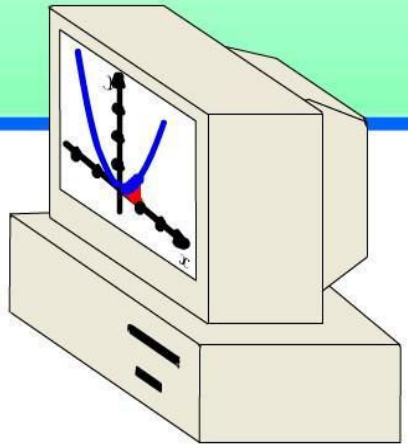
# محاسبات سمبلیک در

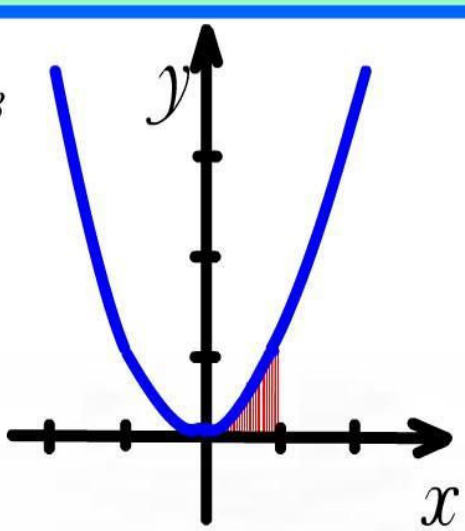
# MATLAB

## راهنمای استفاده از جعبه ابزار ریاضیات سمبلیک

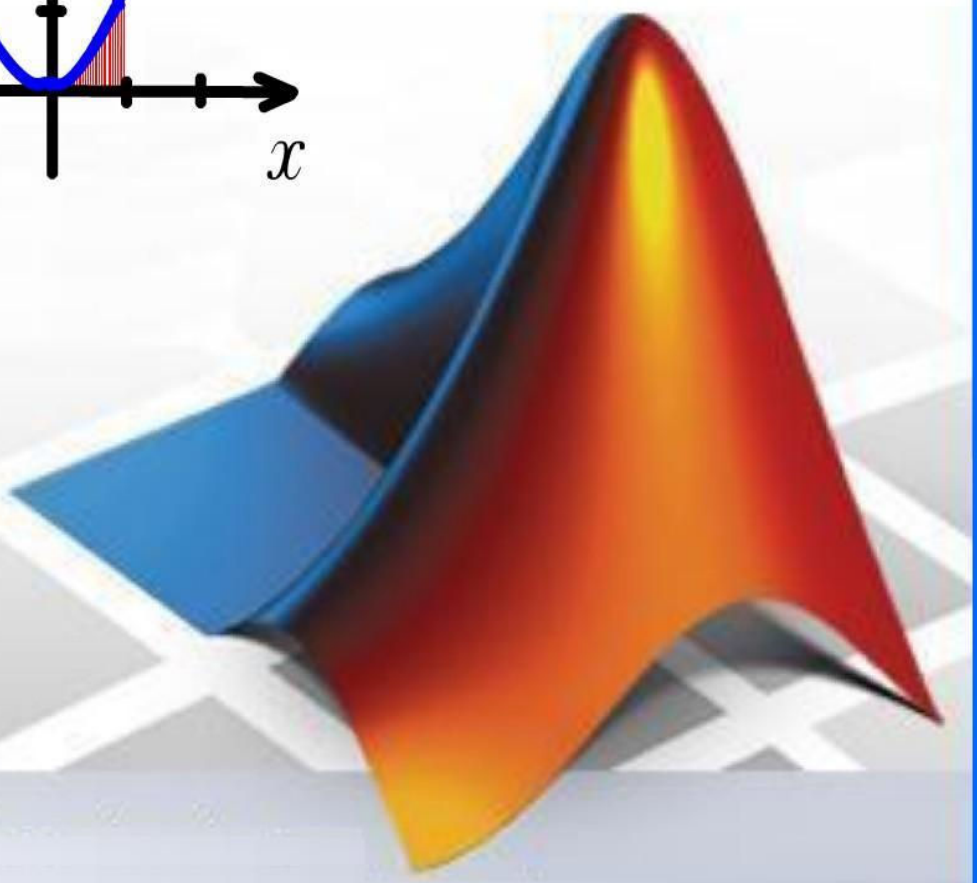
ترجمه: منصور خالقی ، سیف الله رمضان زاده



$$\int x^2 = \frac{1}{3}x^3$$




```
>> quad('x.^2',0,1)

 ans =

    0.3333

>>syms x

>>int(x^2)

ans=

    1/3*x^3
```



The MathWorks

به نام خدا

# محاسبات سمبلیک در MATLAB

## راهنمای استفاده از جعبه ابزار ریاضیات سمبلیک

ترجمه: منصور خالقی ، سیف الله رمضان زاده

The MathWorks

# مقدمه

کتابی که ترجمه شده است نحوه ی استفاده از جعبه ابزار ریاضیات سمبلیک را در MATLAB شرح می دهد. این جعبه ابزار به شما این امکان را می دهد که عملیات ریاضی مانند دیفرانسیل، انتگرال، حد و ... را به صورتی انجام دهید که جواب معادله به صورت عددی نباشد و بر حسب یک نماد مانند $x$ نمایش داده شود. به عنوان مثال اگر بخواهید یک انتگرال نامعین را محاسبه کنید، مطمئنا جوابی را که انتظار دارید بدست بیاورید یک عدد نخواهد بود و عبارتی بر حسب متغیر مستقل انتگرال خواهد بود، در اینجاست که مبحث محاسبات سمبلیک مطرح می شود. در این نوع محاسبات بر روی نماد ها عملیات انجام می شود و جواب نهایی هم یک عبارت از نمادها(سمبل ها) است. اگرچه هنوز معماهای زیادی در مورد محاسبات سمبلیک در کامپیوتر هنوز بدون پاسخ مانده اند، اما پیشرفت های چشم گیری در این زمینه حاصل شده است که MATLAB نیز به لطف استفاده از هسته ی Maple از آنها بهره مند شده است.

این کتاب می تواند برای دانشجویان رشته ی ریاضی و تمامی رشته های مرتبط با ریاضیات بسیار مفید واقع شود. امیدواریم که این کتاب مقدمه ای را برای پیشرفت علمی هرچند ناچیز این سرزمین ایفا کند. ما معتقدیم که دادن زکات علم وظیفه ی هر شخص مسلمان است و خوانندگان محترم اگر ما را دعا کنند بر ما منت نهاده اند.

با تشکر از همه ی کسانی که مشوق ما در تهیه ی این مجموعه بودند،

منصور خالقی، دانشجوی کارشناسی علوم کامپیوتر، دانشگاه تربیت معلم تهران. mansoorexpert@Gmail.com

سیف الله رمضان زاده، دانشجوی مهندسی نرم افزار، دانشگاه تربیت معلم تهران. safecomp85@Gmail.com

# فهرست مطالب


مقدمه ای بر جعبه ابزار ریاضیات سمبلیک ..... ۱

**فصل ۱ – آشنایی با جعبه ابزار ریاضیات سمبلیک و اشیاء سمبلیک ..... ۵**

**بخش ۱** – اشیاء سمبلیک ..... ۷

**بخش ۲** – ایجاد متغیر ها و عبارات سمبلیک ..... ۹

**بخش ۳** – تبدیل متغیر ها ی سمبلیک و عددی ..... ۱۱

ساختن متغیر های حقیقی و مختلط ..... ۱۲

ایجاد کردن توابع سمبلیک (نامعین) ..... ۱۳

استفاده از sym برای دسترسی به توابع Maple ..... ۱۳

متغیر های سمبلیک مستقل ..... ۱۵

**بخش۴** – ایجاد توابع سمبلیک ..... ۱۷

استفاده از عبارات سمبلیک ..... ۱۷

به وجود آوردن M-file ..... ۱۷

**فصل ۲ – استفاده از جعبه ابزار ریاضیات سمبلیک ..... ۱۸**

**بخش۱** – حساب دیفرانسیل و انتگرال ..... ۱۹

مشتق گیری ..... ۱۹

حد ها ..... ۲۱

انتگرال ..... ۲۲

مجموع سری ..... ۲۵

سری تیلور ..... ۲۵

مثال از حساب دیفرانسیل ..... ۲۶

**بخش۲** – ساده سازی و جایگزینی ..... ۳۵

ساده سازی ..... ۳۵

تابع collect ..... ۳۵

تابع expand ..... ۳۶
تابع horner ..... ۳۶
تابع factor ..... ۳۶
تابع simplify ..... ۳۷
تابع simple ..... ۳۸
جایگزینی ..... ۳۹
تابع subexpr ..... ۳۹
تابع sub ..... ۴۰

**بخش۳** – محاسبات دقت متغیر ..... ۴۵
مثال: استفاده از انواع مختلف محاسبه ..... ۴۶
تعداد ارقام در دقت متغیر ..... ۴۶
تبدیل به ممیز شناور ..... ۴۷
مثالی دیگر از محاسبات دقت متغیر ..... ۴۷

**بخش۴** – جبر خطی ..... ۴۹
عملگر های جبری پایه ..... ۴۹
عملگر های جبر خطی ..... ۵۰
مقدار ویژه ..... ۵۲
تابع eig ..... ۵۲
ماتریس روزر ..... ۵۳
شکل استاندارد جردن ..... ۵۶
تجزیه مقادیر منفرد ..... ۵۷
منحنی های مقادیر ویژه ..... ۵۹

**بخش۵** – حل معادلات ..... ۶۷
حل معادلات جبری ..... ۶۷
تابع solve ..... ۶۷
حل چندین معادله جبری ..... ۶۷
معادلات دیفرانسیل ..... ۶۹
تابع dsolve ..... ۶۹

**بخش۶** – توابع ریاضی خاص ..... ۷۳
یک مثال از مسئله ی انعکاس ..... ۷۴

**بخش۷** – استفاده از توابع Maple ..... ۷۷
یک مثال ساده ..... ۷۷
یک مثال از بردارها ..... ۷۹
خطایابی ..... ۸۰
ردیابی کردن ..... ۸۰
بررسی وضعیت آرگومان خروجی ..... ۸۱

**بخش۸** – جعبه ابزار ریاضیات سمبلیک تعمیم یافته ..... ۸۳
package های توابع کتابخانه ای ..... ۸۳
مثال از یک زیر برنامه ..... ۸۴
زیر برنامه های از پیش کامپایل شده ی Maple ..... ۸۶

**فصل ۳ – مرجع توابع ..... ۸۷**
**بخش ۱** – لیست طبقه بندی شده ی توابع ..... ۸۹
توابع حساب دیفرانسیل و انتگرال ..... ۸۹
توابع جبر خطی ..... ۸۹
توابع ساده سازی ..... ۸۹
توابع حل معادله ..... ۹۰
توابع محاسبات دقت متغیر ..... ۹۰
عملگر های حسابی ..... ۹۰
توابع خاص ..... ۹۰
توابع دسترسی به Maple ..... ۹۱
توابع ترسیمی – تعلیمی ..... ۹۱
توابع تبدیلات ..... ۹۱
عملگر های پایه ..... ۹۲
توابع تبدیل انتگرال ..... ۹۲

**بخش ۲** – لیست توابع و شرح توابع به ترتیب حروف الفبا ..... ۹۳
عملگرهای حسابی ..... ۹۳

تابع ccode ..... ۹۵
تابع collect ..... ۹۵
تابع colspace ..... ۹۶
تابع compose ..... ۹۷
تابع conj ..... ۹۸
تابع cosint ..... ۹۸
تابع det ..... ۹۹
تابع diag ..... ۹۹
تابع diff ..... ۱۰۰
تابع digits ..... ۱۰۱
تابع double ..... ۱۰۲
تابع dsolve ..... ۱۰۲
تابع eig ..... ۱۰۴
تابع expm ..... ۱۰۵
تابع expand ..... ۱۰۵
تابع ezcontour ..... ۱۰۶
تابع ezcontourf ..... ۱۰۷
تابع ezmesh ..... ۱۰۹
تابع ezmeshc ..... ۱۱۱
تابع ezplot ..... ۱۱۲
تابع ezplot3 ..... ۱۱۴
تابع ezpolar ..... ۱۱۵
تابع ezsurf ..... ۱۱۶
تابع ezsurfc ..... ۱۱۸
تابع factor ..... ۱۲۰
تابع finsym ..... ۱۲۰
تابع finverse ..... ۱۲۱
تابع fortran ..... ۱۲۲
تابع fourier ..... ۱۲۲
تابع funtool ..... ۱۲۴
تابع horner ..... ۱۲۶
تابع hypergeom ..... ۱۲۷

تابع ifourier ..... ۱۲۸
تابع ilaplace ..... ۱۲۹
تابع imag ..... ۱۳۱
تابع int ..... ۱۳۱
تابع inv ..... ۱۳۲
تابع iztrans ..... ۱۳۳
تابع jacobian ..... ۱۳۴
تابع jordan ..... ۱۳۵
تابع lambertw ..... ۱۳۶
تابع laplace ..... ۱۳۶
تابع latex ..... ۱۳۸
تابع limit ..... ۱۳۹
تابع maple ..... ۱۴۰
تابع mapleinit ..... ۱۴۱
تابع mfun ..... ۱۴۲
تابع mfunlist ..... ۱۴۲
تابع mhelp ..... ۱۴۶
تابع null ..... ۱۴۷
تابع numden ..... ۱۴۸
تابع poly ..... ۱۴۸
تابع poly2sym ..... ۱۴۹
تابع pretty ..... ۱۵۰
تابع procread ..... ۱۵۰
تابع rank ..... ۱۵۱
تابع real ..... ۱۵۲
تابع rref ..... ۱۵۲
تابع rsums ..... ۱۵۲
تابع simple ..... ۱۵۳
تابع simplify ..... ۱۵۴
تابع sinint ..... ۱۵۵
تابع size ..... ۱۵۶
تابع solve ..... ۱۵۶

تابع subexpr ..... ۱۵۸
تابع subs ..... ۱۵۸
تابع svd ..... ۱۵۹
تابع sym ..... ۱۶۱
تابع syms ..... ۱۶۲
تابع sym2poly ..... ۱۶۳
تابع symsum ..... ۱۶۳
تابع taylor ..... ۱۶۴
تابع taylortool ..... ۱۶۶
تابع tril ..... ۱۶۷
تابع triu ..... ۱۶۷
تابع vpa ..... ۱۶۸
تابع zeta ..... ۱۶۹
تابع ztrans ..... ۱۷۰

**راهنمای سازگاری ..... ۱۷۳**
سازگاری با نسخه های قبلی ..... ۱۷۳
توابع منسوخ ..... ۱۷۳

# مقدمه ای بر جعبه ابزار ریاضیات سمبلیک

## جعبه ابزارهای ریاضیات سمبلیک چیستند؟

جعبه ابزار ریاضیات سمبلیک[1] محاسبات سمبلیک را وارد محیط عددی MATLAB® می کند. این جعبه ابزار امکانات عددی و گرافیکی MATLAB را به وسیله ی نوع دیگری از محاسبات ریاضی که در جدول زیر ذکر شده اند تکمیل می کند.

| امکانات ارائه شده | مباحث تحت پوشش |
|---|---|
| حساب دیفرانسیل و انتگرال | مشتق، انتگرال، حد، مجموع سری، سری تیلور |
| جبر خطی | معکوس، دترمینان، مقدار ویژه، تجزیه به مقادیر یکتا و صورت متعارفی یک ماتریس سمبلیک |
| ساده سازی | روش های ساده کردن یک عبارت جبری |
| حل معادلات | راه حل های سمبلیک و عددی معادلات جبری و معادلات دیفرانسیل |
| توابع ریاضی خاص | توابع ریاضی خاص معمول |
| محاسبات دقت متغیر | ارزیابی عددی عبارات ریاضی بر حسب دقت های تعریف شده |
| تبدیل ها | فوریه، لاپلاس، تبدیل z، و تبدیل های معکوس آنها |

موتور محاسباتی که در این جعبه ابزار از آن استفاده می شود هسته ی Maple® می باشد. سیستمی که اولین بار در دانشگاه واترلوی کانادا[2] و اخیرا در بخش فنی کالج ذوریخ توسعه یافت. Maple در معرض فروش قرار گرفت و توسط Waterloo Maple حمایت شد.

این نسخه ها از جعبه ابزار ریاضیات سمبلیک برای کار با MATLAB 6.0 یا نسخه های بالاتر و Maple V نسخه ۵ طراحی شده است.

[1] Symbolic Math Toolbox
[2] Waterloo

## جعبه ابزار های ریاضیات سمبلیک

دو جعبه ابزار وجود دارد:

- جعبه ابزار ریاضیات سمبلیک پایه که مجموعه ای از بیش از ۱۰۰ تابع MATLAB است که دسترسی به هسته ی Maple را به وسیله ی توابعی که به شکل املای توابع MATLAB هستند را فراهم می کند. همچنین این جعبه ابزار به شما اجازه می دهد که دسترسی به توابع جبر خطی Maple داشته باشید.

- جعبه ابزار سمبلیک تعمیم داده شده امکان دسترسی به تمام Package های غیر گرافیکی Maple، ویژگی های برنامه نویسی Maple و زیر برنامه های تعریف شده توسط کاربر در Maple را برای شما فراهم می کند. به وسیله ی این دو جعبه ابزار می توانید M-File هایی را تعریف کنید که در آن به توابع و فضای کاری Maple دسترسی داشته باشید.

اگر شما از قبل یک کپی از نرم افزار Maple V داشته باشید می توانید از آن به جای کتابخانه ای که توسط جعبه ابزار ریاضیات سمبلیک ارائه می شود استفاده کنید، به این ترتیب که در فایل mapleinit.m که در دایرکتوری MATLAB وجود دارد به جای آدرس موجود آدرس محلی که Maple در آن نصب شده است را بنوسید.

## محصولات مرتبط:

شرکت MathWork محصولات متنوعی را ارائه می دهد که به طور ویژه وابسته به نوع کاری که شما از جعبه ابزار ریاضیات سمبلیک می خواهید هستند.

برای مشاهده ی اطلاعات بیشتر در مورد این محصولات می توانید به منابع زیر مراجعه کنید:

- راهنمای آنلاین برنامه، البته در صورتی که برنامه را قبلا نصب کرده باشید.
- وب سایت شرکت MathWork با آدرس http://www.mathworks.com ، بخش محصولات.

## استفاده از این راهنما

این کتاب شامل سه فصل می باشد:

| نام فصل | محتویات |
|---|---|
| آشنایی با جعبه ابزار ریاضیات سمبلیک واشیاء سمبلیک | دستور عملی برای کاربران جدید در استفاده از جعبه ابزار ریاضیات سمبلیک |
| استفاده از جعبه ابزار ریاضیات سمبلیک | توضیح جزئیات استفاده از این جعبه ابزار |
| مرجع توابع | لیست توابع موجود در جعبه ابزار و توضیح آنها |

اگر شما قبلا با جعبه ابزار ریاضیات سمبلیک و اشیاء سمبلیک آشنایی دارید می توانید مطالعه ی این کتاب را از فصل دوم آغاز کنید، در غیر این صورت به شما توصیه می شود که حتما از فصل اول شروع به مطالعه ی این کتاب کنید.

## تکمیل این راهنما به وسیله ی راهنمای MATLAB (دستور help)

برای تکمیل دانسته های خود می توانید علاوه بر استفاده از این راهنما از دستور help در خود MATLAB نیز استفاده کنید. برای این کار دستور help را می توانید به طریق زیر استفاده کنید:

```
>>help function
```

که در آن function نام تابعی از MATLAB است که شما می خواهید راهنمای آن را مشاهده کنید. در این روش MATLAB راهنمای تمام توابع با این نام را نمایش خواهد داد. برای اینکه بخواهید راهنمای توابع با نام مشخص شده را در یک جعبه ابزار خاص جستجو کنید، باید نام آن جعبه ابزار را قبل از نام تابع بنویسید و آنها را با یک علامت / از هم جدا کنید. مثلا اگر تایپ کنید:

```
>>help diff
```

MATLAB راهنمای هر دو تابع char/diff و sym/diff را نمایش خواهد داد، اما با تایپ دستور

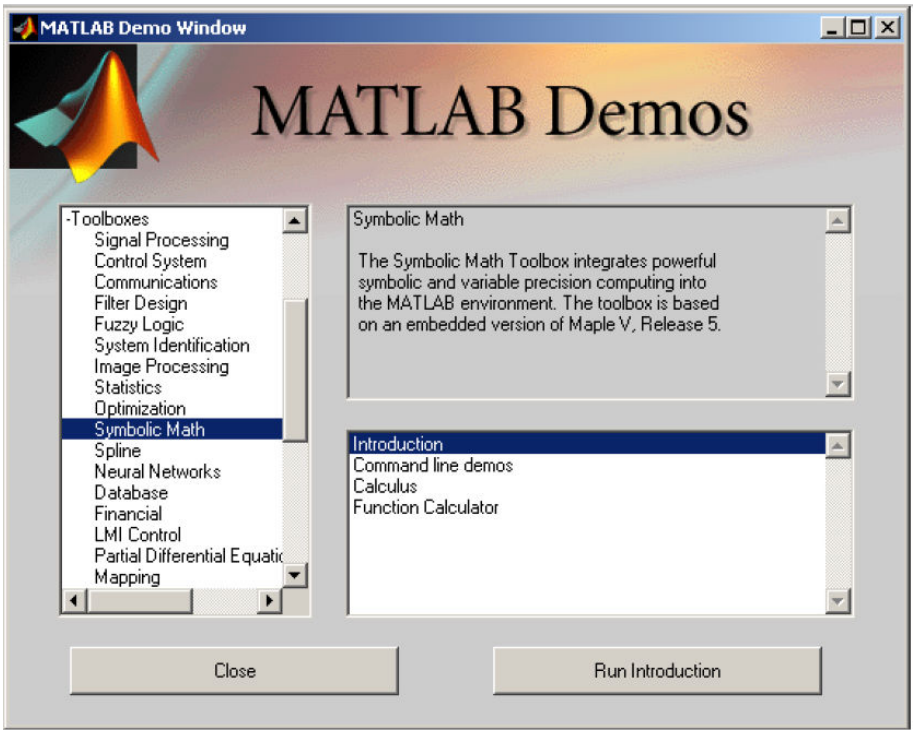
```
>>help sym/diff
```

راهنمای تابع دیفرانسیل را در جعبه ابزار ریاضیات سمبلیک مشاهده خواهیم کرد.

همچنین برای استفاده از راهنمای توابع Maple می توانید به جای دستور help از دستور mhelp استفاده کنید.

## دمو ها

برای مشاهده ی راهنمای سریع آنلاین، می توانید دستور demo را در خط فرمان MATLAB تایپ کنید تا دیالوگ MATLAB Demos نمایش داده شود. در جعبه لیست سمت چپ Symbolic Math و در جعبه لیست سمت راست باید Introduction را انتخاب کنید.



برای تعیین اینکه جعبه ابزار ریاضیات سمبلیک روی سیستم شما نصب شده است یا نه می توانید روی خط فرمان MATLAB دستور زیر را تایپ کنید:

```
>>ver
```

پس از این دستور نسخه نرم افزار MATLAB نصب شده به همراه تمام جعبه ابزار هایی که روی آن نصب شده است چاپ می شود. در صورت عدم مشاهده جعبه ابزار ریاضیات سمبلیک(Symbolic Math Toolbox) می توانید برای نصب آن به راهنمای نصب MATLAB مراجعه کنید.

# فصل ۱

## آشنایی با جعبه ابزار ریاضیات سمبلیک و اشیاء سمبلیک

در این بخش طریقه ی به وجود آوردن اشیاء سمبلیک[1] همچنین متغیر های سمبلیک ذکر می شود. اگر شما یک کپی از نرم افزار Maple دارید بد نیست نگاهی به maple init بیاندازید.

### این فصل شامل بخش های زیر می باشد:

| | |
|---|---|
| **اشیاء سمبلیک:** | توصیف اشیاء سمبلیک و تشریح تفاوت این اشیاء با نوع داده های استاندارد MATLAB . |
| **ایجاد متغیر ها و عبارات سمبلیک:** | تشریح چگونگی ایجاد اشیاء سمبلیک. |
| **تبدیل متغیر ها ی سمبلیک و عددی:** | تشریح چگونگی تبدیل اشیاء سمبلیک و عددی به هم. |
| **ایجاد توابع سمبلیک:** | تشریح چگونگی تعریف توابعی که می توانند روی اشیاء سمبلیک کار کنند. |

[1] Symbolic

# بخش ۱

## اشیاء سمبلیک

جعبه ابزار ریاضیات سمبلیک نوع داده ی جدیدی را در MATLAB تعریف می کند که اشیاء سمبلیک یا sym خوانده می شوند. در MATLAB اشیاء سمبلیک ساختمان داده ای را ارائه می دهند که یک نمایش رشته ای از سمبل های ریاضی را ذخیره می کند. جعبه ابزار ریاضیات سمبلیک از اشیاء سمبلیک برای نشان دادن متغیر ها، عبارات و ماتریس های سمبلیک استفاده می کند. مثال زیر تفاوت بین نوع داده های استاندارد MATLAB (مانند double) را با اشیاء نمادین مشابه آن ها (یا همان symbolic objects) را روشن می کند.

دستور زیر در MATLAB یک عدد دهدهی ممیز شناور (floating-point) بر می گرداند.

```
>>sqrt(2)
ans =
        1.4142
```

حال اگر بخواهیم عدد ۲ را در دستور بالا به یک شی سمبلیک تبدیل کنیم از تابع sym استفاده خواهیم کرد و نتیجه آن را به تابع sqrt ارسال می کنیم:

```
>>a = sqrt(sym(2))
a =
        2^(1/2)
```

نتیجه ای که MATLAB چاپ می کند 2^(1/2) می باشد به معنی $2^{1/2}$ یا همان $\sqrt{2}$ که صورتهای سمبلیک عملگر رادیکال (یا همان ریشه دوم) می باشند که در حقیقت مقدار عددی برای آن ها برگردانده نمی شود.

MATLAB این عبارت سمبلیک را در یک رشته ذخیره می کند که به صورت 2^(1/2) نمایش داده می شود. البته شما همیشه می توانید ارزش عددی یک عبارت سمبلیک را با استفاده از تابع double بدست آورید:

```
>>double(a)
ans =
        1.4142
```

به خاطر داشته باشید که در دستور double(a) ، a یک عبارت سمبلیک است؛ در مواردی که a یک عبارت رشته ای باشد توابع str2double , str2num , evalc , eval , feval نیز کارهایی مشابه تابع double را در مورد رشته ها انجام می دهند که می توانید در مورد آن ها به MATLAB Help مراجعه کنید.

وقتی که شما یک کسر شامل اشیاء سمبلیک ایجاد می کنید، MATLAB صورت (numerator) و مخرج (denominator) کسر را ذخیره می کند. برای مثال:

```
>>sym(2)/sym(5)
ans =
        2/5
```

عملیات حسابی که MATLAB روی اشیاء سمبلیک انجام می دهد، از عملیاتی که روی نوع داده های استاندارد انجام می دهد کاملا متفاوت است. اگر شما دو کسر از نوع داده ی double داشته باشید، MATLAB پاسخ جمع این دو کسر را به صورت یک کسر دهدهی (با مخرج ۱۰، یا همان اعشار در مبنای ۱۰) خواهد داد:

```
>>2/5 + 1/3
ans =
        0.7333
```

حال اگر شما همین دو کسر را با اشیاء سمبلیک ایجاد و با هم جمع کنید، MATLAB یک مخرج مشترک برای دو کسر در نظر خواهد گرفت و سپس کسر ها را با هم ترکیب و طبق رویه ی معمول جمع دو عدد، صورت ها را با هم جمع می کند:

```
>>sym(2)/sym(5) + sym(1)/sym(3)
ans =
        11/15
```

جعبه ابزار ریاضیات سمبلیک شما را قادر می سازد تا محاسبات سمبلیک گوناگونی را که در علوم و ریاضیات به وجود می آید انجام دهید. در فصل ۲ در این رابطه به طور مفصل بحث خواهد شد.

# بخش ۲

## ایجاد متغیر ها و عبارات سمبلیک

دستور sym این امکان را برای شما فراهم می سازد تا متغیر ها و عبارات سمبلیک بسازید. برای مثال دستورات x = sym('x') و a = sym('alpha') متغیر سمبلیک x که با x و متغیر سمبلیک a که با alpha نمایش داده می شود را به وجود می آورد. فرض کنید که می خواهید یک متغیر سمبلیک برای نمایش دادن "نسبت طلایی" داشته باشید:

$$p = \frac{1+\sqrt{5}}{2}$$

دستور rho = sym('(1 + sqrt(5))/2') این امر را برای شما ممکن می سازد. حال شما می توانید عملیات ریاضی مختلفی را روی متغیر rho که یک متغیر سمبلیک است انجام دهید. برای مثال:

```
>>f = rho^2 - rho - 1
f =
        (1/2+1/2*5^(1/2))^2-3/2-1/2*5^(1/2)
```

پس از این دستور، دستور simplify(f) مقدار صفر را بر می گرداند.

حال فرض کنید که می خواهید تابع درجه دوم $f = ax^2 + bx + c$ را بررسی کنید. دستور f=sym('a*x^2+b*x+c') عبارت $ax^2 + bx + c$ به صورت سمبلیک تعریف و در متغیر f ذخیره می شود. توجه داشته باشید که در عبارت بالا متغیر های سمبلیک a و b و c و x ایجاد نخواهند شد و شما برای انجام عملیات ریاضی روی f ناچارید این متغیر ها را نیز صریحا به صورت سمبلیک تعریف کنید. این کار را می تواید با تایپ دستورات زیر انجام دهید:

```
>>a = sym('a')
>>b = sym('b')
>>c = sym('c')
>>x = sym('x')
```

و از آنجایی که ما همیشه دوست داریم از دستورات خلاصه تر استفاده کنیم می توانیم این چهار دستور را در یک دستور خلاصه کنیم:

```
>>syms a b c x
```

به طور کلی شما می توانید با استفاده از دستورات sym یا syms متغیر های symbolic را ایجاد کنید. ما به شما دستور syms را پیشنهاد می کنیم، فقط به خاطر زحمت تایپ کمتر!

# بخش ۳

## تبدیل متغیر ها ی سمبلیک و عددی

یک کمیت کاملا معمولی در MATLAB را در نظر بگیرید:

```
>>t = 0.1
```

تابع sym چهار فرمت خروجی برای نشان دادن مقدار عددی t به صورت سمبلیک ارائه می دهد:

۱) یک نمایش سمبلیک و floating-point برای مقدار t بر می گرداند:

```
>>sym(t,'f')
ans=
        '1.999999999999a'*2^(-4)
```

۲) یک نمایش سمبلیک و به صورت گویا (rational) برای مقدار t بر می گرداند. این فرمت خروجی (یعنی `r`) همان فرمت پیش فرض برای خروجی sym می باشد:

```
>>sym(t,'r')
ans=
        1/10
>>sym(t)
ans =
        1/10
```

۳) مقدار بازگردانده شده نمایش سمبلیک و کسری t می باشد به علاوه ی اختلاف بین مقدار نظری عبارت کسری t و مقدار حقیقی ممیز شناور t (در ماشین)، که به صورت کسری از eps می باشد.

```
>>sym(t,'e')
ans =
        1/10+eps/40
```

۴) دستور زیر که بسط دهدهی عدد t را تا ۳۲ رقم با معنا به صورت سمبلیک چاپ می کند:

```
>>sym(t,'d')
ans =

        .10000000000000000555111512312578
```

با اجرای دستور >>digits(n) قبل از دستور بالا می توان نمایش تعداد ارقام با معنای درست را به n رقم تغییر داد:

```
>>digits(7)
>>sym(t,'d')
ans =
```

.1000000

یک استفاده ی ویژه و موثر از sym تبدیل ماتریس عددی به فرم سمبلیک آن است. دستور >>hilb(3) یک ماتریس Hilbert با ابعاد 3 در 3 تولید می کند:

```
>>A = hilb(3)
A =
        1.0000        0.5000        0.3333
        0.5000        0.3333        0.2500
        0.3333        0.2500        0.2000
```

سمبلیک کردن ماتریس A نتیجه ی زیر را در پی دارد:

```
>>A = sym(A)

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

## ساختن متغیر های حقیقی و مختلط

در دستور sym استفاده از کلمه ی 'real' به شما اجازه می دهد تا مشخصات ریاضی متغیر های سمبلیک را مشخص کنید. بدین ترتیب که دستورات

```
>> x = sym('x','real'); y = sym('y','real');
```

یا به طور خلاصه دستور >> syms x y real متغیر های سمبلیک x و y را از نوع حقیقی تعریف می کند. واضح است که در این حالت متغیر f=x^2+y^2 اکیدا نامنفی است. می توان متغیر z که یک متغیر مختلط است را به این طریق ساخت:

```
>>z = x + i*y
```

تابع conj مزدوج یک عدد مختلط را برمی گرداند. مطمئنا اگر conj را برای مقدار حقیقی x صدا بزنیم مقدار بازگردانده شده همان x خواهد بود.

```
>>conj(z)

ans =

        x - i*y

>>a=z*conj(z)

a=

        (x + y * i) * (x – y * i)
```

پس از اجرای دستور روبرو اجرای تابع simplify نیز همان نتیجه ای را خواهد داشت که اجرای تابع expand خواهد داشت، یعنی ساده کردن عبارت سمبلیک:

```
>>expand(a)

ans=

        x^2 + y^2
```

حال بیایید خصوصیت حقیقی بودن x را از آن بگیریم، کار چندان سختی نیست، با تعریف مجدد x به وسیله ی دستور sym این کار ممکن است، البته این بار با استفاده از کلمه ی 'unreal' :

```
>>syms x unreal  % or x = sym('x','unreal')
```

توجه داشته باشید که دستور >>clear x ، x را مختلط نمی کند! فقط این متغیر را از فضای کاری MATLAB حذف می کند.

## ایجاد کردن توابع سمبلیک (نامعین)

اگر می خواهید یک تابع نامعین و سمبلیک ایجاد کنید دستور زیر را تایپ کنید:

```
>>f = sym('f(x)')
```

از این به بعد f مانند تابع f(x) عمل می کند و می تواند از طریق دستورات مختلف ساخته شود، به عنوان مثال به دستورات زیر توجه کنید:

```
>>syms x h

>> df = (subs(f,x,x+h)-f)/h

df =

        (f(x+h)-f(x))/h
```

در این رابطه تابع subs برای محاسبه های مربوط به فوریه ،لاپلاس و تبدیل z مفید می باشد.

## استفاده از sym برای دسترسی به توابع Maple

می توان به راحتی و به طریقی که می بینید به تابع فاکتوریل Maple با استفاده از sym دسترسی پیدا کرد:

```
>>kfac = sym('k!')
```

مثلا برای محاسبه ی !6 و !n می توان نوشت:

```
>>syms k n
>>subs(kfac,k,6), subs(kfac,k,n)    %   or you can type subs(kfac,6) , subs(kfac,n)
ans =
        720
ans =
        n!
```

یا اگر می خواهید برای مثال !12 را حساب کنید، از تابع prod هم می توانید استفاده کنید. تابع prod یک ماتریس به عنوان ورودی می گیرد و حاصل ضرب اعضای آن را بر می گرداند.

```
>>prod(1:12)

ans=

  479001600
```

یک مثال از ایجاد یک ماتریس سمبلیک :

ماتریس مدور ماتریسی است که هر سطر آن از سطر های قبلی به دست می آید. بدین ترتیب که سطر i ام از به چپ شیفت دادن سطر i-1 ام (به اندازه ی یک ستون) بدست می آید. ما اینجا با استفاده از دستورهای زیر ماتریس A را با اعضای a و b و c ایجاد کرده ایم:

```
>>syms a b c
>>A = [a b c; b c a; c a b]
A =
        [ a, b, c ]
        [ b, c, a ]
        [ c, a, b ]
```

از آنجایی که A مدور است مجموع عناصر هر سطر یا ستون با مجموع عناصر هر یک از دیگر سطر ها و ستون ها برابر است. اجازه بدهید صحت این موضوع را برای سطر اول و ستون دوم بررسی کنیم:

```
>>sum(A(1,:))
ans =
        a+b+c
>>sum(A(1,:)) == sum(A(:,2)) % This is a logical test.
ans =
        1
```

حال عنصر A(2,3) را با beta و بقیه ی عناصر b در A را با alpha جایگزین می کنیم:

```
>>syms alpha beta;
>>A(2,3) = beta;
>>A = subs(A,b,alpha)
A =
        [      a, alpha,      c ]
        [  alpha,      c,   beta ]
        [      c,      a,  alpha ]
```

از این مثال می توان نتیجه گرفت که استفاده از متغیر های سمبلیک در MATLAB بسیار شبیه استفاده از متغیر های عددی معمول MATLAB می باشد.

## متغیر های سمبلیک مستقل (پیش فرض)

طریقه ی انتخاب متغیر های مستقل در هنگام ساختن توابع ریاضی با توجه به توضیحات روشن است. برای مثال به عبارات جدول زیر توجه کنید:

| Mathematical Function | MATLAB Command |
|---|---|
| $f = x^n$ | `f = x^n` |
| $g = \sin(at+b)$ | `g = sin(a*t + b)` |
| $h = J_v(z)$ | `h = besselj(nu,z)` |

اگر ما بخواهیم از این عبارات بدون مشخص کردن متغیر مستقل مشتق بگیریم، طبق تعریف ریاضی بدست می آوریم: $f' = nx^{n-1}$ و $g' = \mathrm{acos}(at+b)$ و $h' = j_v(z)\left(\frac{v}{z}\right) - j_{v+1}(z)$ . در اینجا ما متغیر های مستقل را در عبارات بالا به ترتیب x ، t و z در نظر گرفتیم (یعنی نسبت به این متغیر ها مشتق گرفتیم). بقیه ی سمبل های a ، b ، n و v معمولا ثابت(constant) در نظر گرفته می شوند. با این وجود اگر بخواهیم بر حسب n مشتق بگیریم، می توانیم بنویسیم: $\frac{d}{dn}\mathrm{f(x)}$ یا $\frac{dx^n}{dn}$ که نتیجه ی آن $x^n \ln x$ است.

از دید ریاضی اغلب حروف کوچک الفبای لاتین که نزدیک x ، y یا z قرار دارند، متغیر های مستقل (متغیر های پیش فرض) در نظر گرفته می شوند. تابع findsym هم از این قاعده برای مشخص کردن متغیر های پیش فرض در حساب دیفرانسیل و انتگرال، ساده سازی، حل معادله و توابع transform استفاده می کند. دستورات زیر مثال مناسبی برای این مبحث می باشند:

```
>>syms a b n nu t x z
>>f = x^n; g = sin(a*t + b); h = besselj(nu,z);
```

این دستورات عبارت های سمبلیک f ، g و h را مطابق با معادل ریاضی آن ها می سازد. برای مشتق گرفتن از این عبارات می توانیم از تابع diff استفاده کنیم:

```
>>diff(f)
ans =
        x^n*n/x
```

با استفاده از دستور زیر می توانیم متغیری از f را که متغیر پیش فرض برای مشتق گیری می باشد مشاهده کنیم:

```
>>findsym(f,1)
ans =
        x
```

به طور مشابه findsym(g,1) و findsym(h,1) به ترتیب مقادیر t و z را چاپ می کنند. در واقع دومین آرگومان ارسالی به تابع findsym شماره ی اولویت متغیری را از f که باید برگردانده شود مشخص می کند. اگر تابع findsym را بدون آرگومان دوم صدا بزنیم، مقدار بازگردانده شده لیست تمام متغیر های سمبلیک موجود در آرگومان ورودی تابع خواهد بود:

```
>>findsym(g)
ans =
        a, b, t
```

**نکته:** متغیر های سمبلیک پیش فرض در یک عبارت سمبلیک حروف و یا کارکتر هایی هستند که از لحاظ الفبایی به 'x' نزدیک ترند. اگر فاصله ی دو حرف از 'x' با هم برابر باشد، متغیری پیش فرض در نظر گرفته می شود که از لحاظ الفبایی بعد از 'x' قرار دارد. در جدول زیر چند مثال برای این موضوع ذکر شده است.

| Expression | Variable Returned by findsym |
|---|---|
| x^n | x |
| sin(a*t+b) | t |
| besselj(nu,z) | z |
| w*y + v*z | y |
| exp(i*theta) | theta |
| log(alpha*x1) | x1 |
| y*(4+3*i) + 6*j | y |
| sqrt(pi*alpha) | alpha |

## بخش ۴

# ایجاد توابع سمبلیک

دو راه برای ایجاد توابع وجود دارد: استفاده از عبارات سمبلیک و ایجاد کردن M-file

**استفاده از عبارات سمبلیک :** سری دستورات زیر عبارات سمبلیک r ، t و f را به وجود می آورند. شما می توانید از توابع diff ، int ، subs و بقیه ی توابع سمبلیک برای درست کردن چنین عباراتی استفاده کنید.

```
>>syms x y z
>>r = sqrt(x^2 + y^2 + z^2);
>>t = atan(y/x);
>>f = sin(x*y)/(x*y);
```

**به وجود آوردن M-file :** M-file ها باعث می شوند که استفاده ی کلی تری از توابع صورت گیرد. برای مثال فرض کنید که می خواهید تابع sinc را به صورت sinc(x)=sin(x)/x تعریف و ایجاد کنید. برای انجام این کار یک M-file با کد زیر می توان نوشت:

```
function z = sinc(x)
%SINC The symbolic sinc function
%     sin(x)/x. This function
%     accepts a sym as the input argument.
if isequal(x,sym(0))
   z = 1;
else
   z = sin(x)/x;
end
```

شما می توانید این مثال را برای توابعی که روی چند متغیر کار می کنند نیز بسط دهید.

# فصل ۲

## استفاده از جعبه ابزار ریاضیات سمبلیک

**این فصل شامل بخش های زیر می باشد:**

**حساب دیفرانسیل و انتگرال:** مشتق، انتگرال، حد، مجموع سری، سری تیلور.

**ساده سازی و جایگزینی:** روش های ساده کردن عبارات جبری.

**محاسبات دقت متغیر:** ارزیابی عددی عبارات ریاضی بر حسب دقت های تعریف شده.

**جبر خطی:** معکوس، دترمینان، مقدار ویژه، تجزیه به مقادیر یکتا، صورت متعارفی ماتریس های سمبلیک.

**حل معادلات:** حل عددی و سمبلیک معادلات جبری و معادلات دیفرانسیل.

**توابع ریاضی خاص:** توابع ریاضی خاص معمول.

**استفاده از توابع Maple:** چگونگی استفاده از دستور maple برای دسترسی مستقیم به توابع Maple .

**جعبه ابزار ریاضیات سمبلیک تعمیم یافته:**

چگونگی دسترسی به Package های غیر گرافیکی Maple ، ویژگی های برنامه نویسی Maple و زیر برنامه های Maple.

# بخش ۱

## حساب دیفرانسیل و انتگرال

در فصل قبل تا حدودی با جعبه ابزار ریاضیات سمبلیک در MATLAB آشنا شدید.این جعبه ابزار توابعی را برای اعمال عملگرهای پایه در حساب دیفرانسیل و انتگرال ارائه می دهد. اعمالی مانند مشتق گیری، انتگرال گیری، محاسبه سری[1] ، بسط تیلور و . . .

در این بخش یک زمینه ی کلی در مورد این توابع به شما معرفی می کنیم.

### ۲-۱-۱) مشتق گیری:

اجازه دهید یک عبارت سمبلیک ایجاد کنیم.

