کاربرد MATLAB در سیگنالها و سیستمها و کنترل

مقدمه

بسته نرم‌افزاری MATLAB (Matrix Laboratory) یک سیستم ماتریس‌محور برای محاسبات ریاضی و مهندسی است. MATLAB در طول سالهای اخیر به ابزار بسیار قدرتمندی برای انجام پردازشهای پرزحمت و سنگین مهندسی مبدل شده است؛ به طوری که رسم برخی نمودارها یا انجام بعضی محاسبات را جز به کمک MATLAB نمی‌توان تصور کرد.

MATLAB امروزه دارای جعبه‌ابزارهای (TOOLBOX) متنوعی برای انجام محاسبات مختلف است که از جمله آنها می‌توان به جعبه‌ابزارهای پردازش سیگنال (Signal Processing)، پردازشهای آماری (Statistics)، شبکه‌های عصبی (Neural Network)، دریافت تصویر (Image Acquisition)، پردازش تصویر (Image Processing)، سیستمهای کنترلی (Control Systems)، طراحی فیلتر (Filter Design)، منطق فازی (Fuzzy Logic)، الگوریتم‌های ژنتیک (Genetic Algorithms) و ... اشاره کرد. بعضی از این جعبه‌ابزارها به همراه نرم‌افزار MATLAB ارائه می‌شوند و بعضی دیگر را می‌توان در کارگروههای مختلف و نیز سایت www.Mathworks.com یافت.

هدف این نوشتار، تشریح کامل ساختار MATLAB و جعبه‌ابزارهای آن نیست. هدف ما ارائه مطالب ضروری برای کاربرد مؤثر MATLAB در حل مسایل پردازش سیگنال و سیسمتهای کنترل است.

در بخش ابتدایی این مطلب به ساختارهای اساسی MATLAB و اصول استفاده از آن خواهیم پرداخت.

بخش دوم به کاربرد MATLAB در پردازش تصویر به عنوان یکی از کاربردهای اصلی سیگنالها و سیستمها اختصاص دارد.

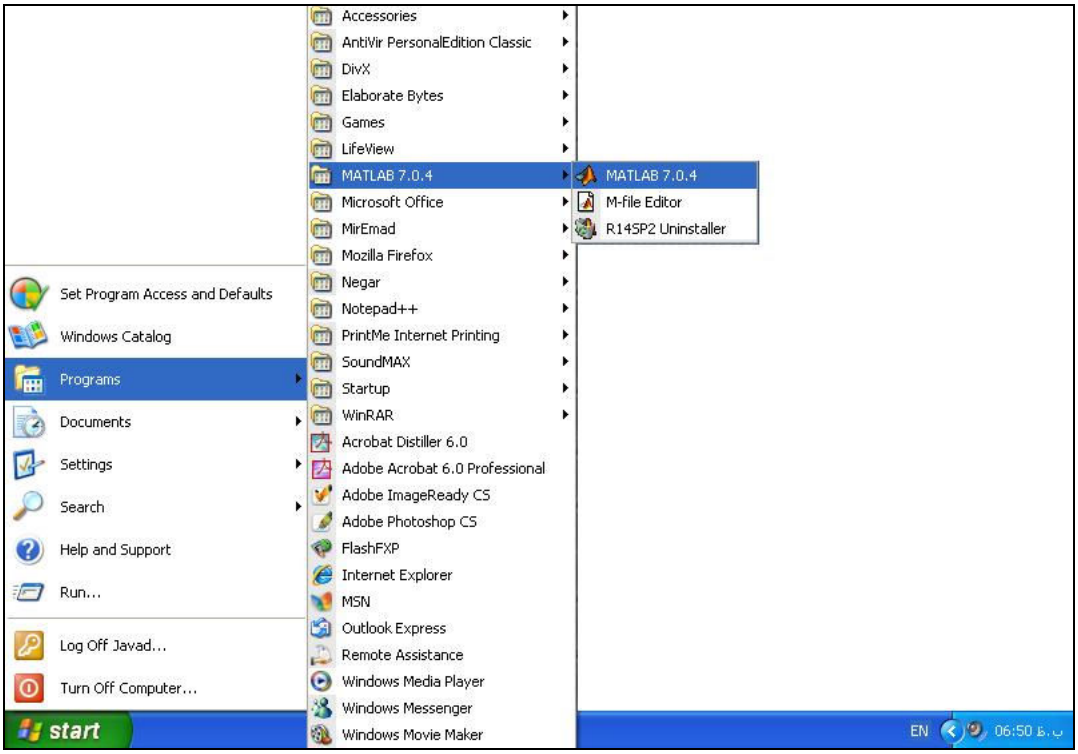
در بخش سوم کاربردهای MATLAB در تحلیل و طراحی سیستمهای کنترلی را مرور خواهیم کرد. نحوه استفاده از ابزارهای قدرتمند LTIVIEW، SISOTOOL و SIMULINK در این بخش تشریح خواهد شد.

بخش اول – MATLAB، یک آزمایشگاه ماتریسی

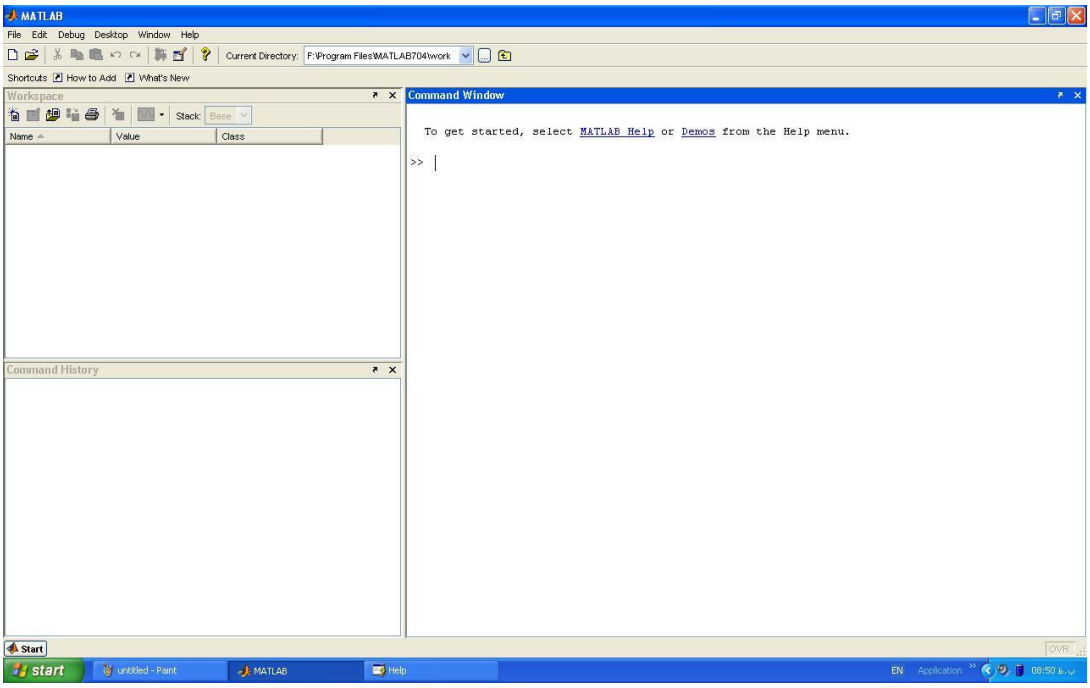
شروع کار با MATLAB

نرم‌افزار کامل MATLAB شامل دو یا سه لوح فشرده است. نیازی به نصب تمام جعبه‌ابزارهای همراه با MATLAB نیست. این کار تنها باعث هدر رفتن فضای حافظه کامپیوتر شما می‌شود. بنابراین در هنگام نصب MATLAB، با انتخاب گزینه Custom Installation، تنها جعبه‌ابزارهای مورد نیاز خود را انتخاب کنید. نکته مهم در هنگام نصب MATLAB، لزوم داشتن یک شماره سریال تحت عنوان PLP است. چنانچه نرم‌افزار MATLAB شما نسخه اصلی (original) نیست، باید برنامه crack که به همراه آن ارائه شده را اجرا کنید و PLP که توسط آن تولید می‌شود را در هنگام نصب MATLAB به کار برید.

پس از نصب MATLAB، از طریق منوی Start باید به محیط MATLAB وارد شوید:



پس از ورود به MATLAB باید منتظر بمانید تا بارگذاری محیط آن به پایان رسیده و آماده دریافت دستورات شما شود:



محیط اصلی کاری MATLAB، محیط فرمانپذیر (Command) است. در این محیط برای اجرای یک دستور یا تابع باید عنوان آن را در تایپ کرده و کلید Enter را فشار دهید. به عنوان مثال با تایپ دستور quit یا exit و فشار کلید Enter از MATLAB خارج می‌شوید. با فشار کلید ↑ می‌توانید دستورات قبلی را مجدداً احضار کنید.

به یاد داشته باشید که MATLAB نسبت به بزرگی و کوچکی حروف حساس است و باید دستورات و متغیرها را به همان صورتی که تعریف شده‌اند به کار برید.

روش دیگر استفاده از MATLAB، ذخیره یک مجموعه دستور در یک فایل با پسوند m. ، تایپ نام فایل در محیط MATLAB و فشار کلید Enter است. به کمک این روش می‌توانید تعداد زیادی از دستورات را به صورت متوالی اجرا کنید. فایل فوق باید در زیرشاخه work در مسیری که MATLAB در آن مسیر نصب شده است قرار داشته باشد (مثلاً F:\Program Files\MATLAB704\work).

در بخشی از محیط MATLAB که Command History نام دارد، آخرین دستوراتی که اجرا نموده‌اید نگهداری می‌شوند (با انتخاب گزینه Command History از منوی desktop می‌توانید این بخش را ببینید). با کلیک کردن روی هر دستور می‌توانید آن را مجدداً اجرا کنید.

متغیرها در MATLAB

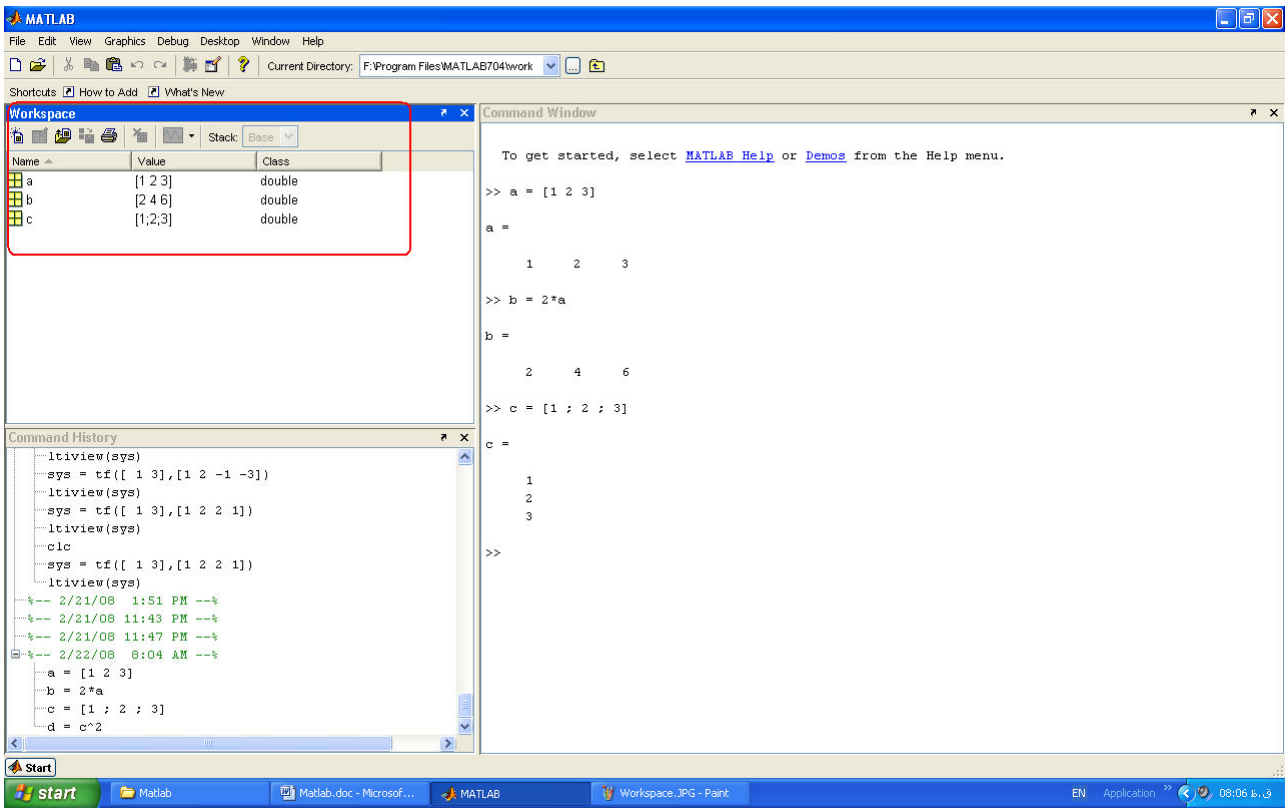
MATLAB زبانی است که تنها برای کار با ماتریسها طراحی شده است. تمام متغیرهای MATLAB از نوع ماتریس هستند. حتی متغیری که باید یک عدد را درون خود نگهداری کند به صورت یک ماتریس ۱×۱ تعریف می‌شود! بنابراین تمام توابع MATLAB روی ماتریسها اجرا می‌شوند.

یکی از جنبه‌های مثبت MATLAB این است که متغیرهای آن نیازی به تعریف ندارند و در هر نقطه از برنامه می‌توانید متغیر جدیدی تعریف کنید. ویژگی جالب دیگر MATLAB آن است که بُعد (اندازه) متغیرها می‌تواند در طول برنامه عوض شود. به بیان دیگر، با اولین استفاده از یک متغیر بُعد آن مشخص می‌شود و در استفاده‌های بعدی قابل تغییر است. به عنوان مثال اگر خط فرمان زیر را به‌کار برید:

```
a = [1  2]
```

یک متغیر ماتریسی ۱×۲ به نام a تعریف کرده‌اید. چنانچه در ادامه برنامه از دستور a = 5 استفاده شود، بُعد متغیر a به یک ماتریس ۱×۱ تغییر خواهد کرد.

MATLAB دارای بخشی به نام Workspace است که متغیرهای تعریف شده و بُعد و مقدار فعلی آنها را در آنجا نگهداری می‌کند:



چنانچه این بخش را در محیط MATLAB نمی‌بینید، از منوی Desktop گزینه Workspace را انتخاب کنید. به کمک فرمان who می‌توانید تمام متغیرهایی که در فضای فعلی MATLAB تعریف شده‌اند را ببینید.

برای پاک کردن تمام متغیرها از فرمان clear استفاده می‌کنیم. چنانچه می‌خواهید فقط یک متغیر را پاک کنید، باید نام آن متغیر را مقابل دستور clear ذکر کنید؛ مثلاً دستور clear c در شکل قبلی، متغیر c را پاک می‌کند.
با خروج از MATLAB تمام متغیرها پاک می‌شوند. چنانچه تمایل دارید متغیرها برای اجرای بعدی MATLAB باقی بمانند، قبل از خروج از دستور save استفاده کنید. با این کار، متغیرها در فایلی به نام matlab.mat ذخیره شده و با ورود مجدد به MATLAB به کمک دستور load می‌توانند بازیابی شوند.

تعریف آرایه‌ها و ماتریسها

همانگونه که پیشتر گفته شد، MATLAB یک نرم‌افزار ریاضی ماتریس‌محور است و تمام متغیرهای آن باید از نوع ماتریسی تعریف شوند.
چنانچه «یک» قلم اطلاعات (مثلاً یک عدد یا یک کاراکتر) را به متغیری نسبت دهید، یک ماتریس ۱×۱ برای آن متغیر در نظر گرفته می‌شود. مثلاً دستور X = 0.56 یا C = 'a' متغیرهای یک‌بعدی X و C را با مقادیر اولیه معلوم (0.56 و کاراکتر 'a') تعریف می‌کند.
برای تعریف یک آرایه سطری، باید مقادیر اولیه مورد نظرتان (که با فاصله خالی یا کاما باید از هم جدا شوند) را مابین دو علامت [] قرار داده و به متغیر نسبت دهید. مثلاً دستور زیر را ببینید:

```
A = [1  2  3  4  5]
```

این دستور یک ماتریس ۵×۱ (یک آرایه سطری ۵×۱) به نام a تعریف کرده و مقادیر 1, 2, 3, 4, 5 را به عناصر آن نسبت می‌دهد.

با اجرای دستورات MATLAB، نتیجه اجرای آن نیز در محیط MATLAB نمایش داده می‌شود؛ مثلاً با اجرای دستورات بالا، نتیجه اجرا به صورت زیر نشان داده می‌شود:

```
A =
   1   2   3   4   5
```

اگر مایل نیستید نتیجه اجرای دستورات نمایش داده شود (مثلاً وقتی با ماتریسهای مفصل مانند یک تصویر کار می‌کنید) می‌توانید در انتهای دستور از علامت ; (سمی‌کالن) استفاده کنید.

چنانچه مایل باشید آرایه را به صورت ستونی تعریف کنید، باید بین عناصر آرایه به جای فاصله خالی یا کاما از علامت ; استفاده کنید. این علامت نشانه ایجاد یک سطر جدید در ماتریس است. مثلاً دستور

```
A = [1 ; 2 ; 3 ; 4 ; 5]
```

باعث می‌شود یک ماتریس ۵×۱ (یک آرایه ستونی ۱×۵) به صورت زیر ایجاد شود:

```
A =
   1
   2
   3
   4
   5
```

برای تعریف یک ماتریس، باید عناصر آن ماتریس را بین علامتهای [] با شروع از سطر اول ماتریس وارد کرده و برای ایجاد سطر جدید از علامت ; استفاده کنید. مثلاً دستور

```
A = [1  2  3 ; 4  5  6 ; 7  8  9]
```

یک ماتریس ۳×۳ به صورت زیر ایجاد می‌کند:

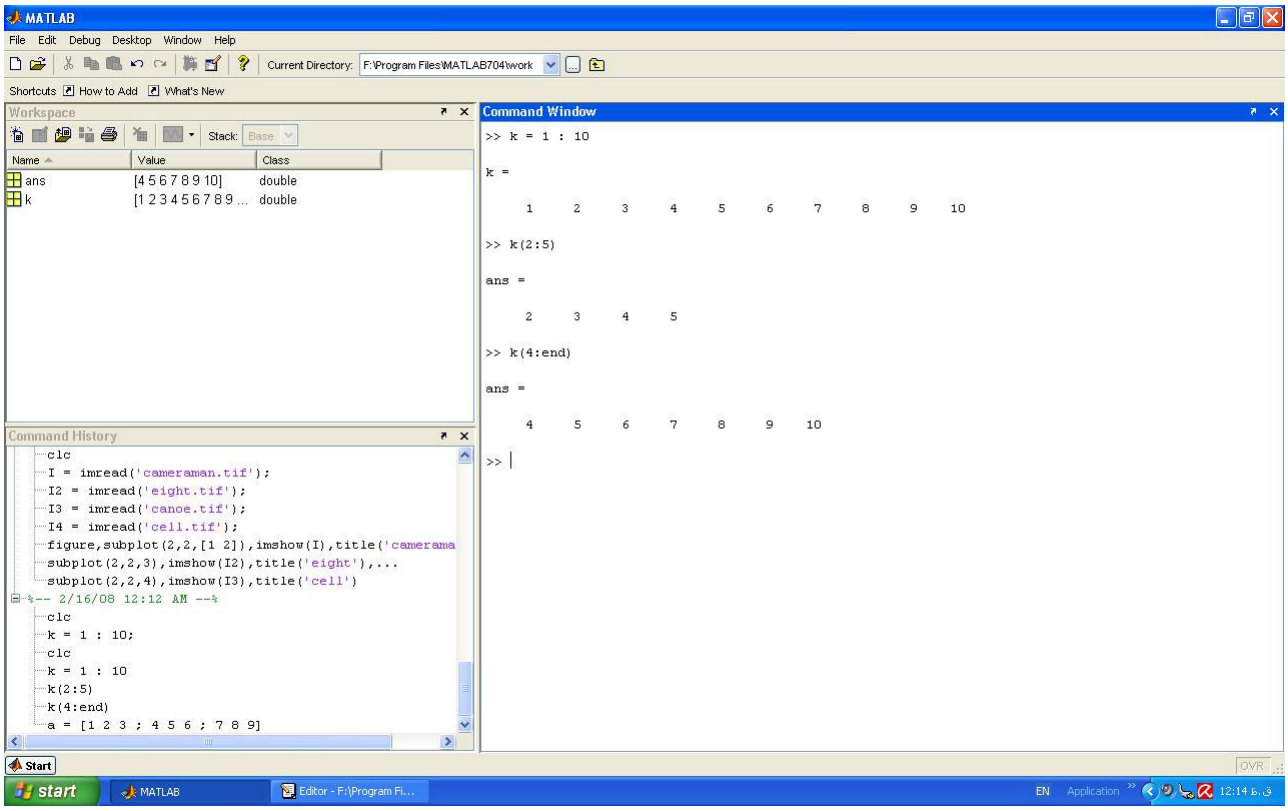
```
A =
   1   2   3
   4   5   6
   7   8   9
```

برای دسترسی به یکی از خانه‌های ماتریس، باید شماره سطر و ستون آن را بین دو علامت ( ) قرار دهید. مثلاً اگر بعد از تعریف ماتریس بالا، مقدار ذخیره شده در سطر ۲ و ستون ۳ ماتریس را بخواهید (در MATLAB شماره سطرها و ستونها از یک آغاز می‌شود)، کافی است دستور a(2,3) را تایپ کنید تا MATLAB به صورت زیر مقدار آن عنصر را به شما نمایش دهد:

```
ans =
    6
```

MATLAB یک متغیر پیش‌فرض به نام ans (مخفف answer) دارد که هرگاه خروجی یک دستور به متغیری نسبت داده نشده باشد، به صورت پیش‌فرض مقدار خروجی آن دستور را در این متغیر قرار می‌دهد.

