به نام ایزد یکتا

راهنما و کاربرد

MATLAB 6.5
&
SIMULINK

تألیف و ترجمه : دکتر انوشیروان فرشیدیان فر

مهندس حمید دلیر

انتشارات ناقوس

فهرست

پیشگفتار مترجمان

در طی چند سال گذشته MATLAB به عنوان یک نرم‌افزار قوی در محیط‌های دانشگاهی و صنعتی برای محاسبات عددی،تحلیل داده‌ها و گرافیک، کاربرد وسیعی یافته است. از سوی دیگر جعبه ابزار[1]هایی هر ساله به این نرم‌افزار اضافه می‌شود که به سادگی بسیاری از کارهای محاسباتی همچون بهینه سازی، کنترل سیستم‌ها، حل معادلات دیفرانسیل و شبیه سازی سیستم‌های دینامیکی را انجام می‌دهد.

اولین نسخهٔ نرم‌افزار MATLAB در اواخر دههٔ هفتاد میلادی در دانشگاه‌های نیومکزیکو و استانفورد جهت استفاده در دروس تئوری ماتریس‌ها، جبر خطی و تحلیل عددی نوشته شده، که اساس آن را برنامه‌های LINPACK و EISPACK تشکیل می‌دادند که سابروتین‌های فرترن[2] جهت انجام محاسبات ماتریسی بودند و شاید نام MATLAB نیز که برگرفته از آزمایشگاه ماتریسی[3] است، بدین دلیل باشد. ریاضیات، امروزه زبان مشترک بسیاری از رشته‌های علوم و مهندسی است و نرم‌افزار MATLAB که می‌توان آن را زبان ریاضیات مدرن نامید. ابزار قدرتمندی برای پردازش اطلاعات در ساختارهای ماتریسی بوده که علاوه بر آن در حل معادلات دیفرانسیل، تحلیل داده‌ها، ترسیمات و گرافیک به عنوان ابزاری قوی، مهندسین و محققین را یاری می‌کند.

مجموعه قابلیت‌های فوق و کاربرد روز افزون این نرم‌افزار در مجامع داخلی وخارجی از یک طرف و عدم وجود مأخذ مناسبی در این زمینه از طرف دیگر، ما را بر آن داشت تا به تهیهٔ این کتاب بپردازیم. در این کتاب تلاش شده است اطلاعات جدیدترین نسخهٔ MATLAB عرضه شود.

چارچوب و شالودهٔ اصلی این کتاب، کتاب مشهور «راهنما و کاربرد نرم‌افزار مطلب در مهندسی»

An Engineer’s Guide to MATLAB with Application

نوشتهٔ :

Magrab, Azarm, Balachandran, Duncan, Herold and walsh

می‌باشد، که هفت فصل اول این کتاب بدون هیچگونه دخل و تصرفی عیناً در کتاب آورده شده است.

---

[1] Tool Box
[2] Fortran Subroutine
[3] Matrix Laboratory

بخاطر اهمیت جعبه ابزار SIMULINK که در کتاب اصلی نبود، بر آن شدیم که مجموعهٔ نسبتاً کاملی را در ارتباط با این جعبهٔ ابزار گردآوری کرده و بعنوان فصل هشتم به کتاب حاضر اضافه نماییم، تا هم مطالب کتاب کامل شده و هم به چارچوب کتاب اصلی لطمه‌ای نخورد.

SIMULINK یک برنامهٔ نرم‌افزاری قوی جهت مدلسازی شبیه‌سازی و تجزیه و تحلیل سیستم‌های دینامیکی می‌باشد. این نرم‌افزار را می‌توان جهت تحلیل سیستم‌های خطی و غیرخطی، سیستم‌های زمان پیوسته و زمان گسسته یا مخلوطی از هر دو بکار برد.

در پایان کتاب نیز مجموعهٔ کاملی از تمامی دستورات مهم MATLAB، که در بخش‌های مختلف آن بکار می‌رود، به همراه توضیح مختصری از نحوهٔ کاربرد آنها آورده شده است.

در اینجا لازم می‌دانیم که از تمامی دست اندرکاران تهیهٔ این کتاب تقدیر و قدردانی نماییم. لذا از سرکار خانم سارا رزاز زاده[٤] که با دقت و حوصلهٔ فراوان کار ویراستاری کتاب را انجام دادند، صمیمانه سپاسگذاریم. همچنین از خانم طناز فرشیدیان‌فر، خانم الهه شاهسون و آقای شهرام فرشیدیان‌فر که زحمت تایپ کتاب را بر عهده داشتند، و از انتشارات ناقوس که کار چاپ و نشر کتاب را انجام دادند، صمیمانه تشکر و تقدیر می‌نماییم.

در پایان از کلیهٔ صاحب نظران و خوانندگان گرامی تقاضا داریم تا پیشنهادات خود را در مورد کتاب به نشانی انتشارات ناقوس ارسال نمایند تا در چاپهای بعدی مورد توجه واقع گردد.

با تشکر فراوان

دکتر انوشیروان فرشیدیان فر

مهندس حمید دلیر

٤- نام ایشان اخیراً به شایان امین تغییر یافته است.

پیشگفتار نویسندگان

اولین هدف این کتاب راهنمایی خوانندگان جهت یادگیری و توسعهٔ دانش بهره وری از نرم افزارهای MATLAB و SIMULINK جهت حل مسائل مهندسی می باشد. معمولاً حل اینگونه مسائل در نرم‌افزار MATLAB با نوشتن برنامه هایی کوتاه میسر می باشد. لذا در این کتاب تلاش شده است تا چگونگی نوشتن برنامه های کوتاه و کارآمد و در عین حال با حفظ خوانایی، سهولت در تصحیح و زمان اجرای کوتاه، آموخته شود.

تأکید عمدهٔ این کتاب بر روی چگونگی استفاده از نرم افزار MATLAB جهت بدست آوردن راه‌حلهایی برای حل انواع مختلف مسائل مهندسی می‌باشد. لذا موضوعات تکنیکی مطرح شده در این کتاب، بصورت خلاصه آورده شده‌اند. از این کتاب می توان در موارد زیر نیز استفاده نمود:

۱) به عنوان کتاب آموزشی جهت استفادهٔ دانشجویان کارشناسی و دانشجویان دوره‌های تحصیلات تکمیلی.

۲) به عنوان مرجعی جهت دستیابی به حلهای عددی گسترهٔ وسیعی از مسائل مهندسی و کاربردهای متفاوت روشهای حل نرم افزار MATLAB.

در این کتاب کاربرد های برنامه‌نویسی مهندسی در موارد ذیل آورده شده است:

۱) تجزیه و تحلیل یک مدل پیش بینانه، مثل معادلات جبری، معادلات دیفرانسیل معمولی، معادلات دیفرانسیل با مشتقات جزئی و یا تقریبی از موارد فوق.

۲) گرفتن نتایج آماری از داده ها.

۳) شماتیک کردن یک مدل و یا مجموعه ای از داده ها جهت قابل فهم نمودن مسئله.

٤) بدست آوردن مدلی تجربی با استفاده از نتایج آزمایشگاهی.

هفت فصل اول این کتاب به معرفی نرم افزار MATLAB، چگونگی استفاده از علائم برداری و ماتریسی و تعاریف مربوط به آنها اختصاص یافته است. تعداد زیادی مثال و توضیحات همراه با جزئیات جهت توضیح بهتر مطالب بکار گرفته شده است. در این کتاب سعی شده است تا اهمیت فرمول نویسی بصورت بردار/ ماتریس و مزایای آن در کوتاهتر شدن برنامه‌ها نشان داده شود. تعداد زیادی مثال نمونه جهت نشان دادن چگونگی نمایش گرافیکی داده‌ها، آورده شده است.

در فصل هشتم کتاب به معرفی نرم افزار SIMULINK و برخی از کاربردهای آن پرداخته شده است.

در برنامه‌ها و توابع آورده شده در سرتاسر کتاب از مناسب ترین توابع MATLAB جهت بدست آوردن نتایج عددی استفاده شده است.

E. B. Magrab
S. Azarm
B. Balachandran
J. H. Duncan
K. E. Herold
G. C. Walsh
CollegePark,MD

فصل اول

مقدمه

در این بخش ویژگیهای اساسی محیط مطلب و شکل کلی دستورات آن معرفی خواهند شد.

۱-۱ درآمدی بر مطلب

نرم افزار مطلب (MATLAB) که نامش را از عبارت *Matrix Laboratory* اخذ کرده است، یک نرم‌افزار محاسباتی جهت پردازش داده‌ها به صورت ماتریسهایی متشکل از اعداد می‌باشد. مطلب محاسبه و تجسم را در یک محیط کامپیوتری قابل انعطاف، تلفیق می‌کند. و یک مجموعهٔ متنوع از

توابع داخلی را ارائه می‌دهد که می‌توانند به طور مستقیم جهت دستیابی به راه‌حل‌های عددی در مورد طیف وسیعی از مسائل مهندسی بکار روند.

۱-۲ روشهای استفاده از مطلب

هنگامی که برنامهٔ مطلب اجرا می‌شود، کاربر با پنجره‌ای که یک مکان نمای چشمک زن[۱] بلافاصله پس از علامت "<<" ظاهر می‌شود، مواجه خواهد شد. معمولاً این پنجره که پنجرهٔ فرمان نامیده می‌شود، یک محیط کاری مشابه ورق کاغذ سفید است. فضایی که بلافاصله پس از علامت "<<" ظاهر می‌شود، خط فرمان نام دارد. که می‌توان به آسانی مقادیر عددی یک ماتریس را، با تایپ کردن ماتریس اعداد در خط فرمان با فرمت مجاز که در بخش ۲-٤ بحث خواهد شد، در آن وارد نمود. برای نسبت دادن این اعداد با یک متغیر، متغیر باید به همراه علامت تساوی، قبل از آنها بکار رود. اگر از هیچ متغیری استفاده نشود، مطلب آنها را به متغیر عمومی *ans* نسبت خواهد داد. در هر دو مورد فوق، می‌توان از ماتریس برای نمایش در پنجرهٔ فرمان استفاده کرد و یا در عبارات محاسباتی دیگری در مطلب، با تایپ کردن نام متغیر تعیین شده از سوی کاربر و یا متغیر عمومی *ans* بکاررود. اما اگر متغیری با نام مشابه برای تساوی با ماتریس دیگری به کار رود، و یا مجموعه‌ای از اعداد جدید در خط فرمان بدون مشخص کردن نام متغیر بکار برده شود، اعداد وارد شدهٔ قبلی برای متغیر تعیین شده از سوی کاربر و یا متغیر عمومی *ans* باز نویسی خواهند شد. بعلاوه تایپ کردن فرمان*clear* فضای کاری را پاک خواهد کرد. این فرمان را با جزئیات بیشتر در ادامه توضیح خواهیم داد.

مطلب این قابلیت را به کاربر می‌دهد که عملیات ریاضیاتی، مثلثاتی و نمایی (جمع، تقسیم، کسینوس و لگاریتم) را همانگونه که با ماشین حساب انجام می‌دهد، انجام دهد. این عملیات بوسیلهٔ ابزارهایی که در مطلب تابع نام دارد، انجام می‌شوند. علاوه بر این توابع پایهٔ ماشین حسابی، نرم افزار مطلب دارای مجموعهٔ گسترده‌ای از توابعی که عملیات بسیار پیچیدهٔ ریاضیاتی را انجام می‌دهند، می‌باشد. همچنین ابزارهایی برای کاربران جهت ایجاد توابع خاص خودشان فراهم شده است که در فصل پنجم شرح داده خواهد شد. مزیت دیگر این توابع اینست که قابلیت استفادهٔ اصولی را جهت نوشتن دستورات برنامه نویسی به کاربر می‌دهند. دستورات برنامه نویسی با عباراتی که در خط فرمان

۱- شکل پنجرهٔ فرمان و تعیین مدیریت فایلها، به محیط ویندوز مربوط می‌شود. این مسئله در مورد سایر سیستم‌های عامل نیز صادق می‌باشد.

وارد می‌شوند، متفاوت هستند. در مطلب فضای مخصوصی به آنها اختصاص داده شده است و روابط بین ورودی – خروجی را به طور مشخص و منظم با استفاده از محیط مطلب تعریف می‌کنند.

گاهی لازم است که کاربر مکرراً مجموعه‌ای از عبارات را در پنجرهٔ فرمان تایپ کند که این کاری خسته کننده خواهد بود. برای رفع این مشکل، مطلب برنامه‌ای را معرفی کرده است که شامل تعدادی فرمان است که اگر در خط فرمان واقع در پنجرهٔ فرمان تایپ شوند، اجرا خواهند شد. یک فایل متنی ممکن است در یک پردازشگر لغت، ویرایشگر متن و یا در محیط اشکال زدایی[۱] و ویرایش متن مطلب، ایجاد شود و بعنوان یک فایل با پسوند *".m"* ذخیره شود. اگر از پردازشگر لغت و یا ویرایشگر متن استفاده شود، فایل با تایپ اسم فایل بدون پسوند *".m"* در محیط پنجرهٔ فرمان مطلب، قابل اجرا خواهد بود. اگر از ویرایشگر مطلب استفاده شود، با کلیک کردن روی نوار ابزار *Tools* و سپس انتخاب *Run*، فایل متنی اجرا خواهد شد. به هر حال قبل از اجرای هر فایل، باید آنرا ذخیره کنیم.

فایل های متنی معمولاً در موارد زیر استفاده می‌شوند:

برنامه شامل بیش از چند خط دستور باشد.

نیاز به اجرای دوبارهٔ برنامه باشد.

ثبت دائمی نتایج مطلوب باشد.

برنامه نیاز به ارتقاء داشته باشد.

اشکال زدائی های اساسی مورد نیاز باشد.

کاربر بخواهد که برنامه را به شخص دیگر و یا سازمانی انتقال دهد. به علاوه، یک متن و یا تابع معمولاً دارای ویژگیهای زیر است.

مستندات، که حداقل موارد زیر را نشان می‌دهد:

هدف و عملیاتی که اجرا می‌شود.

نام برنامه نویس.

تاریخ شروع.

۱- کلیک کردن روی آیکن سمت چپ پنجره فرمان مطلب (مستطیل سفید) و یا انتخاب گزینهٔ *New* یا *M - File* از منوی پایین کشیدنی *File* دسترسی به محیط ویرایشگر یا اشکال زدایی مطلب را ممکن می‌سازد.