```
>>syms a x
>>f = sin(a*x)
>>diff(f)
ans=
        cos(a*x)*a
```

طبق طریقه ی تصمیم گیری که در فصل قبل مشخص شد، تابع diff(f) هم متغیر پیش فرض را در f متغیر 'x' در نظر می گیردو نسبت به آن مشتق می گیرد. می توانیم تابع diff را به صورت زیر صدا بزنیم:

```
>>diff(f,a)
ans =
        cos(a*x)*x
```

برای محاسبه ی مشتق دوم ( یا nام) می توانیم n را به عنوان آرگومان ورودی بعدی به تابع diff ارسال کنیم :

```
>>diff(f,2)     % or  diff(f,x,2)
ans =
        -sin(a*x)*a^2
>>diff(f,a,2)
ans =
        -sin(a*x)*x^2
```

متغیرهای a , b , x , n , t را در محیط MATLAB به صورت سمبلیک تعریف کنید.جدول زیر مثال هایی از تابع diff را بیان می کند:

| f | diff(f) |
|---|---|
| x^n | x^n*n/x |
| sin(a*t+b) | cos(a*t+b)*a |
| exp(i*theta) | i*exp(i*theta) |

دستورات زیر مشتق تابع بسل نسبت به z را بدست می آورد:

```
>>syms nu z
>>b = besselj(nu,z);
>>db = diff(b)
db =
```

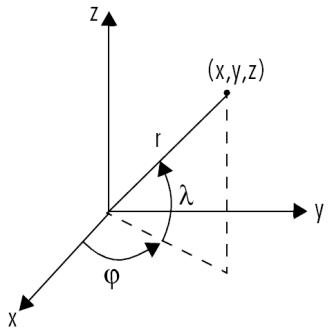
[1] Summation

```
-besselj(nu+1,z)+nu/z*besselj(nu,z)
```

تابع diff() همچنین می تواند یک ماتریس سمبلیک را به عنوان ورودی دریافت کند.در این حالت از تک تک اعضای ماتریس ورودی مشتق گرفته می شود و حاصل به صورت یک ماتریس با همان ابعاد برگردانده می شود. مثال زیر را ملاحظه کنید:

```
>>syms a x
>>A = [cos(a*x),sin(a*x);-sin(a*x),cos(a*x)]
A =
        [ cos(a*x),  sin(a*x)]
        [ -sin(a*x), cos(a*x)]
>>diff(A)
ans =
        [ -sin(a*x)*a , cos(a*x)*a]
        [ -cos(a*x)*a, -sin(a*x)*a]
```

علاوه بر این شما می توانید از یک ستون نسبت به یک سطر مشتق بگیرید. در اینجا می توانید تبدیل مختصات از مختصات اقلیدسی $(x, y, z)$ به مختصات کروی $(r,\lfloor,\prod)$ را مشاهده کنید.در این تبدیل $x=r\cos\lfloor\cos\prod$ و $y=r\cos\lfloor\sin\prod$ و $z=r\sin\lfloor$ است. به خاطر داشته باشید که $\lfloor$ بیانگر زاویه ی عمودی و $\prod$ مشخص کننده ی زاویه ی افقی است:




برای محاسبه ی ماتریس ژاکوبین J در این تبدیل می توانید از تابع jacobian استفاده کنید. تعریف ریاضی برای J چنین است:

$$J = \frac{\partial(x, y, x)}{\partial(r, \lambda, \varphi)}$$

ما چون از حروف لاتین استفاده می کنیم به جای لامبدا از L و به جای فی از f استفاده می کنیم. حال به دستورات زیر توجه کنید:

```
>>syms r l f
>>x = r*cos(l)*cos(f); y = r*cos(l)*sin(f); z = r*sin(l);
>>J = jacobian([x; y; z], [r l f])
J =
        [ cos(l)*cos(f), -r*sin(l)*cos(f), -r*cos(l)*sin(f)]
        [  cos(l)*sin(f),  -r*sin(l)*sin(f), r*cos(l)*cos(f)]
        [          sin(l),          r*cos(l),                0]

>>detJ = simple(det(J))
detJ =
        -cos(l)*r^2
```

نکته ی قابل تذکر این است که آرگومان اول تابع jacobian باید یک بردار ستونی و دومین آرگومان باید یک بردار سطری باشد.علاوه بر این تا زمانی که دترمینان ژاکوبین یک عبارت مثلثاتی نسبتا پیچیده است، ما از تابع simple برای جایگزین کردن عبارات پیچیده با عبارات ساده استفاده می کنیم. بخش "ساده سازی و جایگزینی" در مورد ساده سازی با جزئیات بیشتری بحث خواهد کرد.

| Mathematical Operator | MATLAB Command |
|---|---|
| $\frac{df}{dx}$ | diff(f) or diff(f,x) |
| $\frac{df}{da}$ | diff(f,a) |
| $\frac{d^2f}{db^2}$ | diff(f,b,2) |
| $J = \frac{\partial(r,t)}{\partial(u,v)}$ | J = jacobian([r:t],[u,v]) |

## ۲-۱-۲) حد ها :

ایده ی اساسی در حساب دیفرانسیل و انتگرال محاسبات روی توابع برای به دست آوردن مقدار آن ها یا بررسی نزدیک شدن مقدار توابع به یک مقدار معین است. حتما به خاطر دارید که تعریف مشتق، خود به وسیله ی بیان یک limit انجام می پذیرد.به شرط اینکه این حد موجود باشد داریم: $f'(x) = \lim_{h\to 0}\frac{f(x+h)-f(x)}{h}$

جعبه ابزار ریاضیات سمبلیک شما را قادر می سازد که حد برخی توابع را محاسبه کنید. دستورات زیر می تواند راهنمای خوبی برای شما در استفاده از تابع limit باشد:

```
>>syms h n x
>>limit( (cos(x+h) - cos(x))/h,h,0 )
ans =
        -sin(x)
>>limit( (1 + x/n)^n,n,inf )
ans =
        exp(x)
```

دو مثال ذکر شده در بالا از مهمترین مثال های حد در ریاضیات می باشند. مثال اول محاسبه ی مشتق تابع از طریق تعریف مشتق ( در اینجا تابع cos(x) ) و مثال دوم در مورد تابع نمایی می باشد.

در حالی که بسیاری از حد ها به صورت $\lim_{x\to a} f(x)$ دو طرفه اند (بدین معنی که نزدیک شدن تابع به a از چپ و راست یک نتیجه دارد و حد چپ و راست تابع با هم برابرند) ، حد توابع بخصوصی چنین نیستند. از این رو سه حد $\lim_{x\to 0}\frac{1}{x}$ و $\lim_{x\to 0^-}\frac{1}{x}$ و $\lim_{x\to 0^+}\frac{1}{x}$ سه نتیجه ی مجزا در پی خواهد داشت که به ترتیب عبارتند از: تعریف نشده، $-\infty$ و $+\infty$.

در مواردی که حد تابع تعریف نشده باشد جعبه ابزار ریاضیات سمبلیک مقدار NaN (not a number) را بر می گرداند. بر این اساس دستور limit(1/x,x,0) یا limit(1/x) مقدار ans= NaN را برگشت خواهد داد. با توضیحات داده شده نتایج دستورات زیر کاملا توجیه پذیر است:

```
>>limit(1/x,x,0,'left')
ans =
        -inf
>>limit(1/x,x,0,'right')
ans =
        inf
```

مشاهده کردید که پیش فرض دستور limit(f) این است که x به سمت صفر میل کند، یعنی همان دستور limit(f,x,0) .

در جدول زیر تنظیمات مربوط به دستور limit را مشاهده می کنید. در اینجا ما فرض کرده ایم که f یک تابع از یک شی *سمبلیک* به نام x است ( f(x) ) .

| Mathematical Operation | MATLAB Command |
|---|---|
| $\lim_{x \to 0} f(x)$ | `limit(f)` |
| $\lim_{x \to a} f(x)$ | `limit(f,x,a)` or `limit(f,a)` |
| $\lim_{x \to a-} f(x)$ | `limit(f,x,a,'left')` |
| $\lim_{x \to a+} f(x)$ | `limit(f,x,a,'right')` |

### ۲-۱-۳) انتگرال :

*اگر* f *یک عبارت سمبلیک باشد آنگاه* int(f) *یک عبارت سمبلیک دیگر به نام* F *بر می گرداند، به طوری که :* diff(F)= f .
یعنی int(f) انتگرال نامعین f را (در صورت وجود) برمی گرداند. درست مثل مشتق، دستور int(f,v) انتگرال نامعین f را نسبت به متغیر *سمبلیک* v برمی گرداند. در ضمن دستور int(f) متغیر مستقل را طبق قوانین پیدا کردن متغیر مستقل در تابع findsym تعیین می کند و نسبت به آن انتگرال می گیرد.
در جدول زیر طریقه ی کار تابع int را ملاحظه می کنید:

| Mathematical Operation | MATLAB Command |
|---|---|
| $\int_{\pi/2} x^n dx = \frac{x^{n+1}}{n+1}$ | `int(x^n)` or `int(x^n,x)` |
| $\int_0 \sin(2x)dx = 1$ | `int(sin(2*x),0,pi/2)` or `int(sin(2*x),x,0,pi/2)` |
| $g = \cos(at+b)$ <br> $\int g(t)dt = \sin(at+b)/a$ | `g = cos(a*t + b)` <br> `int(g)` or `int(g,t)` |
| $\int J_1(z)dz = -J_0(z)$ | `int(besselj(1,z))` or `int(besselj(1,z),z)` |

در مقایسه با مشتق، عمل انتگرال گیری *سمبلیک* عمل پیچیده تری است. برای بدست آوردن تابع اولیه ی f (یعنی F) همیشه روش معینی وجود ندارد و ممکن است که تابع اولیه ی F یک تابع ناشناخته باشد. در واقع تابع اولیه ممکن است وجود داشته باشد ولی نرم افزار قادر به پیدا کردن آن نباشد. اما گذشته از این انتگرال های پیچیده MATLAB در بسیاری مواقع قادر است انتگرال های نامعین را با موفقیت محاسبه کند. برای مثال متغیرهای *سمبلیک* زیر را ایجاد کنید:

```
>>syms a b theta x y n x1 u z
```

جدول زیر مثال هایی از انتگرال عباراتی است که شامل متغیر های *سمبلیک* بالا می باشد.

| f | int(f) |
|---|---|
| x^n | x^(n+1)/(n+1) |
| y^(-1) | log(y) |
| n^x | 1/log(n)*n^x |
| sin(a*theta+b) | -1/a*cos(a*theta+b) |
| exp(-x1^2) | 1/2*pi^(1/2)*erf(x1) |
| 1/(1+u^2) | atan(u) |

آخرین مثال نشان می دهد که وقتی MATLAB نتواند تابع اولیه ی f یعنی F را پیدا کند چه اتفاقی می افتد، در این حالت MATLAB به سادگی همان دستور انتگرال را که شامل متغیرهای *سمبلیک* است بر می گرداند، بدون اینکه تغییری کرده باشد!

و اما انتگرال معین روی یک بازه. دستور int(f,a,b) یا int(f,v,a,b) به ترتیب برای پیدا کردن عبارات *سمبلیک* زیر استفاده می شود: $\int_a^b f(x)dx$ و $\int_a^b f(v)dv$

در اینجا چند مثال اضافی را نیز در این مورد مشاهده می کنید:

| f | a, b | int(f,a,b) |
|---|---|---|
| x^7 | 0, 1 | 1/8 |
| 1/x | 1, 2 | log(2) |
| log(x)*sqrt(x) | 0, 1 | -4/9 |
| exp(-x^2) | 0, inf | 1/2*pi^(1/2) |
| besselj(1,z) | 0, 1 | 1/4*hypergeom([1],[2, 2],-1/4) |

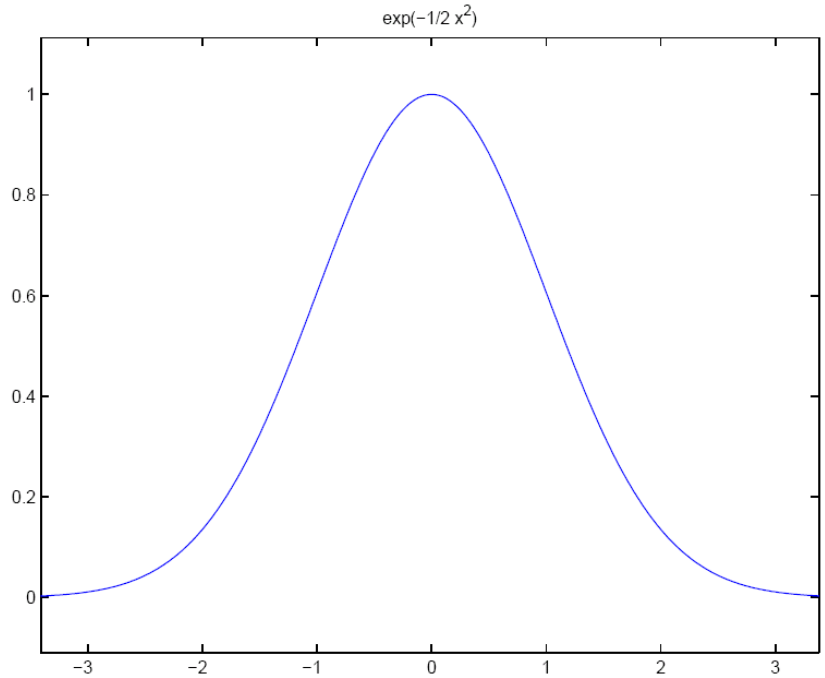
در مورد مثال تابع بسل (besselj) باید گفت که این تابع امکان محاسبه ی یک تقریب عددی از مقدار انتگرال را به وجود می آورد. برای این منظور می توان از تابع double() استفاده کرد. به دستورات زیر توجه کنید:

```
>>a = int(besselj(1,z),0,1)
a =
        1/4*hypergeom([1],[2, 2],-1/4)
>>a = double(a)
a =
        0.2348
```

یکی از موضوعات ظریف و مورد بحث در انتگرال معین تعیین محدوده ی مقدار پارامتر های مختلف می باشد. برای مثال عبارت $e^{-(kx)^2}$ یک عبارت بزرگتر از صفر با منحنی به شکل زنگوله است که برای هر عدد حقیقی k وقتی $x \to \pm\infty$ میل کند، به سمت صفر میل می کند. یک مثال از این منحنی در زیر با فرض $k = \frac{1}{\sqrt{2}}$ رسم شده است.

```
>>syms x
>>k = sym(1/sqrt(2));
>>f = exp(-(k*x)^2);
```

```
>>ezplot(f)
```




هسته ی Maple با عبارات $x^2$ یا $k^2$ به عنوان عبارات مثبت رفتار می کند. در مقابل Maple فرض می کند که متغیر های *سمبلیک* k و x نامعین اند. یعنی این دو رسما متغیر به شمار می روند بدون هیچ گونه مشخصات ریاضی. حال برای شروع کار جهت محاسبه ی انتگرال $\int_{-\infty}^{\infty} e^{-(kx^2)}\, dx$ در جعبه ابزار ریاضیات سمبلیک از دستورات زیر استفاده کنید:

```
>>syms x k;
>>f = exp(-(k*x)^2);
>>int(f,x,-inf,inf)

Definite integration: Can't determine if the integral is convergent.
Need to know the sign of --> k^2
Will now try indefinite integration and then take limits.
Warning: Explicit integral could not be found.
ans =
        int(exp(-k^2*x^2),x= -inf..inf)
```

MATLAB (به دلیل مشخص نبودن علامت $k^2$) پس از چاپ مقداری پیام و هشدار جواب را به صورتی که مشاهده می کنید برمی گرداند. در ادامه خواهید دید که چگونه می توان یک متغیر حقیقی k ایجاد کرد تا بتوان مطمئن بود که $k^2$ همیشه نامنفی است.

توجه داشته باشید که Maple قادر به پیدا کردن علامت عبارت $k^2$ نیست. سوالی که در اینجا مطرح می شود این است که چگونه می توان این مانع را برطرف کرد؟ پاسخ ایجاد کردن متغیر k به صورت حقیقی (real) و با استفاده از تابع sym است. قبلا دیدید که چگونه می توان با استفاده از تابع sym یک متغیر سمبلیک را از نوع حقیقی (real) تعریف کرد. در نتیجه انتگرال بالا در MATLAB به این صورت محاسبه می شود:

```
>>syms k real
>>int(f,x,-inf,inf)
```

```
ans =
        signum(k)/k*pi^(1/2)
```

توجه کنید که اکنون k یک شئ *سمبلیک* در فضای کاری MATLAB و یک متغیر حقیقی در فضای کاری هسته ی Maple می باشد.با تایپ دستور clear k شما فقط k را از فضای کاری MATLAB حذف می کنید. برای اطمینان از اینکه k دیگر هیچ ویژگی خاصی (مثل حقیقی بودن) ندارد، از دستور زیر استفاده کنید:

```
>>syms k unreal
```

این تغییر با استفاده از دستور syms، k را از فضای کاری Maple حذف می کند. در مورد دستور syms به یاد داشته باشید که می توان چندین متغیر را یکجا به صورت حقیقی یا غیر حقیقی تعریف کرد.

```
>>syms w x y z real
```

در این حالت تمام متغیرهای نوشته شده بین دستور syms و real خصوصیت حقیقی بودن را دارا می شوند، یعنی آن ها به عنوان متغیر های real در فضای کاری Maple تعریف می شوند.

### ۲-۱-۴) مجموع سری :

شما می توانید مجموع یک سری *سمبلیک* را در صورت وجود محاسبه کنید. این امکان به وسیله ی تابع symsum به وجود می آید. برای مثال سری $\pi$ به صورت $1+\frac{1}{2^2}+\frac{1}{3^2}+\cdots$ که در نهایت به مقدار $\pi^2/6$ میل می کند یا سری $1+x^2+x^3+\cdots \quad ,|x|<1$ که در نهایت به $1/(1-x)$ میل می کند.

طریقه ی استفاده از تابع symsum با مثالی از محاسبه ی ۲ سری در زیر ذکر شده است:

```
>>syms x k
>>s1 = symsum(1/k^2,1,inf)
>>s2 = symsum(x^k,k,0,inf)
s1 =
        1/6*pi^2
s2 =
        -1/(x-1)
```

### ۲-۱-۵) سری تیلور :

دستورات زیر را در نظر بگیرید:

```
>>syms x
>>f = 1/(5+4*cos(x))
```

```
>>T = taylor(f,8)
T =
        1/9+2/81*x^2+5/1458*x^4+49/131220*x^6
```

از نام تابع taylor() به راحتی می توان برداشت کرد که مقدار برگشت داده شده توسط این تابع بسط تیلور تابع f می باشد.

حتما به خاطر دارید که تعریف ریاضی برای بسط تیلور به این صورت می باشد: $\sum_{n=0}^{\infty}(x-a)\frac{f^{(n)}(a)}{n!}$

در دستورات قبل T بیانگر بسط تیلور تابع f() حول نقطه ی a=0 می باشد. دستور pretty(T) ، T را در یک قالب ریاضیاتی چاپ می کند که ما هم در اینجا از آن استفاده می کنیم:

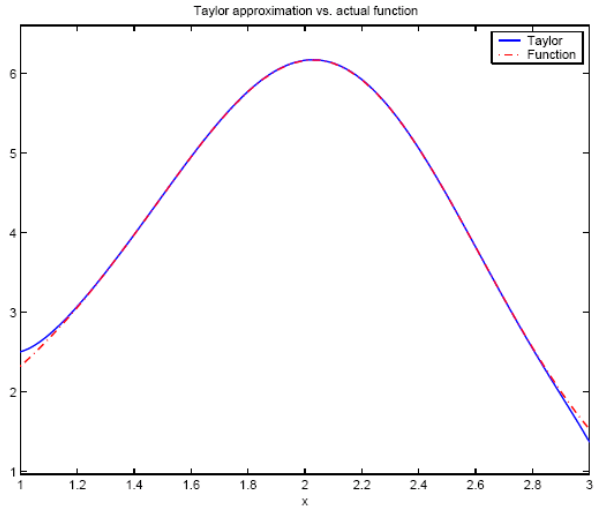
```
>>pretty(T)
                2              4    49     6
1/9 + 2/81 x + 5/1458 x + ------ x
                              131220
```

دستورات زیر ۱۲ جمله ی اول غیر صفر از بسط تیلور تابع g حول نقطه ی x=2 را بدست می دهد:

```
>>syms x
>>g = exp(x*sin(x))
>>t = taylor(g,12,2);
```

حال بیایید نمودار تابع را با بسط تیلور آن مقایسه کنیم. برای این کار این دو منحنی را به وسیله ی دستورات زیر رسم می کنیم:

```
>>xd = 1:0.05:3; yd = subs(g,x,xd);
>>ezplot(t, [1,3]); hold on;
>>plot(xd, yd, 'r-.')
>>title('Taylor approximation vs. actual function');
>>legend('Taylor','Function')
```

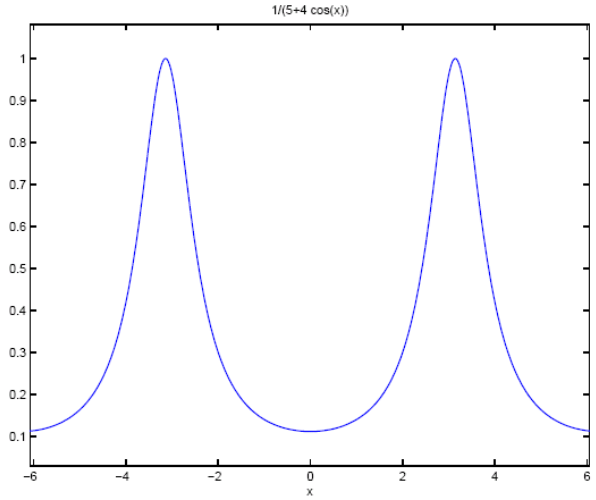



## ۲-۱-۶) مثال از حساب دیفرانسیل :

تابع $f(x)=\frac{1}{5+4\cos(x)}$ نقطه ی شروع خوبی است برای توضیح دادن چندین تابع حساب دیفرانسیل و انتگرال در این جعبه ابزار . البته این تابع ذاتا تابع جالبی است! دستورات

```
>>syms x
>>f = 1/(5+4*cos(x))
```

یک تعریف سمبلیک از تابع f را در محیط MATLAB ایجاد می کند. تابع ezplot(f) نمودار f(x) را به صورتی که نشان داده شده است رسم می کند:




تابع ezplot سعی بر این دارد تا تابع f را در یک بازه ی قابل قبول از محور x ها و محور y ها رسم کند. این بازه ها می توانند با اضافه کردن بازه ها به عنوان آرگومان ورودی یا به وسیله ی دستور axis تغییر داده شوند. دامنه ی پیش فرض x در تابع ezplot ، $-2\pi \le x \le 2\pi$ می باشد. شما می توانید برای رسم یک منحنی از f(x) با شرط $a \le x \le b$ به این صورت عمل کنید: ezplot(f,[a b])

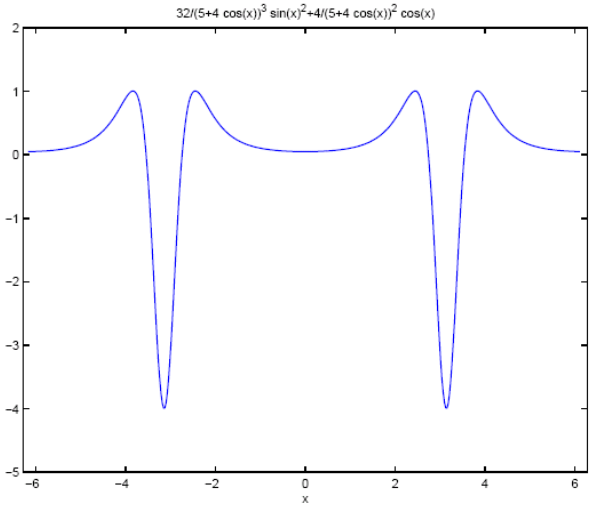
حال بیایید نگاهی هم به مشتق دوم تابع f بیاندازیم:

```
>>f2 = diff(f,2)
f2 =
        32/(5+4*cos(x))^3*sin(x)^2+4/(5+4*cos(x))^2*cos(x)
```

یاد آوری می کنیم که معادل این دستور می توانستیم از diff(f,x,2) استفاده کنیم که همین نتیجه را در بر داشت.

مقیاس پیش فرض تابع ezplot در اینجا یک تکه ی بریده شده از تابع f2 را به ما نشان خواهد داد. بنا بر این حدود و بازه ها را در اینجا به صورت دستی ست می کنیم تا بتوانیم کل نمودار را به درستی مشاهده کنیم.

```
>>ezplot(f2)
>>axis([-2*pi 2*pi -5 2])
```



با توجه به شکل مشخص می شود که مقادیر $f''(x)$ بین 4- و 1 ظاهر می شوند. اما فقط در بازه ی مشخص شده ی x . در صورت خروج از این بازه این نتیجه گیری درست نیست. ما می توانیم با محاسبه ی ماکزیمم و مینیمم یک بازه ی دقیق برای تغییرات تابع $f''(x)$ بدست آوریم. می دانیم که مینیمم و ماکسیمم تابع $f''(x)$ در صفر های تابع $f'''(x)$ اتفاق می افتد. پس دستورات زیر $f'''(x)$ را بدست آورده و آن را در یک قالب ریاضیاتی نمایش می دهند:

```
>>f3 = diff(f2);
>>pretty(f3)
```

$$384\,\frac{\sin(x)}{(5+4\cos(x))^4} + 96\,\frac{\sin(x)\cos(x)}{(5+4\cos(x))^3} - 4\,\frac{\sin(x)}{(5+4\cos(x))^2}$$

در ضمن می توانیم با استفاده از دستورات زیر عبارات بدست آمده را ساده کنیم:

```
>>f3 = simple (f3);
>>pretty (f3)
```

$$4\,\frac{\sin(x)\,(96\sin(x)^2 + 80\cos(x) + 80\cos(x)^2 - 25)}{(5+4\cos(x))^4}$$

حال با استفاده از تابع solve می توانیم صفرهای تابع $f'''(x)$ را بدست آوریم،

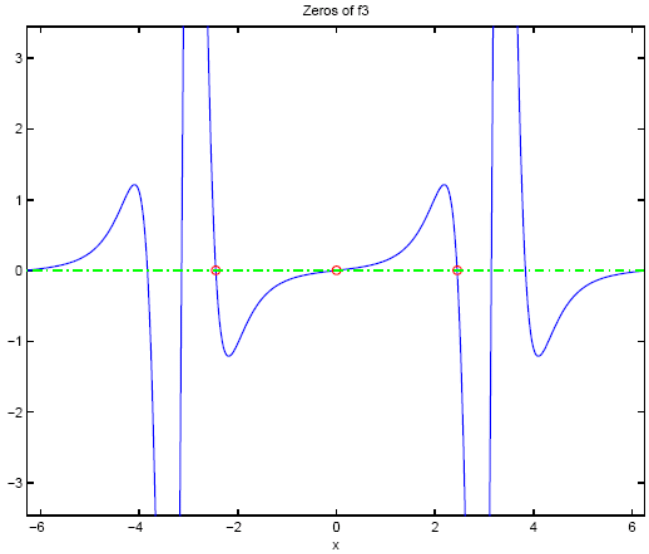
```
>>z = solve(f3)
z =
        [                                                    0 ]
        [ atan((-255-60*19^(1/2))^(1/2),          10+3*19^(1/2)) ]
        [ atan(-(-255-60*19^(1/2))^(1/2),         10+3*19^(1/2)) ]
        [  atan((-255+60*19^(1/2))^(1/2)/(10-3*19^(1/2)))+pi   ]
        [  -atan((-255+60*19^(1/2))^(1/2)/(10-3*19^(1/2)))-pi   ]
```

و با تبدیل نوع سمبلیک به double خواهیم داشت:

```
>>format;       % Default format of 5 digits
>>zr = double(z)
zr =
        0
        0+ 2.4381i
        0- 2.4381i
        2.4483
        -2.4483
```

به این ترتیب ما ۳ ریشه ی حقیقی و ۲ ریشه ی مختلط خواهیم داشت، با این وجود نمودار f3 نشان می دهد که ما هنوز تمام ریشه ها را پیدا نکرده ایم:

```
>>ezplot(f3)
>>hold on;
>>plot(zr,0*zr,'ro')
>>plot([-2*pi,2*pi], [0,0],'g-.');
>>title('Zeros of f3')
```

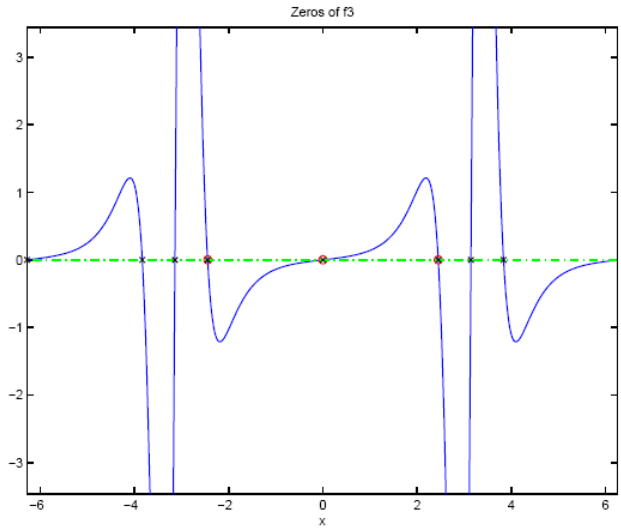



دلیل این موضوع این است که $f'''(x)$ شامل یک فاکتور از $\sin(x)$ می باشد که در ضرایب صحیح $2\pi$ صفر است. با این وجود تابع solve(sin(x)) فقط ریشه های نزدیک $x = 0$ را بر می گرداند. می توانیم لیست کاملی از ریشه های حقیقی را به وسیله ی zr بدست آوریم.

```
>>zr = [0 zr(4) pi 2*pi-zr(4)]
>>zr = [zr-2*pi zr zr+2*pi];
```

حال بیایید نمودار تغییر شکل یافته ی zr را برای یک تصویر کامل از صفر های f3 رسم کنیم:

```
>>plot(zr,0*zr,'kX')
```



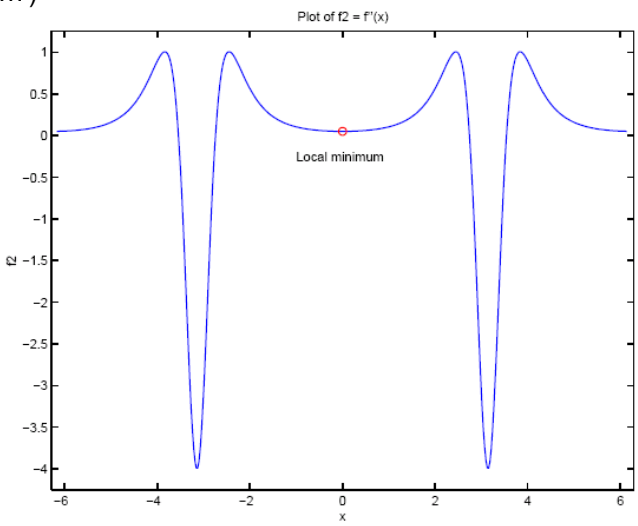

اولین ریشه ی پیدا شده از $f'''(x)$ توسط تابع solve برابر $x = 0$ است. ما تابع f2 را بر حسب $x = 0$ محاسبه کرده ایم:

```
>>f20 = subs(f2,x,0)
f20 =
        0.0494
```

نگاهی به نمودار $f'''(x)$ نشان می دهد که این فقط یک مینیمم نسبی است که ما به وسیله ی رسم نمودار f2 این مطلب را نشان می دهیم:

```
>>clf
>>ezplot(f2)
>>axis([-2*pi 2*pi -4.25 1.25])
>>ylabel('f2');
>>title('Plot of f2 = f'''(x)')
>>hold on
```

```
>>plot(0,double(f20),'ro')
>>text(-1,-0.25,'Local minimum')
```



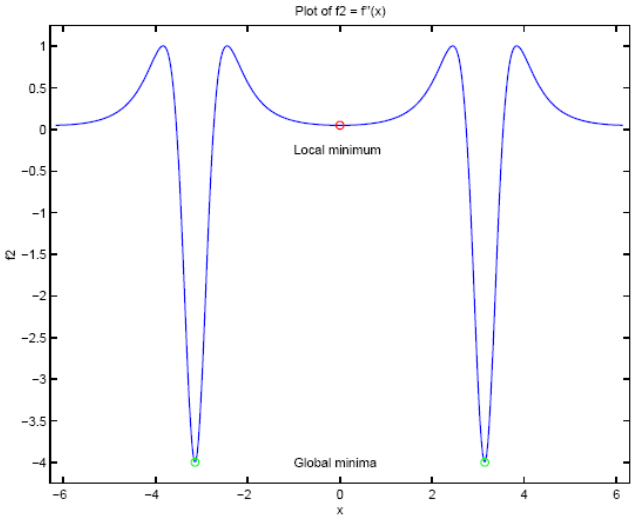

نمودار رسم شده نشان می دهد که مینیمم تابع نزدیک $x=-\pi$ و $x=\pi$ اتفاق می افتد. ما می توانیم به وسیله ی سری دستورات زیر ثابت کنیم که مینیمم ها دقیقا روی $x=\pm\pi$ اتفاق می افتند. ابتدا مقدار تابع $f'''(x)$ را به ازای $x=-\pi$ و $x=\pi$ بدست می آوریم:

```
>>simple([subs(f3,x,-sym(pi)),subs(f3,x,sym(pi))])
ans =
        [ 0, 0]
```

پس $x=-\pi$ و $x=\pi$ نقاط بحرانی تابع $f''(x)$ می باشند. می توانیم به وسیله ی رسم f2(-pi) و f2(pi) روی f2(x) ببینیم که π و –π مینیمم های مطلق تابع $f''(x)$ می باشند.

```
>>m1 = double(subs(f2,x,-pi));  m2 = double(subs(f2,x,pi));
>>plot(-pi,m1,'go',pi,m2,'go')
>>text(-1,-4,'Global minima')
>>[m1,m2]
ans =
        [ -4, -4]
```

که در نمودار زیر نشان داده شده است:



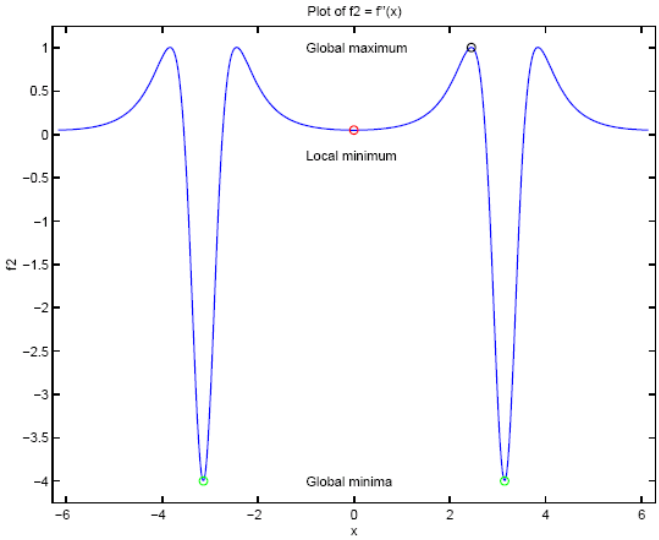

محاسبات انجام شده بخشی از حدس کلی ما را در مورد تابع $f''(x)$ تایید می کند که این بخش همان بازه ی [-4, 1] است. می توانیم درستی این حدس را در مورد بازه های دیگر به وسیله ی امتحان کردن چهارمین ریشه ی $f'''(x)$ که به وسیله ی تابع solve بدست آمده است تحقیق کنیم. به سری دستورات زیر توجه کنیم:

```
>>s = z(4)
s =
        atan((-255+60*19^(1/2))^(1/2)/(10-3*19^(1/2)))+pi
>>sd = double(s)
sd =
        2.4483
```

حال به وسیله ی دستورات زیر نقطه ی (s, f2(s)) را روی نمودار رسم شده ی f2 رسم می کنیم:

```
>>M1 = double(subs(f2,x,s));
>>plot(sd,M1,'ko')
>>text(-1,1,'Global maximum')
```

به طور نظری تایید می شود که s یک ماکزیمم است.




ماکزیمم برابر است با M1 = 1.0051.

بنا بر این حدس ما در مورد بازه ی شامل ماکزیمم $f''(x)$ که [-4, 1] بسیار نزدیک به واقعیت است اما درست نیست. بازه ی درست [-4, 1.0051] می باشد. حال بیایید ببینیم که آیا انتگرال دوگانه ی $f''(x)$ بر حسب x دوباره تابع اصلی $f(x) = \frac{1}{5+4\cos(x)}$ را بر می گرداند؟

```
>>g = int(int(f2))
g =
        -8/(tan(1/2*x)^2+9)
```

که مطمئنا عبارت بدست آمده همان عبارت اولیه نیست. حال به تفاضل $f(x) - g(x)$ توجه کنید:

```
>>d = f - g
>>pretty(d)
                1                 8
            ------------ + ---------------
            5 + 4 cos(x)              2
                            tan(1/2 x)   + 9
>>simple(d)  % or simplify(d)
ans =
        1
```

این مثال این موضوع را روشن می سازد که گرفتن انتگرال دوگانه از مشتق دوم $f(x)$ تابعی را تولید می کند که ممکن است با $f(x)$ تفاوت داشته باشد و اختلاف آن با $f(x)$ یک تابع خطی از x است.

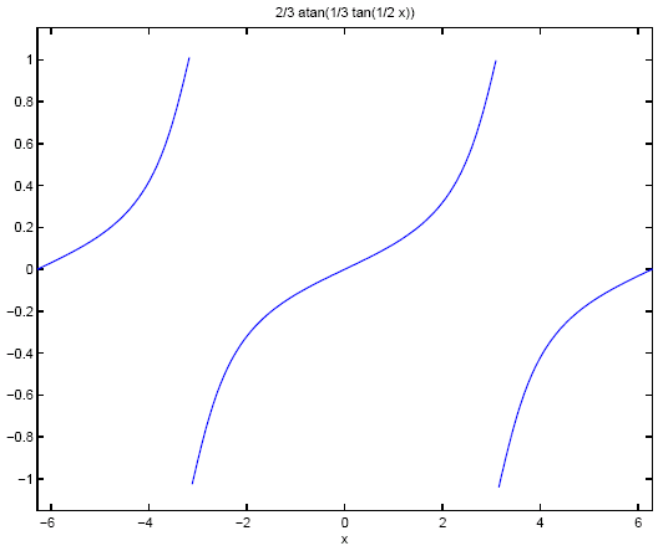
در آخر با یک بار انتگرال گیری از $f(x)$ خواهیم دید که F شامل تابع آرکتانژانت می باشد:

```
>>F = int(f)
F =
```

```
2/3*atan(1/3*tan(1/2*x))
```

$F(x)$ تابع اولیه یک تابع پیوسته است، اما خود $F(x)$ همانگونه که در شکل می بینید پیوسته نیست:

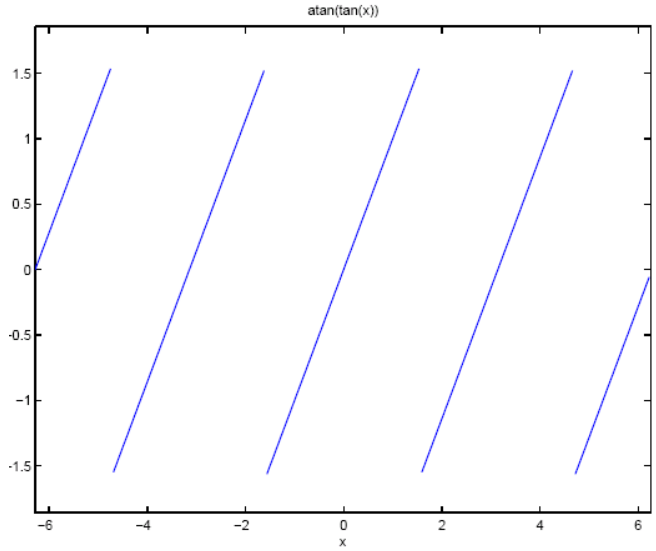
```
>>ezplot(F)
```




به خاطر داشته باشید که $F(x)$ در نقاط $\pm\pi$ پرش (ناپیوستگی) دارد. زیرا تابع tanx در نقاط $\pm\pi/2$ ناپیوستگی دارد. در حقیقت همانگونه که دستور

```
>>ezplot(atan(tan(x)))
```

نشان می دهد، مقدار atan(tan(x)) با مقدار x یکی نیست و در ضرایب فرد $\pi/2$ تعریف نشده است و در این نقاط ناپیوستگی دارد.

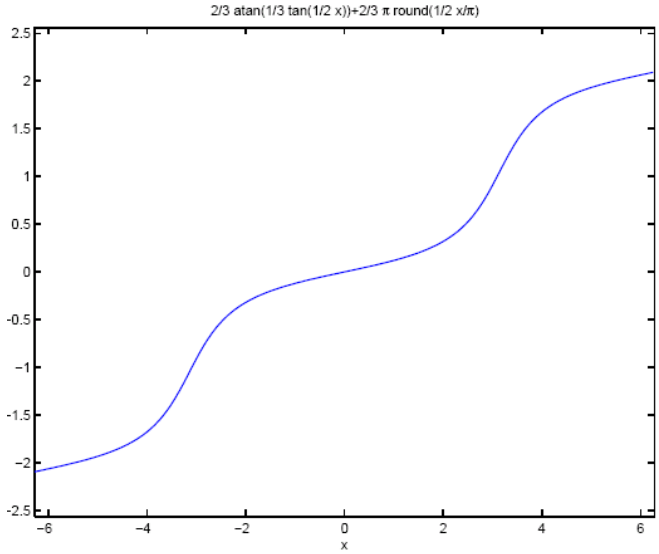



برای بدست آوردن نمایشی از $F(x)$ که در این نقاط ناپیوستگی نداشته باشد، باید تابعی مانند $J(x)$ تولید کنیم که ناپیوستگی ها را ترمیم کند! سپس ضریبی از $J(x)$ را به $F(x)$ اضافه کنیم.

```
>>J = sym('round(x/(2*pi))');
>>c = sym('2/3*pi');
>>F1 = F+c*J
F1 =
        2/3*atan(1/3*tan(1/2*x))+2/3*pi*round(1/2*x/pi)
```

و با رسم F1 خواهیم دید که نمودار F1 ناپیوستگی ندارد.

```
>>ezplot(F1,[-6.28,6.28])
```




توجه داشته باشید که در تابع رسم شده از بازه ی [-6.28, 6.28] استفاده شده است.

# بخش ۲

## ساده سازی و جایگزینی [1]

چندین تابع برای ساده کردن عبارات سمبلیک وجود دارند که برای عمل جایگذاری در عبارات سمبلیک نیز از آن ها استفاده می شود.

### ۲-۲-۱)ساده سازی:

در اینجا سه عبارت سمبلیک مختلف مشاهده می کنید:

```
>>syms x
>>f = x^3-6*x^2+11*x-6
>>g = (x-1)*(x-2)*(x-3)
>>h = x*(x*(x-6)+11)-6
```

و در زیر همین عبارات را در یک قالب ریاضیاتی که به وسیله ی دستورات pretty(f), pretty(g), pretty(h) به وجود آمده اند، ملاحظه می کنید:

```
  3       2
 x  - 6 x   + 11 x - 6

 (x - 1) (x - 2) (x - 3)

 x (x (x - 6) + 11) - 6
```

عباراتی که ملاحظه می کنید همگی نمایش های مختلف از یک تابع چند جمله ای درجه ۳ می باشند. هریک از این ۳ عبارت در شرایط مختلف می تواند نسبت به سایر عبارات، عبارت برتر شناخته شود. شکل تابع f معمول ترین فرم نمایش این چند جمله ای است. f یک ترکیب خطی ساده از توان های x می باشد. شکل تابع g فرم فاکتور گیری شده ی همین تابع می باشد که ریشه های چند جمله ای را نمایش می دهد و دقیق ترین فرم برای بدست آوردن مقادیر نزدیک به ریشه می باشد. اما اگر یک چند جمله ای ریشه هایی به سادگی ریشه های g نداشته باشد، فرم فاکتور گیری شده ی آن به سادگی دست یافتنی نخواهد بود. و اما شکل تابع h که نمایش تو در توی همان چند جمله ای می باشد که برای بدست آوردن مقدار عددی عبارت شامل کمترین عملگرهای ریاضی است.

مسئله ی ساده کردن عبارات سمبلیک به بررسی این موضوع می پردازد که آیا عبارت های سه تابع بالا یک تابع را نشان می دهند یا خیر، عبارت ساده شده شامل متغیر های کمتری است یا نه و یا اصلا کدامیک از عبارت های بدست آمده ساده ترین عبارت است؟

توابعی که در این بخش معرفی می کنیم شامل توابعی هستند که قوانین جبری و مثلثاتی مختلفی را روی عبارت مورد نظر اعمال می کنند تا عبارت یا تابع را از یک شکل به شکل های دیگر و شاید ساده تر تبدیل کنند. توابع expand, collect horner, factor, simplify و simple توابعی هستند که در این بخش معرفی می کنیم.

**تابع collect:** اجرای تابع collect(f) باعث می شود که کلیه ی ضرایب یک توان مشخص از x یکجا به عنوان ضریب آن توان از x جمع آوری شوند. در واقع دستور collect(f) عبارت f را به یک چند جمله ای استاندارد برحسب x تبدیل می

[1] Simplifications and Substitutions

کند. البته پارامتر دومی که به تابع ارسال می شود تعیین می کند که عبارت مورد نظر نسبت به چه متغیری ساده شود که در صورت ارسال نشدن پارامتر دوم، طبق قوانینی که در فصل قبل برای تعیین پارامتر اصلی ذکر شد، عمل می شود. جدول زیر چند خروجی این تابع را به ازای مقادیر ورودی نوشته شده نشان می دهد:

| f | collect(f) |
|---|---|
| (x-1)*(x-2)*(x-3) | x^3-6*x^2+11*x-6 |
| x*(x*(x-6)+11)-6 | x^3-6*x^2+11*x-6 |
| (1+x)*t + x*t | 2*x*t+t |

**تابع expand** : برای این تابع تقریبا می توان گفت که عملگرهای دیگر را در عبارت روی عملگر جمع و تفریق پخش می کند، جدول زیر چند مثال از این تابع را نشان می دهد:

| f | expand(f) |
|---|---|
| a*(x + y) | a*x + a*y |
| (x-1)*(x-2)*(x-3) | x^3-6*x^2+11*x-6 |
| x*(x*(x-6)+11)-6 | x^3-6*x^2+11*x-6 |
| exp(a+b) | exp(a)*exp(b) |
| cos(x+y) | cos(x)*cos(y)-sin(x)*sin(y) |
| cos(3*acos(x)) | 4*x^3-3*x |

**تابع horner** : این تابع عبارت را به عبارتی با x های تو در تو ترجمه می کند، جدول زیر را ببینید:

| f | horner(f) |
|---|---|
| x^3-6*x^2+11*x-6 | -6+(11+(-6+x)*x)*x |
| 1.1+2.2*x+3.3*x^2 | 11/10+(11/5+33/10*x)*x |

**تابع factor** : این تابع یک عبارت چند جمله ای را به حاصل ضرب چند عبارت چند جمله ای با درجه ی کمتر تجزیه می کند. اگر عبارت f در فراخوانی تابع factor(f) قابل فاکتور گیری نباشد یا MATLAB قادر به انجام آن نباشد خود عبارت f برگردانده می شود، چند مثال را در جدول زیر مشاهده می کنید:

| f | factor(f) |
|---|---|
| x^3-6*x^2+11*x-6 | (x-1)*(x-2)*(x-3) |
| x^3-6*x^2+11*x-5 | x^3-6*x^2+11*x-5 |
| x^6+1 | (x^2+1)*(x^4-x^2+1) |

به عنوان مثالی دیگر در زیر یک ماتریس ۹ در ۲ ایجاد کرده ایم که عنصر اول سطر n ام x^n+1 و عنصر دوم همان سطر عبارت فاکتورگیری شده ی عبارت اول می باشد.