چنانچه آرایه شما از عناصری با فواصل معلوم و مساوی تشکیل شده باشد، به کمک عملگر : می‌توانید آن آرایه یا ماتریس را تعریف کنید و دیگر نیازی به تعریف تمام عناصر نیست. شکل زیر را ببینید:



دستور k = 1:10 یک آرایه ۱۰ عنصری با مقادیر ۱ تا ۱۰ تعریف می‌کند. اگر بخواهیم فاصله عناصر آرایه ۰/۵ باشد باید از دستور زیر استفاده کنیم:

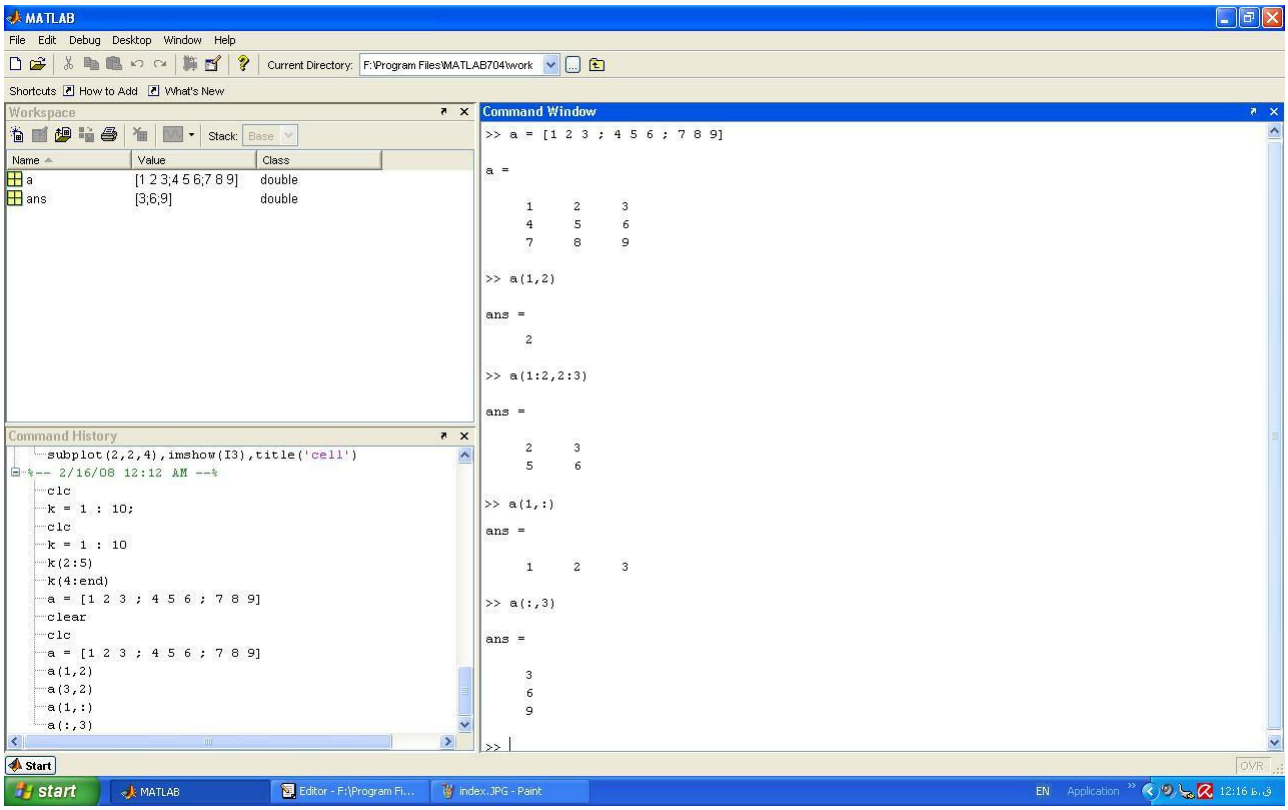
```
k = 1 : 0.5 : 10
```

کاربرد کلی عملگر : در تعریف آرایه‌ها به شکل زیر است:

مقدار آخرین عنصر آرایه : فاصله عناصر آرایه : مقدار اولین عنصر آرایه = نام آرایه

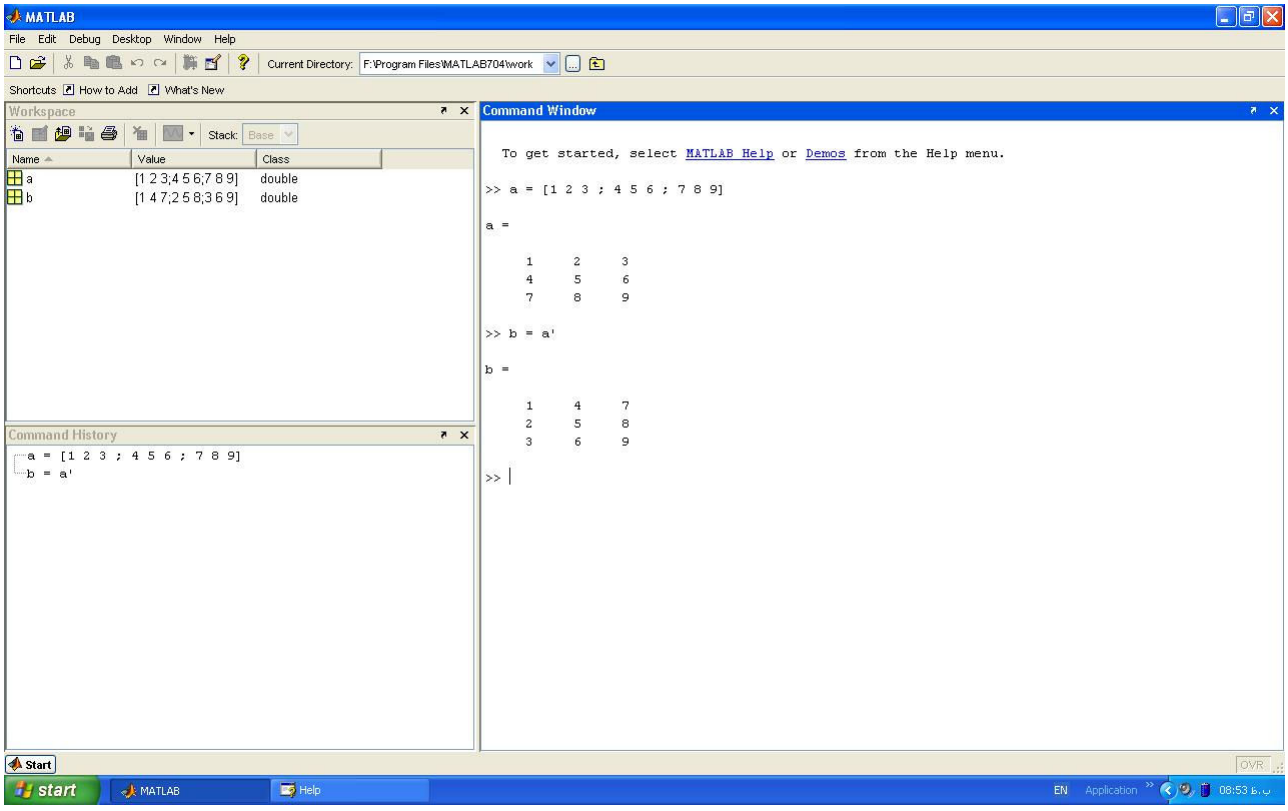
چنانچه بخواهیم با بخشی از یک آرایه یا ماتریس کار کنیم، می‌توان از عملگر : استفاده کرد. در شکل بالا دستور k(2:5) عناصر ۲ تا ۵ آرایه k و دستور k(4:end) عناصر ۴ تا انتهای آرایه k را در محیط MATLAB به نمایش درمی‌آورد.

چند نمونه از کاربردهای عملگر : در نمایش زیرماتریسها را در شکل زیر می‌بینید:

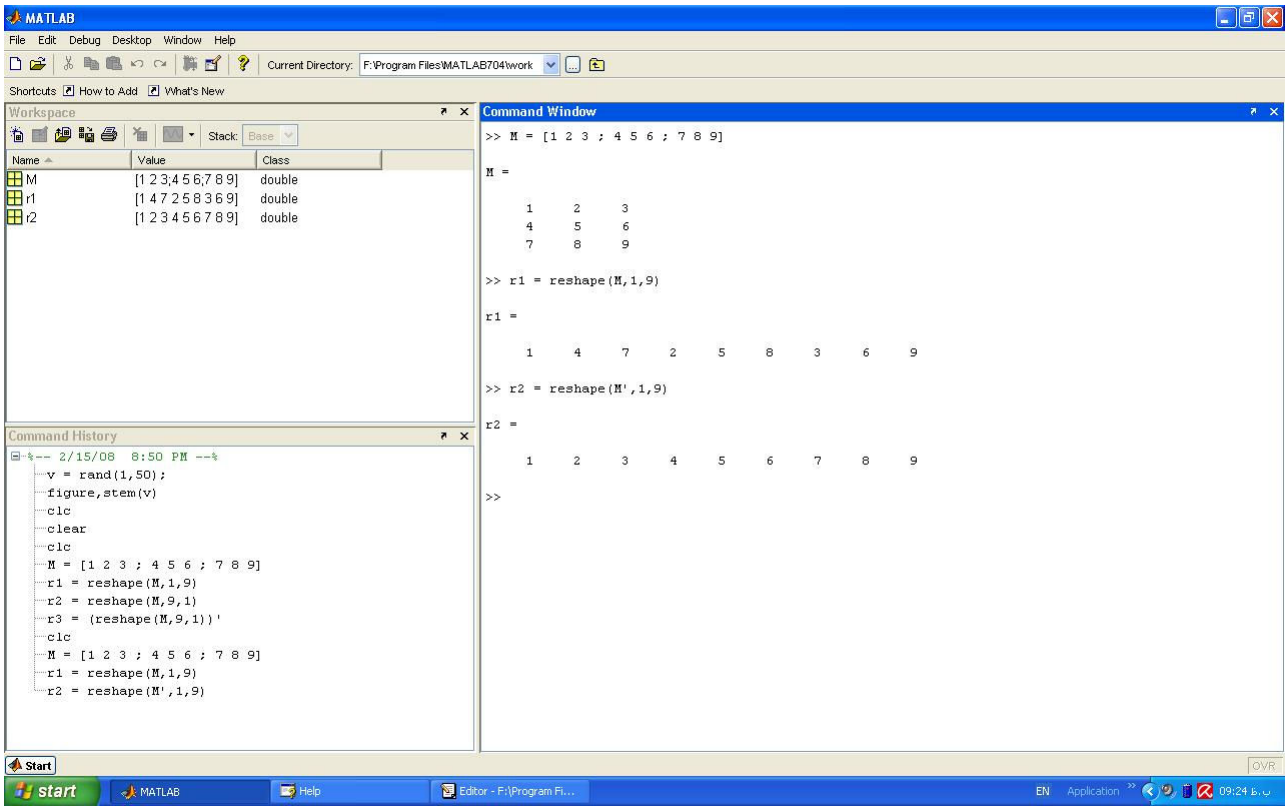


دستور a(1,:) تمام عناصر موجود در سطر ۱ و دستور a(:,3) تمام عناصر موجود در ستون ۳ را نمایش می‌دهد.

عملگر ' : این عملگر ترانهاده (Transpose) یک ماتریس را ایجاد می‌کند:



دستور reshape: به کمک این دستور می‌توانید به ماتریسی که قبلاً تعریف شده، بعدی جدید بدهید:



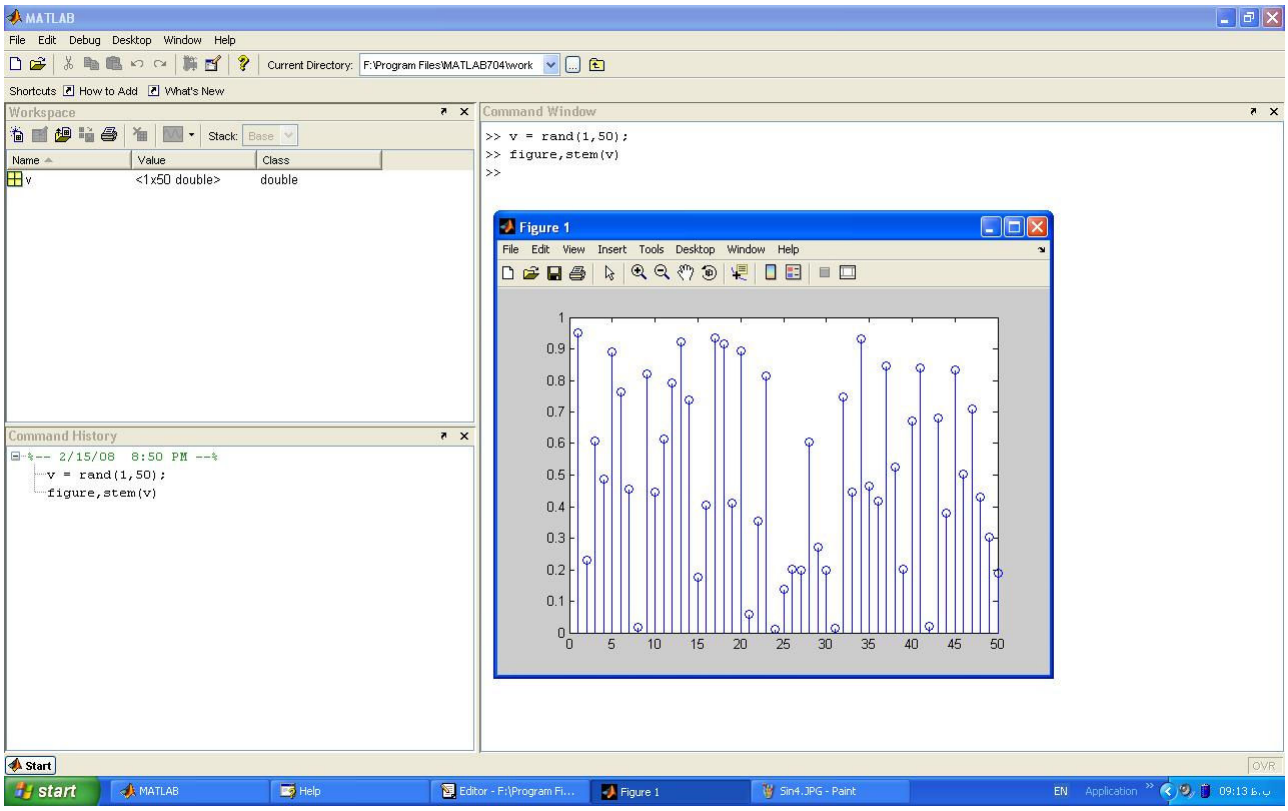
دستور reshape اول، ماتریس M که ۳×۳ است را به ماتریس (آرایه) r1 با بُعد ۹×۱ تبدیل می‌کند. با این کار شما می‌توانید با کنار هم گذاشتن ستونهای ماتریس یک آرایه ایجاد کنید.

دستور reshape دوم، ستونهای ترانهاده ماتریس M (که در واقع سطرهای ماتریس M هستند) را کنار هم گذاشته و ماتریس (آرایه) r2 با بُعد ۹×۱ را می‌سازد.

دستورهای ترسیمی

یکی از نقاط قوت مهم MATLAB، توانایی ترسیم توابع و نتایج محاسبات با ساده‌ترین دستورات است. در زیر به مهمترین دستورات ترسیمی MATLAB می‌پردازیم.

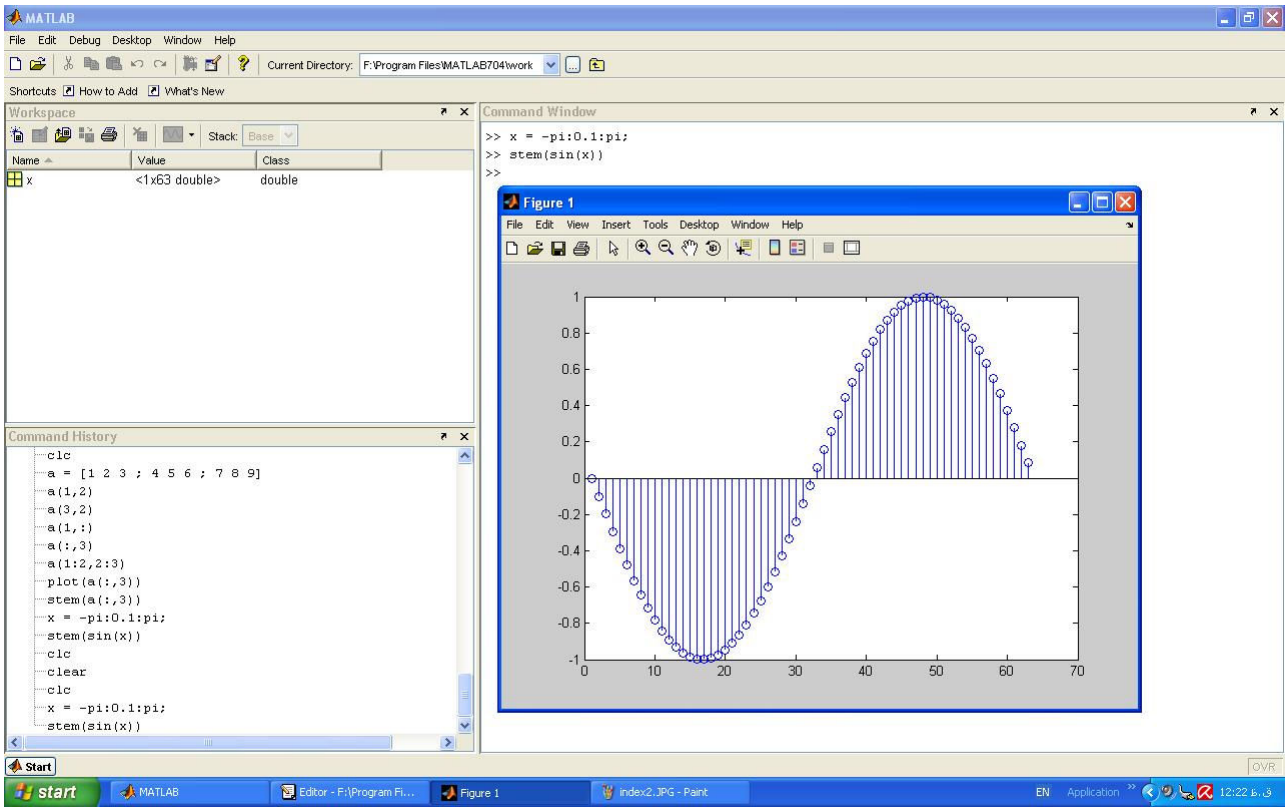
دستور stem : به کمک این دستور می‌توانید مقادیر یک آرایه را به صورت گسسته نمایش دهید:



دستور v = rand(1,50) یک ماتریس ۱×۵۰ با مقادیر تصادفی ایجاد کرده و در متغیر v ذخیره می‌کند.

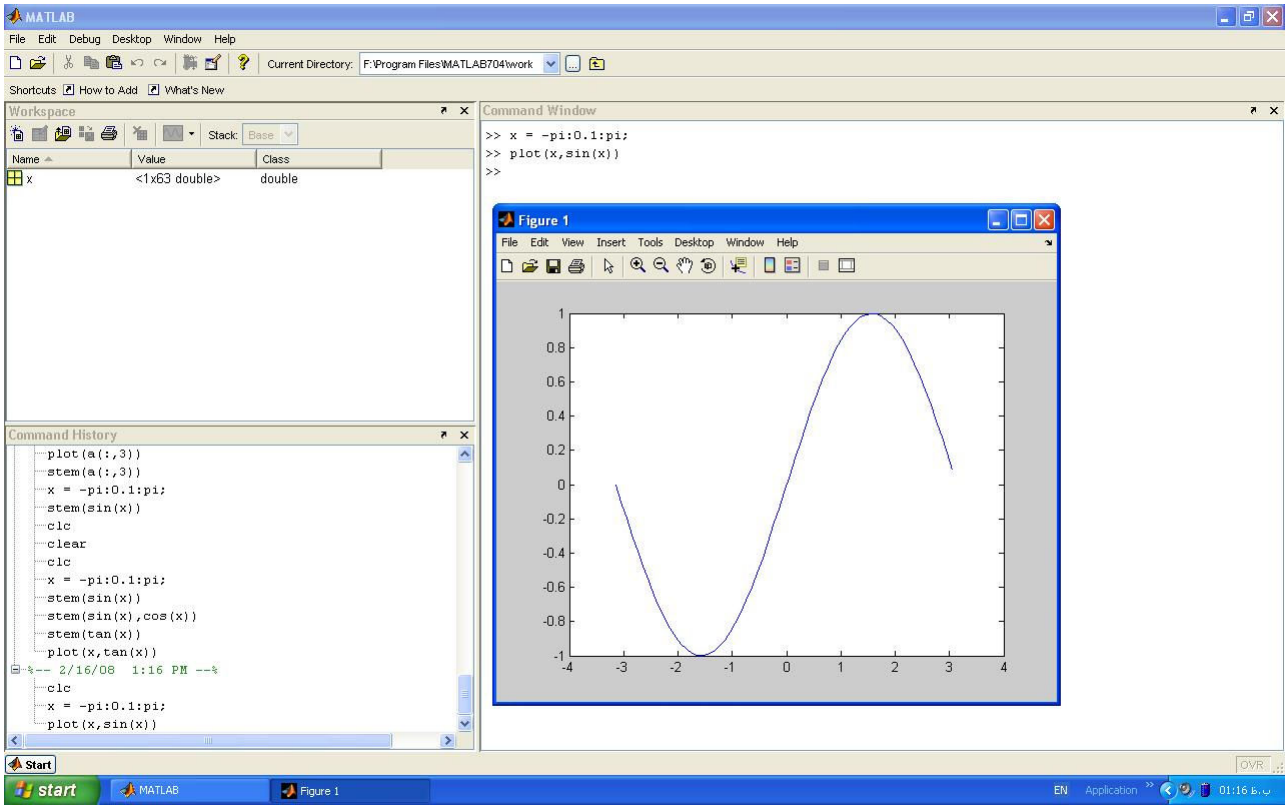
دستور figure یک شکل خالی ایجاد می‌کند که می‌توان در آن هر ترسیمی انجام داد.

دستور stem(v) مقادیر بردار v را به صورت گسسته نمایش می‌دهد.



دستور اول یک بردار از π- تا π+ با فواصل ۰/۱ ایجاد کرده و مقادیر آن (که ۶۳ عدد هستند) را در آرایه x ذخیره می‌کند. نتیجه اجرای دستور stem(sin(x)) را در شکل بالا مشاهده می‌کنید.

دستور plot: متغیرهای MATLAB همگی ماتریسی و گسسته هستند. به کمک دستور plot می‌توان این مقادیر گسسته را به شکل پیوسته نمایش داد. دستور plot(x,y) مقادیر بردار y را بر حسب x به صورت پیوسته نمایش می‌دهد:



با اجرای دستورات ساده می‌توان جزییات بیشتری در شکل رسم شده را نشان داد.

دستور title(' نام شکل ') نامی که در دستور آمده را بالای شکل نمایش می‌دهد.

دستورات xlabel(' نام محور x ') و ylabel(' نام محور y ') نامهای ذکر شده را در کنار محورهای مربوطه نمایش می‌دهد.

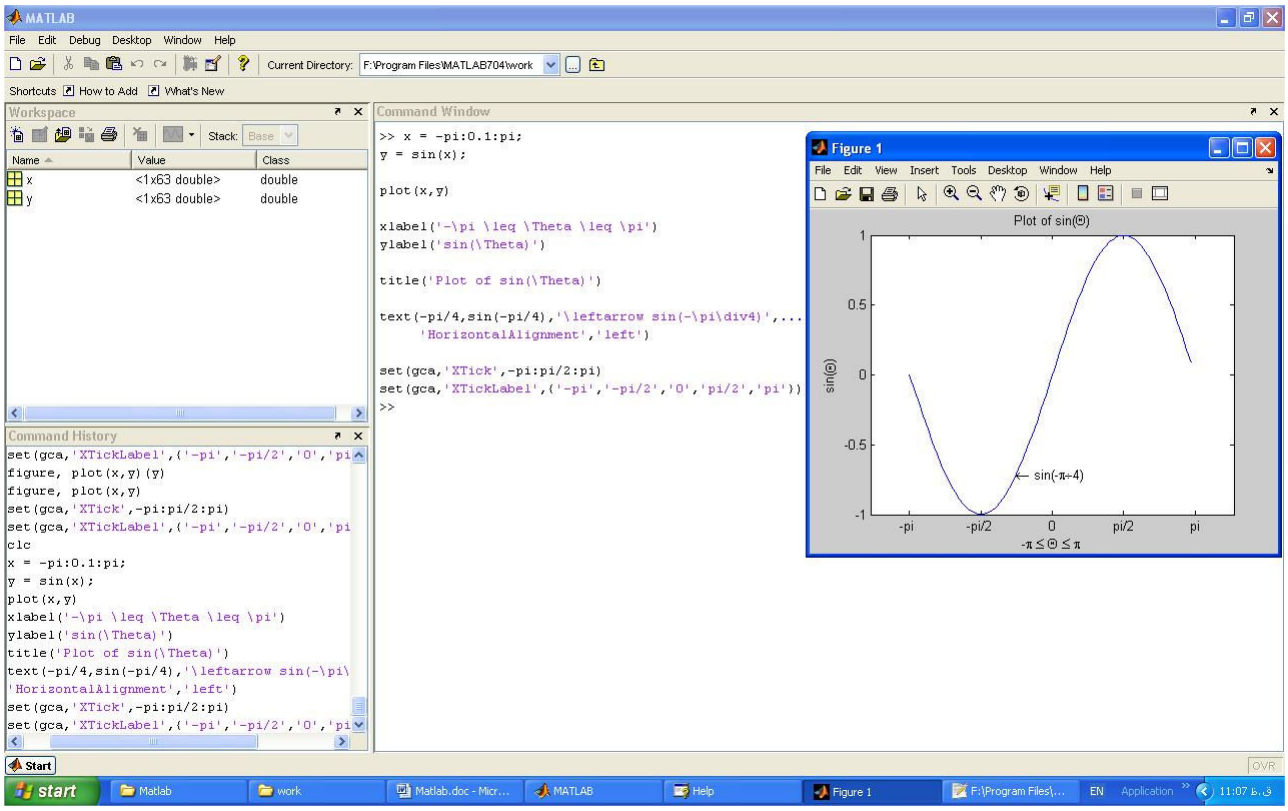
دستور set(gca,'XTick', بردار فواصل) نقاطی که باید در محور x تیک زده شوند را مشخص می‌کند. برای تیک زدن محور y باید از برچسب YTick استفاده کنیم.