تاریخ تجدید نظر.

توصیف ورودی (ورودیها): شماره، مفهوم و نوع.

توصیف خروجی (خروجیها): شماره، مفهوم و نوع.

ورودی: که شامل چندین بررسی جهت اطمینان از اینکه همهٔ مقادیر ورودی، شرایط لازم برای استفاده در فایلهای متنی و یا توابع را دارا می‌باشد.

مقداردهی اولیه: که به متغیرهای دلخواه به مقادیر عددی نسبت داده می‌شوند.

محاسبات: که عمدتاً بصورت عددی انجام می‌شوند.

خروجی: که نتایج به صورت گرافیکی و یا کمیتهای عددی نشان داده می‌شوند.

۱-۳ نامگذاری متغیرها

مطلب این قابلیت را به کاربر می‌دهد که متغیرهایی با طول حداکثر ۳۱ کاراکتر حرفی و یا عددی ایجاد کند. کاراکترهای پس از ۳۱ امین کاراکتر در نظر گرفته نمی‌شوند. نام هر متغیر باید با یک حرف کوچک و یا بزرگ شروع شود که پس از آن می توان از هر ترکیبی از حروف کوچک یا بزرگ، اعداد و یا زیر خط (_) استفاده نمود. اما نمی توان از فاصله خالی بین این کاراکترها استفاده کرد.

نامگذاری متغیرها بسیار حساس است، زیرا متغیری با نام *junk* با متغیر *junK* متفاوت می‌باشد. دو شیوهٔ متداول در این زمینه وجود دارد : یکی استفاده از زیر خط و دیگری استفاده از حروف بزرگ می‌باشد. به عنوان مثال اگر فشار خروجی کمیت مورد نظر باشد در خط فرمان مطلب یا فایل متنی و یا تابع می توان از *exit _ Pressure* و یا *ExitPressure* استفاده نمود.

متداولترین کاراکترها و نشانه‌های مورد استفاده در مطلب به همراه مفهومشان در جدول الف- ۱ در ضمیمهٔ انتهای این فصل آمده‌اند.

۱-۴ کنترل متغیرها

در هر بار استفاده از نرم‌افزار مطلب- یعنی فاصلهٔ زمانی بین اجرا تا خروج- آخرین مقادیر بدست‌آمده از همهٔ متغیرهای تعریف شده بوسیلهٔ هر عبارت و یا محاسبه شده توسط یک برنامه تا وقتی که تابع *clear* بکار نرفته است در حافظه باقی می‌ماند. همانگونه که قبلاً اشاره شد، آخرین

مقادیر عددی که به این متغیرها نسبت داده شده‌اند، در هر دفعه استفاده از مطلب براحتی با وارد نمودن نام متغیر و یا استفاده از آن در یک عبارت محاسباتی قابل دستیابی می‌باشند (به شرطی که دستور *clear* بکار نرفته باشد). این متغیرها، متغیرهای کلی نامیده می‌شوند.

مطالب وارد شده در پنجرهٔ فرمان مطلب تا زمانی که حافظهٔ طوماری اشباع نشده باشد در پنجره باقی می‌مانند و با حرکت دادن نوارهای لغزان افقی و عمودی قابل دسترسی خواهند بود. پس از اشباع شدن حافظهٔ طوماری اطلاعاتی که در ابتدا وارد شده‌اند، به ترتیب پاک خواهند شد. اما عباراتی که بوسیلهٔ برنامه‌ها محاسبه می‌شوند، قابل دسترسی نخواهند بود؛ ولی توجه دارید که نام متغیرها و مقادیر عددی شان همانگونه که گفته شد، قابل دسترسی هستند. عبارات تایپ شدهٔ قبلی را می‌توان توسط منوی *Edit* در بالای پنجرهٔ فرمان مطلب و انتخاب گزینهٔ *Clear session* ، پاک نمود. اما با اینکار متغیرهایی را که توسط دستور *clear* پاک می‌شوند از بین نمی‌روند. همچنین آیکن‌های *copy* و *paste* را می‌توان جهت فراخوانی عبارات تایپ شدهٔ قبلی به خط فعلی فرمان و یا کپی کردن دستورات مطلب از پنجرهٔ فرمان مطلب به پنجرهٔ پردازشگر لغت و بالعکس استفاده نمود. برای لیست کردن متغیرهایی که پس از آخرین استفاده از فرمان *clear*، ساخته شده‌اند می‌توان دستور whos را در پنجرهٔ فرمان تایپ کرد و یا از طریق منوی *File* و انتخاب گزینهٔ *work space show* را انتخاب کرد. همچنین با این شیوه می‌توان نام متغیرها، اندازهٔ آنها، تعداد بایتهای اشغال شده توسط هر متغیر و نوع آنها که شامل : عددی، رشته‌ای (بخش۳-۱)، سیمبولیک و یا توابع محلی است (بخش ۵-۳) را آشکار ساخت. مزیت روش دوم اینست که اگر پنجرهٔ فرمان مطلب فعال باشد، این اطلاعات به آسانی قابل دسترسی خواهند بود؛ زیرا مطلب بطور مداوم آنها را به روز می‌کند.

۱-۵ کنترل توابع و برنامه‌ها

برنامه و توابع با تایپ کردن نام فایل (بدون پسوند *".m"*) در پنجرهٔ فرمان مطلب اجرا می‌شوند. اما دایرکتوری مطلب باید به مسیر دایرکتوری که فایل در آنجا می‌باشد، تغییر مسیر دهد. اطلاعات مربوط به مسیر توسط انتخاب گزینهٔ *set path* در منوی *file* وارد می‌شود. با این کار پنجرهٔ *path Browser* باز می‌شود. سپس روی گزینهٔ *Browse* کلیک کرده و دایرکتوری مورد نظر را انتخاب می‌کنیم. سپس گزینهٔ *Path* را انتخاب کرده و روی گزینهٔ *Add to path* و یا *Add to front* و یا *Add to back* کلیک می‌کنیم. ممکن است هنگامی که در پنجرهٔ *Path Browser* هستیم، بیش از یک دایرکتوری ظاهر شود. قبل از بستن پنجرهٔ *path Browser* توصیه می‌شود که روی گزینهٔ *File* کلیک کرده و سپس *Save path* انتخاب شود تا مسیرهای انتخاب شده برای دفعهٔ بعد که از مطلب استفاده می‌شود، در حافظه باقی بمانند.

۱-۶ مدیریت پنجرهٔ فرمان

مطلب برای خوانایی بیشترِ حروف و اعدادی که در پنجرهٔ فرمان ظاهر می‌شوند، چند گزینه توسط تابع *format* ارائه می‌دهد. که دو تابعی که در زیر نشان داده شده است، دارای کاربرد بیشتری می‌باشند:

```
format compact
```

و

```
format long e
```

اولی بیشتر فضاهای خالی را حذف می‌کند و دومی کلید دوگانه‌ای جهت تبدیل فرمت پیش فرض ۵ رقم به فرمت ۱۶ رقم به انضمام ۳ رقم جهت توان می‌باشد. دستور *format long e* جهت اصلاح متن‌هایی بکار می‌رود. که اعداد حاصل از آنها در یک بازهٔ کوچک و یا نسبتاً وسیع تغییر می‌کند. برای بازگشت به تنظیمات پیش فرض باید عبارت زیر را تایپ نمود:

```
format short
```

برای تعیین فونت، اندازه، یا شیوهٔ تایپ که در پنجرهٔ فرمان مطلب ظاهر می‌شود ابتدا منوی *file* و سپس گزینهٔ *preferences* را انتخاب می‌کنیم. در این پنجره گزینهٔ *command window font* را انتخاب کرده و اصلاحات لازم را انجام می‌دهیم.

دو ورودی کاربردی توسط صفحه کلید *c^* (فشردن همزمان *ctrl* و *c*) و *p^* (فشردن همزمان *ctrl* و*p*) می‌باشند، عملکرد *p^*، قرار دادن آخرین ورودیهای تایپ شده از صفحه کلید در پنجرهٔ فرمان مطلب می‌باشد، که با فشردن کلید*Enter* کامل می‌شود. همچنین قبل از فشردن کلید *Enter* کاربر می‌تواند عبارات را اصلاح کند. اگر کلید *Enter* فشار داده نشود و بجای آن *p^* مجدداً *p^* وارد شود، آنگاه ورودیهای بعدی که اخیراً تایپ شده‌اند، جایگزین ورودیهای قبلی می‌شوند. نتایج مشابه را با استفاده از کلیدهای بالابر $(\uparrow)$ و یا پایین بر $(\downarrow)$ نیز می‌توان بدست آورد.

*c^* جهت متوقف کردن یا خروج از حالت توقف یک برنامه یا تابع در حال اجرا بکار می‌رود.

۱-۷ خروجی توابع و برنامه ها در پنجرهٔ فرمان

چنانچه کاربر بخواهد مقادیر عددی یا یک سری از اعداد به شکل بردار یا ماتریس، (به بخشهای۲-۳ و۲-٤ مراجعه کنید.) را در قسمت پنجرهٔ فرمان مطلب وارد کند، برنامه (و یا تابع) به صورت عبارت زیر خواهد بود:

```
VariableName = input ('Any message')
```

که در اینجا input یکی از توابع مطلب می‌باشد و پیام داخل پرانتز در پنجرهٔ فرمان نشان داده خواهد شد. پس از اجرای این فرمان و ورود داده‌ها، کلید *Enter* را فشار داده، مقدار (یا مقادیر) وارد شده به متغیر مربوطه نسبت داده خواهد شد. شیوه‌های دیگر وارد کردن داده‌ها در بخش ۳-۳ آمده است و اطلاعات بیشتر در مورد استفاده از فرمان input در بخش ۳-۲ ارائه شده است.

از طرفی دو روش برای گرفتن نتایج برنامه و نمایش آنها در پنجرهٔ فرمان وجود دارد. یک روش پاک کردن علامت نقطه ویرگول (;) در انتهای عبارت می‌باشد. که این مطلب در بخش۱-۸ نشان داده شده است. (همچنین به جدول الف ـ ۱ در ضمیمهٔ انتهای فصل مراجعه کنید). در این حالت، مطلب نام متغیر را به همراه علامت تساوی در پنجرهٔ فرمان ارائه می‌دهد و سپس به خط بعدی رفته و مقدار (مقادیر) متغیر را نشان می‌دهد. این شیوه در هنگام غلط زدایی کاربرد بسیار دارد. هنگامی که می‌خواهیم مقادیر خروجی را برای وضوح با شرح بیشتری نمایش دهیم، می‌توان از یکی از دو دستور؛

```
disp
```

و یا

```
fprintf
```

استفاده نمود. که در بخش ۳-۱ توضیح داده خواهند شد.

۱-۷-۱ راهنمای حین کار

مطلب قابلیتهای یک راهنمای حین کار را دارا است، که می‌توان به چند طریق به آن دسترسی یافت. یک روش، کلیک کردن روی آیکن علامت سوال (؟) در نوار ابزار می‌باشد. این آیکن پنجرهٔ *Help* را

باز می‌کند که باید پس از هر بار استفاده به مینی‌مایز شود تا پنجرهٔ فرمان و پنجرهٔ پردازشگر لغت (یا ویرایشگر متن) قابل دسترسی باشد. روش دیگر تایپ عبارت زیر در پنجرهٔ فرمان است.

```
help FunctionName
```

که *FunctionName* نام تابعی است که اطلاعات در مورد آن نیاز است.

روش سوم انتخاب *Help Desk (HTML)* از منوی *Help* می‌باشد که سیستم جستجوگر شبکهٔ کامپیوتر را فعال می‌کند تا توصیف کامل تری از دستور (دستورات) نمایش داده شود. همچنین این شکل از اطلاعات از طریق کلیک کردن روی آیکن علامت سوال (؟) در پنجرهٔ فرمان و انتخاب گزینهٔ *Go to Help Desk* در پنجرهٔ راهنمای مطلب قابل دستیابی خواهند بود.

۱-۸ شکل اساسی دستورات در مطلب

تمام متغیرها در مطلب (بجز متغیرهایی که توسط جعبه ابزار سیمبولیک استفاده می‌شوند) باید قبل از استفاده در عبارات محاسباتی، مقدار دهی عددی شوند. تایپ کردن نشانهٔ متغیر، علامت مساوی و مقدار (مقادیر) عددی و سپس فشردن کلید *Enter* کار مقدار دهی را انجام می‌دهد.

