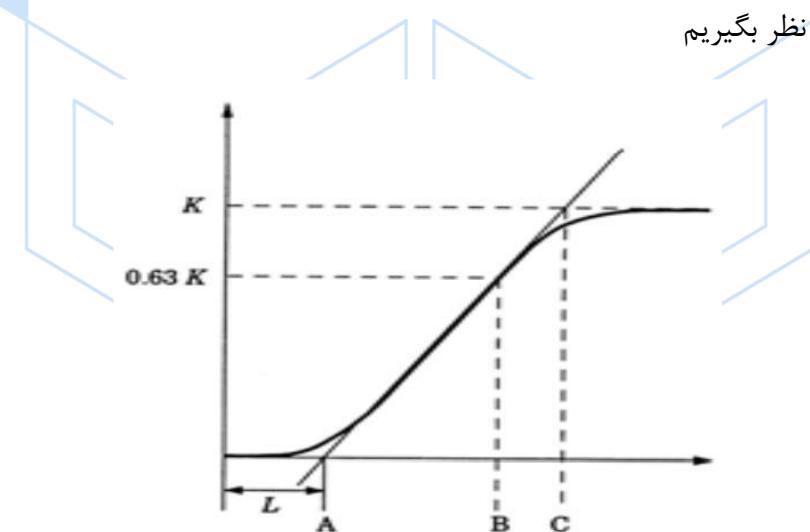
$$G(s) = \frac{K}{1 + T_{ar}s}$$

در این مدل مقدار  $K$  برابر است با مقدار نهایی سیستم به ازای ورودی پله برای محاسبه این مقدار به سیستم یک پاسخ پله میدهیم و خروجی آن را در حالت نهایی بدست میاوریم یا اگرتابع تبدیل سیستم در دسترس باشد مقدار تابع تبدیل به ازای  $S$  برابر با صفر را محاسبه میکنیم تا مقدار  $K$  بدست بیاید برای محاسبه  $T_{ar}$  چندین روش وجود دارد به طور مثال میتوانیم مساحت بین نمودار و مقدار نهایی را حساب کنیم و مقدار بدست امده را تقسیم بر  $K$  کنیم فرمول هایی این روش به صورت زیر است

$$T_{ar} = \frac{A_0}{K} \quad A_0 = \int_0^{\infty} (S(\infty) - S(t)) dt = \int_0^{\infty} (K - S(t)) dt$$

راه دیگر بدست اوردن  $T_{ar}$  این است که خطی که شیب ماکزیمم بر نمودار دارد را رسم کنیم و محل تقاطع این خط و خط  $K$  را در نظر بگیریم و زمانی که این دو خط هم دیگر را قطع کرده اند نقطه  $C$  را

به عنوان  $T_{ar}$  در نظر بگیریم



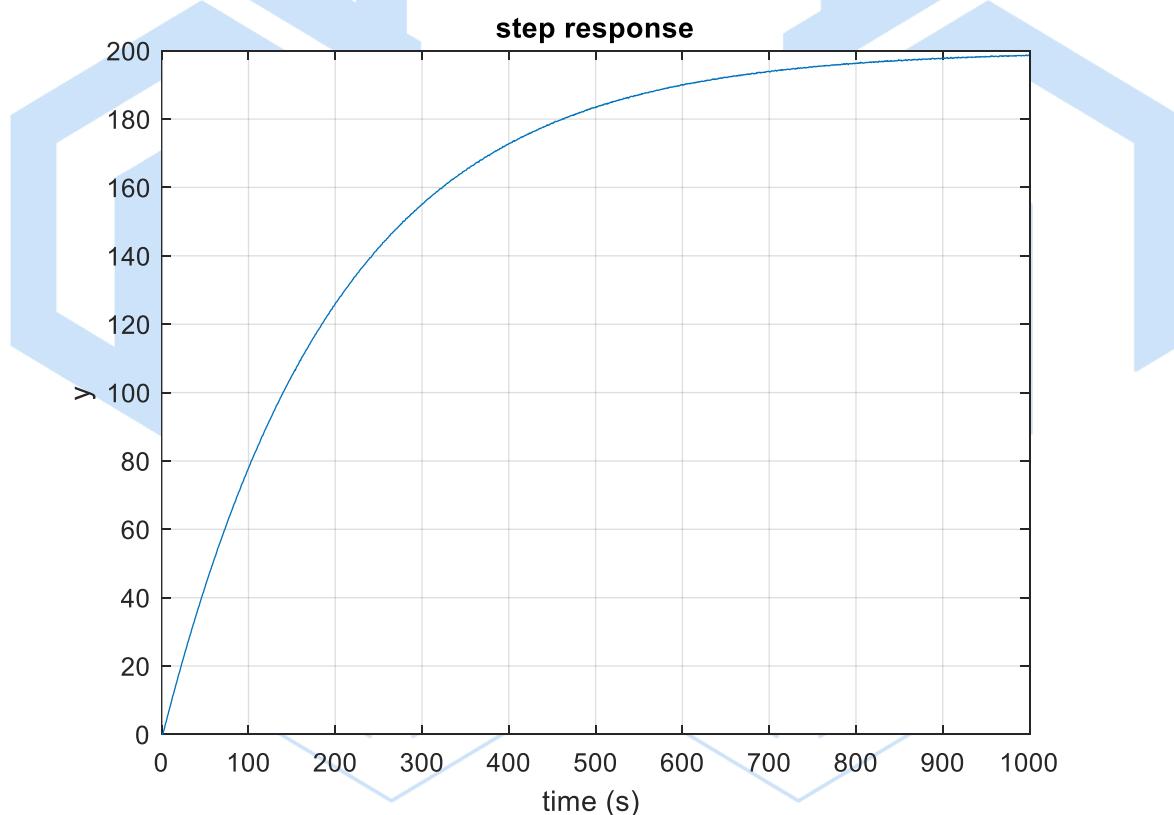
روش دیگر این است که ببینیم در چه زمانی سیستم به 63 درصد مقدار نهایی خود میرسد نقطه  $B$  و این زمان را به عنوان مقدار  $T_{ar}$  در نظر بگیریم

اگر سیستم مرتبه اول را همراه با تاخیر در نظر بگیریم برای محاسبه تاخیر باید خطی مماس با شیب حداقل بر نمودار رسم کنیم و محل برخورد این خط و محور افقی نقطه A را برابر با مقدار تاخیر در نظر بگیریم.

مورد پاسخ فرکانسی و دیاگرام های نایکوئیست و بود نیز کافی است نقطهنهایی یا Ultimate point تعیین شود. نقطهنهایی عبارت است از پاسخ فرکانسی سیستم در کمترین فرکانسی که فاز -180 درجه گردد. در نمودار بود هنگامی که دامنه 20 db شکسته شود، قطب و صفرها را پیدا می کنیم. اگر نمودار نایکوئیست به صورت دایره ایی ادامه دار باشد، وجود تاخیر را نشان می دهد.

### قسمت ب

در ابتدا به سیستم ناشناخته پاسخ پله ای اعمال میکنیم و با استفاده از پاسخ بدست امده به شناسایی سیستم میپردازیم

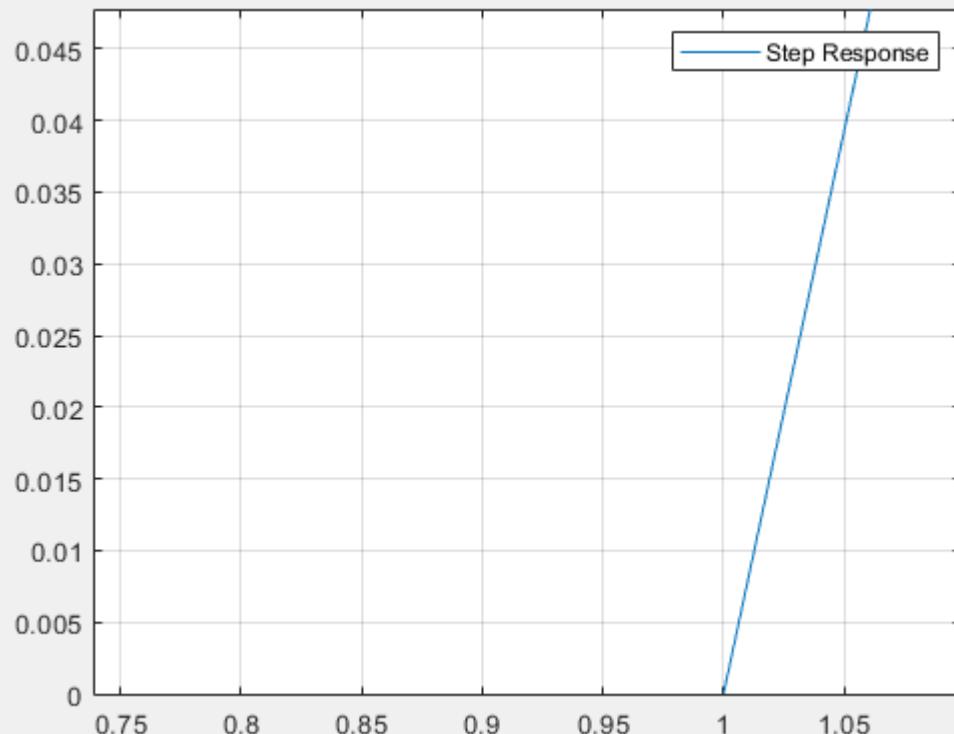


با توجه به اینکه سیستم دارای نوسان و فراجهش نیست می توان فهمید که سیستم حالت فوق میرا دارد و میتوان با مدل های مرتبه اول آن را مدل سازی کرد.

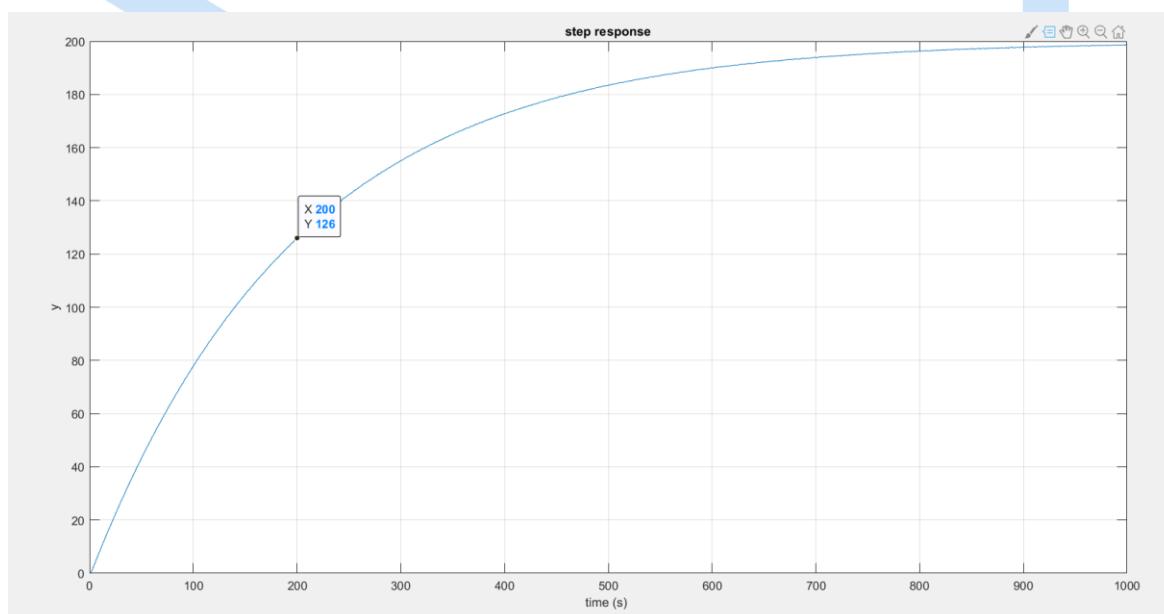
بهره  $k$  سیستم برابر است با نسبت حالت ماندگار خروجی به ورودی که در اینجا با توجه به اینکه ورودی پله است به شکل زیر بدست می آید.

$$k = \frac{200}{1} = 200$$

با توجه به شبیه خط مماس بر نمودار پاسخ پله مقدار تاخیر برابر با یک میباشد.