دستور set(gca,'XTickLabel', نام تیک‌ها) به تیکهایی که به کمک دستور قبل مشخص شده، نامهای مورد نظر کاربر را اختصاص می‌دهد. برای نامگذاری تیکهای محور y از برچسب YTickLabel استفاده می‌کنیم.

دستور text(x,y,' متن ') متن مشخص شده را در مختصات (x,y) شکل نمایش می‌دهد.

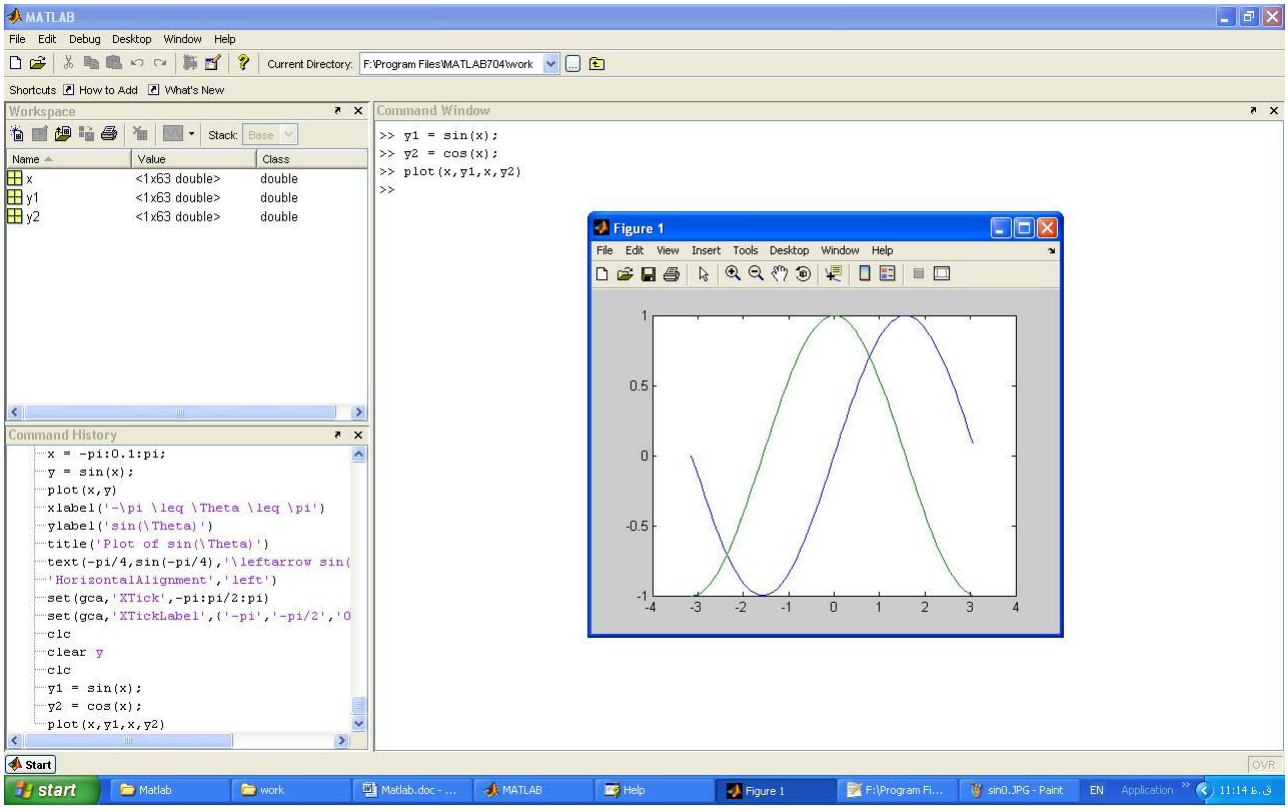
دستور axis([x-min x_max y_min y_max]) مقیاس‌بندی خودکار محورها را به دلخواه کاربر تغییر می‌دهد.

اثر اجرای این دستورات اضافه را در شکل زیر می‌بینید:

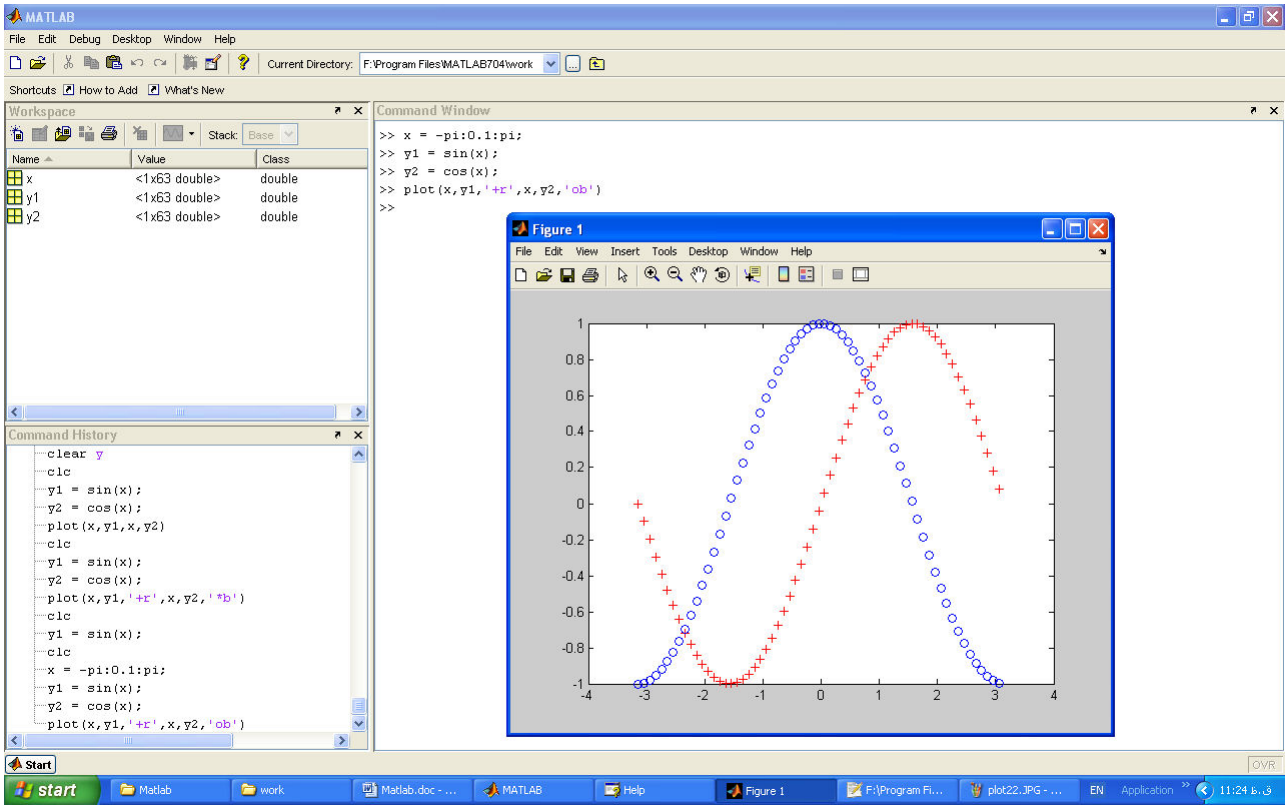


اگر دستوری آن قدر بزرگ باشد که در یک خط جا نشود، با تایپ سه نقطه در انتهای خط، ادامه دستور را در خط بعد پی می‌گیریم (دستور text را در شکل بالا ببینید). به علاوه می‌توان چند دستور را در یک خط قرار داد. برای این کار دستورات را با نشانه ; یا , از هم جدا کنید.

به کمک دستور plot می‌توان چند نمودار را در کنار هم روی یک شکل نمایش داد.



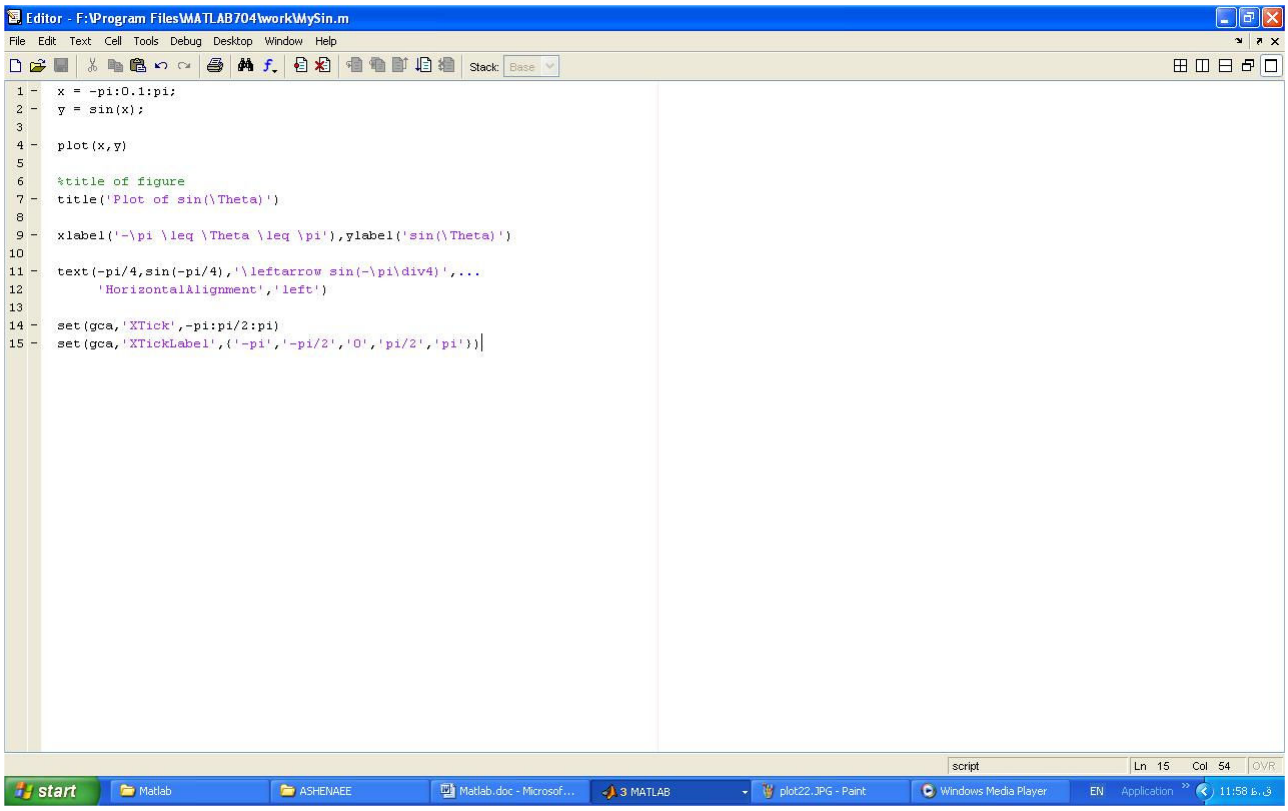
به کمک گزاره‌های ترسیمی می‌توان نحوه نمایش شکل را تغییر داد؛ مثلاً دستور plot(x,y,'+') از نشانه + برای رسم شکل استفاده می‌کند. دستور plot(x,y,'r') شکل را به رنگ قرمز رسم می‌کند. از این تکنیک می‌توان برای متمایز کردن چند نمودار که روی یک شکل رسم شده استفاده کرد (گزاره‌های رنگی عبارتند از: 'r' برای رنگ قرمز، 'g' برای رنگ سبز، 'b' برای رنگ آبی و 'w' برای رنگ سفید).



دستور plot(x,y1,'+r',x,y2,'ob') نمودار y1 برحسب x را به رنگ قرمز و با علامت + و نمودار y2 برحسب x را به رنگ آبی و با علامت o نمایش می‌دهد.

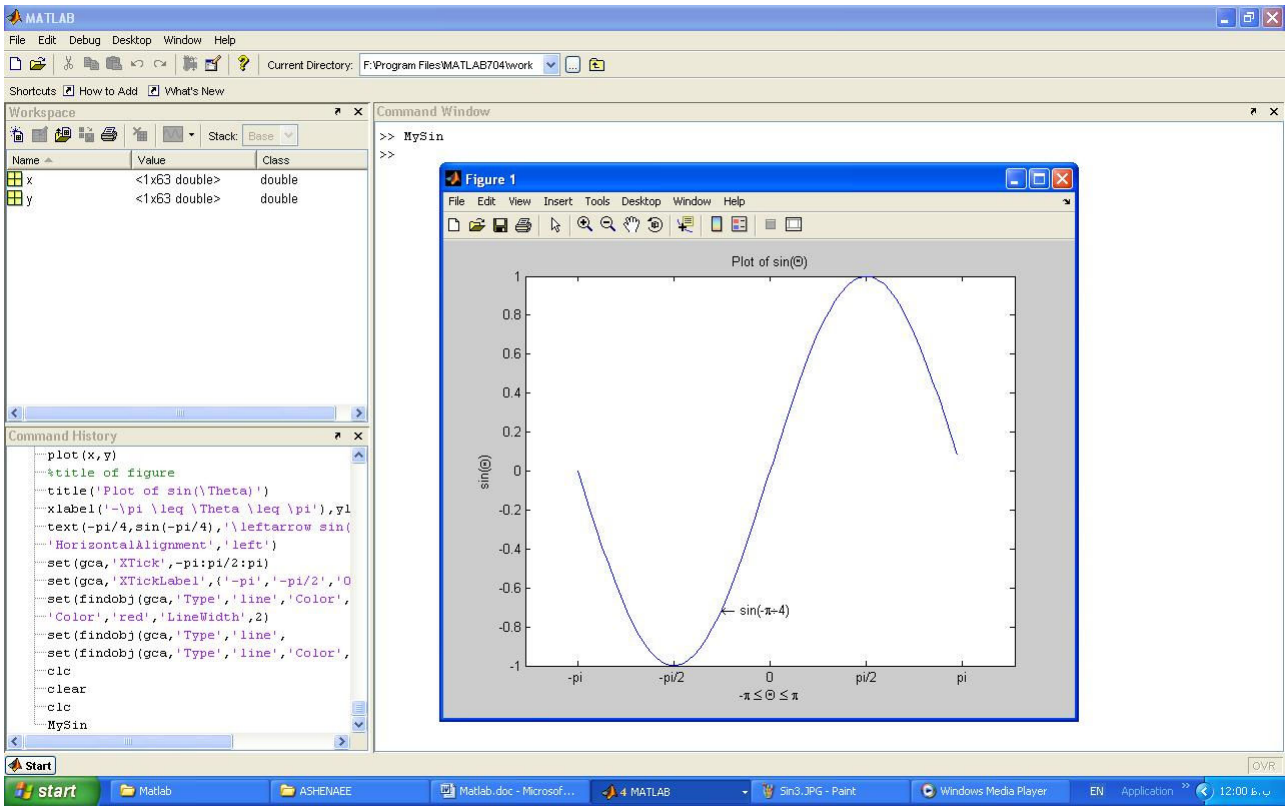
ایجاد m-file

وقتی تعداد دستوراتی که کاربر می‌خواهد اجرا کند زیاد باشد، می‌تواند آنها را در یک فایل متنی بنویسد و با نام دلخواه (که مشابه نام توابع خود MATLAB نباشد) و پسوند m. در شاخه work مسیری که MATLAB در آن مسیر نصب شده ذخیره کند. برای فراخوانی این دستورات کافی است نام آن را در محیط MATLAB تایپ کنید و کلید Enter را فشار دهید:



اگر در ابتدای خطی علامت % قرار داشته باشد، MATLAB آن خط را به‌عنوان دستور در نظر می‌گیرد.

نحوه اجرای m-file بالا در محیط MATLAB به شکل زیر است:



ماتریسهای خاص

در MATLAB توابعی وجود دارد که به کمک آنها می‌توان ماتریسهای خاص را به راحتی ایجاد کرد. مثلاً دستور a = zeros(m,n) یک ماتریس با m سطر و n ستون با عناصر صفر ایجاد می‌کند و نام آن را a می‌گذارد. اگر به جای دستور zeros از دستور ones استفاده کنیم، تمامی عناصر ماتریس مزبور ۱ خواهند بود. دستور eye(n) یک ماتریس مربعی واحد (عناصر روی قطر اصلی ۱ و بقیه صفر هستند) با اندازه n ایجاد می‌کند.

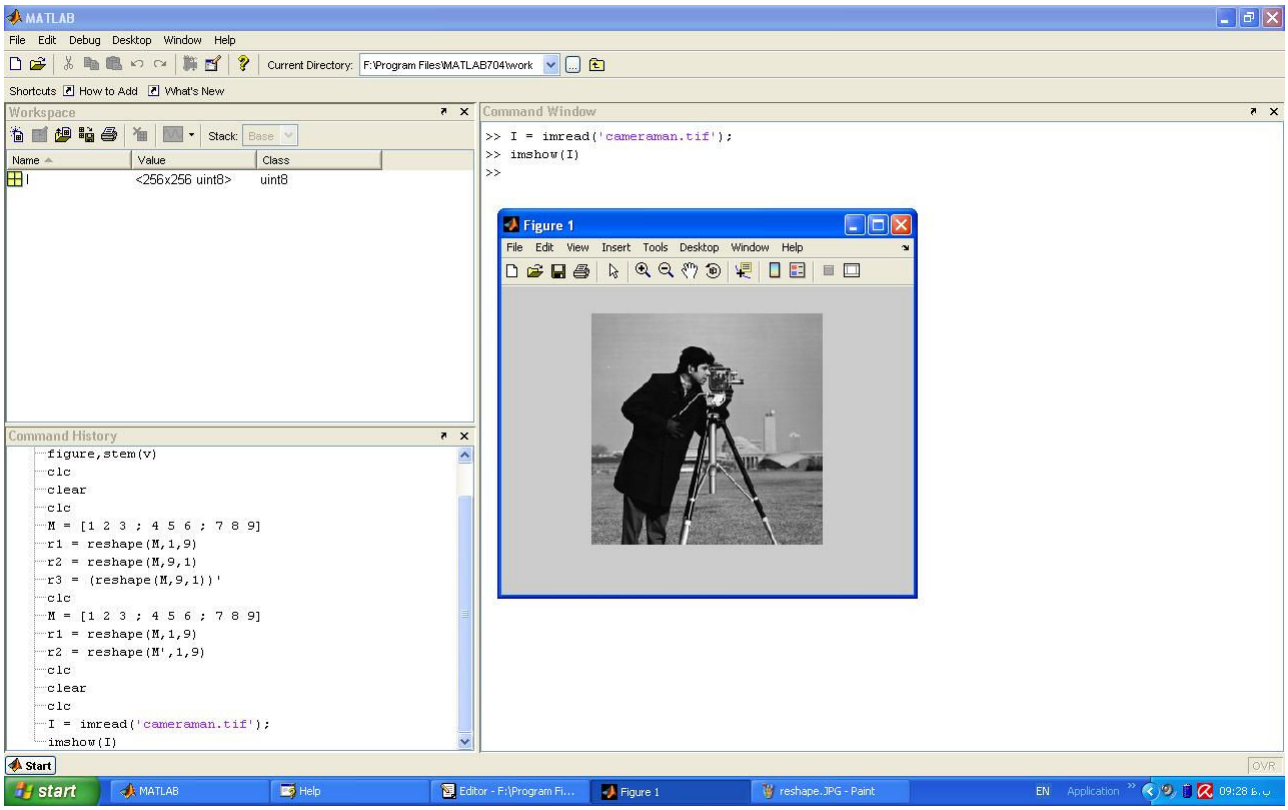
بخش دوم – کاربرد MATLAB در پردازش تصویر

پردازش سیگنال تصویر یکی از مهمترین قابلیت‌های نرم‌افزار MATLAB است. جعبه‌ابزار Image Processing امکانات فراوانی برای پردازش تصاویر سیاه و سفید و رنگی در اختیار ما می‌گذارد. در ادامه به توابع پایه برای پردازش تصاویر سیاه و سفید خواهیم پرداخت.

سیگنال تصویر

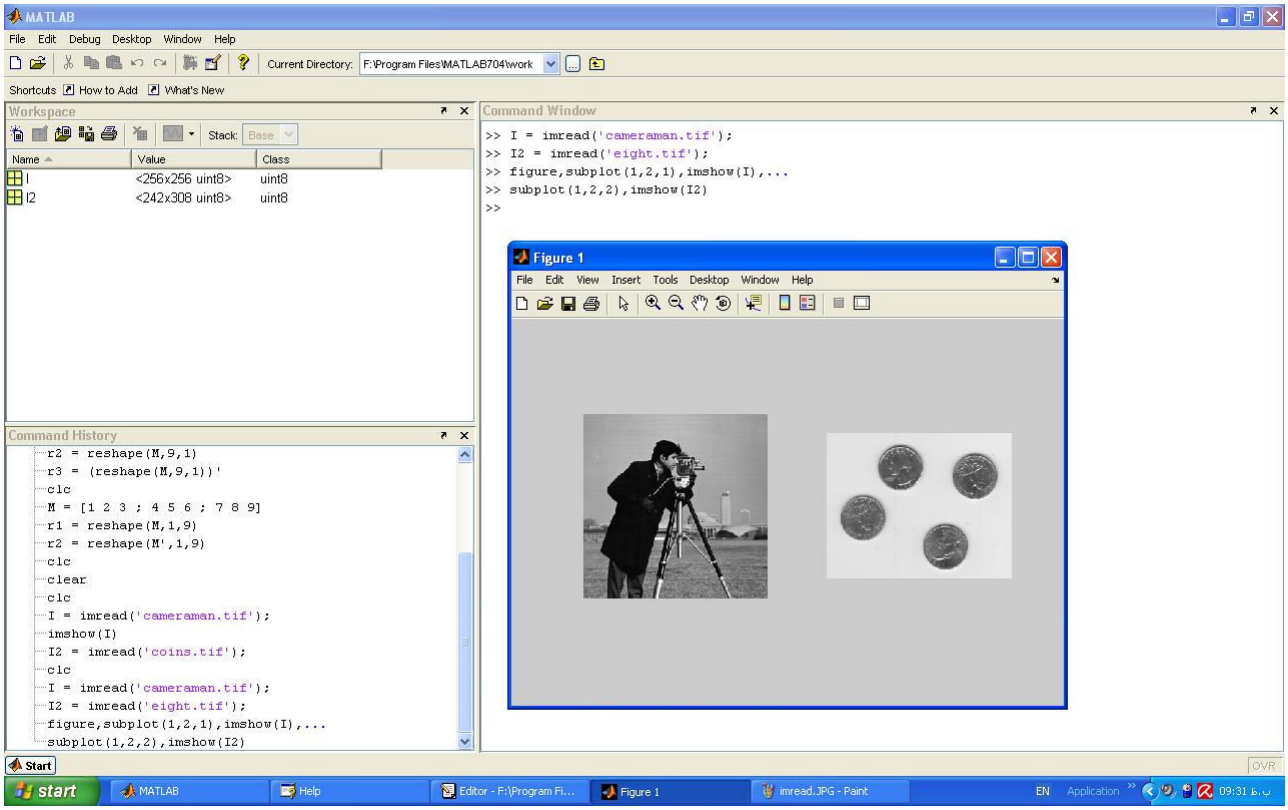
در یک تصویر سیاه و سفید، هر نقطه یا پیکسل تصویر(Pixel) یک سطح روشنایی بین صفر تا ۲۵۵ دارد که سطح صفر مربوط به رنگ سیاه و سطح ۲۵۵ مربوط به رنگ سفید است. بنابراین می‌توان سطح روشنایی هر پیکسل تصویر را با یک عدد ۸ بیتی نمایش داد.