بنابراین، اگر بخواهیم $p = 7.1$ ، $x = 4.92$ و $k = -1.7$ باشد، آنگاه باید عملیات زیر را در پنجرهٔ فرمان انجام دهیم:

```
» p = 7.1                    ← کاربر تایپ می‌کند.
p =                          ← مطلب پاسخ می‌دهد.
    7.1000
» x = 4.92                   ← کاربر تایپ می‌کند.
x =                          ← مطلب پاسخ می‌دهد.
    4.9200
» k = -1.7                   ← کاربر تایپ می‌کند.
k =                          ← مطلب پاسخ می‌دهد.
   -1.7000
```

اگر کاربر بخواهد خروجیها را از پنجرهٔ فرمان حذف کند می‌تواند مطابق زیر از علامت نقطه ویرگول (;) در انتهای دستور مربوطه استفاده کند:

```
>> p = 7.1;
>> x = 4.9.2;
>> k = −1.7;
>>
```

همچنین مطلب این قابلیت را به کاربر می‌دهد تا چندین عبارت را در یک خط قرار دهد. زمانی که کلید *Enter* فشرده شود مکان نما به خط بعد خواهد رفت. در این حالت هر عبارت توسط علامت کاما (,) و یا نقطه ویرگول (;) از سایر عبارتها جدا می‌شود. هنگامی که از علامت کاما استفاده می‌شود، ورودی تکرار خواهد شد. بنابراین اگر عبارت زیر تایپ شود:

```
>> p = 7.1, x = 4.92, k = −1.7
```

پاسخ مطلب به شکل زیر خواهد بود:

```
p =
  7.1000
x =
  4.9200
k =
  − 1.7000
>>
```

استفاده از نقطه ویرگول بجای کاما باعث خواهد شد که خروجیها نشان داده نشود. پنج عملگر ریاضیاتی برای انجام جمع، تفریق، ضرب، تقسیم و توان اسکالر به ترتیب + ، − ، × ، / و ^ می‌باشند. برای مثال عبارت ریاضیاتی زیر:

$$t = \left(\frac{1}{1 + px} \right)^k$$

در مطلب به صورت زیر نوشته می‌شود:

```
t = (1/(1 + p * x))^k
```

کمیتهای p ، x و k قبل از اجرای عبارت فوق باید مقداردهی عددی شوند. در صورت انجام نشدن این کار پیغام خطایی ظاهر خواهد شد. با فرض اینکه مقادیر p، x و k برابر مقادیر قبلی باشد، سیستم مقدار زیر را نمایش خواهد داد[۱] :

```
440. 8779
```

در محاسبهٔ t وجود پرانتزها الزامی می‌باشند تا عملیات ریاضیاتی روی مجموعهٔ صحیحی از کمیتها با رعایت اولویت پرانتزها اعمال شود. در مطلب ترتیبی جهت محاسبهٔ عبارات ریاضیاتی وجود دارد، در نتیجه می‌توان تعداد پرانتزها را کاهش داد. با این وجود پرانتزهایی را که از دیدگاه مطلب غیرضروری می‌باشند، می‌توانند جهت قابل فهم تر کردن عبارات استفاده شوند. اولین اولویت با توان و به دنبال آن ضرب و تقسیم و جمع و تفریق می‌باشد. در مطلب در بین هر پرانتز و در مورد کل عبارات محاسباتی عملیات از چپ به راست انجام می‌شود. به مثالهای نشان داده شده در جدول ۱-۱ که شامل کمیتهای اسکالر c، d، g و x می‌باشند، توجه کنید. تابع زیر:

```
sqrt
```

ریشهٔ دوم آرگومان خود را محاسبه می‌کند.

جدول ۱-۱ (مثالهایی دربارهٔ شکل نحوی دستورات در مطلب)

شکل نوشتن در مطلب	شکل نوشتن در ریاضیات
`1 - d * c ^ ( x + 2 )`	$1-dc^{x+2}$
`d * c^ x + 2or2 + d * c^ X`	$dc^{x}+2$
`( 2 / d ) * c ^ ( x + 2 )or 2 / d * c ^ ( x + 2 )`	$(2/d)c^{x+2}$
`( d * c ^ x + 2 ) / g ^ 2.7`	$\left(dc^{x}+2\right)/g^{2.7}$
`( d * c ^ x + 2 )^. 5orsqrt ( d * c ^ x + 2 )`	$\sqrt{dc^{x}+2}$

۱- از اینجا به بعد پنجره فرمان را فراخوانی نخواهیم کرد، بلکه تنها خروجی‌ها را در فرمت مناسب نشان می‌دهیم.

همچنین نرم افزار مطلب شامل مجموعه ای وسیع از توابع مقدماتی و متوسط می باشد. چند تابع مقدماتی مطلب در جداول ۱-۲ و ۱-۳ آورده شده اند. آرگومانهای این توابع می توانند عدد، بردار و یا ماتریس باشد. تعریف بردارها و ماتریسها و چگونگی تشکیل آنها در مطلب در بخش ۲-۳ و۲-٤ آمده است. مثال زیر چگونگی استفاده از توابع ساخته شدهٔ داخلی مطلب را نشان می دهد:

$$y = \sqrt{|\pi - sin(x)/cosh(a) - ln_e(x+a)|}$$

این عبارت در مطلب به صورت زیر نوشته می‌شود:

```
y=sqrt(abs(pi-sin(x)/cosh(a)-log(x+a)))
```

در عبارت فوق $pi = \pi$ است. همچنین فرض می‌شود که x و a قبل از اجرای عبارت فوق، مقدار دهی شده‌اند.

۱-۹ چند راهکار برای استفاده از مطلب

موارد ذیل چند نکته در مورد چگونگی استفاده از محیط مطلب جهت ایجاد برنامه‌ها و توابع می‌باشد.

از فایلهای راهنما بطور گسترده استفاده شود. این کار موجب کاهش خطاهای ناشی از اشتباه نحوی و یا بکار بردن غیر صحیح و نامناسب توابع مطلب خواهد شد.

برنامه‌ها و توابع را ابتدا در ویرایشگر متن نوشته و سپس به عنوان m.file ذخیره کنید. این کار موجب صرفه‌جویی در زمان و تعداد دستورها خواهد شد و مهمتر اینکه فرآیند غلطیابی را بخصوص اگر از ویرایشگر یا غلط یاب مطلب استفاده شود، راحت تر خواهد کرد.

حتی الامکان باید تعداد عبارات و دستورهای برنامه‌ها و توابع را کاهش داد. اینکار معمولاً منجر به خوانایی و فشردگی بیشتر برنامه می‌شود.

هنگام استفادهٔ عملی از برنامه‌ها و یا توابع، نمایشهای گرافیکی به طور گسترده مورد استفاده قرار می‌گیرند. که اینکار معمولاً فرآیند برنامه‌نویسی را از طریق شناساندن موقعیتهای خطا خیز برنامه نویسی، کوتاه می‌کند و قادر است درک و فهم فرآیندی را که باید مدل و یا تحلیل شود، آسانتر سازد.

نکتهٔ پراهمیت دیگر این است که، درستی خروجی‌های برنامه‌ها و یا توابع باید توسط ابزارهای مستقل تأیید شوند.

جدول (۱-۳) توابع هیپربولیک و مثلثاتی مطلب توابع هیپربولیک و مثلثاتی مطلب				
معکوس تابع	تابع	معکوس تابع	تابع	تابع
asinh (x)	sinh (x)	asin (x)	sin (x)	sine
acosh (x)	cosh (x)	acos (x)	cos (x)	cosine
atanh (x)	tanh (x)	atan (x)†	tan (x)	tangent
asech (x)	sech (x)	asec (x)	sec (x)	secant
acsch (x)	csch (x)	acsc (x)	csc (x)	cosecant
acoth (x)	coth (x)	acot (x)	cot (x)	cotangent

†atan2(y,x) نسخهٔ قابل استفاده در چهار ربع مثلثاتی می‌باشد.

تمرین‌ها

۱-۱ عبارات زیر[1] تنش های تماسی اصلی را به ترتیب در جهات x، y و z هنگامیکه دو کره با نیروی F به یکدیگر فشرده می‌شوند، توصیف می‌کند.

$$\sigma_x = \sigma_y = -p_{max}\left[\left(1-\frac{z}{a}tan^{-1}\left(\frac{a}{z}\right)\right)(1-\upsilon_1)-0.5\left(1+\frac{z^2}{a^2}\right)^{-1}\right]$$

$$\sigma_z = \frac{-p_{\max}}{1+z^2/a^2}$$

که در آن داریم:

$$a = \sqrt[3]{\frac{3F}{8}\frac{\left(1-\upsilon_1^{\,2}\right)/E_1+\left(1-\upsilon_2^{\,2}\right)/E_2}{1/d_1+1/d_2}}$$

1- J. E. Shigley and C. R. Mischke, Mechanical Engineering Design, 5th ed. , McGraw-Hill, NewYork, 1989.

$$p_{max} = \frac{3F}{2\pi a^2}$$

و υ_j، E_j و d_j که $j = 1,2,3,...$، به ترتیب ضریب پواسان، مدول یا نگ و قطر دو کره می‌باشند. این معادلات را به زبان مطلب بنویسید. سپس آنها را با مقادیر زیر مقایسه کنید:

$\upsilon_1 = \upsilon_2 = 0.3$، $E_1 = E_2 = 3\times10^7$، $d_1 = 1.5$، $d_2 = 2.75$، $F = 100lb$ و $z = 0.01in$. این عبارات را برای استفاده در تمرین ٦-٤ ذخیره نمایید.

{پاسخ: $a = 0.013in$، $p_{max} = 281{,}580psi$، $\sigma_x = -108{,}580\,psi$ و $\sigma_z = -177{,}120\,psi$}

۱-۲ عبارات زیر[1] تنش های تماسی اصلی در جهات x، y و z برای دو استوانه با محورهای موازی که توسط نیروی F به یکدیگر فشرده می‌شوند را به ترتیب نشان می‌دهد:

$$\sigma_x = -2\nu_2 p_{max}\left(\sqrt{1+\frac{z^2}{b^2}} - \frac{z}{b}\right)$$

$$\sigma_y = -p_{max}\left(\left(2-\left(1+\frac{z^2}{b^2}\right)^{-1}\right)\sqrt{1+\frac{z^2}{b^2}} - 2\frac{z}{b}\right)$$

$$\sigma_z = \frac{-p_{max}}{\sqrt{1+z^2/b^2}}$$

$$\tau_{yz} = 0.5\left(\sigma_y - \sigma_z\right)$$

که در آن؛

$$p_{max} = \frac{2F}{\pi bL}$$

$$b = \sqrt{\frac{2F}{\pi L}\frac{\left(1-\upsilon_1^{\,2}\right)/E_1 + \left(1-\upsilon_2^{\,2}\right)/E_2}{1/d_1 + 1/d_2}}$$

1- J. E. Shigley and C. R. Mischke, ibid.

و υ_j ، E_j و d_j که $j = 1,2,3,...$ به ترتیب ضریب پواسان، مدول یا نگ و قطر دو استوانه می‌باشند. این معادلات را به زبان مطلب بنویسید. سپس آنها را با مقادیر زیر مقایسه کنید:

$\upsilon_1 = \upsilon_2 = 0.3$ ، $E_1 = E_2 = 3\times10^7$ ، $d_1 = 1.5$ ، $d_2 = 2.75$ ، $F = 100lb$ ، $L = 2$ و $z = 0.01in$. این عبارات را برای استفاده در تمرین ٦-٥ ذخیره نمایید. {پاسخ: $b = 0.0014in$، $p_{max} = 23,251psi$ ، $\sigma_x = -7,085.7\,psi$ ، $\sigma_y = -4,843.8\,psi$ و $\sigma_z = -18,775\,psi$.}

۱-۳ عدد بار یک یاتاقان هیدرودینامیکی توسط رابطهٔ زیر داده شده است[1]:

$$N_L = \frac{\pi\varepsilon\sqrt{\pi^2\left(1-\varepsilon^2\right)+16\varepsilon^2}}{\left(1-\varepsilon^2\right)^2}$$

که ε نسبت خروج از مرکزی می‌باشد.

این معادله را به زبان مطلب بنویسید. سپس آن را با مقادیر زیر مقایسه کنید:

$\varepsilon = 0.8$. {پاسخ: $N_l = 72.022$ }

۱-۴ یک پیچ رزوه‌دار با ارتفاع h را در نظر بگیرید که مدول یا نگ مادهٔ تشکیل‌دهندهٔ آن، E می‌باشد. سختی k پیچ را هنگامی که از داخل سوراخی با قطر d_o عبور می‌کند، می‌توان با عبارت زیر تخمین زد[2]:

$$k = \frac{\pi E d_o \tan 30^\circ}{\ln\dfrac{\left(d_2 - d_0\right)\left(d_1 + d_0\right)}{\left(d_2 + d_0\right)\left(d_1 - d_0\right)}}$$

که d_1 قطر واشر زیر پیچ و d_2 به صورت زیر می‌باشد:

$$d_2 = d_1 + h\tan 30^\circ$$

این معادلات را به زبان مطلب بنویسید. سپس آنها را با مقادیر زیر مقایسه کنید. توجه کنید که آرگومان توابع مثلثاتی باید بر حسب رادیان باشد.

1- R. L. Norton, Machine Design, An Integrated Approach, Prentice-Hall, Upper Saddle River, NJ, 1996.

2- A. H. Burr and J. B. Cheatham, Mechanical Analysis and Design, 2nd ed. , Prentice Hall, Upper Saddle River, NJ, 1995, p. 423.

$\{k = 2.8842 \times 10^7\ lb/in\ \ d_2 = 1.3467in$:پاسخ. $E = 3\times10^7$ ، $d_1 = 0.625$ ، $d_o = 0.25$ ، $h = 1.25$

۱-۵ تنشهای شعاعی و مماسی در لوله ای طویل، که سطح داخلی آن به شعاع a و دمای T_a و سطح خارجی آن به شعاع b و دمای T_b می باشند، به ترتیب عبارتند از[1] :

$$\sigma_r = \frac{\alpha E(T_a - T_b)}{2(1-\upsilon)ln(b/a)}\left[\frac{a^2}{b^2-a^2}\left(\frac{b^2}{r^2}-1\right)ln\left(\frac{b}{a}\right)-ln\left(\frac{b}{r}\right)\right]$$

$$\sigma_t = \frac{\alpha E(T_a - T_b)}{2(1-\upsilon)ln(b/a)}\left[1-\frac{a^2}{b^2-a^2}\left(\frac{b^2}{r^2}+1\right)ln\left(\frac{b}{a}\right)-ln\left(\frac{b}{r}\right)\right]$$

که در آن r مختصات شعاعی لوله، E مدول یانگ مربوط به جنس لوله و α ضریب انبساط حرارتی لوله می‌باشند.

توزیع درجه حرارت دیوارهٔ لوله در جهت شعاعی به شکل زیر خواهد بود:

$$T = T_b + \frac{(T_a - T_b)ln(b/r)}{ln(b/a)}$$

این معادلات را به زبان مطلب بنویسید. سپس آنها را با مقادیر زیر مقایسه کنید:

$$b = 0.5\ \ ،a = 0.25\ \ ،T_b = 300\ \ ،T_a = 500\ \ ،\upsilon = 0.3\ \ ،E = 3\times10^7\ \ ،\alpha = 1.2\times10^{-5}$$

$\{T = 383.0075$ ،$\sigma_t = 5,231.9$ ،$\sigma_r = -8,011.5$:پاسخ} .$r = 0.375$ و

۱-۶ مدت زمانی که نیاز است ارزش مقداری پول به اندازهٔ p ، که سرمایه نامیده می‌شود، به اندازهٔ $r_p p$ افزایش یابد، در حالیکه ارزش پول n بار در سال و در هر بار به اندازهٔ $i\%$ افزایش می‌یابد (ni افزایش ارزش پول در یک سال است) مطابق رابطهٔ زیر خواهد بود:

$$T = \frac{ln\, r_p}{n\,ln(1+0.01i)}\ \text{سال}$$

این معادله را به زبان مطلب بنویسید. سپس آن را با مقادیر زیر مقایسه کنید:

$$r_p = 2\ \ ،i = 0.5\ \ ،n = 12$$.{پاسخ: $T = 11.5813$ سال}

1- A. H. Burr and J. B. Cheatham, ibid. , p. 496.