```
>>syms x ;
>>n = (1:9)' ;
>>p = x.^n + 1 ;
>>f = factor(p) ;
>>[p, f]
ans=
        [      x+1    ,                                  x+1  ]
        [      x^2+1 ,                                x^2+1 ]
        [      x^3+1 ,                        (x+1)*(x^2-x+1) ]
        [      x^4+1 ,                                x^4+1 ]
        [      x^5+1 ,                 (x+1)*(x^4-x^3+x^2-x+1)  ]
        [       x^6+1,                      (x^2+1)*(x^4-x^2+1)  ]
        [      x^7+1 ,          (x+1)*(1-x+x^2-x^3+x^4-x^5+x^6)  ]
        [      x^8+1 ,                                x^8+1  ]
        [      x^9+1 ,               (x+1)*(x^2-x+1)*(x^6-x^3+1) ]
```

علاوه بر این تابع factor می تواند برای تجزیه ی یک عدد صحیح به عوامل اول تشکیل دهنده ی آن به کار رود، قطعه کد زیر که می تواند در یک m-file نوشته شود، یک ماتریس مانند ماتریس بالا ایجاد می کند، با این تفاوت که درایه ی اول هر سطر یک عدد صحیح است.

```
N = sym(1);
for k = 2:11
        N(k) = 10*N(k-1)+1;
end
[N' factor(N')]
```

که نتیجه ی آن به صورت زیر می باشد:

```
[                  1,                                1]
[                 11,                             (11)]
[                111,                          (3)*(37)]
[               1111,                       (11)*(101)]
[              11111,                       (41)*(271)]
[             111111,              (3)*(7)*(11)*(13)*(37)]
[            1111111,                     (239)*(4649)]
[           11111111,              (11)*(73)*(101)*(137)]
[          111111111,                (3)^2*(37)*(333667)]
[         1111111111,             (11)*(41)*(271)*(9091)]
[        11111111111,                  (513239)*(21649)]
```

**تابع simplify:** این تابع یکی از توابع قوی و چند منظوره جهت ساده سازی عبارات می باشد که جهت ساده سازی از چندین تابع جبری استفاده می کند. این تابع قادر به ساده سازی عباراتی که شامل انتگرال، سری، لگاریتم، تابع بسل، توابع نمایی، تابع گاما و ... هستند می باشد. در جدول زیر چند مثال از این تابع را مشاهده می کنید:

| f | simplify(f) |
|---|---|
| x*(x*(x-6)+11)-6 | x^3-6*x^2+11*x-6 |
| (1-x^2)/(1-x) | x+1 |
| (1/a^3+6/a^2+12/a+8)^(1/3) | ((2*a+1)^3/a^3)^(1/3) |
| syms x y positive<br>log(x*y) | log(x)+log(y) |
| exp(x) * exp(y) | exp(x+y) |
| besselj(2,x) + besselj(0,x) | 2/x*besselj(1,x) |
| gamma(x+1)-x*gamma(x) | 0 |
| cos(x)^2 + sin(x)^2 | 1 |

**تابع simple:** هدفی که این تابع برای ساده سازی دنبال می کند جدای از تعریف ریاضی برای ساده سازی یک عبارت می باشد. ساده سازی از دیدگاه این تابع(!) یافتن عبارتی است که تعداد کارکتر های کمتری داشته باشد. البته دلیل ریاضی موجهی برای اینکه ثابت کنیم که یک عبارت از عبارت دیگر ساده تر است،فقط به این دلیل که کارکتر های آن کمتر است وجود ندارد، ولی در عمل معمولا چنین نتیجه ای درست از آب در می آید!

تابع simple کلیه ی توابع ساده سازی از جمله simplify ، collect ، factor و ... را به کار می گیرد تا عبارتی را پیدا کند که کمترین طول را داشته باشد و سپس آن عبارت را به عنوان نتیجه برگشت خواهد داد.

استفاده از تابع simple چند نوع خروجی می تواند تولید کند، مثلا simple(f) کلیه ی توابع ساده سازی که روی f توسط تابعsimple پیاده می شوند را روی خط فرمان MATLAB چاپ می کند و سپس نتیجه را که کوتاه ترین عبارت است چاپ می کند. مثلا اگر فرض کنیم f=cos(x)^2+sin(x)^2 آنگاه simple(f) چنین نتیجه ای را در برخواهد داشت:

```
simplify:
        1
radsimp:
        cos(x)^2+sin(x)^2
combine(trig):
        1
factor:
        cos(x)^2+sin(x)^2
expand:
        cos(x)^2+sin(x)^2
convert(exp):
        (1/2*exp(i*x)+1/2/exp(i*x))^2-1/4*(exp(i*x)-1/exp(i*x))^2
convert(sincos):
        cos(x)^2+sin(x)^2
convert(tan):
        (1-tan(1/2*x)^2)^2/(1+tan(1/2*x)^2)^2+4*tan(1/2*x)^2/
        (1+tan(1/2*x)^2)^2
collect(x):
        cos(x)^2+sin(x)^2

ans =
        1
```

این طریق چاپ نتایج برای زمانی مفید است که شما بخواهید بررسی کنید که آیا عبارت نتیجه واقعا ساده ترین عبارت است یا نه. اگر مایل نیستید که چنین عباراتی در خروجی چاپ شود می توانید تابع simple را بدین صورت فراخوانی کنید:

```
>>f = simple(f)
f =
        1
```

و اما اگر مایلید بدانید که کدام تابع ساده سازی کوتاهترین عبارت را برگردانده، می توانید تابع simple را با دو خروجی فراخوانی کنید:

```
>>[f, how] = simple(f)
f =
        1
how =
        combine
```

در جدول زیر تابع simple با تابع simplify برای دو عبارت مقایسه شده اند:

| f | simplify(f) | simple(f) |
|---|---|---|
| (1/a^3+6/a^2+12/a+8)^(1/3) | ((2*a+1)^3/a^3)^(1/3) | (2*a+1)/a |
| syms x y positive<br>log(x*y) | log(x)+log(y) | log(x*y) |

گاهی اوقات چند بار استفاده از تابع simple باعث می شود که چند تابع ساده سازی مختلف برای ساده کردن عبارت استفاده شوند، به عنوان مثال مقداری که دستور simple(simple((1/a^3+6/a^2+12/a+8)^(1/3))) برمی گرداند 2+1/a می باشد، در تابع simple داخلی از radsimp برای تولید عبارت (2*a+1)/a و در تابع simple بیرونی از تابع combine(trig) برای تبدیل این عبارت به عبارت 2+1/a استفاده شده است.

تابع simple یک تابع موثر برای ساده سازی عبارات مثلثاتی می باشد، در جدول زیر چند مثال در این باره می بینید:

| f | simple(f) |
|---|---|
| cos(x)^2+sin(x)^2 | 1 |
| 2*cos(x)^2-sin(x)^2 | 3*cos(x)^2-1 |
| cos(x)^2-sin(x)^2 | cos(2*x) |
| cos(x)+(-sin(x)^2)^(1/2) | cos(x)+i*sin(x) |
| cos(x)+i*sin(x) | exp(i*x) |
| cos(3*acos(x)) | 4*x^3-3*x |

## ۲-۲-۲) جایگزینی:

دو تابع برای جاگزینی یک عبارت سمبلیک وجود دارد: subexpr و subs .

**تابع subexpr** : دستورات

```
>>syms a x
>>s = solve(x^3+a*x+1)
```

معادله ی x^3+a*x+1 = 0 را نسبت به x حل می کنند:

```
s =
[                    1/6*(-108+12*(12*a^3+81)^(1/2))^(1/3)-2*a/
                          (-108+12*(12*a^3+81)^(1/2))^(1/3)]
[ -1/12*(-108+12*(12*a^3+81)^(1/2))^(1/3)+a/
     (-108+12*(12*a^3+81)^(1/2))^(1/3)+1/2*i*3^(1/2)*(1/
     6*(-108+12*(12*a^3+81)^(1/2))^(1/3)+2*a/
     (-108+12*(12*a^3+81)^(1/2))^(1/3))]
[ -1/12*(-108+12*(12*a^3+81)^(1/2))^(1/3)+a/
     (-108+12*(12*a^3+81)^(1/2))^(1/3)-1/2*i*3^(1/2)*(1/
     6*(-108+12*(12*a^3+81)^(1/2))^(1/3)+2*a/
     (-108+12*(12*a^3+81)^(1/2))^(1/3))]
```

از تابع pretty برای چاپ نتایج در یک قالب قابل خواندن استفاده می کنیم:

```
s =
          [                      1/3       a                    ]
          [                 1/6 %1    - 2 -----                 ]
          [                                1/3                   ]
          [                              %1                      ]
          [                                                      ]
          [        1/3     a           1/2 /      1/3       a  \]
          [- 1/12 %1   + ----- + 1/2 i 3   |1/6 %1    + 2 -----|]
          [                1/3             |                1/3|]
          [              %1                \              %1   /]
          [                                                      ]
          [        1/3     a           1/2 /      1/3       a  \]
          [- 1/12 %1   + ----- - 1/2 i 3   |1/6 %1    + 2 -----|]
          [                1/3             |                1/3|]
          [              %1                \              %1   /]

                                        3       1/2
                  %1 := -108 + 12 (12 a   + 81)
```

در اینجا تابع pretty خاصیتی را که از Maple به ارث برده استفاده می کند. یعنی عباراتی را که چندین بار استفاده می شوند را با عباراتی به شکل %n جایگزین می کند. تابع subexpr چنین عباراتی را که چندین با تکرار می شوند را در یک ماتریس ستونی به نام sigma ذخیره می کند. با مثال زیر ادامه می دهیم:

```
>>r = subexpr(s)
sigma =
      -108+12*(12*a^3+81)^(1/2)

r =
[                                1/6*sigma^(1/3)-2*a/sigma^(1/3)]
[ -1/12*sigma^(1/3)+a/sigma^(1/3)+1/2*i*3^(1/2)*(1/6*sigma^
     (1/3)+2*a/sigma^(1/3))]
[ -1/12*sigma^(1/3)+a/sigma^(1/3)-1/2*i*3^(1/2)*(1/6*sigma^
     (1/3)+2*a/sigma^(1/3))]
```

به خاطر داشته باشید که تابع subexpr متغیری با نام sigma در محیط کار MATLAB ایجاد خواهد کرد.

**تابع sub :** می خواهیم مقدار ویژه و بردار ویژه ی ماتریس مربعی A را پیدا کنیم:

```
>>syms a b c
>>A = [a b c; b c a; c a b];
>>[v,E] = eig(A)
v =
```

```
[    -(a+(b^2-b*a-c*b-c*a+a^2+c^2)^(1/2)-b)/(a-c)    ,
        -(a-(b^2-b*a-c*b-c*a+a^2+c^2)^(1/2)-b)/(a-c),                    1]
[    -(b-c-(b^2-b*a-c*b-c*a+a^2+c^2)^(1/2))/(a-c)    ,
        -(b-c+(b^2-b*a-c*b-c*a+a^2+c^2)^(1/2))/(a-c),                    1]
[ 1,
                        1,                                               1]
E =
[    (b^2-b*a-c*bc*
                a+a^2+c^2)^(1/2),              0,                        0]
[                 0,              -(b^2-b*a-c*bc*
                                  a+a^2+c^2)^(1/2),                      0]
[ 0,                                   0,                          b+c+a]
```

فرض کنید که ما می خواهیم عبارت کوتاهتری را جایگزین عبارت طولانی زیر کنیم:

```
(b^2-b*a-c*b-c*a+a^2+c^2)^(1/2)
```

ابتدا از تابع subexpr استفاده می کنیم:

```
>>v = subexpr(v,'S')
S =
        (b^2-b*a-c*b-c*a+a^2+c^2)^(1/2)
v =
        [   -(a+S-b)/(a-c),                -(a-S-b)/(a-c),                1]
        [    -(b-c-S)/(a-c),               -(b-c+S)/(a-c),                1]
        [                 1,                            1,                1]
```

سپس با جایگذاری S در E به وسیله ی تابع subs ادامه می دهیم:

```
>>E = subs(E,S,'S')
E =
        [   S,                          0,                    0]
        [   0,                         -S,                    0]
        [   0,                          0,                b+c+a]
```

حال فرض کنید که می خواهیم مقدار v را با فرض a=10 حساب کنیم. می توانیم این کار را با استفاده از تابع subs انجام دهیم:

```
>>subs(v,a,10)
```

این دستور تمام متغیر های a در v را با مقدار 10 جایگزین می کند:

```
V=
        [  -(10+S-b)/(10-c),      -(10-S-b)/(10-c),                    1]
        [    -(b-c-S)/(10-c),      -(b-c+S)/(10-c),                    1]
        [                  1,                    1,                    1]
```

ملاحظه می کنید که عبارتی که به وسیله ی جایگذاری S به وجود آمده بسیار ساده تر از عبارت قبلی می باشد. توجه کنید که متغیر a در S با مقدار 10 جایگزین نشده است. تابع subs تابع بسیار مفیدی برای جایگزینی عبارات مختلف که شامل چندین متغیر هستند می باشد. حال می خواهیم اندکی به متغیر S بپردازیم، فرض کنید که علاوه بر جایگزینی مقدار a می خواهیم متغیر های b و c را نیز در S با مقادیر به ترتیب 2 و 10 جایگزین کنیم. برای این کار می توانیم متغیر های a و b و c را در محیط کار MATLAB تعریف و مقدار دهی کنیم. در این صورت تابع subs مقدار عددی آرگومان ورودی را با توجه به متغیر های موجود در محیط کار MATLAB ارزیابی می کند. برای این مثال با دستورات زیر پیش بروید:

```
>>a = 10; b = 2; c = 10;
>>subs(S)
ans =
        8
```

برای مشاهده ی متغیر های موجود در محیط کار MATLAB که حالا سه متغیر جدید نیز به آن ها اضافه شده است از دستور whos استفاده کنید:

```
>>whos
      Name        Size        Bytes        Class
      A           3x3          878         sym object
      E           3x3          888         sym object
      S           1x1         186          sym object
      a           1x1         8            double array
      ans         1x1         140          sym object
      b           1x1         8            double array
      c           1x1         8            double array
      v           3x3         982          sym object
```

حالا دیگر متغیر های a و b و c از نوع double می باشند، اما متغیر های A و E و S از نوع سمبلیک باقی مانده اند. البته اگر بخواهیم که متغیر های a و b و c به صورت سمبلیک باقی بمانند می توانیم خودمان آن ها را به صورت سمبلیک در محیط کار MATLAB به صورت زیر تعریف کنیم:

```
>>syms a b c
>>subs(S,{a,b,c},{10,2,10})      % or you can type: subs(S,[a,b,c],[10,2,10])
ans =
      8
```

برای اطمینان از اینکه متغیر ها از نوع سمبلیک باقی مانده اند می توانید دوباره دستور whos را تایپ کنید.

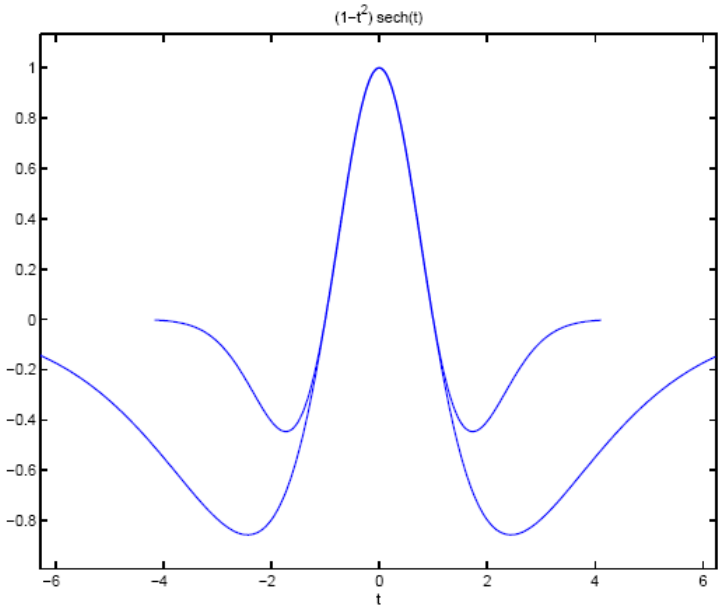
دستور subs را می توان با مقادیر double نیز ترکیب کرد و نتیجه ی عددی عبارت سمبلیک را بدست آورد. فرض کنید که متغیر های زیر را داریم:

```
>>syms t
>>M = (1-t^2)*exp(-1/2*t^2);
>>P = (1-t^2)*sech(t);
```

و می خواهیم بدانیم که متغیر های M و P از لحاظ نموداری چه تفاوتی با هم دارند. یک روش استفاده از دستورات زیر است:

```
>>ezplot(M); hold on; ezplot(P)
```

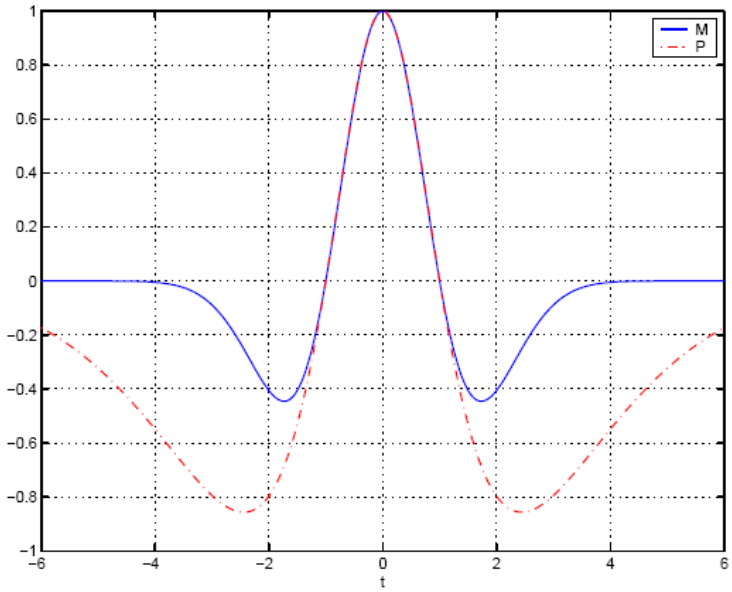
اما نموداری که از این طریق برای ما چاپ می شود کمک زیادی به ما در این زمینه نخواهد کرد، در واقع می توانستیم آن را خواندنی تر هم رسم کنیم!



به جای این دستورات می توان از ترکیب دستور subs با مقادیر double استفاده کرد. در دستورات زیر از جایگزین کردن یک عبارت سمبلیک با یک آرایه double استفاده می شود:

```
>>T = -6:0.05:6;
>>MT = double(subs(M,t,T));
>>PT = double(subs(P,t,T));
>>plot(T,MT,'b',T,PT,'r-.')
>>title(' ')
>>legend('M','P')
>>xlabel('t'); grid
```

نمودار های زیر با دو رنگ مختلف رسم شده اند که این خصوصیت تشخیص آن ها را از یکدیگر آسان تر می کند.




در آخر به خاطر داشته باشید که استفاده از تابع subs با رشته ها حل مسائلی چون سری فوریه، لاپلاس و تبدیلات z را بسیار ساده تر می کند.

# بخش ۳

## محاسبات دقت متغیر

در این جعبه ابزار ۳ نوع مختلف از عملگرهای حسابی وجود دارند:
عددی: محاسبات ممیز شناور MATLAB
گویا: محاسبات سمبلیک دقیق Maple
vpa : محاسبات دقت متغیر Maple
برای مثال دستور MATLAB زیر از نوع محاسبه ی عددی برای انجام عملیات استفاده می کند:

```
>>format long
>>1/2+1/3
        0.83333333333333
```

با استفاده از جعبه ابزار سمبلیک MATLAB دستور زیر از نوع محاسبه سمبلیک استفاده می کند.

```
>>sym(1/2)+1/3
ans=
        5/6
```

و همچنین با استفاده از جعبه ابزار vpa و دستورات زیر از محاسبات دقت متغیر برای انجام عملیات استفاده می کنند:

```
>>digits(25)
>>vpa('1/2+1/3')
ans=
        .8333333333333333333333333
```

عملگر های ممیز شناور که نوع محاسبه عددی از آن ها استفاده می کند، سریعترین نوع محاسبات نسبت به دو نوع دیگر گفته شده است و همچنین به حافظه ی کمتری جهت انجام محاسبات نیاز دارد، ولی نتایج آن خیلی دقیق نیست. تعداد ارقام چاپ شده در خروجی MATLAB برای متغیر های double به وسیله ی دستور format کنترل می شود. اما نوع double همیشه در داخل MATLAB به صورت ممیز شناور ۸ بایتی است که سخت افزار کامپیوتر هم نوع را پشتیبانی می کند.
در محاسبه ی نتیجه ی عددی بالا، در حقیقت ۳ خطای گرد کردن وجود دارد. یکی در تقسیم 1 بر 3، یکی در اضافه کردن 1/2 به نتیجه ی این تقسیم و دیگری در تبدیل عدد ذخیره شده در مبنای ۲ به مبنای ۱۰ جهت نمایش آن می باشد.
در کامپیوتر هایی که از استاندارد حسابی ممیز شناور IEEE استفاده می کنند، مقدار داخلی، نتیجه ی بسط اعشاری 5/6 تا ۵۳ رقم اعشار(۵۳ بیت) می باشد. این مقدار اعشار تقریبا معادل ۱۶ رقم در مبنای ۱۰ می باشد. اما در این مورد خاص خروجی نمایش داده شده فقط ۱۵ رقم اعشار را نمایش می دهد.
عملگر های سمبلیک که در حساب گویا از آن ها استفاده می شود، پر هزینه ترین نوع از ۳ نوع گفته شده می باشند. هم از لحاظ پیچیدگی زمانی و هم از لحاظ پیچیدگی حافظه. نتیجه ی این عملگر ها بسته به زمان و حافظه ای که برای تکمیل محاسبات در دسترس می باشد، دقیق تر هستند.
vpa از میان ۳ نوع گفته شده هم از لحاظ هزینه و هم از لحاظ دقت در میانه قرار دارد. یک متغیر عمومی که به وسیله ی تابع digits مقدارش قابل تنظیم است، تعداد ارقام با معنی برای متغیر های دهدهی را مشخص می کند. افزایش دادن تعداد ارقام مقدار دقت متغیر را افزایش می دهد، اما باعث افزایش حافظه و زمان مورد نیاز نیز می گردد. مقدار پیش فرض ست شده برای تابع digits مقدار 32 می باشد که این مقدار تقریبا برابر است با همان دقت متغیر ممیز شناور.

در اصطلاحات Maple از عبارت «ممیز شناور سخت افزار» برای آنچه که ما «عددی» یا «ممیز شناور» می نامیم استفاده می شود و از عبارت «محاسبات ممیز شناور» برای آنچه که ما «محاسبات دقت متغیر» می نامیم استفاده می شود.

## مثال: استفاده از انواع مختلف محاسبه

**محاسبات حقیقی:** به طور پیش فرض، جعبه ابزار ریاضی سمبلیک از عملگرهای حقیقی استفاده می کند، همانند محاسبات سمبلیک دقیق Maple. زمانی که شما متغیر های سمبلیک را به وسیله ی دستور sym تعریف می کنید، محاسبات حقیقی احضار می شوند.

تابع sym ماتریس های از نوع double را به فرم سمبلیک آن ها تبدیل می کند. برای مثال اگر ماتریس double به صورت

```
A =
1.1000    1.2000    1.3000
2.1000    2.2000    2.3000
3.1000    3.2000    3.3000
```

باشد، شکل سمبلیک آن به صورت زیر خواهد بود:

```
>>s=sym(A)
S =
[11/10,  6/5, 13/10]
[21/10, 11/5, 23/10]
[31/10, 16/5, 33/10]
```

برای این ماتریس A ممکن است مشاهده کنید که درایه های بدست آمده نسبت اعداد صحیح کوچک هستند که این کسر از آن اعداد صحیح تشکیل شده است. اما در سوی دیگر دستور زیر ماتریسی را برگردانده که درایه های آن نسبت دو عدد صحیح کوچک نیست،

```
>>E = [exp(1) sqrt(2); log(3) rand]
E =
        2.71828182845905      1.41421356237310
        1.09861228866811      0.21895918632809
```

بنا بر این دستور sym(E) نمایش ممیز شناور این ماتریس را از اول به صورت نمایش سمبلیک تبدیل می کند:

```
ans=
        [ 3060513257434037*2^(-50),        3184525836262886*2^(-51) ]
        [ 2473854946935174*2^(-51),        3944418039826132*2^(-54) ]
```

## تعداد ارقام در دقت متغیر:

تعداد ارقام در دقت متغیر چیزی متمایز است از نمایش اعداد گویا به وسیله ی ممیز دهدهی. در این بحث نمایش اعداد به شکل *number e± pow* (مانند 123456789 e+23) نیز وجود دارد. برای استفاده از متغیر دقیق به جای گویا می توانید این متغیر ها را به وسیله ی تابع vpa به وجود آورید.

برای ماتریسی که کاملا شامل مقادیر double است، تابع vpa نمایشی از آن را به وجود می آورد که در محاسبات دقت متغیر از آن استفاده می شود. با مثال زیر ادامه دهید و از دستور digits(4) استفاده کنید. دستور vpa را روی ماتریس S اجرا کنید:

```
>>vpa(S)
S =
        [ 1.100,    1.200,    1.300 ]
        [ 2.100,    2.200,    2.300 ]
        [ 3.100,    3.200,    3.300 ]
```

و پس از دستور digits(25) خواهیم داشت:

```
>>F = vpa(E)
F =
        [ 2.7182818284590455348848 08,    1.4142135623730949234300 17 ]
        [ 1.0986122886681100041528 23,    .2189591863280899719512718 ]
```

## تبدیل به ممیز شناور:

برای تبدیل یک متغیر گویا یا یک متغیر دقیق به شکل ممیز شناور آن در MATLAB از دستور double می توان استفاده کرد. در مثال زیر هم دستور double(sym(E)) و هم دستور double(vpa(E)) مقدار E را برمی گردانند.

## یک مثال دیگر:

این مثال شاید جالب تر باشد، با عبارت سمبلیک f = sym('exp(pi*sqrt(163))') شروع کنید، دستور زیر یک نمایش ممیز شناور ایجاد می کند:

```
>>double(f)
ans=
        2.625374126407687e+17
```

از آرگومان دوم دستور vpa برای تعیین تعداد ارقام نمایش داده شده استفاده کنید:

```
>>vpa(f,18)
ans=
        262537412640768744.

>>vpa(f,25)
ans=
        262537412640768744.0000000
```

حال این گمان به وجود می آید که f یک مقدار صحیح دارد. این بدگمانی با دستور vpa(f,30) که نتیجه ی 262537412640768743.999999999999 را دارد تقویت هم می شود. سرانجام دستور vpa(f,40) با نتیجه ی 262537412640768743.9999999999992500725944 نشان می دهد که این عدد با اینکه بسیار به یک عدد صحیح نزدیک است، اما دقیقا یک عدد صحیح نیست.

# فصل ۴

# جبر خطی

## عملگر های جبری پایه

عملگرهای جبری پایه ای که در MATLAB برای کار با اشیای سمبلیک به کار می روند همانند عملگر هایی هستند که برای اشیایی از نوع double استفاده می شوند. در مثال های زیر این موضوع نشان داده شده است.

```
syms t;
G = [cos(t) sin(t); -sin(t) cos(t)]
```

تبدیل داده شده یک چرخش روی صفحه نسبت به زاویه t ایجاد می کند و G به کار رفته در دستور بالا یک ماتریس تبدیل برای این چرخش است:

```
G =
    [ cos(t), sin(t) ]
    [ -sin(t), cos(t) ]
```

با دوبار اعمال این تبدیل، گویا تبدیلی نسبت به زاویه 2*t انجام داده ایم. برای ایجاد ماتریسی که این تبدیل را شبیه سازی کند(تبدیل نسبت به 2*t)کافیست G را در خودش ضرب کرده یا آن را به توان ۲ برسانیم. برای مثال دستوری به صورت A=G*G<<یا A=G^2<< خروجی زیر را تولید می کند:

```
[  cos(t)^2-sin(t)^2, 2*cos(t)*sin(t)]
[ -2*cos(t)*sin(t), cos(t)^2-sin(t)^2]
```

که با استفاده از ساده سازی خروجی به کمک تابع simple، مطلب فوق به خوبی قابل مشاهده است:

```
>>A=simple(A)

A =
    [ cos(2*t), sin(2*t)]
    [-sin(2*t), cos(2*t)]
```

نکته جالبی که در این ماتریس وجود دارد این است که این ماتریس یک ماتریس قائم است و ماتریس وارون و ترانهاده آن یکی هستند. این موضوع را نیز می توان آزمایش کرد:

```
>>I = G.' *G
I =
   [cos(t)^2+sin(t)^2,                    0]
   [                  0, cos(t)^2+sin(t)^2]

>>I = simple(I)
```

```
I =
   [1, 0]
   [0, 1]
```

## عملگر های جبر خطی

در این قسمت سعی می کنیم با اجرای چند دستور با عملگر های جبر خطی آشنا شویم. دستور زیر یک ماتریس هیلبرت ۳در۳ ایجاد می کند:

```
>>H = hilb(3)
H =
    1.0000    0.5000    0.3333
    0.5000    0.3333    0.2500
    0.3333    0.2500    0.2000
```

همان طور که می بینید خروجی در قالب short و به صورت ممیز شناور است و شباهت چندانی به ماتریس های هیلبرتی که ما به صورت دستی می نویسیم ندارد. پس در این مورد نیز بهتر کار را به صورت سمبلیک ادامه بدهیم:

```
>>H = sym(H)
H=
    [    1, 1/2, 1/3]
    [ 1/2, 1/3, 1/4]
    [ 1/3, 1/4, 1/5]
```

حال می توانیم عملیاتی با دقت بینهایت نیز روی این ماتریس انجام دهیم. (توجه کنید که این عملیات روی شکل سمبلیک این ماتریس جواب دقیقی ایجاد خواهند کرد و نه شکل ممیز شناور آن.)

```
>>inv(H)
ans=
       [    9,   -36,    30]
       [ -36,  192, -180]
       [  30, -180,  180]

>>det(H)
ans=
      1/2160
```

می توانیم با استفاده از عمگر \ معادله های خطی را نیز حل کنیم:

```
>>b = [1 1 1]'
>>x = H\b     % Solve Hx = b
x=
         [    3]
         [ -24]
         [  30]
```

همان طور که می بینید هر سه نتیجه فوق (دترمینان، ماتریس معکوس،جواب معادله خطی) جواب های کاملاً دقیقی هستند.

از سوی دیگر با استفاده از digits(16) و تابع vpa به صورت V=vpa(hilb(3)) >> خروجی زیر را خواهیم داشت:

```
[                    1.,  .5000000000000000,  .3333333333333333]
[.5000000000000000,  .3333333333333333,  .2500000000000000]
[.3333333333333333,  .2500000000000000,  .2000000000000000]
```

همان طور که می بینید نتایج تا ۱۶ رقم گرد شده اند.

هنگام معکوس کردن ماتریس این خطاها بر اساس اعضای ماتریس ممکن است بزرگرتر شوند که برای hilb(3) تقریباً برابر ۵۰۰ است. مثال زیر نشان دهنده همین امر است(دو رقم از دست می رود).

```
>>inv(V)
[  9.000000000000082,  -36.00000000000039,   30.00000000000035]
[ -36.00000000000039,   192.0000000000021,  -180.0000000000019]
[  30.00000000000035,  -180.0000000000019,   180.0000000000019]
```

در ادامه، عملیات دترمینان گیری و حل معادله را انجام می دهیم:

```
>> det(V)

ans=

     .462962962962958e-3

>>V\b

ans=

     [ 3.000000000000041]
     [-24.00000000000021]
     [ 30.00000000000019]
```

از آنجایی که H یک ماتریس غیر منفرد است null(H) یک ماتریس خالی و colspace(H) جایگشتی از ماتریس واحد ایجاد می کند:

```
>>null(H)
ans=
       [ empty sym ]
>>colspace(H)
ans=
      [  1,  0,   0]
      [  0,   0,  1]
      [  0,   1,  0]
```

حال برای آنکه مثال فوق را جذاب تر کنیم می خواهیم مقداری برای H(1,1) پیدا کنیم که به ازای آن H یک ماتریس منفرد شود:

```
>> syms s ,H(1,1)=s ,Z=det(H) ,sol=solve(Z)
H =
    [   s,  1/2, 1/3]
    [ 1/2, 1/3, 1/4]
    [ 1/3, 1/4, 1/5]

Z =
    1/240*s-1/270

sol =
      8/9
```

حال کافی است مقدار sol را جایگزین sکنیم:

```
>> H = subs(H,s,sol)
H =
   [ 8/9, 1/2, 1/3]
   [ 1/2, 1/3, 1/4]
   [ 1/3, 1/4, 1/5]
```

برای آزمایش درستی عملیات فوق دترمینان H را محاسبه می کنیم:

```
>>det(H)
ans=
     0
```

در این حالت اگر سعی در محاسبه وارون H کنیم MATLAB نیز پیغام خطایی در مورد منفرد بودن H می دهد.

```
>>inv(H)
        ??? error using ==> inv
        Error, (in inverse) singular matrix
```

به عنوان تمرین null(H) وcolspace(H) را محاسبه کنید. چه نتیجه ای می گیرید؟ البته این موضوع باید ذکر شود که گرچه H یک ماتریس منفرد است اما vpa(H) اینچنین نیست.

## مقدار ویژه

**تابع eig**: از این تابع برای محاسبه مقدار ویژه (eigenvalue) یا بردار ویژه (eigenvector) ماتریس مربعی A استفاده می شود و کاربرد سمبلیک آن به صورت زیر است:

```
E = eig(A)
[V,E] = eig(A)
```

و کاربرد variable-precision آن به صورت زیر است:

```
E = eig(vpa(A))
[V,E] = eig(vpa(A))
```

مقادیر ویژه ماتریس مربعی A، صفر های چند جمله ای مشخصه A، det(A-x*I) هستند.

برای محاسبه چند جمله ای مشخصه از تابع poly استفاده می کنیم:

```
>>poly(A)
```

برای مثال برای ماتریس H بخش قبلی:

```
>>H
H=
   [8/9, 1/2, 1/3]
   [1/2, 1/3, 1/4]
   [1/3, 1/4, 1/5]

>>p=poly(H)
p=
     x^3-64/45*x^2+253/2160*x
```

H یک ماتریس منفرد است پس یکی از مقادیر ویژه آن صفر است.

دستور زیر ماتریس های Tو E را ایجاد می کند که در آن ستون های T بردار های ویژه و عناصر قطراصلی E مقادیر ویژه هستند.

```
>>[T,E]=eig(H)
T =
    [       1,  28/153+2/153*12589^(1/2),     28/153-2/153*12589^(12)]
    [      -4,                         1,                          1]
    [   10/3,   92/255-1/255*12589^(1/2),  292/255+1/255*12589^(12)]

E=
   [ 0,                          0,                          0]
   [ 0,  32/45+1/180*12589^(1/2),                          0]
   [ 0,                          0, 32/45-1/180*12589^(1/2)]

>>solve(p)
ans=
                          0
      32/45+1/180*12589^(1/2)
       32/45-1/180*12589^(1/2)
```

برای درک بهتر ساختار T وE شاید بهتر باشد آن ها را به  شکل دهدهی تبدیل کنیم:

```
>>Td = double(T) , Ed = double(E)

Td =
     1.0000    1.6497   -1.2837
    -4.0000    1.0000    1.0000
     3.3333    0.7051    1.5851

Ed =
    0         0         0
    0    1.3344         0
    0         0    0.0878
```

همان طور انتظار داشتیم یکی از عناصر قطر اصلی E صفر است(H منفرد است). دو مقدار بعدی جواب معادله درجه ۲ $x^2 - 64/45 * x + 253/2160$ هستند که عامل درجه ۲  چند جمله ای  p است.

```
>>syms x
>>g = simple(factor(poly(H))/x)
g=
   x^2-64/45*x+253/2160
```

**نکته:** شکل دقیق سمبلیک مقدار ویژه تنها هنگامی به دست می آید که بتوان چند جمله ای مشخصه آن را به صورت ضرب چند جمله ای های گویایی  با درجه های ۴ یا کمتر بیان کرد.

## ماتریس روزر:

این ماتریس یک ماتریس کلاسیک در آنالیز عددی است که شرط فوق را ارضا می کند.

برای چاپ این ماتریس از sym(rosser()) یا در بعضی از نسخه های قدیمی تر MATLAB از sym(gallery('rosser')) استفاده کنید.

---

**توجه:**

اگر زیاد با ماتریس ها سر و کار دارید حتماً مستندات MATLAB را درباره gallery مطالعه کنید، به وسیله gallery می توانید به مجموعه نسبتاً کاملی از ماتریس های کلاسیک دست پیدا کنید.

---

```
>>R=sym(rosser())
R=
    [ 611,  196, -192,  407,    -8,   -52,  -49,    29]
    [ 196,  899,  113, -192,   -71,   -43,   -8,   -44]
    [-192,  113,  899,  196,    61,    49,    8,    52]
    [ 407, -192,  196,  611,     8,    44,   59,   -23]
    [  -8,  -71,   61,    8,   411,  -599,  208,   208]
    [ -52,  -43,   49,   44,  -599,   411,  208,   208]
    [ -49,   -8,    8,   59,   208,   208,   99,  -911]
    [  29,  -44,   52,  -23,   208,   208,  -911,   99]

>>p = poly(R);
>>pretty(factor(p))
                 2                          2                          2
x (x  -  1020)  (x   -  1020  x  +   100) (x   -  1040500) (x  -  1000)
```

همان طور که می بینید چند جمله ای مشخصه R به خوبی به صورت ضرب دو عامل خطی و سه عامل درجه دو بیان شده است. به سادگی دیده می شود که چهار جواب برای مقدار ویژه ۰، ۱۰۲۰ و دو ریشه درجه ۲ هزار هستند. برای پیدا کردن تمام جواب ها از eig استفاده می کنیم:

```
>>eig(R)
ans=
      [                    0]
      [                 1020]
      [  510+100*26^(1/2)]
      [   510-100*26^(1/2)]
      [     10*10405^(1/2)]
      [    -10*10405^(1/2)]
      [                 1000]
      [                 1000]
```

البته ماتریس روزر یک مثال ساده نیست، به ندرت یک ماتریس کامل ۸ در۸ می تواند چند جمله ای مشخصه ای با چنین عوامل ساده ای داشته باشد. برای مثال اگر دو عنصر گوشه R را از ۲۹ به ۳۰ تغییر دهیم و سعی کنیم چند جمله ای مشخصه را مجدداً محاسبه کنیم نتیجه ای به صورت زیر خواهیم داشت:

```
>>S = R;  S(1,8) = 30;  S(8,1) = 30;
>>p = poly(S)

p=
    40250968213600000+51264008540948000*x-
       1082699388411166000*x^2+4287832912719760*x^-3-
```

```
5327831918568*x^4+82706090*x^5+5079941*x^6-
4040*x^7+x^8
```

همچنین می توانید آزمایش کنید که factor(p) برابر p خواهد بود.

این بار سعی می کنیم تا مقادیر ویژه S را پیدا کنیم:

```
>>F=eig(S)
F=
   [ -1020.05321425589151659318942526 00]
   [ -.17053529728768998575200874607757]
   [  .21803980548301606860857564424981]
   [  999.94691786044276755320289228602]
   [  1000.12069829338413357128170754 54]
   [  1019.52435526320163583249332782 91]
   [  1019.99355012916292573480918081 73]
   [  1020.42018820150472781854574988 40]
```

همان طور که می بینید این مقادیر بسیار نزدیک به مقادیر ویژه ماتریس روزر هستند، در ضمن مقادیر عددی F نتیجه محاسبات ممیز شناور Maple هستند.

نکته دیگری که وجود دارد این است در بسیاری از مواردی که سعی در پیدا کردن مقادیر ویژه سمبلیک یک ماتریس می کنید ممکن است شکل دقیق سمبلیک آن موجود نباشد.

در ادامه با انجام چندین عملیات محاسباتی بر روی یک ماتریس تبدیل این بخش را به پایان می رسانیم.

$$A = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

```
>>syms t
>>A = sym([0 1; -1 0]);
>>G = expm(t*A)

G=
   [ cos(t),  sin(t)]
   [ -sin(t), cos(t)]

>> g = eig(G)
g =
   [ cos(t)+(cos(t)^2-1)^(1/2)]
   [  cos(t)-(cos(t)^2-1)^(1/2)]
```

حال می توانیم با استفاده از simple ، g را ساده کنیم. در زیر به جای چندین بار استفاده از simple از یک حلقه برای انجام این کار استفاده کرده ایم:

```
>>for  j = 1:4
      [g,how] = simple(g)
end

g =
   [ cos(t)+(-sin(t)^2)^(1/2)]
   [  cos(t)-(-sin(t)^2)^(1/2)]

how =
       simplify

g =
   [ cos(t)+i*sin(t)]
```

```
    [  cos(t)-i*sin(t)]

how =
        radsimp

g =
    [ exp(i*t)]
    [ 1/exp(i*t)]

how =
        convert(exp)

g =
    [  exp(i*t)]
    [ exp(-i*t)]

how =
        combine
```

همان طور که می بینید در هر مرحله متغیر how روشی را که simple برای ساده سازی به کار برده نشان می دهد.

## شکل استاندارد جردن

شکل استاندارد جردن نتیجه تلاش برای قطری سازی ماتریس به وسیله تبدیلات مشابهتی است. یعنی می خواهیم برای ماتریس A، ماتریس نا منفرد V را پیدا کنیم به طوری که inv(V)*A*V یا به صورت مختصرترJ=V\A*V تا حد ممکن شکل قطری داشته باشد. تقریباً برای اکثر ماتریس ها، شکل استاندارد جردن به صورت یک ماتریس قطری از مقادیر ویژه است که ستون های ماتریس تبدیل، بردارهای ویژه هستند. و این هنگامی صورت می گیرد که ماتریس متقارن باشد یا دارای مقادیر ویژه مجزا باشد.

**نکته:** بعضی از ماتریس های نامتقارن با مقادیر ویژه مضاعف را نمی توان به شکل قطری تبدیل کرد. در شکل استاندارد جردن عناصر قطر اصلی مقادیر ویژه هستند، در ضمن بعضی از عناصر بالای قطر به جای صفر دارای مقدار یک هستند.

دستور J=jordan(A) شکل استاندارد جردن را محاسبه می کند.

دستور [V , J]=jordan(A) علاوه بر شکل استاندارد جردن، ماتریس تبدیل را نیز محاسبه می کند. ستون های V شکل تعمیم یافته بردار های ویژه هستند.

شکل استاندار جردن شدیداً به پریشانی وانحراف حساس است. تقریباً هر تغییری درA می تواند باعث قطری شدن شکل جردن آن شود، در نتیجه محاسبه شکل جردن در سیستم ممیز شناور چندان قابل اعتماد نخواهد بود. در ضمن این موضوع تاکید می کند که A باید کاملاً شناخته شده باشد (یعنی هیچ یک از اعضای آن حاوی خطای گرد کردن نباشند). در ضمن اعضای A باید اعداد صحیح یا نسبتی از اعداد صحیح کوچک باشند. همچنین استفاده از jordan(vpa(A)) نیز توصیه نمی شود.

```
>>A = sym([12,32,66,116;-25,-76,-164,-294;
          21,66,143,256;-6,-19,-41,-73])

A =
   [  12,   32,    66,   116]
   [ -25,  -76,  -164,  -294]
   [  21,   66,   143,   256]
   [  -6,  -19,   -41,   -73]

>>[V,J] = jordan(A)

V =
   [   4,  -2,   4,  3]
   [  -6,   8, -11, -8]
   [   4,  -7,  10,  7]
   [  -1,   2,  -3, -2]

J =
   [ 1, 1, 0, 0]
   [ 0, 1, 0, 0]
   [ 0, 0, 2, 1]
   [ 0, 0, 0, 2]
```

در نتیجه همان طور که مشاهده می کنید A درای دو مقدار ویژه ۱ و دو مقدار ویژه ۲ است. همچنین این ماتریس تنها دو بردار ویژه دارد: V(:,1) و V(:,3) که روابط زیر را ارضا می کنند:

```
A*V(:,1) = 1*V(:,1)
A*V(:,3) = 2*V(:,3)
```

دو ستون دیگر A نیز تعمیمی از بردار های ویژه هستند:

```
A*V(:,2) = 1*V(:,2) + V(:,1)
A*V(:,4) = 2*V(:,4) + V(:,3)
```

از لحاظ نماد گذاری ریاضی با **vj** = v(:,j) ستون های V و مقادیر ویژه روابط زیر را ارضا می کنند:

$$(A - \lambda_2 I)v_4 = v_3$$

$$(A - \lambda_1 I)v_2 = v_1$$

## تجزیه مقادیر منفرد

در جعبه ابزار ریاضیات سمبلیک فقط از محاسبات عددی با دقت متغیر برای تجزیه بردارهای منفرد کامل می توان استفاده کرد. عدم استفاده از محاسبات سمبلیک به دلیل طولانی بودن نتیجه محاسبات و پیچیدگی آن است. اگر A یک ماتریس سمبلیک از اعداد ممیز شناور یا از اعداد با دقت متغیر باشد دستور زیر مقادیر منفرد A را با دقت تعیین شده توسط digits محاسبه می کند.