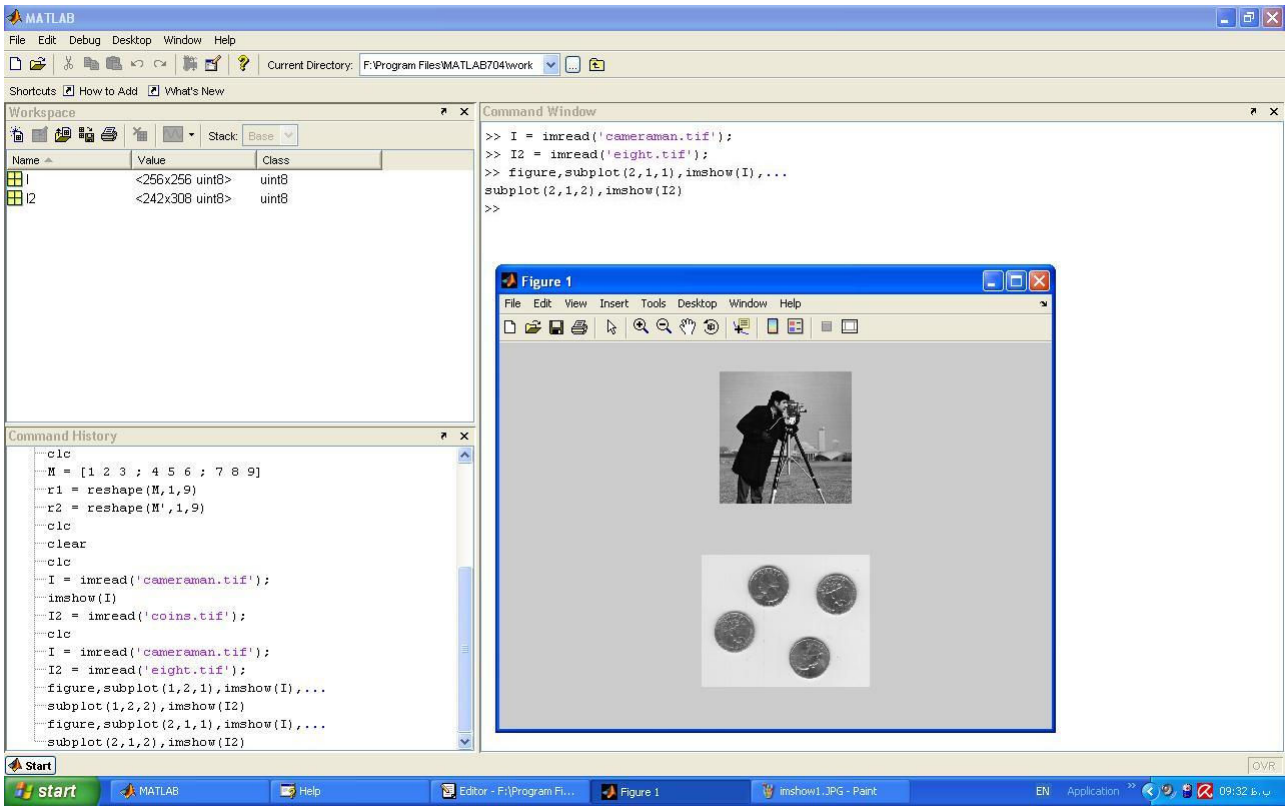
یک تصویر سیاه و سفید با اندازه m×n (مثلاً ۴۸۰×۶۴۰)، در واقع یک ماتریس از پیکسلها با m سطر و n ستون است که مقدار هر خانه ماتریس (یک بایت) نشان‌دهنده سطح روشنایی آن پیکسل تصویر می‌باشد. به کمک دستور imread (image read) می‌توان یک تصویر را به صورت یک ماتریس وارد MATLAB کرد. تابع imshow این ماتریس را به صورت تصویر نمایش می‌دهد:

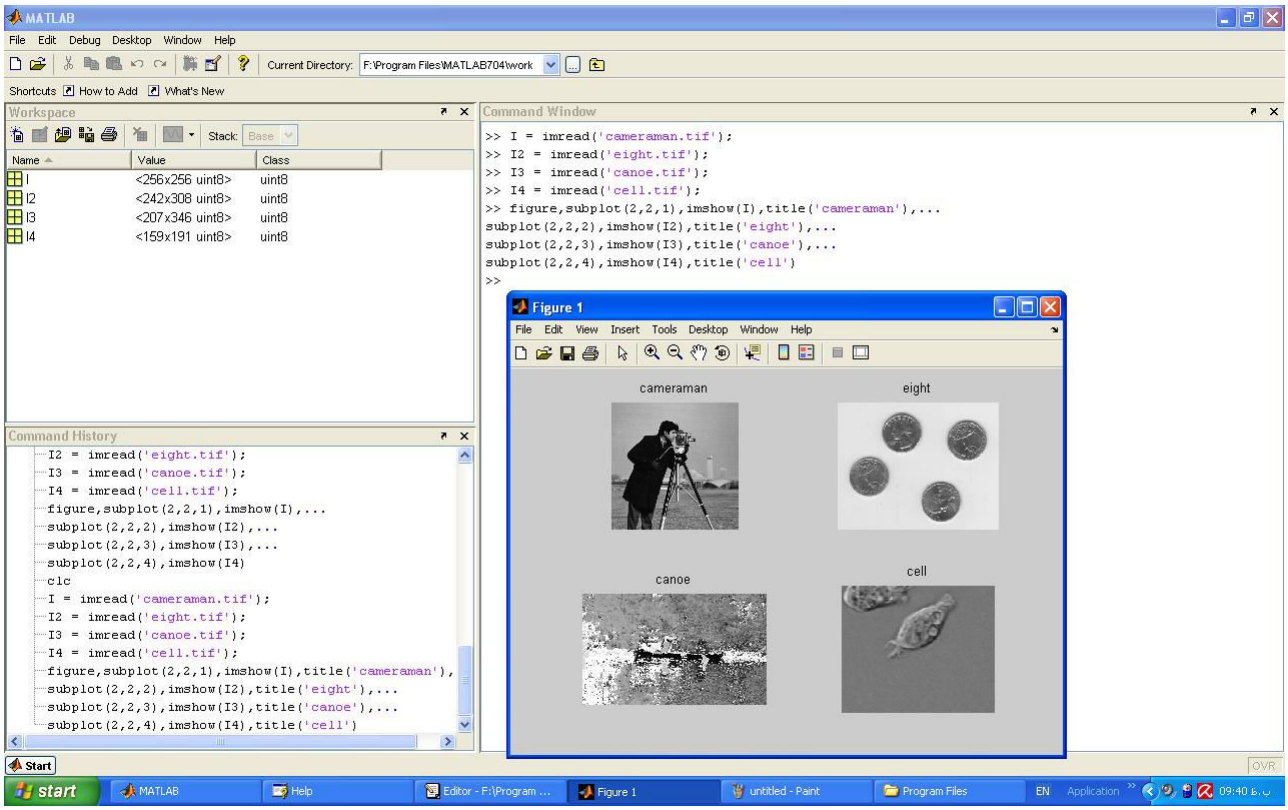


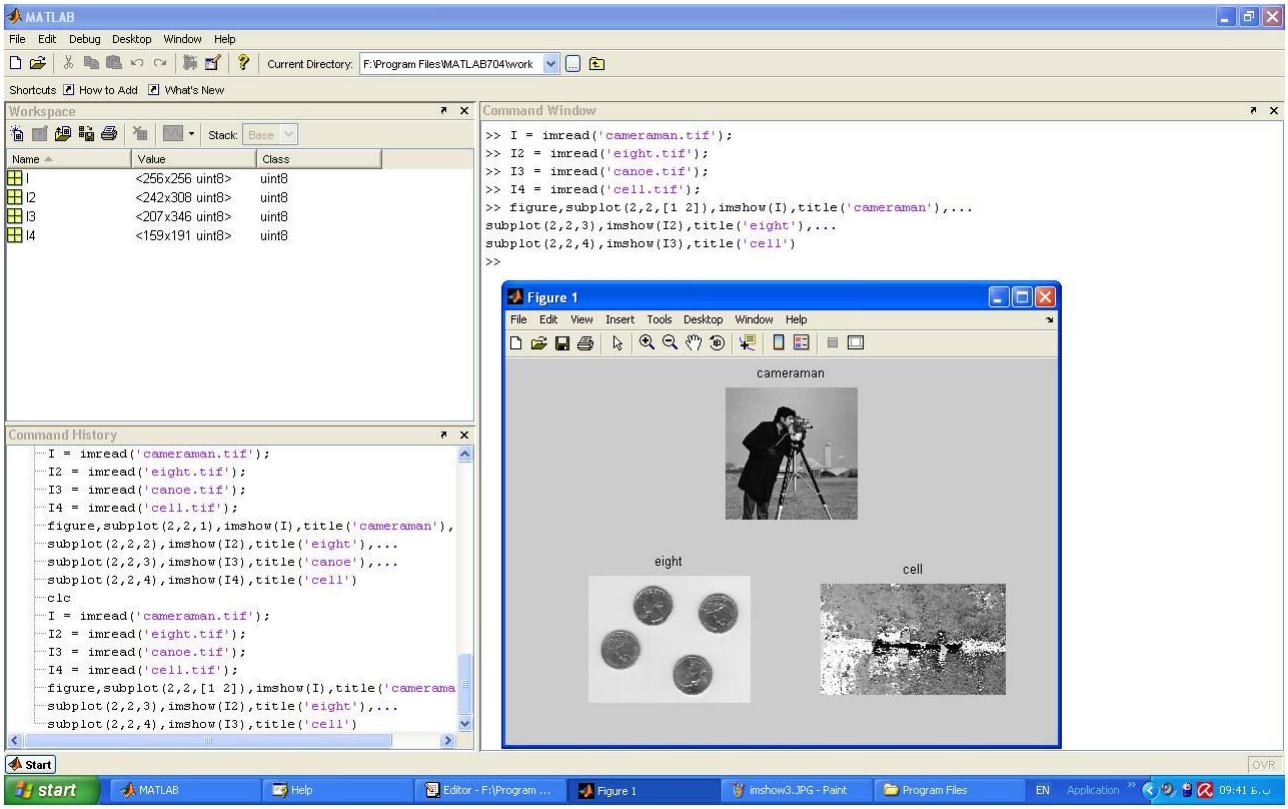
عناصر یک تصویر می‌تواند از نوع uint8 (یک بایت که می‌تواند بین صفر تا ۲۵۵ باشد) یا از نوع double (عدد حقیقی بین صفر تا یک) باشد.

به کمک تابع subplot می‌توان چند ترسیم را روی یک شکل انجام داد:









پردازش سیگنالها و سیستم‌های دوبعدی نیاز به معلومات بیشتری دارد. بنابراین به کمک تابع reshape تصویر را به یک سیگنال یک‌بعدی تبدیل می‌کنیم و پس از پردازش به کمک همین تابع دوباره تصویر را به اندازه اصلی برمی‌گردانیم.

دستور conv

به کمک این دستور می‌توان یک فیلتر را روی یک سیگنال (مثلاً یک تصویر) اعمال کرد. این کار به معنی کانوالو کردن (convolution) سیگنال تصویر با پاسخ ضربه فیلتر است. قطعه کد زیر یک تصویر ۲۵۶×۲۵۶ به نام cameraman.tif که همراه با MATLAB ارائه می‌شود را خوانده و یک فیلتر متوسط‌گیر با پاسخ ضربه $h=[\frac{1}{3},\frac{1}{3},\frac{1}{3}]$ را روی آن اعمال کرده و نتیجه را نمایش می‌دهد. به نحوه reshape کردن توجه کنید:

```
Im = imread ( ' cameraman.tif ' );
I = im2double ( Im );

% size of this image is 256*256
Vector = reshape ( I , 1 , 256*256 ); %Image as a vector

h1 = [1/3 1/3 1/3]; % or "h1 = ones(1,3)/3"

Filt = conv ( Vector , h1 );

F = Filt ( 2 : end-1 );

Out = reshape ( F , 256 , 256 );  vector as image

imshow ( Out );
```

چون دستور conv فقط می‌تواند روی داده‌های از نوع double عمل کند، به کمک تابع im2double تصویر خوانده شده را به این نوع داده تبدیل می‌کنیم.

کانوالو کردن یک فیلتر با پاسخ ضربه به طول n، باعث می‌شود n-1 واحد به نتیجه کانولوشن اضافه شود. بنابراین برای اینکه اندازه تصویر خروجی با تصویر اصلی یکی باشد، به کمک دستور F = Filt (2 : end-1) یک پیکسل از ابتدا و یک پیکسل از انتهای بردار خروجی کم می‌کنیم.

تابع reshape اول برای تبدیل تصویر دوبعدی به یک بردار یک‌بعدی و reshape دوم برای تبدیل نتیجه پردازش که یک بردار یک‌بعدی است به ماتریس دوبعدی تصویر به کار می‌رود. با استفاده از قطعه کد بالا، ستونهای تصویر به دنبال هم چیده شده و پردازش می‌شوند. می‌توانید به جای این کار، سطرهای تصویر را به دنبال هم بچینید (به کمک آنچه قبلاً در قسمت reshape دیدیم) و پردازش کنید. این دو تکنیک را در کانولوشن تصویر با یک فیلتر بالاگذر با پاسخ ضربه $h=[-1,2,-1]$ به کار ببرید و نتایج را با هم مقایسه کنید.

پردازش تصویر در حوزه فرکانس

برای پردازش تصویر در حوزه فرکانس، ابتدا آن را به یک سیگنال یک‌بعدی تبدیل می‌کنیم و به کمک تابع fft از آن تبدیل فوریه (Fast Fourier Transform) می‌گیریم. نتیجه تابع fft یک بردار مختلط است که به کمک توابع abs و angle می‌توان مقادیر دامنه و فاز آن را به دست آورد.

قطعه کد زیر اصول پردازش سیگنال تصویر به صورت یک بعدی در فضای فرکانسی را نشان می‌دهد:

```
Im = imread('eight.tif');
I = im2double(Im);

[row col] = size(I);
% size of this image is row*col
x = reshape(I,1,row*col); %Image as a vector

% Fourier Transform
X = fft(x);
X = fftshift(X);
```

```
% Amplitude of Fourier Transform
absX = abs(X);

% Phase of Fourier Transform
angX = angle(X);
% High-pass filter
H = zeros(1,row*col);
H(1 : row*col/3) = 1;
H(end - row*col/3 : end) = 1;

absY = H .* absX;

Y = absY .* exp(j*angX);
Y = fftshift(Y);

y = ifft(Y);
y = real(y);

Out = reshape(y,row,col);

imshow(Out),title('High-pass Filter');
```

تابع fftshift مولفه‌های کم‌فرکانس فوریه را به مرکز طیف منتقل می‌کند تا تحلیل آن ساده‌تر باشد. همان‌گونه که در قطعه کد بالا می‌بینید، H اندازه پاسخ فرکانسی یک فیلتر بالاگذر ایده‌آل است که در اندازه تبدیل فوریه تصویر ضرب شده است. Y تبدیل فوریه‌ای است که از ترکیب اندازه جدید با فاز تبدیل فوریه اصلی به دست می‌آید ($Y = absY.e^{j.angX}$). با اعمال مجدد تابع fftshift مؤلفه‌های فرکانسی را به مکان اصلیشان برمی‌گردانیم و به کمک تابع ifft (Inverse FFT) تبدیل فوریه جدید را به حوزه زمان برگردانده و قسمت حقیقی آن که همان تصویر خروجی است را به کمک تابع reshape به ماتریس تصویر تبدیل می‌کنیم.

توابع جعبه‌ابزار پردازش تصویر MATLAB

Image Display
colorbar Display color bar (MATLAB function)
image Create and display image object (MATLAB function)
imagesc Scale data and display as image (MATLAB function)
immovie Make movie from multiframe indexed image
imshow Display image in a MATLAB figure window
imtool Display image in the Image Viewer
montage Display multiple image frames as rectangular montage
subimage Display multiple images in single figure
warp Display image as texture-mapped surface

Image File I/O
dicomanon Anonymize a DICOM file
dicomdict Specify which DICOM data dictionary to use
dicominfo Read metadata from a DICOM message
dicomread Read a DICOM image
dicomuid Generate DICOM unique identifier
dicomwrite Write a DICOM image
dicom-dict.txt Text file containing DICOM data dictionary
imfinfo Return information about image file (MATLAB function)
imread Read image file (MATLAB function)
imwrite Write image file (MATLAB function)

Image Types and Type Conversions
dither Convert image using dithering
double Convert data to double precision (MATLAB function)
gray2ind Convert intensity image to indexed image
grayslice Create indexed image from intensity image by thresholding
graythresh Compute global image threshold using Otsu's method
im2bw Convert image to binary image by thresholding
im2double Convert image array to double precision
im2int16 Convert image array to 16-bit signed integer
im2java Convert image to instance of Java image object (MATLAB function)
im2java2d Convert image to instance of Java buffered image object
im2single Convert image array to single precision
im2uint16 Convert image array to 16-bit unsigned integers
im2uint8 Convert image array to 8-bit unsigned integers
ind2gray Convert indexed image to intensity image
ind2rgb Convert indexed image to RGB image
int16 Convert data to signed 16-bit integers (MATLAB function)
label2rgb Convert a label matrix to an RGB image
mat2gray Convert matrix to intensity image
rgb2gray Convert RGB image or colormap to grayscale
rgb2ind Convert RGB image to indexed image
uint16 Convert data to unsigned 16-bit integers (MATLAB function)
uint8 Convert data to unsigned 8-bit integers (MATLAB function)

Modular Tool Creation Functions
imageinfo Image Information tool
imcontrast Adjust Contrast tool
imdisplayrange Display range tool
impixelinfo Pixel Information tool
impixelinfoval Pixel Information tool, without text label
impixelregion Pixel Region tool
impixelregionpanel Pixel Region scroll panel

Navigational tools
immagbox Image Information tool
imoverview Adjust Contrast tool
imoverviewpanel Display range tool
imscrollpanel Pixel Information tool

Utility Functions
axes2pix Convert axes coordinate to pixel coordinate
getimage Get image data from axes
getimagemodel Retrieve imagemodel objects from image handles
imattributes Return information about image attributes
imgca Get handle to current image axes
imgcf Get handle to current image figure
imgetfile Image Open File dialog box
imhandles Get all image handles
impositionrect Create position rectangle
iptaddcallback Add function handle to callback list
iptcheckhandle Check validity of handle
iptgetapi Get Application Programmer Interface from a handle
ipticondir Directories containing IPT and MATLAB icons
iptremovecallback Delete function handle from callback list
iptwindowalign Align figure windows
truesize Adjust display size of image

Spatial Transformations
checkerboard Create checkerboard image
findbounds Find output bounds for spatial transformation
fliptform Flip the input and output roles of a TFORM structure
imcrop Crop image

imresize Resize image
imrotate Rotate image
imtransform Apply 2-D spatial transformation to image
makeresampler Create resampling structure
maketform Create geometric transformation structure
tformarray Geometric transformation of a multidimensional array
tformfwd Apply forward geometric transformation
tforminv Apply inverse geometric transformation

Image Registration
cp2tform Infer geometric transformation from control point pairs
cpcorr Tune control point locations using cross-correlation
cpselect Control point selection tool
cpstruct2pairs Convert CPSTRUCT to valid pairs of control points
normxcorr2 Normalized two-dimensional cross-correlation

Image Analysis
bwboundaries Trace region boundaries in binary image
bwtraceboundary Trace object in binary image
edge Find edges in intensity image
hough Hough transform
houghlines Extract line segments based on the Hough transform
houghpeaks Identify peaks in the Hough transform
qtdecomp Perform quadtree decomposition
qtgetblk Get block values in quadtree decomposition
qtsetblk Set block values in quadtree decomposition

Texture Analysis
entropy Entropy of an intensity image
entropyfilt Local entropy of an intensity image
graycomatrix Gray-level co-occurrence matrix
graycoprops Properties of a gray-level co-occurrence matrix
rangefilt Local range of an image
stdfilt Local standard deviation of an image

Pixel Values and Statistics
corr2 Compute 2-D correlation coefficient
imcontour Create contour plot of image data
imhist Display histogram of image data
impixel Determine pixel color values
improfile Compute pixel-value cross-sections along line segments
mean2 Compute mean of matrix elements
pixval Display information about image pixels
regionprops Measure properties of image regions
std2 Compute standard deviation of matrix elements

Image Arithmetic
imabsdiff Compute absolute difference of two images
imadd Add two images, or add constant to image
imcomplement Complement image
imdivide Divide two images, or divide image by constant
imlincomb Compute linear combination of images
immultiply Multiply two images, or multiply image by constant
imsubtract Subtract two images, or subtract constant from image

Image Enhancement
adapthisteq Perform adaptive histogram equalization using CLAHE
decorrstretch Apply a decorrelation stretch to a multi-channel image
histeq Enhance contrast using histogram equalization
imadjust Adjust image intensity values or colormap
imnoise Add noise to an image
intlut Compute new array values based on lookup table (LUT)

medfilt2 Perform 2-D median filtering
ordfilt2 Perform 2-D order-statistic filtering
stretchlim Return a pair of intensities that can be used to increase the contrast of an image
wiener2 Perform 2-D adaptive noise-removal filtering

Image Restoration (Deblurring)
deconvblind Restore image using blind deconvolution
deconvlucy Restore image using accelerated Richardson-Lucy algorithm
deconvreg Restore image using regularized filter
deconvwnr Restore image using Wiener filter
edgetaper Taper the discontinuities along the image edges
otf2psf Convert optical transfer function to point spread function
psf2otf Convert point spread function to optical transfer function

Linear Filtering
conv2 Perform 2-D convolution (MATLAB function)
convmtx2 Compute 2-D convolution matrix
convn Perform N-D convolution (MATLAB function)
filter2 Perform 2-D filtering (MATLAB function)
fspecial Create predefined filters
imfilter Multidimensional image filtering

Linear 2-D Filter Design
freqspace Determine 2-D frequency response spacing (MATLAB function.)
freqz2 Compute 2-D frequency response
fsamp2 Design 2-D FIR filter using frequency sampling
ftrans2 Design 2-D FIR filter using frequency transformation
fwind1 Design 2-D FIR filter using 1-D window method
fwind2 Design 2-D FIR filter using 2-D window method

Image Transforms
dct2 Compute 2-D discrete cosine transform
dctmtx Compute discrete cosine transform matrix
fan2para Convert fan-beam projection data to parallel-beam
fanbeam Compute fan-beam transform
fft2 Compute 2-D fast Fourier transform (MATLAB function)
fftn Compute N-D fast Fourier transform (MATLAB function)
fftshift Reverse quadrants of output of FFT (MATLAB function)
idct2 Compute 2-D inverse discrete cosine transform
ifft2 Compute 2-D inverse fast Fourier transform (MATLAB function)
ifftn Compute N-D inverse fast Fourier transform (MATLAB function)
ifanbeam Compute inverse fan-beam transform
iradon Compute inverse Radon transform
para2fan Convert parallel-beam projections to fan-beam
phantom Generate a head phantom image
radon Compute Radon transform

Intensity and Binary Images
conndef Default connectivity array
imbothat Perform bottom-hat filtering
imclearborder Suppress light structures connected to image border
imclose Close image
imdilate Dilate image
imerode Erode image
imextendedmax Find extended-maxima transform
imextendedmin Find extended-minima transform
imfill Fill image regions
imhmax Calculate H-maxima transform
imhmin Calculate H-minima transform
imimposemin Impose minima
imopen Open image
imreconstruct Perform morphological reconstruction

imregionalmax Find regional maxima of image
imregionalmin Find regional minima of image
imtophat Perform tophat filtering
watershed Find image watershed regions