۱-۷ دبی جرمی یک گاز در حال خارج شدن از یک منبع در فشار p_0 و تحت شرایط آدیاباتیک بازگشت پذیر متناسب با رابطهٔ زیر خواهد بود[۱]:

$$\psi = \sqrt{\frac{k}{k-1}}\sqrt{\left(\frac{p_e}{p_0}\right)^{2/k} - \left(\frac{p_e}{p_0}\right)^{(k+1)/k}}$$

که در آن p_e، فشار بیرونی مربوط به خروجی منبع و k ثابت فرآیند آدیاباتیک بازگشت پذیر مربوط به گاز می‌باشد.

این معادله را به زبان مطلب بنویسید. سپس آن را با مقادیر زیر مقایسه کنید:

$k = 1.4$ ، $\frac{p_e}{p_0} = 0.3$. {پاسخ: $\psi = 0.4271$}

۱-۸ ضریب تخلیهٔ جریان از میان یک کانال باز با مقطع سهموی به صورت زیر می‌باشد[۲]:

$$K = \frac{1.2}{x}\left[\sqrt{16x^2+1} + \frac{1}{4x}ln\left(\sqrt{16x^2+1}+4x\right)\right]^{-2/3}$$

که در آن x نسبت ماکزیمم عمق آب به عرض کانال در سطح آب می‌باشد.

این معادله را به زبان مطلب بنویسید. سپس آن را با مقادیر زیر مقایسه کنید: $x = 0.45$. {پاسخ: $K = 1.3394$}

۱-۹ نشان دهید که با فرمول زیر[۳]، کاربر می‌تواند عدد π را با محاسبهٔ تنها یک عبارت فرمول ($n = 0$) با خطای 10^{-7} و با محاسبهٔ دو عبارت آن ($n = 0,1$) با خطای 10^{-15} محاسبه کند. در حقیقت به ازای هر عبارت اضافی محاسبه شده، تقریبا خطای عدد π با ضریب 10^{-8} کاهش خواهد یافت. بنابراین، مجموع چهار جملهٔ اول ($n = 0,1,2,3$) فرمول زیر، قادر است عدد π را تا ۳۱ رقم اول آن ارائه دهد، و این حقیقتی است که تنها به کمک جعبه ابزار سیمبولیک مطلب می توان آنرا اثبات کرد.

1- W. Beitz and K. H. Kuttner, Eds. , Handbook of Mechanical Engineering, Springer-Verlag, NewYork, 1994, p. C15.

2- H. W. King, Handbook of Hydraulics, 4th ed. McGraw-Hill, NY, 1954, p. 7-24.

3- S. Ramanujan, "Modular equations and approximations to π," Quart. J. Math, 45, pp. 350-372.

$$\frac{1}{\pi} = \frac{\sqrt{8}}{9801} \sum_{n=0}^{\infty} \frac{(4n)!(1103 + 26390\, n)}{(n!)^4\, 396^{4n}}$$

توجه : فاکتوریل با استفاده از تابع gamma به شکل زیر محاسبه می‌شود:

$gamma(n+1) = n!$ ، $gamma(4n+1) = (4n)!$ و $n = 0,1,2,...$.

ضمیمهٔ A

خلاصه ای از کاراکترهای خاص مطلب

جدول A- الف (کاراکترهای خاص مطلب و خلاصه ای در مورد نحوهٔ استفاده از آنها)

کاراکتر	نام	استفاده
.	نقطه	(a) نقطه اعشار. (b) قسمتی از عملگرهای ریاضیاتی جهت نشان دادن نوع خاصی از عملیات برداری و یا ماتریسی، که عملیات نقطه‌ای نامیده می‌شود، مثل c=a.*b.
،	کاما	(a) جدا کنندهٔ آرگومان‌های ماتریس‌ها، مثل b(2,7) و توابع، مثل besselj(l,x) و یا عناصر داخل قلاب‌های سازندهٔ بردارها، مثل v=[l,x] و یا آرگومان‌های خروجی توابع مثل [x,s]=max(a).
؛	نقطهٔ ویرگول	(a) هنگامی که در انتهای عبارتی بکار رود نمایش نتایج را حذف می‌کند. (b) بیانگر انتهای یک سطر در هنگام ایجاد یک ماتریس می‌باشد. مثل [x m= y z ; a b c]
:	دو نقطه	(a) جدا کننده در دستورات ساخت بردار، مثل $x = a:b:c$. (b) برای ماتریس z علامت دو نقطه نشان دهندهٔ همهٔ سطرها اگر به شکل $z(:,k)$ و یا نشان دهندهٔ همهٔ ستونها اگر به شکل $z(k,:)$ نوشته شود، خواهد بود.

( )	پرانتز	(a) نشان دهندهٔ اندیس یک عنصر ماتریس z می باشد. مثلاً $z(j,k)$ مبین عنصر سطر j ام و ستون k ام ماتریس z خواهد بود. (b) تعیین کنندهٔ حوزهٔ محاسبات در عبارات ریاضی مثل $a\hat{}(b+c)$ می‌باشد. (c) در بر گیرندهٔ آرگومان توابع مثل $sin(x)$ می‌باشد.
[ ]	قلاب	یک رشته از اعداد یا حروف را به صورت بردار و یا ماتریس تولید می‌کند.
{ }	آکولاد	یک ساختار و یا ماتریس سلولی را ایجاد می‌کند.
%	درصد	محدود کنندهٔ توضیحات ؛ برای مشخص کردن شروع توضیحات استفاده می‌شود که کامپایلر مطلب هر آنچه را پس از آن بیاید، نادیده می‌گیرد. تنها یک مورد استثناء وجود دارد و آن هنگامی که علامت درصد در داخل یک جفت کوتیشن جهت تعریف رشته ای بودن یک عبارت استفاده شود. مثل: $a = 'p1 = 14\%$ of the total.'.
‘	کوتیشن	(a) 'Expression' نشان می‌دهد که Expression یک رشته (حرفی) می‌باشد. (b) نشاندهندهٔ ترانهاده (ترانسپوز) یک بردار و یا ماتریس می‌باشد.
...	سه نقطه	نشاندهندهٔ ادامهٔ یک عبارت مطلب در خط بعد که برای خواناتر شدن برنامه بکار می‌رود.
	جای خالی	وابسته کنندهٔ متنی: یا در نظر گرفته نمی‌شود که نشان دهندهٔ فاصله در عبارت ساخت داده‌ها مثل $c = [a\ b]$ است و یا به عنوان یک کاراکتر در عبارات رشته‌ای در نظر گرفته می‌شود.

فصل دوم

ماتریسها و مطلب

در این فصـــل نحوهٔ استفاده از دستورات مطلب در ارتباط با بـــردارها، ماتریســـها و عملیات بین آنـــها بیان خواهد شـــد.

۲- ۱ مقدمه

مطلب (MATLAB) زبانی است که ساختارهای عملکردی و دستورات اجرایی آن بر اساس مجموعه‌ای از اصول عملیات ماتریسی و کاربردهای آنها بنا شده است. بنابراین جهت بهره برداری کامل از مزایای زبان مطلب، برخی از تعاریف اساسی ماتریس و نشانه‌های آنها را بطور خلاصه ارائه می‌کنیم و کاربردهای آنها را در قالب چند مثال نشان خواهیم داد. مطالب بیان شده در این فصل بطور گسترده‌ای در فصل های بعدی استفاده خواهند شد.

۲- ۲ ماتریسها و بردارها

آرایهٔ a شامل m سطر و n ستون، یک ماتریس از مرتبهٔ $(m \times n)$ نامیده می‌شود و کلاً شامل mn عنصر می‌باشد که در آرایهٔ زیر قرار گرفته اند:

$$a = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & & \\ \vdots & \ddots & & \\ a_{m1} & & & a_{mn} \end{bmatrix}$$

عناصر ماتریس به شکل a_{ij} نشان داده می شود. که i شمارهٔ سطر و j شمارهٔ ستون می‌باشد. مرتبهٔ ماتریس a توسط دستور $Size$ به دو شکل؛

```
Size(a)
```

و

```
[m,n]=size(a)
```

تعیین می‌شود.

در شکل اول دو مقدار نشان داده خواهد شد: مقدار اول تعداد سطر ها (m) و دومی تعداد ستونها (n) می‌باشد. در شکل دوم، m مقداری است که به تعداد سطرها نسبت داده می‌شود و n عددی مبین تعداد ستونهاست. این مطلب به صورت کاملتر در بخش ۵- ۲ توضیح داده شده است.

چند حالت خاص از این ماتریس در زیر نشان داده شده است.

۲-۲- ۱ ماتریس مربعی

در ماتریس فوق هنگامیکه $m = n$ باشد، ماتریس تبدیل به ماتریس مربعی می شود.

۲-۲- ۲ ماتریس قطری

هنگامیکه به ازای $i \neq j$، $a_{ij} = 0$ باشد؛ ماتریس قطری به شکل زیر خواهیم داشت:

$$a = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & & \\ \vdots & & \ddots & \\ 0 & & & a_{mn} \end{bmatrix}$$

بنابراین اگر $a = [1\,3\,8]$ باشد (بخش ۲- ۳ را در مورد ایجاد بردار مطالعه کنید) خواهیم داشت:

```
a = [ 1 3 8 ]
z = diag(a)
```

که پاسخ آن ماتریس (3×3) زیر می باشد:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 8 \end{bmatrix}$$

که در مطلب به صورت زیر نمایش داده می‌شود:

$$z = \begin{matrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 8 \end{matrix}$$

به بیان دیگر اگر b یک ماتریس مربعی باشد، آنگاه عبارت:

```
diag(b)
```

مبین عناصر قطری ماتریس b می باشد.

هنگامیکه $a_{ii} = 1$ و $m = n$ باشد، آنگاه ماتریس یکهٔ I به شکل زیر خواهد بود:

$$I = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & & \\ \vdots & & \ddots & \\ 0 & & & 1 \end{bmatrix}$$

دستور مطلب برای ایجاد یک ماتریس یکانی به شکل زیر می باشد:

```
eye(n)
```

که در آن n ، مرتبهٔ ماتریس مربعی است. بنابراین دستور زیر:

```
c=eye(3)
```

خروجی زیر را نتیجه می دهد:

$$C =$$

$$\begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$$

۲-۲- ۳ ماتریسهای ستونی و سطری (بردارها)

هنگامیکه $a_{ij} = a_{i1}$ باشد، a ماتریس ستونی و یا در اصطلاح بردار خوانده می‌شود که به صورت زیر است:

$$a = \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{ml} \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix}$$

اما وقتی که $a_{ij} = a_{1j}$ یعنی ماتریس a تنها یک ردیف داشته باشد، در این‌صورت آنرا یک ماتریس سطری و یا یک بردار می‌نامیم؛ که به صورت زیر می باشد:

$$a=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \end{bmatrix}=\begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix}$$

که این شیوهٔ نمایش یک بردار در مطلب می باشد.

۲-۲-۴ ترانهادهٔ یک ماتریس و یا یک بردار

ترانهادهٔ یک ماتریس در مطلب توسط علامت (') مشخص شده و به شکل زیر تعریف مـــی‌گردد. اگـــر a ماتریسی از مرتبهٔ $(m\times n)$ باشد،داریم:

$$a=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & & \\ a_{m1} & & \ddots & a_{mn} \end{bmatrix}$$

ترانهادهٔ آن که ماتریسی $(n\times m)$ است، به شکل زیر بیان می شود:

$$w=a=\begin{bmatrix} w_{11}=a_{11} & w_{12}=a_{21} & \cdots & w_{1m}=a_{m1} \\ w_{21}=a_{12} & w_{22}=a_{22} & & \\ \vdots & & & \\ w_{n1}=a_{1n} & \cdots & \ddots & w_{nm}=a_{mn} \end{bmatrix}$$

ترانهادهٔ بردارهای سطری و ستونی به شکل زیر است:

اگر :

$$a'=\begin{bmatrix} a_1 & a_2 & \cdots & a_m \end{bmatrix}$$

و، اگر:

$$a'=\begin{bmatrix} a_{11} \\ a_{12} \\ \vdots \\ a_{1m} \end{bmatrix}$$

طول بردار a را که تعداد عناصر آن می باشد, می توان با بکار بردن دستور؛

```
L = Length(a)
```

و یا

```
L = Size(a)
```

بدست آورد.

در حالیکه مرتبهٔ یک ماتریس را تنها می توان با بکار بردن دستور *size* تعیین نمود. مزیت استفاده از دستور *size* اینست که نیازی به دانستن اینکه a یک بردار است و یا یک ماتریس، نمی باشد. اما برای بردارها استفاده از دستور *Length* آسانتر خواهد بود.

۲- ۳ ایجاد بردار

بردارها را در مطلب می توان به دو شکل زیر نوشت:

$$f = [a\ x\ b\ ...] \text{ و یا } f = [a,x,b,...]$$

که a، x، b و... متغیر، عدد،دستور و یا متغیر رشته‌ای می‌باشند (بخش ۳- ۲ را ملاحظه کنید). اگر متغیر و یا دستور باشند، همهٔ متغیرها باید قبل از اجرای این دستورات توسط مقادیر عددیی که برای هر یک از این آنها بدست آمده است، مقداردهی شوند.

عبارات و اعداد می‌توانند بصورت هر ترکیب و ترتیبی ظاهر شوند. در فرم زیر:

$$f = [a\ x\ b\ ...]$$

فضای خالی بین نشانه‌ها لازم است در حالیکه در فرم؛

$$f = [a,x,b,...]$$

این مسئله کاملاً اختیاری است.

ذکر این مطلب ضروری است که اگر a یک عبارت محاسباتی باشد، مستقیماً در مکان a در عبارات فوق قرار می‌گیرد؛ در اینصورت فضای خالی بین کاراکتر های عددی- حرفی و عملگرهای ریاضیاتی وجود نخواهد داشت. برای مثال اگر $a = h + d^g$ باشد، در این صورت f به شکل؛

$$f = [h + d\hat{}g\ x\ b\ ...]$$

و یا

$$f = [h + d\hat{}g, x, b, ...]$$

نوشته می شود.