```
>>S=svd(A)
```

همچنین دستور [U,S,V] = svd(A) دو ماتریس قائم U وV ماتریس قطری S ایجاد می کند به نحوی که داشته باشیم:

```
A = U*S*V'
```

اجازه دهید توجهمان به ماتریس n درn ای که اعضای آن طبق رابطه زیر تعریف می شوند معطوف کنیم:

```
A(i,j) = 1/(i-j+1/2)
```

که برای n=5 داریم:

```
[   2    -2  -2/3  -2/5  -2/7]
[ 2/3     2    -2  -2/3  -2/5]
[ 2/5   2/3     2    -2  -2/3]
[ 2/7   2/5   2/3     2    -2]
[ 2/9   2/7   2/5   2/3     2]
```

می توان نشان داد که بسیاری از مقادیر منفرد A نزدیک به π هستند.

ساده ترین راه ایجاد A استفاده از ساختار حلقه است:

```
>>for  i=1:n
       for  j=1:n
            A(i,j) = sym(1/(i-j+1/2));
       end
   end
```

راه  حل کار آمد تر برای ایجاد A استفاده از دستورات زیر است:

```
>>[J,I] = meshgrid(1:n);
>>A = sym(1./(I - J+1/2));
```

از آنجایی که اعضای A نسبت هایی از اعداد صحیح کوچک هستند می توان از vpa(A) برای نمایش با دقت متغیر استفاده کرد.در نتیجه از دستور S=svd(vpa(A)) می توان برای محاسبه مقادیر منفرد با دقت مشخص شده استفاده کرد، برای مثال باn=16 و digits(30) نتیجه به صورت زیر خواهد بود:

```
S=

   [  1.2096813760566898533245568535 7 ]
   [  2.6916215868606660677478276359 4 ]
   [  3.0779029723111974865842472735 4 ]
   [  3.1350405439974465484389890126 1 ]
   [  3.1410604466347006380521837192 4 ]
   [  3.1415575435991808369105065826 0 ]
   [  3.1415907545860584872898257711 9 ]
   [  3.1415925692549230647028486310 2 ]
   [  3.1415926505265488081556947961 3 ]
   [  3.1415926534996105314385683856 4 ]
   [  3.1415926535876736171239261238 4 ]
   [  3.1415926535897543920684990722 0 ]
   [  3.1415926535897927034263555905 1 ]
```

```
[ 3.1415926535897932332529014278 1 ]
[ 3.1415926535897932384306684671 2 ]
[ 3.1415926535897932384625503597 4 ]
```

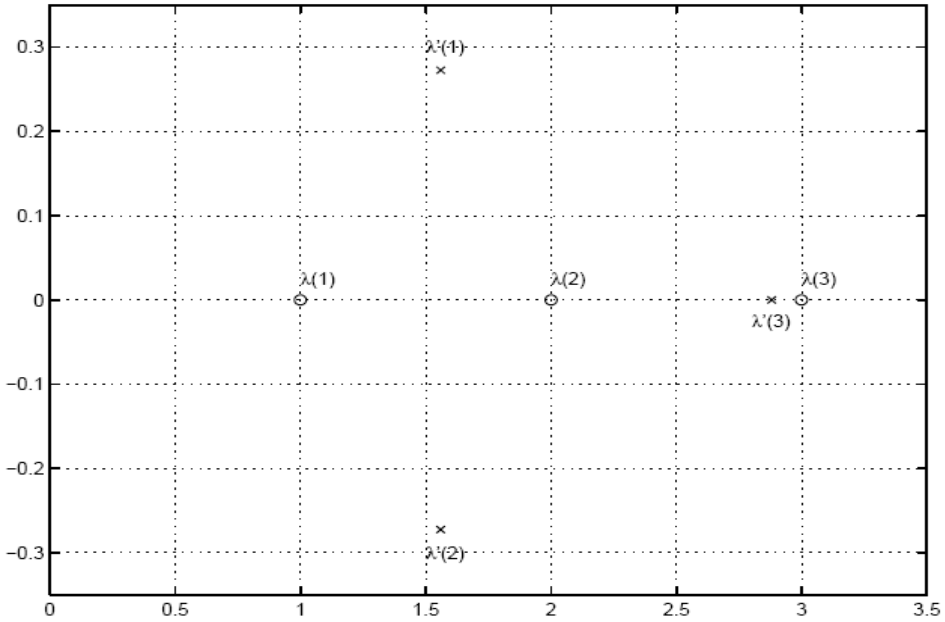
دو راه برای مقایسه Sو pi وجود دارد که در زیر نشان داده شده است:

```
>>format short e
   [double(pi*ones(16,1)-S) pi-double(S)]

ans=

1.9319e+00   1.9319e+00
4.4997e-01   4.4997e-01
6.3690e-02   6.3690e-02
6.5521e-03   6.5521e-03
5.3221e-04   5.3221e-04
3.5110e-05   3.5110e-05
1.8990e-06   1.8990e-06
8.4335e-08   8.4335e-08
3.0632e-09   3.0632e-09
9.0183e-11   9.0183e-11
2.1196e-12   2.1196e-12
3.8846e-14   3.8636e-14
5.3504e-16   4.4409e-16
5.2097e-18            0
3.1975e-20            0
9.3024e-23            0
```

## منحنی های مقادیر ویژه

مثالی که در پیش رو خواهیم داشت شامل تکنیک های عددی، سمبلیک و گرافیکی زیادی است که به مطالعه رفتار مقادیر ویژه می پردازد. تکنیک های به کار رفته کاربرد زیادی در نظریه اعداد، پریش و ... دارند.

در این مثال ماتریس ۳ در۳ A را در نظر می گیریم که دارای مقادیر ویژه ۱،۲و۳ است. سپس A را با ماتریس E با پارامتر $t: A \to A + tE$ به هم میزنیم.هنگامی که $t$ از $0$ تا $10^{-6}$ تغییر می کند مقادیر ویژه $\lambda_1 = 1$، $\lambda_2 = 2$ و $\lambda_3 = 3$ به مقادیر $\lambda_1' \approx 1.5596 + 0.2716\mathrm{i}$, $\lambda_2' \approx 1.5596 - 0.2726\mathrm{i}$ و $\lambda_3' \approx 2.8808$ تبدیل می شوند.




همان طور که می بینید باید مقداری به صورت $t = \tau, 0 < \tau < 10^{-6}$ وجود داشته باشد که به ازای آن ماتریس به هم خورده $A(t) = A + tE$ دارای مقدار ویژه مضاعف $\lambda_1 = \lambda_2$ شود. حال می خواهیم مقدار $\tau$ را پیدا کنیم. ابتدا ماتریس A را ایجاد می کنیم:

```
>> A = gallery(3)

A =
    -149    -50   -154
     537    180    546
     -27     -9    -25
```

این مثالی از یک ماتریس است که مقادیر ویژه آن به خطاهای گرد کردن هنگام محاسبات حساس هستند.

```
>>format long
>> e = eig(A)

e =
   0.99999999999642
   2.00000000000579
   2.99999999999780
```

البته نتیجه به دست آمده از کامپیوتری به کامپیوتری دیگر ممکن است کمی تفاوت داشته باشد.

هر چند مقادیر ویژه واقعی همان ۱،۲و۳ هستند ولی همان طور که می بینید جواب به دست آمده دارای خطاست. البته حل این موضوع در جعبه ابزار ریاضیات سمبلیک ساده است.

```
>>B = sym(A);
    e = eig(B)'
    p = poly(B)
    f = factor(p)

e =
   [1,  2,  3]
```

```
p =
   x^3-6*x^2+11*x-6
f =
   (x-1)*(x-2)*(x-3)
```

آیا مقادیر ویژه به این دلیل که به هم نزدیک هستند به پریشانی حاصل از خطای گرد کردن حساس هستند؟
معمولاً ما از اعدادی مثل ۲،۱و ۳ به عنوان اعداد کاملاً مجزا تعبیر می کنیم، اما در این مورد مجزا بودن یا نزدیک بودن امری است که باید در مقایسه با ماتریس اصلی صورت گیرد. برای مثال اگر در مثال فوق ماتریس A را با A/1000 جایگزین کنیم مقادیر ویژه برابر .001,.002,.003 خواهند بود که از آن ها به عنوان مقادیر نزدیک به هم تعبیر می کنیم.

اما حساسیت مقادیر ویژه بیشتر از "نزدیکی" است.
با یک انتخاب دقیق برای پریشانی ماتریس می توان دو تا از مقادیر ویژه را که شدیداً به خطای گرد کردن حساس هستند را به مقدار واقعیشان نزدیک کرد.
یک جهت پریشانی خوب می تواند از طریق ضرب خارجی بردار های ویژه چپ و راستی که با حساس ترین مقادیر ویژه همراه هستند به دست آید.
دستور زیر ماتریس پریش E را به وجود می آورد:

```
>>E = [130,-390,0;43,-129,0;133,-399,0]
E =
   130  -390   0
    43  -129   0
   133  -399   0
```

حال می توانیم با استفاده از دستور زیر پریشانی را در ماتریس A به وجود بیاوریم:

```
>>syms x t
>>A = A+t*E

A=
   [  -149+130*t,    -50-390*t,   -154]
   [    537+43*t,   180-129*t,    546]
   [   -27+133*t,      -9-399*t,    -25]
```

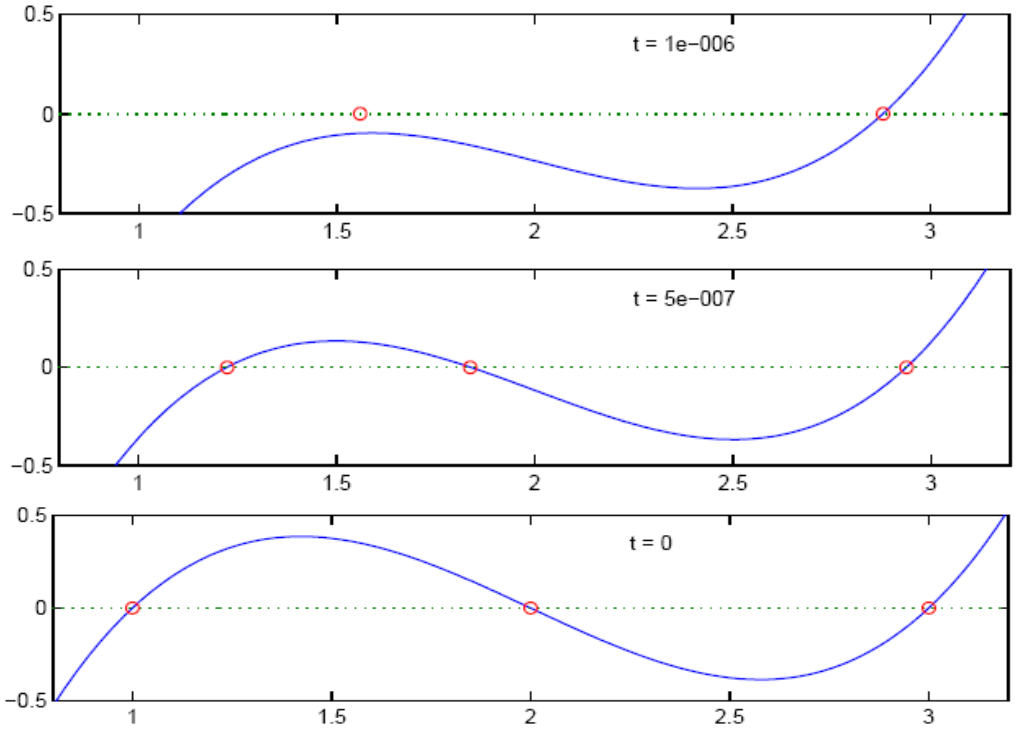
در ادامه چند جمله ای مشخصه A را حساب می کنیم:

```
>>p = poly(A)

p =
    x^3-6*x^2+11*x-t*x^2+492512*t*x-6-1221271*t

>>pretty(collect(p,x))
        3                2
      x   +  (- t - 6) x  + (492512 t + 11) x  -  6  -  1221271  t
```

همان طور که می بینید بین ضرایب ثابت A و t یک ارتباط خطی وجود دارد و هنگامی که t در یک فاصله کوچک تغییر می کند، [0 , 1.0e-6]، می توانیم به ریشه مضاعف مطلوب دست پیدا کنیم، البته این موضوع به صورت گرافیکی بهتر قابل درک است. اشکال زیر نمودار هایی از p به عنوان تابعی از x با مقادیر $t = 0, t = 0.5e-6, t = 1.0e-6$ هستند که به ازای هر مقدار t، مقادیر ویژه نیز به صورت عددی محاسبه شده اند.

```
x = .8:.01:3.2;
```

```
for  k = 0:2
   c = sym2poly(subs(p,t,k*0.5e-6));
   y = polyval(c,x);
   lambda = eig(double(subs(A,t,k*0.5e-6)));
   subplot(3,1,3-k)
   plot(x,y,'-',x,0*x,':',lambda,0*lambda,'o')
   axis([.8 3.2 -.5 .5])
   text(2.25,.35,['t = ' num2str( k*0.5e-6 )]);
end
```




نمودار سوم نشان دهنده چند جمله ناپریشان و ریشه های آن است (۱،۲و۳).

نمودار دوم نشان می دهد که دو ریشه ابتدایی چند جمله ای به یکدیگر نزدیک می شوند.

و نمودار اول نشان می دهد که دو ریشه مختلط می شوند و تنها یک ریشه حقیقی باقی می ماند.

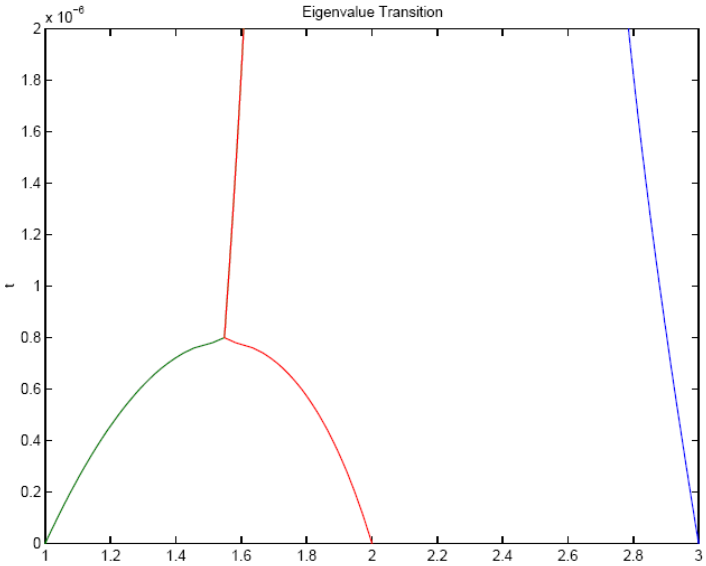
دستور زیر مقادیر ویژه واقعی A را به دست می آورد:

```
>>e = eig(A);
>>pretty(e)
```

```
[                                  1/3                                    ]
[                        1/3 %1     - 3 %2 + 2 + 1/3 t                    ]
[                                                                         ]
[          1/3                                   1/2         1/3          ]
[- 1/6 %1     + 3/2 %2 + 2 + 1/3 t + 1/2 i 3     (1/3 %1     + 3 %2)]
[                                                                         ]
[          1/3                                   1/2         1/3          ]
[- 1/6 %1     + 3/2 %2 + 2 + 1/3 t - 1/2 i 3     (1/3 %1     + 3 %2)]

                                  2     3
%1 := 3189393 t - 2216286 t   + t   + 3 (-3 + 4432572 t
                          2                          3
      - 1052829647418 t + 358392752910068940 t
                        4 1/2
      - 181922388795 t )

                                          2
       - 1/3 + 492508/3 t - 1/9 t
%2 := ---------------------------
                        1/3
                     %1
```

همان طور که می بینید e(2) و e(3) مزدوج یکدیگرند.

```
tvals = (2:-.02:0)' * 1.e-6;
r = size(tvals,1);
c = size(e,1);
lambda = zeros(r,c);
for  k = 1:c
    lambda(:,k) = double(subs(e(k),t,tvals));
end
plot(lambda,tvals)
xlabel('\lambda'); ylabel('t');
title('Eigenvalue Transition')
```




همان طور که در شکل بالا می بینید برای مقادیر t>0.8e-6 نمودار دو تا از مقادیر ویژه برهم منطبق می شوند و برای مقادیرt<0.8e-6 دو ریشه حقیقی به مزدوج مختلط تبدیل می شوند.اما دقیق ترین مقدار t که نشان دهنده این گذر باشد چیست؟

اجازه دهید این مقدار را $\tau$ بنامیم. یک راه برای محاسبه دقیق $\tau$ استفاده از تفکیک کننده چند جمله ای است.همان طور که می دانید تفکیک کننده چند جمله ای درجه دو همان فرمول آشنای دلتا($\Delta$) است. هیچ تابعی به صورت ***discrim*** در جعبه ابزار ریاضیات سمبلیک وجود ندارد اما در Maple چنین تابعی وجود دارد برای اطلات بیشتر در مورد این تابع از دستور زیر استفاده کنید:

```
>>mhelp  discrim
```

شکل استفاده این تابع در MATLAB به صورت زیر است:

```
>>syms  a b c x
>>maple('discrim', a*x^2+b*x+c, x)

ans =
      -4*a*c+b^2
```

برای محاسبه تفکیک کننده چند جمله ای مثال قبل از دستوری به صورت زیر استفاده کنید:

```
>>discrim = maple('discrim',p,x)

discrim =
          4-5910096*t+1403772863224*t^2-477857003880091920*t^3+242563185060*t^4
```

مقدار $\tau$ برابر یکی از چهار ریشه این چند جمله ای خواهد بود.

```
>>digits(24)
>>s = solve(discrim);
>>tau = vpa(s)

tau =
      [                                  1970031.04061804553618913]
      [                                              .783792490602e-6]
      [   .1076924816049e-5+.318896441018863170083895e-5*i]
      [   .1076924816049e-5-.318896441018863170083895e-5*i]
```

از میان این چهار جواب می دانیم که tau(2) نقطه مورد نظر است زیرا به حدسی که قبلاً زدیم نزدیک تر است.

یک راه معمولی ترمحاسبه $\tau$ بر این حقیقت استوار است که در ریشه مضاعف هم تابع و هم مشتقش به صفر میل می کنند. دستور زیر هم p=0 و هم dp/dx=0 را حل می کند:

```
>>sol = solve(p,diff(p,'x'))

sol =
      t :  [4x1 sym]
      x:  [4x1 sym]
```

در ادامه برای محاسبه $\tau$ مراحل زیر را پیروی می کنیم:

```
>>format  short
>>tau = double(sol.t(2))

tau =
      7.8379e-07
```

دستور فوق نشان می دهد که دومین عنصر sol.t جواب مطلوب برای $\tau$ است. همچنین دومین عنصر sol.x مقدار ویژه مضاعف است:

```
>>sigma = double(sol.x(2))
```

```
sigma =
        1.5476
```

اجازه دهید نشان دهیم که این مقدار $\tau$ واقعاً مقدار ویژه مضاعف را در $\sigma = 1.5476$ تولید می کند. برای این کار مقدار $\tau$ را جایگزین t در ماتریس پریشان $A(t) = A+tE$ و مقدار ویژه $A(t)$ را محاسبه می کنیم.

```
>>e = eig(double(subs(A,t,tau)))

e =
    1.5476
    1.5476
    2.9047
```

# بخش ۵

## حل معادلات

### حل معادلات جبری

**تابع solve:** اگر S یک عبارت سمبلیک باشد، آنگاه دستور solve(S) سعی در پیدا کردن مقداری برای متغیر سمبلیک به کار رفته در S(آنطور که دستور findsym تعیین می کند) می کند که به ازای آن S دارای مقدار صفر خواهد شد.

برای مثال:

```
>>syms  a b c x
>>S=a*x^2+b*x+c;
>>solve(S)
```

همان طور که می بینید تابع solve با استفاده از فرمول آشنای حل معادلات درجه دو(دلتا $\Delta$) جواب زیر را برای متغیر x تولید می کند:

```
ans =
        [1/2/a*(-b+(b^2-4*a*c)^(1/2))]
        [ 1/2/a*(-b-(b^2-4*a*c)^(1/2))]
```

جواب نهایی یک بردار(vector) سمبلیک است(در این مثال) که اعضای آن جواب های مسئله هستند.

اگر می خواهید تساوی را برای متغیر خاصی حل کنید، باید آن متغیر را به عنوان آرگومان اضافی مشخص کنید، برای مثال اگر می خواهید S به کار رفته در مسئله قبل را به ازای متغیر b حل کنید باید از دستور زیر استفاده کنید:

```
>>b = solve(S,b)
```

که جواب زیر را برمی گرداند:

```
b =
    -(a*x^2+c)/x
```

توجه کنید که در مثال های قبل ما به حل تساوی هایی از نوع $f(x) = 0$ پرداختیم، اگر می خواهید تساوی هایی از نوع $f(x) = q(x)$ را حل کنید باید از رشته های تک کوتیشنی استفاده کنید.

برای مثال دستور زیر برداری با چهار جواب برمی گرداند:

```
>>s = solve('cos(2*x)+sin(x)=1')
s =
        [          0]
        [         pi]
        [  1/6*pi  ]
        [  5/6*pi  ]
```

همان طور که می بینید در مثال قبل عبارت cos(2*x)+sin(x)=1 را داخل جفت کوتشین تکی قرار دادیم.

### حل چندین معادله جبری

حال می خواهیم توجهمان را کمی به دستگاه تساوی های جبری معطوف کنیم.

فرض کنید عبارت های زیر به ما داده شده است:

$$x^2y^2 = 0$$

$$x - \frac{y}{2}$$

و ما می خواهیم این دستگاه را به ازای x وy حل کنیم.

ابتدا متغیر های سمبلیک ضروری را ایجاد می کنیم:

```
>>syms x y alpha
```

راه های زیادی برای آدرس دهی خروجی تابع solve وجود دارد، یکی از این راه ها فراخوانی به صورت دو-خروجی است:

```
>>[x,y] = solve(x^2*y^2, x-y/2-alpha)
```

که برای مثال فوق مقادیر زیر را برمی گرداند.

```
>>x =
    [        0]
    [        0]
    [  alpha ]
    [  alpha ]

>>y =
    [  -2*alpha ]
    [  -2*alpha ]
    [          0 ]
    [          0 ]
```

در نتیجه به نظر می رسد بردار نتیجه v، v= [x, y] دارای مقادیر زائد باشد. این امر به دلیل تساوی اول ($x^2y^2 = 0$) است زیرا دارای دو جواب برای x و y است: $x = \pm 0, y = \pm 0$.

تغییر تساوی به صورت زیر:

```
>>eqs1 = 'x^2*y^2=1, x-y/2-alpha'
>>[x,y] = solve(eqs1)
```

چهار جواب مجزای زیر را تولید می کند:

```
x =
    [ 1/2*alpha+1/2*(alpha^2+2)^(1/2)]
    [  1/2*alpha-1/2*(alpha^2+2)^(1/2)]
    [  1/2*alpha+1/2*(alpha^2-2)^(1/2)]
    [   1/2*alpha-1/2*(alpha^2-2)^(1/2)]
```

از آنجایی که ما متغیر وابسته خاصی را مشخص نکردیم، تابع solve از findsym برای تعیین متغیر استفاده کرده است.

البته این روش استفاده از تابع solve برای عبارات کوچک مفید است، برای مثال اگر بخواهیم یک عبارت دهتایی ایجاد کنیم تایپ عباراتی به صورت : [x1,x2,x3,x4,x5,x6,x7,x8,x9,x10] = solve(...) هم دشوار و هم زمان بر خواهد بود.

برای فایق آمدن به این مشکل تابع solve می تواند یک ساختار(structure) برگرداند که اعضای آن جواب های مسئله هستند.

فرض کنید عبارات u^2-v^2 = a^2, u + v = 1, a^2-2*a = 3 را داریم آنگاه دستور زیر:

```
>>S = solve('u^2-v^2 = a^2','u + v = 1','a^2-2*a = 3')
```

ساختار زیر را بر می گرداند:

```
>>S =
    a: [2x1 sym]
    u: [2x1 sym]
    v: [2x1 sym]
```

برای مشاهده هر یک از اعضای ساختار S کافی است از" نام ساختار" به همراه ترکیب " نام عضو." استفاده کنیم.

برای مثال:

```
>>S.a
ans =
      [ -1 ]
      [  3 ]
```

حال می توان با استفاده از اندیس ها و نام اعضای ساختار S به قسمت های خاصی از جواب دست یافت.

برای مثال اگر بخواهیم دومین جواب را مشاهده کنیم از عبارتی به صورت s2 = [S.a(2), S.u(2), S.v(2)] استفاده می کنیم، که جواب زیر را بر می گرداند:

```
>>s2 =
    [  3,   5,   -4]
```

توجه کنید که عبارتی به صورت M = [S.a, S.u, S.v] ، ماتریس پاسخ M را تولید می کند که سطرهای آن جواب های مسئله هستند.

```
>>M =
    [  -1,   1,   0]
    [   3,   5,  -4]
```

توجه کنید که دستگاه های خطی تساوی های هم زمان را می توان با استفاده از تقسیم ماتریس ها نیز حل کرد. برای مثال:

```
>>clear u v x y
>>syms u v x y
>>S = solve(x+2*y-u, 4*x+5*y-v);
>>sol = [S.x;S.y]
```

و

```
>>A = [1 2; 4 5];
>>b = [u; v];
>>z = A\b
```

خروجی های زیر را تولید می کنند:

```
>>sol =
      [ -5/3*u+2/3*v]
      [   4/3*u-1/3*v]

>>z =
     [ -5/3*u+2/3*v]
     [   4/3*u-1/3*v]
```

همان طور که می بینید S و z راه حل های یکسانی را ارائه می کنند.

## معادلات دیفرانسیل

**تابع dsolve**: از تابع dsolve برای حل سمبلیک معادلات دیفرانسیل معمولی استفاده می شود. معادلات سمبلیکی که برای نمایش معادلات دیفرانسیل به کار می روند حاوی کاراکتر D خواهند بود که نشان دهنده مشتق است. توجه کنید که برای مشتقات مرتبه ۲و۳و...n ازنمادهای D2 وD3و....DN استفاده می کنیم.

بنا برین عبارتی به صورت D2y نشان دهنده $d^2y/dt^2$ خواهد بود که در آن y متغیر وابسته و t متغیر مستقل خواهد بود.

ذکر این نکته لازم است که در هنگام نام گذاری متغیر، نام متغیر نباید شامل کاراکتر D باشد. در ضمن می توان متغیر مستقل t را به هر نام دلخواه دیگری تغییر داد که در این صورت نام این متغیر باید به عنوان آخرین آرگومان به dsolve فرستاده شود.

توجه کنید که برای تعیین شرایط مرزی می توانید آن ها را به صورت معادلات اضافی به dsolve ارسال کنید. اگر شرایط مرزی را تعیین نکنید جواب معادله شامل مقادیر ثابت C2,C1 و ... خواهد بود.

**نکته:** شیوه ایجاد خروجی تابع dsolve همانند تابع solve است یعنی شما می توانید خروجی را به صورت متغیر هایی به تعداد متغیر های وابسته یا به صورت ساختار(structure) ذخیره کنید.

در ادامه با ذکر چند مثال بیشتر با این تابع آشنا می شویم.

**مثال ۱(معادلات خطی):**

```
>>dsolve('Dy=1+y^2')
```

در این مثال y به عنوان متغیر وابسته و t به عنوان متغیر مستقل پیش فرض به کار می رود.

خروجی به صورت زیر خواهد بود:

```
ans =
      tan(t+C1)
```

توجه کنید که در اینجا یک تفاوت اساسی را با تابع solve مشاهده می کنیم و آن اینکه در اینجا نیاز نیست که متغیر y را قبلاً به کمک syms تعریف کنیم زیرا اسامی ای که بعد از D به کار می روند به عنوان متغیر وابسته شناخته می شوند.

**نکته :** هر چند استفاده از این روش منجر به خطا نخواهد شد ولی توجه کنید که متغیر y(هم چنین t) نیز وارد محیط کاری MATLAB نخواهد شد و دستوری به صورت y>> تولید خطا خواهد کرد.

برای مشخص کردن شرایط مرزی باید آنها را به عنوان آرگومان اضافی به dsolve ارسال کنید:

```
>>y = dsolve('Dy=1+y^2','y(0)=1')
y =
    tan(t+1/4*pi)
```

**نکته :** یکی از توابع پر کاربرد در این زمینه تابع diff که برای مشتق گیری از آن استفاده می شود.هر چند در این کتاب فرض شده است که شما با این توابع آشنایی دارید(در غیر این صورت حتماً به کتاب های مقدماتی و هم چنین help نرم افزار مراجعه کنید). اما مثال زیر جهت آشنایی بیشتر با این تابع و هم چنین کاربرد آن به صورت سمبلیک ذکر شده است.

```
>>syms t
>>diff(y,t)
ans=
      1+tan(t+1/4*pi)^2
```

**مثال ۲(معادلات غیر خطی ):**

همان طور که می دانید معادلات غیرخطی حتی هنگامی که شرایط مرزی مشخص است ممکن است دارای چندین جواب باشند،که مثال زیر این موضوع را نشان می دهد:

```
>>x = dsolve('(Dx)^2+x^2=1','x(0)=0')
x =
    [-sin(t)]
    [ sin(t)]
```

**مثال ۳(معادلات با چندین شرایط مرزی):**

```
>>y = dsolve('D2y=cos(2*x)-y','y(0)=1','Dy(0)=0', 'x');
>>simplify(y)
y =
   -2/3*cos(x)^2+1/3+4/3*cos(x)
```

و یا برای حل معادله ای به صورت $\frac{d^3u}{dx^3} = u$ با شرایط مرزی $u(0) = 1, u'(0) = -1, u''(0) = \pi$ به راحتی از دستوری به صورت زیر استفاده می کنیم:

```
>>u = dsolve('D3u=u','u(0)=1','Du(0)=-1','D2u(0) = pi','x')
```

**حل چندین معادله دیفرانسیل:** از تابع dsolve می توان برای حل چندین معادله همراه با چندین متغیر نیز استفاده کرد. برای مثال:

```
>>S = dsolve('Df = 3*f+4*g', 'Dg = -4*f+3*g')
```

جواب نهایی یک ساختار خواهد بود و برای دسترسی به جواب ها کافی است از دستورات S.f یا S.g استفاده کنید.

```
>>f = S.f
f =
   exp(3*t)*(cos(4*t)*C1+sin(4*t)*C2)
>>g = S.g
g =
    exp(3*t)*(-sin(4*t)*C1+cos(4*t)*C2)
```

البته در این حالت نیز می توان شرایط مرزی را مشخص کرد:

```
>>[f,g] = dsolve('Df=3*f+4*g, Dg =-4*f+3*g', 'f(0) = 0, g(0) = 1')
f =
   exp(3*t)*sin(4*t)
g =
    exp(3*t)*cos(4*t)
```

جدول زیر شامل چندین مثال به همراه شکل استفاده دستورات در MATLAB می باشد.

توجه کنید که آخرین عنصر جدول یک معادله دیفرانسیل Airy می باشد.

| Differential Equation | MATLAB Command |
|---|---|
| $\frac{dy}{dt} + 4y(t) = e^{-t}$<br>$y(0) = 1$ | `y = dsolve('Dy+4*y = exp(-t)', 'y(0) = 1')` |
| $\frac{d^2y}{dx^2} + 4y(x) = e^{-2x}$<br>$y(0) = 0, y(\pi) = 0$ | `y = dsolve('D2y+4*y = exp(-2*x)', 'y(0)=0', 'y(pi) = 0', 'x')` |
| $\frac{d^2y}{dx^2} = xy(x)$<br>$y(0) = 0, y(3) = \frac{1}{\pi}K_{\frac{1}{3}}(2\sqrt{3})$<br>(The Airy equation) | `y = dsolve('D2y = x*y','y(0) = 0', 'y(3) = besselk(1/3, 2*sqrt(3))/pi', 'x')` |

توابع Airy نقش مهمی در ریاضیات به خصوص در شبیه سازی ریاضی انتشار امواج آب بازی می کنند.

به عنوان تمرین نشان دهید تبدیل فوریه تابع Airy به صورت $\exp\left(\frac{iw^3}{3}\right)$ است.

## بخش ۶

# توابع ریاضی خاص

در جعبه ابزار Maple بیش از ۵۰ تابع در زمینه ی ریاضیات کاربردی وجود دارد. این توابع به وسیله ی تابع mfun قابل دسترسی می باشند. این تابع مقدار عددی یک تابع را طبق آرگومان های ورودی آن ارزیابی می کند. همچنین تابع mfun امکان دسترسی به توابعی را که در استاندارد MATLAB وجود ندارند فراهم می کند، مانند تابع Fresnel cosine integral برای مثال فرض کنید که می خواهید مقدار تابع انتگرال cos هیپربولیک را روی نقاط 2+i و 0 و 4.5 حساب کنید. ابتدا برای مشاهده ی لیست توابع قابل دسترسی به وسیله ی تابع mfun با دستور زیر شروع کنید:

```
>>help mfunlist
```

این دستور لیست خلاصه ای از تعریف ریاضی توابع قابل دسترسی به وسیله ی تابع mfun را به همراه نام آن ها در Maple و آرگومان های ورودی آن ها فراهم می کند. با مشاهده ی لیست ملاحظه می کنید که تابع مورد نظر chi نام دارد و یک آرگومان ورودی از نوع عدد مختلط می پذیرد. برای مشاهده ی اطلاعات بیشتر می توانید از طریق دستور mhelp به راهنمای Maple دسترسی پیدا کرده و در مورد تابع مورد نظر اطلاعات کسب کنید، مانند دستور زیر:

```
>>mhelp Chi
```

حال با دستورات زیر ادامه دهید:

```
>>z = [2+i 0 4.5];
>>w = mfun('Chi',z)
  w =
        2.0303 + 1.7227i NaN 13.9658
```

توابعی که به وسیله ی mfun قابل دسترسی می باشند را همراه با توضیحات آن ها در لیست زیر که به وسیله ی راهنمای MATLAB چاپ شده اند را ملاحظه می کنید:

```
>>help mfunlist
 MFUNLIST Special functions for MFUN.
   The following special functions are listed in alphabetical order
   according to the third column. n denotes an integer argument ,
   x denotes a real argument, and z denotes a complex argument. For
   more detailed descriptions of the functions, including any
   argument restrictions, see the Reference Manual, or use MHELP.

 bernoulli      n      Bernoulli Numbers
 bernoulli      n,z    Bernoulli Polynomials
 BesselI        x1,x   Bessel Function of the First Kind
 BesselJ        x1,x   Bessel Function of the First Kind
 BesselK        x1,x   Bessel Function of the Second Kind
 BesselY        x1,x   Bessel Function of the Second Kind
 Beta           z1,z2  Beta Function
 binomial       x1,x2  Binomial Coefficients
 EllipticF -    z,k    Incomplete Elliptic Integral, First Kind
 EllipticK -    k      Complete Elliptic Integral, First Kind
 EllipticCK -   k      Complementary Complete Integral, First Kind
 EllipticE -    k      Complete Elliptic Integrals, Second Kind
 EllipticE -    z,k    Incomplete Elliptic Integrals, Second Kind
 EllipticCE -   k      Complementary Complete Elliptic Integral, Second Kind
 EllipticPi -   nu,k   Complete Elliptic Integrals, Third Kind
```

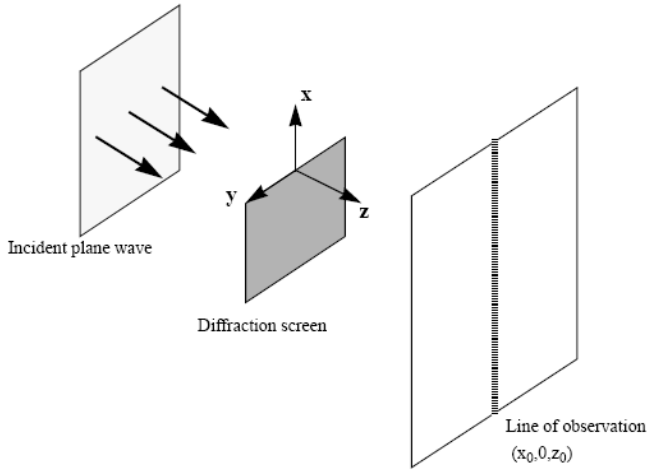
```
EllipticPi -     z,nu,k  Incomplete Elliptic Integrals, Third Kind
EllipticCPi -    nu,k    Complementary Complete Elliptic Integral, Third Kind
erfc             z       Complementary Error Function
erfc             n,z     Complementary Error Function's Iterated Integrals
Ci               z       Cosine Integral
dawson           x       Dawson's Integral
Psi              z       Digamma Function
dilog            x       Dilogarithm Integral
erf              z       Error Function
euler            n       Euler Numbers
euler            n,z     Euler Polynomials
Ei               x       Exponential Integral
Ei               n,z     Exponential Integral
FresnelC         x       Fresnel Cosine Integral
FresnelS         x       Fresnel Sine Integral
GAMMA                    z       Gamma Function
harmonic         n       Harmonic Function
Chi              z       Hyperbolic Cosine Integral
Shi              z       Hyperbolic Sine Integral
GAMMA                    z1,z2   Incomplete Gamma Function
W                z       Lambert's W Function
W                n,z     Lambert's W Function
lnGAMMA                  z       Logarithm of the Gamma function
Li               x       Logarithmic Integral
Psi              n,z     Polygamma Function
Ssi              z       Shifted Sine Integral
Si               z       Sine Integral
Zeta             z       (Riemann) Zeta Function
Zeta             n,z     (Riemann) Zeta Function
Zeta             n,z,x   (Riemann) Zeta Function

         Orthogonal Polynomials (Extended Symbolic Math Toolbox only)
T        n,x             Chebyshev of the First Kind
U        n,x             Chebyshev of the Second Kind
G        n,x1,x          Gegenbauer
H        n,x             Hermite
P        n,x1,x2,x       Jacobi
L        n,x             Laguerre
L        n,x1,x          Generalized Laguerre
P        n,x             Legendre

   See also mfun, mhelp.

   Reference page in Help browser
      doc mfunlist
```

## یک مثال از مسئله ی انعکاس[1]:

این مثال، یک مثال از نظریه ی انعکاس در علم الکترودینامیک است[2]. فرض کنید ما یک موج صفحه ای با شدت $I_0$ و عدد موج $k$ داریم. فرض کنید که این موج صفحه ای موازی صفحه ی $xy$ می باشد و همانطور که در شکل زیر نشان داده شده است، این موج روی محور $z$ ها در حرکت است. این موج صفحه ای را موج ضمنی می نامیم. صفحه ی انعکاسی نصف صفحه

[1] Diffraction
[2] J.D.Jackson, *Classical Electrodynamics*, John Wiley & Sons, 1962

ی $xy$ که در آن $x < 0$ است را اشغال می کند. موج صفحه ای مورد نظر با این صفحه ی انعکاسی برخورد می کند و ما موج منعکس شده را روی خطی با مختصات $(x, 0, z_0)$ مشاهده می کنیم که در آن $z_0 > 0$ می باشد.




شدت موج منعکس شده از رابطه ی زیر بدست می آید:

$$I = \frac{I_0}{2}\left[\left(C(\zeta)+\frac{1}{2}\right)^2+\left(S(\zeta)+\frac{1}{2}\right)^2\right]$$

و همچنین داریم:

$$\zeta = \sqrt{\frac{k}{2z_0}}\cdot x$$

$$C(\zeta) = \int_0^{\zeta}\cos\left(\frac{\pi}{2}-t^2\right)dt$$

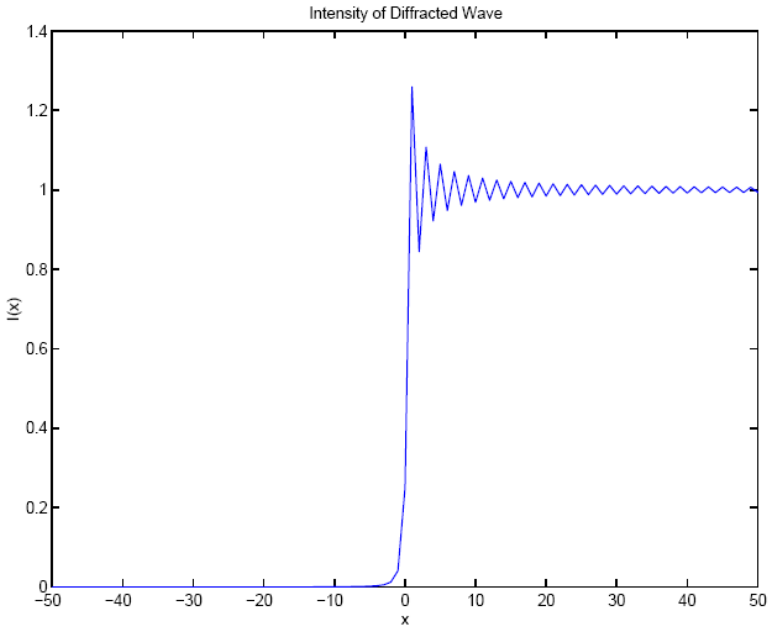
$$S(\zeta) = \int_0^{\zeta}\sin\left(\frac{\pi}{2}-t^2\right)dt$$

حال به این سوال پاسخ می دهیم که شدت موج منعکس شده روی خط مشاهده شده چقدر است؟ از آنجایی که $k$ و $z_0$ ثابت های مستقل از $x$ هستند قرار می دهیم:

$$\sqrt{\frac{k}{2z_0}} = 1$$

و برای راحتی کار فرض می کنیم که شدت اولیه ی $I_0 = 1$ است. کد زیر نمودار تابع شدت موج را به عنوان تابعی از $x$ رسم می کند:

```
>>x = -50:50;
>>C = mfun('FresnelC',x);
>>S = mfun('FresnelS',x);
>>I0 = 1;
>>T = (C+1/2).^2 + (S+1/2).^2;
>>I = (I0/2)*T;
>>plot(x,I);
>>xlabel('x');
>>ylabel('I(x)');
>>title('Intensity of Diffracted Wave');
```




از روی نمودار می توان نتیجه گرفت که همانطور که انتظار می رفت اثر انعکاس نزدیک لبه ی صفحه ی انعکاسی ($x = 0$) بیشتر است.

به خاطر داشته باشید که مقادیر بزرگ و مثبت $x$ با مشاهده ی نقاط دور از صفحه انعکاسی ارتباط دارد. در اینجا ما انتظار داریم که صفحه ی انعکاسی اثری روی موج صفحه ای نداشته باشد، یعنی شدت موج منعکس شده باید به اندازه ی شدت همان موج صفحه ای باشد. به طور مشابه چیزی که گفته شد مقادیر بزرگ و منفی $x$ با مشاهده ی نقاط زیر صفحه ی انعکاسی و دور از لبه ی آن ارتباط دارد. البته انتظار ما در این ناحیه این است که موج منعکس شده شدتی معادل صفر داشته باشد. این نتیجه گیری می تواند به وسیله ی قرار دادن $x = [inf\ , -inf]$ در کد نوشته شده و بدست آوردن $\mathbf{I}$ بررسی شود.

# بخش ۷

## استفاده از توابع Maple

تابع maple به شما اجازه می دهد که به توابع Maple دسترسی مستقیم داشته باشید. این تابع آرگومان هایی از نوع sym و double و string را به عنوان ورودی می پذیرد و بسته به نوع ورودی آرگومان هایی از همین نوع را برگشت خواهد داد. همچنین شما می توانید از تابع maple برای خطایابی برنامه هایی که در مورد ریاضیات سمبلیک طراحی کرده اید بهره ببرید.

**۲-۷-۱) یک مثال ساده:** فرض کنید که ما می خواهیم یک تابع M-file بنویسیم که دو آرگومان ورودی از نوع چند جمله ای یا عدد صحیح می گیرد و بزرگترین مقسوم علیه مشترک آن ها را بر می گرداند. برای مثال بزرگترین مقسوم علیه مشترک دو عدد ۱۴ و ۲۱ برابر است با ۷. یا بزرگترین مقسوم علیه دو چند جمله ای $x^2 - y^2$ و $x^3 - y^3$ برابر است با $x - y$.

اولین چیزی که باید بدانیم این است که چگونه تابع بزرگترین مقسوم علیه مشترک در Maple را از درون MATLAB فراخوانی کنیم. از تابع mhelp برای گرفتن راهنمایی از Maple در مورد تابع بزرگترین مقسوم علیه مشترک (gcd) استفاده می کنیم:

```
>>mhelp gcd
gcd - greatest common divisor of polynomials

lcm - least common multiple of polynomials

Calling Sequence:
        gcd(a,b,'cofa','cofb')
        lcm(a,b,...)

Parameters:
        a, b - multivariate polynomials over an algebraic number
        field or an algebraic function field
        cofa,cofb - (optional) unevaluated names

Description:
- The gcd function computes the greatest common divisor of two
polynomials a and b.

- If the coefficients of a and b are integers, then a primitive
unit normal greatest common divisor is returned. In other words,
the coefficients of the result are relatively prime integers and
the leading coefficient is a positive integer.

- If the coefficients of a or b are rational numbers or belong to
an algebraic number or function field, then the monic greatest
common divisor of a and b is computed. See type,algnum and
type,algfun.

- Algebraic numbers and functions may be represented by radicals
```

(see type,radical) or with the RootOf notation. See evala.

- Names occurring inside a RootOf or a radical are viewed as elements of the coefficient field, provided the RootOf defines an algebraic function. Therefore, they may occur in denominators as well. Other names are not allowed in denominators.

- If a or b contains objects that are not algebraic numbers or algebraic functions, these objects will be frozen before the computation proceeds. See frontend.

- The RootOf and the radicals defining the algebraic numbers must form an independent set of algebraic quantities, otherwise an error is returned. Note that this condition needs not be satisfied if the expression contains only algebraic numbers in radical notation (i.e. 2^(1/2), 3^(1/2), 6^(1/2)). A basis over Q for the radicals can be computed by Maple in this case.

- Since the ordering of the variables depends on the session, the result may also depend on the session when a and b have several variables.

- The lcm function computes the least common multiple of an arbitrary number of polynomials.

- The optional third argument cofa is assigned the cofactor a/gcd(a,b).

- The optional fourth argument cofb is assigned the cofactor b/gcd(a,b).

.
.
.

از آنجایی که ما حالا دیگر املای درست صدا زدن توابع Maple را از درون MATLAB یاد گرفتیم می توانیم یک M-file ساده جهت بدست آوردن بزرگترین مقسوم علیه مشترک بنویسیم. یک M-file جدید در یک دایرکتوری دلخواه (مثلا @sym ) به وجود بیاورید و کد زیر را در آن بنویسید:

```
function g = gcd(a, b)
g = maple('gcd',a, b);
```

حال اگر ما این M-file را اجرا کنیم خواهیم داشت:

```
>>syms x y
>>z = gcd(x^2-y^2,x^3-y^3)
>>w = gcd(6, 24)
z =
        -y+x
w =
        6
```

حال بیایید تابعی را که نوشته ایم طوری بسط دهیم که بتوانیم بزرگترین مقسوم علیه مشترک را در مورد عناصر متناظر در دو ماتریس را هم اجرا کنیم. پس کد M-file مورد نظر را بدین صورت می نویسیم:

```
function g = gcd(a,b)

if any(size(a) ~= size(b))
        error('Inputs must have the same size.')
```

```
end

for k = 1: prod(size(a))
        g(k) = maple('gcd',a(k), b(k));
end

g = reshape(g,size(a));
```

نتیجه ی اجرای تابعی که هم اکنون نوشتیم را در اینجا ملاحظه می کنید:

```
>>A = sym([2 4 6; 3 5 6; 3 6 4]);
>>B = sym([40 30 8; 17 60 20; 6 3 20]);
>>gcd(A,B)
ans =
        [  2,  2,  2  ]
        [  1,  5,  2  ]
        [  3,  3,  4  ]
```

**۲-۷-۲) یک مثال از بردارها:** فرض کنید که می خواهیم سینوس یک ماتریس سمبلیک را محاسبه کنیم. یک راه برای انجام این کار بدین صورت می باشد که یک M-file به صورت زیر بنویسیم:

```
function y = sin1(x)
for k = 1: prod(size(x))
        y(k) = maple('sin',x(k));
end
y = reshape(y,size(x));
```

بنا بر این با نوشتن دستورات زیر خواهیم داشت:

```
>>syms x y
>>A = [0 x; y pi/4]
>>sin1(A)
A =
        [        0,      x     ]
        [        y,      pi/4  ]
ans =
        [        0,          sin(x)  ]
        [    sin(y),     1/2*2^(1/2)  ]
```

راه کارآمد تری که می توانست ما را به هدفمان برساند استفاده از تابع map است که از توابع Maple می باشد. این تابع یک تابع Maple را روی تمامی عناصر یک آرایه اجرا می کند. در همین مثال محاسبه ی سینوس می توانستیم تابع M-file را به صورت زیر نیز بنویسیم.