Binary Images
applylut Perform neighborhood operations using lookup tables
bwarea Area of objects in binary image
bwareaopen
bwdist Distance transform
bweuler Euler number of binary image
bwhitmiss Binary hit-and-miss operation
bwlabel Label connected components in 2-D binary image
bwlabeln Label connected components in N-D binary image
bwmorph Perform morphological operations on binary image
bwpack Pack binary image
bwperim Find perimeter of objects in binary image
bwselect Select objects in binary image
bwulterode Ultimate erosion
bwunpack Unpack a packed binary image
imregionalmin Regional minima of image
imtophat Perform tophat filtering
makelut Construct lookup table for use with applylut

Structuring Element (STREL) Creation and Manipulation
getheight Get the height of a structuring element
getneighbors Get structuring element neighbor locations and heights
getnhood Get structuring element neighborhood
getsequence Extract sequence of decomposed structuring elements
isflat Return true for flat structuring element
reflect Reflect structuring element
strel Create morphological structuring element
translate Translate structuring element

Region-Based Processing
poly2mask Convert region-of-interest polygon to mask
roicolor Select region of interest, based on color
roifill Smoothly interpolate within arbitrary region
roifilt2 Filter a region of interest
roipoly Select polygonal region of interest

Neighborhood and Block Processing
bestblk Choose block size for block processing
blkproc Implement distinct block processing for image
col2im Rearrange matrix columns into blocks
colfilt Perform neighborhood operations using column-wise functions
im2col Rearrange image blocks into columns
nlfilter Perform general sliding-neighborhood operations

Colormap Manipulation
brighten Brighten or darken colormap (MATLAB function)
cmpermute Rearrange colors in colormap
cmunique Find unique colormap colors and corresponding image
colormap Set or get color lookup table (MATLAB function)
imapprox Approximate indexed image by one with fewer colors
rgbplot Plot RGB colormap components (MATLAB function)

Color Space Conversions
applycform Apply device-independent color space transformation
hsv2rgb Convert HSV values to RGB color space (MATLAB function)
iccread Read ICC color profile
iccwrite Write ICC color profile

isicc Determine if ICC color profile is valid
lab2double Convert color values to double
lab2uint16 Convert color values to uint16
lab2uint8 Convert color values to uint8
makecform Create device-independent color space transform structure
ntsc2rgb Convert NTSC values to RGB color space
rgb2hsv Convert RGB values to HSV color space (MATLAB function)
rgb2ntsc Convert RGB values to NTSC color space
rgb2ycbcr Convert RGB values to YCbCr color space
whitepoint Returns XYZ values of standard illuminants
xyz2double Convert XYZ color values to double
xyz2uint16 Convert XYZ color values to uint16
ycbcr2rgb Convert YCbCr values to RGB color space

Toolbox Preferences
iptgetpref Get value of Image Processing Toolbox preference
iptsetpref Set value of Image Processing Toolbox preference

Toolbox Utility Functions
getrangefromclass Get dynamic range of image based on its class
iptcheckconn Check validity of connectivity argument
iptcheckinput Check validity of array
iptcheckmap Check validity of colormap
iptchecknargin Check number of input arguments
iptcheckstrs Check validity of strings
iptnum2ordinal Convert positive integer to ordinal string

Interactive Mouse Utility Functions
getline Select polyline with mouse
getpts Select points with mouse
getrect Select rectangle with mouse

Array Operations
circshift Shift array circularly (MATLAB function)
padarray Pad an array

Demos
iptdemos Display index of Image Processing Toolbox demos

Performance
ippl Check for presence of Intel Performance Primitives Library (IPPL)

بخش سوم – تحلیل سیستمهای کنترلی به کمک MATLAB

یکی از قابلیتهای مهم MATLAB، کاربرد آن در تحلیل و طراحی سیستمهای کنترلی است. رسم نمودارهای مختلف و تحلیل نحوه عملکرد سیستمهای کنترل خطی بدون کمک کامپیوتر بسیار دشوار و بعضاً غیرممکن است و MATLAB در این میان به عنوان یک ابزار ریاضی مهندسی، کمک فراوانی به مهندسین کنترل جهت تحلیل و طراحی سیستم‌ها می‌نماید. در ذیل با اصول اساسی تحلیل سیستمهای کنترلی به کمک MATLAB آشنا می‌شویم.

تعریف تابع تبدیل در حوزه لاپلاس

می‌دانید که تابع تبدیل سیستمهای LTI در حوزه لاپلاس گویا و شامل دو چندجمله‌ای برحسب s در صورت و مخرج است:

$$H(s) = \frac{B(s)}{A(s)} = \frac{b_1 s^{n-1} + \cdots + b_{n-1} s + b_n}{a_1 s^{m-1} + \cdots + a_{m-1} s + a_m}$$

این چند جمله‌ای‌ها را می‌توان به صورت حاصل‌ضرب صفرها و قطبها نوشت:

$$H(s) = \frac{Z(s)}{P(s)} = k\frac{(s-z_1)(s-z_2)\cdots(s-z_m)}{(s-p_1)(s-p_2)\cdots(s-p_n)}$$

که k بهره سیستم، Z_i صفرهای سیستم و P_i قطبهای سیستم هستند.

در MATLAB می‌توان یک تابع تبدیل را در هر دو شکل تعریف کرد.

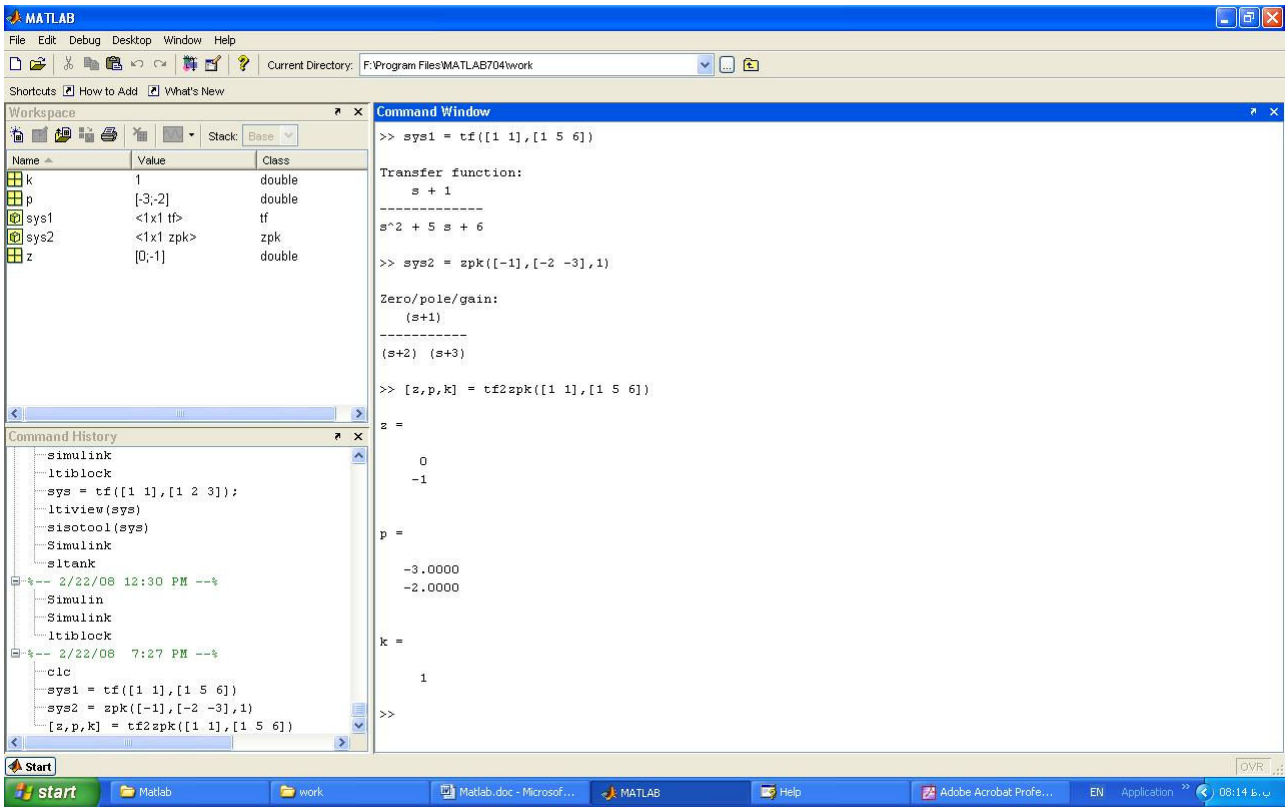
دستور tf (Transfer Function)

به کمک دستور *tf ([بردار ضرایب چند جمله‌ای مخرج] , [بردار ضرایب چندجمله‌ای صورت])* می‌توان یک تابع تبدیل کـه فـرم آن مانند شکل اول باشد را تعریف کرد.

دستور zpk (Zero-Pol-Gain)

به کمک دستور *zpk ([بردار مقادیر صفر] , [بردار مقادیر قطب] , بهره)* می‌توان یک تابع تبدیل به فرم دوم را تعریف نمود.

شکل زیر نحوه به کار بردن توابع فوق و تبدیل آنها به یکدیگر را نشان می‌دهد:



تجزیه تابع تبدیل و عکس تبدیل لاپلاس

برای تجزیه تابع تبدیل به کسرهای جزیی برای سهولت محاسبه عکس تبدیل لاپلاس، از تابع residue استفاده می‌شود.

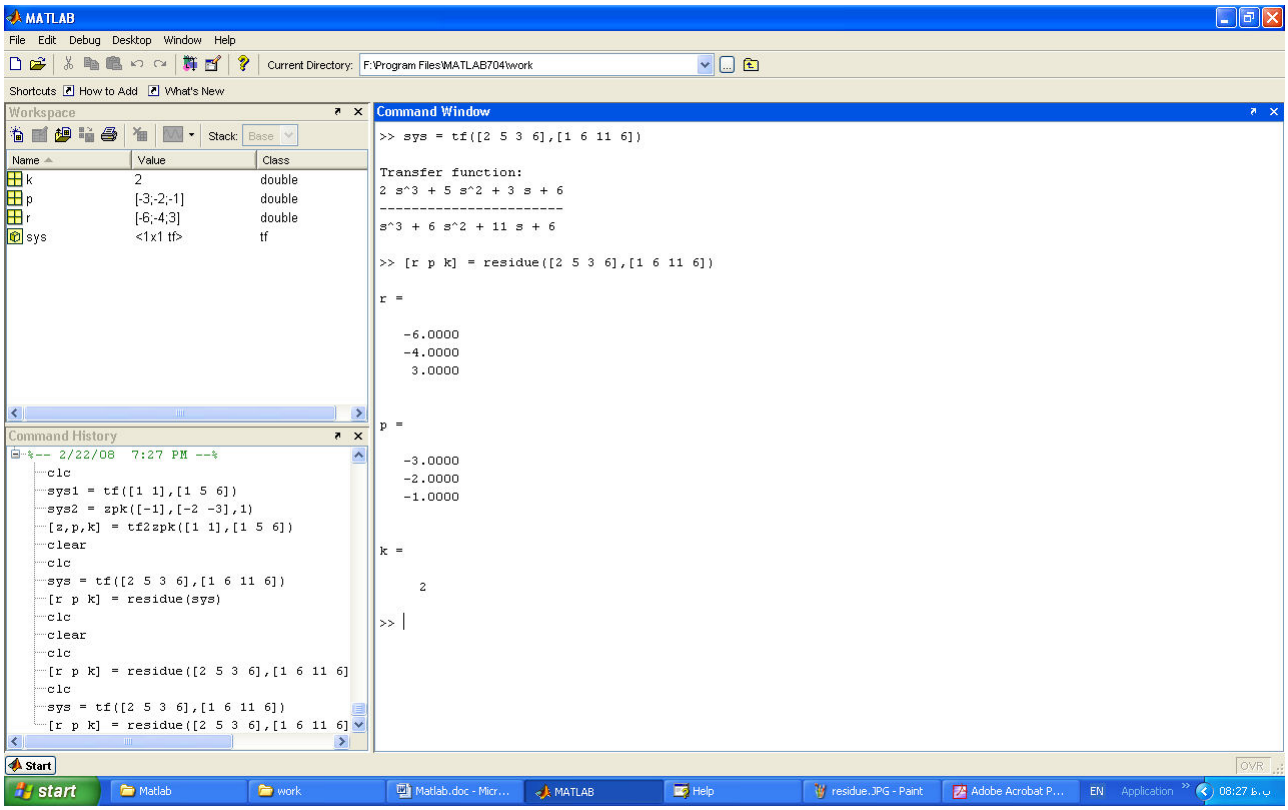
فرض کنید سیستمی به شکل زیر تعریف شده باشد:

$$\frac{b(s)}{a(s)} = \frac{b_1 s^m + b_2 s^{m-1} + b_3 s^{m-2} + \ldots + b_{m+1}}{a_1 s^n + a_2 s^{n-1} + a_3 s^{n-2} + \ldots + a_{n+1}}$$

می‌خواهیم این تابع تبدیل را به کسرهای جزیی به شکل زیر تبدیل کنیم:

$$\frac{b(s)}{a(s)} = \frac{r_1}{s-p_1} + \frac{r_2}{s-p_2} + \ldots + \frac{r_n}{s-p_n} + k(s)$$

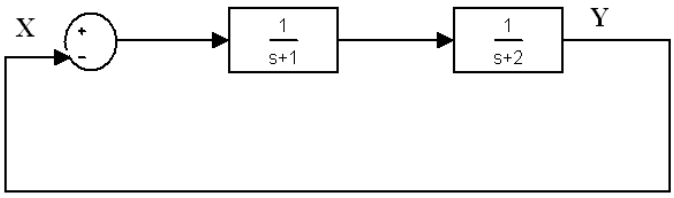
تابع [r p k] = residue(a ,b) ضرایب را محاسبه و در قالب بردارهای r و p و k ارائه می‌کند. a و b ضرایب صورت و مخرج تابع تبدیل هستند. شکل زیر را ببینید:



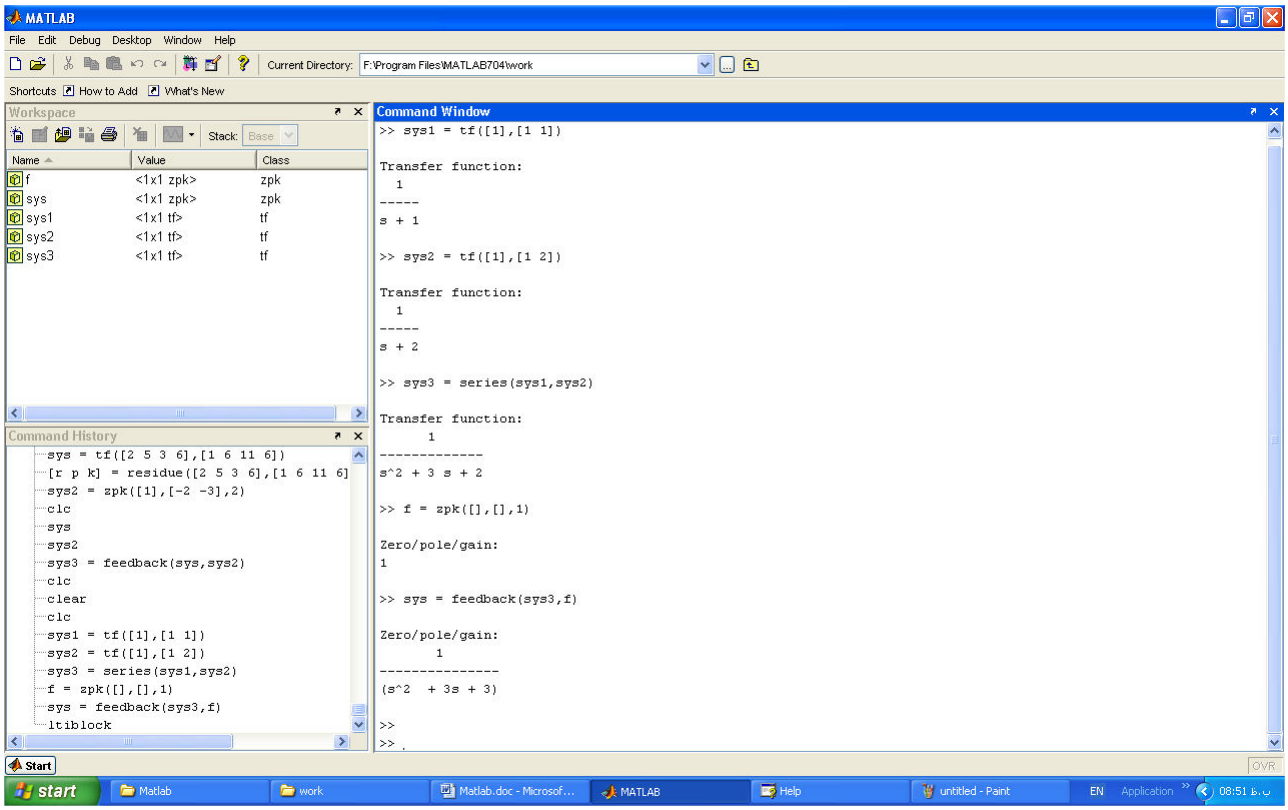
استفاده از تابع tf در اینجا لزومی ندارد و تنها برای نمایش شکل تابع تبدیل به کار رفته است.

ترکیب سیستمها

به کمک توابع series، parallel و feedback می‌توانید سیستمهای مختلف را با هم ترکیب کرده و سیستم جدید بسازید. فرض کنید می‌خواهیم سیستم زیر را بسازیم:



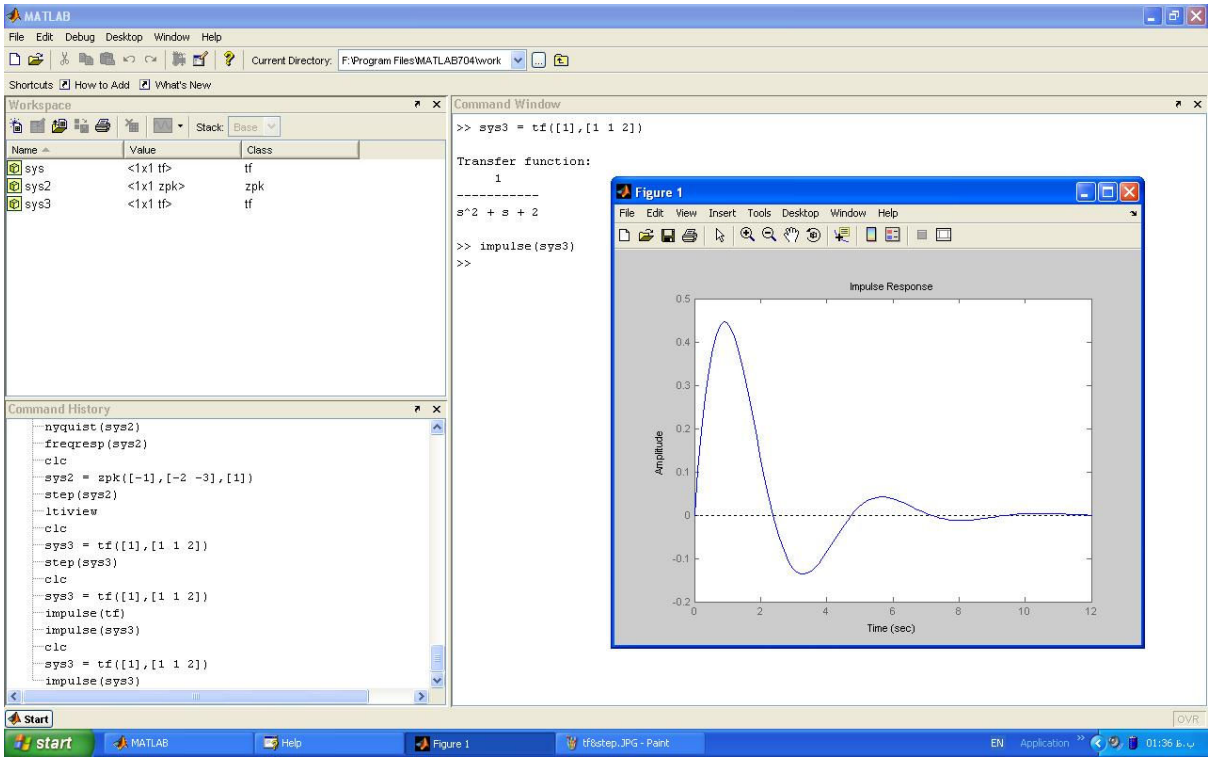
شکل زیر نحوه ساختن سیستم فوق را نشان می‌دهد:



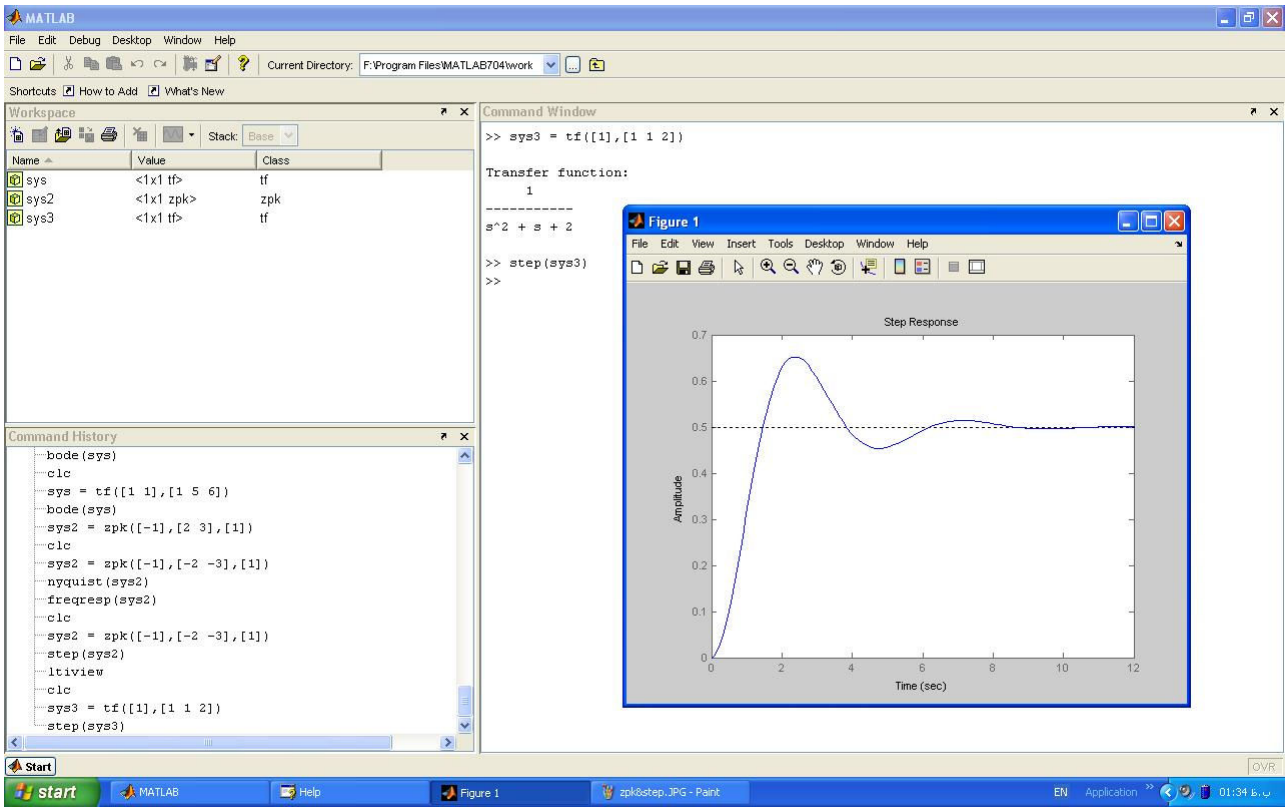
تحلیل ترسیمی سیستمهای کنترلی

MATLAB توابع بسیار مفیدی برای رسم پاسخها و نمودارهای مرتبط با سیستمهای کنترلی در اختیار کاربر می‌گذارد که در ذیل کاربرد بعضی از آنها نشان داده شده است.