همچنین با توجه به نمودار  $T_{ar}$  سیستم برابر است با 200



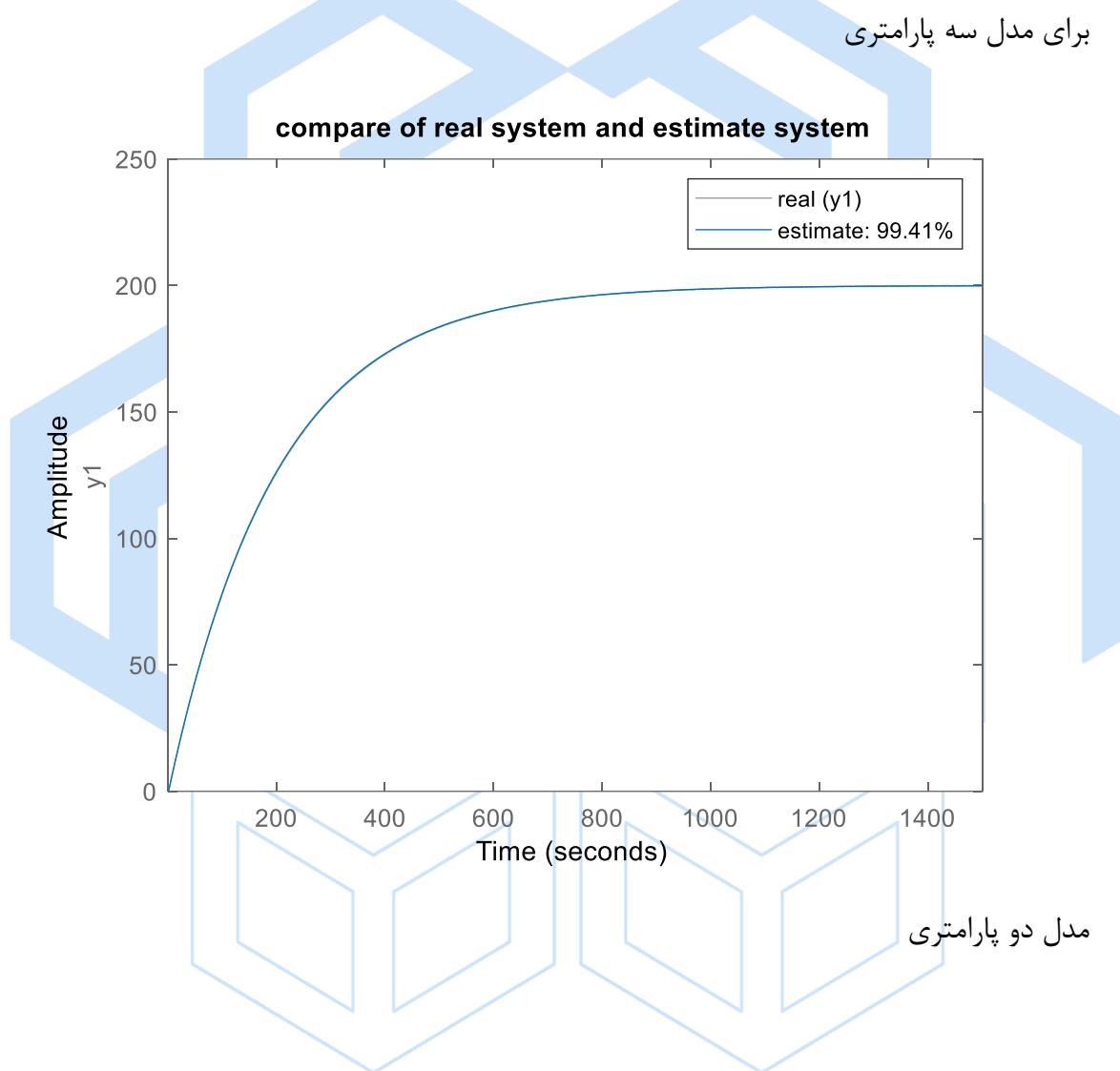
پس ثابت زمانی سیستم برابر با 199 میباشد

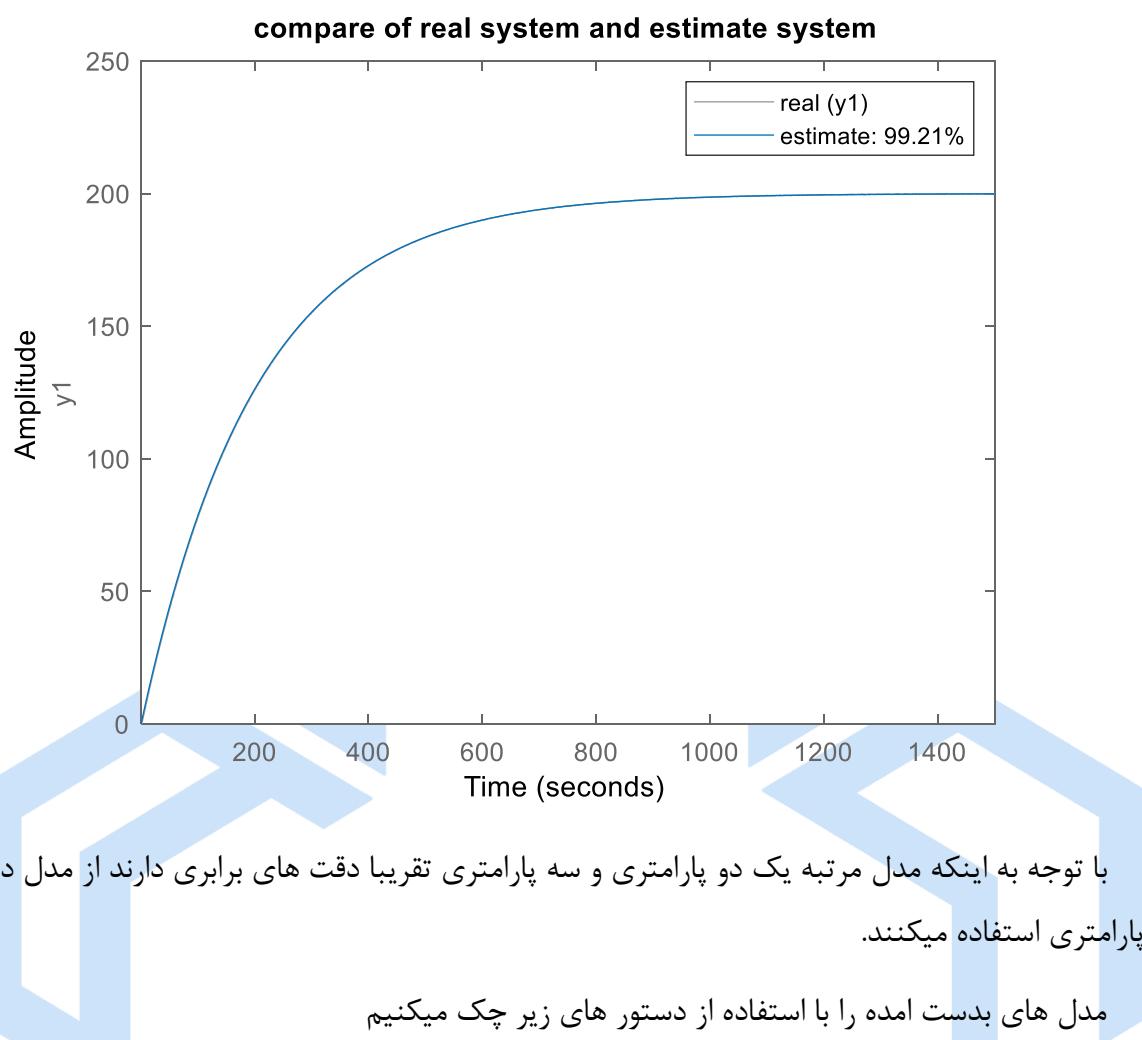
مدل های سه پارامتری و دو پارامتری که میتوان برای سیستم شناسایی کرد به شرح زیر است

$$G = \frac{200}{1 + 200s}$$

$$G = \frac{200 e^{-s}}{1 + 199s}$$

حال اگر دقیقیت مدل های بدست آمده را با استفاده از دستور compare محاسبه کنیم داریم





```
p1d=procest(data3, 'P1D')
p1=procest(data3, 'P1')
```

```

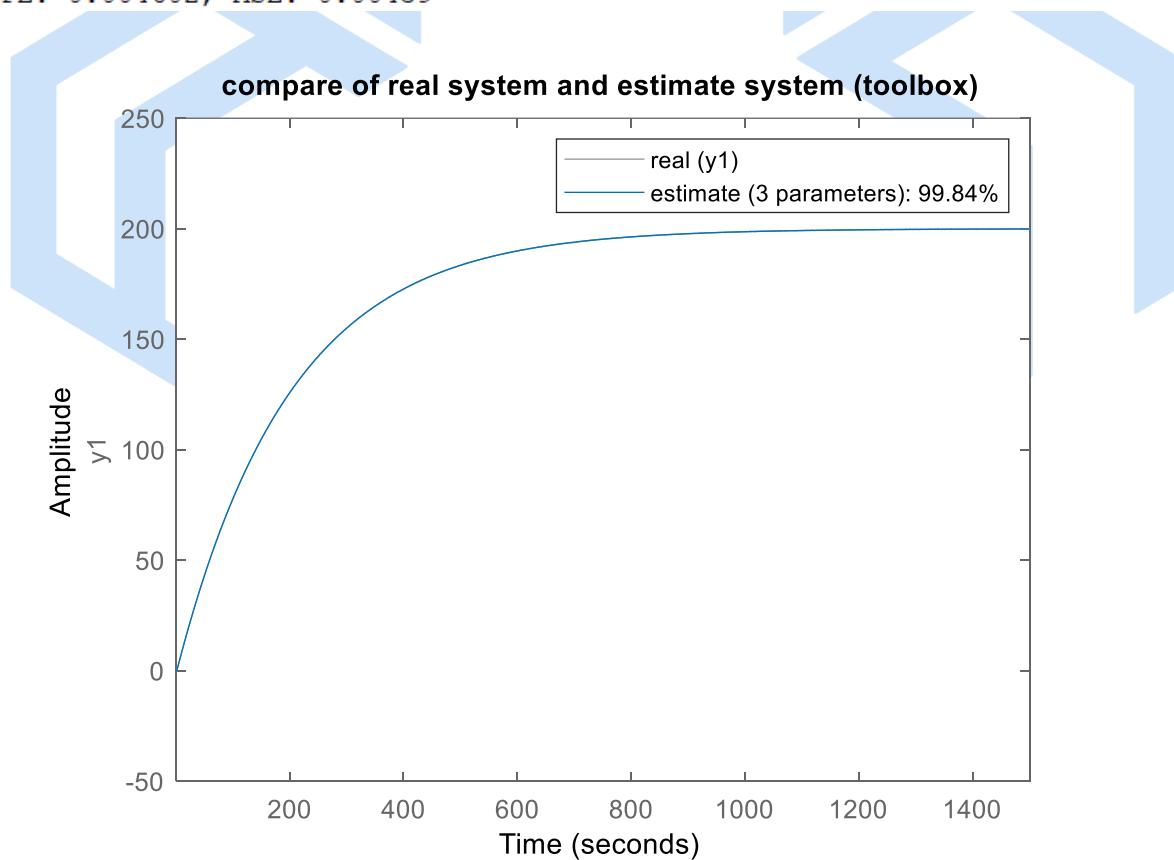
pld =
Process model with transfer function:
    Kp
G(s) = ----- * exp(-Td*s)
        1+Tp1*s

    Kp = 200
    Tp1 = 200.01
    Td = 1.324

Parameterization:
'P1D'
Number of free coefficients: 3
Use "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using PROCEST on time domain data "data3".
Fit to estimation data: 99.85%
FPE: 0.004602, MSE: 0.00459

```



```

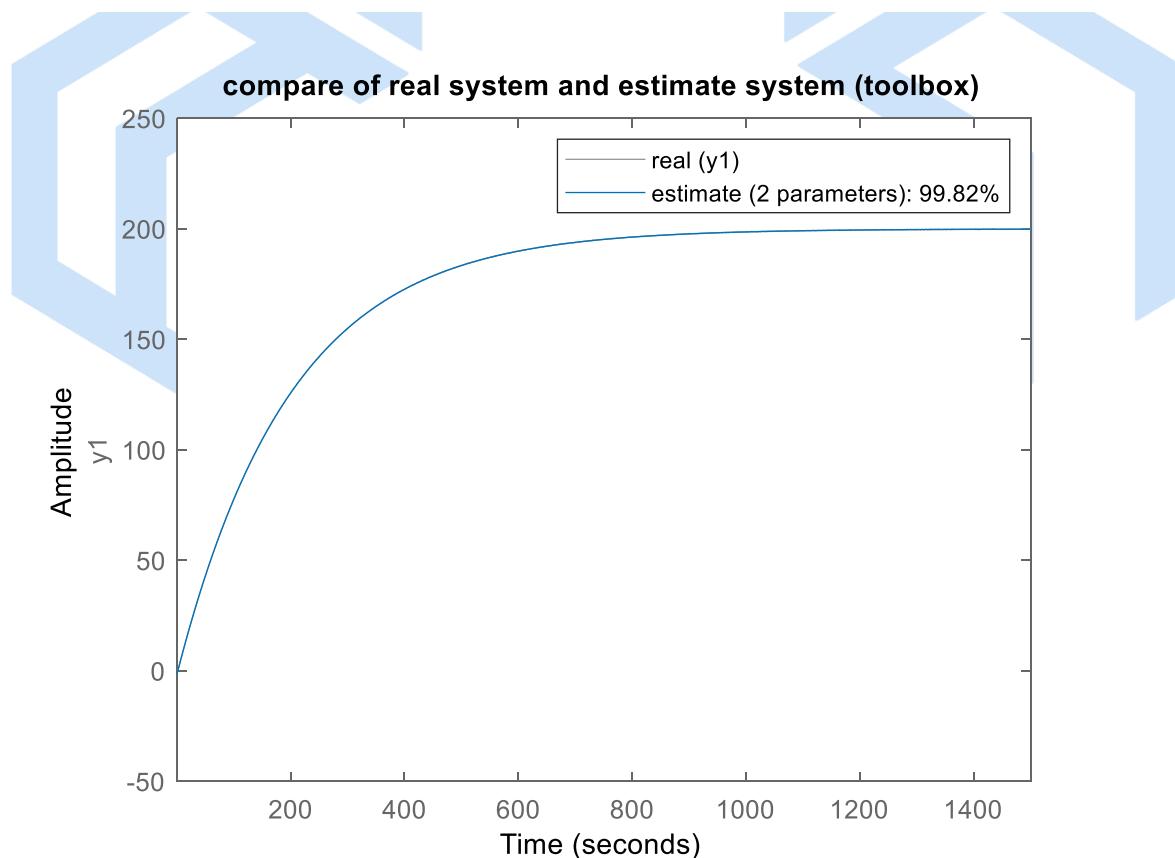
p1 =
Process model with transfer function:
    Kp
G(s) = -----
           1+Tp1*s

    Kp = 200
    Tp1 = 200.04

Parameterization:
'P1'
Number of free coefficients: 2
Use "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using PROCEST on time domain data "data3".
Fit to estimation data: 99.83%
FPE: 0.005429, MSE: 0.005418