```
function y = sin2(x)

if prod(size(x)) == 1              % scalar case
        y = maple('sin',x);
else                               % array case
        y = maple('map','sin',x);
end
```

توجه کنید که این تابع با اعداد و آرایه ها دو رفتار متفاوت دارد و تابع map را فقط روی آرایه (و نه روی اسکالر) اعمال می کند. البته این طرز نوشتن باعث می شود کمتر تابع map فراخوانی شود و در نتیجه سرعت عملیات بیشتر می شود.

تابع sin2 به دلیل اینکه فقط یک بار Maple را فراخوانی می کند به طور قابل ملاحظه ای سریع تر از تابع sin1 ( که به اندازه ی prod(size(A)) مرتبه Maple را فراخوانی می کند ) عمل می کند.

تابع map می تواند برای توابعی از Maple که چندین آرگومان ورودی می گیرند نیز به کار رود. الگوی فراخوانی این تابع در این گونه موارد به صورت زیر می باشد:

```
maple('map', Maple function, sym array, arg2, arg3, ..., argn)
```

برای مثال یک راه برای فراخوانی تابع collect که به صورت M-file می باشد تایپ دستور collect(S,x) است. همانطور که قبلا هم دیدیم این دستور ضرایب توان های یکسان از متغیر $x$ در $S$ را یکجا جمع آوری می کند و در واقع $S$ را به یک چند جمله ای استاندارد تبدیل می کند. در اینجا دستور مربوط به فراخوانی تابع collect جهت پیاده سازی در یک تابع M-file نوشته شده است:

```
r = maple('map','collect',sym(s),sym(x));
```

برای مشاهده ی اطلاعات بیشتر در مورد تابع map می توانید از دستور mhelp به صورت زیر استفاده کنید:

```
>>mhelp map
```

**۲-۷-۳) خطایابی:** دستور maple امکاناتی را جهت خطایابی ارائه می کند که این امکانات عبارتند از: ردیابی کردن[1] و بررسی وضعیت آرگومان خروجی[2].

**ردیابی کردن:** دستور maple traceon باعث می شود تمام دستورات بعدی Maple و نتایج آن ها نمایش داده شود. برای مثال به دستورات زیر توجه کنید:

```
>>maple traceon
>>a = sym('a');
>>exp(2*a)
```

که نتیجه ی این دستورات نمایش تمامی فراخوانی های هسته ی Maple و به صورت زیر می باشد:

```
statement:
        (2)*(a);
result:
        2*a
statement:
        2*a;
result:
        2*a
statement:
        exp(2*a);
result:
        exp(2*a)
statement:
        exp(2*a);
result:
        exp(2*a)

ans =
```

[1] Trace Mode
[2] Status output argument.

```
exp(2*a)
```

برای غیر فعال کردن چاپ این نتایج دستور traceoff را اجرا کنید.

**بررسی وضعیت آرگومان خروجی:** تابع maple بسته به درخواست ما می تواند دو خروجی داشته باشد: result و status . اگر عملیات صدا زدن تابع maple موفقیت آمیز انجام شود، نتیجه ی عملیات در متغیر result و عدد صفر در متغیر status قرار خواهد گرفت، اما اگر در این عملیات خطایی صورت گیرد عنوان خطا در متغیر result و کد آن در متغیر status قرار خواهد گرفت. به عنوان مثال تابع discrim از Maple تفکیک کننده ی[1] یک عبارت چند جمله ای (یا به عبارتی دیگر همان دلتا) را محاسبه می کند و الگوی نوشتن آن به صورت discrim(p,x) می باشد که در آن p یک چند جمله ای از x است. فرض کنید که ما در فراخوانی این تابع ارسال آرگومان دوم به آن را فراموش کنیم، در این صورت:

```
>>syms a b c x
>>[result, status] = maple('discrim', a*x^2+b*x+c)
result =
        Error, (in discrim) invalid arguments
status =
        2
```

و اما با تصحیح اشتباه خود خواهیم داشت:

```
>>[result, status] = maple('discrim', a*x^2+b*x+c, x)
result =
        -4*a*c+b^2
status =
        0
```

[1] discriminant

# بخش ۸

## جعبه ابزار سمبلیک تعمیم داده شده[1]

جعبه ابزار سمبلیک تعمیم داده شده این امکان را برای شما فراهم می کند که به تمام package های غیر گرافیکی Maple ، زبان برنامه نویسی Maple و زیر برنامه های آماده ی آن دسترسی داشته باشیم. بنا بر این جعبه ابزار سمبلیک تعمیم داده شده امکان دسترسی به ساختمان بزرگی از نرم افزار های نوشته شده به زبان Maple را فراهم می سازد.
زبان برنامه نویسی Maple شامل حلقه ها (for ... do ... od , while ...do ... od) و عملگر های شرطی ( if ... elif ... else ... fi ) و ... می باشد. برای اطلاع در مورد نحوه ی بکار گیری آن ها به راهنمای Maple مراجعه فرمایید.
در این بخش فرا می گیرید که چگونه package های Maple را باز کنید و چگونه از زیر برنامه های آماده ی Maple استفاده کنید. جهت آگاهی بیشتر می توانید به منابعی که در اینجا معرفی کرده ایم مراجعه فرمایید.

Char, B.W., K.O. Geddes, G.H. Gonnet, B.L. Leong, M.B. Monagan, and S.M. Watt, *First Leaves: A Tutorial Introduction to Maple V*, Springer-Verlag, NY, 1991.

Char, B.W., K.O. Geddes, G.H. Gonnet, B.L. Leong, M.B. Monagan, and S.M. Watt, *Maple V Language Reference Manual*, Springer-Verlag, NY, 1991.

Char, B.W., K.O. Geddes, G.H. Gonnet, B.L. Leong, M.B. Monagan, and S.M. Watt, *Maple V Library Reference Manual*, Springer-Verlag, NY, 1991.

Heck, A., *Introduction to Maple*, Springer-Verlag, NY, 1996.

Nicolaides, R. and N. Walkington, *Maple: A Comprehensive Introduction*, Cambridge University Press, Cambridge, 1996.

### ۲-۸-۱) package های توابع کتابخانه ای

کتابخانه ها یا package های اختصاصی شده می توانند مورد استفاده ی جعبه ابزار سمبلیک تعمیم داده شده قرار گیرند. این package ها شامل موارد زیر می باشند:

- Combinatorial Functions
- Differential Equation Tools
- Differential Forms
- Domains of Computation
- Euclidean Geometry
- Gaussian Integers
- Gröbner Bases

[1] Extended Symbolic Math Toolbox

•Permutation and Finitely Presented Groups
•Lie Symmetries
•Boolean Logic
•Graph Networks
•Newman-Penrose Formalism
•Number Theory
•Numerical Approximation
•Orthogonal Polynomials
•p-adic Numbers
•Formal Power Series
•Projective Geometry
•Simplex Linear Optimization
•Statistics
•Total Orders on Names
•Galois Fields
•Linear Recurrence Relation Tools
•Financial Mathematics
•Rational Generating Functions
•Tensor Computations

می توانید از تابع with در Maple جهت باز کردن این package ها استفاده کنید. برای مثال فرض کنید که می خواهید از package چند جمله ای های قائم یعنی orthogonal polynomials استفاده کنید. ابتدا نام این package در Maple به وسیله ی دستور mhelp index[packages] بدست آورید. نتیجه ی این دستور به این صورت است:

```
Index of descriptions for packages of library functions
Description:
-   The following packages are available:
        ...
        orthopoly orthogonal polynomials
        ...
```

پس از این می توانید به وسیله ی دستور mhelp orthopoly به اطلاعات مورد نیاز در مورد این package دست پیدا کنید. برای باز کردن package :

```
>>maple('with(orthopoly);')
ans =
        [G, H, L, P, T, U]
```

و پاسخ که در واقع لیست شده ی نام توابع در package با نام orthopoly می باشد. حالا دیگر این توابع در فضای کاری Maple بارگذاری شده اند و شما می توانید آن ها را مانند توابع عادی Maple فراخوانی کنید.

## ۲-۸-۲) مثال از یک زیر برنامه

مثال زیر پاسخ این سوال را به شما می دهد که چگونه می توان در MATLAB به وسیله ی جعبه ابزار سمبلیک تعمیم داده شده به یک زیر برنامه ی Maple دسترسی پیدا کرد. این مثال هرکدام از تقریب های عددی و یا سمبلیک نزدیک به عدد $\pi$ را با استفاده از روش ریچارد برنت (بر اساس الگوریتم گوس) محاسبه می کند. در اینجا کد نوشته شده ی آن را به زبان Maple مشاهده می کنید:

```
pie := proc(n)
# pie(n) takes n steps of an arithmetic geometric mean
# algorithm for computing pi. The result is a symbolic
```

```
# expression whose length roughly doubles with each step.
# The number of correct digits in the evaluated string also
# roughly doubles with each step.
# Example: pie(5) is a symbolic expression with 1167
# characters which, when evaluated, agrees with pi to 84
# decimal digits.
local a,b,c,d,k,t;
a := 1:
b := sqrt(1/2):
c := 1/4:
t := 1:
for k from 1 to n do
        d := (b-a)/2:
        b := sqrt(a*b):
        a := a+d:
        c := c-t*d^2:
        t := 2*t:
od;
        (a+b)^2/(4*c):
end;
```

فرض کنید که کد نوشته شده برای این زیر برنامه در فایلی به نام pie.src ذخیره شده است. استفاده از جعبه ابزار سمبلیک تعمیم داده شده را با دستورات زیر شروع می کنیم:

```
>>procread('pie.src')
```

این دستور فایل مشخص شده را خوانده، پس از حذف خطوط توضیحی و کارکتر های سطر جدید، رشته ی بدست آمده از فایل را به Maple ارسال می کند. (پس از این متغیر ans در MATLAB یک مقدار رشته ای معادل کدی که از فایل خوانده شده است را نشان می دهد)

برای استفاده از توابع Maple می توانید از تابع pie استفاده کنید:

```
>>p = maple('pie',5)
```

این دستور مقداری را برگشت می دهد که با عبارات زیر شروع و خاتمه می یابد:

```
p =
        1/4*(1/32+1/64*2^(1/2)+1/32*2^(3/4)+ ...
                ... *2^(1/2))*2^(3/4))^(1/2))^(1/2))^2)
```

شما می توانید از تابع sym جهت تبدیل یک رشته به یک شئ سمبلیک استفاده کنید. کار جالبی که می توان انجام داد تبدیل محاسبات از نوع سمبلیک به نوع عددی است. در زیر برنامه ی نوشته شده نکته ی کلیدی انتصابی است که در سطر دوم به متغیر b انجام شده است. اگر این انتصاب به سادگی به صورت b := sqrt(1/2) نوشته می شد، محاسبات انجام شده از لحاظ سمبلیک درست بود، اما اگر این دستور به صورت نوشته شده همراه با ممیز اعشاری اصلاح شود (یعنی b := sqrt(1./2.) )، آنگاه کل محاسبات به وسیله ی متغیر دقیق حسابی (با تنظیمات جاری موجود برای digits) انجام می شود. اگر این تغییر انجام شود، آنگاه دستورات

```
>>digits(100)
>>procread('pie.src')
>>p = maple('pie',5)
```

نتیجه را همراه با ۱۰۰ رقم اعشار نمایش خواهند داد:

```
p =
        3.14159265358979323 ... 5628703211672038
```

البته در این مثال ۱۶ رقم آخر نمایش داده شده با مقدار واقعی آن ها در عدد π تفاوت دارد، زیرا الگوریتمی که ما به کار برده ایم برای ۵ بار اجرای دستورات داخل حلقه، تا ۸۴ رقم درست از عدد π را به ما می دهد.
در ضمن به خاطر داشته باشید که شما می توانید یک M-file دلخواه ایجاد کنید که به زیر برنامه های Maple دسترسی داشته باشد.

## ۲-۸-۲) زیر برنامه های از پیش کامپایل شده ی Maple

وقتی که Maple یک کد (متن اسکی) را به داخل فضای کاری خود باز می کند، آن را به یک فرمت داخلی تبدیل می کند. بعدا شما می توانید از تابع Maple جهت ذخیره ی این کد با فرمت داخلی گفته شده استفاده کنید. بهتر است که شما از کامپایل شدن دوباره ی زیر برنامه در دفعات بعدی باز کردن آن جلوگیری کنید تا سرعت انجام عملیات بیشتر شود.
برای مثال شما می توانید کد pie.src که در مثال قبل گفته شد را با استفاده از دستورات زیر به یک زیر برنامه ی از پیش کامپایل شده تبدیل کنید.

```
>>clear maplemex
>>procread('pie.src')
>>maple('save(`pi.m`)');
```

دستور clear maplemex فضای کاری Maple را ریست کرده و به حالت اولیه باز می گرداند. تا زمانی که دستور save در Maple کلیه ی متغیرهای جلسه ی جاری را ذخیره می کند، ما متغیر های اضافی را حذف خواهیم کرد.

برای خواندن زیر برنامه ی از پیش کامپایل شده ی مورد نظر دستور زیر را تایپ کنید (به علامت ` دقت کنید):

```
>>maple('read','`pie.m`');
```

از اینجا به بعد مانند کد های با متن اسکی ما می توانیم به وسیله ی دستور Maple به این تابع دسترسی داشته باشیم:

```
>>p = maple('pie',5)
```

به خاطر داشته باشید که توابع از پیش کامپایل شده ی Maple دارای پسوند m. می باشند، از این جهت باید دقت کنید که آن ها را با M-file های MATLAB اشتباه نگیرید.

# فصل ۳

## مرجع توابع

**این فصل شامل بخش های زیر می باشد:**

**لیست طبقه بندی شده ی توابع:** جدول توابع جعبه ابزار ریاضیات سمبلیک به ترتیب حروف الفبا

**لیست توابع و شرح توابع به ترتیب حروف الفبا:**
لیست الفبایی توابع جعبه ابزار ریاضیات سمبلیک و تشریح آنها

## بخش ۱

# لیست طبقه بندی شده ی توابع

در این فصل تشریح کامل توابع جعبه ابزار ریاضیات سمبلیک همراه با شرح جزئیات آن ها گرد آوری شده است. این فصل با جدول توابع شروع شده و با بیان صفحات مرجع برای لیست الفبایی توابع ادامه می یابد.

### توابع حساب دیفرانسیل و انتگرال:

| | |
|---|---|
| دیفرانسیل، مشتق | diff |
| انتگرال | int |
| ماتریس ژاکوبین | jacobian |
| حد عبارت | limit |
| مجموع سری | symsum |
| بسط سری تیلور | taylor |

### توابع جبر خطی:

| | |
|---|---|
| یافتن پایه های ستون های ماتریس | colspace |
| دترمینان | det |
| ایجاد یا استخراج قطر اصلی | diag |
| مقدار ویژه و بردار ویژه | eig |
| ماتریس نمایی سمبلیک | expm |
| معکوس ماتریس | inv |
| صورت متعارفی جردن | jordan |
| پایه های فضاهای پوچ | null |
| چند جمله ای مشخصه | poly |
| درجه ی ماتریس (بُعد) | rank |
| شکل کاهش یافته پلکانی سطری یک ماتریس | rref |
| مقدار منحصر به فرد تجزیه | svd |
| مثلث پایینی ماتریس | tril |
| مثلث بالایی ماتریس | triu |

### توابع ساده سازی:

| | |
|---|---|
| جمع آوری ضرایب مشترک هر توان در چند جمله ای | collect |

| | |
|---|---|
| پخش عملگر ها روی عملگر جمع و تفریق | expand |
| فاکتور گیری | factor |
| نمایش تو در توی چند جمله ای ها | horner |
| صورت و مخرج کسر | numden |
| جستجو برای کوتاه ترین شکل عبارت | simple |
| ساده کردن | simplify |
| جایگزینی در عبارات | subexpr |

## توابع حل معادله:

| | |
|---|---|
| ترکیب تابعی | compose |
| حل معادلات دیفرانسیل | dsolve |
| مکعوس توابع | finverse |
| حل معادلات جبری | solve |

## توابع محاسبات دقت متغیر:

| | |
|---|---|
| تنظیم مقدار دقت متغیر | digits |
| محاسبات دقت متغیر | vpa |

## عملگر های حسابی:

| | |
|---|---|
| جمع | + |
| تفریق | - |
| ضرب | * |
| ضرب عناصر آرایه | .* |
| تقسیم راست(عملگر ماتریسی) | / |
| تقسیم راست (آرایه) | ./ |
| تقسیم چپ (عملگر ماتریسی) | \ |
| تقسیم چپ (آرایه) | .\ |
| توان(عملگر ماتریسی) | ^ |
| توان (آرایه) | .^ |
| ترانهاده ی ماتریس(همراه با مزدوج کردن اعداد مختلط موجود) | ' |
| ترانهاده ی ماتریس(بدون مزدوج کردن اعداد مختلط موجود) | .' |

## توابع خاص:

| | |
|---|---|
| تابع کسینوس انتگرال با تعریف $Ci(x)$ | cosint |
| تابع تعمیم یافته ی هندسه ی فوق فضایی | hypergeom |
| حل معادله ی $\lambda(x)e^{\lambda(x)} = x$ | lambertw |
| تابع سینوس انتگرال با تعریف $Si(x)$ | cosint |
| تابع زتای ریمن | zeta |

## توابع دسترسی به Maple:

| | |
|---|---|
| دستیابی به هسته یMaple | maple |
| مقدار دهی اولیه ی Maple | mapleinit |
| ارزیابی عددی توابع Maple | mfun |
| دسترسی به Maple help | mhelp |
| لیست توابع برای تابع mfun | mfunlist |
| نصب یک زیر برنامه ی Maple | procread |

## توابع ترسیمی – تعلیمی :

| | |
|---|---|
| رسم کننده ی خط واصل | ezcontour |
| رسم کننده ی خط واصل پر | ezcontourf |
| رسم کننده ی شبکه ای | ezmesh |
| ترکیبی از رسم کننده ی شبکه ای و رسم کننده ی خط واصل | ezmeshc |
| تابع رسم کننده | ezplot |
| رسم کننده منحنی سه بعدی | ezplot3 |
| رسم کننده مختصات قطبی | ezpolar |
| رسم کننده سطحی | ezsurf |
| ترکیبی از رسم کننده ی شبکه ای و رسم کننده ی سطحی | ezsurfc |
| تابع ماشین حساب | funtool |
| مجموع سری ریمان | rsums |
| محاسبه کننده ی سری تیلور | taylortool |

## توابع تبدیلات:

| | |
|---|---|
| تبدیل اشیاء سمبلیک به رشته | char |
| تبدیل اشیاء سمبلیک به double | double |
| تبدیل بردار ضرایب به چند جمله ای سمبلیک | poly2sym |
| تبدیل چند جمله ای سمبلیک به بردار ضرایب | sym2poly |

## عملگر های پایه:

| | |
|---|---|
| نمایش عبارت سمبلیک به زبان C | ccode |
| مزدوج عدد مختلط | conj |
| تعیین متغیر های سمبلیک یک عبارت | findsym |
| نمایش عبارت سمبلیک به زبان فرترن | fortran |
| قسمت موهومی عدد مختلط | imag |
| نمایش عبارت سمبلیک به شکل لاستیکی! | latex |
| نمایش عبارت سمبلیک با شکلی ریاضیاتی | pretty |
| قسمت حقیقی عدد مختلط | real |
| ایجاد یک شئ سمبلیک | sym |
| ایجاد همزمان چند شئ سمبلیک | syms |

## توابع تبدیل انتگرال:

| | |
|---|---|
| تبدیل فوریه | fourier |
| معکوس تبدیل فوریه | ifourier |
| معکوس تبدیل لاپلاس | ilaplace |
| معکوس تبدیل z | iztrans |
| تبدیل لاپلاس | laplace |
| تبدیل z | ztrans |

# بخش ۲

## لیست توابع و شرح توابع به ترتیب حروف الفبا

### عملگرهای حسابی :

**هدف:** انجام عملگرهای حسابی روی اشیاء سمبلیک.

**املای درست:**

```
A+B
A-B
A*B   A.*B
A\B   A.\B
A/B    A./B
A^B   A.^B
A'     A.'
```

**شرح:**

+ جمع ماتریسی. دستور A+B ، ماتریس های A و B را با هم جمع می کند. دو ماتریس باید ابعاد برابر داشته باشند یا اینکه یکی از این دو اسکالر باشد.

- تفریق ماتریسی. دستور A-B ، ماتریس B را از ماتریس A کم می کند. دو ماتریس باید ابعاد برابر داشته باشند یا اینکه یکی از این دو اسکالر باشد.

* ضرب ماتریسی. دستور A*B حاصل ضرب جبری دو ماتریس را بر می گرداند. ستون های ماتریس A باید با سطر های ماتریس B برابر باشند یا اینکه یکی از این دو اسکالر باشد.

.* ضرب آرایه ای. دستور A.*B ماتریسی بر می گرداند که حاصل ضرب نظیر به نظیر درایه های دو ماتریس (آرایه) است. دو ماتریس باید ابعاد برابر داشته باشند یا اینکه یکی از این دو اسکالر باشد.

\ تقسیم چپ ماتریسی. X = A\B معادله ی سمبلیک خطی A*X=B را حل می کند. به خاطر داشته باشید که دستور A\B تقریبا معادل است با inv(A)*B . اگر جواب X وجود نداشته باشد یا اینکه جواب منحصر به فرد نداشته باشد پیغام های هشدار مشاهده خواهد شد. ماتریس A می تواند مستطیلی نیز باشد، اما باید با سایر داده های مسئله سازگاری داشته باشد. هنوز پاسخی برای مسئله ی کوچکترین مربعات محاسبه نشده است.

.\ تقسیم چپ آرایه ای. حاصل A.\B ماتریسی مانند C است که به ازای هر i و j خواهیم داشت: C(i,j)=B(i,j)/A(i,j) . دو ماتریس باید ابعاد برابر داشته باشند یا اینکه یکی از این دو اسکالر باشد.

/ تقسیم راست ماتریسی. X=A/B معادله ی سمبلیک خطی X*A=B را حل می کند. به خاطر داشته باشید که B/A مشابه دستور '.(A.' \ B.') است. اگر جواب X وجود نداشته باشد یا اینکه جواب منحصر به فرد نداشته باشد پیغام های هشدار مشاهده خواهد شد. ماتریس A می تواند مستطیلی نیز باشد، اما باید با سایر داده های مسئله سازگاری داشته باشد. هنوز پاسخی برای مسئله ی کوچکترین مربعات محاسبه نشده است.

./ تقسیم راست آرایه ای. حاصل A./B ماتریسی مانند C است که به ازای هر i و j خواهیم داشت: C(i,j)=A(i,j)/B(i,j) . دو ماتریس باید ابعاد برابر داشته باشند یا اینکه یکی از این دو اسکالر باشد.

^ توان ماتریسی. X^P ماتریس مربعی X را P مرتبه در خود ضرب می کند. اگر X یک اسکالر و P یک ماتریس مربعی باشد، آنگاه X به توان ماتریس P می رسد، البته با استفاده از مقدار ویژه و بردار ویژه. در صورتی که هم X و هم P ماتریس باشند پیغام خطا نمایش داده می شود و مقداری محاسبه نمی شود.

.^ توان آرایه ای. حاصل A.^B ماتریسی مانند C است که به ازای هر i و j خواهیم داشت: C(i,j)=A(i,j)^B(i,j) . دو ماتریس باید ابعاد برابر داشته باشند یا اینکه یکی از این دو اسکالر باشد.

' ترانهاده ی ماتریس. با استفاده از این دستور، در ماتریس ترانهاده ی حاصل، تمامی اعداد مختلط ماتریس اولیه مزدوج خواهند شد.

.' عمل ترانهاده روی آرایه. تفاوت این دستور با دستور قبل در این است که اعداد مختلط در ماتریس حاصل تغییری نسبت به ماتریس اولیه نخواهند داشت.

**مثال ها:**

```
>>syms a b c d;
>>A = [a b; c d];
>>A*A/A
ans=
        [ a, b]
        [ c, d]
>>A*A-A^2
ans=
        [ 0, 0]
        [ 0, 0]

>>syms a11 a12 a21 a22 b1 b2;
>>A = [a11 a12; a21 a22];
>>B = [b1 b2];
>>X = B/A;
>>x1 = X(1)
```

```
>>x2 = X(2)
x1 =
        (-a21*b2+b1*a22)/(a11*a22-a12*a21)
x2 =
        (a11*b2-a12*b1)/(a11*a22-a12*a21)
```

**موارد مرتبط:**

null, solve

## تابع ccode :

**هدف:** نمایش یک عبارت سمبلیک با املای زبان C.

**املای درست:**

ccode(s)

**شرح:** ccode(s) یک قطعه کد به زبان C را بر می گرداند که قادر به ارزیابی مقدار عبارت سمبلیک است.

**مثال ها:**

```
>>syms x
>>f = taylor(log(1+x));
>>ccode(f)
        t0 = x-x*x/2.0+x*x*x/3.0-x*x*x*x/4.0+x*x*x*x*x/5.0;

>>H = sym(hilb(3));
>>ccode(H)
        H[0][0] = 1.0;          H[0][1] = 1.0/2.0;          H[0][2] = 1.0/3.0;
        H[1][0] = 1.0/2.0;          H[1][1] = 1.0/3.0;          H[1][2] = 1.0/4.0;
        H[2][0] = 1.0/3.0;          H[2][1] = 1.0/4.0;          H[2][2] = 1.0/5.0;
```

**موارد مرتبط:**

fortran, latex, pretty

## تابع collect :

**هدف:** جمع آوری یکجای ضرایب هر توان از متغیر اصلی عبارت سمبلیک.

**املای درست:**

R = collect(S)
R = collect(S,v)

**شرح:** کلیه ی ضرایب یک توان مشخص از x (متغیر اصلی) را یکجا به عنوان ضریب آن توان از x جمع آوری می کند. در واقع دستور collect(f) عبارت f را به یک چند جمله ای استاندارد برحسب x تبدیل می کند. این دستور روی تک تک عناصر یک آرایه ی سمبلیک هم قابل اجرا می باشد.

**مثال ها:**

```
>>syms x y;
>>R1 = collect((exp(x)+x)*(x+2))
>>R2 = collect((x+y)*(x^2+y^2+1), y)
>>R3 = collect([(x+1)*(y+1),x+y])
R1 =
        x^2+(exp(x)+2)*x+2*exp(x)
R2 =
        y^3+x*y^2+(x^2+1)*y+x*(x^2+1)
R3 =
        [(y+1)*x+y+1, x+y]
```

**موارد مرتبط:**

expand, factor, simple, simplify, syms

## تابع colspace :

**هدف:** یافتن پایه های ستون های ماتریس.

**املای درست:**

B = colspace(A)

**شرح:** colspace(A) یک ماتریس بر می گرداند که ستون های آن پایه های ستون های ماتریس A می باشند. به خاطر داشته باشید که تعداد ستون های ماتریس بر گردانده شده (یعنی size(colspace(A),2) ) رتبه ی ماتریس A می باشد. می دانیم که تعداد ستون های ماتریس بر گردانده شده به اندازه تعداد ستون هایی از A است که مستقل خطی هستند و چون رتبه ی سطری و رتبه ی ستونی یک ماتریس با هم برابرند عدد مورد نظر رتبه ی ماتریس A می باشد.

**مثال ها:**

```
>>A = sym([2,0;3,4;0,5]);
>>B = colspace(A);
>>A,B
A =
        [2,  0]
        [3,  4]
        [0,  5]
B =
        [ 1      ,      0]
        [ 0      ,      1]
        [-15/8   ,      5/4]
```

**موارد مرتبط:**

null , orth in the online MATLAB Function Reference

## تابع compose :

**هدف:** ترکیب توابع.

**املای درست:**

```
compose(f,g)
compose(f,g,z)
compose(f,g,x,z)
compose(f,g,x,y,z)
```

**شرح:** دستور compose(f,g) مقدار f(g(y)) را با فرض f=f(x) و g=g(y) بر می گرداند. در اینجا x یک متغیر سمبلیک از f و y یک متغیر سمبلیک از g است که هر دوی آن ها به وسیله ی تابع findsym تعیین می شوند.

دستور compose(f,g,z) مقدار f(g(z)) را با فرض f=f(x) و g=g(y) بر می گرداند. در اینجا نیز x یک متغیر سمبلیک از f و y یک متغیر سمبلیک از g است که هر دوی آن ها به وسیله ی تابع findsym تعیین می شوند.

دستور compose(f,g,x,z) مقدار f(g(z)) را بر می گرداند و x را به عنوان متغیر اصلی برای f در نظر می گیرد. بدین معنی که اگر f=cos(x/t) باشد، دستور compose(f,g,x,z) مقدار cos(g(z)/t) را بر می گرداند. به همین ترتیب دستور compose(f,g,t,z) مقدار cos(x/g(z)) را بر می گرداند.

دستور compose(f,g,x,y,z) مقدار f(g(z)) را بر می گرداند و x را به عنوان متغیر اصلی برای f و y را به عنوان متغیر اصلی برای g در نظر می گیرد. برای f=cos(x/t) و g=sin(y/u)، دستور compose(f,g,x,y,z) مقدار cos(sin(z/u)/t) و به همین ترتیب دستور compose(f,g,x,y,z) مقدار cos(sin(y/z)/t) را برمی گرداند.

**مثال ها:**

فرض کنید:

```
syms x y z t u;
f = 1/(1 + x^2); g = sin(y); h = x^t; p = exp(-y/u);
```

آنگاه:

```
compose(f,g)          ->    1/(1+sin(y)^2)
compose(f,g,t)        ->    1/(1+sin(t)^2)
compose(h,g,x,z)      ->    sin(z)^t
compose(h,g,t,z)      ->    x^sin(z)
compose(h,p,x,y,z)    ->    exp(-z/u)^t
compose(h,p,t,u,z)    ->    x^exp(-y/z)
```

**موارد مرتبط:**

finverse, subs, syms

## تابع conj :

**هدف:** مزدوج عدد مختلط.

**املای درست:**

conj(x)

**شرح:** دستور conj(x) مزدوج عدد مختلط x را بر می گرداند.

در واقع برای عدد مختلط x داریم: conj(X) = real(X) - i*imag(X)

**موارد مرتبط:**

real, imag

## تابع cosint :

**هدف:** بدست آوردن کسینوس انتگرال.

**املای درست:**

Y = cosint(X)

**شرح:** این تابع، تابع Ci(x) می باشد که تعریف آن در ریاضیات به صورت زیر است:

$$Ci(x) = \gamma + \ln(x) + \int_0^x \frac{\cos t - 1}{t} dt$$

و البته $\gamma$ در این فرمول مقدار ثابت اویلر یعنی ...0.577215664 است.

**مثال ها:**

```
>>cosint(7.2)
ans=
        0.0960.
>>cosint([0:0.1:1])
ans=
     Columns 1 through 7
        Inf          -1.7279      -1.0422      -0.6492      -0.3788      -0.1778      -0.0223
     Columns 8 through 11
        0.1005       0.1983       0.2761       0.3374

>>syms x;
```

```
>>f = cosint(x);
>>diff(f)
ans=
        cos(x)/x
```

**موارد مرتبط:**

sinint

## تابع det :

**هدف:** دترمینان ماتریس.

**املای درست:**

r=det(A)

**شرح:** دستور det(A) دترمینان ماتریس A را محاسبه می کند، که A یک ماتریس سمبلیک یا عددی می باشد. اگر A یک ماتریس سمبلیک باشد حاصل هم سمبلیک و اگر A عددی باشد حاصل هم عددی خواهد بود.

**مثال ها:**

```
>>syms a b c d;
>>det([a, b; c, d])
ans=
        a*d - b*c
>>A = sym([2/3 1/3;1 1])
>>r = det(A)
A =
        [ 2/3,   1/3]
        [    1,     1]
r =
        1/3
```

## تابع diag :

**هدف:** ایجاد یا استخراج قطر ماتریس.

**املای درست:**

diag(A,k)
diag(A)

**شرح:** دستور diag(A,k)، که در آن A یک بردار سطری یا ستونی با n عنصر است، یک ماتریس سمبلیک ستونی از مرتبه ی n+abs(k) بر می گرداند که قطر اصلی ماتریس A می باشد. آرگومان دوم ارسالی به این تابع که به صورت پیش فرض صفر در نظر گرفته می شود، شماره ی قطر را مشخص می کند که در واقع باعث می شود kامین قطر برگردانده شود. اگر k>0 باشد kامین قطر بالای قطر اصلی و اگر k<0 باشد kامین قطر پایین قطر اصلی برگردانده می شود.

**مثال ها:**

```
>>v = [a b c]
>>diag(v)        %diag(v,0)
ans=
        [ a, 0, 0 ]
        [ 0, b, 0 ]
        [ 0, 0, c ]
>>diag(v,-2)
ans=
        [ 0, 0, 0, 0, 0]
        [ 0, 0, 0, 0, 0]
        [ a, 0, 0, 0, 0]
        [ 0, b, 0, 0, 0]
        [ 0, 0, c, 0, 0]
```

به خاطر داشته باشید که ما فرض کرده ایم که:

```
A =
      [ a, b, c ]
      [ 1, 2, 3 ]
      [ x, y, z ]
>>diag(A)
ans=
        [ a ]
        [ 2 ]
        [ z ]
>>diag(A,1)
ans=
        [ b ]
        [ 3 ]
```

**موارد مرتبط:**

tril, triu

## تابع diff :

**هدف:** بدست آوردن مشتق.

**املای درست:**

```
diff(S,'v')
diff(S,n)
diff(S,'v',n)
```

**شرح:** دستور diff(S) مشتق عبارت سمبلیک S بر حسب متغیر اصلی که به وسیله ی تابع findsym بدست می آید را محاسبه می کند.

دستور diff(S,'v') یا diff(S,sym('v')) مشتق عبارت سمبلیک S را بر حسب متغیر 'v' محاسبه می کند.

دستور diff(S,n) به ازای عدد صحیح n مشتق عبارت S را تا n مرتبه محاسبه می کند.

دستور diff(S,'v',n) و diff(S,n,'v') نیز قابل اجرا می باشند.

**مثال ها:**

```
>>syms x t
>>diff(sin(x^2))
ans=
        2*cos(x^2)*x
>>diff(t^6,6)
ans=
        720
```

**موارد مرتبط:**

int, jacobian, findsym

## تابع digits :

**هدف:** ست کردن دقت متغیر.

**املای درست:**

```
digits(d)
d = digits
digits
```

**شرح:** digits تعداد ارقام با معنای درست دهدهی را مشخص می کند که Maple از آن برای انجام محاسبات دقت متغیر استفاده می کند و مقدار پیش فرض آن 32 می باشد.

تابع digits(n) نمایش تعداد ارقام با معنای درست را به n رقم تغییر می دهد.

دستور digits دقت محاسبات را بر حسب تعداد اعشار برمی گرداند.

**مثال ها:**

```
>>z = 1.0e-16;
>>x = 1.0e+2;
>>digits(14)
>>y = vpa(x*z+1)
y =
        1.0000000000000
>>digits(15)
```

```
>>y = vpa(x*z+1)
y =
        1.00000000000001
```

**موارد مرتبط:**

double, vpa

## تابع double :

**هدف:** تبدیل ماتریس سمبلیک به فرم عددی MATLAB.

**املای درست:**

R = double(S)

**شرح:** دستور R = double(S) شئ سمبلیک S را به نوع عددی تبدیل می کند. اگر S یک عبارت ثابت یا عدد ثابت سمبلیک باشد، دستور double یک عدد ممیز شناور با دقت double بر می گرداند. اگر S یک ماتریس باشد که اعضای آن هر کدام یک عبارت ثابت یا عدد ثابت سمبلیک باشد، دستور double یک ماتریس که اعضای آن هر کدام عدد ممیز شناور با دقت double هستند، بر می گرداند.

**مثال ها:**

```
>>double(sym('(1+sqrt(5))/2'))
ans=
        1.6180.

>>a = sym(2*sqrt(2));
>>b = sym((1-sqrt(3))^2);
>>T = [a, b]
>>double(T)
ans =
        2.8284      0.5359
```

**موارد مرتبط:**

sym, vpa

## تابع dsolve :

**هدف:** حل سمبلیک یک معادله ی دیفرانسیل عادی.

**املای درست:**

r = dsolve('eq1,eq2,...', 'cond1,cond2,...', 'v')
r = dsolve('eq1','eq2',...,'cond1','cond2',...,'v')

**شرح:** دستور r = dsolve('eq1,eq2,...', 'cond1,cond2,...', 'v') معادله های دیفرانسیل عادی که به وسیله ی 'eq1,eq2,...' مشخص شده اند را به صورت سمبلیک حل می کند. این دستور متغیر v را به عنوان متغیر مستقل در نظر می گیرد که مقدار پیش فرض آن t است و cond1,cond2,.... شرایط مرزی را برای معادلات مشخص می کنند.
حرف D مشتق نسبت به متغیر مستقل را مشخص می کند، با پیش فرض اولیه ی d/dx . بعد از حرف D یک عدد صحیح می آید که تعداد دفعات مشتق گیری را مشخص می کند. برای مثال D2 معادل d2/dx2 می باشد. هر کارکتری که بلافاصله بعد از عملگر مشتق نوشته شود، متغیر وابسته، که مشتق نسبت به آن گرفته می شود را مشخص می کند.
برای مثال D3y مشتق سوم y(t) یا y(x) را مشخص می کند. شرایط مرزی با معادلاتی مانند y(a) = b یا Dy(a) = b مشخص می شوند که در آن y یک متغیر وابسته و a و b اعداد ثابت هستند. اگر تعداد شرایط مرزی تعیین شده کمتر از تعداد معادلات وارد شود، مقادیر دلخواه و ثابت C1,C2,.... به عنوان شرایط مرزی در نظر گرفته می شوند.
در ضمن شما می توانید معادلات و شرایط مرزی ارسال شده به تابع را به صورت عبارات سمبلیک جدا از هم وارد کنید. با این شرایط تابع dsolve تا ۱۲ آرگومان ورودی را مجاز می شمرد. در صورت مشخص نکردن آرگومان خروجی این تابع لیستی از راه حل ها را در اختیار شما می گذارد. در شرایطی که dsolve راه حلی جبری برای معادله پیدا نکند، یک پیغام هشدار چاپ خواهد کرد. در این شرایط شما می توانید از جواب های عددی که MATLAB به وسیله ی توابع ode45 و ode23 ارائه می دهد استفاده کنید.

**مثال ها:**

```
>>dsolve('Dy = a*y')
ans=
        C1*exp(a*t)
>>dsolve('Df = f + sin(t)')
ans=
        -1/2*cos(t)-1/2*sin(t)+exp(t)*C1
>>dsolve('(Dy)^2 + y^2 = 1','s')
        [         -1]
        [          1]
        [  sin(s-C1)]
        [ -sin(s-C1)]
>>dsolve('Dy = a*y', 'y(0) = b')
ans=
        b*exp(a*t)
>>dsolve('D2y = -a^2*y', 'y(0) = 1', 'Dy(pi/a) = 0')
ans=
        cos(a*t)
>>dsolve('Dx = y', 'Dy = -x')
        x: [1x1 sym]
        y: [1x1 sym]
```

**موارد مرتبط:**

syms

## تابع eig :

**هدف:** بدست آوردن مقدار ویژه و بردار ویژه ی یک ماتریس سمبلیک.

**املای درست:**

```
lambda = eig(A)
[V,D] = eig(A)
[V,D,P] = eig(A)
lambda = eig(vpa(A))
[V,D] = eig(vpa(A))
```

**شرح:** دستور lambda = eig(A) یک بردار سمبلیک که شامل مقدار ویژه ی ماتریس مربعی و سمبلیک A را بر می گرداند.
دستور [V,D] = eig(A) ماتریس V را که ستون های آن بردارهای ویژه و ماتریس قطری D را که شامل مقادیر ویژه هست، بر می گرداند. اگر ماتریس V هم اندازه ی ماتریس A باشد، آنگاه ماتریس A یک مجموعه ی مستقل خطی از بردار های ویژه را داراست و خواهیم داشت: A*V = V*D
دستور [V,D,P] = eig(A) علاوه بر V و D بردار P را نیز بر می گرداند. بردار P برداریست از اندیس ها که طول آنها برابر تعداد بردار های ویژه ی مستقل خطی است. چنان که : A*V = V*D(P,P)
دستورات lambda = eig(vpa(A)) و [V,D] = eig(vpa(A)) به ترتیب مقادیر ویژه و بردار های ویژه را به صورت عددی و با استفاده از محاسبات دقت متغیر محاسبه می کنند. اگر A یک مجموعه ی کامل از بردار های ویژه نداشته باشد، سطر های V مستقل خطی نخواهند بود.

**مثال ها:**

```
>>R = sym(gallery('rosser'));
>>eig(R)
ans =
        [                    0  ]
        [                 1020  ]
        [     510+100*26^(1/2)  ]
        [      510-100*26^(1/2)  ]
        [        10*10405^(1/2)  ]
        [       -10*10405^(1/2)  ]
        [                 1000  ]
        [                 1000  ]
>>eig(vpa(R))
ans =
        [          -1020.04901842999682384631379130 55  ]
        [       .56512999999999999999999999999800e-28  ]
        [        .98048640721516997177589097485157e-1  ]
        [          1000.00000000000000000000000000 2  ]
        [          1000.00000000000000000000000000 3  ]
        [          1019.90195135927848300282241090 24  ]
        [          1020.00000000000000000000000000 3  ]
        [          1020.04901842999682384631379130 55  ]

>>A = sym(gallery(5));
>>[v,lambda] = eig(A)
v =
        [                 0  ]
```

```
        [          21/256  ]
        [         -71/128  ]
        [         973/256  ]
        [               1  ]
lambda =
        [ 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]
```

**موارد مرتبط:** jordan, poly, svd, vpa

## تابع expm :

**هدف:** بدست آوردن ماتریس نمایی سمبلیک.

**املای درست:**

expm(A)

**شرح:** دستور expm(A) ماتریس نمایی ماتریس A را بر می گرداند.

**مثال ها:**

```
>>syms t;
>>A = [0 1; -1 0];
>>expm(t*A)
ans=
        [        cos(t),          sin(t)  ]
        [        -sin(t),         cos(t)  ]
```

## تابع expand :

**هدف:** بسط دادن عبارات سمبلیک به صورت سمبلیک (پخش توابع و عملگر های دیگر روی عملگر جمع و تفریق).

**املای درست:**

R = expand(S)

**شرح:** R = expand(S) عملگر های موجود در S را روی عملگر های جمع و تفریق پخش می کند. این تابع بیشتر برای چند جمله ای ها به کار می رود، اما می توان این تابع را برای عبارات مثلثاتی، نمایی و لگاریتمی هم به کار برد.

**مثال ها:**

```
>>expand((x-2)*(x-4))
ans=
        x^2-6*x+8
>>expand(cos(x+y))
ans=
```

```
        cos(x)*cos(y)-sin(x)*sin(y)
>>expand(exp((a+b)^2))
ans=
        exp(a^2)*exp(a*b)^2*exp(b^2)
expand([sin(2*t), cos(2*t)])
ans=
        [2*sin(t)*cos(t), 2*cos(t)^2-1]
```

**موارد مرتبط:**

collect, factor, horner, simple, simplify, syms

## تابع ezcontour :

**هدف:** رسم منحنی تراز.

**املای درست:**

```
ezcontour(f)
ezcontour(f,domain)
ezcontour(...,n)
```

**شرح:** دستور ezcontour(f) منحنی تراز تابع f(x,y) را رسم می کند که در آن f یک عبارت سمبلیک است که تابعی ریاضیاتی از دو متغیر x و y را نشان می دهد.

تابع f روی بازه ی پیش فرض یعنی $-2\pi \le x \le 2\pi$ و $-2\pi \le y \le 2\pi$ رسم می شود. MATLAB طبق بازه های تعیین شده ناحیه را به صورت تور مانند تقسیم بندی می کند و روی گره های این تور مقدار تابع f را محاسبه می کند و این نقاط را رسم می کند. اگر تابع f بر روی نقاط مورد نظر تعریف نشده باشد، در آن ناحیه منحنی رسم نخواهد شد.

دستور ezcontour(f,domain) منحنی تراز تابع f(x,y) را روی بازه ی domain رسم می کند. domain می تواند به صورت یک ماتریس(بردار) ۱ در ۴ به صورت [xmin, xmax, ymin, ymax] یا یک ماتریس ۱ در ۲ به صورت [min, max] باشد که در ماتریس ۱ در ۲ داریم: $min < x < max \quad , \quad min < y < max$

اگر تابع f برای متغیر های u و v تعریف شده باشد (به جای متغیر های x و y) آنگاه نقاط umin و umax و vmin و vmax در ماتریس ۱ در ۴ بالا به صورت الفبایی مرتب و مقدار دهی می شوند. بنا بر این دستور ezcontour(u^2 - v^3,[0,1],[3,6]) منحنی تراز تابع $u^2 - v^3$ را روی بازه $0 < u < 1,\ 3 < v < 6$ رسم می کند.

ezcontour(...,n) تابع f روی بازه پیش فرض و با استفاده از تقسیم بندی تور مانند n در n رسم می کند. مقدار پیش فرض برای n عدد 60 می باشد.

ezcontour به طور خودکار یک عنوان و برچسب محور ها را اضافه می کند.

**مثال ها:** فرمول زیر یک تابع ریاضیاتی بر حسب دو متغیر $x$ و $y$ تعریف می کند.

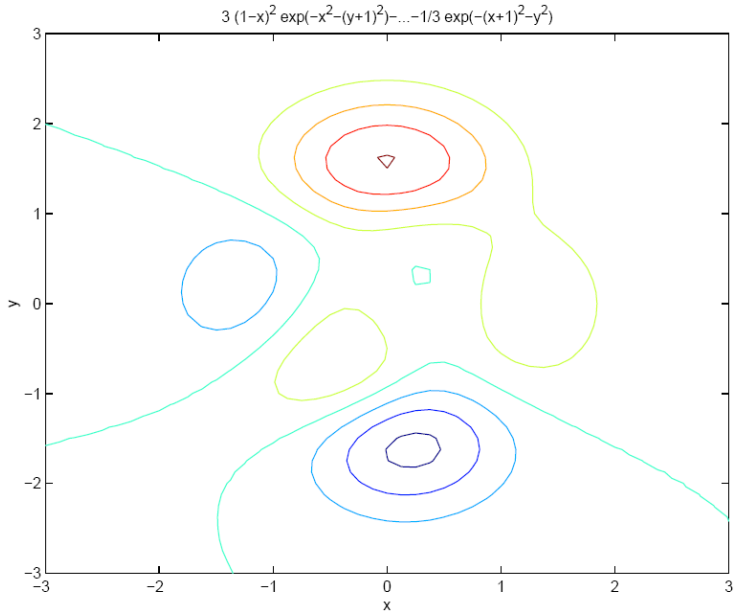
$$f(x,y) = 3(1-x)^2 e^{-x^2-(y+1)^2} - 10\left(\frac{x}{5} - x^3 - y^5\right)e^{-x^2-y^2} - \frac{1}{3}e^{-(x+1)^2-y^2}$$

با دستورات زیر شروع می کنیم:

```
>>syms x y
>>f = 3*(1-x)^2*exp(-(x^2)-(y+1)^2) ...
        - 10*(x/5 - x^3 - y^5)*exp(-x^2-y^2) ...
        - 1/3*exp(-(x+1)^2 - y^2);
```

برای راحتی و خوانایی این عبارت بر روی ۳ خط نوشته شده است. دستور زیر این تابع را در بازه $-3$ تا $3$ رسم می کند و تقسیم بندی شبکه ای ناحیه 49 در 49 می باشد.