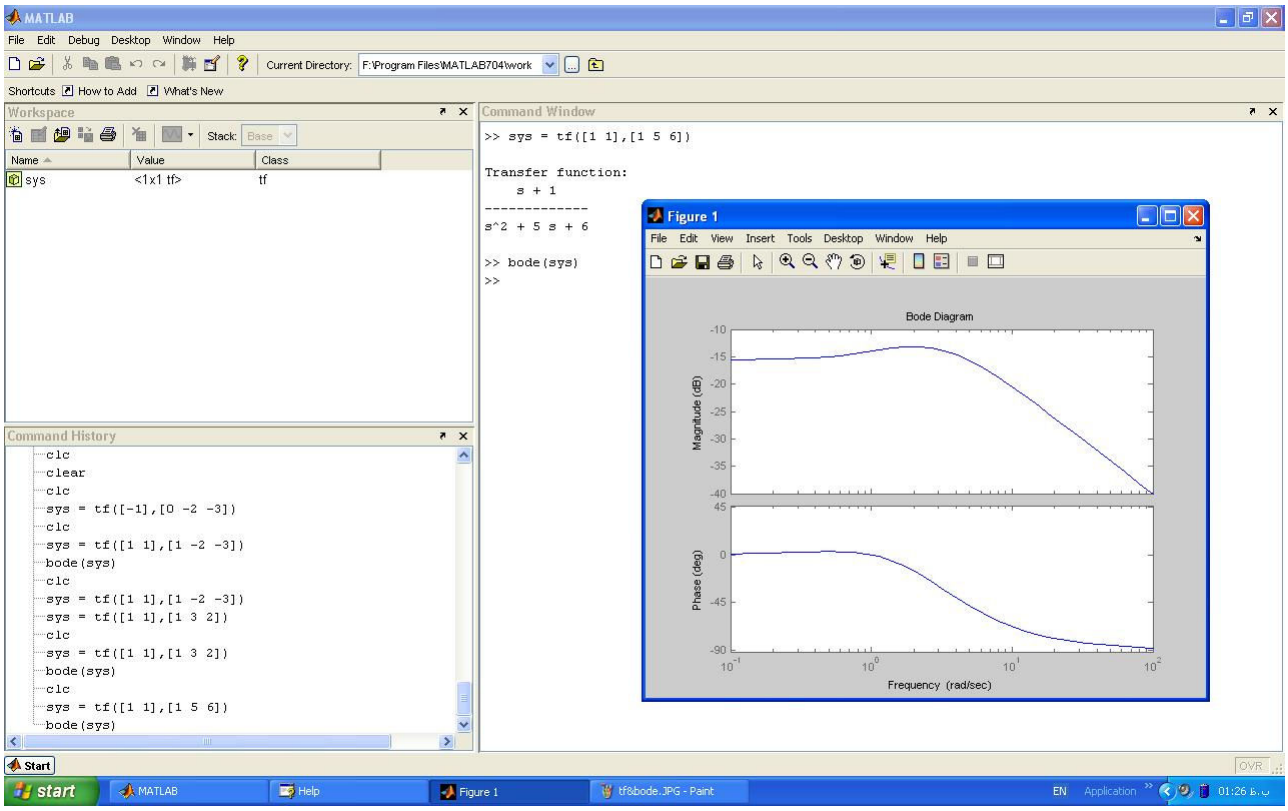
رسم پاسخ ضربه به کمک تابع impulse



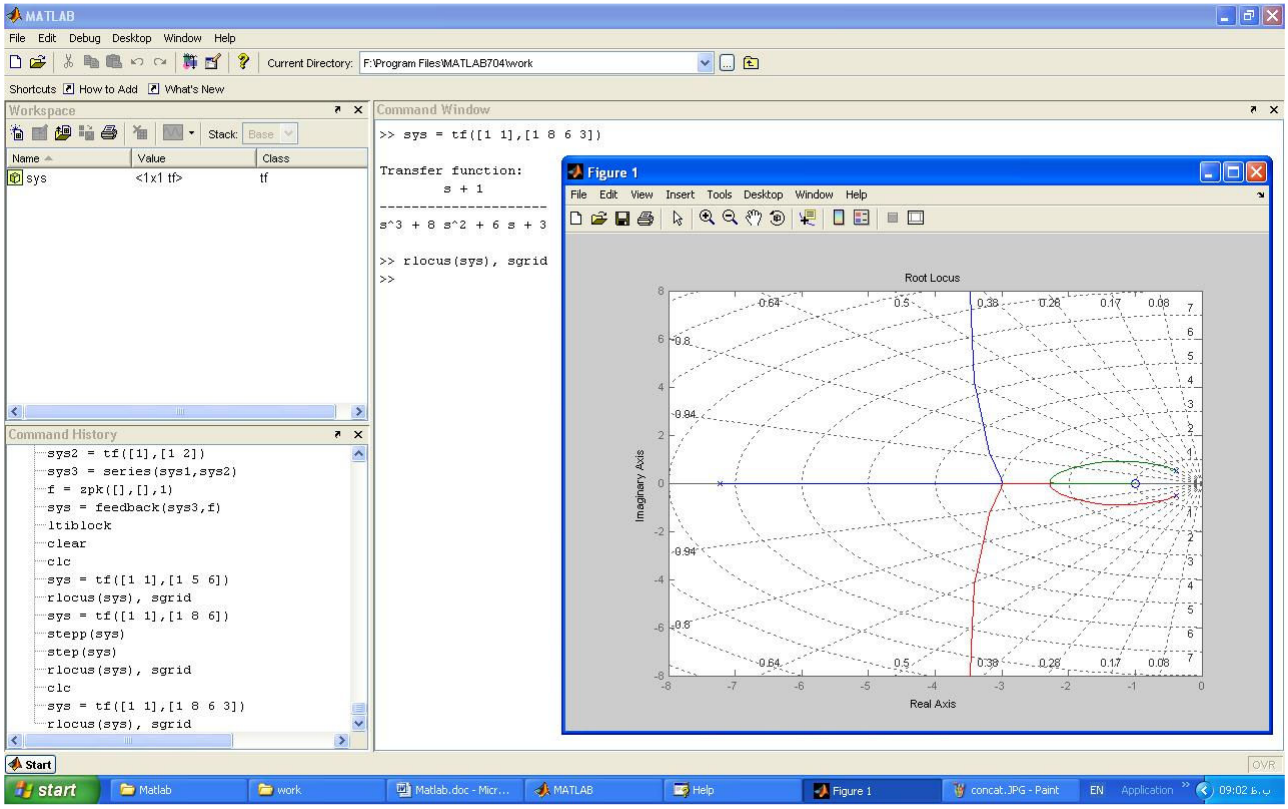
رسم پاسخ پله به کمک تابع step



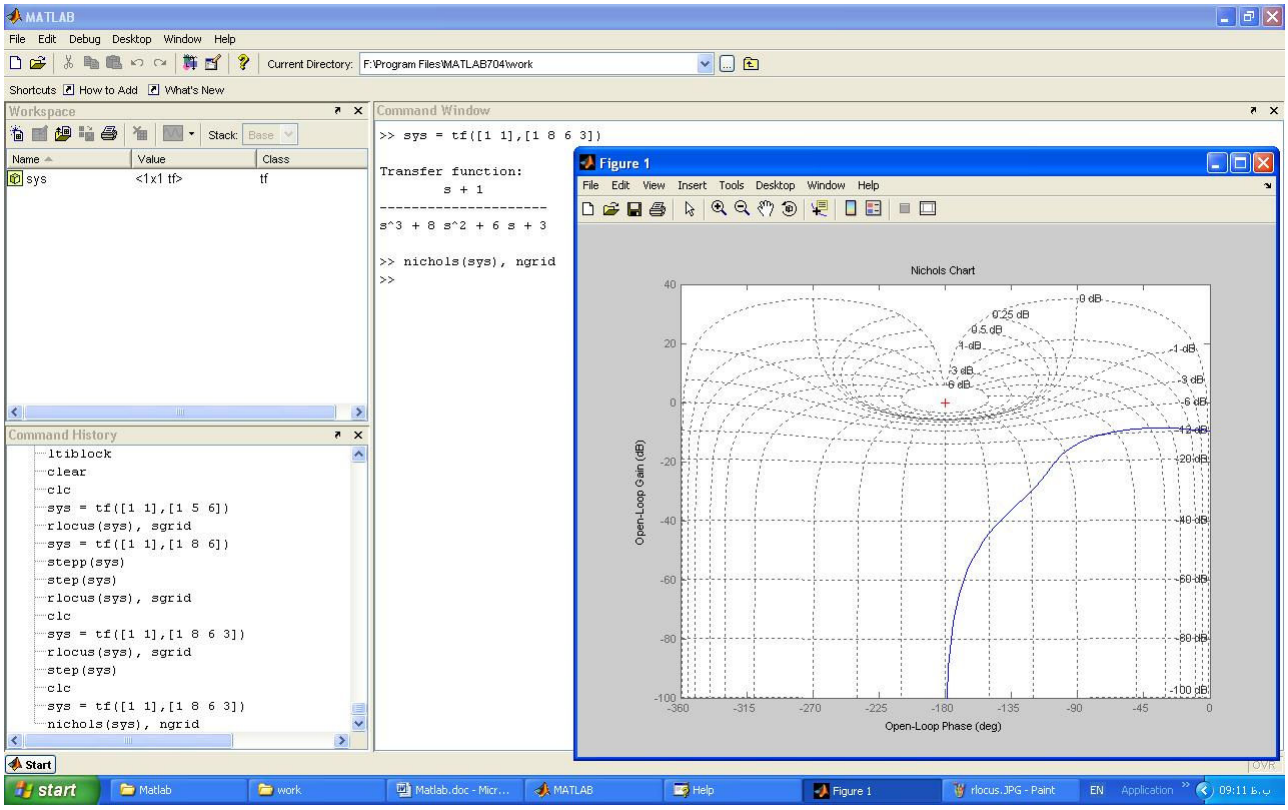
رسم نمودارهای بوده به کمک تابع bode



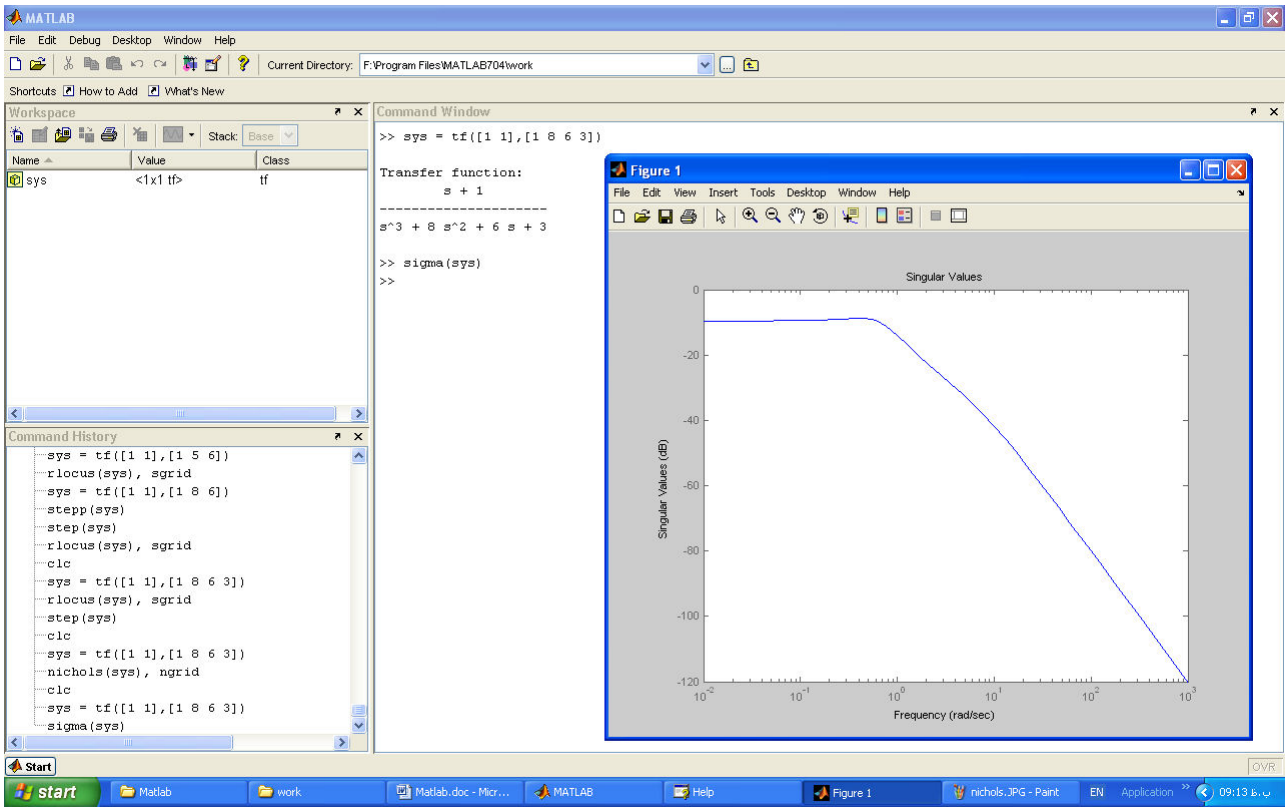
تحلیل روت-لوکاس به کمک تابع rlucas



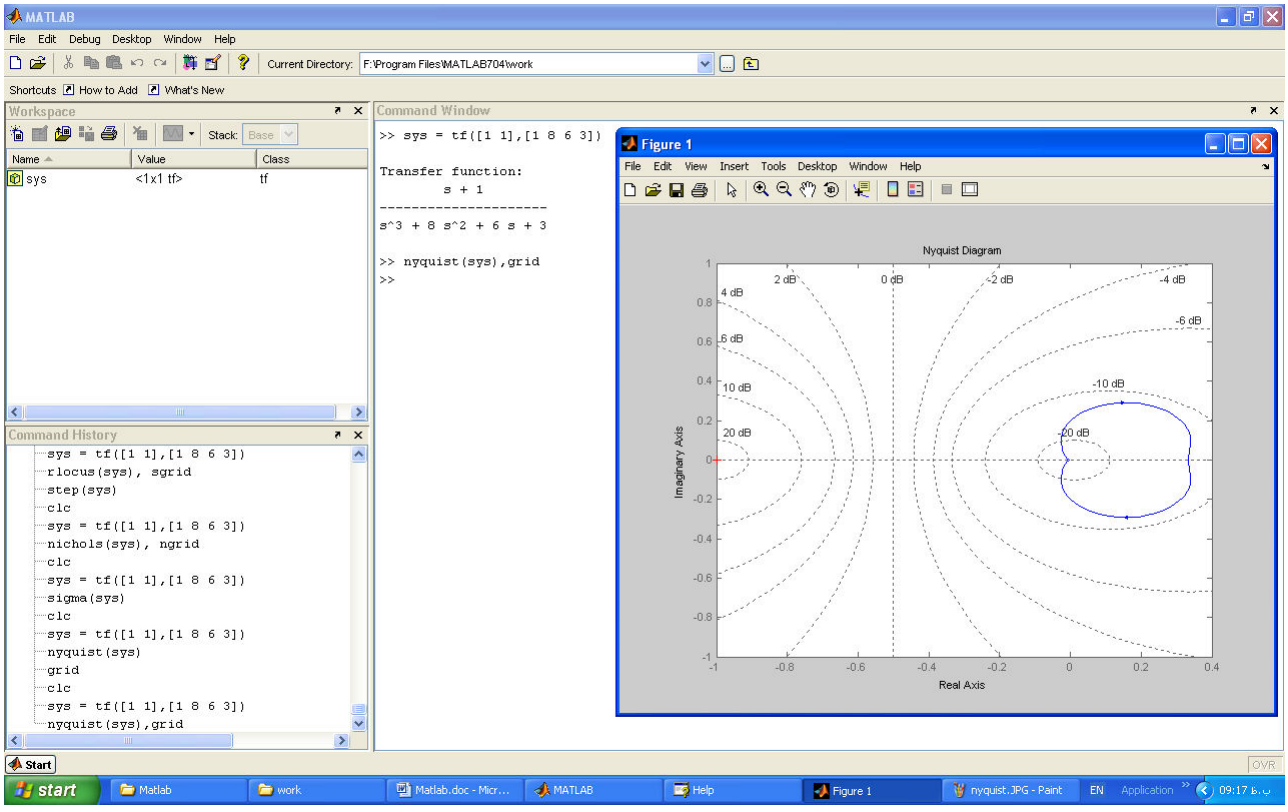
تحلیل نیکولز و تابع nichols



مقادیر تکین و تابع sigma

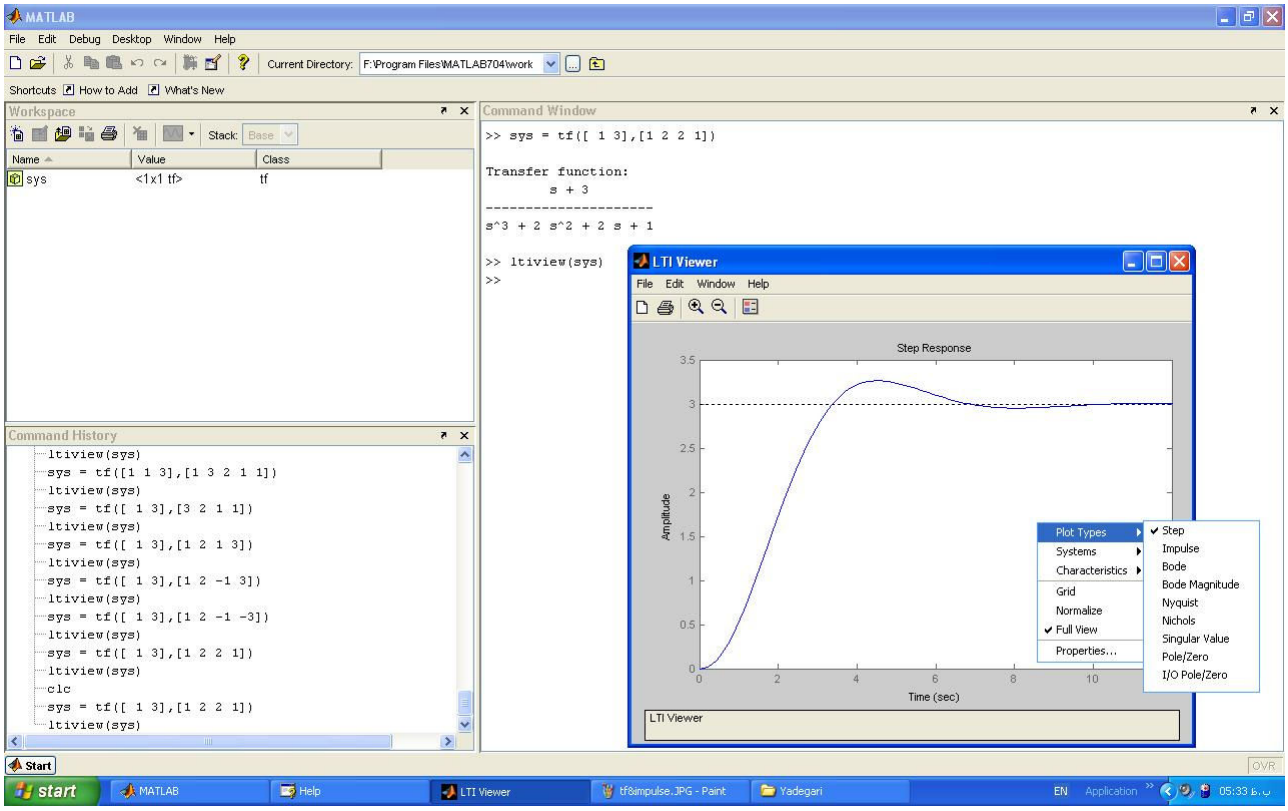


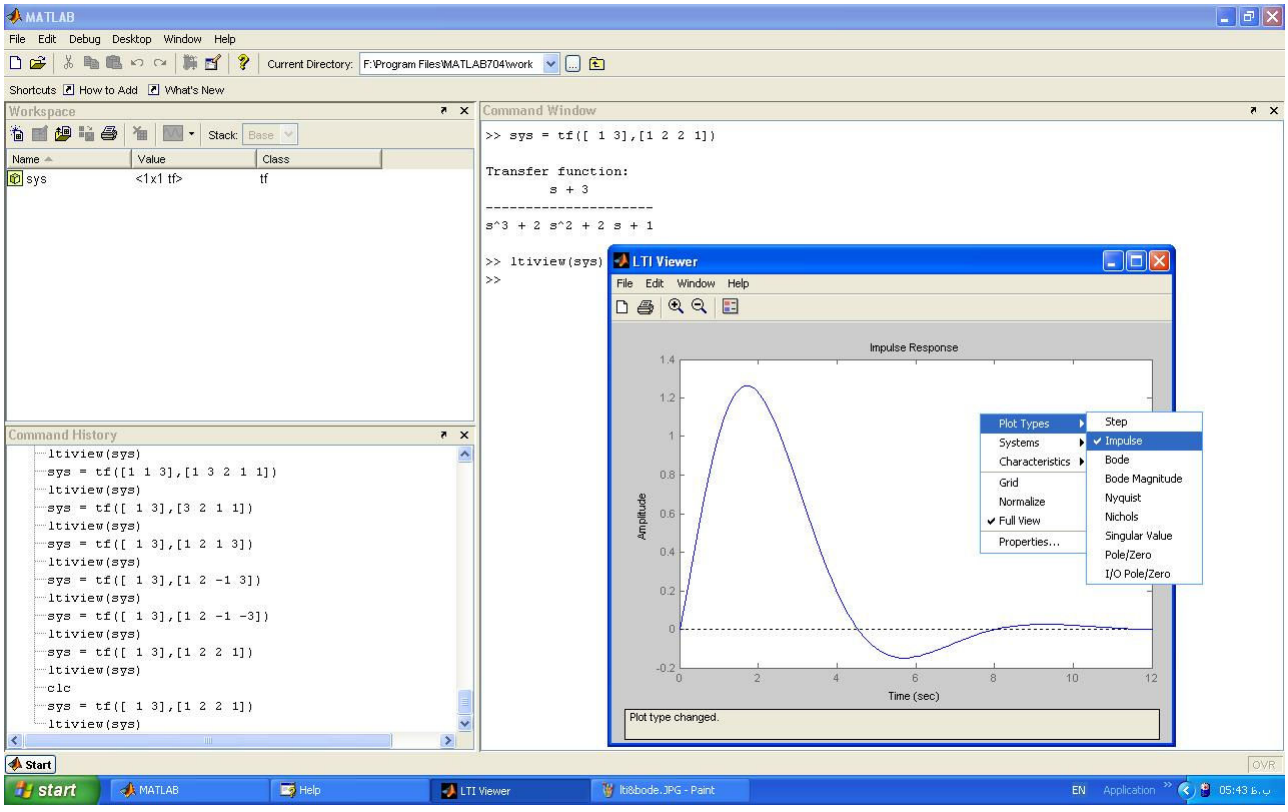
تحلیل نایکوییست و تابع nyquist

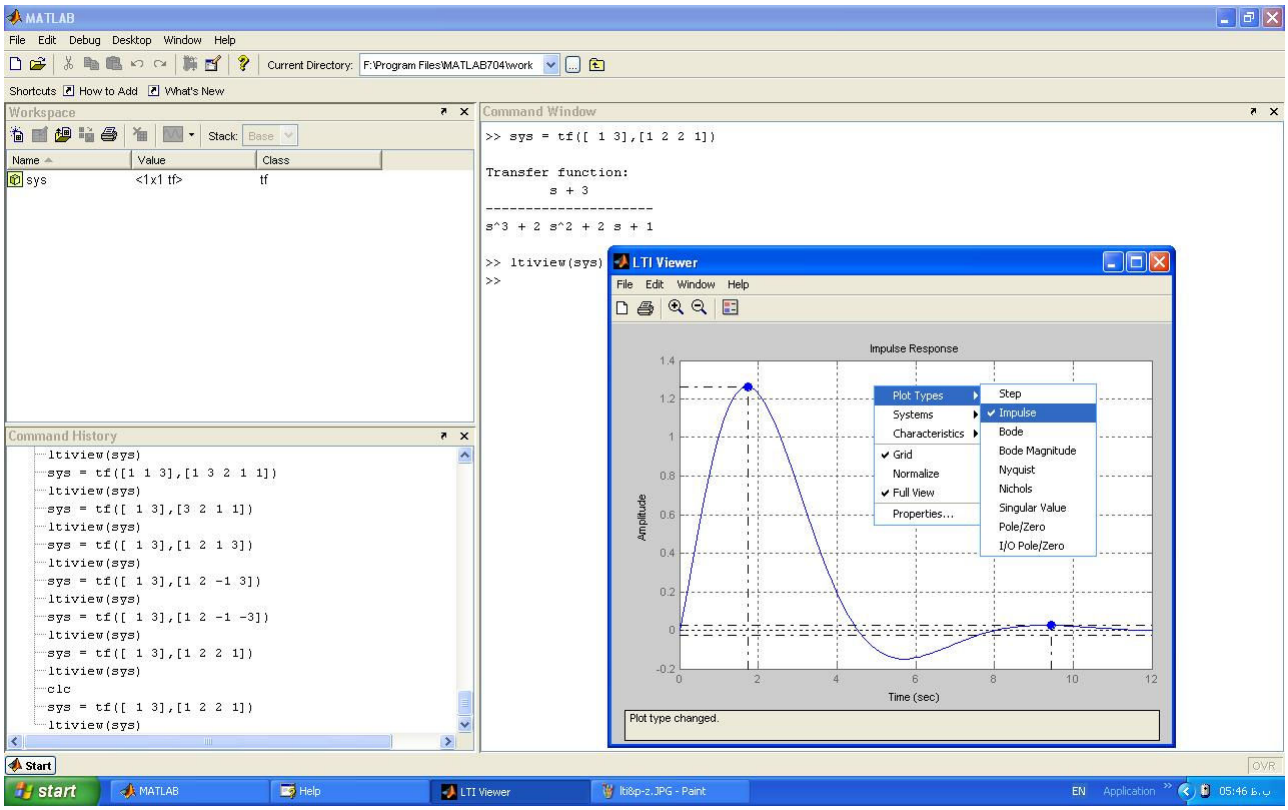


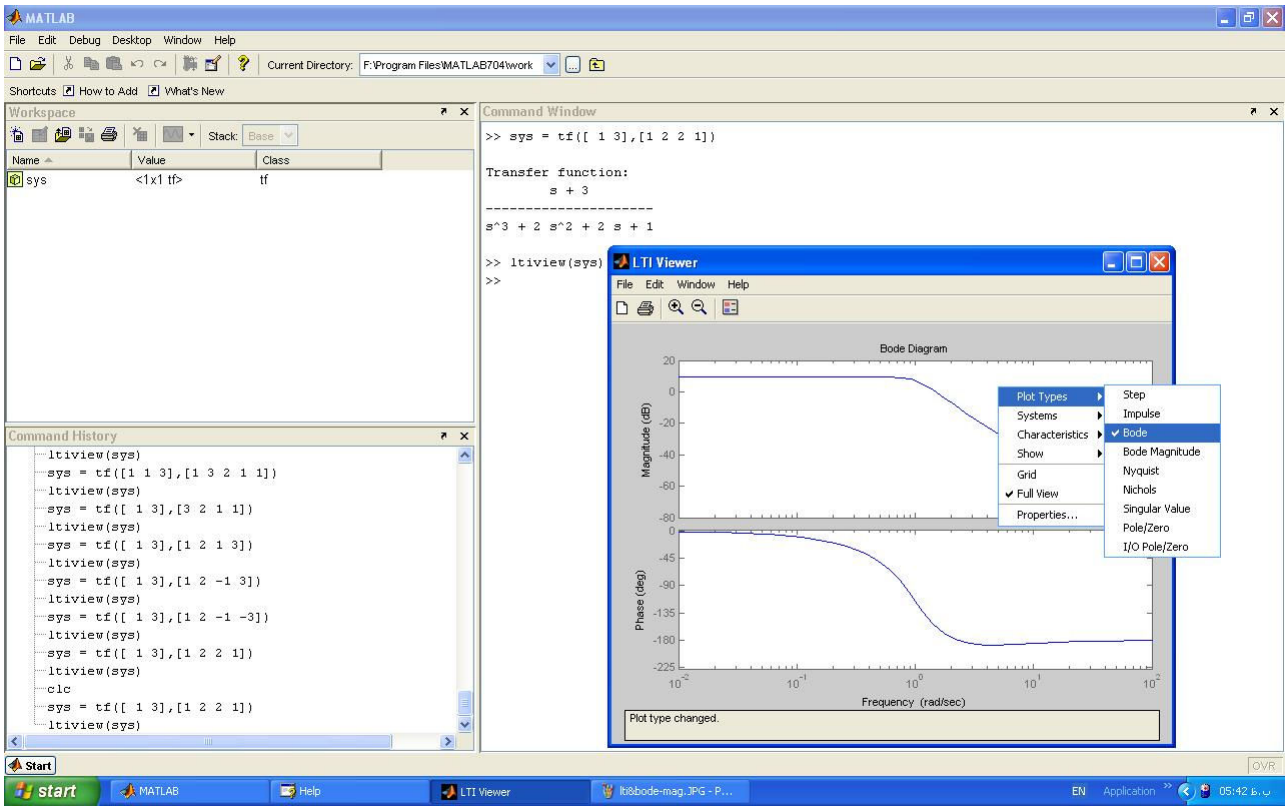
ابزار ltiview

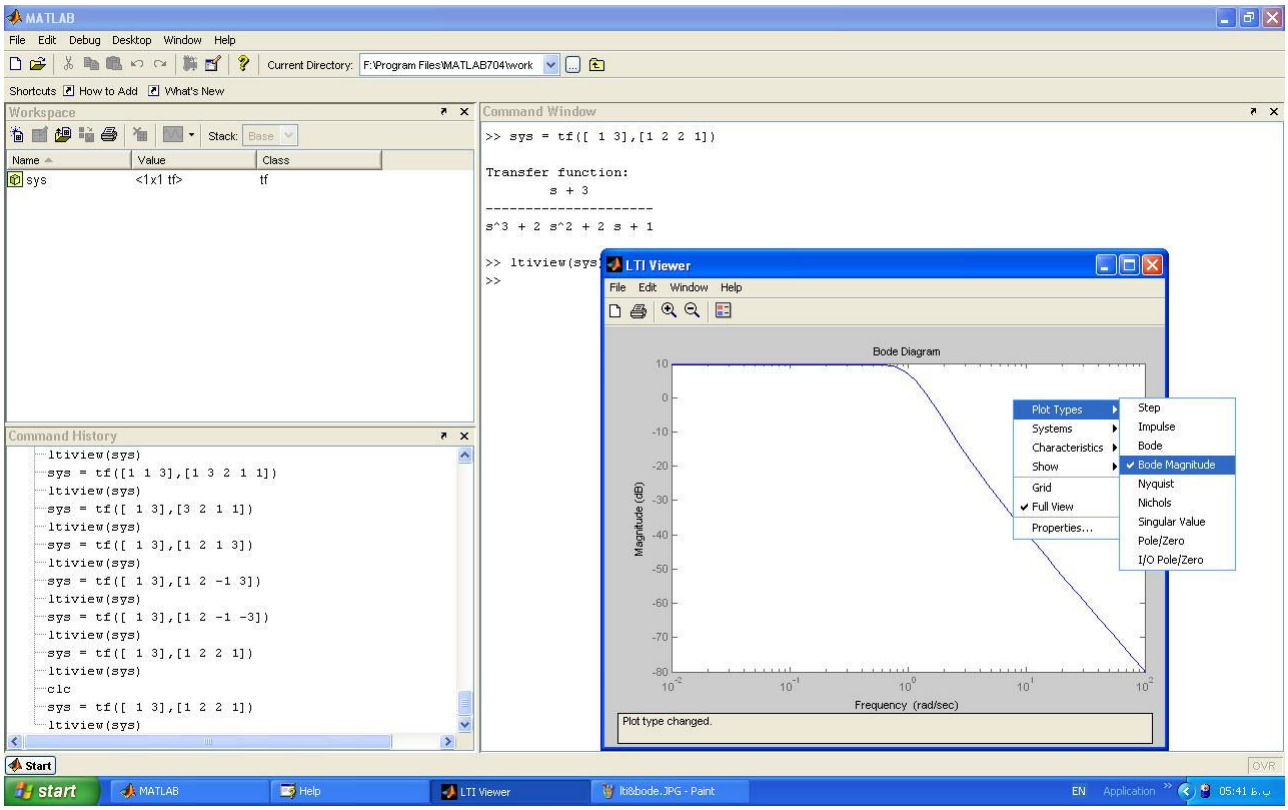
به کمک این ابزار می‌توان همه نمودارهای مرتبط با یک سیستم را بدون نیاز به فراخوانی تک‌تک توابع بررسی کرد:

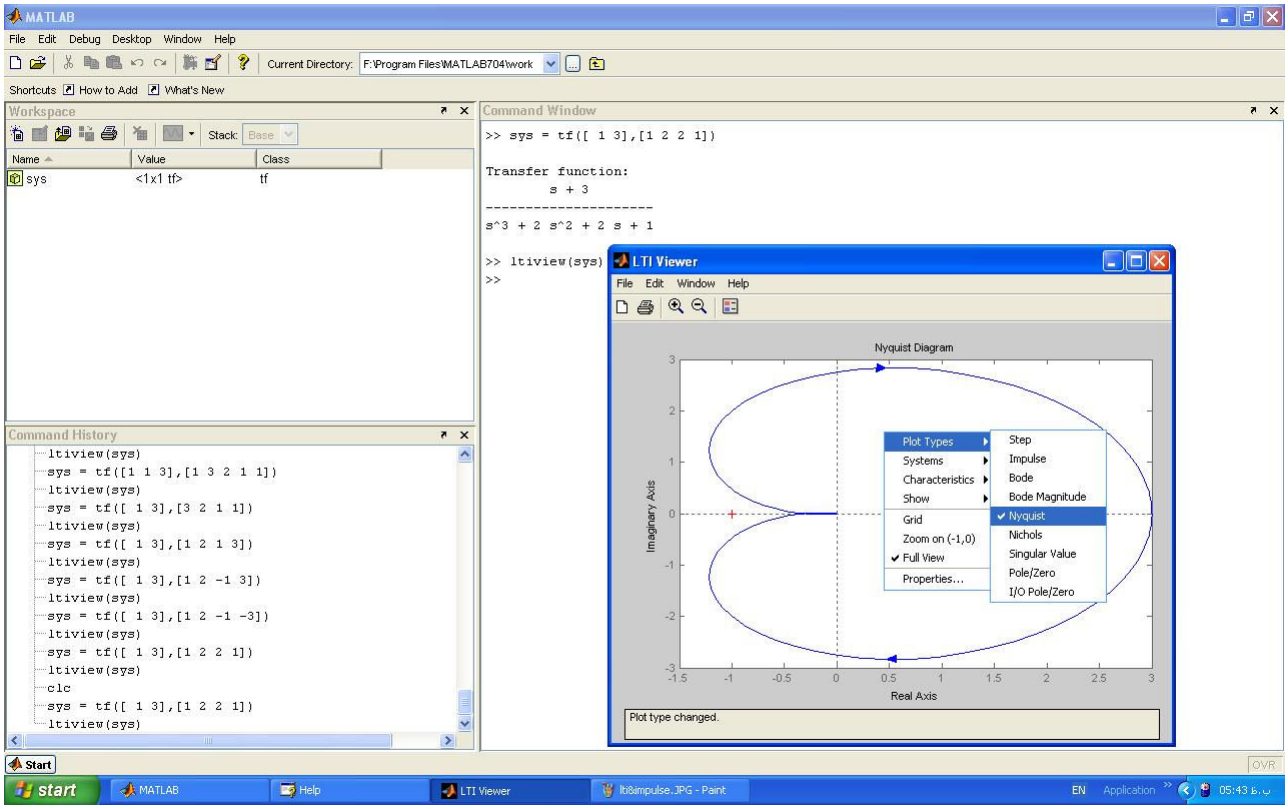


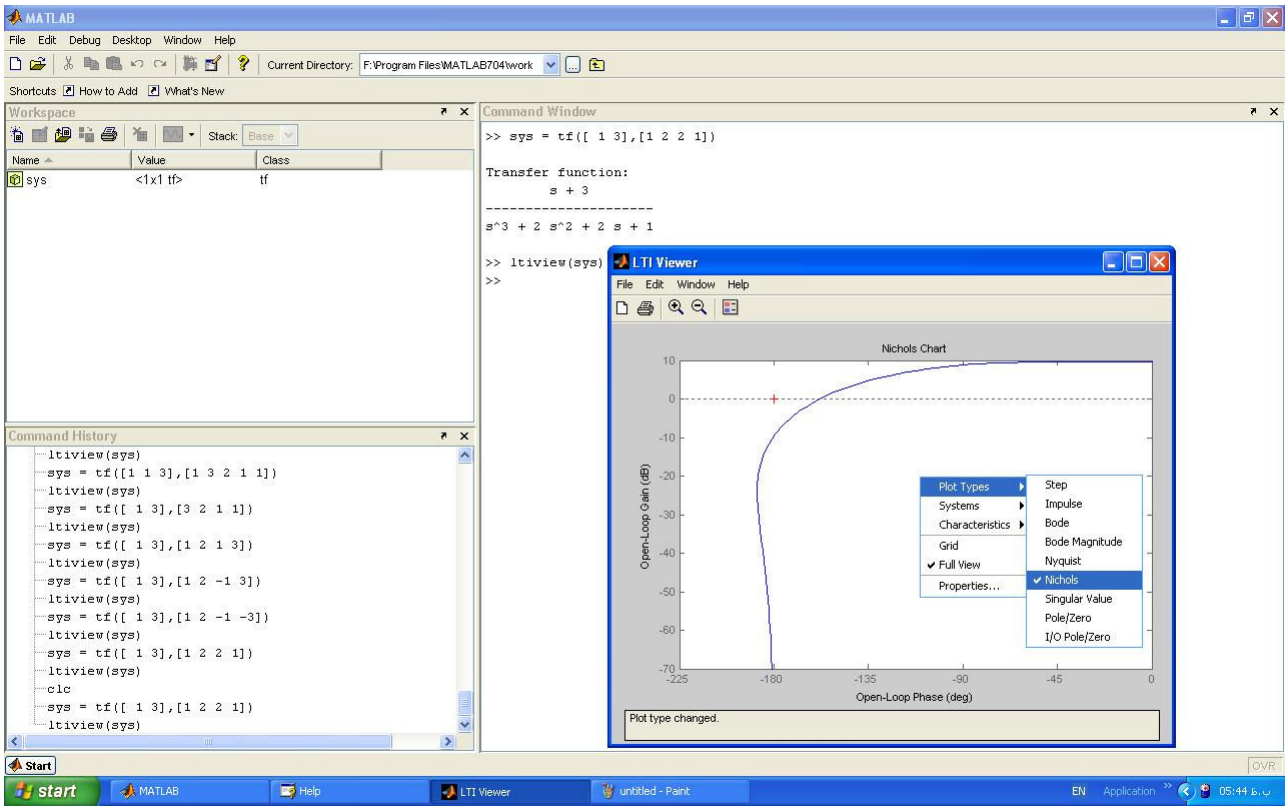


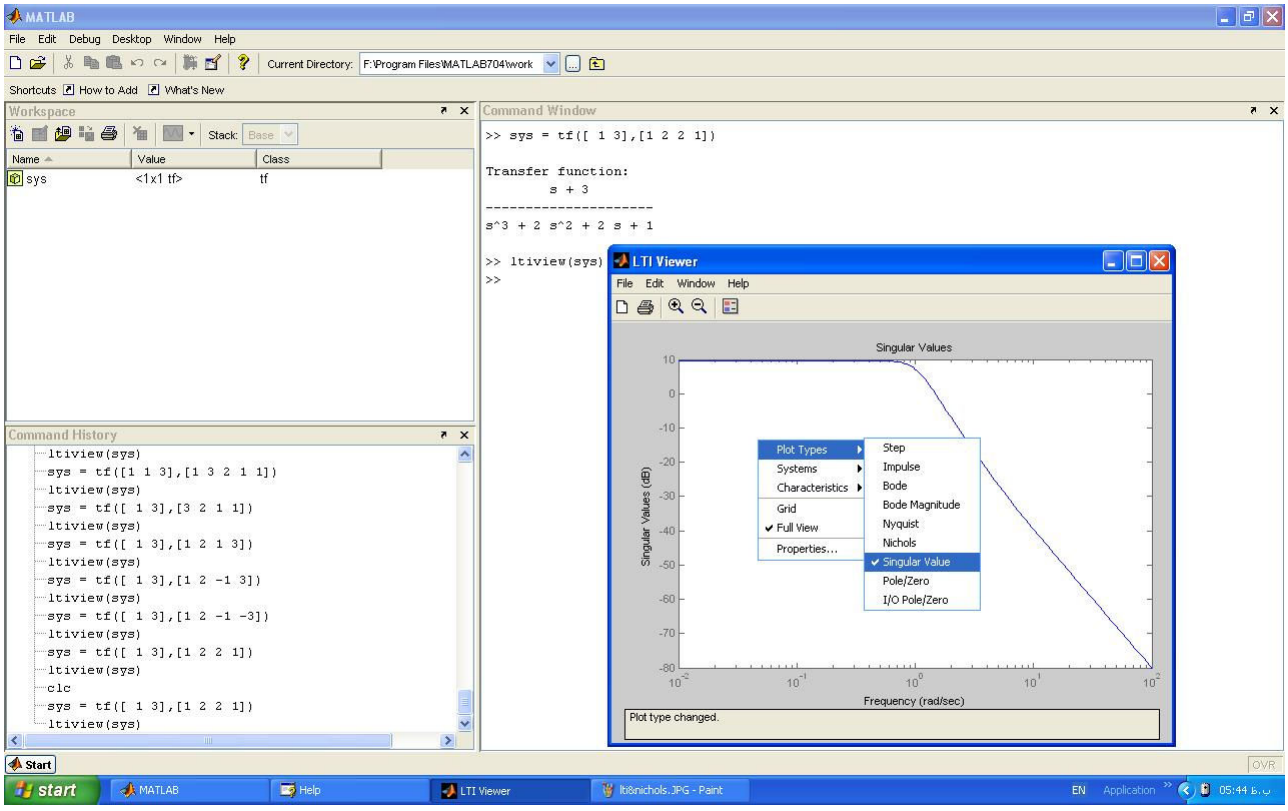

MATLAB
File Edit Debug Desktop Window Help
Current Directory: F:\Program Files\MATLAB704\work
Shortcuts How to Add What's New
Workspace
Stack: Base
Name Value Class
sys <1x1 tf> tf
Command Window
>> sys = tf([1 3],[1 2 2 1])
Transfer function:
s + 3
----
s^3 + 2 s^2 + 2 s + 1
>> ltiview(sys)
>>
Command History
ltiview(sys)
sys = tf([1 1 3],[1 3 2 1 1])
ltiview(sys)
sys = tf([1 3],[3 2 1 1])
ltiview(sys)
sys = tf([1 3],[1 2 1 3])
ltiview(sys)
sys = tf([1 3],[1 2 -1 3])
ltiview(sys)
sys = tf([1 3],[1 2 -1 -3])
ltiview(sys)