مطلب چندین شیوهٔ دیگر برای مقداردهی اولیه به عناصر یک بردار و یا ماتریس ارائه می‌دهد (روشهای ایجاد ماتریس در بخش ۲ – ٤ آمده است.) اولین روش که در ادامه بیان شده است از علامت دو نقطه (:) جهت تعیین بازهٔ مقادیر و افزایش بین دو مقدار متوالی استفاده می‌کند. در این روش، افزایش بین دو مقدار متوالی باید مشخص شود. در روش دوم از بازهٔ مقادیر و تعداد مقادیر دلخواه استفاده می شود، در اینجا تعداد مقادیر دارای اهمیت می باشد. با استفاده از علامت دو نقطه جهت ایجاد یک بردار، مطابق زیر خواهیم داشت:

```
x=s:d:f
```

که در آن، s نقطهٔ شروع و یا مقدار اولیه،d قدر نسبت افزایش و یا کاهش، f نقطهٔ اتمام و یا آخرین مقدار می‌باشد.

بنابراین بردار سطری x همانطورکه در زیر نشان داده شده است، ساخته می‌شود:

$$x = \begin{bmatrix} s & s+d & s+2d \dots & s+nd \end{bmatrix}$$

که در آن $s+nd \le f$ می‌باشد. توجه کنید که تعداد مقادیر n که برای ساختن x مورد نیاز است مستقیماً مشخص نشده است. مقادیر f ، d و s می توانند هر ترکیبی از مقادیر عددی، متغیرها و یا عبارات محاسباتی باشند. تعداد جملاتی که توسط عبارات فوق ایجاد شده است با دستور زیر مشخص می‌شود:

```
n =length (x)
```

اگر d حذف شود، مطلب آنرا یک فرض می کند. بنابراین دستور:

```
x=s:f
```

بردار زیر را ایجاد می کند:

$$x = \begin{bmatrix} s, & s+1, & s+2, \dots, & s+n \end{bmatrix}$$

که در آن $s+n \le f$ می باشد. مجدداً f و s می توانند هر ترکیبی از مقادیر عددی، متغیرها و یا عبارات محاسباتی باشند.

از طرف دیگرکاربر می تواند n مقدار با فاصلهٔ یکسان که با s شروع و به f ختم می‌شوند را با استفاده از دستور زیر ایجاد کند:

```
x =linspace(s,f,n)
```

که در آن فاصلهٔ بین دو مقدار متوالی توسط رابطهٔ زیر در مطلب محاسبه می شود:

$$d = \frac{f-s}{n-1}$$

مقادیر f و s می توانند مثبت و یا منفی و همچنین $s > f$ یا $s < f$ باشد. هرگاه مقداری عددی برای n تعیین نشده باشد، مقدار آن بطور پیش فرض 100 درنظر گرفته می‌شود. بنابراین دستور *linspace* بردار زیر را ایجاد خواهد کرد:

$$x = [s \;\; s+d \;\; s+2d \;\; \ldots \;\; f = s+(n\text{-}1)d]$$

اگر نیاز به فاصلهٔ یکسان در مقیاس لگاریتمی باشد، در اینصورت از دستور زیر استفاده می‌شود:

```
x =logspace( s,f,n )
```

که در آن اولین مقدار 10^s ، آخرین مقدار 10^f و d همان مقدار قبل می باشد. بنابراین؛

$$x = [10^s \quad 10^{s+d} \quad 10^{s+2d} \quad \ldots \quad 10^f]$$

اگر مقدار n مشخص نشده باشد، مطلب مقدار n را 50 در نظر خواهد گرفت.

برای آشنایی بیشتر با رفتار بردارها در مطلب، عبارت زیر را در نظر بگیرید:

$$b = [b_1 \;\; b_2 \;\; b_3 \ldots b_n]$$

این عبارت به این معناست که برداری تحت عنوان b ساخته ایم که دارای یک سطر و n ستون می باشد. برای نمایش b_3، در مطلب آنرا بصورت $b(3)$ می نویسیم. در این صورت مطلب مقدار عددی که قبلاً به $b(3)$ نسبت داده شده است را نشان خواهد داد؛ که همان عنصر سوم بردار b می‌باشد. در اینجا مطلب توانایی تشخیص $(1\times n)$ بودن ماتریس b را دارد و در برخی از موارد لزوم استفاده از دو اندیس را از میان می برد. یعنی نوشتن $b(3)$ هنگامیکه b یک بردار است همانطورکه قبلاً بیان شد، مشابه نوشتن $b(1,3)$ خواهد بود. اگر کاربر عمداً و یا سهواً عبارت $b(3,1)$ را تایپ کند، پیغام خطایی ظاهر خواهد شد؛ زیرا این سطر (سطر سوم) تعریف نشده است.

بر عکس اگر b را به شکل زیر در نظر بگیریم:

$$b = [b_1\ b_2\ b_3\ \ldots b_n]'$$

یک بردار ستونی ایجاد کرده ایم که یک ماتریس $(n \times 1)$ می‌باشد. درصورتیکه بخواهیم مولفهٔ سوم این بردار را فراخوانی کنیم مجدداً عبارت *b(3)* را تایپ می کنیم و مطلب مقدار عددی متناظر با b_3 را نمایش خواهد داد. که این کار معادل نوشتن عبارت *b(3,1)* خواهد بود. اگر کاربر عمداً یا سهواً عبارت *b(1,3)* را تایپ کند در اینصورت پیغام خطایی ظاهر خواهد شد؛ زیرا این ستون (ستون سوم)، تعریف نشده است.

فرض کنید که می خواهیم بردار x که دارای هفت مقدار *[-2,1,3,5,7,9,10]* می باشد را ایجاد کنیم. اینکار را با استفاده از دستور:

$$x = [-2\ \ 1 : 2 : 9\ \ 10]$$

و یا

$$x = [-2,\ 1,\ 3,\ 5,\ 7,\ 9,\ 10\ ]$$

انجام خواهیم داد. به این معنا که عناصر این بردار عبارت از: $x_1 = -2$، $x_2 = 1$، $x_3 = 3$، $x_4 = 5$، $x_5 = 7$، $x_6 = 9$ و $x_7 = 10$ خواهند بود و *length(x)=7* می باشد. می توانیم به عناصر بردار x توسط عبارت مطلب $x(j), j = 1,2, \ldots, 7$ دست یابیم. برای مثال عبارت $x(5)$ مقدار *7* را نمایش خواهد داد.

هنگامیکه یک عدد را با یک بردار جمع و یا از آن کم می کنیم، عدد به تمام عناصر بردار اضافه و یا از تمام عناصر آن کم خواهد شد. بنابراین عبارت:

$$z = x - 1$$

بردار $z = [-3\ 0\ 2\ 4\ 6\ 8\ 9]$ را نتیجه خواهد داد. اما قوانین مربوط به ضرب و تقسیم و توان دارای چند محدودیت می‌باشند. (بخش ۲- ٦ را ملاحظه کنید).

از طرف دیگر ممکن است تمایل به اصلاح تنها چند عضو یک بردار را داشته باشیم. برای مثال فرض کنید $z = [-2\ 1\ 3\ 5\ 7\ 9\ 10]$ باشد. در این صورت برای اینکه عضو دوم ماتریس z را بر *2* تقسیم کنیم خواهیم داشت:

$$z = [-2\ \ 1\ \ 3\ \ 5\ \ 7\ \ 9\ \ 10];$$

$$z(2) = z(2)/2;$$

که بردار $z = [-2\ 0.5\ 3\ 5\ 7\ 9\ 10]$ را ارائه خواهد کرد. اگر علاوه بر اصلاح فوق عناصر سوم و چهارم بردار z را در 3 ضرب کنیم و سپس 1 را از آنها کم کنیم؛

$$z = [-2\ \ 1\ \ 3\ \ 5\ \ 7\ \ 9\ \ 10]\ ;$$

$$z(2) = z(2) / 2;$$

$$z(3:4) = z(3:4) * 3 - 1;$$

در نتیجه بردار z به $z = [-2\ 0.5\ \ 8\ \ 14\ \ 7\ \ 9\ \ 10]$ تبدیل خواهد شد. توجه داشته باشید که سایر عناصر بردار z بدون تغییر باقی می مانند.

روشهای دیگری جهت دستیابی کاربر به عناصر یک بردار وجود دارد.

بردار هشت عضوی زیر را در نظر بگیرید:

$$y = [-1\ \ 6\ \ 15\ \ -7\ \ 31\ \ 2\ \ -4\ \ -5\ ];$$

اگر کاربر بخواهد بردار جدید x که شامل عناصر سوم تا پنجم بردار y می شود را ایجاد کند، می‌تواند از دستور زیر استفاده نماید:

$$x = y(3:5)$$

که بردار سه عضوی $x = [15\ -7\ \ 31]$ ایجاد خواهد شد.

فرض کنید که می خواهیم بردار x را مشتمل بر دو عضو اول و دو عضو آخر بردار y ایجاد کنیم؛ اینکار را می توان توسط دستور:

$$x = [\,y(1)\ y(2)\ \ y(7)\ \ y(8)]$$

و یا

$$x = y([1\ 2\ \ 7\ \ 8])$$

انجام داد و یا ابتدا بردار $index$ را تعریف کرده و سپس از آن به شکل زیر استفاده می کنیم؛

$$index = [1\ \ 2\ \ 7\ \ 8];$$

$$x = y(index)$$

دو شکل آخر دستور فوق کاربردهای زیادی دارند. حال متناظر با بردار y بردار z که دارای هشت عضو می باشد را به شکل زیر تعریف می کنیم:

$$y = [-1 \;\; 6 \;\; 15 \;\; -7 \;\; 31 \;\; 2 \;\; -4 \;\; -5];$$

$$z = [10 \;\; 20 \;\; 30 \;\; 40 \;\; 50 \;\; 60 \;\; 70 \;\; 80];$$

فرض کنید می‌خواهیم بردار y را به ترتیب نزولی (منفی ترین به مثبت ترین) با استفاده از تابع *sort* مرتب کنیم و سپس ترتیب عناصر بردار z را متناظر با ترتیب جدید عناصر بردار y تغییر دهیم. با استفاده از *Help sort* متوجه می‌شویم که یکی از اشکال تابع *sort* به شکل زیر می‌باشد:

```
[ynew,indx]= sort(y)
```

که در آن *ynew* برداری شامل عناصر مرتب شدهٔ y و *indx* برداری شامل موقعیت اصلی عناصر y می‌باشد. بنابراین برنامهٔ زیر؛

$$y = [-1 \; 6 \;\; 15 \;\; -7 \;\; 31 \;\; 2 \;\; -4 \;\; -5];$$

$$z = [10 \;\; 20 \;\; 30 \;\; 40 \;\; 50 \;\; 60 \;\; 70 \;\; 80];$$

```
[ynew,indx]= sort(y)
```

$$znew = z(indx)$$

پس از اجرا، پاسخ زیر را در بر خواهد داشت:

$$ynew \rightarrow [-7 \; -5 \; -4 \; -1 \; 2 \; 6 \; 15 \; 31]$$

$$indx \rightarrow [4 \; 8 \; 7 \; 1 \; 6 \; 2 \; 3 \; 5]$$

$$znew \rightarrow [40 \; 80 \; 70 \; 10 \; 60 \; 20 \; 30 \; 50]$$

بنابراین می بینیم که $indx(1) = 4$ است، یعنی $ynew(1)$ قبلاً همان $y(4)$ بوده است. لذا برای بدست آوردن بردار z متناظر، به سادگی *znew* را به عنوان بردار z که اندیسهایش توسط بردار *indx* تعیین شده است، تعریف می‌کنیم.

می‌توانیم این قابلیت را با معرفی دستور *find* توسعه دهیم. که مکان (نه مقدار) تمامی عناصر یک بردار (یا ماتریس) را که شرایط یا دستورات ویژهٔ تعیین شده از سوی کاربر را ارضاء می‌کند، تعیین خواهد کرد. کاربردهای این دستور را با ایجاد بردار جدید s که شامل عناصری از بردار y می‌شود که منفی و یا صفر هستند، بیان خواهیم کرد. عملگر رابطه‌ای =< در مطلب مبین $\leq$ می‌باشد (جدول ٤-١ را ملاحظه کنید). بنابراین؛

```
y = [-1 6 15 -7 31 2 -4 -5];
indxx = find(y<=0)
```

خروجی زیر را در برخواهد داشت:

$$indxx \rightarrow [14\ 7\ 8]$$

لذا با تایپ کردن دستورات زیر،

```
y = [-1 6 15 -7 31 2 -4 -5];
indxx = find(y<=0) ;
s = y(indxx)
```

نتیجهٔ زیر حاصل خواهد شد:

$$s \rightarrow [-1\ -7\ -4\ -5]$$

این دستورات می‌توانند به صورت فشرده تر به شکل زیر نوشته شوند:

```
y = [-16 15 -7 31 2 -4 -5];
s = y( find( y <= 0 ))
```

(بخش ٦-٣-٨ را جهت دیدن کاربرد دیگر دستور *find* ملاحظه کنید.)