```

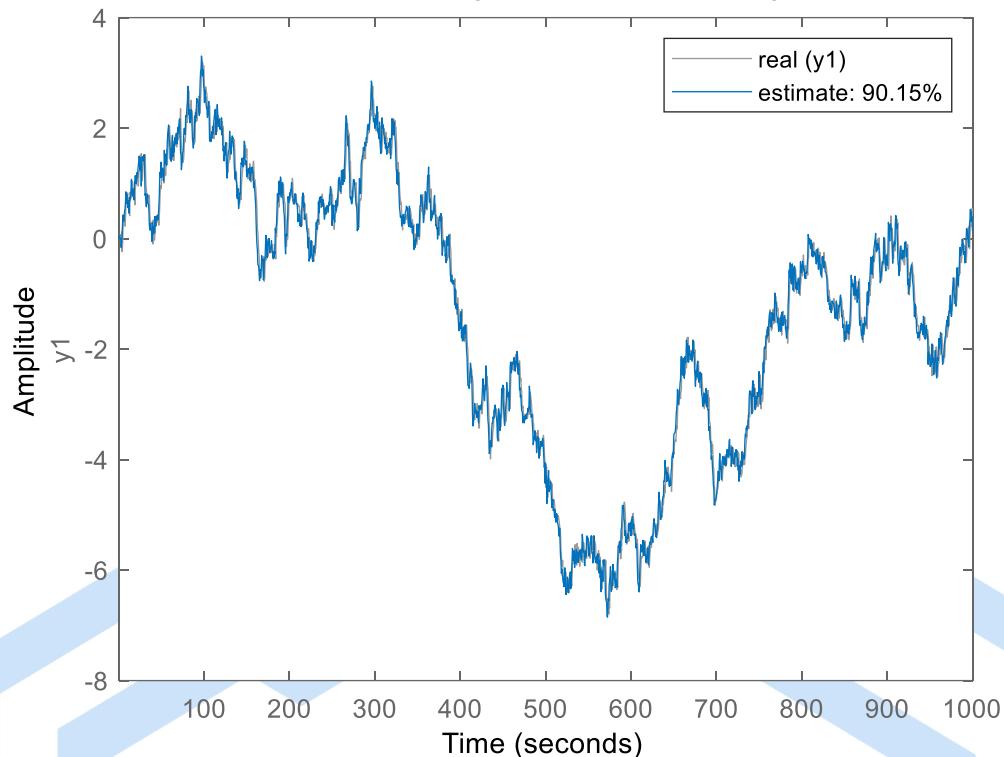


به ازای ورودی رندوم با میانگین صفر و واریانس 0.1 سیستم دقت سیستم تخمین زده شده برابر است

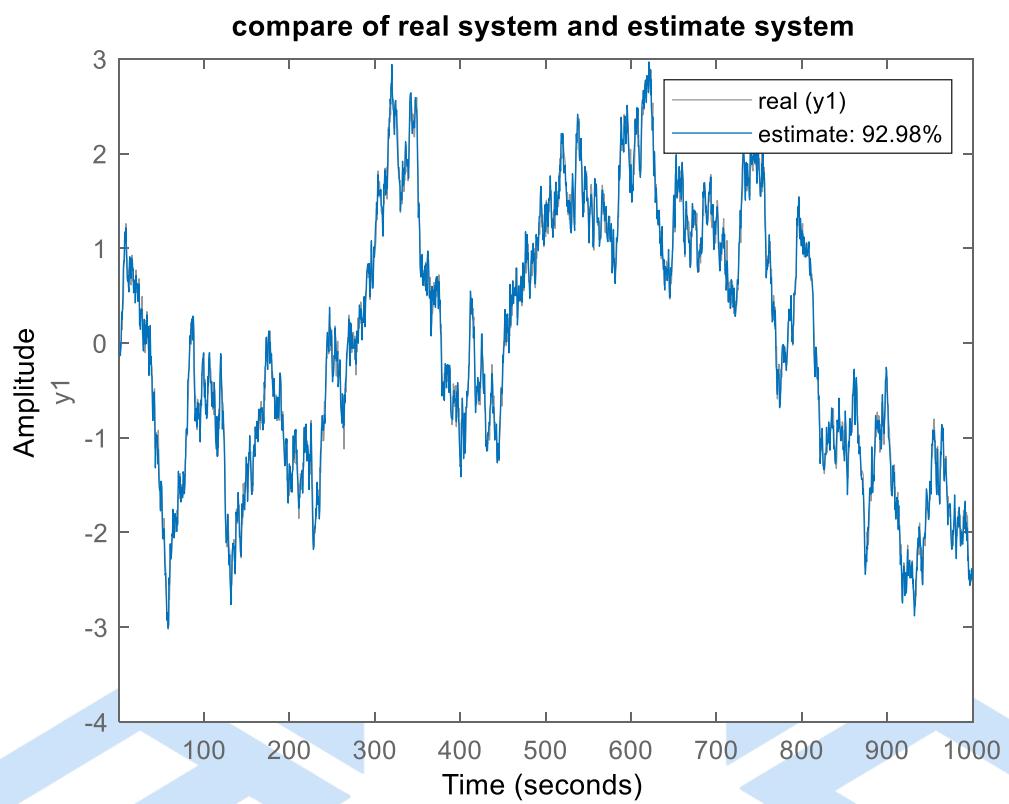
با

سیستم دو پارامتری

compare of real system and estimate system



سیستم سه پارامتری



### سوال سه

```
val =
```

```
Time domain data set with 300 samples.  
Sample time: 0.1 seconds
```

Outputs	Unit (if specified)
y1	

Inputs	Unit (if specified)
u1	

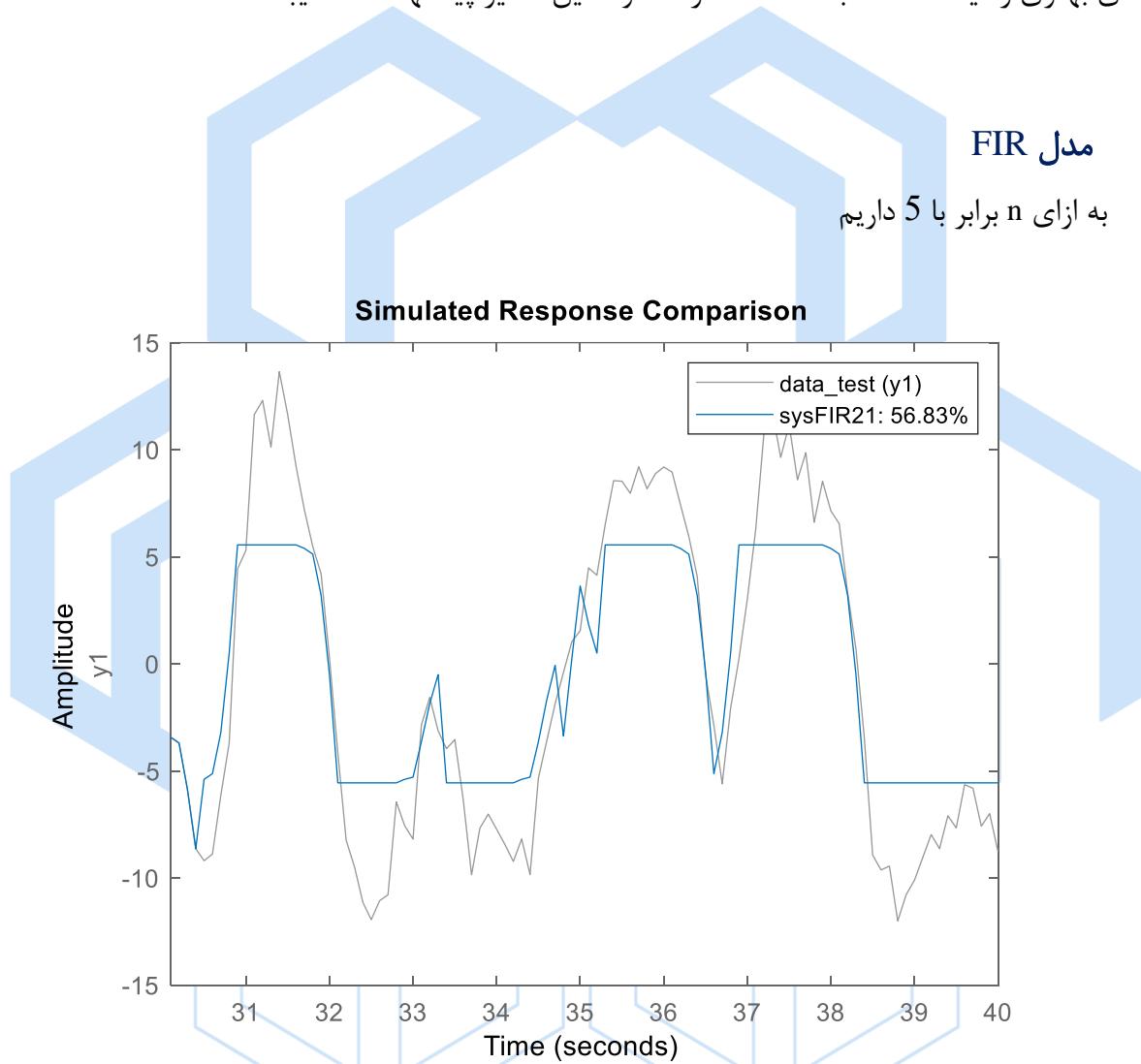
زمان نمونه برداری در دیتاست داده شده برابر با 0.1 میباشد.

به دلیل طولانی نشدن گزارش همه مراحل طراحی را نیاورده ایم و فقط در ابتدای حدسه اولیه در مورد هر یک از پارامترها در مدل مربوطه را تست میکنیم و سپس با استفاده از قوانین سرانگشتی و تغییر پارامترها به بهبود مدل میپردازیم

در ابتداء مقدار تاخیر سیستم را با استفاده از دستور زیر محاسبه میکنیم

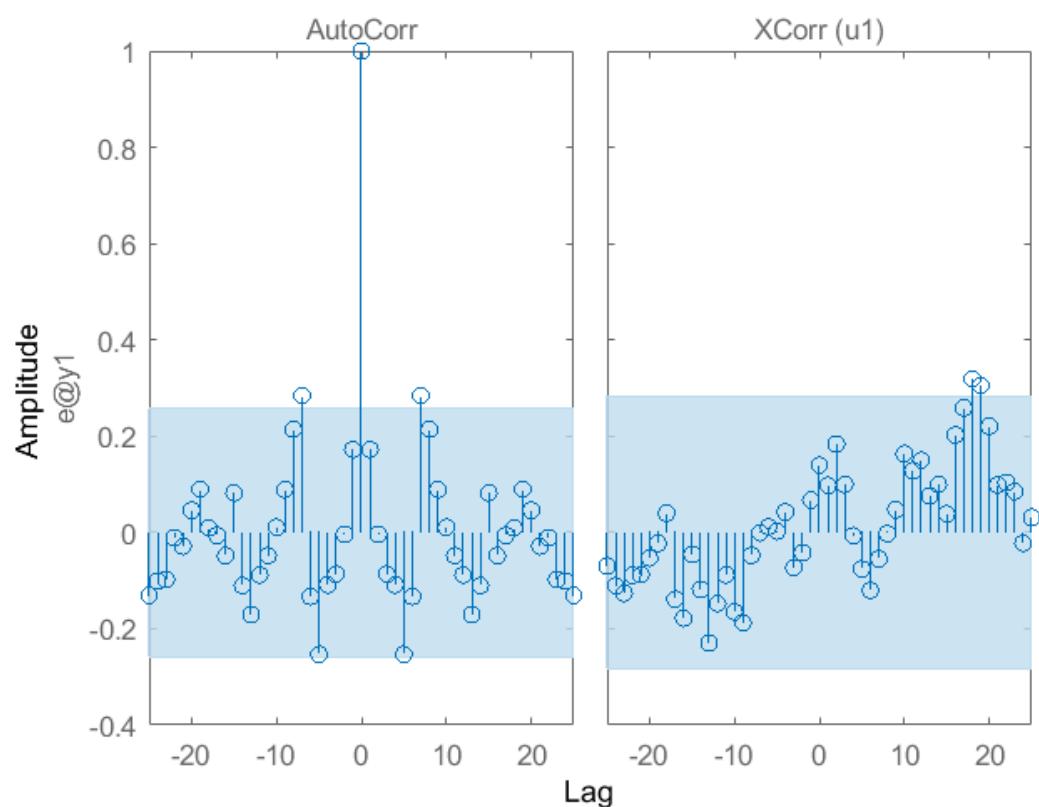
```
%% delay  
nk = delayest(data_train);
```

با توجه به این دستور تاخیر سیستم برابر با 2 میباشد اما این عدد بدست امده تنها یک حدس اولیه در مورد تاخیر سیستم میباشد و در هر یک از مدل ها ممکن است با افزایش یا کاهش این عدد به دقت های بهتری رسید اما اعداد بدست امده در محدوده این تاخیر پیشنهاد شده میباشد