```
>>ezcontour(f,[-3,3],49)
```




البته در این مورد خاص به دلیل طولانی بودن عنوان منحنی، MATLAB خلاصه شده آن را چاپ کرده است.

**موارد مرتبط:**

contour, ezcontourf, ezmesh, ezmeshc, ezplot, ezplot3, ezpolar, ezsurf, ezsurfc

## تابع ezcontourf :

**هدف:** رسم منحنی تراز پر.

**املای درست:**

```
ezcontourf(f)
ezcontourf(f,domain)
ezcontourf(...,n)
```

**شرح:** دستور ezcontourf(f) منحنی تراز تابع f(x, y) را رسم می کند که در آن f یک عبارت سمبلیک است که تابعی ریاضیاتی از دو متغیر x و y را نشان می دهد.

تابع f روی بازه ی پیش فرض یعنی $-2\pi \le x \le 2\pi$ و $-2\pi \le y \le 2\pi$ رسم می شود. MATLAB طبق بازه های تعیین شده ناحیه را به صورت تور مانند تقسیم بندی می کند و روی گره های این تور مقدار تابع f را محاسبه می کند و این نقاط را رسم می کند. اگر تابع f بر روی نقاط مورد نظر تعریف نشده باشد، در آن ناحیه منحنی رسم نخواهد شد.

دستور ezcontourf(f,domain) منحنی تراز تابع f(x, y) را روی بازه ی domain رسم می کند. domain می تواند به صورت یک ماتریس(بردار) ۱ در ۴ به صورت [xmin, xmax, ymin, ymax] یا یک ماتریس ۱ در ۲ به صورت [min, max] باشد که در ماتریس ۱ در ۲ داریم: $min < x < max \quad , \quad min < y < max$

اگر تابع f برای متغیر های u و v تعریف شده باشد (به جای متغیر های x و y) آنگاه نقاط umin و umax و vmin و vmax در ماتریس ۱ در ۴ بالا به صورت الفبایی مرتب و مقدار دهی می شوند. بنا بر این دستور ezcontourf(u^2 - v^3,[0,1],[3,6]) منحنی تراز تابع $u^2 - v^3$ را روی بازه $0 < u < 1,\ 3 < v < 6$ رسم می کند.

ezcontourf(...,n) تابع f روی بازه پیش فرض و با استفاده از تقسیم بندی تور مانند n در n رسم می کند. مقدار پیش فرض برای n عدد 60 می باشد.

ezcontourf به طور خودکار یک عنوان و برچسب محور ها را اضافه می کند.

**مثال ها:** فرمول زیر یک تابع ریاضیاتی بر حسب دو متغیر $x$ و $y$ تعریف می کند.

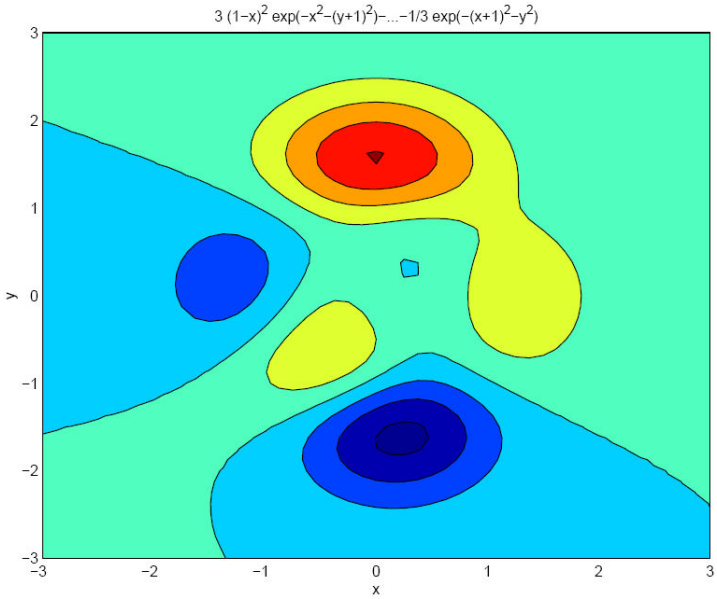
$$f(x,y) = 3(1-x)^2 e^{-x^2-(y+1)^2} - 10\left(\frac{x}{5} - x^3 - y^5\right)e^{-x^2-y^2} - \frac{1}{3}e^{-(x+1)^2-y^2}$$

با دستورات زیر شروع می کنیم:

```
>>syms x y
>>f = 3*(1-x)^2*exp(-(x^2)-(y+1)^2) ...
        - 10*(x/5 - x^3 - y^5)*exp(-x^2-y^2) ...
        - 1/3*exp(-(x+1)^2 - y^2);
```

برای راحتی و خوانایی این عبارت بر روی ۳ خط نوشته شده است. دستور زیر این تابع را در بازه $-3$ تا 3 رسم می کند و تقسیم بندی شبکه ای ناحیه 49 در 49 می باشد.

```
>>ezcontourf(f,[-3,3],49)
```




البته در این مورد خاص به دلیل طولانی بودن عنوان منحنی، MATLAB خلاصه شده آن را چاپ کرده است.

**موارد مرتبط:**

contour, ezcontourf, ezmesh, ezmeshc, ezplot, ezplot3, ezpolar, ezsurf, ezsurfc

## تابع ezmesh :

**هدف:** رسم شبکه ای سه بعدی.

**املای درست:**

```
ezmesh(f)
ezmesh(f,domain)
ezmesh(x,y,z)
ezmesh(x,y,z,[smin,smax,tmin,tmax]) or ezmesh(x,y,z,[min,max])
ezmesh(...,n)
ezmesh(...,'circ')
```

**شرح:** تابع ezmesh(f) یک نمودار هندسی از تابع f رسم می کند که در آن f یک عبارت سمبلیک است که تابعی ریاضیاتی از دو متغیر x و y را نشان می دهد.

تابع f روی بازه ی پیش فرض یعنی $-2\pi \le x \le 2\pi$ و $-2\pi \le y \le 2\pi$ رسم می شود. MATLAB طبق بازه های تعیین شده ناحیه را به صورت تور مانند تقسیم بندی می کند و روی گره های این تور مقدار تابع f را محاسبه می کند و این نقاط را رسم می کند. اگر تابع f بر روی نقاط مورد نظر تعریف نشده باشد، در آن ناحیه منحنی رسم نخواهد شد.

دستور ezmesh(f,domain) نمودار تابع f(x,y) را روی بازه ی domain رسم می کند. domain می تواند به صورت یک ماتریس(بردار) ۱ در ۴ به صورت [xmin, xmax, ymin, ymax] یا یک ماتریس ۱ در ۲ به صورت [min, max] باشد که در ماتریس ۱ در ۲ داریم: $min < x < max$ , $min < y < max$

اگر تابع f برای متغیر های u و v تعریف شده باشد (به جای متغیر های x و y) آنگاه نقاط umin و umax و vmin و vmax در ماتریس ۱ در ۴ بالا به صورت الفبایی مرتب و مقدار دهی می شوند. بنا بر این دستور ezmesh(u^2 - v^3,[0,1],[3,6]) نمودار تابع $u^2 - v^3$ را روی بازه $0 < u < 1, 3 < v < 6$ رسم می کند.

تابع ezmesh(x,y,z) رویه پارامتریک را با پارامتر های $x = x(s,t)$ و $y = y(s,t)$ و $z = z(s,t)$ روی مربع $-2\pi < s < 2\pi$ و $2\pi < t < 2\pi$ رسم می کند.

تابع ezmesh(x,y,z,[smin,smax,tmin,tmax]) یا ezmesh(x,y,z,[min,max]) رویه پارامتریک را با استفاده از بازه های تعیین شده رسم می کند.

ezmesh(...,n) تابع f روی بازه پیش فرض و با استفاده از تقسیم بندی تور مانند n در n رسم می کند. مقدار پیش فرض برای n عدد 60 می باشد.

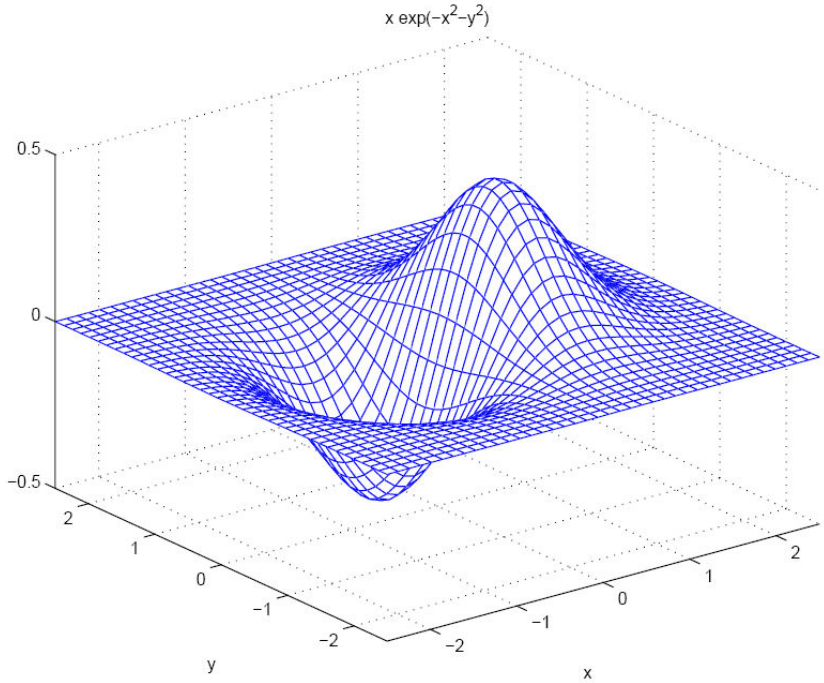
تابع ezmesh(...,'circ') تابع f را روی یک ناحیه دایره ای رسم می کند.

**مثال ها:** در این مثال تابع زیر را با یک رسم شبکه ای 40 در 40 رسم می کنیم، در اینجا نمودار شبکه ای را با استفاده از یک رنگ آبی یکنواخت رسم می کنیم.

$$f(x,y) = xe^{-x^2-y^2}$$

دستورات زیر را ببینید:

```
>>syms x y
>>ezmesh(x*exp(-x^2-y^2),[-2.5,2.5],40)
>>colormap([0 0 1])
```




**موارد مرتبط:**

ezcontour, ezcontourf, ezmeshc, ezplot, ezplot3, ezpolar, ezsurf, ezsurfc , mesh

## تابع ezmeshc :

**هدف:** رسم منحنی با ترکیب رسم شبکه ای و رسم منحنی تراز.

**املای درست:**

```
ezmeshc(f)
ezmeshc(f,domain)
ezmeshc(x,y,z)
ezmeshc(x,y,z,[smin,smax,tmin,tmax]) or ezmeshc(x,y,z,[min,max])
ezmeshc(...,n)
ezmeshc(...,'circ')
```

**شرح:** تابع ezmeshc(f) یک نمودار هندسی از تابع f رسم می کند که در آن f یک عبارت سمبلیک است که تابعی ریاضیاتی از دو متغیر x و y را نشان می دهد.

تابع f روی بازه ی پیش فرض یعنی $-2\pi \le x \le 2\pi$ و $-2\pi \le y \le 2\pi$ رسم می شود. MATLAB طبق بازه های تعیین شده ناحیه را به صورت تور مانند تقسیم بندی می کند و روی گره های این تور مقدار تابع f را محاسبه می کند و این نقاط را رسم می کند. اگر تابع f بر روی نقاط مورد نظر تعریف نشده باشد، در آن ناحیه منحنی رسم نخواهد شد.

تابع ezmeshc(f,domain) نمودار تابع f(x, y) را روی بازه ی domain رسم می کند. domain می تواند به صورت یک ماتریس(بردار) ۱ در ۴ به صورت [xmin, xmax, ymin, ymax] یا یک ماتریس ۱ در ۲ به صورت [min, max] باشد که در ماتریس ۱ در ۲ داریم: $min < x < max$ , $min < y < max$

اگر تابع f برای متغیر های u و v تعریف شده باشد (به جای متغیر های x و y) آنگاه نقاط umin و umax و vmin و vmax در ماتریس ۱ در ۴ بالا به صورت الفبایی مرتب و مقدار دهی می شوند. بنا بر این دستور ezmeshc(u^2 - v^3,[0,1],[3,6]) نمودار تابع $u^2 - v^3$ را روی بازه $0 < u < 1, 3 < v < 6$ رسم می کند.

تابع ezmeshc(x,y,z) رویه پارامتریک را با پارامتر های $x = x(s,t)$ و $y = y(s,t)$ و $z = z(s,t)$ روی مربع $-2\pi < s < 2\pi$ و $2\pi < t < 2\pi$ رسم می کند.

تابع ezmeshc(x,y,z,[smin,smax,tmin,tmax]) یا ezmeshc(x,y,z,[min,max]) رویه پارامتریک را با استفاده از بازه های تعیین شده رسم می کند.

ezmeshc(...,n) تابع f روی بازه پیش فرض و با استفاده از تقسیم بندی تور مانند n در n رسم می کند. مقدار پیش فرض برای n عدد 60 می باشد.

تابع ezmeshc(...,'circ') تابع f را روی یک ناحیه دایره ای رسم می کند.

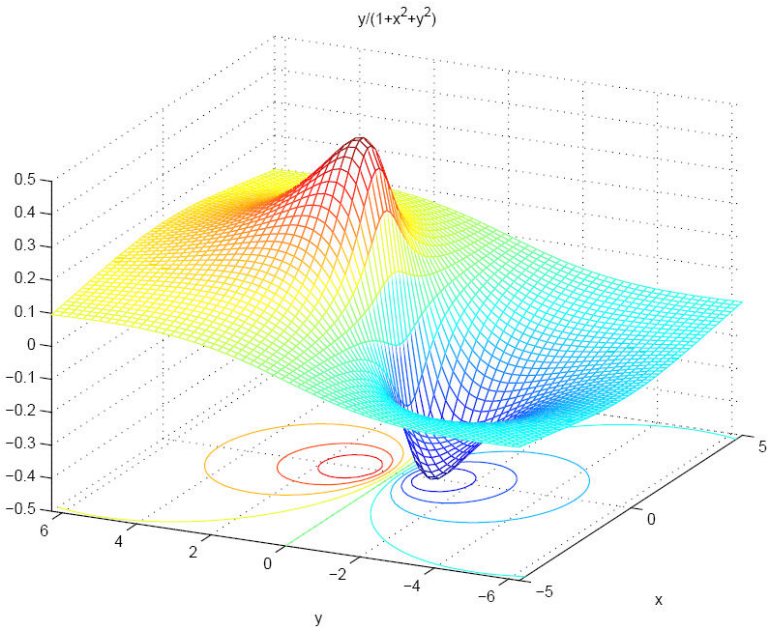
**مثال ها:** رسم شبکه سه بعدی و منحنی تراز عبارت زیر روی بازه $-5 < x < 5$ , $-2*pi < y < 2*pi$

$$f(x,y) = \frac{y}{1+x^2+y^2}$$

دستورات زیر را در نظر بگیرید:

```
>>syms x y
>>ezmeshc(y/(1 + x^2 + y^2),[-5,5,-2*pi,2*pi])
```

به وسیله درگ کردن با ماوس می توانید این نمودار را بچرخانید تا بتوانید خطوط منحنی تراز را بهتر ببینید. در این مثال از زاویه افقی 65- و زاویه فراز 26 استفاده شده است.




**موارد مرتبط:**

zcontour, ezcontourf, ezmesh, ezplot, ezplot3, ezpolar, ezsurf,ezsurfc , meshc

## تابع ezplot :

**هدف:** رسم کننده توابع.

**املای درست:**

```
ezplot(f)
ezplot(f,[min,max])
ezplot(f,[xmin,xmax,ymin,ymax])
ezplot(x,y)
ezplot(x,y,[tmin,tmax])
ezplot(...,figure)
```

**شرح:** تابع ezplot(f) تابع f=f(x) را روی بازه پیش فرض یعنی $-2\pi < x < 2\pi$ رسم می کند.

تابع ezplot(f,[min,max]) تابع f=f(x) را روی بازه تعیین شده رسم می کند. این تابع یک پنجره جدید با نام **Figure No. 1** باز می کند و نتیجه را در آن نمایش می دهد. اگر از قبل پنجره نموداری باز باشد، آنگاه ezplot نتیجه را در پنجره ای با بزرگترین شماره نمایش می دهد.

تابع ezplot(f,[min,max],fign) پنجره رسم را (در صورت نیاز) باز می کند و نمودار را با برچسب fign در آن رسم می کند.

برای توابع ضمنی به صورت f=f(x,y) نیز داریم:

تابع ezplot(f) عبارت f(x,y)=0 را روی بازه پیش فرض یعنی $-2\pi < x < 2\pi$ و $-2\pi < y < 2\pi$ رسم می کند.

تابع ezplot(f,[xmin,xmax,ymin,ymax]) عبارت *f(x,y)=0* را روی بازه $xmin < x < xmax$ و $ymin < y < ymax$ رسم می کند.

تابع ezplot(f,[min,max]) عبارت *f(x,y)=0* را روی بازه $min < x < max$ و $min < y < max$ رسم می کند.

اگر تابع f برای متغیر های u و v تعریف شده باشد (به جای متغیر های x و y) آنگاه نقاط umin و umax و vmin و vmax در ماتریس ۱ در ۴ بالا به صورت الفبایی مرتب و مقدار دهی می شوند. بنا بر این دستور ezplot(u^2 - v^2 - 1,[-3,2,-2,3]) عبارت $u^2 - v^2 - 1$ را روی بازه $-3 < u < 2$ و $-2 < v < 3$ رسم می کند.

تابع ezplot(x,y) منحنی مسطحی را که بر طبق پارامتر های $x = x(t)$ و $y = y(t)$ تعریف شده است، روی بازه $0 < t < 2\pi$ رسم می کند.

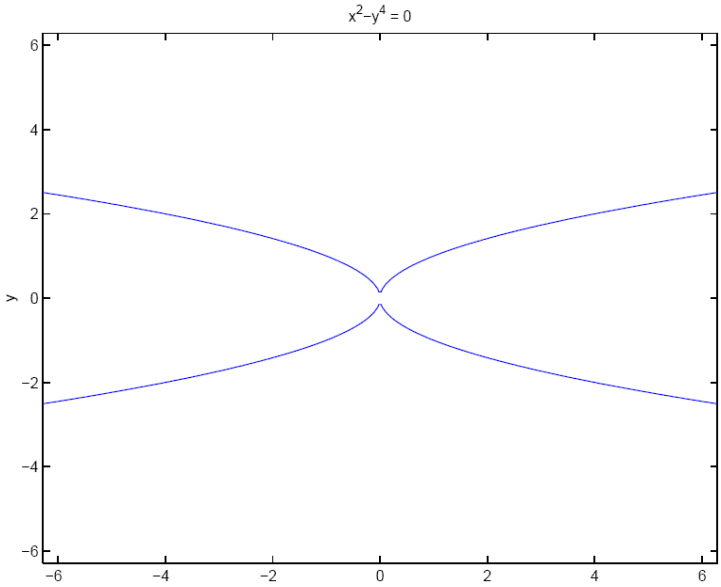
تابع ezplot(x,y,[tmin,tmax]) منحنی مسطحی را که بر طبق پارامتر های $x = x(t)$ و $y = y(t)$ تعریف شده است، روی بازه $tmin < t < tmax$ رسم می کند.

تابع ezplot(...,figure) تابع داده شده را روی بازه مشخص شده در پنجره رسمی که به وسیله figure نام آن مشخص می شود، رسم می کند.

اگر شما بازه ای را برای تابع مشخص نکنید، ezplot تابع شما را روی بازه 2π , 2π- و البته در جاهایی که تابع روی آن تعریف شده باشد رسم می کند. ezplot مقادیر کرانی را در یک بازه حذف می کند و آنها را رسم نخواهد کرد.

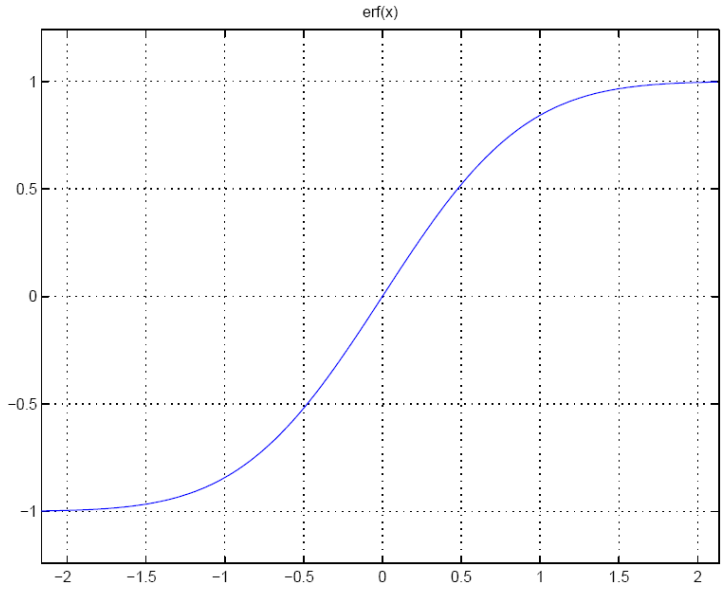
**مثال ها:** این مثال تابع ضمنی روبرو را بر روی بازه 2π , 2π- رسم می کند: $x^2 - y^4 = 0$

```
>>syms x y
>>ezplot(x^2-y^4)
```




```
>>syms x
>>ezplot(erf(x))
>>grid
```

این دستورات نموداری از تابع خطا را رسم خواهند کرد:




**موارد مرتبط:**

ezcontour, ezcontourf, ezmesh, ezmeshc, ezplot3, ezpolar, ezsurf, ezsurfc, plot

## تابع ezplot3 :

**هدف:** رسم منحنی های پارامتریک سه بعدی.

**املای درست:**

ezplot3(x,y,z)

```
ezplot3(x,y,z,[tmin,tmax])
ezplot3(...,'animate')
```

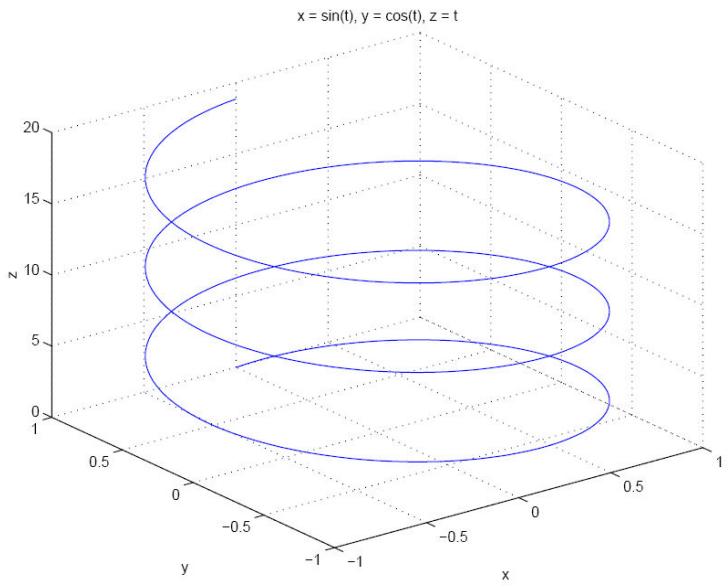
**شرح:** تابع ezplot3(x,y,z) منحنی فضایی با پارامتر های $x = x(t)$ و $y = y(t)$ و $z = z(t)$ را روی بازه $0 < t < 2\pi$ رسم می کند.

تابع ezplot3(x,y,z,[tmin,tmax]) منحنی فضایی با پارامتر های $x = x(t)$ و $y = y(t)$ و $z = z(t)$ را روی بازه $tmin < t < tmax$ رسم می کند.

تابع ezplot3(...,'animate') یک دنباله محدود و قابل قبول از منحنی فضایی را رسم می کند.

**مثال ها:** این مثال یک منحنی فضایی با پارامتر های $x = sin(t)$ و $y = cos(t)$ و $z = t$ را روی بازه $0 < t < 6\pi$ رسم می کند.

```
>>syms t; ezplot3(sin(t), cos(t), t,[0,6*pi])
```




**موارد مرتبط:**

ezcontour, ezcontourf, ezmesh, ezmeshc, ezplot, ezpolar, ezsurf, ezsurfc, plot3

## تابع ezpolar :

**هدف:** رسم با مختصات قطبی.

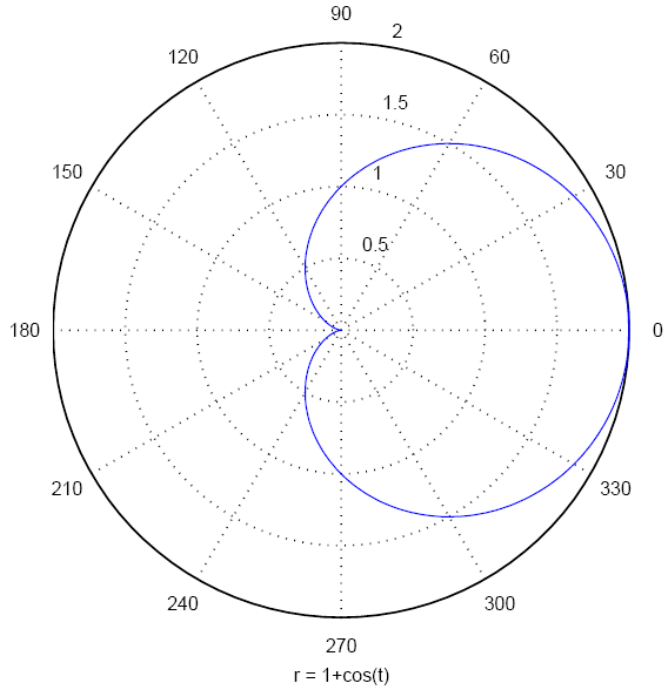
**املای درست:**

```
ezpolar(f)
ezpolar(f,[a,b])
```

**شرح:** تابع ezpolar(f) منحنی قطبی $rho = f(theta)$ را روی بازه پیش فرض یعنی $0 < theta < 2\pi$ رسم می کند.
تابع ezpolar(f,[a,b]) همان تابع را روی بازه $a < theta < b$ رسم می کند.

**مثال ها:** دستورات زیر یک منحنی قطبی از تابع $1 + \cos(t)$ را روی بازه $0 < t < 2\pi$ رسم می کنند.

```
>>syms t
>>ezpolar(1+cos(t))
```




## تابع ezsurf :

**هدف:** رسم رنگی و سه بعدی رویه ها.

**املای درست:**

```
ezsurf(f)
ezsurf(f,domain)
ezsurf(x,y,z)
ezsurf(x,y,z,[smin,smax,tmin,tmax]) or ezsurf(x,y,z,[min,max])
ezsurf(...,n)
ezsurf(...,'circ')
```

**شرح:** تابع ezsurf(f) یک نمودار هندسی از تابع f رسم می کند که در آن f یک عبارت سمبلیک است که تابعی ریاضیاتی از دو متغیر x و y را نشان می دهد.
تابع f روی بازه ی پیش فرض یعنی $-2\pi \le x \le 2\pi$ و $-2\pi \le y \le 2\pi$ رسم می شود. MATLAB طبق بازه های تعیین شده ناحیه را به صورت تور مانند تقسیم بندی می کند و روی گره های این تور مقدار تابع f را محاسبه می

کند و این نقاط را رسم می کند. اگر تابع f بر روی نقاط مورد نظر تعریف نشده باشد، در آن ناحیه منحنی رسم نخواهد شد.

تابع ezsurf(f,domain) نمودار تابع f(x, y) را روی بازه ی domain رسم می کند. domain می تواند به صورت یک ماتریس(بردار) ۱ در ۴ به صورت [xmin, xmax, ymin, ymax] یا یک ماتریس ۱ در ۲ به صورت [min, max] باشد که در ماتریس ۱ در ۲ داریم: min < x < max , min < y < max

اگر تابع f برای متغیر های u و v تعریف شده باشد (به جای متغیر های x و y) آنگاه نقاط umin و umax و vmin و vmax در ماتریس ۱ در ۴ بالا به صورت الفبایی مرتب و مقدار دهی می شوند. بنا بر این دستور ezsurf(u^2 - v^3,[0,1],[3,6]) نمودار تابع $u^2 - v^3$ را روی بازه $0 < u < 1, 3 < v < 6$ رسم می کند.

تابع ezsurf (x,y,z) رویه پارامتریک را با پارامتر های $x = x(s,t)$ و $y = y(s,t)$ و $z = z(s,t)$ روی مربع $-2\pi < s < 2\pi$ و $2\pi < t < 2\pi$ رسم می کند.

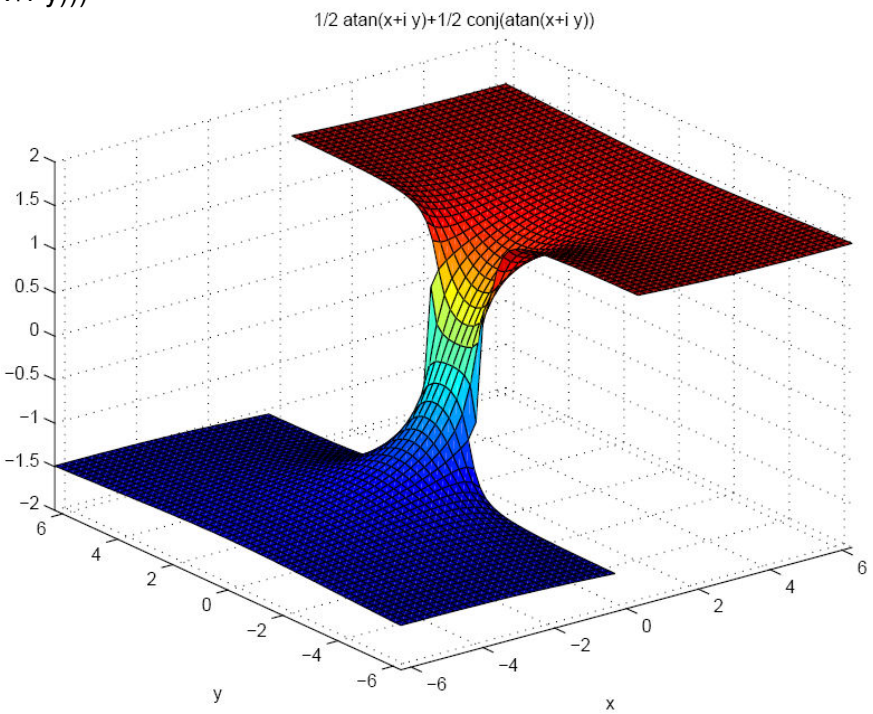
تابع ezsurf(x,y,z,[smin,smax,tmin,tmax]) یا ezsurf(x,y,z,[min,max]) رویه پارامتریک را با استفاده از بازه های تعیین شده رسم می کند.

ezsurf (...,n) تابع f روی بازه پیش فرض و با استفاده از تقسیم بندی تور مانند n در n رسم می کند. مقدار پیش فرض برای n عدد 60 می باشد.

تابع ezsurf(...,'circ') تابع f را روی یک ناحیه دایره ای رسم می کند.

**مثال ها:** تابع ezsurf تابع ریاضیاتی را در محدوده هایی(نقاطی) که تعریف نشده است رسم نمی کند. (MATLAB این نقاط را با کلمه NaN مشخص می کند.) این مثال نحوه فیلتر کردن نقاط تنها و نقاط ناپیوستگی را توسط MATLAB مشخص می کند. در اینجا تابع $f(x, y)= real(atan(x+iy))$ را روی بازه $-2\pi < x < 2\pi$ و $-2\pi < y < 2\pi$ رسم می کند:

```
>>syms x y
>>ezsurf(real(atan(x+i*y)))
```



همچنین به خاطر داشته باشید که ezsurf یک منحنی همراه برچسب های محور ها و عنوان نمودار رسم خواهد کرد.

**موارد مرتبط:**

ezcontour, ezcontourf, ezmesh, ezmeshc, ezplot, ezpolar, ezsurfc, surf

## تابع ezsurfc :

**هدف:** رسم منحنی با ترکیب رسم رنگی سه بعدی و رسم منحنی تراز.

**املای درست:**

```
ezsurfc(f)
ezsurfc(f,domain)
ezsurfc(x,y,z)
ezsurfc(x,y,z,[smin,smax,tmin,tmax]) or ezsurfc(x,y,z,[min,max])
ezsurfc(...,n)
ezsurfc(...,'circ')
```

**شرح:** تابع ezsurfc(f) یک نمودار هندسی از تابع f رسم می کند که در آن f یک عبارت سمبلیک است که تابعی ریاضیاتی از دو متغیر x و y را نشان می دهد.

تابع f روی بازه ی پیش فرض یعنی $-2\pi \le x \le 2\pi$ و $-2\pi \le y \le 2\pi$ رسم می شود. MATLAB طبق بازه های تعیین شده ناحیه را به صورت تور مانند تقسیم بندی می کند و روی گره های این تور مقدار تابع f را محاسبه می کند و این نقاط را رسم می کند. اگر تابع f بر روی نقاط مورد نظر تعریف نشده باشد، در آن ناحیه منحنی رسم نخواهد شد.

تابع ezsurfc(f,domain) نمودار تابع f(x,y) را روی بازه ی domain رسم می کند. domain می تواند به صورت یک ماتریس(بردار) ۱ در ۴ به صورت [xmin, xmax, ymin, ymax] یا یک ماتریس ۱ در ۲ به صورت [min, max] باشد که در ماتریس ۱ در ۲ داریم: $min < x < max$ , $min < y < max$

اگر تابع f برای متغیر های u و v تعریف شده باشد (به جای متغیر های x و y) آنگاه نقاط umin و umax و vmin و vmax در ماتریس ۱ در ۴ بالا به صورت الفبایی مرتب و مقدار دهی می شوند. بنا بر این دستور ezsurfc(u^2 - v^3,[0,1],[3,6]) نمودار تابع $u^2 - v^3$ را روی بازه $0 < u < 1, 3 < v < 6$ رسم می کند.

تابع ezsurfc (x,y,z) رویه پارامتریک را با پارامتر های $x = x(s,t)$ و $y = y(s,t)$ و $z = z(s,t)$ روی مربع $-2\pi < s < 2\pi$ و $2\pi < t < 2\pi$ رسم می کند.

تابع ezsurfc(x,y,z,[smin,smax,tmin,tmax]) یا ezsurfc(x,y,z,[min,max]) رویه پارامتریک را با استفاده از بازه های تعیین شده رسم می کند.

ezsurfc (...,n) تابع f روی بازه پیش فرض و با استفاده از تقسیم بندی تور مانند n در n رسم می کند. مقدار پیش فرض برای n عدد 60 می باشد.

تابع ezsurfc(...,'circ') تابع f را روی یک ناحیه دایره ای رسم می کند.

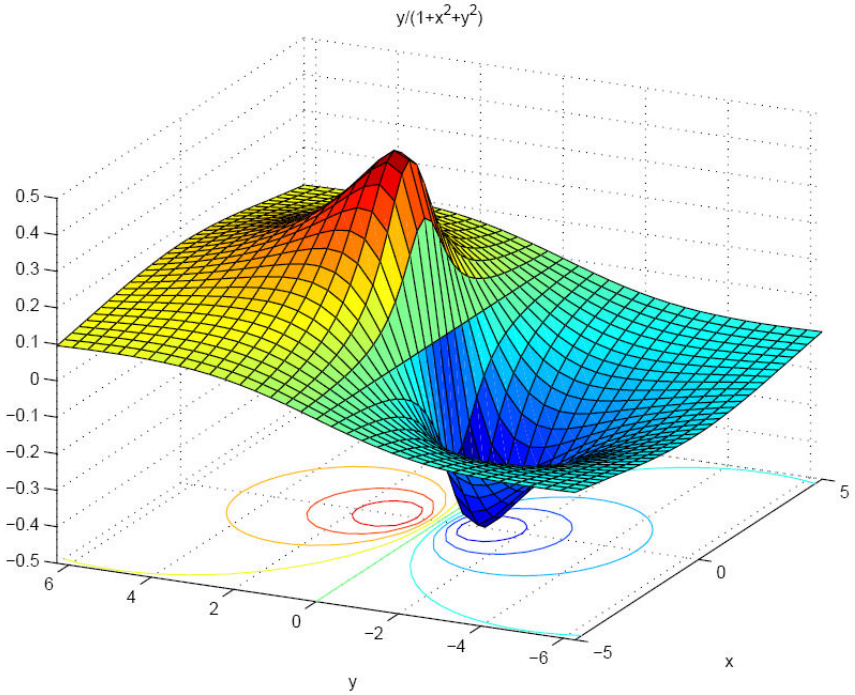
**مثال ها:** رسم منحنی سه بعدی و منحنی تراز عبارت زیر روی بازه $-5 < x < 5$ , $-2*pi < y < 2*pi$

$$f(x,y) = \frac{y}{1+x^2+y^2}$$

دستورات زیر را در نظر بگیرید:

```
>>syms x y
>>ezmeshc(y/(1 + x^2 + y^2),[-5,5,-2*pi,2*pi])
```

به وسیله درگ کردن با ماوس می توانید این نمودار را بچرخانید تا بتوانید خطوط منحنی تراز را بهتر ببینید. در این مثال از زاویه افقی 65- و زاویه فراز 26 استفاده شده است.




**موارد مرتبط:**

ezcontour, ezcontourf, ezmesh, ezmeshc, ezplot, ezpolar, ezsurf, surfc

## تابع factor :

**هدف:** فاکتور گیری و تجزیه اعداد.

**املای درست:**

factor(X)

**شرح:** تابع factor(X) می تواند یک عدد صحیح مثبت، یک آرایه از عبارات سمبلیک یا یک آرایه از مقادیر صحیح مثبت را به عنوان آرگومان ورودی دریافت کند. اگر آرگومان ورودی یک عدد صحیح مثبت باشد تابع factor(X) عوامل اول عدد X را در یک ماتریس سطری به عنوان خروجی برمی گرداند.

اگر S یک ماتریس از چند جمله ای ها یا اعداد صحیح باشد، آنگاه تابع factor(S) روی تک تک درایه های ماتریس عمل می کند. اگر هرکدام از اعدادی که قرار است به عوامل اول تجزیه شوند بیشتر از ۱۶ رقم داشته باشند شما مجبورید آن ها را از نوع سمبلیک تعریف کنید، مانند sym('1234567899876543 21')

**مثال ها:**

```
>>factor(x^3-y^3)
ans=
        (x-y)*(x^2+x*y+y^2)
>>factor([a^2-b^2, a^3+b^3])
ans=
        [(a-b)*(a+b), (a+b)*(a^2-a*b+b^2)]
>>factor(sym('12345678901234567890'))
ans=
        (2)*(3)^2*(5)*(101)*(3803)*(3607)*(27961)*(3541)
```

**موارد مرتبط:**

collect, expand, horner, simplify, simple

## تابع finsym :

**هدف:** پیدا کردن متغیر های سمبلیک در یک ماتریس یا یک عبارت سمبلیک.

**املای درست:**

r = findsym(S)
r = findsym(S,n)

**شرح:** تابع findsym(S) تمام متغیرهای سمبلیک موجود در S را به ترتیب الفبایی و در یک لیست جدا شده توسط کاما برمی گرداند. اگر S شامل هیچ متغیر سمبلیک نباشد این تابع یک رشته خالی برمی گرداند.
تابع findsym(S,n) نزدیک ترین n متغیری که از نظر الفبایی به x نزدیک ترند را برمی گرداند.

**مثال ها:**

```
>>syms a x y z t
>>findsym(sin(pi*t))
ans=
        t
>>findsym(x+i*y-j*z)
ans=
        x, y, z
>>findsym(a+y,1)
ans=
         y
```

**موارد مرتبط:**

compose, diff, int, limit, taylor

## تابع finverse :

**هدف:** بدست آوردن معکوس توابع.

**املای درست:**

```
g = finverse(f)
g = finverse(f,u)
```

**شرح:** دستور g = finverse(f) معکوس تابع f را بر می گرداند. f یک تابع سمبلیک بر حسب متغیر سمبلیک x می باشد. پس از این دستور g یک تابع سمبلیک خواهد بود به طوری که: g(f(x)) = x . در واقع finverse(f) معکوس تابع $f$ یعنی $f^{-1}$ را در صورت وجود بر می گرداند.
g = finverse(f,u) از متغیر سمبلیک u استفاده می کند که متغیر u یک متغیر سمبلیک مستقل است. پس از این دستور g یک تابع سمبلیک خواهد بود به طوری که: g(f(u)) = u . از این دستور موقعی استفاده می شود که تابع f شامل بیشتر از یک متغیر سمبلیک باشد.

**مثال ها:**

```
>>finverse(1/tan(x))
ans=
        atan(1/x)
>>finverse(exp(u-2*v),u)
ans=
        2*v+log(u)
```

**موارد مرتبط:**

compose, syms

## تابع fortran :

**هدف:** مشاهده معادل یک عبارت سمبلیک به زبان فرترن.

**املای درست:**

```
fortran(S)
```

**شرح:** تابع fortran(S) کد فرترن معادل با عبارت S را بر می گرداند.

**مثال ها:**