یکی از مزایای عمدهٔ نمایش ماتریسی و برداری غیر صریح در مطلب اینست که کاربر می‌تواند مجموعه ای از عملیات را روی آرایه‌ای از اعداد با شیوه ای سریع و فشرده اعمال کند. فرض کنید می‌خواهیم عبارت *sin(x)* را هنگامیکه x در بازهٔ $-\pi \leq x \leq \pi$ با قدر نسبت $\pi / 5$ تغییر می‌کند محاسبه کنیم، در اینصورت از دستورات زیر استفاده می‌کنیم:

$$x = -pi : pi/5 : pi;$$

$$y = sin(x)$$

بردار زیر که شامل $n = length(y) = 11$ عنصر می‌باشد، حاصل می‌شود:

$y \rightarrow$[0.0000 -0.5878 -0.9511 -0.9511 -0.5878 0.0000 0.5878 0.9511 0.9511

0.5878 0.0000]

۲-۴ ایجاد ماتریس

ماتریس $a\ (4\times 3)$؛

$$a = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$

می تواند به هر یک از شیوه‌های زیر ساخته شود. سه کمیت a_{11}، a_{12} و a_{13} را در نظر بگیرید. این مقادیر می‌توانند با هم ترکیب شده و تشکیل بردار v_1 را به شکل زیر بدهند:

$$v_1 = \begin{bmatrix} a_{11} & a_{12} & a_{13} \end{bmatrix}$$

به شیوهٔ مشابه، می توانیم سه بردار v_2، v_3 و v_4 را به شکل زیر ایجاد کنیم:

$$v_2 = \begin{bmatrix} a_{21} & a_{22} & a_{23} \end{bmatrix}$$

$$v_3 = \begin{bmatrix} a_{31} & a_{32} & a_{33} \end{bmatrix}$$

$$v_4 = \begin{bmatrix} a_{41} & a_{42} & a_{43} \end{bmatrix}$$

حال با استفاده از این چهار بردار ماتریس a را به شکل زیر خواهیم ساخت:

$$a = \begin{bmatrix} v_1 ; v_2 ; v_3 ; v_4 \end{bmatrix}$$

که علامت های نقطه ویرگول مبین انتهای هر سطر ماتریس می‌باشند. هر ماتریس باید تعداد ستون یکسان داشته باشد. ماتریس a را می توان مستقیماً با بکار بردن دستور زیر ایجاد نمود؛

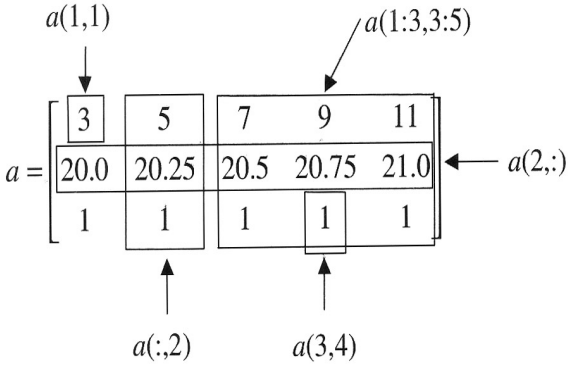
$$a = [a_{11}\, a_{12}\, a_{13}; a_{21}\, a_{22}\, a_{23}; a_{31}\, a_{32}\, a_{33}; a_{41}\, a_{42}\, a_{43}]$$

یا اگر یک دستور واضح تر مورد نظر باشد، می توان از دستور زیر استفاده نمود:

$$\begin{aligned} a = [\, & a_{11}\, a_{12}\, a_{13};\ldots \\ & a_{21}\, a_{22}\, a_{23};\ldots \\ & a_{31}\, a_{32}\, a_{33};\ldots \\ & a_{41}\, a_{42}\, a_{43}\,] \end{aligned}$$

که در آن استفاده از علامت(...) ضروری می باشد. (جدول A- الف را در ضمیمهٔ انتهای فصل اول مشاهده کنید). روش دیگر جهت تعیین انتهای هر سطر استفاده از کلید *Enter* می‌باشد:

```
a = [a11 a12 a13
a21 a22 a23
a31 a32 a33
a41 a42 a43];
```



شکل ۲-۱) چگونگی دستیابی به عناصر یک ماتریس

در همهٔ اشکال فوق، a_{ij} می تواند عدد, متغیر، عبارت محاسباتی و یا رشته‌ای از حروف باشد. اگر a_{ij} متغیر و یا عبارات محاسباتی باشد، در اینصورت متغیر های ساده و یا متغیرهایی که شامل عبارات محاسباتی می‌باشند، باید توسط مقادیر عددی، به وسیلهٔ کاربر و یا از عبارتهای محاسباتی اجرا شدهٔ قبلی، قبل از اجرای عبارتهای فعلی، مقداردهی شوند. عبارات محاسباتی و اعداد می‌توانند در هر ترکیبی ظاهر شوند. اگر متغیر رشته‌ای باشند، در این صورت تعداد کاراکترهای هر سطر باید یکسان باشد. (بخش ۳ - ۱ را ملاحظه کنید).

دو تابع کاربردی دیگر که جهت ایجاد داده برای عناصر یک ماتریس استفاده می‌شوند عبارتند از:

```
one = ones (r,c)
```

که یک ماتریس $(r \times c)$ ایجاد کرده که عناصر آن دارای مقدار 1 می باشند، و دستور:

```
zero = zeros (r,c)
```

یک ماتریس $(r \times c)$ ایجاد می‌کند که عناصر آن دارای مقدار 0 می‌باشند. این توابع را می توان به سادگی جهت تعویض عبارتهای محاسباتی معادل بکار برد مثل: $one(1:r,1:c)=1$ و $zero(1:r,1:c)=0$.

بنابراین دستور،

```
on = ones (2,5)
```

ماتریس (5×2) زیر را ایجاد خواهد کرد:

$$\begin{matrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{matrix}$$

و دستور؛

```
zer = zeros (3,2)
```

ماتریس (3×2) دارای عناصر صفر، مطابق زیر ایجاد می کند:

$$\begin{matrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{matrix}$$

اکنون ساختار ماتریسی (5×3) زیر را در نظر بگیرید:

$$a=\begin{bmatrix} 3 & 5 & 7 & 9 & 11 \\ 20.0 & 20.25 & 20.5 & 20.75 & 21.0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

این ماتریس توسط دستور زیر ساخته می شود؛

$$a=[3:2:11;\ linspace(20,21,5);\ ones(1,5)]$$

که نتیجهٔ آن بصورت زیر خواهد بود:

3.0000 5.0000 7.0000 9.0000 11.0000
20.0000 20.2500 20.5000 20.7500 21.0000
1.0000 1.0000 1.0000 1.0000 1.0000

ما می‌توانیم به عناصر این ماتریس همانگونه که در شکل۲-۱ نشان داده شده است، دسترسی پیدا کنیم. بنابراین،

a (1,1) → 3

a (3,4) → 1

a (:,2) → [5 20.25 1]′

a (2,:) → [20 20.25 20.5 20.75 21]

و

a (1:3,3:5) → [7 9 11; 20.5 20.75 21; 1 1 1]

که یک ماتریس (3×3) می باشد. دستور:

a (:,2)

یعنی "همهٔ سطرهای ستون دوم"، و دستور:

a (2,:)

یعنی"همهٔ ستونهای سطر دوم". همچنین کاربر می‌تواند از علامت دو نقطه (:) که در آن فاصلهٔ بین دو مقدار متوالی $1+$ می‌باشد استفاده کند. یعنی:

a (1:3,3:5)

که مفهوم آن، سطرهای اول تا سوم و ستونهای سوم تا پنجم ماتریس a می‌باشد. بنابراین اگر؛

$$b = a(1:3,3:5) = \begin{bmatrix} 7 & 9 & 11 \\ 20.5 & 20.75 & 21 \\ 1 & 1 & 1 \end{bmatrix}$$

در این صورت ماتریس جدید $b(3\times 3)$ را تشکیل داده‌ایم.

اکنون یک ماتریس جدید، هم اندازه با ماتریس a که تمامی عناصر آن دارای مقدار 4 هستند را توسط دستورات زیر تشکیل خواهیم داد:

$$a = [3:2:11; linspace(20,21,5); ones(1,5)];$$

$$z = 4 * ones(size(a))$$

که ماتریس زیر حاصل خواهد شد:

$$\begin{matrix} 4 & 4 & 4 & 4 & 4 \\ 4 & 4 & 4 & 4 & 4 \\ 4 & 4 & 4 & 4 & 4 \end{matrix}$$

می‌توانیم عناصر یک ماتریس را به شیوه‌ای مشابه که برای بردارها استفاده شد، تغییر دهیم.

با اجرای دستور زیر:

```
z = magic(4)
```

ماتریس زیر بدست خواهد آمد:

$$\begin{matrix} 16 & 2 & 3 & 13 \\ 5 & 11 & 10 & 8 \\ 9 & 7 & 6 & 12 \\ 4 & 14 & 15 & 1 \end{matrix}$$

تابع *magic* ماتریسی ایجاد خواهد کرد که مجموع عناصر همهٔ سطرها، ستونها، دو قطر اصلی و فرعی آن یکسان می‌باشد. برای یک ماتریس (4×4) این مقدار 34 است.

اکنون تمام عناصر سطر دوم را بر2 تقسیم می‌کنیم سپس تمامی عناصر ستون دوم را با ستون چهارم جمع کرده و حاصل را در ستون چهارم قرار می‌دهیم. اینکار توسط دستورات زیر انجام خواهد شد:

```
z = magic (4);
z (2,:) =z (2,: )/2:
z (:, 4) = z (:,4) + z (:, 2);
```

که نتیجهٔ آن به شکل زیر خواهد بود:

$$\begin{matrix} 16 & 2 & 3 & 15 \\ 2.5 & 5.5 & 5 & 9.5 \\ 9 & 7 & 6 & 19 \\ 4 & 14 & 15 & 15 \end{matrix}$$

برای صفر کردن عناصر روی قطر اصلی تابع اولیه از دستورات زیر استفاده خواهیم کرد:

```
z = magic (4) ;
z = z-diag (diag (z) )
```

که ماتریس زیر بدست می‌آید:

$$\begin{matrix} 0 & 2 & 3 & 13 \\ 5 & 0 & 10 & 8 \\ 9 & 7 & 0 & 12 \\ 4 & 14 & 15 & 0 \end{matrix}$$

برای تعویض عناصر قطری با مقدار5 از دستورات زیر استفاده می‌کنیم:

```
z = magic(4);
z = z-diag(diag(z))+5*eye(4)
```

که نتیجهٔ آن به صورت زیر می‌باشد:

$$\begin{matrix} 5 & 2 & 3 & 13 \\ 5 & 5 & 10 & 8 \\ 9 & 7 & 5 & 12 \\ 4 & 14 & 15 & 5 \end{matrix}$$

برای جایگزینی مقادیر 11، 54، 23 و 61 در عناصر قطری ماتریس z از دستورات زیر استفاده می‌کنیم:

```
z = magic(4);
z = z-diag(diag(z))+diag([11 23 54 61])
```

که نتیجهٔ آن به صورت زیر می باشد:

$$\begin{matrix} 11 & 2 & 3 & 13 \\ 5 & 23 & 10 & 8 \\ 9 & 7 & 54 & 12 \\ 4 & 14 & 15 & 61 \end{matrix}$$

مطلب از دو تابع جهت ایجاد ماتریس‌ها با تکرار یک مقدار اسکالر، بردار سطری و یا ستونی و یا یک ماتریس استفاده می کند. این دو تابع عبارتند از:

```
repmat
```

و

```
meshgrid
```

که از تابع *repmat* استفاده می‌کند. شکل کلی تابع *repmat* به شکل زیر است.

```
repmat( x,r,c )
```

که در آن x اسکالر، بردار و یا ماتریس می‌باشد و r و c به ترتیب تعداد دفعاتی می‌باشند که سطرها و ستونهای x تکرار می‌شوند. برای مثال، تابع *repmat* می‌تواند جهت ایجاد یک بردار سطری و یا ستونی که دارای عناصر یکسان می‌باشد، بکار رود. بنابر این اگر بخواهیم که یک بردار سطری مشتمل بر شش مقدار 45.72 ایجاد کنیم، از دستور زیر استفاده می‌کنیم:

$w = repmat(45.72,1,6)$

این عبارت با عبارت زیر معادل می باشد.

$w = [45.72,45.72, 45.72, 45.72, 45.72, 45.72]$

این بردار را می توان توسط دستور زیر نیز ایجاد نمود؛

$w(1,1:6) = 45.72$

اگر بخواهیم ماتریسی (3×3) با این مقادیر ایجاد کنیم، از دستور زیر استفاده می کنیم:

$w = repmat(45.72,3,3)$

که آنرا می‌توان با دستور زیر نیز جایگزین نمود:

$w(1:3,1:3) = 45.72$

هر یک از این دستورات نتیجهٔ زیر را در پنجرهٔ فرمان مطلب ارائه خواهد داد،

$$\begin{matrix} 45.7200 & 45.7200 & 45.7200 \\ 45.7200 & 45.7200 & 45.7200 \\ 45.7200 & 45.7200 & 45.7200 \end{matrix}$$

حال بردار S، به شکل زیر را در نظر بگیرید:

$s = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \end{bmatrix}$

دستور زیر؛

$v = repmat(s,3,1)$

معادلهای[1] عددی ماتریس v را به شکل زیر ایجاد خواهد کرد:

$$v = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ a_1 & a_2 & a_3 & a_4 \\ a_1 & a_2 & a_3 & a_4 \end{bmatrix}$$

۱- منظور ما از معادلهای عددی اینست که v_{ij} ها در مطلب دارای مقادیر عددیی که بـه آنهـا نسـبت داده شـده است می‌باشند. این شیوهٔ نوشتن در اینجا جهت بهتر نشان دادن عملکرد تـابع *repmat*، بـا اسـتفاده از نمـایش سیمبولیکی عناصر ماتریسهای جواب استفاده شده است.