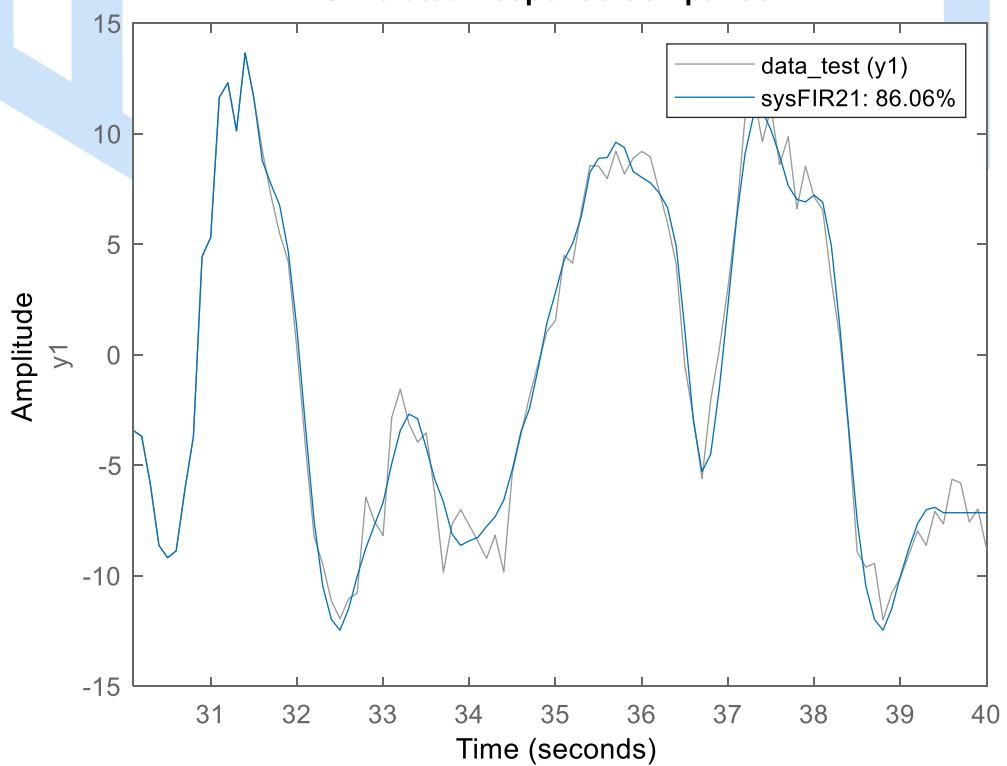


همان گونه که دیده میشود نتوانسته خواسته مسئله را برقرار کند پس مقدار  $n$  را بیشتر میکنیم با افزایش  $n$  به مقدار 16 سیستم به درصد خواسته شده میرسد

### resid FIR



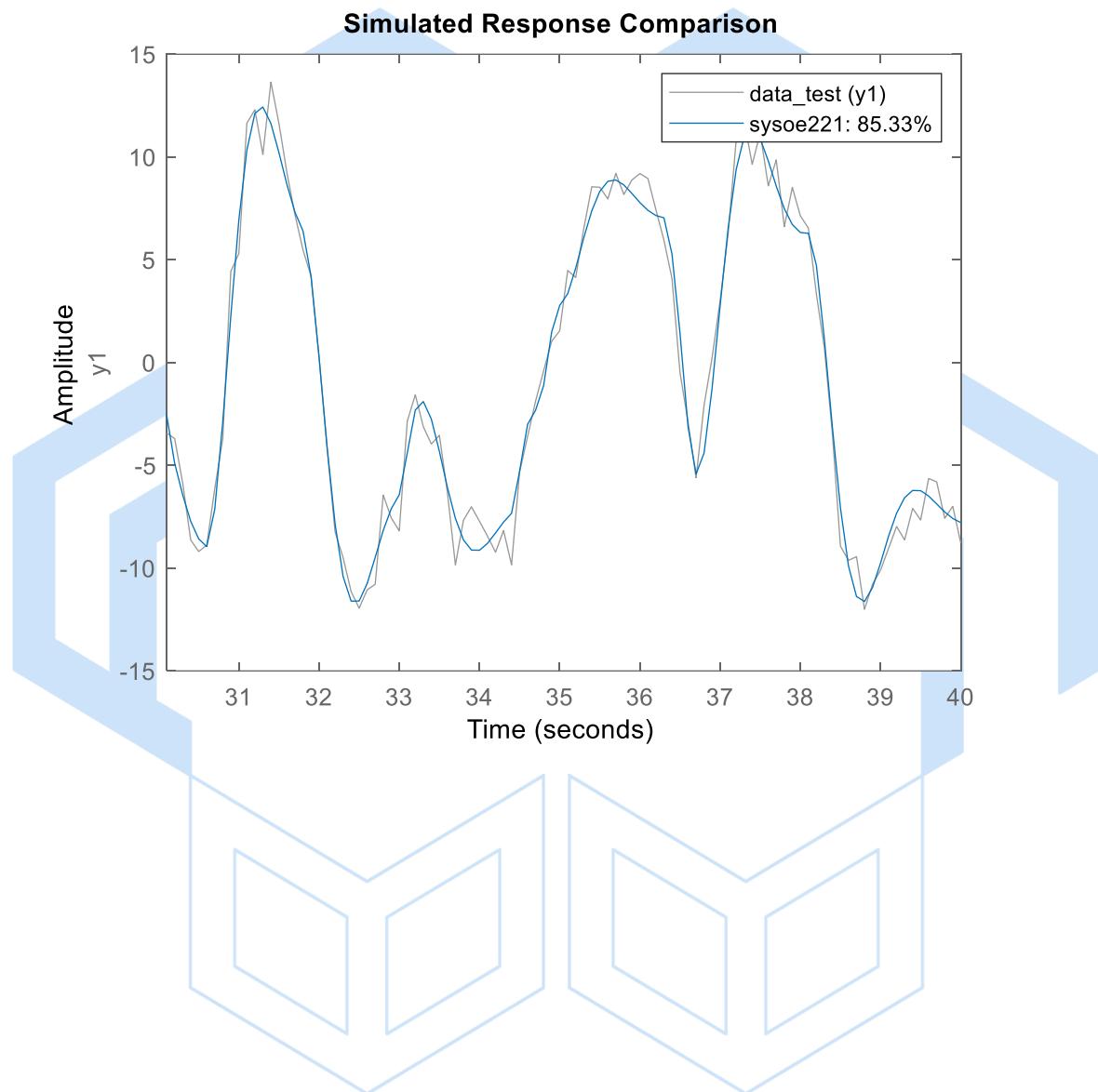
### Simulated Response Comparison



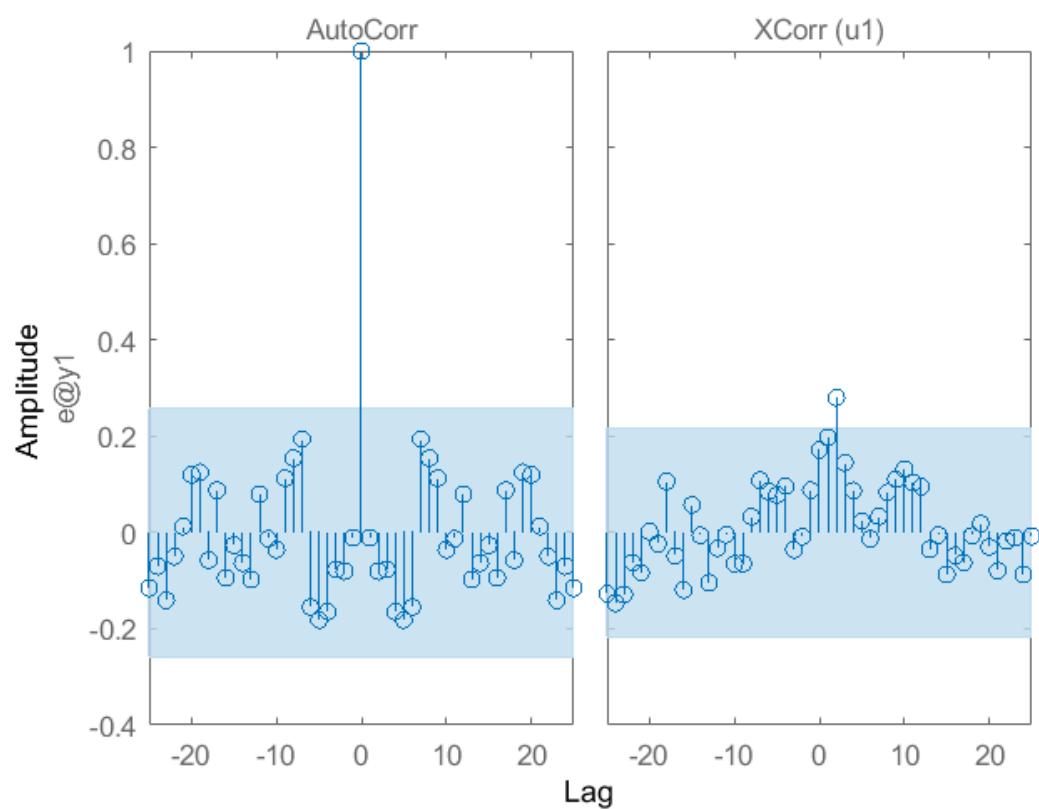
## OE مدل

با در نظر گرفتن پارامتر های مدل به صورت زیر داریم

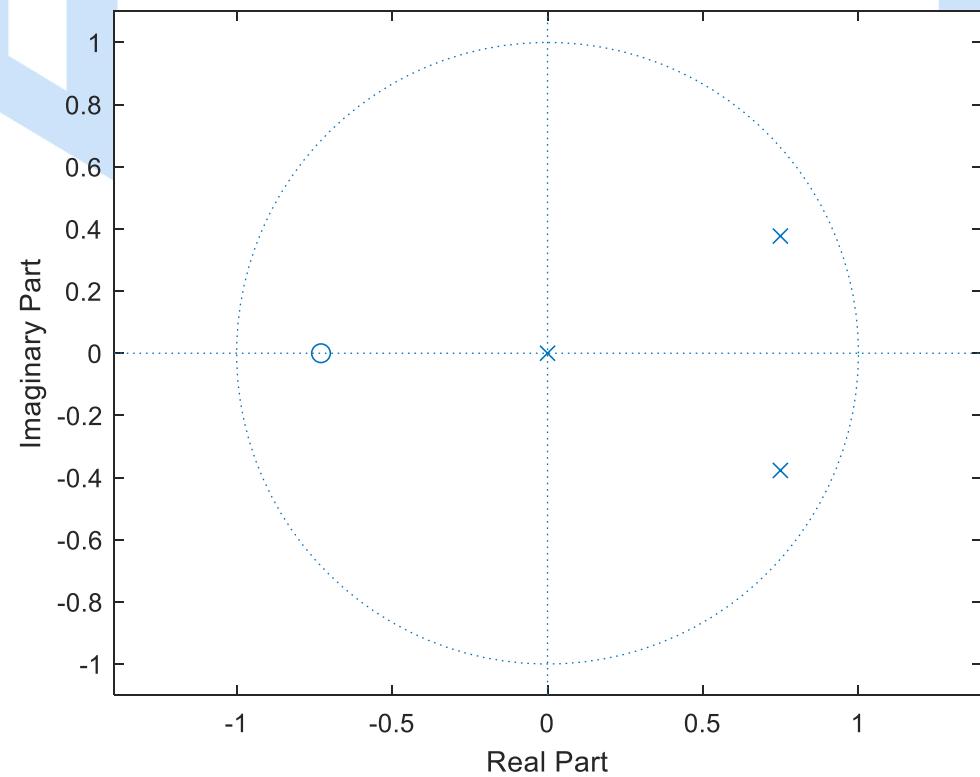
```
nb = 2 ;  
nf = 2 ;
```



### resid OE



### PZPlane OE



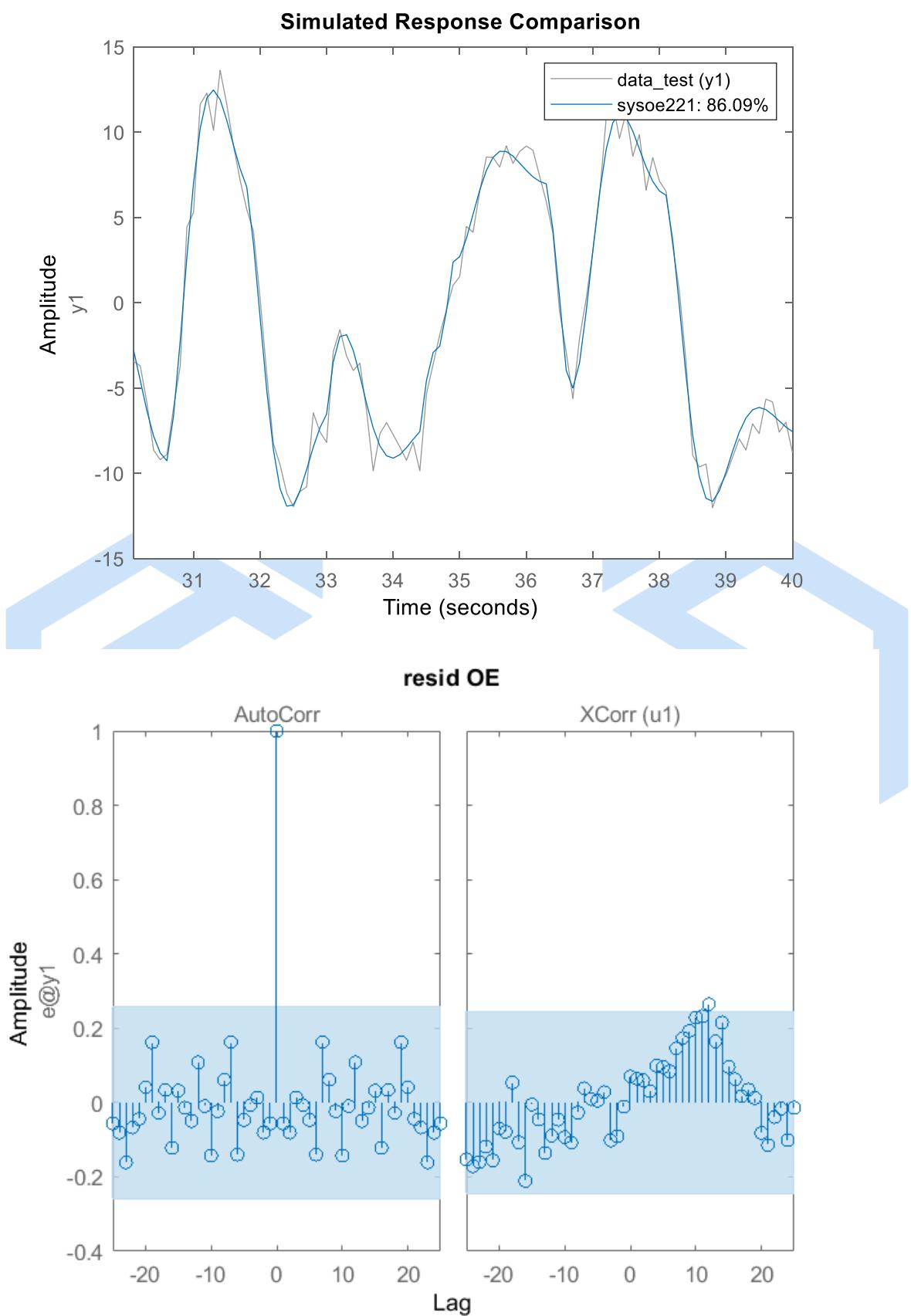
از انجایی که حدس اولیه به دقت لازم رسیده است سعی میکنیم تا پارامتر های مدل را کمتر کنیم تا  
مدل ساده تری داشته باشیم  
با پارامتر های