sys = tf([1 3],[1 2 2 1])
ltiview(sys)
clc
sys = tf([1 3],[1 2 2 1])
ltiview(sys)
LTI Viewer
File Edit Window Help
Impulse Response
Amplitude
Time (sec)
Plot Types
Systems
Characteristics
Grid
Normalize
Full View
Properties...
Step
Impulse
Bode
Bode Magnitude
Nyquist
Nichols
Singular Value
Pole/Zero
I/O Pole/Zero
Plot type changed.
Start
start
MATLAB
Help
LTI Viewer
lti8p-2.JPG - Paint
EN
Application
05:46 ب.ظ

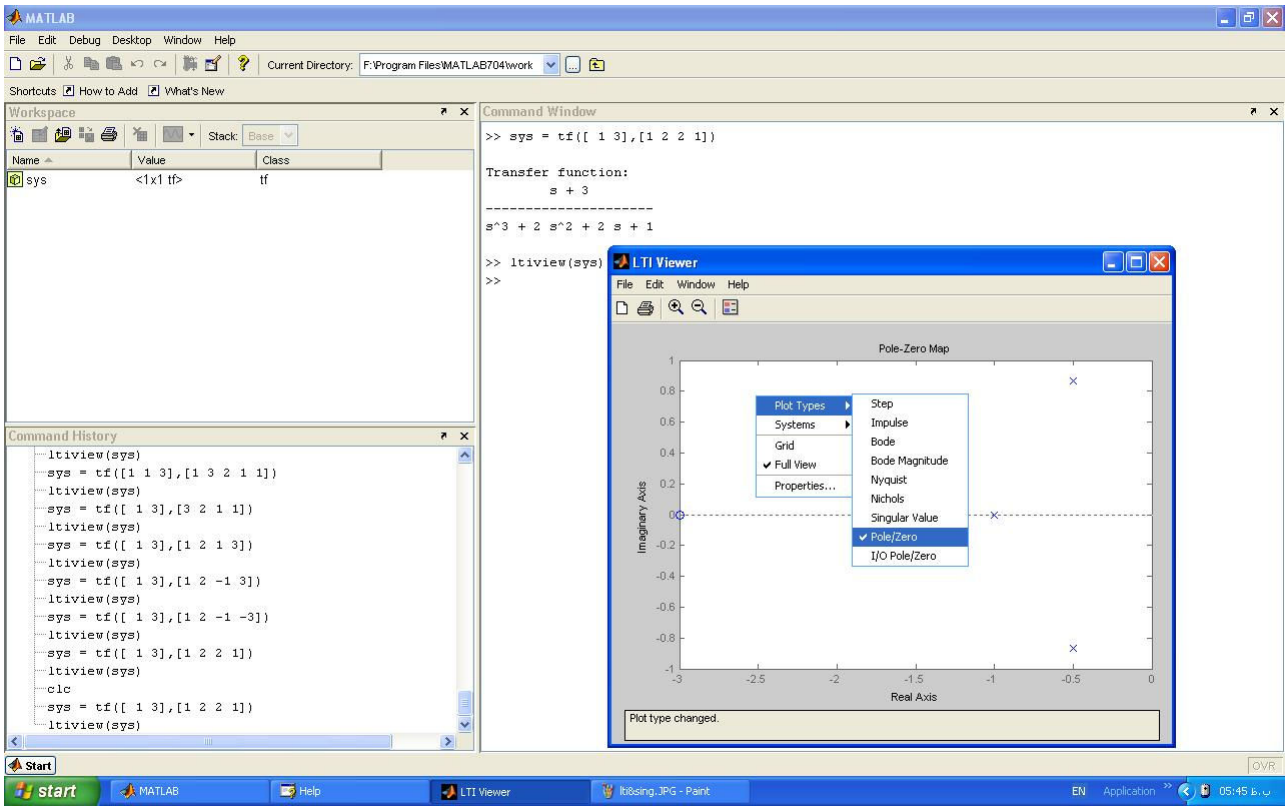

MATLAB
File Edit Debug Desktop Window Help
Current Directory: F:\Program Files\MATLAB704\work
Shortcuts How to Add What's New
Workspace
Stack: Base
Name Value Class
sys <1x1 tf> tf
Command Window
>> sys = tf([1 3],[1 2 2 1])
Transfer function:
s + 3
----
s^3 + 2 s^2 + 2 s + 1
>> ltiview(sys)
>>
Command History
ltiview(sys)
sys = tf([1 1 3],[1 3 2 1 1])
ltiview(sys)
sys = tf([1 3],[3 2 1 1])
ltiview(sys)
sys = tf([1 3],[1 2 1 3])
ltiview(sys)
sys = tf([1 3],[1 2 -1 3])
ltiview(sys)
sys = tf([1 3],[1 2 -1 -3])
ltiview(sys)
sys = tf([1 3],[1 2 2 1])
ltiview(sys)
clc
sys = tf([1 3],[1 2 2 1])
ltiview(sys)
LTI Viewer
File Edit Window Help
Bode Diagram
Magnitude (dB)
Phase (deg)
Frequency (rad/sec)
Plot Types
Systems
Characteristics
Show
Grid
Full View
Properties...
Step
Impulse
Bode
Bode Magnitude
Nyquist
Nichols
Singular Value
Pole/Zero
I/O Pole/Zero
Plot type changed.
Start
start
MATLAB
Help
LTI Viewer
lti&bode-mag.JPG - P...
EN
Application
05:42 ب.ظ