یعنی دستور فوق سه سطر بردار v، که در اینجا هر سطر دارای چهار ستون می‌باشد را ایجاد خواهدکرد. دستور زیر:

$$repmat(s,3,2)$$

معادلهای عددی ماتریس v را به شکل زیر ایجاد می‌کند:

$$v=\begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_1 & a_2 & a_3 & a_4 \\ a_1 & a_2 & a_3 & a_4 & a_1 & a_2 & a_3 & a_4 \\ a_1 & a_2 & a_3 & a_4 & a_1 & a_2 & a_3 & a_4 \end{bmatrix}$$

از طرف دیگر، دستور؛

$$\upsilon = repmat(s',1,3)$$

ماتریسی با سه ستون متشکل از بردار ستونی s'، که هر ستون در این مورد دارای چهار سطر می‌باشد را به شکل زیر نتیجه خواهد داد:

$$\upsilon=\begin{bmatrix} a_1 & a_1 & a_1 \\ a_2 & a_2 & a_2 \\ a_3 & a_3 & a_3 \\ a_4 & a_4 & a_4 \end{bmatrix}$$

دستور زیر

$$\upsilon = repmat(s',2,3)$$

ماتریس v به شکل زیر را نتیجه می‌دهد:

$$\upsilon=\begin{bmatrix} a_1 & a_1 & a_1 \\ a_2 & a_2 & a_2 \\ a_3 & a_3 & a_3 \\ a_4 & a_4 & a_4 \\ a_1 & a_1 & a_1 \\ a_2 & a_2 & a_2 \\ a_3 & a_3 & a_3 \\ a_4 & a_4 & a_4 \end{bmatrix}$$

دو بردار سطری s و t را درنظر بگیرید. در این صورت دستور:

$$[u,v]=meshgrid(s,t)$$

نتایج یکسانی را با استفاده از دو دستور زیر ارائه خواهد نمود:

$$u=repmat(s,length(t),1)$$

$$v=repmat(t',1,length(s))$$

در هر مورد، u و v ماتریسهایی از مرتبهٔ $(length(t) \times length(s))$ می‌باشند. بنابراین اگر؛

$$s=[s_1 \quad s_2 \quad s_3 \quad s_4]$$

$$t=[t_1 \quad t_2 \quad t_3]$$

در این صورت دستور زیر:

$$[u,v]=meshgrid(s,t)$$

معادلهای عددی دو ماتریس (4×3) زیر را، ایجاد خواهد کرد.

$$u=\begin{bmatrix} s_1 & s_2 & s_3 & s_4 \\ s_1 & s_2 & s_3 & s_4 \\ s_1 & s_2 & s_3 & s_4 \end{bmatrix} \quad (۲-۱-\text{الف})$$

$$v=\begin{bmatrix} t_1 & t_1 & t_1 & t_1 \\ t_2 & t_2 & t_2 & t_2 \\ t_3 & t_3 & t_3 & t_3 \end{bmatrix} \quad (۲-۱-\text{ب})$$

می‌توان از تابع *meshgrid* جهت باز گرداندن یک ماتریس به شکل زیر استفاده نمود:

$$w=meshgrid(s,t)$$

که ماتریس $w=u$ را ایجاد می‌کند، که u توسط معادلهٔ (۲ - ۱ - الف) قبلاً نشان داده شده است.

این روش در مثال۲- ۲ نشان داده شده است.

دو تابعی که برای ایجاد تغییرات در ماتریس ها بکار می‌رود عبارتند از: $fliplr(a)$ ، $flipud(a)$ که به ترتیب سطرها و ستونهای ماتریس a را بر عکس می‌کنند. ماتریس 2×5 زیر را در نظر بگیرید:

$$a=\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \end{bmatrix}$$

این ماتریس بوسیلهٔ دستور زیر ایجاد شده است؛

$$a=[a_{11}\,a_{12}\,a_{13}\,a_{14}\,a_{15};a_{21}\,a_{22}\,a_{23}\,a_{24}\,a_{25}]$$

در این صورت خواهیم داشت:

$$fliplr(a)=\begin{bmatrix} a_{15} & a_{14} & a_{13} & a_{12} & a_{11} \\ a_{25} & a_{24} & a_{23} & a_{22} & a_{21} \end{bmatrix}$$

$$flipud(a)=\begin{bmatrix} a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \end{bmatrix}$$

و

$$flipud(fliplr(a))=\begin{bmatrix} a_{25} & a_{24} & a_{23} & a_{22} & a_{21} \\ a_{15} & a_{14} & a_{13} & a_{12} & a_{11} \end{bmatrix}$$

نتایج خروجی $fliplr(a)$ و $flipud(a)$ می‌توانند با استفاده از علامت دو نقطه (:) در آرگومان ماتریس نیز به دست آیند: برای مثال، دستور زیر؛

```
c = fliplr ( a )
```

نتایج مشابه دستور زیر را ایجاد خواهد کرد:

```
c = a(:, length ( a ) : -1 : 1 )
```

حال بردار زیر را در نظر بگیرید:

$$c = \begin{bmatrix} a & fliplr(a) \end{bmatrix}' = \begin{bmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \\ a_{13} & a_{21} \\ a_{14} & a_{24} \\ a_{15} & a_{25} \\ a_{15} & a_{25} \\ a_{14} & a_{24} \\ a_{13} & a_{23} \\ a_{12} & a_{22} \\ a_{11} & a_{21} \end{bmatrix}$$

که در آن دو سطر پنجم و ششم یکسان می‌باشند. برای حذف یکی از این سطرها، می‌توان آنرا برابر مقدار تهی قرار داد. این کار با استفاده از علامت [ ] که در آن فاصلهٔ خالی بین دو قلاب وجود ندارد، انجام می شود. در این صورت هر یک از عبارتهای؛

$c(length(a),:) = \quad [\,]$

یا

$c(length(a)+1,:) = \quad [\,]$

ماتریس زیر را ایجاد خواهد کرد:

$$c = \begin{bmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \\ a_{13} & a_{23} \\ a_{14} & a_{24} \\ a_{15} & a_{25} \\ a_{14} & a_{24} \\ a_{13} & a_{23} \\ a_{12} & a_{22} \\ a_{11} & a_{21} \end{bmatrix}$$

که درآن مرتبهٔ ماتریس c، (2×9) می باشد. دستور [] =$c(length(a),:)$ یعنی همهٔ ستونها و سطر شمارهٔ $length(a)$ در ماتریسc باید به مقدار [] نسبت دهی شود (در این مورد حذف شوند). اگر چه می دانیم که طول a ، 5 می باشد، با این حال محول کردن عمل شمارش به مطلب، تمرین خوبی جهت استفاده از تابع $length(a)$ خواهد بود. حال دو ماتریس a (2×5) و b را مطابق زیر ایجاد می کنیم:

$$a=\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \end{bmatrix}\ b=\begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} & b_{15} \\ b_{21} & b_{22} & b_{23} & b_{24} & b_{25} \end{bmatrix}$$

حال به استفادهٔ آنها در سه عملیات مطلب در زیر توجه کنید:

جمع: c = a+b

$$a=\begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} & a_{13}+b_{13} & a_{14}+b_{14} & a_{15}+b_{15} \\ a_{21}+b_{21} & a_{22}+b_{22} & a_{23}+b_{23} & a_{24}+b_{24} & a_{25}+b_{25} \end{bmatrix}$$

بنابراین، c یک ماتریس (5×2) می باشد.

افزایش ستون: c = [a b]

$$c=\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & b_{11} & b_{12} & b_{13} & b_{14} & b_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & b_{21} & b_{22} & b_{23} & b_{24} & b_{25} \end{bmatrix}$$

بنابراین، c یک ماتریس (10×2) می باشد،

افزایش سطر: c = [a;b]

$$c=\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ b_{11} & b_{12} & b_{13} & b_{14} & b_{15} \\ b_{21} & b_{22} & b_{23} & b_{24} & b_{25} \end{bmatrix}$$

بنابراین، c یک ماتریس (5×4) می باشد.

علاوه بر این، اگر

$$x = [x_1\ x_2\ x_3]$$

$$y = [y_1\ y_2\ y_3]$$

در این صورت دستور زیر؛

$$z = [x'\ y']$$

و یا

$$z = [x; y]'$$

ماتریس زیر را ایجاد خواهد کرد:

$$z = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \end{bmatrix}$$

در حالیکه دستور؛

$$z = [x'; y']$$

ماتریس زیر را نتیجه می دهد؛

$$z = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

این ارتباطات هنگام قرار دادن داده‌ها در یک ترتیب دلخواه بسیار مفید خواهند بود.

۲ – ۵ عملیات نقطه‌ای

اکنون علامت نقطه (.) در مطلب را معرفی خواهیم کرد که قابل استفاده برای ماتریسهای هم مرتبه و عملیات ریاضیاتی روی عناصر ماتریسها می‌باشد. دو ماتریس (4×3) :

$$x=\begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \end{bmatrix}$$

و

$$m=\begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix}$$

را در نظر بگیرید. اکنون عملیات نقطه ای مطلب را بطور کامل شرح خواهیم داد؛

$$z_m = x.*m=\begin{bmatrix} x_{11}*m_{11} & x_{12}*m_{12} & x_{13}*m_{13} & x_{14}*m_{14} \\ x_{21}*m_{21} & x_{22}*m_{22} & x_{23}*m_{23} & x_{24}*m_{24} \\ x_{31}*m_{31} & x_{32}*m_{32} & x_{33}*m_{33} & x_{34}*m_{34} \end{bmatrix} \quad (٢ - ٢ - \text{الف})$$

$$z_d = x./m=\begin{bmatrix} x_{11}/m_{11} & x_{12}/m_{12} & x_{13}/m_{13} & x_{14}/m_{14} \\ x_{21}/m_{21} & x_{22}/m_{22} & x_{23}/m_{23} & x_{24}/m_{24} \\ x_{31}/m_{31} & x_{32}/m_{32} & x_{33}/m_{33} & x_{34}/m_{34} \end{bmatrix} \quad (٢ - ٢ - \text{ب})$$

$$z_e = x.\hat{}\,m=\begin{bmatrix} x_{11}\hat{}\,m_{11} & x_{12}\hat{}\,m_{12} & x_{13}\hat{}\,m_{13} & x_{14}\hat{}\,m_{14} \\ x_{21}\hat{}\,m_{21} & x_{22}\hat{}\,m_{22} & x_{23}\hat{}\,m_{23} & x_{24}\hat{}\,m_{24} \\ x_{31}\hat{}\,m_{31} & x_{32}\hat{}\,m_{32} & x_{33}\hat{}\,m_{33} & x_{34}\hat{}\,m_{34} \end{bmatrix} \quad (٢ - ٢ - \text{ج})$$

به بیان دیگر :

$$z_{mij} = x_{ij} m_{ij} \qquad \{or\ zm(i,j) = x(i,j) * m(i,j)\}$$

$$z_{mij} = x_{ij} / m_{ij} \qquad \{or\ zd(i,j) = x(i,j) / m(i,j)\}$$

$$z_{mij} = x_{ij}^{\,m_{ij}} \qquad \{or\ ze(i,j) = x(i,j)\hat{}\,m(i,j)\}$$

که در آن $i=1,2,3$ و $j=1,2,3,4$ می‌باشد. توجه کنید که علامت نقطه (.) باید قبل از علامت ضرب، تقسیم و توان قرار گیرد. علامت نقطه برای عملیات جمع یا تفریق لازم نمی باشد زیرا نشانه‌های ماتریسی، همان عملیات جمع و یا تفریق آرایه به آرایه را انجام می‌دهند. معادلهٔ (٢ - ٦) را ملاحظه کنید. اکنون چند مورد خاص دربارهٔ این عملیات را امتحان خواهیم کرد. در مورد ضرب نقطه ای

خواهیم دید که اگر $x = x_0$ باشد، که x_0 یک ثابت عددی می باشد، در این صورت عملیات نقطه‌ای لازم نخواهد بود زیرا که:

$$x_0.*m=\begin{bmatrix} x_0*m_{11} & x_0*m_{12} & x_0*m_{13} & x_0*m_{14} \\ x_0*m_{21} & x_0*m_{22} & x_0*m_{23} & x_0*m_{24} \\ x_0*m_{31} & x_0*m_{32} & x_0*m_{33} & x_0*m_{34} \end{bmatrix}=x_0\begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix}=x_0*m$$

به طور مشابه هنگامیکه $m = m_0$ ، ثابت عددی می‌باشد، خواهیم داشت:

$$x.*m = x*m_0$$

در مورد تقسیم نقطه‌ای هنگامیکه $m = m_0$, ثابت عددی می‌باشد، داریم:

$$x./m_0 = \begin{bmatrix} x_{11}/m_0 & x_{12}/m_0 & x_{13}/m_0 & x_{14}/m_0 \\ x_{21}/m_0 & x_{22}/m_0 & x_{23}/m_0 & x_{24}/m_0 \\ x_{31}/m_0 & x_{32}/m_0 & x_{33}/m_0 & x_{34}/m_0 \end{bmatrix}$$

$$=\frac{1}{m_0}\begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \end{bmatrix}=x/m_0$$

و بنابراین استفاده از عملیات نقطه‌ای ضرورتی ندارد. اما هنگامیکه $x = x_0$، ثابت عددی می‌باشد خواهیم داشت:

$$x_0./m = \begin{bmatrix} x_0/m_{11} & x_0/m_{12} & x_0/m_{13} & x_0/m_{14} \\ x_0/m_{21} & x_0/m_{22} & x_0/m_{23} & x_0/m_{24} \\ x_0/m_{31} & x_0/m_{32} & x_0/m_{33} & x_0/m_{34} \end{bmatrix}$$

$$=x_0\begin{bmatrix} 1/m_{11} & 1/m_{12} & 1/m_{13} & 1/m_{14} \\ 1/m_{21} & 1/m_{22} & 1/m_{23} & 1/m_{24} \\ 1/m_{31} & 1/m_{32} & 1/m_{33} & 1/m_{34} \end{bmatrix}=x_0./m$$

که در این مورد، عملیات نقطه‌ای لازم خواهد بود.

بررسی مورد آخر نشان می‌دهد که اگر $m = m_0$ یا $x = x_0$ ثابت عددی باشد، همواره از عملیات نقطه‌ای در مورد توان باید استفاده کنیم. بنابراین اگر $z = x.\hat{}\,m$، در این صورت؛

$z = x_0.\hat{}\, m$

و

$z = x.\hat{}\, m_0$

خواهد بود.