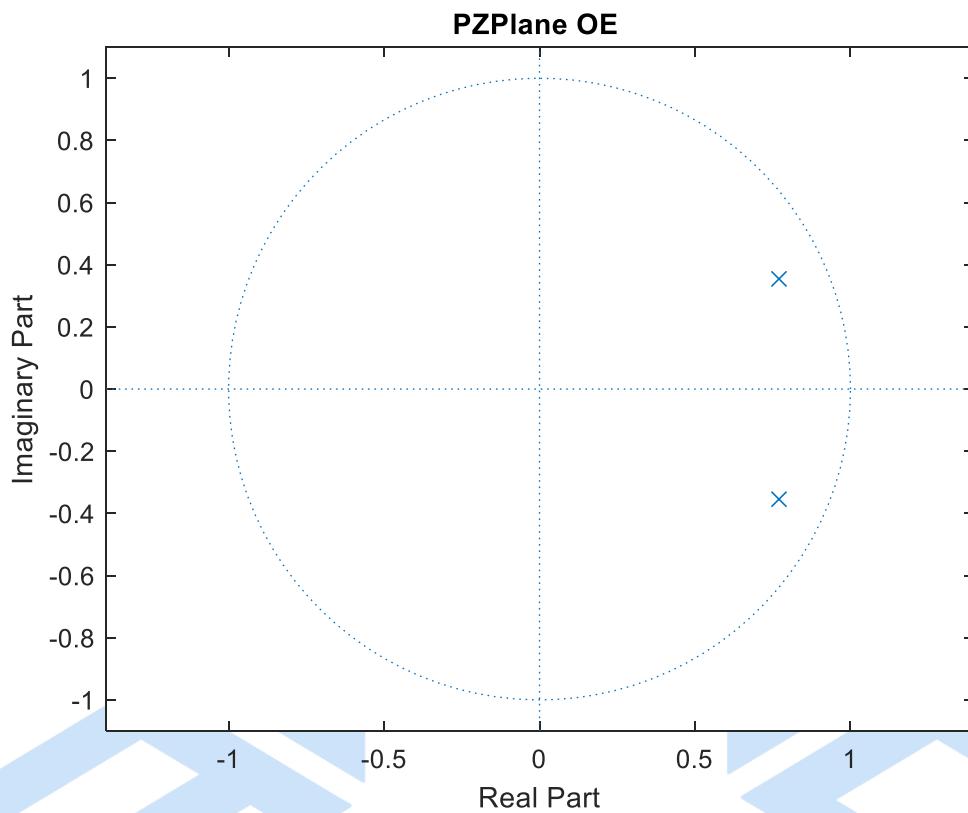
```
nb = 1 ;  
nf = 2 ;
```

```
Num of Parameters of OE: 3  
Sum of Squared Errors (SSE)OE: 207.0672
```

```
zeros_oe =  
  
0x1 empty double column vector
```

```
poles_oe =  
  
0.7701 + 0.3545i  
0.7701 - 0.3545i
```





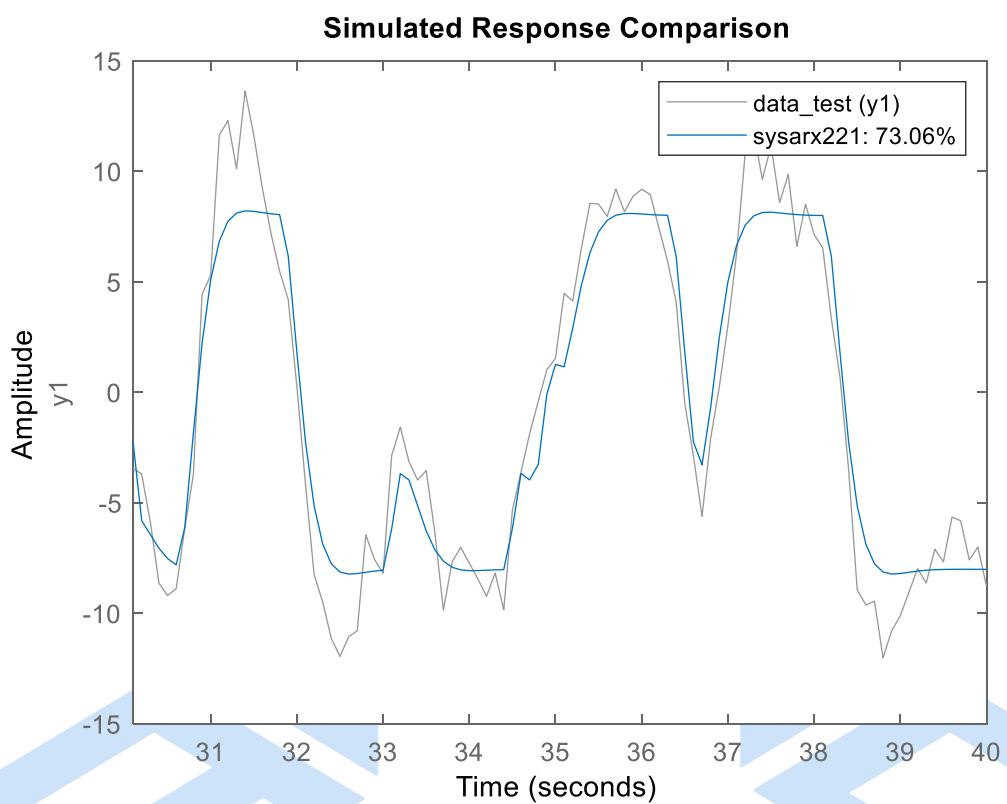
از انجایی که با مدل ساده تری نتایج قابل قبولی بدست اورده ایم پس بر اساس اصل خصت این مدل را از این خانواده انتخاب میکنیم

**مدل ARX**

با استفاده از پارامتر های زیر

$na = 2 ;$   
 $nb = 2 ;$

نتایج به صورت زیر میباشد:

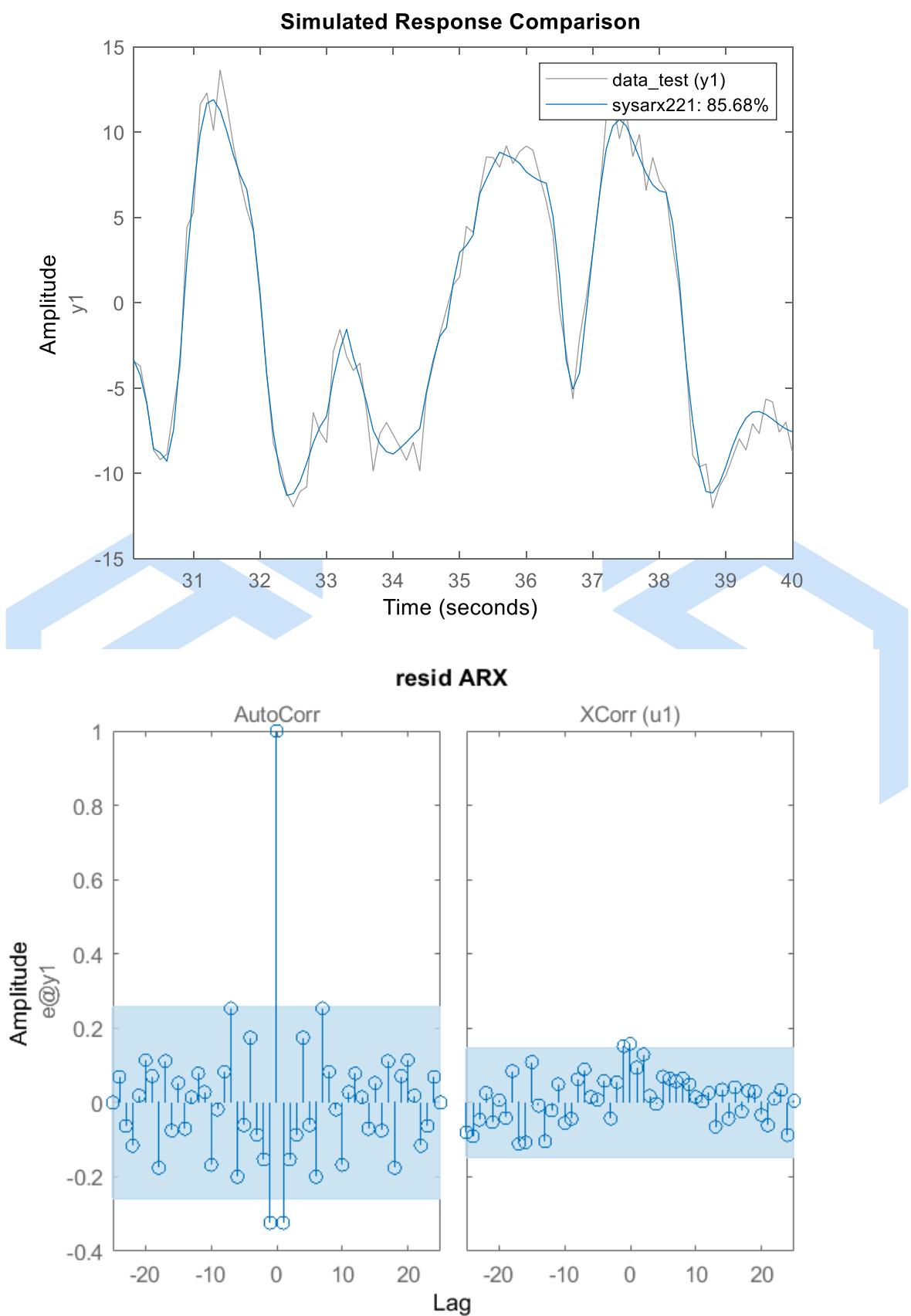


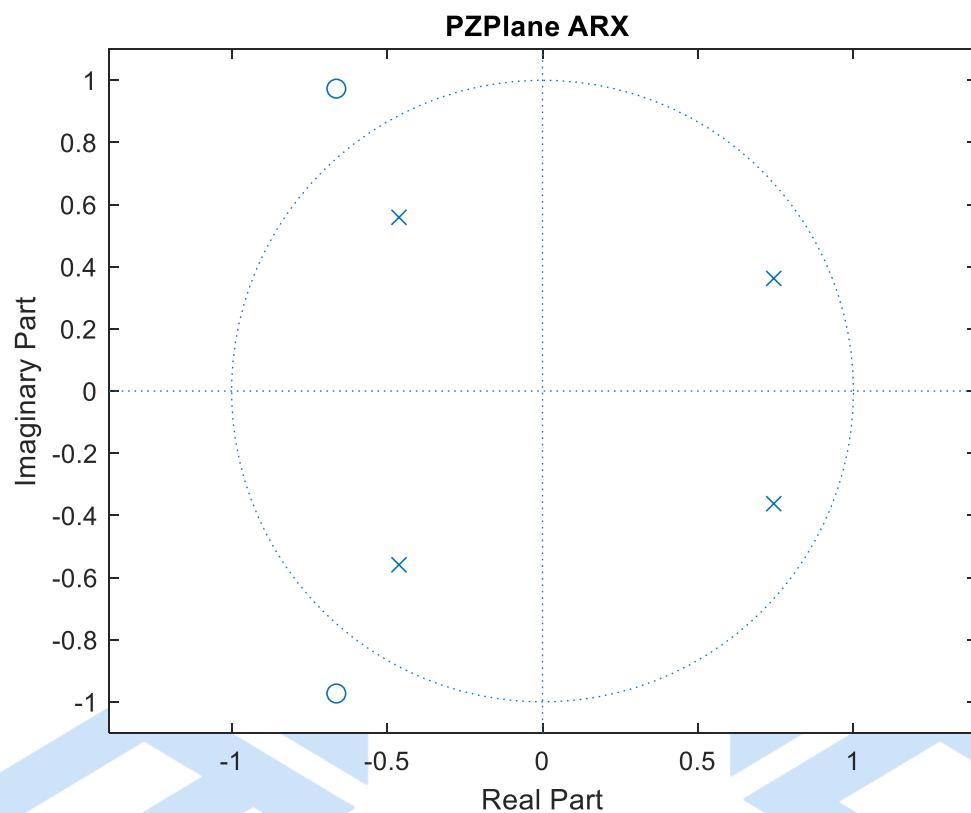
نتیجه خروجی به دقت مد نظر نرسیده است پس پارامتر های مدل را افزایش میدهیم

```
%% ARX Model
na = 4 ;
nb = 3 ;
Num of Parameters of ARX: 7
Sum of Squared Errors (SSE) ARX: 230.7801

zeros_arx =
-0.6631 + 0.9730i
-0.6631 - 0.9730i

poles_arx =
-0.4616 + 0.5589i
-0.4616 - 0.5589i
0.7436 + 0.3623i
0.7436 - 0.3623i
```