```
>>syms x
>>f = taylor(log(1+x));
>>fortran(f)
ans=
        t0 = x-x**2/2+x**3/3-x**4/4+x**5/5
>>H = sym(hilb(3));
>>fortran(H)
ans=
        H(1,1) = 1              H(1,2) = 1.E0/2.E0     H(1,3) = 1.E0/3.E0
        H(2,1) = 1.E0/2.E0      H(2,2) = 1.E0/3.E0     H(2,3) = 1.E0/4.E0
        H(3,1) = 1.E0/3.E0      H(3,2) = 1.E0/4.E0     H(3,3) = 1.E0/5.E0
```

**موارد مرتبط:**

ccode, latex, pretty

## تابع fourier :

**هدف:** تبدیل انتگرال فوریه.

**املای درست:**

F = fourier(f)
F = fourier(f,v)
F = fourier(f,u,v)

**شرح:** F = fourier(f) تبدیل فوریه تابع سمبلیک f برحسب متغیر پیش فرض x می باشد. مقدار بازگشت داده شده پیش فرض تابعی بر حسب w می باشد. تبدیل فوریه بر روی یک تابع بر حسب x اعمال می شود و یک تابع بر حسب w برمی گرداند: $f = f(x) \Rightarrow F = F(w)$

اگر f = f(w) آنگاه فوریه تابعی بر حسب t بر می گرداند. $F=F(t)$ که در آن:

$$F(w) = \int_{-\infty}^{\infty} f(x)e^{-iwx}dx$$

که در آن x یک متغیر سمبلیک در تابع f است که به وسیله تابع finsym تعیین شده است.

تابع F = fourier(f,v) تابع برگشت داده شده را بر حسب متغیر v ارائه می دهد (به جای متغیر پیش فرض w) یعنی:

$$F(v) = \int_{-\infty}^{\infty} f(x)e^{-ivx}dx$$

F = fourier(f,u,v) ، f را تابعی از u و F را تابعی از v (به ترتیب به جای متغیر های پیش فرض x و w) قرار می دهد.

$$F(v) = \int_{-\infty}^{\infty} f(u)e^{-ivu}du$$

**مثال ها:**

| Fourier Transform | MATLAB Command |
|---|---|
| $f(x) = e^{-x^2}$<br>$F[f](w) = \int_{-\infty}^{\infty} f(x)e^{-ixw}dx$<br>$= \sqrt{\pi}e^{-w^2/4}$ | `f = exp(-x^2)`<br>`fourier(f)`<br>returns<br>`pi^(1/2)*exp(-1/4*w^2)` |
| $g(w) = e^{-\lvert w\rvert}$<br>$F[g](t) = \int_{-\infty}^{\infty} g(w)e^{-itw}dw$<br>$= \frac{2}{1+t^2}$ | `g = exp(-abs(w))`<br>`fourier(g)`<br>returns<br>`2/(1+t^2)` |
| $f(x) = xe^{-\lvert x\rvert}$<br>$F[f](u) = \int_{-\infty}^{\infty} f(x)e^{-ixu}dx$<br>$= -\frac{4i}{(1+u^2)^{2u}}$ | `f = x*exp(-abs(x))`<br>`fourier(f,u)`<br>returns<br>`-4*i/(1+u^2)^2*u` |

ادامه جدول:

| Fourier Transform | MATLAB Command |
|---|---|
| $f(x,v) = e^{-x^2\lvert v\rvert}\frac{\sin v}{v}, x \text{ real}$<br>$F[f(v)](u) = \int_{-\infty}^{\infty} f(x,v)e^{-ivu}dv$<br>$= -\operatorname{atan}\frac{u-1}{x^2} + \operatorname{atan}\frac{u+1}{x^2}$ | `syms x real`<br>`f = exp(-x^2*abs(v))*sin(v)/v`<br>`fourier(f,v,u)`<br>returns<br>`-atan((u-1)/x^2)+atan((u+1)/x^2)` |

**موارد مرتبط:**

ifourier, laplace, ztrans

## تابع funtool :

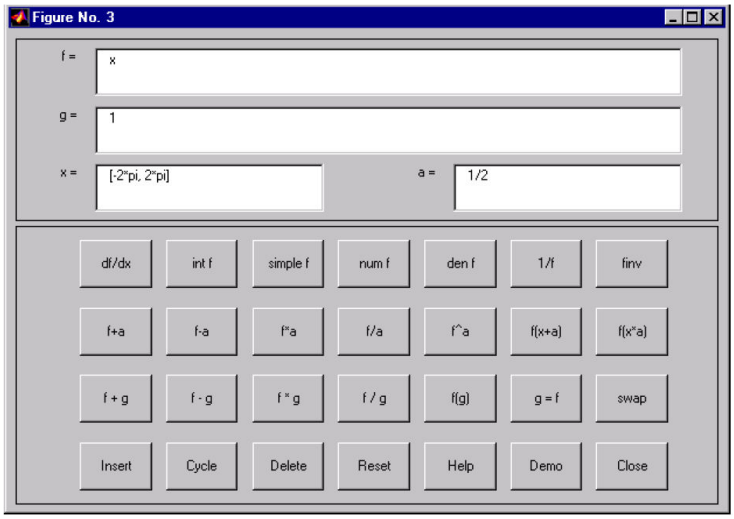
**هدف:** تابع ماشین حساب.

**املای درست:**

funtool

**شرح:** funtool یک ماشین حساب را نمایش می دهد که توابعی را که به یک متغیر وابسته هستند را می تواند نمایش دهد. با کلیک کردن روی یک دکمه (به عنوان مثال funtool)، funtool یک منحنی که جمع، تفریق، حاصل ضرب یا نسبت دو تابعی را نشان می دهد که ما تعیین می کنیم. funtool شامل یک حافظه برای توابع است که به شما این امکان را می دهد که بتوانید توابع را برای بازیابی های بعدی ذخیره کنید.

در ابتدا funtool یک منحنی از جفت توابع f(x) = x و g(x) = 1 نمایش می دهد. منحنی این توابع روی بازه [-2*pi, 2*pi] رسم می شوند. funtool امکاناتی را برای شما فراهم می کند که می توانید توابع f و g را ذخیره، بازیابی، تعریف مجدد، ترکیب و تبدیل کنید.




در بالای پنجره نمایش داده شده چند کادر متنی قابل ویرایش وجود دارد:

کادر f= : یک عبارت سمبلیک را که تابع f را نشان می دهد نمایش می دهد. برای تعریف مجدد f می توانید این کادر را ویرایش کنید.

کادر g= : یک عبارت سمبلیک را که تابع g را نشان می دهد نمایش می دهد. برای تعریف مجدد g می توانید این کادر را ویرایش کنید.

کادر x= : دامنه x که برای رسم تابع از آن استفاده می شود را نشان می دهد. برای تغییر دامنه می توانید دامنه مورد نظر خود را در این کادر وارد کنید.

کادر a= : فاکتور ثابتی را که برای معتدل کردن f از آن استفاده می شود را نشان می دهد (به شرح کلید ها در بخش بعد توجه کنید). این کادر متنی نیز قابل تغییر است.

بازتاب تغییراتی که در این کادر های متنی می دهید بلافاصله توسط funtool رسم می شود.

در پایین پنجره funtool هم کلید هایی وجود دارد که تابع f را می توانند تبدیل کنند یا عملیات دیگری انجام دهند. سطر اول از این کلید ها f را با تبدیل ها مختلفی از خودش با کلید های زیر جایگزین می کند:

کلید **df/dx** : مشتق تابع f.

کلید **int f** : انتگرال تابع f.

کلید **simple f** : شکل ساده شده تابع f، در صورت امکان.

کلید **num f** : صورت تابع f.

کلید **den f** : مخرج تابع f .

کلید **1/f** : عکس تابع f

کلید**finv** : معکوس تابع f.

البته کلید های int f و finv f ممکن است نتوانند انتگرال و معکوس تابع را بدست آورند که این بستگی به تابع دارد.

کلید های سطر دوم نیز بسته به مقدار a که ما در کادر متنی بالا وارد می کنیم به شرح زیر می باشد:

کلید **f+a** : جایگزین کردن f(x) به وسیله f(x) + a

کلید **f-a** : جایگزین کردن f(x) به وسیله f(x) – a

کلید **f*a** : جایگزین کردن f(x) به وسیله f(x) * a

کلید **f/a** : جایگزین کردن f(x) به وسیله f(x) / a

کلید **f^a** : جایگزین کردن f(x) به وسیله f(x) ^ a

کلید **f(x+a)**:جایگزین کردن f(x) به وسیله f(x + a)

کلید **f(x*a)**:جایگزین کردن f(x) به وسیله f(x * a)

چهار کلید اول سطر سوم تابع f را به وسیله ترکیبات خود این تابع جایگزین می کند:

کلید **f+g** : جایگزین کردن f(x) به وسیله f(x) + g(x)

کلید **f-g** : جایگزین کردن f(x) به وسیله f(x) – g(x)

کلید **f*g** : جایگزین کردن f(x) به وسیله f(x) * g(x)

کلید **f/g** : جایگزین کردن f(x) به وسیله f(x) / g(x)

بقیه کلید های سطر سوم توابع f و g را تعویض می کنند:

کلید **g=f** : جایگزین کردن f(x) به وسیله g(x)

کلید **swap** : جایگزین کردن f(x) به وسیله g(x) و g(x) به وسیله f(x)

سه کلید اول سطر چهارم به شما این امکان را می دهد که توابع را از حافظه ماشین حساب بازیابی کنید یا تابعی را در آن ذخیره کنید:

کلید **Insert** : تابع f را به آخر لیست توابع ذخیره شده اضافه می کند.

کلید **Cycle** : تابع f را با تابع بعدی در لیست توابع ذخیره جابجا می کند.

کلید **Delete** : تابع f را از لیست توابع ذخیره شده حذف می کند.

بقیه کلید های سطر چهارم هم مربوط به توابع متفرقه هستند:

کلید **Reset** : ماشین حساب را به حالت اولیه بر می گرداند.

کلید **Help** : یک راهنمای آنلاین برای ماشین حساب نمایش می دهد.

کلید **Demo** : یک دموی کوتاه از ماشین حساب را نمایش می دهد.

کلید **Close** : ماشین حساب را می بندد

**موارد مرتبط:**

ezplot, syms

## تابع horner :

**هدف:** نمایش هارنر چند جمله ای ها.

**املای درست:**

R = horner(P)

**شرح:** فرض کنید که P یک ماتریس از چند جمله ای های سمبلیک است. horner(P) هر یک از عناصر این ماتریس را به شکل تو در توی آن تبدیل می کند. این شکل پس از چند بار فاکتور گیری روی این عناصر حاصل می شود.

**مثال ها:**

```
>>horner(x^3-6*x^2+11*x-6)
ans=
        -6+(11+(-6+x)*x)*x
>>horner([x^2+x;y^3-2*y])
ans=
        [       (1+x)*x         ]
        [       (-2+y^2)*y      ]
```

**موارد مرتبط:**

expand, factor, simple, simplify, syms

## تابع hypergeom :

**هدف:** تعمیم یافته تابع فوق مقیاسات فضایی.

**املای درست:**

hypergeom(n, d, z)

**شرح:** تابع hypergeom(n, d, z) تعمیم یافته تابع فوق مقیاسات فضایی یعنی $F(n, d, z)$ می باشد که به عنوان تابع فوق مقیاسات فضایی تعمیم یافته بارنز شناخته می شود و به صورت ${}_jF_k$ نوشته می شود که در آن j و k عبارتند از: j = length(n) و k = length(d) . برای اسکالر های a ، b و c ، تابع فوق مقیاسات فضایی گاوس می باشد یعنی تابع زیر:

$${}_2F_1(a,b;c;z)$$

تعریف این سری توانی به صورت زیر می باشد:

$$F(n, d, z) = \sum_{k=0}^{\infty} \frac{C_{n,k}}{C_{d,k}} \cdot \frac{z^k}{k!}$$

که در آن:

$$C_{\vartheta,k} = \prod_{j=1}^{|\vartheta|} \frac{\Gamma(v_j+k)}{\Gamma(v_j)}$$

**مثال ها:**

```
>>syms a z
>>hypergeom([],[],z)
ans=
        exp(z)
>>hypergeom(1,[],z)
ans=
        -1/(-1+z)
>>hypergeom(1,2,'z')
ans=
        (exp(z)-1)/z
>>hypergeom([1,2],[2,3],'z')
ans=
        2*(exp(z)-1-z)/z^2
>>hypergeom(a,[],z)
ans=
        (1-z)^(-a)
>>hypergeom([],1,-z^2/4)
ans=
        besselj(0,z)
```

## تابع ifourier :

**هدف:** معکوس تبدیل انتگرال فوریه.

**املای درست:**

```
f = ifourier(F)
f = ifourier(F,u)
f = ifourier(F,v,u)
```

**شرح:** f = ifourier(F) معکوس تبدیل انتگرال فوریه روی اسکالر سمبلیک F با متغیر مستقل پیش فرض W می باشد. مقدار پیش فرض بازگردانده شده یک تابع بر حسب متغیر x می باشد. معکوس تبدیل فوریه بر روی یک تابع از w اعمال شده و یک تابع از x را بر می گرداند.

$F=F(w) \Rightarrow f=f(x)$

اگر F=F(x) باشد، آنگاه ifourier یک تابع از t بر می گرداند. *f=f(t)* با تعریف زیر:

$$f(x) = 1/(2\pi) \int_{-\infty}^{\infty} F(w)e^{iwx}dw$$

f = ifourier(F,u) ، f را به جای اینکه تابعی از x قرار دهد یک تابع از u قرار می دهد.

$$f(u) = 1/(2\pi)\int_{-\infty}^{\infty} F(w)e^{iwu}dw$$

در اینجا u یک اسکالر سمبلیک است.

f = ifourier(F,v,u) ، F را به عنوان تابعی از v و f را به عنوان تابعی از u دریافت می کند (به ترتیب به جای متغیر های w و x).

$$f(u) = 1/(2\pi)\int_{-\infty}^{\infty} F(v)e^{ivu}dv$$

**مثال ها:**

| Inverse Fourier Transform | MATLAB Command |
|---|---|
| $f(w) = e^{w^2/(4a^2)}$ | `syms a real`<br>`f = exp(-w^2/(4*a^2))` |
| $F^{-1}[f](x) = \int_{-\infty}^{\infty} f(w)e^{ixw}dw$ | `F = ifourier(f)`<br>`F = simple(F)` |
| $= \frac{a}{\sqrt{\pi}}e^{-(ax)^2}$ | returns<br>`a*exp(-x^2*a^2)/pi^(1/2)` |

| Inverse Fourier Transform | MATLAB Command |
|---|---|
| $g(x) = e^{-\lvert x\rvert}$ | `g = exp(-abs(x))` |
| $F^{-1}[g](t) = \int_{-\infty}^{\infty} g(x)e^{itx}dx$ | `ifourier(g)` |
| $= \frac{\pi}{1+t^2}$ | returns<br>`1/(1+t^2)/pi` |
| $f(w) = 2e^{-\lvert w\rvert} - 1$ | `f = 2*exp(-abs(w)) - 1` |
| $F^{-1}[f](t) = \int_{-\infty}^{\infty} f(w)e^{itw}dw$ | `simple(ifourier(f,t))` |
| $= \frac{-(-2+\pi\delta(t))}{\frac{(1+t^2)}{\pi}}$ | returns<br>`-(-2+pi*Dirac(t))/(1+t^2)/pi` |

| | |
|---|---|
| $f(w,v) = e^{-w^2\lvert v\rvert}\frac{\sin v}{v}, w \text{ real}$ | `syms w real`<br>`f = exp(-w^2*abs(v))*sin(v)/v` |
| $F^{-1}[f(v)](t) = \int_{-\infty}^{\infty} f(w,v)e^{ivt}dv$ | `ifourier(f,v,t)` |
| $= \frac{1}{2\pi}\left(\operatorname{atan}\frac{t+1}{w^2} - \operatorname{atan}\frac{t-1}{w^2}\right)$ | returns<br>`-1/2*(-atan((t+1)/w^2)`<br>`+atan((-1+t)/w^2))/pi` |

**موارد مرتبط:**

fourier, ilaplace, iztrans

## تابع ilaplace :

**هدف:** معکوس تبدیل لاپلاس.

**املای درست:**

F = ilaplace(L)
F = ilaplace(L,y)
F = ilaplace(L,y,x)

**شرح:** F = ilaplace(L) معکوس تبدیل لاپلاس روی اسکالر سمبلیک L با متغیر مستقل پیش فرض S می باشد. مقدار برگشت داده شده به طور پیش فرض تابعی از t می باشد. معکوس تبدیل لاپلاس روی تابعی از S اعمال می شود و تابعی از t را بر می گرداند.

$$L = L(s) \Rightarrow F = F(t)$$

اگر L=L(t) آنگاه ilaplace تابعی از x را بر می گرداند. F=F(X) با تعریف زیر :

$$F(t) = \int_{c-i\infty}^{c+i\infty} L(s)e^{st}ds$$

که در آن c یک عدد حقیقی انتخاب شده است.

دستور F = ilaplace(L,y) ، F را به جای اینکه تابعی از x قرار دهد، تابعی از y قرار می دهد.

$$F(y) = \int_{c-i\infty}^{c+i\infty} L(y)e^{sy}ds$$

در اینجا y یک اسکالر سمبلیک است.

F = ilaplace(L,y,x) ، F را به عنوان تابعی از x و L را به عنوان تابعی از y دریافت می کند (به ترتیب به جای متغیر های پیش فرض t و s ).

$$F(x) = \int_{c-i\infty}^{c+i\infty} L(y)e^{xy}dy$$

**مثال ها:**

| Inverse Laplace Transform | MATLAB Command |
|---|---|
| $f(s) = \frac{1}{s^2}$ | `f = 1/s^2` |
| $L^{-1}[f] = \frac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty} f(s)e^{st}ds$ | `ilaplace(f)` |
| | returns |
| $= t$ | `t` |
| $g(t) = \frac{1}{(t-a)^2}$ | `g = 1/(t-a)^2` |
| $L^{-1}[g] = \frac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty} g(t)e^{xt}dt$ | `ilaplace(g)` |
| | returns |
| $= xe^{ax}$ | `x*exp(a*x)` |
| $f(u) = \frac{1}{u^2-a^2}$ | `syms x u`<br>`syms a real`<br>`f = 1/(u^2-a^2)` |
| $L^{-1}[f] = \frac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty} g(u)e^{xu}du$ | `simplify(ilaplace(f,x))` |
| | returns |
| $= \frac{\sinh(x\lvert a\rvert)}{\lvert a\rvert}$ | `sinh(x*abs(a))/abs(a)` |

**موارد مرتبط:**

ifourier, iztrans, laplace

## تابع imag :

**هدف:** قسمت موهومی عدد مختلط سمبلیک.

**املای درست:**

imag(Z)

**شرح:** تابع imag(Z) قسمت موهومی عدد متخلط سمبلیک Z را بر می گرداند.

**موارد مرتبط:**

conj, real

## تابع int :

**هدف:** انتگرال گیری.

**املای درست:**

```
R = int(S)
R = int(S,v)
R = int(S,a,b)
R = int(S,v,a,b)
```

**شرح:** int(S) انتگرال نامعین S را نسبت به متغیر سمبلیکی که finsym مشخص می کند بر می گرداند.

int(S,v) انتگرال نامعین S را نسبت به متغیر سمبلیک v بر می گرداند.

int(S,a,b) انتگرال معین S را نسبت به متغیر سمبلیکی که finsym مشخص می کند روی بازه (a,b) بر می گرداند.

int(S,v,a,b) انتگرال معین S را نسبت به متغیر سمبلیک v روی بازه (a,b) بر می گرداند.

**مثال ها:**

```
>>int(-2*x/(1+x^2)^2)
ans=
        1/(1+x^2)
>>int(x/(1+z^2),z)
ans=
        x*atan(z)
>>int(x*log(1+x),0,1)
ans=
        1/4
>>int(2*x, sin(t), 1)
ans=
        1-sin(t)^2
>>int([exp(t),exp(alpha*t)])
ans=
        [exp(t), 1/alpha*exp(alpha*t)]
```

**موارد مرتبط:**

diff, symsum

## تابع inv :

**هدف:** معکوس ماتریس.

**املای درست:**

```
R = inv(A)
```

**شرح:** تابع inv(A) معکوس ماتریس سمبلیک A را بر می گرداند.

**مثال ها:**

```
>>A = sym([2,-1,0;-1,2,-1;0,-1,2]);
>>inv(A)
ans=
        [ 3/4, 1/2, 1/4]
        [ 1/2,    1, 1/2]
        [ 1/4, 1/2, 3/4]
>>syms a b c d
>>A = [a b; c d]
>>inv(A)
ans=
        [  d/(a*d-b*c), -b/(a*d-b*c)]
        [ -c/(a*d-b*c),  a/(a*d-b*c)]
```

حالا فرض کنید که یک M-File به صورت زیر ایجاد کرده اید:

```
%% Generate a symbolic N-by-N Hilbert matrix.
function A = genhilb(N)
        syms t;
        for i = 1: N
                for j = 1: N
                        A(i,j) = 1/(i + j - t);
                end
        end
```

در این صورت دستورات زیر معکوس ماتریس سمبلیک هیلبرت ۲ در ۲ را بر می گرداند:

```
>>inv(genhilb(2))
ans=
        [          -(-3+t)^2*(-2+t),          (-3+t)*(-2+t)*(-4+t)    ]
        [        (-3+t)*(-2+t)*(-4+t),            -(-3+t)^2*(-4+t)    ]
```

**موارد مرتبط:**

vpa (صفحه عملگر های حسابی)

## تابع iztrans :

**هدف:** معکوس تبدیل z.

**املای درست:**

```
f = iztrans(F)
f = iztrans(F,k)
f = iztrans(F,w,k)
```

**شرح:** f = iztrans(F) ، معکوس تبدیل z اسکالر سمبلیک F با متغیر مستقل پیش فرض Z می باشد. مقدار پیش فرض برگشت داده شده تابعی از n می باشد.

$$f(n) = \frac{1}{2\pi i} \oint_{|z|=R} F(z) z^{n-1} dz, n = 1, 2, \ldots$$

که در آن R یک عدد صحیح مثبت انتخاب می شود، به این ترتیب تابع $F(z)$ یک تابع قابل حل به طریق جبری و بیرون از دایره $|z| = R$ قرار دارد.

اگر F=F(n) باشد، آنگاه iztrans تابعی از k را بر می گرداند. $f = f(k)$

f = iztrans(F,w) ، f را به جای اینکه تابعی از n قرار دهد تابعی از k قرار می دهد. در اینجا k یک اسکالر سمبلیک است.

f = iztrans(F,w,k) ، F را به عنوان تابعی از w دریافت می کند (به جای متغیر پیش فرض findsym(F) ) و تابعی از k را بر می گرداند.

$$F = F(w) \Rightarrow f = f(k)$$

**مثال ها:**

| **Inverse Z-Transform** | **MATLAB Operation** |
|---|---|
| $f(z) = \frac{2z}{(z-2)^2}$ | `f = 2*z/(z-2)^2` |
| $Z^{-1}[f] = \frac{1}{2\pi i} \oint_{\lvert z\rvert=R} f(s) z^{n-1} dz$ | `iztrans(f)` |
| | returns |
| $= n2^n$ | 2^n*n |
| **Inverse Z-Transform** | **MATLAB Operation** |
| $g(n) = \frac{n(n+1)}{n^2+2n+1}$ | `g = n*(n+1)/(n^2+2*n+1)` |
| $Z^{-1}[g] = \frac{1}{2\pi i} \oint_{\lvert n\rvert=R} g(n) n^{k-1} dn$ | `iztrans(g)` |
| | returns |
| $= -1^k$ | `(-1)^k` |
| $f(z) = \frac{z}{z-a}$ | `f = z/(z-a)` |
| $Z^{-1}[f] = \frac{1}{2\pi i} \oint_{\lvert z\rvert=R} f(z) z^{k-1} dz$ | `iztrans(f,k)` |
| | returns |
| $= a^k$ | `a^k` |

**موارد مرتبط:**

ifourier, ilaplace, ztrans

## تابع jacobian:

**هدف:** ماتریس ژاکوبین.

**املای درست:**

R = jacobian(w,v)

**شرح:** تابع jacobian(w,v) ژاکوبین w را نسبت به v محاسبه می کند که در آن w یک عبارت اسکالر سمبلیک یا یک بردار ستونی سمبلیک است. v یک بردار سطری سمبلیک است. (i,j)امین عنصر از نتیجه برابر است با $\frac{\partial w(i)}{\partial v(j)}$.

**مثال ها:**

```
>>w = [x*y*z; y; x+z];
>>v = [x,y,z];
>>R = jacobian(w,v) , b = jacobian(x+z, v)
R =
        [y*z, x*z, x*y]
        [  0,    1,    0]
        [  1,    0,    1]
b =
        [  1,    0,    1]
```

**موارد مرتبط:**

diff

## تابع jordan:

**هدف:** صورت متعارفی جردن.

**املای درست:**

J = jordan(A)
[V,J] = Jordan(A)

**شرح:** تابع jordan(A) صورت متعارفی جردن (نرمال) A را بر می گرداند که در آن A یک ماتریس سمبلیک یا عددی است. ماتریس باید به دقت شناسایی شود. بنا بر این، اعضای آن باید یا عدد صحیح یا نسبت دو عدد صحیح کوچک باشند. هر خطایی در ماتریس ورودی ممکن است به طور کلی نتایج صورت متعارفی جردن را تغییر دهد.

[V,J] = Jordan(A) هم J را که صورت متعارفی جردن است محاسبه می کند و هم V را که تشابه تبدیل می باشد و ستون های آن تعمیم یافته بردار های ویژه می باشند. علاوه بر این : V\A*V=J

**مثال ها:**

```
>>A = [1 -3 -2; -1 1 -1; 2 4 5] , [V,J] = jordan(A)
A =
        1        -3        -2
       -1         1        -1
        2         4         5
V =
       -1        -1         1
        0        -1         0
        1         2         0
J =
        3         0         0
        0         2         1
        0         0         2
>>V\A*V
ans =
        3         0         0
        0         2         1
        0         0         2
```

**موارد مرتبط:**

eig, poly

## تابع lambertw :

**هدف:** تابع لمبرت w .

**املای درست:**

Y = lambertw(X)

**شرح:** Y = lambertw(X) تابع لمبرت W را برای مقادیر X به عنوان یک ماتریس عددی یا یک ماتریس سمبلیک محاسبه می کند. تابع لمبرت W معادله زیر را حل می کند که در آن W تابعی از x است. $we^{w} = x$

**مثال ها:**

```
>>lambertw([0 -exp(-1); pi 1])
        [ 0          -1.0000 ]
        [ 1.0737   0.5671 ]
>>syms x y
>>lambertw([0 x;1 y])
        [                0, lambertw(x)]
        [ lambertw(1), lambertw(y)]
```

**مراجع:**

[1] Corless, R.M, Gonnet, G.H., Hare, D.E.G., and Jeffrey, D.J., *Lambert's W Function in Maple*, Technical Report, Dept. of Applied Math., Univ. of Western Ontario, London, Ontario, Canada.
[2] Corless, R.M, Gonnet, G.H., Hare, D.E.G., and Jeffrey, D.J., *On Lambert's W Function*, Technical Report, Dept. of Applied Math., Univ. of Western Ontario, London, Ontario, Canada.

## تابع laplace :

**هدف:** تبدیل لاپلاس.

**املای درست:**

laplace(F)
laplace(F,t)
fourier(F,w,z)

**شرح:** L= laplace(F) تبدیل لاپلاس را برای نماد اسکالر F با متغیر مستقل پیش فرض t حساب می کند. خروجی پیش فرض تابعی از s خواهد بود. تبدیل لاپلاس به تابعی از t اعمال می شود و تابعی از s را برمی گرداند.

$$F = F(t) \Longrightarrow L = L(s)$$

اگر $F = F(s)$ باشد آنگاه تبدیل لاپلاس تابعی از t را بر می گرداند، طبق تعریف

$$L(s) = \int_0^\infty F(t)e^{-st}dt$$

که t در آن یک متغیر سمبلیک است که می توان آن را با findsym تعیین کرد.

L=laplace(F,t) به جای متغیر پیش فرض s، L را تابعی از t قرار می دهد

$$L(t) = \int_0^\infty F(x)e^{-tx}dx$$

که در اینجا L یک نماد اسکالر خواهد بود.

L=fourier(F,w,z) به جای متغیر های پیش فرض s و t ، L را تابعی از z ، و F را تابعی از w قرار می دهد:

$$L(z) = \int_0^\infty F(w)e^{-zw}dw$$

**مثال ها:**

| دستور MATLAB | تبدیل لاپلاس |
|---|---|

| | |
|---|---|
| $f(t) = t^4$<br>$L[f] = \int_0^\infty f(t)e^{-ts}dt$<br>$= \frac{24}{s^5}$ | f = t^4<br>laplace(f)<br>خروجی :<br>24/s^5 |
| $g(s) = \frac{1}{\sqrt{s}}$<br>$L[g](t) = \int_0^\infty g(s)e^{-st}ds$<br>$= \sqrt{\frac{\pi}{t}}$ | g = 1/sqrt(s)<br>laplace(g)<br>خروجی :<br>(pi/t)^(1/2) |
| $f(t) = e^{-at}$<br>$L[f](x) = \int_0^\infty f(t)e^{-tx}dx$<br>$= \frac{1}{x+a}$ | ff = exp(−a ∗ t)<br>laplace(f, x)<br>خروجی :<br>1/(x + a) |

**موارد مرتبط:**

fourier, ilaplace, ztrans

## تابع latex :

**هدف:** نمایش LaTeX عبارت سمبلیک.

**املای درست:**

latex(S)

**شرح:** latex(S) نمایش LaTeX عبارت سمبلیک s را بر می گرداند.

**مثال ها:**

```
>>syms x
>>f = taylor(log(1+x));
>>latex(f)
ans=
        x-1/2\,{x}^{2}+1/3\,{x}^{3}-1/4\,{x}^{4}+1/5\,{x}^{5}
>>H = sym(hilb(3));
>>latex(H)
ans=
        \left[ \begin {array}{ccc} 1&1/2&1/3\\\noalign{\medskip}1/2&1/3&1/4\\\noalign{\medskip}1/3&1/4&1/5\end
        {array} \right]
>>syms alpha t
>>A = [alpha t alpha*t];
>>latex(A)
ans=
        \left [\begin {array}{ccc} \alpha&t&\alpha\,t\end {array}\right ]
```

**موارد مرتبط:**

pretty, ccode, fortran

## تابع limit :

**هدف:** حد یک عبارت سمبلیک.

**املای درست:**

limit(F,x,a)
limit(F,a)
limit(F)
limit(F,x,a,'right')
limit(F,x,a,'left')

**شرح:** limit(F,x,a) حد عبارت سمبلیک F را وقتی که x به سمت a میل می کند، محاسبه می کند.
limit(F,a) از finsym(F) برای پیدا کردن متغیر مستقل استفاده می کند.
limit(F) حد F را وقتی که متغیر مستقل به سمت صفر میل می کند، محاسبه می کند.
از limit(F,x,a,'right') برای محاسبه حد راست و از limit(F,x,a,'left') برای محاسبه حد چپ استفاده می کنیم.

**مثال ها:**

```
>>syms x a t h;
>>limit(sin(x)/x)
ans=
        1
>>limit(1/x,x,0,'right')
ans=
        inf
>>limit(1/x,x,0,'left')
ans=
        -inf
>>limit((sin(x+h)-sin(x))/h,h,0)
ans=
        cos(x)
>>v = [(1 + a/x)^x, exp(-x)];
>>limit(v,x,inf,'left')
ans=
        [exp(a), 0]
```

**موارد مرتبط:**

pretty, ccode, fortran

## تابع maple :

**هدف:** دسترسی به هسته maple.

**املای درست:**

```
r = maple('statement')
r = maple('function',arg1,arg2,...)
[r, status] = maple(...)
maple('traceon') or maple trace on
maple('traceoff') or maple trace off
```

**شرح:** r = maple('statement') ، statement را به هسته maple ارسال می کند و نتیجه را بر می گرداند. اگر در جایی نیاز به استفاده ازنقطه-ویرگول در دستور maple بود باید آن را به statemet اضافه کنیم.

r = maple('function',arg1,arg2,...) توابع maple را با هر تعداد آرگومان فراخوانی می کند. اگر لازم باشد آرگومان ها به عبارت سمبلیک تبدیل می شوند. اگر آرگومان sym باشد maple نیز sym بر می گرداند در غیر این صورت خروجی از نوع کلاس char خواهد بود.

[r, status] = maple(...) یک ویژگی هشدار-خطا است که در صورت موفق آمیز بودن اجرا r را برابر با نتیجه اجرا و status را برابر صفر قرار می دهد. در غیر این صورت r یک خطای مرتبط با اجرا و status برابر با عددی مثبت خواهد بود.

maple('traceon') باعث می شود تمام زیر مجموعه ها، عبارات و نتیجه maple چاپ شود. maple('traceoff') باعث غیر فعال شدن این ویژگی می شود.

**مثال ها:**

هریک از عبارات زیر عدد  را تا ۱۰۰ رقم اعشار محاسبه می کند:

```
maple('evalf(Pi,100)')
maple evalf Pi 100
maple('evalf','Pi',100)
```

و یا

```
>>[result,status] = maple('BesselK',4.3)
result =
        Error, (in BesselK) expecting 2 arguments, got 1
status =
        2
>>syms x
>>v = [x^2-1;x^2-4]
>>maple traceon % or maple trace on
>>w = factor(v)
v =
        [ x^2-1]
        [ x^2-4]
>>map(ifactor,array([[x^2-1],[x^2-4]]));
ans=
        Error, (in ifactor) invalid arguments
>>map(factor,array([[x^2-1],[x^2-4]]));

        matrix([[(x-1)*(x+1)], [(x-2)*(x+2)]])
w =
        [ (x-1)*(x+1)]
        [ (x-2)*(x+2)]
```

مثال فوق نشان می دهد که دستور factor ابتدا factor را برای کلاس integer از maple فراخوانی می کند (ifactor) تا تعیین کند که آیا عبارت داده شده به اعداد صحیح قابل تفکیک هست یا نه. در صورت خطا جعبه ابزار ریاضیات سمبلیک  عبارت را به صورت سمبلیک برای maple فراخوانی می کند  (factor).

**موارد مرتبط:**

mhelp, procread

## تابع mapleinit :

**هدف:** مقدار دهی اولیه هسته maple.

**املای درست:**

mapleinit

**شرح:** mapleinit برای تعیین مسیری که کتابخانه های Maple وجود دارند به کار می رود.همچنین برای بار گذاری کتابخانه های جبر خطی و تبدیلات انتگرال و مقدار دهی ارقام به کار می رود.

mapleinit را نباید مستقیماً فراخوانی کرد بلکه باید از طریق maplemex فراخوانی شود. کاربر می تواند با ویرایش m–فایل mapleinit میسر فعلی کتابخانه های Maple را تغییر دهد.برای انجام این کار باید متغیر initstring را تغییر داد.

یونیکس: فرض کنید شما یک نسخه از کتابخانه های Maple را در مسیر '/usr/local/Maple/lib' دارید، می توانید mapleinit را به صورت زیر تغییر دهید:

maplelib = '/usr/local/Maple/lib'

سپس آن نسخه کپی از کتابخانه های Maple را که در MATLAB وجود دارد حذف کنید.

ویندوز: فرض کنید شما یک نسخه از کتابخانه های Maple را در مسیر 'C:\MAPLE\LIB' دارید، می توانید mapleinit را به صورت زیر تغییر دهید:

maplelib = 'C:\MAPLE\LIB'

سپس آن نسخه کپی از کتابخانه های Maple را که در MATLAB وجود دارد حذف کنید.

## تابع mfun :

**هدف:** محاسبه مقدار عددی توابع ریاضی Maple.

**املای درست:**

Y = mfun('function',par1,par2,par3,par4)

**شرح:** mfun('function',par1,par2,par3,par4) برای محاسبه عددی توابع شناخته شده Maple به کار می رود. هر آرگومان par معادل مقدار عددی آرگومان تابع maple می باشد. تا چهار آرگومان را می توان تنظیم کرد. آخرین پارامتر مشخص شده می تواند یک ماتریس باشد، ابعاد تمامی پارامتر های دیگر به تابع maple مورد نظر بستگی دارد. برای به دست آوردن اطلاعات در مورد آرگومان های توابع MATLAB می توانید از دستورات زیر استفاده کنید:

```
help mfunlist
mhelp function
```

دقت محاسبه تابع فوق تا ۱۶ رقم می باشد. هر عنصر جواب از نوع کلاس های عددی MATLAB خواهد بود. یکتایی در توابع با NaN نشان داده می شود.

**مثال ها:**

```
>>mfun('FresnelC',0:5)
ans=
        [ 0 0.7799 0.4883 0.6057 0.4984 0.5636 ]
>>mfun('Chi',[3*i 0])
ans=
        [ 0.1196 + 1.5708i NaN ]
```

**موارد مرتبط:**

mfunlist, mhelp

## تابع mfunlist :

**هدف:** توابعی را که با mfun قابل محاسبه هستند را لیست می کند.

**املای درست:**

mfunlist

**شرح:** mfunlist توابعی را که با mfun قابل محاسبه هستند را لیست می کند(در جدول زیر آمده اند).برای جزئیات بیشتر می توانید از دستور زیر استفاده کنید:

```
>>mhelp function
```

**محدودیت:**

معمولاً دقت محاسبات تا نزدیک شدن به ریشه آن کاهش می یابد، مخصوصاً هنگامی که آرگومان تابع نسبتاً بزرگ باشد. زمان اجرا نیز به آرگومان تابع بستگی دارد اما در کل از محاسبات استاندارد MATLAB کند تر است.

**موارد مرتبط:**

mfun, mhelp

**مراجع:**

[1] Abramowitz, M. and Stegun, I.A., *Handbook of Mathematical Functions*, Dover Publications, 1965.

**قرارداد ها:**

**جدول توابع mfun:** به خاطر داشته باشید که x,y آرگومان های حقیقی، z,z1,z2 آرگومان های مختلط و m,n اعداد صحیح هستند.

| نام تابع | توصیف | نام تابع mfun | آرگومان |
|---|---|---|---|
| چند جمله ای ها و اعداد برنولی | $\frac{e^{xt}}{e^t-1}=\sum_{n=0}^{\infty}B_n(x).\frac{t^{n-1}}{n!}$ | bernoulli(n)<br>bernoulli(n,t) | $n \geq 0$<br>$0<\lvert t\rvert<2\pi$ |
| توابع بسل | Bessell, BesselJ<br>توابع بسل نوع اول و<br>BesselK, BesselY<br>توابع بسل نوع دوم هستند | BesselJ(v,x)<br>BesselY(v,x)<br>Bessell(v,x)<br>BesselK(v,x) | vعددی حقیقی است |
| توابع بتا | $B(x,y)=\frac{\Gamma(x).\Gamma(y)}{\Gamma(x+y)}$ | Beta(x,y) | |
| ضرایب دو جمله ای | $\binom{m}{n}=\frac{m!}{n!(m-n)!}$<br>$=\frac{\Gamma(m+1)}{\Gamma(n+1)\Gamma(m-n+1)}$ | binomial(m,n) | |
| انتگرال بیضوی کامل | انتگرال بیضوی کامل Legendre نوع اول و دوم و سوم | EllipticK(k)<br>EllipticE(k)<br>EllipticPi(a,k) | a و k حقیقی هستند<br>$-\infty<a<\infty$<br>$0<k<1$ |
| انتگرال بیضوی کامل با ضریب مکمل | انتگرال بیضوی کامل با ضرایب مکمل نوع اول و دوم وسوم | EllipticCK(k)<br>EllipticCE(k)<br>EllipticCPi(a,k) | a و k حقیقی هستند<br>$-\infty<a<\infty$<br>$0<k<1$ |
| توابع مکمل خطا و انتگرال های تکراری | $erfc(z)=\frac{2}{\sqrt{\pi}}\int_z^{\infty}e^{-t^2}dt$<br>$=1-\mathrm{erf}(z)$<br>$erfc(-1,z)=\frac{2}{\sqrt{\pi}}e^{-z^2}$<br>$erfc(n,z)=\int_z^{\infty}\mathrm{erfc}(n-1,z)dt$ | erfc(z)<br>erfc(n,z) | $n>0$ |

| | | | |
|---|---|---|---|
| انتگرال دایسون | $F(x) = e^{-x^2}\int_0^x e^{-t^2}\,dt$ | dawson(x) | |
| تابع دی گاما (پسی) | $\Psi(x) = \frac{d}{dx}\ln(\Gamma(x)) = \frac{\Gamma'(x)}{\Gamma(x)}$ | Psi(x) | |
| انتگرال دی لگاریتمی | $f(x) = \int_1^x \frac{\ln(t)}{1-t}dt$ | dilog(x) | $x > 1$ |
| تابع خطا | $\mathrm{erf}(z) = \frac{2}{\sqrt{\pi}}\int_0^z e^{-t^2}\,dt$ | erf(z) | |
| اعداد اویلری و چند جمله ای | $\frac{1}{ch(t)} = \sum_{n=0}^{\infty} E_n \frac{t^n}{n!}$ | euler(n)<br>euler(n,z) | $n \geq 0$<br>$\lvert t\rvert < \frac{\pi}{2}$ |
| انتگرال نمایی | $Ei(n,z) = \int_1^{\infty} \frac{e^{-zt}}{t^n}dt$<br>$Ei(x) = PV - \int_{-\infty}^{x} \frac{e^t}{t}dt$ | Ei(n,z)<br>Ei(x) | $n \geq 0$<br>$Real(z) > 0$ |
| انتگرال سینوسی و کسینوسی فرسنل | $C(x) = \int_0^x \cos\left(\frac{\pi}{2}t^2\right)dt$<br>$S(x) = \int_0^x \sin\left(\frac{\pi}{2}t^2\right)dt$ | FresnelC(x)<br>FresnelS(x) | |
| تابع گاما | $\Gamma(z) = \int_0^{\infty} t^{z-1}e^{-t}dt$ | GAMMA(z) | |

| | | | |
|---|---|---|---|
| تابع هارمونیک | $h(n) = \sum_{k=1}^{n} \frac{1}{k} = \Psi(\mathrm{n}+1) + \boldsymbol{\gamma}$ | harmonic(n) | $n > 0$ |
| انتگرال هذلولی سینوسی و کسینوسی | $Shi(z) = \int_0^z \frac{\sinh(t)}{t} dt$<br>$Chi(z) = \gamma + \ln(z) + \int_0^z \frac{\cosh(t) - 1}{t} dt$ | Shi(z)<br>Chi(z) | |
| توابع فوق هندسی | $F(n,d,z) = \sum_{k=0}^{\infty} \frac{\prod_{i=1}^{j} \frac{\Gamma(n_i+k)}{\Gamma(n_i)} z^k}{\prod_{i=1}^{m} \frac{\Gamma(d_i+k)}{\Gamma(d_i)} k!}$ | hypergeom(n,d,x)<br>که در آن:<br>n = [n1,n2,...]<br>d = [d1,d2,...] | n1,n2,... اعداد حقیقی<br>d1,d2,... اعداد حقیقی نامنفی |
| انتگرال بیضوی ناقص | انتگرال بیضوی ناقص Legendre نوع اول و دوم و سوم | EllipticF(x,k)<br>EllipticE(x,k)<br>EllipticPi(x,a,k) | $0 < x \leq \infty$<br>a حقیقی<br>$-\infty < a < \infty$<br>$k$ حقیقی<br>$0 < k < 1$ |
| تابع گامای ناقص | $\Gamma(a,z) = \int_z^{\infty} e^{-t} t^{a-1} dt$ | GAMMA(z1,z2) | |
| لگاریتم تابع گاما | $ln\, \Gamma(z) = \ln(\Gamma(z))$ | lnGAMMA(z) | |
| انتگرال لگاریتمی | $Li(x) = PV\left\{\int_0^x \frac{dt}{ln\, t}\right\} = Ei(\ln x)$ | Li(x) | $x > 1$ |
| تابع پلی گاما | $\Psi^{(n)}(z) = \frac{d^n}{dz}\Psi(z)$ | Psi(n,z) | $n \geq 0$ |
| انتگرال سینوسی شیفت داده شده | $Ssi(z) = Si(z) - \frac{\pi}{2}$ | Ssi(z) | |

**چند جمله ای های متعامد:**

برای کار با توابع زیر نیاز به بسته نرم افزاری توابع متعامد Maple دارید. که تنها با جعبه ابزار ریاضیات سمبلیک تعمیم داده شده همراه هستند. قبل از استفاده از توابع باید بسته فوق را با دستور زیر مقدار دهی اولیه کنید:

```
maple('with','orthopoly')
```

**جدول توابع چند جمله ای متعامد:**

نکته : در تمام موارد زیر n عدد صحیح نا منفی و x عددی حقیقی است.

| چند جمله ای | نام تابع maple | آرگومان |
|---|---|---|
| Gegenbauer | G(n,a,x) | a یک عبارت جبری نا گویا یا یک عدد گویای بزرگتر از 1/2- است |
| هرمیت | H(n,x) | |
| Laguerre | L(n,x) | |
| Generalized Laguerre | L(n,a,x) | a یک عبارت جبری نا گویا یا یک عدد گویای بزرگتر از 1- است |
| Legendre | P(n,x) | |
| ژاکوبی | P(n,a,b,x) | aوb عبارات جبری نا گویا یا اعداد گویای بزرگتر از 1- هستند |
| چبیشف نوع اول و دوم | T(n,x)<br>U(n,x) | |

## تابع mhelp :

**هدف:** دسترسی به راهنمای Maple.

**املای درست:**

```
mhelp topic
mhelp('topic')
```

**شرح:** از mhelp topic یا mhelp('topic') برای دسترسی به راهنمای maple استفاده می کنیم که topic در آن موضوع مورد نظر است.

**مثال ها:** هر دو دستور زیر راهنمای آنلاین Maple را در مورد BesselI ارائه می دهند.