MATLAB
File Edit Debug Desktop Window Help
Current Directory: F:\Program Files\MATLAB704\work
Shortcuts How to Add What's New
Workspace
Stack: Base
Name Value Class
sys <1x1 tf> tf
Command Window
>> sys = tf([1 3],[1 2 2 1])
Transfer function:
s + 3
----
s^3 + 2 s^2 + 2 s + 1
>> ltiview(sys)
>>
Command History
ltiview(sys)
sys = tf([1 1 3],[1 3 2 1 1])
ltiview(sys)
sys = tf([1 3],[3 2 1 1])
ltiview(sys)
sys = tf([1 3],[1 2 1 3])
ltiview(sys)
sys = tf([1 3],[1 2 -1 3])
ltiview(sys)
sys = tf([1 3],[1 2 -1 -3])
ltiview(sys)
sys = tf([1 3],[1 2 2 1])
ltiview(sys)
clc
sys = tf([1 3],[1 2 2 1])
ltiview(sys)
LTI Viewer
File Edit Window Help
Bode Diagram
Magnitude (dB)
Frequency (rad/sec)
Plot type changed.
Plot Types
Systems
Characteristics
Show
Grid
Full View
Properties...
Step
Impulse
Bode
Bode Magnitude
Nyquist
Nichols
Singular Value
Pole/Zero
I/O Pole/Zero
Start
start
MATLAB
Help
LTI Viewer
lti8bode.JPG - Paint
EN
Application
05:41 ب.ظ


MATLAB
File Edit Debug Desktop Window Help
Current Directory: F:\Program Files\MATLAB704\work
Shortcuts How to Add What's New
Workspace
Stack: Base
Name Value Class
sys <1x1 tf> tf
Command Window
>> sys = tf([1 3],[1 2 2 1])
Transfer function:
s + 3
----
s^3 + 2 s^2 + 2 s + 1
>> ltiview(sys)
>>
Command History
ltiview(sys)
sys = tf([1 1 3],[1 3 2 1 1])
ltiview(sys)
sys = tf([1 3],[3 2 1 1])
ltiview(sys)
sys = tf([1 3],[1 2 1 3])
ltiview(sys)
sys = tf([1 3],[1 2 -1 3])
ltiview(sys)
sys = tf([1 3],[1 2 -1 -3])
ltiview(sys)
sys = tf([1 3],[1 2 2 1])
ltiview(sys)
clc
sys = tf([1 3],[1 2 2 1])
ltiview(sys)
LTI Viewer
File Edit Window Help
Nyquist Diagram
Imaginary Axis
Real Axis
Plot type changed.
Plot Types
Systems
Characteristics
Show
Grid
Zoom on (-1,0)
Full View
Properties...
Step
Impulse
Bode
Bode Magnitude
Nyquist
Nichols
Singular Value
Pole/Zero
I/O Pole/Zero
Start
start
MATLAB
Help
LTI Viewer
lti8impulse.JPG - Paint
EN
Application
05:43 ب.ظ

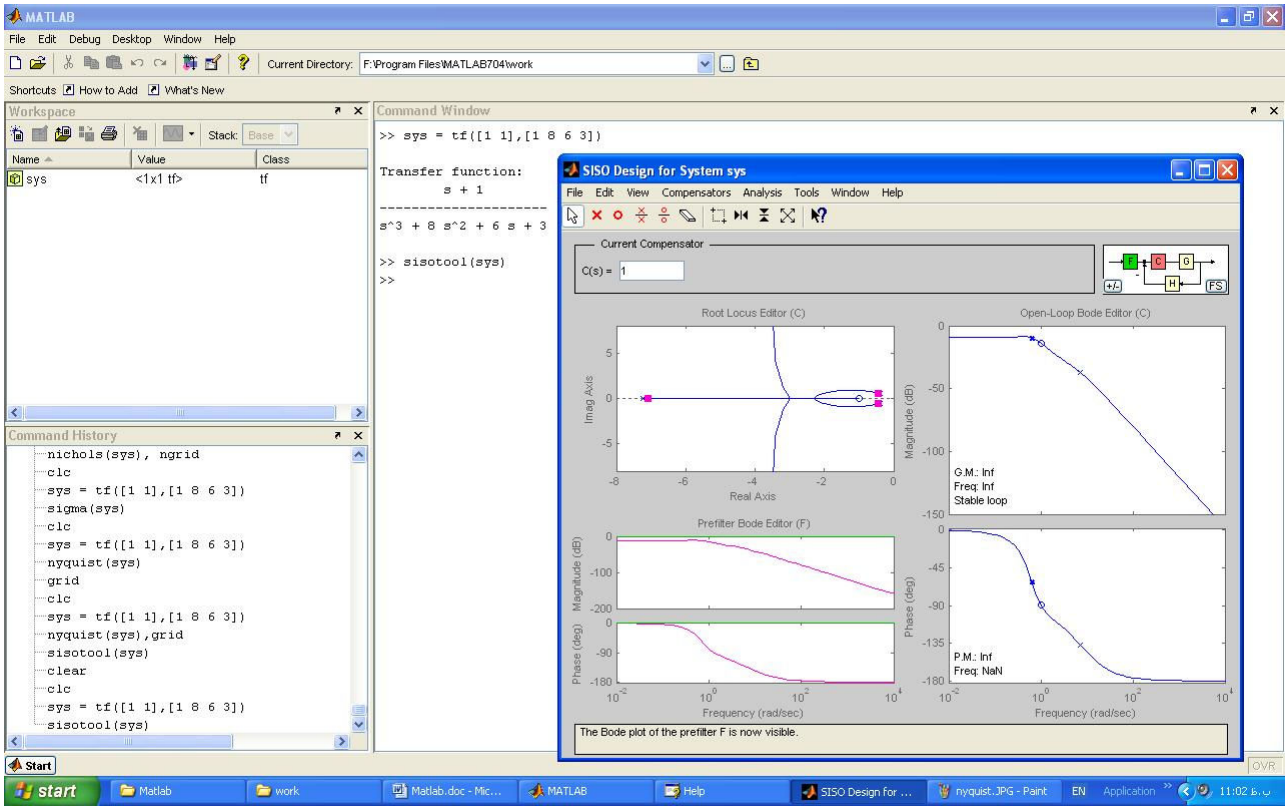

MATLAB
File Edit Debug Desktop Window Help
Current Directory: F:\Program Files\MATLAB704\work
Shortcuts How to Add What's New
Workspace
Stack: Base
Name Value Class
sys <1x1 tf> tf
Command Window
>> sys = tf([1 3],[1 2 2 1])
Transfer function:
s + 3
----
s^3 + 2 s^2 + 2 s + 1
>> ltiview(sys)
>>
Command History
ltiview(sys)
sys = tf([1 1 3],[1 3 2 1 1])
ltiview(sys)
sys = tf([1 3],[3 2 1 1])
ltiview(sys)
sys = tf([1 3],[1 2 1 3])
ltiview(sys)
sys = tf([1 3],[1 2 -1 3])
ltiview(sys)
sys = tf([1 3],[1 2 -1 -3])
ltiview(sys)
sys = tf([1 3],[1 2 2 1])
ltiview(sys)
clc
sys = tf([1 3],[1 2 2 1])
ltiview(sys)
LTI Viewer
File Edit Window Help
Nichols Chart
Open-Loop Gain (dB)
Open-Loop Phase (deg)
Plot Types
Systems
Characteristics
Grid
Full View
Properties...
Step
Impulse
Bode
Bode Magnitude
Nyquist
Nichols
Singular Value
Pole/Zero
I/O Pole/Zero
Plot type changed.
Start
start
MATLAB
Help
LTI Viewer
untitled - Paint
EN
Application
05:44 ب.ظ


MATLAB
File Edit Debug Desktop Window Help
Current Directory: F:\Program Files\MATLAB704\work
Shortcuts How to Add What's New
Workspace
Stack: Base
Name Value Class
sys <1x1 tf> tf
Command Window
>> sys = tf([1 3],[1 2 2 1])
Transfer function:
s + 3
----
s^3 + 2 s^2 + 2 s + 1
>> ltiview(sys)
>>
Command History
ltiview(sys)
sys = tf([1 1 3],[1 3 2 1 1])
ltiview(sys)
sys = tf([1 3],[3 2 1 1])
ltiview(sys)
sys = tf([1 3],[1 2 1 3])
ltiview(sys)
sys = tf([1 3],[1 2 -1 3])
ltiview(sys)
sys = tf([1 3],[1 2 -1 -3])
ltiview(sys)
sys = tf([1 3],[1 2 2 1])
ltiview(sys)
clc
sys = tf([1 3],[1 2 2 1])
ltiview(sys)
LTI Viewer
File Edit Window Help
Singular Values
Singular Values (dB)
Frequency (rad/sec)
Plot Types
Systems
Characteristics
Grid
Full View
Properties...
Step
Impulse
Bode
Bode Magnitude
Nyquist
Nichols
Singular Value
Pole/Zero
I/O Pole/Zero
Plot type changed.
Start
start
MATLAB
Help
LTI Viewer
lti8nichols.JPG - Paint
EN
Application
05:44 ب.ظ



ابزار sisotool

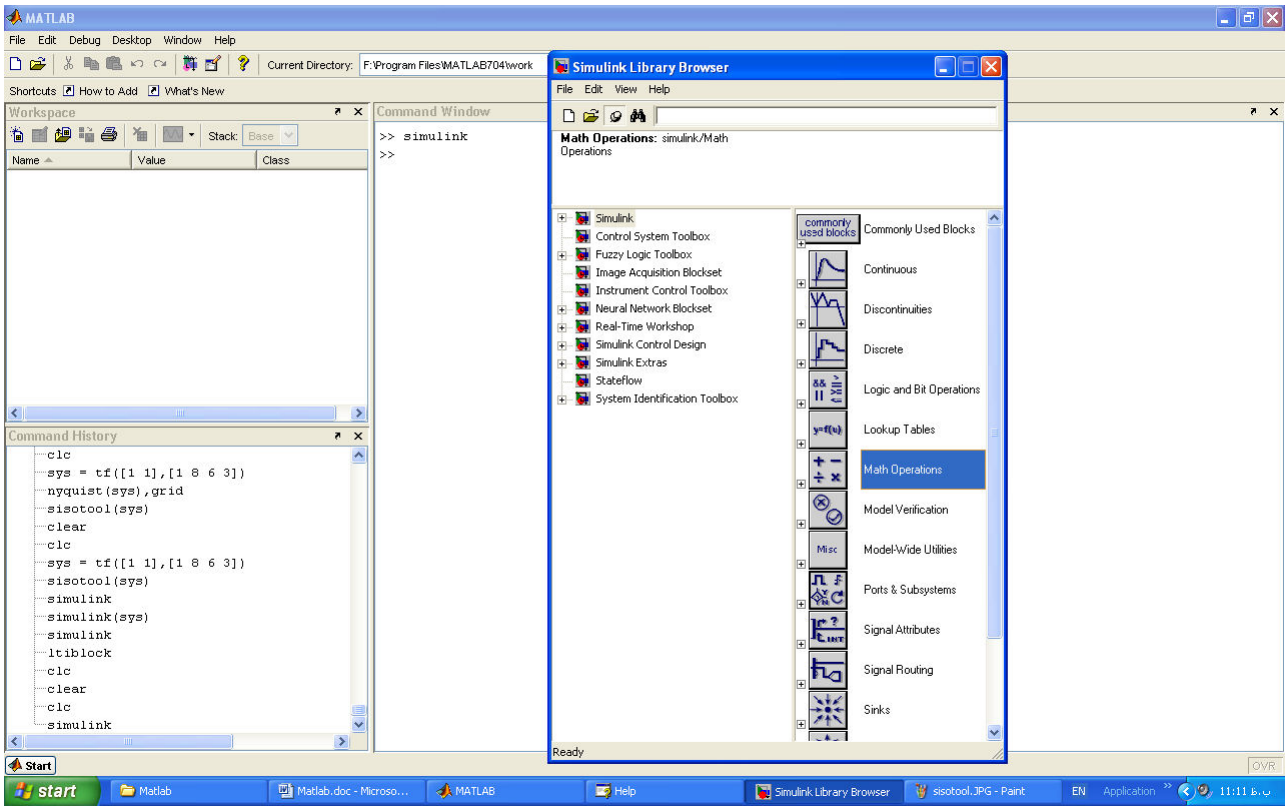
sisotool یک ابزار طراحی و تحلیل سیستمهای یک ورودی – یک خروجی (Single Input Single Output) است.



به کمک این ابزار می‌توانید به راحتی ساختار سیستم خود را تغییر داده و اثر این تغییرات روی نمودارهای سیستم را بررسی کنید. مثلاً با کلیک روی کلید FS (گوشه بالا سمت راست) می‌توانید ساختار فیدبک و با کلیک روی نشانه +/- علامت فیدبک را عوض کنید. از طریق منوی Analysis می‌توانید نمودارهای مهم سیستم را ببینید. به علاوه می‌توانید به کمک منوی Tools، سیستم خود را به simlulink صادر کنید.

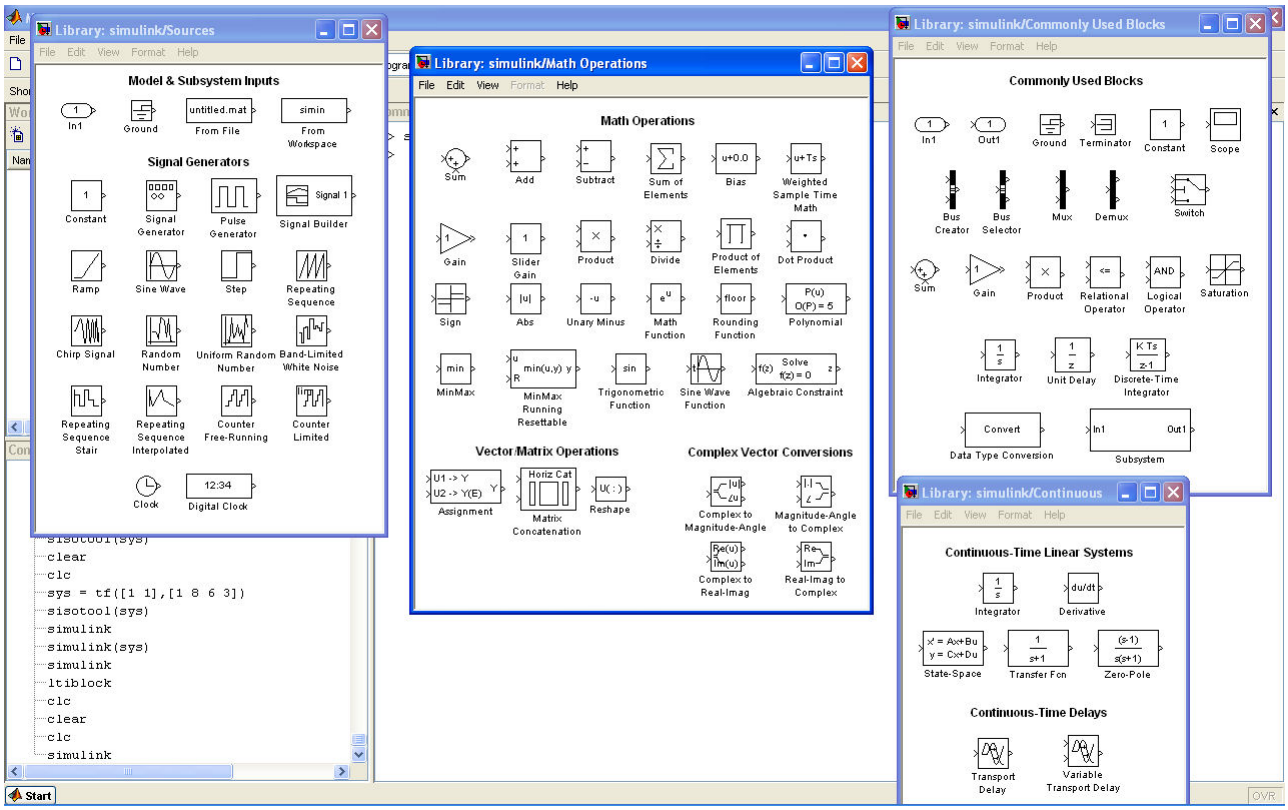
ابزار simulink

simulink ابزاری قوی برای وارد کردن جزییات سیستمهای مختلف به MATLAB به کمک بلوک دیاگرامهای گرافیکی است. این ابزار می‌تواند در ارتباط با جعبه‌ابزارهای متنوعی استفاده شود. با اجرای دستور simulink کتابخانه این ابزار که شامل بلوکهای متنوع است در دسترس کاربر قرار می‌گیرد:



برای استفاده از simulink در تحلیل سیستمهای کنترلی از فرمان ltiblock استفاده کنید. اجرای این دستور محیطی برای تحلیل سیستمهای کنترلی را در اختیار شما می‌گذارد تا بلوکهای simulink را از داخل کتابخانه به داخل این محیط آورده و پس از انجام اتصالات لازم، عملکرد سیستم را شبیه‌سازی کنید.

در ذیل بعضی از بلوکهای مفید simulink برای تحلیل سیستمهای کنترلی را می‌بینید:



بلوک scope می‌تواند در تحلیل شکلهای خروجی بسیار مفید باشد.

توابع جعبه‌ابزار سیستمهای کنترلی

General
ctrlpref Set Control System Toolbox preferences
ltimodels Detailed help on the various types of LTI models
ltiprops Detailed help on available LTI model properties

Creating Linear Models
filt Specify a digital filter
frd Create a frequency-response data models
lti/set Set/modify properties of LTI models
ss, dss Create state-space models (continuous/discrete)
tf Create transfer function models
zpk Create zero/pole/gain models

Data Extraction
dssdata Descriptor version of ssdata
frdata Extract frequency-response data
lti/get Access values of LTI model properties
ssdata Extract state-space matrices
tfdata Extract numerators and denominators
zpkdata Extract zero/pole/gain data

Conversions
c2d Convert from continuous- to discrete-time models
chunits Convert the units property for FRD models
d2c Convert from discrete- to continuous-time models
d2d Test true for continuous-time models
frd Convert to a frequency-response data model
ss Convert to a state-space model

tf Convert to a transfer function model
zpk Convert to a zero/pole/gain model

System Interconnections
append Group LTI systems by appending inputs and outputs
connect Derive state-space models from block diagram descriptions
feedback Feedback connections of two systems
lft Generalized feedback interconnection (Redheffer star product)
parallel Generalized parallel connection (see also overloaded +)
series Generalized series connection (see also overloaded *)

System Gain and Dynamics
bandwidth System bandwidth
dcgain D.C. (low-frequency) gain
bandwidth System bandwidth
damp Natural frequency and damping of system poles
dsort Norms of LTI systems
esort Sort continuous poles by real part
iopzmap Input/output pole/zero map
lti/norm Norms of LTI systems
modsep Region-based modal decomposition
pole, eig System poles
pzmap Pole/zero map
stabsep Stable/unstable decomposition

Time Domain Analysis
covar Covariance of response to white noise
gensig Generate input signal for lsim
impulse Impulse response
initial Response of state-space system with given initial state
lsim Response to arbitrary inputs
ltiview Response analysis GUI (LTI Viewer)
step Step response

Frequency Domain Analysis
allmargin All crossover frequencies and related gain/phase margins
bode Bode diagrams of the frequency response
bodemag Bode magnitude diagram only
evalfr Evaluate frequency response at given frequency
freqresp Frequency response over a frequency grid
frd/interp Interpolate frequency-esponse data
ltiview Response analysis GUI (LTI Viewer)
margin Gain and phase margins
nichols Nichols plot
nyquist Nyquist plot
sigma Plot the pole/zero map of an LTI model

Classical Design
rlocus Evans root locus
sisotool SISO design GUI (root locus and loop-shaping techniques)

Pole Placement
acker SISO pole placement
estim Form estimator given estimator gain
place MIMO pole placement
reg Form regulator given state-feedback and estimator gain

LQR/LQG Design
augstate Augment output by appending states
lqg Single-step LQG design

lgr, dlqr Linear-quadratic (LQ) state-feedback regulator
lqrd Discrete LQ regulator for continuous plants
lqrreg Form LQG regulator given LQ gain and Kalman estimator
lqgy LQ regulator with output weighting
kalman Kalman estimator
kalmand Discrete Kalman estimator for continuous plants

State-Space Models

balreal Grammian-based input/output balancing
canon State-space canonical forms
ctrb Controllability matrix
gram Controllability and observability grammians
minreal Minimal realization and pole/zero cancellation
modred Model state reduction
margin Calculate gain and phase margins
ngrid Superimpose grid lines on a Nichols plot
nichols Calculate Nichols plot
nyquist Calculate Nyquist plot
obsv Observability matrix
sminreal Structurally minimal realization
ss2ss State coordinate transformation
ssbal Diagonal balancing of state-space realizations

Time Delays

delay2z Replace delays by poles at z=0 or FRD phase shift
hasdelay True for models with time delays
pade Pade approximation of time delays
totaldelay Total delay between each input/output pair

Model Dimensions and Characteristics

class Model type ('tf', 'zpk', 'ss', or 'frd')
isct True for continuous-time models
isdt True for discrete-time models
isproper True for proper models
issiso True for single-input/single-output models
lti/ndims Number of dimensions
lti/isempty True for empty LTI models
reshape Reshape array of linear models size Model sizes and order

Overloaded and Arithmetic Operators

+ and - Add and subtract systems (parallel connection)
* Multiply systems (series connection)
\ Left divide -- sys1\sys2 means inv(sys1)*sys2
/ Right divide -- sys1/sys2 means sys1*inv(sys2)
^ Powers of a given system
' Pertransposition
.' Transposition of input/output map
[..] Concatenate models along inputs or outputs
stack Stack models/arrays along some array dimension
lti/*inv* Inverse of an LTI system
conj Complex conjugation of model coefficients

Matrix Equation Solvers

bdschur Block diagonalization of a square matrix
care, dare Solve algebraic Riccati equations
gcare, gdare Solve generalized algebraic Riccati equations
lyap, dlyap Solve Lyapunov equations
lyapchol, dlyapschol Square-root Lyapunov equations

Command-Line Plot Customization

bodeplot Bode magnitude and phase plus plot handle
getoptions Get the plot options handle
hsvplot Hankel singular value plus plot handle
impulseplot Impulse response plus plot handle
initialplot Initial condition plus plot handle
iopzplot Pole/zero maps for input/output pairs plus plot handle
lsimplot Time response to arbitrary inputs plus plot handle
nicholsplot Nichols plot plus plot handle
nyquistplot Nyquist plus plot handle
pzplot Pole/zero plus plot handle
rlocusplot Root locus plus plot handle
setoptions Set plot options
sigmaplot Singular values of the frequency response plus plot handle
stepplot Step response plus plot handle