**ARMAX مدل**

با استفاده از پارامتر های زیر

```
na = 2 ;
nb = 2 ;
nc = 1 ;
```

نتایج زیر را بدست اورده ایم

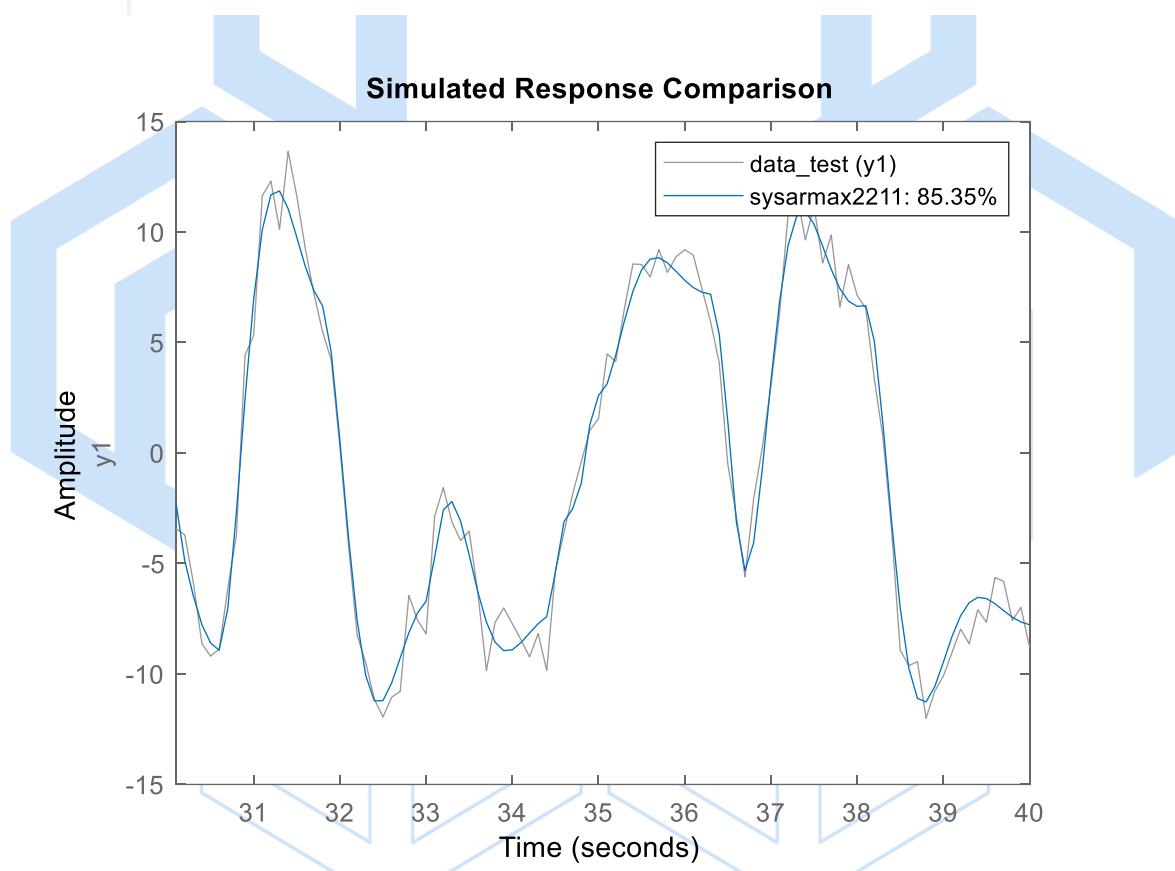
Num of Parameters of ARMAX: 5  
Sum of Squared Errors (SSE)ARMAX: 227.8102

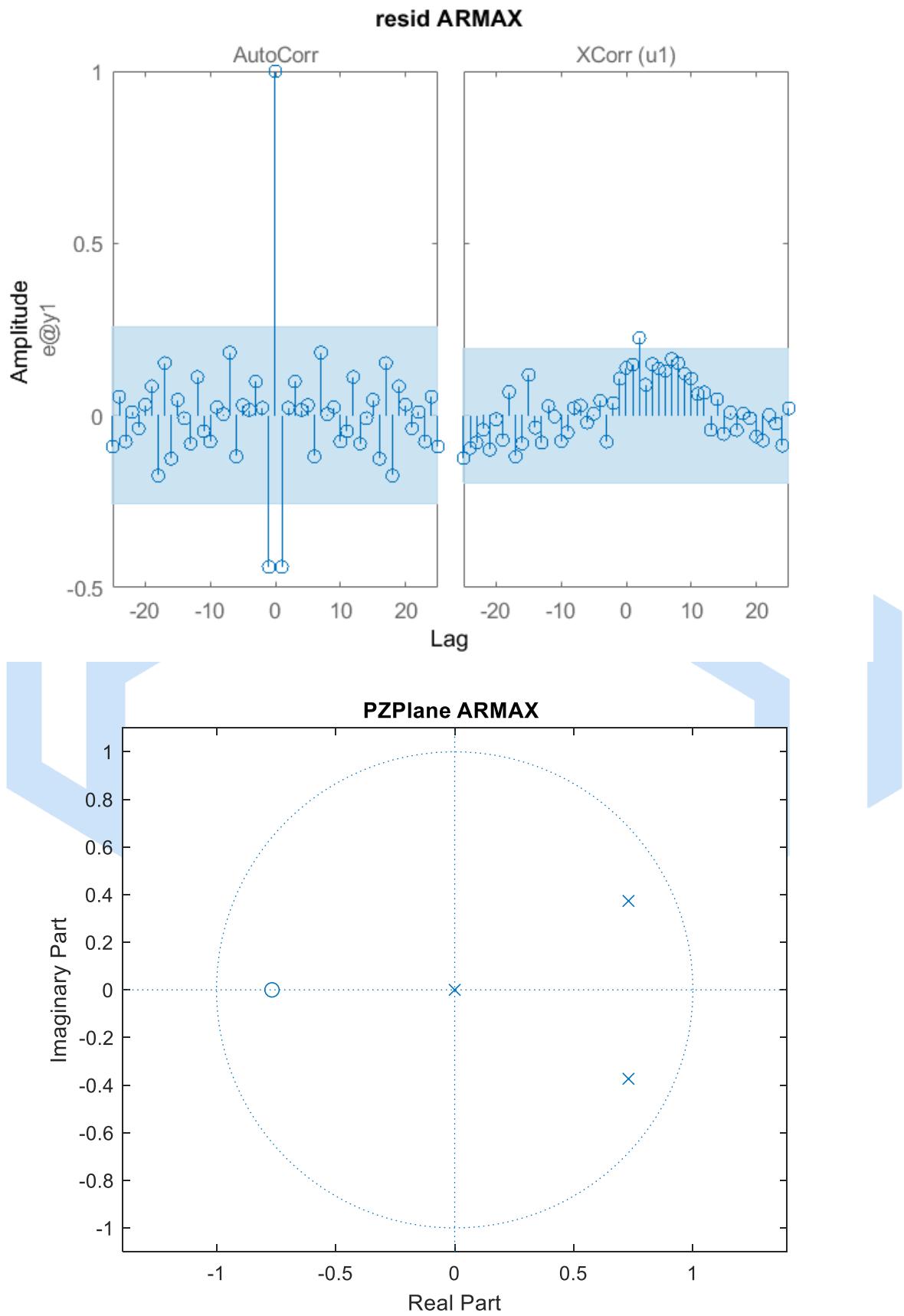
zeros\_armax =

-0.7670

poles\_armax =

0.7295 + 0.3735i  
0.7295 - 0.3735i

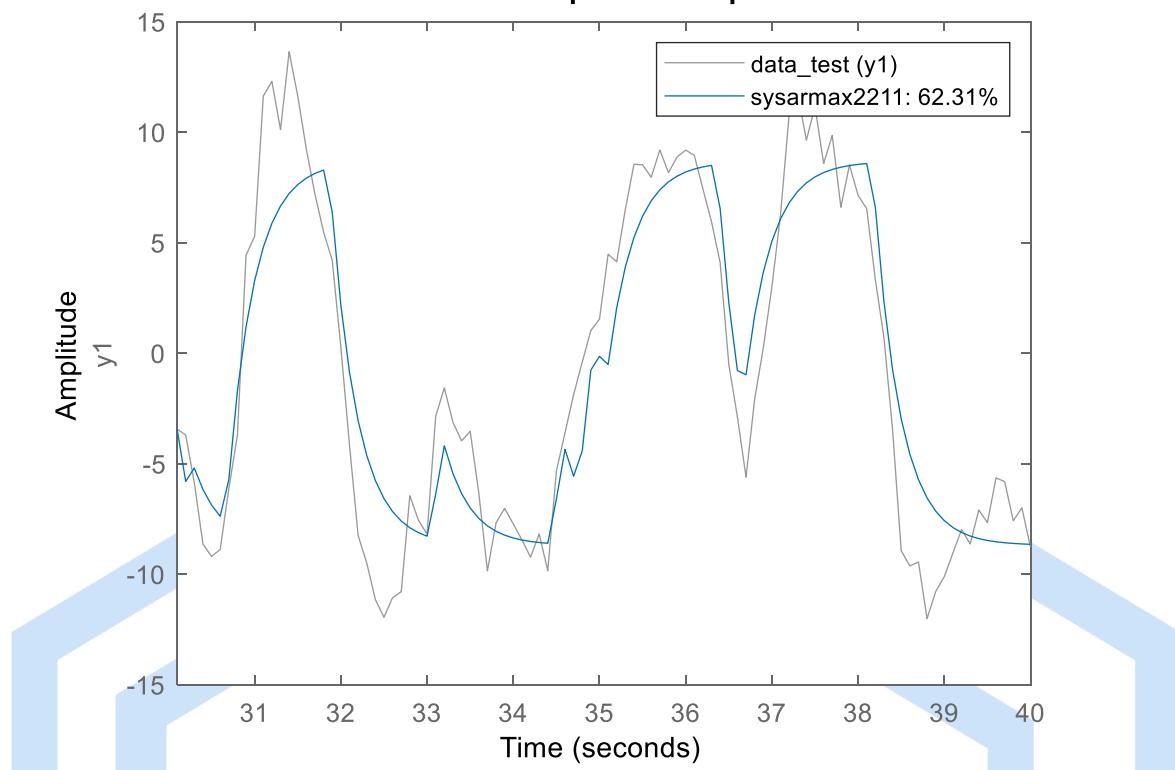




با کاهش پارامتر ها دقت مدل کمتر میشود به طور مثال با پارامتر های زیر

```
na = 1 ;  
nb = 2 ;  
nc = 1 ;
```

Simulated Response Comparison



ARARX مدل

به ازای پارامتر های زیر داریم

```
na = 2 ;  
nb = 1 ;  
nd = 2 ;
```

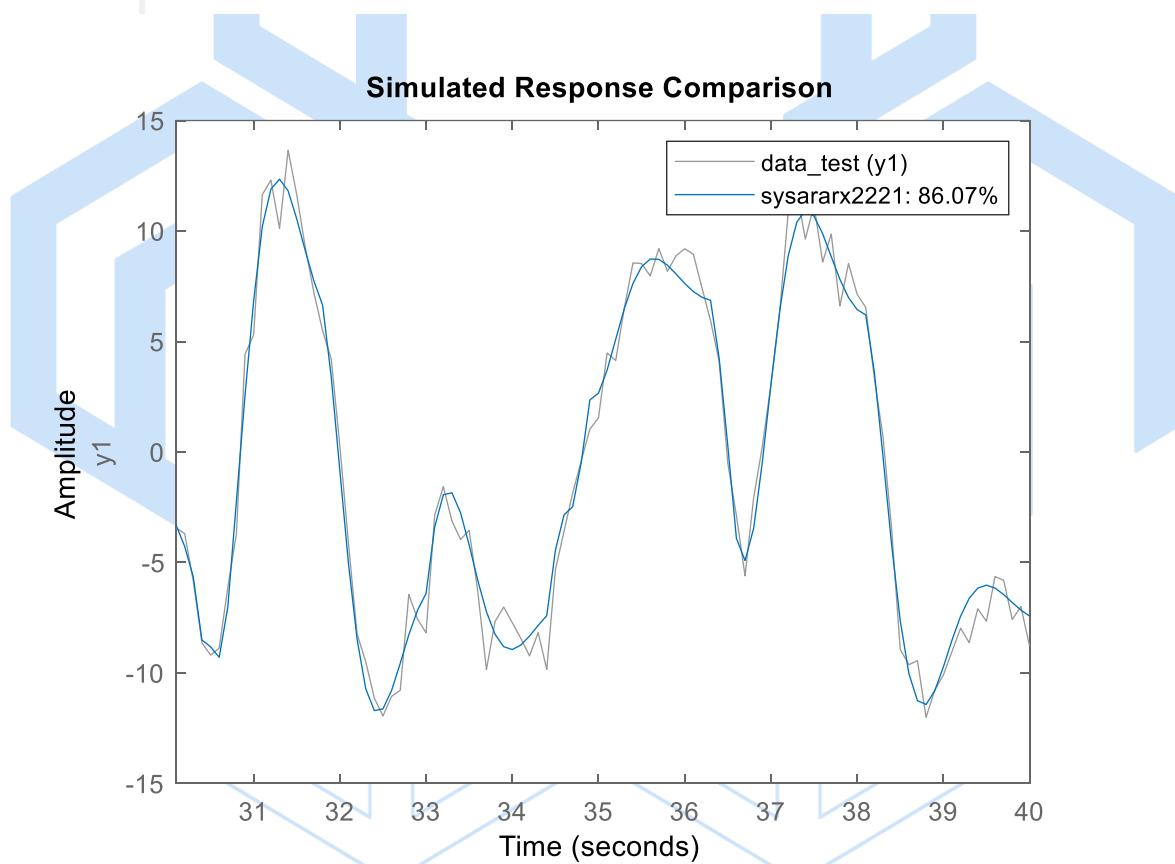
```
Num of Parameters of ARARX: 5  
Sum of Squared Errors (SSE)ARARX: 211.3745
```

```
zeros_ararx =
```