```
>>mhelp Bessell
>>mhelp('Bessell')
```

**موارد مرتبط:**

maple

## تابع null :

**هدف:** پایه های فضاهای پوچ.

**املای درست:**

Z = null(A)

**شرح:** ستون های z=null(A) پایه های فضای پوچ A را تشکیل می دهند.

size(z,2) بیانگر پوچی A است.

A*z برابر صفر خواهد بود.

اگر A رتبه پر داشته باشد z خالی خواهد بود.

**مثال ها:**

```
>>A = sym(magic(4));
>>Z = null(A)
>>A*Z
ans=
        [ -1]
        [ -3]
        [ 3 ]
        [ 1 ]
        [ 0 ]
        [ 0 ]
        [ 0 ]
        [ 0 ]
```

**موارد مرتبط:**

colspace, rank, rref, svd , عملگرهای ریاضی

## تابع numden :

**هدف:** صورت و مخرج.

**املای درست:**

[N,D] = numden(A)

**شرح:** [N,D] = numden(A) اعضای A را به شکل کسری تبدیل می کند، به نحوی که صورت و مخرج کسر چند جمله ای هایی با ضرایب عددی هستند.

A یک ماتریس عددی یا سمبلیک است. N و D ماتریس های سمبلیکی هستند که به ترتیب نشان دهنده صورت و مخرج کسر هستند.

**مثال ها:**

```
[n,d] = numden(sym(4/5))
n =
        4
d =
        5
>>[n,d] = numden(x/y + y/x)
n =
        x^2+y^2
d =
        y*x
>>A = [a, 1/b] , [n,d] = numden(A)
A =
        [a, 1/b]
n =
        [a,   1]
d =
        [1,  b]
```

## تابع poly :

**هدف:** چند جمله ای مشخصه یک ماتریس.

**املای درست:**

```
p = poly(A)
p = poly(A, v)
```

**شرح:** اگر A یک آرایه عددی باشد poly(A) ضرایب چند جمله ای مشخصه A را بر می گرداند. اگر A سمبلیک باشد poly(A) چند جمله ای مشخصه A را با متغیر پیش فرض x بر می گرداند. متغیر v می تواند به آرگومان دوم تعیین شود.

توجه کنید اگر A عددی باشد دستور poly(sym(A)) تقریباً نزدیک به poly2sym(poly(A)) می باشد. این تقریب به دلیل خطای گرد کردن است.

**مثال ها:**

```
>>syms z
>>A = gallery(3)
>>p = poly(A) , q = poly(sym(A)) , s = poly(sym(A),z)
```

```
A =
        [ -149 -50   -154 ]
        [  537  180  546  ]
        [ -27   -9    -25    ]
p =
        [ 1.0000 -6.0000 11.0000 -6.0000 ]
q=
        [ x^3-6*x^2+11*x-6 ]
s =
        [ z^3-6*z^2+11*z-6 ]
```

**موارد مرتبط:**

poly2sym, jordan, eig, solve

## تابع poly2sym :

**هدف:** تبدیل بردار ضرایب یک چند جمله ای به چند جمله ای سمبلیک.

**املای درست:**

r = poly2sym(c)
r = poly2sym(c, v)

**شرح:** r = poly2sym(c) یک چند جمله ای سمبلیک بر می گرداند که ضرایب آن در بردار عددی c قرار دارند. متغیر سمبلیک پیش فرض x است. متغیر v می تواند به عنوان آرگومان دوم تعیین شود.اگر c=[c1 c2 c3 .... cn] باشد آنگاه r=poly2sym(c) به صورت زیر خواهد بود:

$$c_1x^{n-1} + c_2x^{n-2} + \cdots + c_n$$

poly2sym از روش پیش فرض sym برای تبدیل ضرایب عددی به مقادیر سمبلیک استفاده می کند. در این روش ضرایب سمبلیک نسبت هایی تقریبی از اعداد صحیح هستند، اگر sym بتواند نسبت ساده ای را برای تقریب عدد پیدا کند از آن برای نمایش استفاده می کند در غیر این صورت آن را به صورت مضربی از توان ۲ نشان می دهد.

اگر x دارای مقدار عددی باشد و sym بتواند نسبت دقیقی برای اعداد c پیدا کند آنگاه eval(poly2sym(c)) و polyval(c,x) مقادیری مساوی بر می گردانند.

**مثال ها:**

```
>>poly2sym([1 3 2])
ans=
        x^2 + 3*x + 2
>>poly2sym([.694228, .333, 6.2832])
ans=
        6253049924220329/9007199254740992*x^2+333/1000*x+3927/625
>>poly2sym([1 0 1 -1 2], y)
ans=
        y^4+y^2-y+2
```

**موارد مرتبط:**

sym, sym2poly

## تابع pretty :

**هدف:** چاپ خوش نمای عبارت سمبلیک.

**املای درست:**

```
pretty(S)
pretty(S,n)
```

**شرح:** عبارت سمبلیک را به صورت حروفچینی شده در خروجی چاپ می کند.
pretty(S) عبارت سمبلیک را با پهنای خط پیش فرض ۷۹ چاپ می کند.با pretty(S,n) می توان پهنای خط را از ۷۹ به n تغییر داد.

**مثال ها:**

```
>>A = sym(pascal(2)) , B = eig(A) , pretty(B)
A =
        [ 1 , 1 ]
        [ 1 , 2 ]
B =
        [ 3/2+1/2*5^(1/2) ]
        [ 3/2-1/2*5^(1/2)  ]
ans=
        [                 1/2  ]
        [         3/2 + 1/2 5  ]
        [                      ]
        [                 1/2  ]
        [ 3/2 - 1/2 5          ]
```

## تابع procread :

**هدف:** نصب پروسه های Maple.

**املای درست:**

```
procread('filename')
```

**شرح:** procread('filename') فایل مشخصی که شامل متن منبع پروسه Maple می باشد را می خواند، توضیحات وکاراکتر های خط جدید را حذف می کند و نتیجه را به صورت یک رشته به Maple ارسال می کند. البته جعبه ابزار ریاضیات سمبلیک تعمیم داده شده را باید نصب کرده باشید.

**مثال ها:**

فرض کنید فایل ident.src شامل کد منبع برای یک پروسه maple باشد:

```
ident := proc(A)
#       ident(A) computes A*inverse(A)
        local X;
        X := inverse(A);
        evalm(A &* X);
end;
```

آنگاه دستور procread('ident.src') پروسه فوق را نصب می کند. بعداً می توان با دستوراتی مثل maple('ident',magic(3)) یا maple('ident',vpa(magic(3))) از آن استفاده کرد.

**موارد مرتبط:**

maple

## تابع rank :

**هدف:** مرتبه ماتریس سمبلیک.

**املای درست:**

rank(A)

**شرح:** rank(A) مرتبه ماتریس سمبلیک A را محاسبه می کند. مرتبه ماتریس برابر با تعداد ردیف ها یا ستون های خطی مستقل است.

**مثال ها:**

```
>>rank([a b;c d])
ans=
        2
>>rank(sym(magic(4)))
ans=
        3
```

## تابع real :

**هدف:** بخش حقیقی سمبلیک.

**املای درست:**

real(Z)

**شرح:** real(Z) بخش حقیقی عبارت سمبلیک Z را مشخص می کند.

**موارد مرتبط:**

conj, imag

## تابع rref :

**هدف:** شکل کاهش یافته پلکانی سطری یک ماتریس.

**املای درست:**

rref(A)

**شرح:** rref(A) شکل کاهش یافته پلکانی سطری ماتریس A را بر می گرداند.

**مثال ها:**

```
>>rref(sym(magic(4)))
ans=
        [ 1, 0, 0,  1 ]
        [ 0, 1, 0,  3 ]
        [ 0, 0, 1, -3 ]
        [ 0, 0, 0,  0 ]
```

## تابع rsums :

**هدف:** محاسبه مجموع ریمان.

**املای درست:**

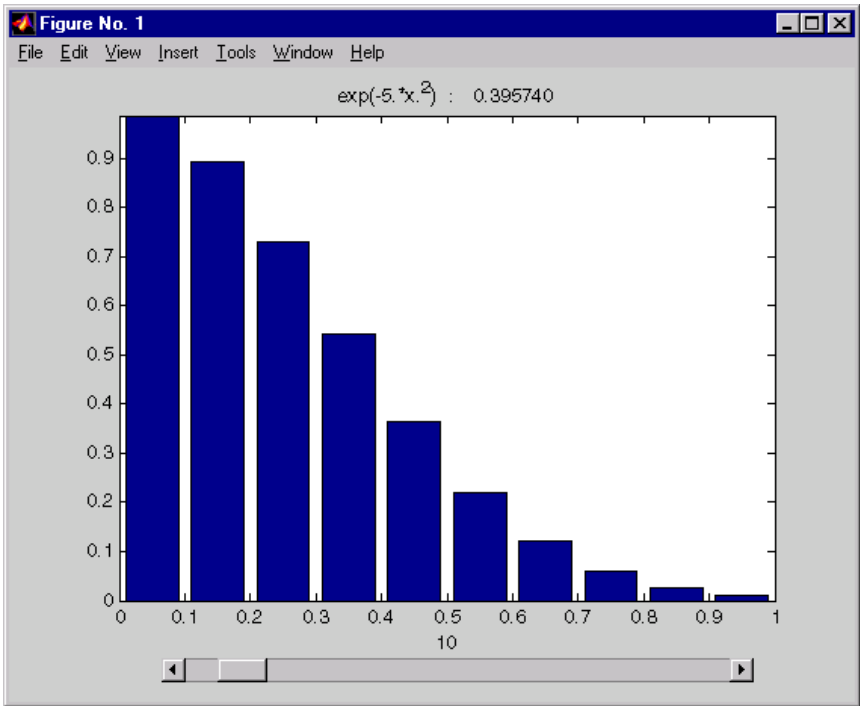
rsums(f)

**شرح:** rsums(f) انتگرال f(x) را با استفاده از مجموع ریمان به صورت تقریبی محاسبه می کند.

rsums(f) نتیجه را به صورت گرافیکی نشان می دهد، شما می توانید تعداد جملات مجموع ریمان را با استفاده از کشوی لغزنده تنظیم کنید. تعداد جملات می توان از ۲ تا ۱۲۸ باشد.

**مثال ها:**

دستور rsums exp(-5*x^2) نمودار زیر را رسم می کند.




## تابع simple :

**هدف:** شکل ساده تر عبارت سمبلیک.

**املای درست:**

```
r = simple(S)
[r,how] = simple(S)
```

**شرح:** simple(S) چندین روش ساده سازی جبری را بر روی عبارت سمبلیک S آزمایش می کند تا عبارتی با طول کمتر را پیدا کند، سپس کوتاه ترین عبارت را به عنوان جواب برمی گرداند.اگر S یک ماتریس باشد جواب نهایی شکل ساده تر کل ماتریس خواهد بود و این بدان معنی است که الزاماً تک تک عناصر ماتریس در شکل ساده ترشان نمایش داده نمی شوند. اگر خروجی ای داده نشود simple(S) تمام نمایش های ممکن را نشان می دهد و کوتاه ترین را بر می گرداند.
[r,how] = simple(S) ساده سازی میانی را نشان نمی دهد اما کوتاه ترین جواب را بر می گرداند، همچنین یک رشته که چگونگی ساده سازی را بیان می کند برگشت داده می شود.r یک sym و how یک رشته است.

**مثال ها:**

| عبارت | فرم ساده شده | روش ساده سازی |
|---|---|---|
| cos(x)^2 + sin(x)^2 | 1 | simplify |
| 2 * cos(x)^2 − sin(x)^2 | 3 * cos(x)^2 − 1 | simplify |
| cos(x)^2 − sin(x)^2 | cos(2 * x) | combine(trig) |
| cos(x) + (−sin(x)^2)^(1/2) | cos(x) + i * sin(x) | radsimp |
| cos(x) + i * sin(x) | exp(i * x) | convert(exp) |
| (x + 1) * x * (x − 1) | x^3 − x | collect(x) |
| x^3 + 3 * x^2 + 3 * x + 1 | (x + 1)^3 | factor |
| cos(3 * acos(x)) | 4 * x^3 − 3 * x | expand |

**موارد مرتبط:**

collect, expand, factor, horner, simplify

## تابع simplify :

**هدف:** ساده سازی سمبلیک.

**املای درست:**

R = simplify(S)

**شرح:** simplify(S) هر کدام از اعضای ماتریس سمبلیک S را با استفاده از قوانین maple ساده می کند.

**مثال ها:**

```
>>simplify(sin(x)^2 + cos(x)^2)
ans=
        1
>>simplify(exp(c*log(sqrt(a+b))))
```

```
ans=
        (a+b)^(1/2*c)
>>S = [(x^2+5*x+6)/(x+2),sqrt(16)];
>>R = simplify(S)
ans=
        R = [x+3,4]
```

**موارد مرتبط:**

collect, expand, factor, horner, simple

## تابع sinint :

**هدف:** تابع انتگرال سینوسی.

**املای درست:**

Y = sinint(X)

**شرح:** sinint(X) انتگرال سینوسی نقاط X را محاسبه می کند. X یک ماتریس عددی یا سمبلیک است. نتیجه یک ماتریس عددی است. تعریف تابع انتگرال سینوسی به صورت زیر است:

$$\mathrm{Si}(x) = \int_0^x \frac{\sin(t)}{t}\,dt$$

**مثال ها:**

```
>>sinint([pi 0;-2.2 exp(3)])
ans=
        [  1.8519 ,        0 ]
        [ -1.6876 , 1.5522 ]
>>sinint(1.2)
ans=
        1.1080
>>diff(sinint(x))
ans=
        sin(x)/x
```

**موارد مرتبط:**

cosint

## تابع size :

**هدف:** ابعاد ماتریس سمبلیک.

**املای درست:**

```
d = size(A)
[m,n] = size(A)
d= size(A, n)
```

**شرح:** فرض کنید A یک ماتریس سمبلیک یا عددی m در n باشد. عبارت d = size(A) یک بردار عددی با دوعنصر بر می گرداند که d=[m,n].

عبارت [m,n] = size(A) دو خروجی را به دو عدد مجزای m و n نسبت می دهد.

عبارت d = size(A,n) طول بعد n ام را بر می گرداند. برای مثال size(A,1) تعداد ردیف های A و size(A,2) تعداد ستون های A را بر می گرداند.

**مثال ها:**

```
>>syms a b c d
>>A = [a b c ; a b d; d c b; c b a];
>>d = size(A) , r = size(A, 2)
d =
        4 3
r =
        3
```

**موارد مرتبط:**

length, ndims in the online MATLAB Function Reference

## تابع solve :

**هدف:** حل سمبلیک تساوی های جبری.

**املای درست:**

```
g = solve(eq)
g = solve(eq,var)
g = solve(eq1,eq2,...,eqn)
g = solve(eq1,eq2,...,eqn,var1,var2,...,varn)
```

**شرح:** عبارت یا تساوی منفرد: ورودی solve می تواند یک عبارت یا یک رشته باشد اگر eq یک عبارت (x^2-2*x+1) یا یک رشته باشد ('x^2-2*x+1') که شامل علامت تساوی نباشد آنگاه solve(eq) معادله eq=0 را به ازای متغیر پیش فرض (به وسیله findsym تعیین می شود) حل می کند.

solve(eq,var) معادله را به ازای متغیر var حل می کند.

دستگاه معادلات: در این حالت ورودی به صورت مجموعه ای از چند عبارت یا چند رشته است که بیانگر یک دستگاه می باشند. solve(eq1,eq2,...,eqn) دستگاهی شامل عبارات eq1 ,eq2,...,eqn را به ازای n متغیری که findsym تعیین می کند، حل می کند.

سه نوع متفاوت خروجی ممکن است به وجود بیاید:برای یک تساوی و یک خروجی نتیجه با چند جواب برای تساوی غیر خطی برگشت داده می شود.برای یک دستگاه معادلات با تعداد مساوی خروجی نتایج به صورت الفبایی مرتب می شوند.برای یک دستگاه معادلات با یک خروجی یک ساختار که شامل جواب است برگشت داده می شود.

برای هر دو نوع تساوی (منفرد یا دستگاه) در صورتی که جواب سمبلیک پیدا نشود جواب به صورت عددی برگشت داده می شود.

**مثال ها:**

```
>>solve('a*x^2 + b*x + c')
ans=
        [  1/2/a*(-b+(b^2-4*a*c)^(1/2)) , 1/2/a*(-b-(b^2-4*a*c)^(1/2)) ]
>>solve('a*x^2 + b*x + c','b')
ans=
        -(a*x^2+c)/x
>>S = solve('x + y = 1','x - 11*y = 5')

S.y =
         -1/3
S.x =
         4/3
>>A = solve('a*u^2 + v^2', 'u - v = 1', 'a^2 - 5*a + 6')
A =
        a: [4x1 sym]
        u: [4x1 sym]
        v: [4x1 sym]

A.a =
        [ 2 ]
        [ 2 ]
        [ 3 ]
        [ 3 ]
A.u =
        [ 1/3+1/3*i*2^(1/2) ]
        [ 1/3-1/3*i*2^(1/2)  ]
        [ 1/4+1/4*i*3^(1/2) ]
        [ 1/4-1/4*i*3^(1/2)  ]
A.v =
        [ -2/3+1/3*i*2^(1/2) ]
        [ -2/3-1/3*i*2^(1/2)  ]
        [ -3/4+1/4*i*3^(1/2) ]
        [ -3/4-1/4*i*3^(1/2)  ]
```

**موارد مرتبط:**

arithmetic operators, dsolve, findsym

## تابع subexpr :

**هدف:** چاپ عبارت سمبلیک به صورت جملاتی با زیر عبارات مشترک.

**املای درست:**

```
[Y,SIGMA] = subexpr(X,SIGMA)
[Y,SIGMA] = subexpr(X,'SIGMA')
```

**شرح:** [Y,SIGMA] = subexpr(X,SIGMA) یا [Y,SIGMA] = subexpr(X,'SIGMA') عبارت سمبلیک X را به صورت جملاتی با زیر عبارات مشترک چاپ می کند. زیر عباراتی به صورت %1,%2 و... به وسیله pretty(S) چاپ می شوند.

**مثال ها:** دستور زیر عبارت t را به صورت جملاتی با زیر عبارات مشترک در r باز نویسی می کند:

```
>>t = solve('a*x^3+b*x^2+c*x+d = 0');
>>[r,s] = subexpr(t,'s');
```

**موارد مرتبط:**

pretty, simple, subs

## تابع subs :

**هدف:** جانشینی سمبلیک در یک عبارت یا ماتریس سمبلیک.

**املای درست:**

```
R = subs(S)
R = subs(S,old,new)
```

**شرح:** subs(S) تمام متغیر ها در عبارت سمبلیک S را با مقادیری که از فراخوانی تابع به دست آمده اند یا متغیر های فضای کاری MATLAB جایگزین می کند.

subs(S,old,new) تمام عبارت های old موجود در عبارت سمبلیک S را با new جایگزین می کند. old می تواند یک متغیر سمبلیک یا یک رشته نشان دهنده متغیر باشد.new می تواند یک متغیر یا عبارت سمبلیک یا عددی باشد.

اگر old و new آرایه های سلولی با اندازه های مساوی باشند هر عضو old با یک عضو متناظر در new جایگزین می شود. اگر S و old اسکالر باشند و new یک آرایه یا آرایه سلولی باشد، اسکالر ها بسط پیدا می کنند تا یک جواب آرایه ای ایجاد کنند. اگر new یک آرایه سلولی از ماتریس های عددی باشد جایگزینی به صورت عنصر به عنصر صورت می گیرد. (یعنی عبارتی به صورت subs(x*y,{x,y},{A,B}) وقتی A و B عددی هستند A.*B را برمی گرداند).

اگر subs(s,old,new) عبارت s را تغییر ندهد آنگاه subs(s,new,old) اجرا می شود.این امر باعث می شود نیاز به دانستن ترتیب آرگومان ها را از بین می برد.

subs(s,old,new,0) در صورت عدم تغییر s آرگومان ها را جا به جا نمی کند.

**مثال ها:** یک ورودی: فرض کنید a=980 و C1=3 در فضای کاری MATLAB وجود دارند.

```
>>y = dsolve('Dy = -a*y')
y =
        C1*exp(-a*t)
>>subs(y)
ans =
        3*exp(-980*t)
```

جانشینی تکی:

```
>>subs(a+b,a,4)
ans=
        4+b.
```

جانشینی چند گانه:

```
>>subs(cos(a)+sin(b),{a,b},{sym('alpha'),2})
ans=
        cos(alpha)+sin(2)
```

بسط اسکالر:

```
>>subs(exp(a*t),'a',-magic(2))
ans=
        [ exp(-t), exp(-3*t)]
        [ exp(-4*t), exp(-2*t)]
```

بسط اسکالر چندگانه:

```
>>subs(x*y,{x,y},{[0 1;-1 0],[1 -1;-2 1]})
ans=
        [0 ,-1]
        [2 , 0]
```

**موارد مرتبط:**

simplify, subexpr

## تابع svd :

**هدف:** تجزیه سمبلیک مقدار ویژه.

**املای درست:**

```
sigma = svd(A)
sigma = svd(vpa(A))
[U,S,V] = svd(A)
[U,S,V] = svd(vpa(A))
```

**شرح:** sigma = svd(A) یک بردار سمبلیک است که شامل مقادیر ویژه ماتریس سمبلیک A است.

sigma = svd(vpa(A)) مقادیر ویژه را با استفاده محاسبات با دقت متغیر متغیر، به صورت عددی محاسبه می کند.

[U,S,V] = svd(A) و [U,S,V] = svd(vpa(A)) ماتریس های پیرو U و V را برمی گرداند. ستون های U و V بردار های ویژه هستند. S ماتریس قطری هم بعد با A است که که عناصر قطر آن نامنفی هستند و به صورت نزولی چیده شده اند.

داریم:

$$A = U * S * V'$$

نکته: بردار های ویژه سمبلیک قابل دستیابی نیستند.

**مثال ها:**

```
>>digits(3)
>>A = sym(magic(4));
>>svd(A)
>>svd(vpa(A))
>>[U,S,V] = svd(A)
ans=
        [          0    ]
        [          34   ]
        [ 2*5^(1/2) ]
        [ 8*5^(1/2) ]
ans=
        [ .311e-6*I ]
        [     4.47    ]
        [     17.9    ]
        [     34.1    ]
U =
        [ -.500,  .671,   .500, -.224  ]
        [ -.500, -.224,  -.500, -.671  ]
        [ -.500,  .224,  -.500,  .671  ]
        [ -.500, -.671,   .500,  .224  ]
S =
        [  34.0,      0,       0,       0 ]
        [     0,   17.9,       0,          0 ]
        [     0,       0,  4.47,          0 ]
        [     0,       0,       0, .835e-15 ]
V =
        [ -.500, . 500,  .671, -.224  ]
        [ -.500, -.500, -.224, -.671  ]
        [ -.500, -.500,  .224,  .671  ]
        [ -.500,  .500, -.671,  .224  ]
```

**موارد مرتبط:**

digits, eig, vpa

## تابع sym :

**هدف:** ایجاد اعداد، متغیر ها و اشیا سمبلیک.

**املای درست:**

```
S = sym(A)
x = sym('x')
x = sym('x','real')
x = sym('x','unreal')
S = sym(A,flag)
```

که flag در آن یکی از کارکتر های 'r', 'd', 'e', 'f' است

**شرح:** S = sym(A) شی S از کلاس sym را بر اساس A ایجاد می کند. اگر ورودی رشته باشد خروجی یک متغیر یا عدد سمبلیک است. اگر ورودی یک اسکالر یا ماتریس عددی باشد خورجی نمایش سمبلیک ورودی خواهد بود.

x = sym('x') متغیر سمبلیک با نام 'x' را به وجود می آورد و نتیجه را در x ذخیره می کند. x = sym('x','real') متغیر x را به صورت real ایجاد می کند و این یعنی conj(x)=x.

x = sym('x','unreal') متغیر x را کاملاً قراردادی ایجاد بدون هیچ ویژگی اضافی ایجاد می کند (تنها تضمین می کند که x از نوع real نیست).

alpha = sym('alpha') و r = sym('Rho','real') تنها دو مثال بیشتر هستند.

دستوری مانند pi = sym('pi') و delta = sym('1/10') عددی سمبلیک ایجاد می کنند که تقریب ممیز شناور را به ارث نمی برند. متغیر pi که به این روش ایجاد می شود موقتاً در توابع توکار با نامی یکسان جایگزین می شود.

S = sym(A,flag) اسکالر یا ماتریس عددی A را به شکل سمبلیک آن تبدیل می کند. شیوه تبدیل اعداد ممیز شناور به فرم سمبلیک توسط آرگومان دوم تعیین می شود که این آرگومان می تواند یکی از مقادیر 'f','r','e' و یا 'd' باشد. مقدار پیش فرض این آرگومان 'r' است.

'f' نمایش متغیر سمبلیک به صورت ممیز شناور است. تمام متغیر ها در این حالت به یکی از دو صورت زیر نمایش داده می شوند:

'1.F'*2^(e)' یا '-1.F'*2^(e)'.

که F در آن یک رشته با ۱۳ رقم مبنای ۱۶ و e یک عدد صحیح است. این روش مقدار ممیز شناور را به صورت دقیق ذخیره می کند اما ممکن است برای دستکاری های بعدی مناسب نباشد. برای مثال sym(1/10,'f') معادل '1.999999999999a'*2^(-4) خواهد بود زیرا نمی توان 1/10 را به صورت دقیق در دستگاه ممیز شناور نشان داد.

'r' نمایش گویای متغیر سمبلیک است. اعداد ممیز شناوری که به وسیله دستوراتی به صورت p/q, p*pi/q, sqrt(p), 2^q و 10^q برای اعداد نسبتاً کوچک p و q به وجود می آیند به شکل سمبلیک تبدیل می شوند. این عمل به شیوه ای موثر خطای گرد کردن در محاسبات را از بین می برد، ولی باز هم ممکن است اعداد شناور را به صورت دقیق نمایش ندهد اگر هیچ تقرب دقیقی برای عبارتی به صورت p*2^q برای اعداد بزرگ p و q یافت نشود مجدداً از فرم ممیز شناور استفاده می شود. برای مثال معادل sym(4/3,'r') عبارت '4/3' است ولی معادل sym(1+sqrt(5),'r') عبارت 7286977268806824*2^(-51) می باشد.

'e' نمایش متغیر سمبلیک به همراه خطای تخمینی است. این حالت تکمیل شده حالت 'r' به همراه متغیر 'eps' است ،که میزان تفاوت بین مقدار تئوری و ممیز شناور را نشان می دهد. برای مثال sym(3*pi/4,'e') معادل 3*pi/4- 103*eps/249 خواهد بود.

'd' نمایش دهدهی متغیر سمبلیک است.تعداد ارقام در این روش براساس تنظیمات digits می باشد که به وسیله vpa استفاده شده است. برای کمتر از ۱۶ رقم دقت ممکن است کاهش بیابد اما برای بیشتر از آن تضمینی نیست. برای مثال

فرض کنید تنظیمات قبلی به صورت digits(10) باشد آنگاه sym(4/3,'d') معادل 1.333333333 خواهد بود. حال اگر فرض کنیم تنظیمات قبلی به صورت digits(20) باشد آنگاه sym(4/3,'d') معادل 1.3333333333333332593 خواهد بود، همان طور که مشاهده می کنید این نتیجه رشته ای نیست که تنها به 3 ختم شود اما در نمایش دهدهی دقیق برای اعداد ممیز شناور نزدیک ترین جواب است.

**موارد مرتبط:**

digits, double, syms

## تابع syms :

**هدف:** روشی سریع برای ایجاد اشیا سمبلیک.

**املای درست:**

```
syms arg1 arg2 ...
syms arg1 arg2 ... real
syms arg1 arg2 ... unreal
```

**شرح:** syms arg1 arg2 ... شکل ساده شده عبارات زیر است:

```
arg1 = sym('arg1');
arg2 = sym('arg2'); ...
```

syms arg1 arg2 ... real شکل ساده شده عبارات زیر است:

```
arg1 = sym('arg1','real');
arg2 = sym('arg2','real'); ...
```

syms arg1 arg2 ... unreal شکل ساده شده عبارات زیر است:

```
arg1 = sym('arg1','unreal');
arg2 = sym('arg2','unreal'); ...
```

اسم متغیر ها حتماً باید با حروف اافبای انگلیسی شورع شود و تنها می تواند از حروف انگلیسی و عدد تشکیل شود.

**مثال ها:**

```
>>syms x beta real  % is equivalent to >>x = sym('x','real');
>>beta = sym('beta','real');
```

برای از بین بردن حالت real برای اشیای x و beta از دستور زیر استفاده می کنیم:

```
>>syms x beta unreal
```

نکته:توجه کنید که دستور clear x حالت real را از x حذف نمی کند، برای اینکار باید از دستورات syms x unreal، clear mex یا clear all استفاده کنید که دو دستور آخر موجب بار گذاری مجدد هسته Maple در فضای کاری MATLAB می شود که غیر کارا بودن و زمان بر بودن این دستورات را نشان می دهد.

**موارد مرتبط:**

sym

## تابع sym2poly :

**هدف:** تبدیل چند جمله ای سمبلیک به بردار ضرایب عددی.

**املای درست:**

```
c = sym2poly(s)
```

**شرح:** sym2poly یک بردار عددی ردیفی بر می گرداند که اعضای آن ضرایب عددی چند جمله ای سمبلیک هستند.این اعداد به ترتیب نزولی توان چند جمله ای چیده شده اند.یعنی اولین عدد ضریب جمله ای با توان بیشتر و آخرین عدد ضریب جمله ای با توان کمتر است.

**مثال ها:**

```
>>syms x u v;
>>sym2poly(x^3 - 2*x - 5)
ans=
        [ 1 0 -2 -5 ]
>>sym2poly(u^4 - 3 + 5*u^2)
ans=
        [ 1 0 5 0 -3 ]
>>sym2poly(sin(pi/6)*v + exp(1)*v^2)
ans=
        [ 2.7183  0.5000  0 ]
```

**موارد مرتبط:**

poly2sym
polyval in the online MATLAB Function Reference

## تابع symsum :

**هدف:** مجموع سمبلیک.

**املای درست:**

```
r = symsum(s)
r = symsum(s,v)
r = symsum(s,a,b)
r = symsum(s,v,a,b)
```

**شرح:** symsum(s) مجموع عبارت s را به ازای متغیر k که توسط findsym تعیین می شود،از 0 تا k-1 به صورت سمبلیک محاسبه می کند.

symsum(s,v) مجموع عبارت s را به ازای متغیر v از 0 تا v-1 به صورت سمبلیک محاسبه می کند.

symsum(s,a,b) و symsum(s,v,a,b) مجموع عبارت s را از a تا b محاسبه می کنند.

**مثال ها:**

```
>>syms k n x
>>symsum(k^2)
ans=
        1/3*k^3-1/2*k^2+1/6*k
>>symsum(k)
ans=
        1/2*k^2-1/2*k
>>symsum(sin(k*pi)/k,0,n)
ans=
         -1/2*sin(k*(n+1))/k+1/2*sin(k)/k/(cos(k)-1)*cos(k*(n+1))-
                                        1/2*sin(k)/k/(cos(k)-1)
>>symsum(k^2,0,10)
ans=
        385
>>symsum(x^k/sym('k!'), k, 0,inf)
ans=
        exp(x)
```

نکته :توجه کنید که در مثال آخر از sym برای تولید شکل سمبلیک فاکتوریل k! استفاده کردیم، علامت ! عملگری برای ایجاد فاکتوریل در MATLAB نیست.

**موارد مرتبط:**

findsym, int, syms

## تابع taylor :

**هدف:** بسط سری تیلور.

**املای درست:**

```
r = taylor(f)
r = taylor(f,n,v)
r = taylor(f,n,v,a)
```

**شرح:** taylor(f,n,v) ترتیب (n-1) تقریب چند جمله ای مک لورن برای f را بر می گرداند.که در آن f یک عبارت سمبلیک نشان دهنده تابع و v متغیر مستقل است که می تواند یک رشته یا متغیر سمبلیک باشد.

taylor(f,n,v,a) تقریب بسط سری تیلور را حول نقطه a بر می گرداند. آرگومان a می تواند یک مقدار عددی، یک مقدار سمبلیک یا رشته ای باشد که نشان دهنده متغیر عددی باشد.

شما می توانید آرگومان های n,v و a را با هر ترتیبی برای taylor بفرستید، خود تابع آرگومان ها را از روی نوع و موقعیتشان تشخیص می دهد.
همچنین شما می توانید هر کدام از آرگومان های n,v و a را به دلخواه حذف کنید. اگر v را حذف کنید taylor از findsym برای پیدا کردن متغیر مستقل استفاده می کند. مقدار پیش فرض n نیز ۶ می باشد.
سری تیلور برای هر تابع تحلیلی f(x) به صورت زیر تعریف می شود:

$$f(x) = \sum_{n=0}^{\infty} (x-a)^n . \frac{f^{(n)}(a)}{n!}$$

**مثال ها:** جدول زیر کاربرد های مختلفی از taylor و رابطه بین سری تیلور و مک لورن را نشان می دهد.

| دستور MATLAB | عملیات ریاضی |
|---|---|
| $\gg syms\ x$<br>$\gg taylor(f)$ | $\sum_{n=0}^{5} x^n \frac{f^{(n)}(0)}{n!}$ |
| $\gg \mathrm{taylor(f, m)}$<br>m عدد صحیح مثبت | $\sum_{n=0}^{m} x^n \frac{f^{(n)}(0)}{n!}$ |
| $\gg \mathrm{taylor(f, a)}$<br>a عددی حقیقی | $\sum_{n=0}^{5} (x-a)^n \frac{f^{(n)}(a)}{n!}$ |
| $\gg \mathrm{taylor(f, m1, m2)}$<br>m1 و m2 اعداد صحیح مثبت هستند. | $\sum_{n=0}^{m1} (x-m2)^n \frac{f^{(n)}(m2)}{n!}$ |
| $\gg \mathrm{taylor(f, m, a)}$<br>a عددی حقیقی و m عدد صحیح مثبت | $\sum_{n=0}^{m} (x-a)^n \frac{f^{(n)}(a)}{n!}$ |

اگر f تابعی با دو یا بیشتر متغیر باشد (f=f(x,y,...)) پارامتر چهارمی وجود دارد که به شما اجازه می دهد متغیر را برای بسط تیلور انتخاب کنید.

| عملیات ریاضی | دستور MATLAB |
|---|---|
| $\sum_{n=0}^{5} \frac{y^n}{n!} \cdot \frac{\partial^n}{\partial y^n} f(x, y=0)$ | ≫ taylor(f, y) |
| $\sum_{n=0}^{m} \frac{y^n}{n!} \cdot \frac{\partial^n}{\partial y^n} f(x, y=0)$ | ≫ taylor(f, y, m)<br>یا<br>≫ taylor(f, m, y)<br>m عدد صحیح مثبت |
| $\sum_{n=0}^{m} \frac{(y-a)^n}{n!} \cdot \frac{\partial^n}{\partial y^n} f(x, y=a)$ | ≫ taylor(f, m, y, a)<br>a عددی حقیق و m عدد صحیح مثبت |
| $\sum_{n=0}^{5} \frac{(y-a)^n}{n!} \cdot \frac{\partial^n}{\partial y^n} f(x, y=a)$ | ≫ taylor(f, y, a)<br>a عددی حقیقی است. |

**موارد مرتبط:**

findsym

## تابع taylortool :

**هدف:** ماشین حساب مجموع تیلور.

**املای درست:**

```
taylortool
taylortool('f')
```

**شرح:** taylortool یک برنامه گرافیکی را اجرا می کند که نشان دهنده مجموع جزئی N جمله بسط تیلور با نقطه پایه x=a می باشد. مقادیر پیش فرض این ماشین حساب به صورت زیر است:

f = x*cos(x),N=7,a=0,[-2*pi,2*pi]

taylortool('f') ماشین حساب فوق را برای تابع f اجرا می کند.

**مثال ها:**

```
taylortool('exp(x*sin(x))')
taylortool('sin(tan(x)) - tan(sin(x))')
```

**موارد مرتبط:**

funtool, rsums

## تابع tril :

**هدف:** ماتریس پایین مثلثی سمبلیک.

**املای درست:**

tril(X)
tril(X,K)

**شرح:** tril(X) قسمت پایین مثلثی ماتریس X را بر می گرداند.

tril(X, K) ماتریس را به این صورت پایین مثلثی می کند که عناصر پایین قطر K ام از ماتریس را حفظ می کند و بقیه عناصر (بالای قطر K ام) را صفر می کند. K می تواند مثبت، منفی یا صفر باشد.

**مثال ها:**

```
>>A =sym([ a, b, c ];[ 1, 2, 3 ];[ a+1, b+2, c+3 ]);
>>tril(A)
ans=
        [    a,    0,    0 ]
        [    1,    2,    0 ]
        [ a+1, b+2, c+3 ]
>>tril(A,1)
ans=
        [    a,    b,    0 ]
        [    1,    2,    3 ]
        [ a+1, b+2, c+3 ]
>>tril(A,-1)
ans=
        [    0,    0,    0 ]
        [    1,    0,    0 ]
        [ a+1, b+2,    0 ]
```

**موارد مرتبط:**

diag, triu

## تابع triu :

**هدف:** ماتریس بالا مثلثی سمبلیک.

**املای درست:**

triu(X)
triu(X, K)

**شرح:** triu(X) قسمت بالا مثلثی ماتریس X را بر می گرداند.
triu(X, K) ماتریس را به این صورت بالا مثلثی می کند که عناصر بالای قطر K ام از ماتریس را حفظ می کند و بقیه عناصر (زیر قطر K ام) را صفر می کند. K می تواند مثبت، منفی یا صفر باشد.

**مثال ها:**

فرض کنید:

```
A =
[    a,    b,    c ]
[    1,    2,    3 ]
[ a+1, b+2, c+3 ]
```

در این صورت:

```
>>triu(A)
ans=
        [        a,        b,        c ]
        [        0,        2,        3 ]
        [        0,        0,      c+3 ]
>>triu(A,1)
ans=
        [        0,        b,        c ]
        [        0,        0,        3 ]
        [        0,        0,        0 ]
>>triu(A,-1)
ans=
        [        a,        b,        c ]
        [        1,        2,        3 ]
        [        0,      b+2,      c+3 ]
```

**موارد مرتبط:**

diag, tril

## تابع vpa :

**هدف:** تعیین دقت متغیر در محاسبات.

**املای درست:**

R = vpa(A)
R = vpa(A,d)

**شرح:** vpa(A) از حساب دقت متغیر(VPA) برای محاسبه دقت هر عنصر از A تا d رقم اعشار استفاده می کند، که d تنظیم جاری تابع digits در زمان فراخوانی vpa می باشد.

vpa(A,d) از d رقم اعشار به جای مقدار پیش فرض d (در تابع digits) استفاده می کند.

**مثال ها:**

```
>>digits(25)
>>q = vpa(sin(sym('pi')/6)) , p = vpa(pi) , w = vpa('(1+sqrt(5))/2')
q =
        .5000000000000000000000000
p =
        3.141592653589793238462643
w =
        1.618033988749894848204587
```

دستور vpa pi 75 عدد پی را تا ۷۵ رقم اعشار محاسبه می کند.

```
>>A = vpa(hilb(2),25)
>>B = vpa(hilb(2),5)
A =
        [                          1. ,  .5000000000000000000000000  ]
        [ .5000000000000000000000000 ,  .3333333333333333333333333  ]
B =
        [          1. ,   .50000  ]
        [    .50000 ,   .33333  ]
```

**موارد مرتبط:**

digits, double

## تابع zeta :

**هدف:** تابع زتای ریمن[1].

**املای درست:**

Y = zeta(X)
Y = zeta(n, X)

**شرح:** تابع zeta(X) مقدار تابع زتا روی عناصر X ، روی یک ماتریس عددی یا روی یک ماتریس سمبلیک را بدست می آورد.

تابع زتا به صورت زیر تعریف می شود:

$$\zeta(w) = \sum_{k=1}^{\infty} \frac{1}{k^w}$$

[1] Riemann Zeta Functions

zeta(n, X)، n امین مشتق تابع zeta(X) را بر می گرداند.

**مثال ها:**

```
>>zeta(1.5)
ans=
        2.6124.
>>zeta(1.2:0.1:2.1)
ans=
        Columns 1 through 7
            5.59   16    3.93   19    3.1055    2.6124    2.2858    2.0543    1.8822
        Columns 8 through 10
            1.7497    1.6449    1.5602
>>zeta([x 2;4 x+y])
ans=
        [     zeta(x) ,   1/6*pi^2  ]
        [ 1/90*pi^4 , zeta(x+y)  ]
>>diff(zeta(x),x,3)
ans=
        zeta(3,x)
```

## تابع ztrans:

**هدف:** تبدیل z .

**املای درست:**

```
F = ztrans(f)
F = ztrans(f,w)
F = ztrans(f,k,w)
```

**شرح:** F = ztrans(f) تبدیل z اسکالر سمبلیک f با متغیر مستقل پیش فرض n. مقدار پیش فرض بازگردانده شده تابعی از z می باشد.

$$f = f(n) \Rightarrow F = F(z)$$

تبدیل z از f به صورت زیر تعریف می شود:

$$F(z) = \sum_{0}^{\infty} \frac{f(n)}{z^n}$$

که در آن n متغیر سمبلیکی است که در f وجود دارد و به وسیله findsym مشخص می شود. اگر f=f(z)، در این صورت ztrans(f) تابعی از w را بر می گرداند.

$$F = F(w)$$

F = ztrans(f,w)، f را تابعی از w در نظر می گیرد (به جای مقدار پیش فرض z).

$$F(w) = \sum_{0}^{\infty} \frac{f(n)}{w^n}$$

F = ztrans(f,k,w) ، f را به عنوان تابعی از متغیر سمبلیک k دریافت می کند.

$$F(w) = \sum_{0}^{\infty} \frac{f(k)}{w^k}$$

**مثال ها:**

| **Z-Transform** | **MATLAB Operation** |
|---|---|
| $f(n) = n^4$ | `f = n^4` |
| $Z[f] = \sum_{n=0}^{\infty} f(n) z^{-n}$ $= \frac{z(z^3 + 11z^2 + 11z + 1)}{(z-1)^5}$ | `ztrans(f)` returns `z*(z^3+11*z^2+11*z+1)/(z-1)^5` |
| $g(z) = a^z$ | `g = a^z` |
| $Z[g] = \sum_{z=0}^{\infty} g(z) w^{-z}$ $= \frac{w}{a-w}$ | `simplify(ztrans(g))` returns `-w/(-w+a)` |
| $f(n) = \sin an$ | `f = sin(a*n)` |
| $Z[f] = \sum_{n=0}^{\infty} f(n) w^{-n}$ $= \frac{w \sin a}{1 - 2w\cos a + w^2}$ | `ztrans(f,w)` returns `w*sin(a)/(w^2-2*w*cos(a)+1)` |

**موارد مرتبط:**

fourier, iztrans, laplace

# راهنمای سازگاری

## سازگاری با نسخه های قبلی ، توابع منسوخ

### سازگاری با نسخه های قبلی

نسخه های قبلی جعبه ابزار ریاضیات سمبلیک با نسخه های 4.0 یا 4.1 نرم افزار MATLAB یا با Maple V کار می کنند. هدف فراهم کردن دسترسی به Maple با یک املای زبانی بود که کاربران MATLAB با آن آشنا باشند. این هدف تقریبا بدون تغییر دادن هیچ یک از دو سیستم انجام شد.

با این وجود امکان به وجود آوردن یک سری دستور مجتمع و یکپارچه بدون تغییر MATLAB وجود نداشت. برای مثال اگر f و g رشته هایی باشند که عبارات سمبلیک را نشان می دهند، ممکن است ما بخواهیم برای مجموع آنها از املای f+g به جای symadd(f,g) استفاده کنیم، اما f+g مبادرت به جمع کارکتر های مشابه آنها در دو رشته خواهد کرد و آنها را به صورت الحاق دو رشته با یک علامت + بین آنها نخواهد نوشت. به طور مشابه اگر A یک ماتریس از رشته ها باشد، ممکن است ما بخواهیم به جای دستور sym(A,i,j) با استفاده از دستور A(i,j) به عبارات ماتریس دسترسی پیدا کنیم، اما چون A یک ماتریس از عبارات رشته ای است دستور A(i,j) یک کارکتر بر می گرداند و نه یک عبارت کامل.

این نسخه از جعبه ابزار ریاضیات سمبلیک امکانات گسترده ای را برای کار با اشیاء جدید MATLAB و اشیاء Maple V فراهم آورده است. به این دلیل این نسخه با نسخه ۱ از جعبه ابزار ریاضیات سمبلیک به طور کامل سازگار نیست.

### توابع منسوخ

این نسخه از جعبه ابزار ریاضیات سمبلیک اکثر امکانات نسخه ۱ را دارد. برای مثال توابع منسوخ زیر در این نسخه از جعبه ابزار ریاضیات سمبلیک نیز وجود دارد، البته به شما توصیه می شود که از آنها استفاده نکنید، چون نسخه های بعدی جعبه ابزار ریاضیات سمبلیک ممکن است شامل آنها نباشد.

در نسخه ۱ این توابع رشته را به عنوان آرگومان ورودی دریافت می کنند و یک رشته به عنوان خروجی بر می گردانند. در نسخه ۲ این توابع هم رشته می گیرند و هم عبارات سمبلیک و یک عبارت سمبلیک را بر می گردانند. نسخه ۲ شامل بسیاری از توابع جدید می باشد و بسیاری از عملگر ها نیز در این نسخه مجددا تعریف شده اند که می توانید این توابع جدید و عملگر ها را جایگزین توابع قدیمی کنید. برای مثال دستورات

```
>>f = '1/(5+4*cos(x))'
>>g = int(int(diff(f,2)))
>>e = symsub(f,g)
>>simple(e)
```

از نسخه ۱ را می توانید به وسیله دستورات

```
>>syms x
>>f = 1/(5+4*cos(x))
```

```
>>g = int(int(diff(f,2)))
>>e = f - g
>>simple(e)
```

جایگزین کنید.

| تابع | شرح تابع |
|---|---|
| determ | Symbolic matrix determinant |
| linsolve | Solve simultaneous linear equations |
| eigensys | Symbolic eigenvalues and eigenvectors |
| singvals | Symbolic singular values and singular vectors |
| numeric | Convert symbolic matrix to numeric form |
| symop | Symbolic operations |
| symadd | Add symbolic expressions |
| symsub | Subtract symbolic expressions |
| symmul | Multiply symbolic expressions |
| symdiv | Divide symbolic expressions |
| sympow | Power of symbolic expression |
| eval | Evaluate a symbolic expression |

به عنوان مثالی دیگر دستورات

```
H = sym(hilb(3))
I = sym(eye(3))
X = linsolve(H,I)
t = sym(0)
for j = 1:3
        t = symadd(t,sym(X,j,j))
end
t
```

در نسخه ۱ به وسیله دستورات

```
H = sym(hilb(3))
I = eye(3)
X = H\I
t = sum(diag(X))
```

در نسخه ۲ جایگزین می شوند.

دیگر شما نمی توانید از دستور sym به شکل M = sym(3,3,'1/(i+j-t)') استفاده کنید. به جای آن شما باید کد خود را به چیزی شبیه به این تبدیل کنید:

```
syms t
[J,I] = meshgrid(1:3)
M = 1./(I+J-t)
```

همانند نسخه ۱، در نسخه ۲ هم شما می توانید به توابع diff، int، solve و dsolve رشته ارسال کنید، البته در نسخه ۲ این توابع به جای رشته یک عبارت سمبلیک را بر می گردانند.

برای برخی از محاسبات نسخه جدید Maple نتایج را در یک فرمت جدید تولید می کند، برای مثال در نسخه ۱ دستور

```
>>[x,y] = solve('x^2 + 2*x*y + y^2 = 4', 'x^3 + 4*y^3 = 1')
```

نتایج

```
x =
        [          -RootOf(_Z^3-2*_Z^2-4*_Z-3)-2 ]
        [   -RootOf(3*_Z^3+6*_Z^2-12*_Z+7)+2 ]
y =
        [           RootOf(_Z^3-2*_Z^2-4*_Z-3)   ]
        [    RootOf(3*_Z^3+6*_Z^2-12*_Z+7)    ]
```

را تولید می کنند، در حالی که همین دستورات در نسخه ۲ نتایج را پس از بدست آوردن مقدار RootOf چاپ می کند.