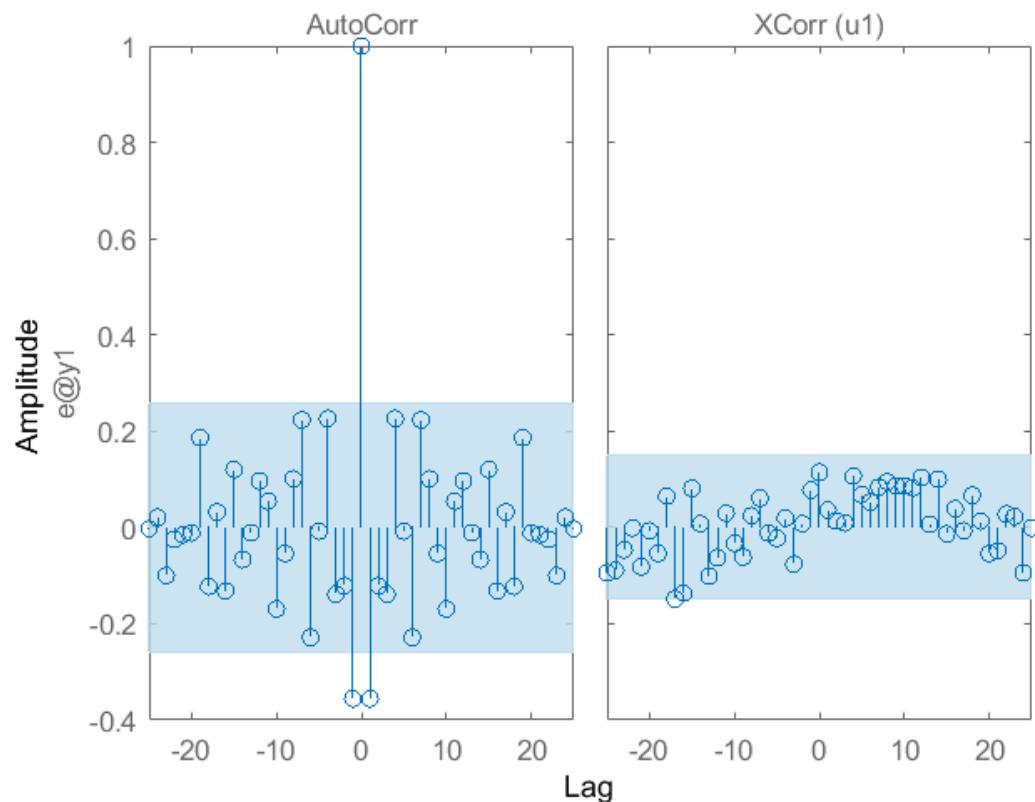
```
0×1 empty double column vector
```

```
poles_ararx =
```

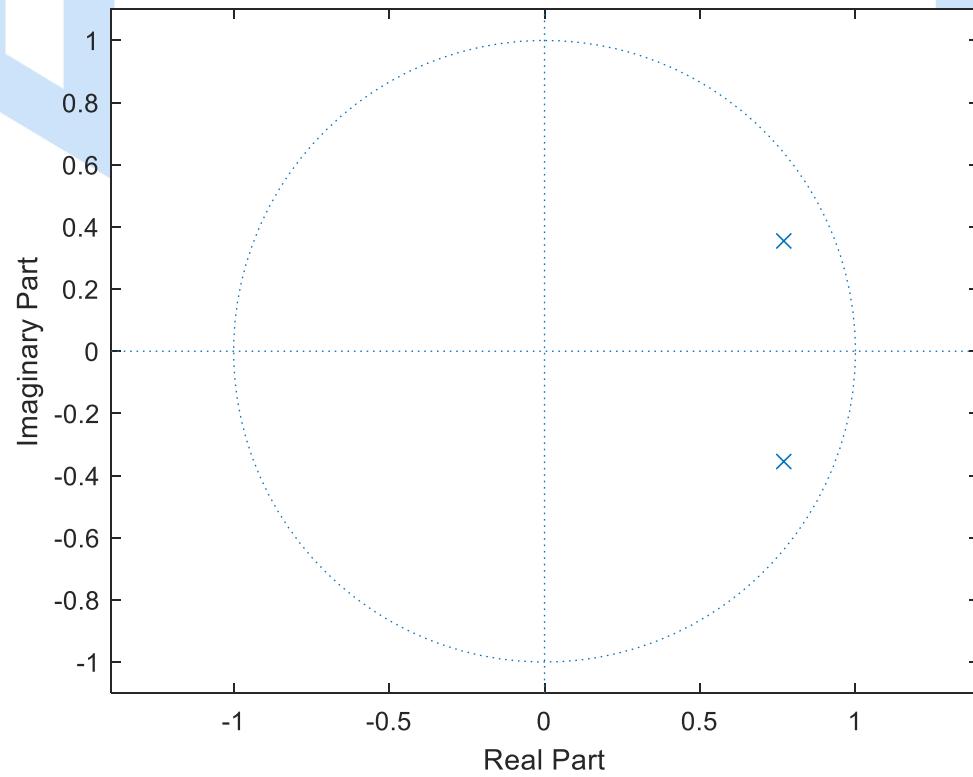
```
0.7694 + 0.3548i  
0.7694 - 0.3548i
```



### resid ARARX

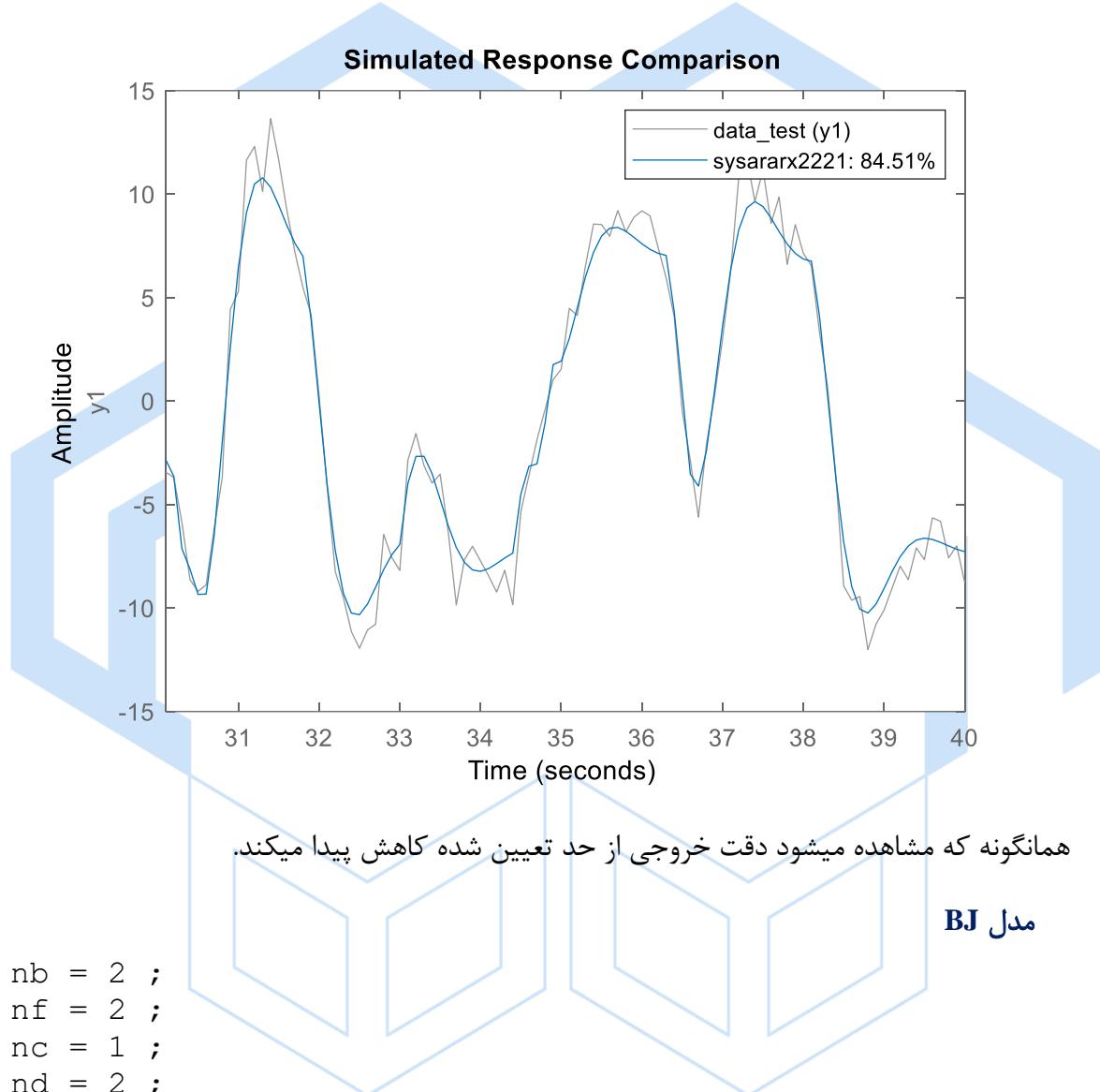


### PZPlane ARARX



مشاهده شد که سیستم از دقت کافی برخوردار است پس سعی میکنیم پارامتر های سیستم را کمتر کنیم

```
na = 2 ;  
nb = 1 ;  
nd = 1 ;
```



```
Num of Parameters of BJ: 7  
Sum of Squared Errors (SSE)BJ: 232.3877
```

```
zeros_bj =
```

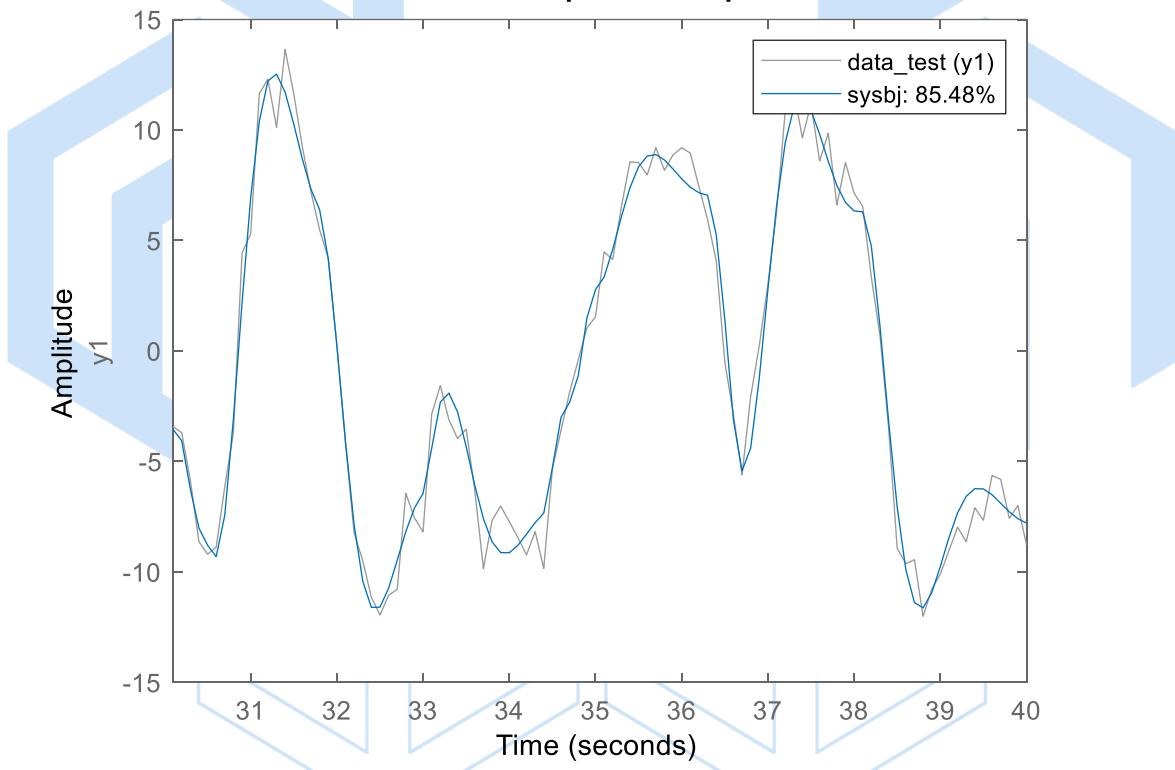
```
-0.7264
```

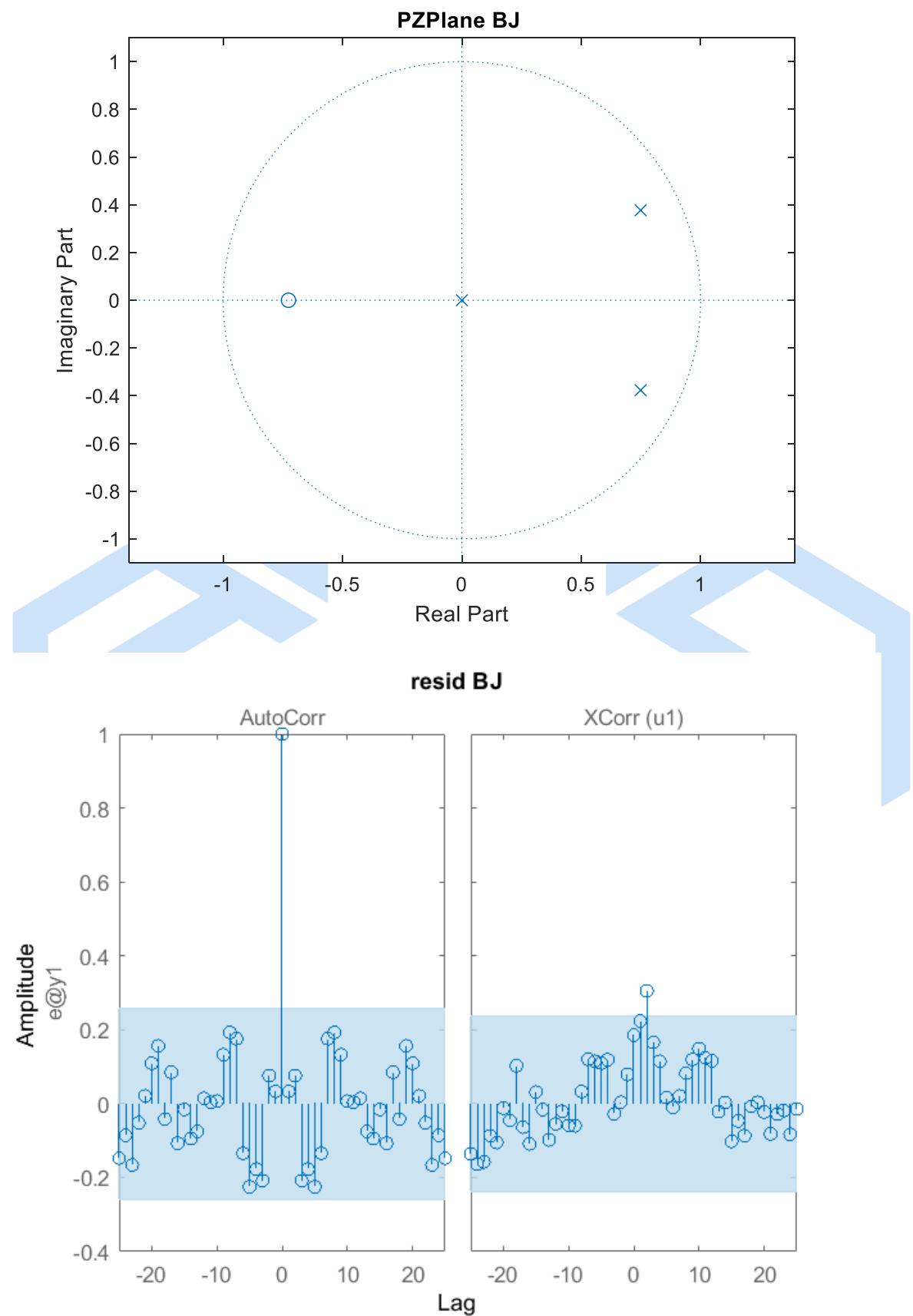
```
poles_bj =
```

```
0.7485 + 0.3770i
```

```
0.7485 - 0.3770i
```

**Simulated Response Comparison**





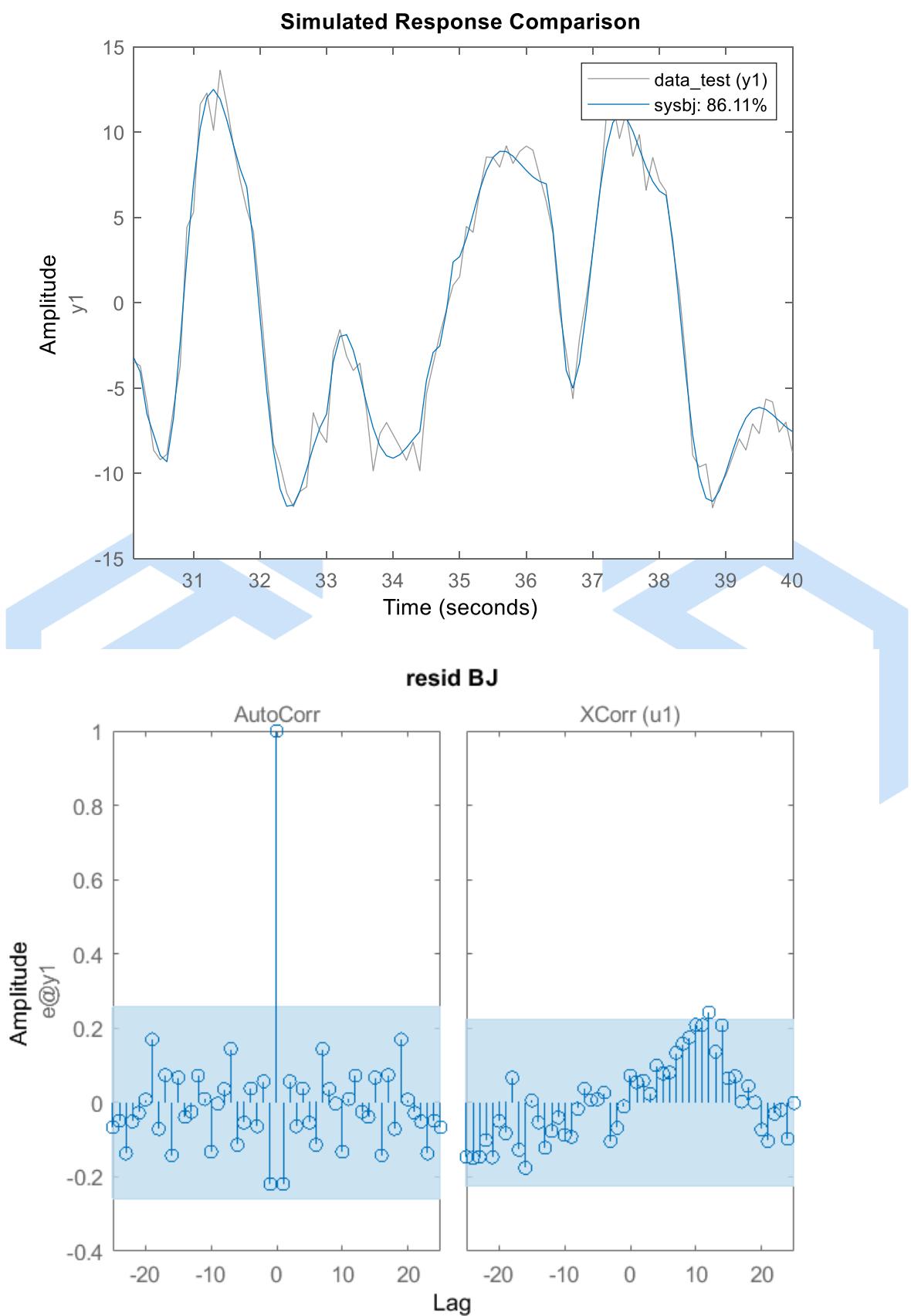
همان گونه که مشاهده میشود مدل بالا به خوبی توانسته خواسته مسئله را بطرف کند حال سعی در کاهش پارامتر ها و پیدا کردن مدل ساده تر میکنیم

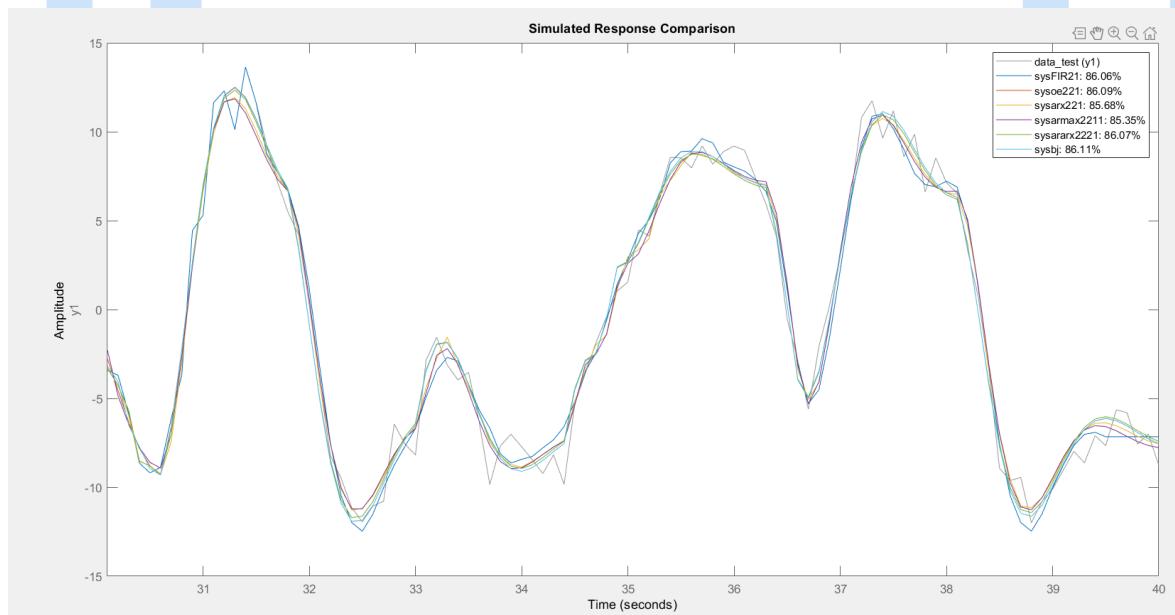
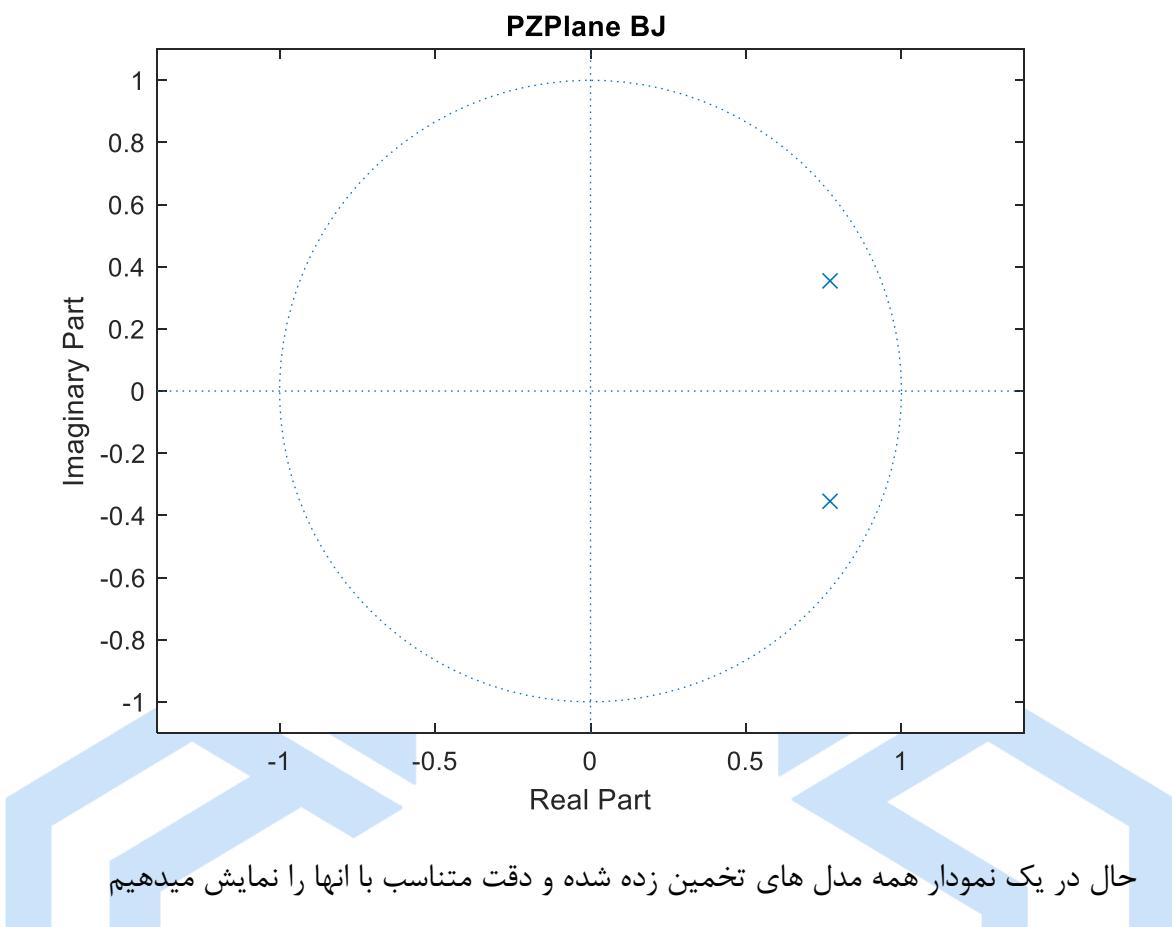
```
nb = 1 ;  
nf = 2 ;  
nc = 1 ;  
nd = 1 ;
```

```
Num of Parameters of BJ: 5  
Sum of Squared Errors (SSE) BJ: 207.2982
```

```
zeros_bj =  
  
0x1 empty double column vector
```

```
poles_bj =  
  
0.7702 + 0.3545i  
0.7702 - 0.3545i
```





همه مدل ها به دقت خواسته شده در مسئله رسیده اند حال بر مبنای سادگی باید مدل مناسب را انتخاب کنیم.

دقت بدست امده	تعداد پارامتر ها	مدل
86.06	16	<b>FIR</b>
86.09	3	<b>OE</b>
85.68	7	<b>ARX</b>
85.35	5	<b>ARMAX</b>
86.07	5	<b>ARARX</b>
86.11	5	<b>BJ</b>

شکل 1

برای بهترین مدل از نظر سادگی میتوان به مدل OE اشاره کرد. و بعد از آن مدل BJ پارامتر های کم و دقت بالا دارد