برای روشنتر شدن عملیات نقطه‌ای در مورد توان به نحوهٔ محاسبهٔ 2^j برای $j=1,2,...,8$ توجه‌کنید. برنامهٔ ذیل اینکار را انجام خواهد داد:

```
x = 1 : 8;
y = 2.^x
```

که خروجی آن مطابق زیر می‌باشد:

```
2   4   8   16   32   64   128   256
```

بنابراین قراردادن نقطه اعشار، قبل از عملگر توان (^) به مطلب می‌فهماند که باید عدد 2 را به توان تمامی عناصر بردار x برساند و نتایج را در عناصر متناظر بردار y که دارای همان طول می‌باشد، قرار دهد. برنامهٔ قبل را می توان به شکل فشردهٔ زیر نوشت:

```
y = 2.^(1 : 8)     % or  y = 2.^[1 : 8]
```

اگر مسئله معکوس شود و بخواهیم بردار x^2 را تعیین کنیم، برنامه به صورت زیر خواهد بود :

```
y = (1 : 8).^2     % or  y = [1 : 8].^2
```

که خروجی زیر حاصل می شود:

```
1   4   9   16   25   36   49   64
```

اگر تابع $f(y)$ بطور مثال هر تابعی مثل $bessely$، $cosh$، sin باشد و y را ماتریسی (4×3) در نظر بگیریم، در این صورت داریم:

$$z = f(y) = \begin{bmatrix} f(y_{11}) & f(y_{12}) & f(y_{13}) & f(y_{14}) \\ f(y_{21}) & f(y_{22}) & f(y_{23}) & f(y_{24}) \\ f(y_{31}) & f(y_{32}) & f(y_{33}) & f(y_{34}) \end{bmatrix}$$

توجه کنید که اگر مرتبهٔ تمامی کمیتها یکسان باشد، می‌توان عملیات نقطه‌ای را به صورت ترکیبی بکار برد. برای مثال، اگر a، b، c، d، و f ماتریسهایی (2×3) باشند، در این صورت عبارت:

$$z = \left[\tan a - f\left(\frac{b}{c}\right)^{d} \right]^{2}$$

در مطلب بصورت زیر نوشته می‌شود:

$$z = (\tan(a) - f.*(b./c).\hat{}\,d).\hat{}\,2;$$

عناصر ماتریس z که در زیر مشاهده می‌کنید مقادیر عددی محاسبه شده توسط عبارت فوق می‌باشند (البته با فرض اینکه a و b و c و d و f ماتریس‌های 3×4 باشند) که به شکل زیر نمایش داده می‌شوند.

$$z = \begin{bmatrix} (\tan(a_{11}) - f_{11}*(b_{11}/c_{11})\hat{}\,d_{11})\hat{}\,2 & (\tan(a_{12}) - f_{12}*(b_{12}/c_{12})\hat{}\,d_{12})\hat{}\,2 & \cdots \\ (\tan(a_{21}) - f_{21}*(b_{21}/c_{21})\hat{}\,d_{21})\hat{}\,2 & (\tan(a_{22}) - f_{22}*(b_{22}/c_{22})\hat{}\,d_{22})\hat{}\,2 & \cdots \\ (\tan(a_{31}) - f_{31}*(b_{31}/c_{31})\hat{}\,d_{31})\hat{}\,2 & (\tan(a_{32}) - f_{32}*(b_{32}/c_{32})\hat{}\,d_{32})\hat{}\,2 & \cdots \end{bmatrix}$$

برای مشاهدهٔ کاربرد دیگری از عملیات نقطه‌ای به مثال دستور meshgrid می‌پردازیم که با گرفتن دو بردار $s = [s_1 \quad s_2 \quad s_3 \quad s_4]$ و $t = [t_1 \quad t_2 \quad t_3]$ در دستور زیر استفاده شدند؛

$$[u, v] = meshgrid\ (s, t)$$

نتیجهٔ آن ایجاد دو ماتریس (4×3) زیر بود:

$$u = \begin{bmatrix} s_1 & s_2 & s_3 & s_4 \\ s_1 & s_2 & s_3 & s_4 \\ s_1 & s_2 & s_3 & s_4 \end{bmatrix} \quad and \quad v = \begin{bmatrix} t_1 & t_1 & t_1 & t_1 \\ t_2 & t_2 & t_2 & t_2 \\ t_3 & t_3 & t_3 & t_3 \end{bmatrix} \quad (٣ - ٢)$$

فرض کنید می‌خواهیم عناصر متناظر u و v را در هم ضرب کنیم. در این صورت استفاده از ضرب نقطه‌ای به شکل زیر:

$$z = u.*v$$

نتیجهٔ زیر را بدست می‌دهد (معادلهٔ (٢ - ٢ الف) را ملاحظه کنید):

$$z = \begin{bmatrix} s_1*t_1 & s_2*t_1 & s_3*t_1 & s_4*t_1 \\ s_1*t_2 & s_2*t_2 & s_3*t_2 & s_4*t_2 \\ s_1*t_3 & s_2*t_3 & s_3*t_3 & s_4*t_3 \end{bmatrix} \quad (٤ - ٢)$$

عناصر ماتریس z را می‌توان بعنوان حاصل ضرب عناصر متناظر s و t تعریف کرد. در مورد جمع، تفریق، تقسیم و توان نیز نتایج مشابهی بدست خواهد آمد؛ زیرا علامت ضرب $(\times)$ را می توان با یکی از عملگرهای فوق جایگزین کرد.

اکنون توابع؛

sum

و

cumsum

که معمولاً همراه عملیات نقطه‌ای مورد استفاده قرار می گیرند را بررسی می کنیم. در دستور *sum* ، هنگامیکه آرگومان آن یک بردار باشد، تابع *sum* مجموع مقادیر عناصر بردار را که یک عدد می‌باشد، ارائه می‌دهد. در حالیکه اگر آرگومانش یک ماتریس باشد، مجموع عناصر ستونهای ماتریس را به صورت جداگانه و به شکل برداری سطری، که تعداد عناصر آن برابر تعداد ستونهای ماتریس اصلی می باشد، بدست خواهد داد؛ لذا اگر z یک ماتریس (4×3) با عناصر z_{ij} باشد، در این صورت:

$$sum(z)=\left[\sum_{n=1}^{3}z_{n1}\quad\sum_{n=1}^{3}z_{n2}\quad\sum_{n=1}^{3}z_{n3}\quad\sum_{n=1}^{3}z_{n4}\right] \quad (۲-۵-\text{الف})$$

خواهد بود که برداری دارای چهار عنصر می باشد، در حالیکه:

$$sum(z')=\left[\sum_{n=1}^{4}z_{1n}\quad\sum_{n=1}^{4}z_{2n}\quad\sum_{n=1}^{4}z_{3n}\right] \quad (۲-۵-\text{ب})$$

یک بردار سه عنصره می‌باشد. ذکر این نکته ضروری است که z می تواند نتیجهٔ چند عملیات نقطه‌ای و ترکیبی از آنها باشد.

با اعمال تابع *cumsum* روی بردار v متشکل از n عنصر v_j، برداری شامل n عنصر مطابق زیر خواهیم داشت:

$$y=cumsum(v)=\left[\sum_{k=1}^{1}v_k\quad\sum_{k=1}^{2}v_k\quad\ldots\quad\sum_{k=1}^{n}v_k\right]$$

از طرف دیگر اگر w ماتریسی $(m\times n)$ متشکل از عناصر w_{jk} باشد، در این صورت دستور *cumsum(w)* ماتریس زیر را نتیجه خواهد داد:

$$a=\begin{bmatrix}\sum_{k=1}^{1}w_{k1} & \sum_{k=1}^{1}w_{k2} & \cdots & \sum_{k=1}^{1}w_{kn}\\ \vdots\sum_{k=1}^{2}w_{k1} & \sum_{k=1}^{2}w_{k2} & & \\ & & \ddots & \\ \sum_{k=1}^{m}w_{k1} & & & \sum_{k=1}^{m}w_{kn}\end{bmatrix}$$

برای روشنتر شدن چگونگی استفاده از دستور *sum* معادلهٔ زیر را در نظر بگیرید:

$$z = \sum_{m=1}^{4} m^m$$

برنامه‌ای که جهت محاسبهٔ عبارت فوق بکار می‌رود به شکل زیر است:

```
m = 1 : 4;
z = sum( m.^m )
```

که پس از اجرا، نتیجهٔ $z = 288$ را ارائه می‌دهد. این برنامه می‌تواند به شکل فشرده‌تر به صورت زیر نوشته شود:

```
z = sum((1 : 4).^(1 : 4))
```

اکنون نتایج این قسمت را با محاسبهٔ عبارت زیر[1] برای $N = 305$ و پنج مقدار متوالی x با اختلاف یکسان در بازهٔ $0 \le x \le 2$ نشان خواهیم داد.

$$sec\, hx = 4\pi \sum_{n=1,3,5}^{N\to\infty} \frac{n(-1)^{(n-1)/2}}{(n\pi)^2 + 4x^2}$$

همچنین مقادیر بدست آمده از سری را با مقادیر حقیقی‌شان مقایسه خواهیم کرد. بنابراین برنامهٔ زیر را داریم:

```
nn = 1 : 2 : 305;                                          % (1×153)
xx = linspace(0,2,5);                                      % (1×5)
[x.n] = meshgrid(xx,nn);                                   % (153×5)
s = 4 * pi * sum(n.*(-1).^((n-1)/2))./((pi*n).^2 + 4*x.^2); % (1×5)
se = sech(xx);                                             % (1×5)
compare = [s'  se']                                        % (5×2)
```

که پس از اجرا نتایج زیر در پنجرهٔ فرمان مطلب نشان داده خواهد شد:

---

[1] *L. B. W. Jolley, Summation of Series, 2nd ed., Dover Publication, New York, 1961.*

1.0021 1.0000
0.8889 0.8868
0.6501 0.6481
0.4272 0.4251
0.2679 0.2658

مرتبهٔ آرگومانهای دستور *meshgrid* را بگونه‌ای انتخاب کرده‌ایم تا ماتریسهایی از مرتبهٔ (5×153) ایجاد کنیم. زیرا دستور *sum* عمل جمع را روی تمامی عناصر سطرها به صورت ستون به ستون انجام می دهد. برنامهٔ فوق را می توان به صورت فشرده تر به شکل زیر نوشت:

```
[x,n]=meshgrid(linspace(0,2,5),1:2:305);
s=4*pi*sum(n.*(-1).^((n-1)/2)./((pi*n).^2+4*x.^2));
compare=[s' sech(linspace(0,2,5)')]
```

۲- ۶ اعمال ریاضی روی ماتریسها

اکنون چند عملیات اساسی در مورد ماتریسها که شامل: جمع، تفریق، ضرب، معکوس ماتریس، دترمینان، حل معادلات ویافتن ریشه‌ها (مقادیر ویژه) می‌شود را معرفی خواهیم کرد. سپس از این نتایج برای بدست آوردن حل‌های عددی دسته های مختلف مسائل مهندسی استفاده خواهیم نمود.

۲- ۶ - ۱ جمع و تفریق

اگر دوماتریس a و b را که دارای مرتبهٔ یکسان $(m\times n)$ می‌باشند، را در نظر بگیریم در این صورت داریم:

$$a\pm b=\begin{bmatrix} a_{11}\pm b_{11} & a_{12}\pm b_{12} & \cdots & a_{1n}\pm b_{1n} \\ a_{21}\pm b_{21} & a_{22}\pm b_{22} & & \\ \vdots & & \ddots & \\ a_{m1}\pm b_{m1} & \cdots & & a_{mn}\pm b_{mn} \end{bmatrix} \quad (٦-٢)$$

۲- ۶ - ۲ ضرب

اگر دو ماتریس a از مرتبهٔ $(m\times k)$ و ماتریس b از مرتبهٔ $(k\times n)$ را در نظر بگیریم، در این صورت خواهیم داشت:

$$c=ab=\begin{bmatrix}\sum_{j=1}^{k}a_{1j}b_{j1} & \sum_{j=1}^{k}a_{1j}b_{j2} & \cdots & \sum_{j=1}^{k}a_{1j}b_{jn}\\ \sum_{j=1}^{k}a_{2j}b_{j1} & \sum_{j=1}^{k}a_{2j}b_{j2} & & \\ \vdots & & \ddots & \vdots \\ \sum_{j=1}^{k}a_{mj}b_{j1} & \cdots & & \sum_{j=1}^{k}a_{mj}b_{jn}\end{bmatrix} \quad (۷-۲)$$

که در آن c از مرتبهٔ $(m\times n)$ می باشد. در نظر داشته باشید که ضرب دو ماتریس تنها هنگاهیکه ارقام نزدیکتر به هم مرتبه‌هایشان یکسان باشد (در این مورد k) تعریف می‌شود.

به بیان دیگر، $(m\times k)(k\times n)=(m\times n)$؛ که این عبارت نشان می دهد ما k جمله را همانطور که در معادلهٔ (۲- ۷) نشان داده شد با هم جمع زده‌ایم. دستور مطلب برای ضرب ماتریسی به شکل زیر می‌باشد:

$$c=a*b$$

تعمیم معادلهٔ (۲- ۷) در مورد ضرب ماتریسی $f=cd=abd$ در مثال۲- ۵ آورده شده است.

توجه کنید که اگر $m=n$ باشد، در حالت کلی $ab\neq ba$، و همچنین می‌توان نشان داد که اگر $c=ab$ باشد، در اینصورت؛

$$c'=(ab)'=b'a'$$

است. همینطور، اگر a ماتریسی یکه $(a=I)$ و $m=n$ باشد، در اینصورت؛

$$Ib=bI=b$$

در ادامه به نتیجه‌گیری در مورد ضرب ماتریسها خواهیم پرداخت و تفسیری در مورد این نتایج ارائه می‌دهیم. سری زیر را در نظر بگیرید[۱]:

۱- سری‌هایی بدین شکل نتیجهٔ حل دسته‌ای از معادلات دیفرانسیل با شرائط مرزی خاص می باشند.

$$w(x,y)=\sum_{j=1}^{k} d_j e_j(x) g_j(y)=\sum_{j=1}^{k} f_j(x) g_j(y)$$

فرض کنید می‌خواهیم مقدار $w(x,y)$، را به ازای بازه ای از مقادیر x وy به صورت $x=x_1,x_2,...,x_m$ و $y=y_1,y_2,...,y_n$ بیابیم، در این صورت می توان عبارت؛

$$w(x_i,y_j)=\sum_{l=1}^{k} f_l(x_i) g_l(y_j) \quad i=1,2,...,m \quad j=1,2,...,n$$

را به عنوان یک عنصر ماتریس w از مرتبهٔ $(m\times n)$ در نظر گرفت. فرض کنید f ماتریسی از مرتبهٔ $(m\times k)$؛

$$f=\begin{bmatrix} f_1(x_1) & f_2(x_1) & & f_k(x_1) \\ f_1(x_2) & f_2(x_2) & \cdots & \vdots \\ \vdots & & \ddots & \\ f_1(x_m) & \cdots & & f_k(x_m) \end{bmatrix}$$

و g ماتریسی از مرتبهٔ $(k\times n)$؛

$$f=\begin{bmatrix} g_1(y_1) & g_1(y_2) & \cdots & g_1(y_n) \\ g_2(y_1) & g_2(y_2) & & \\ \vdots & & \ddots & \vdots \\ g_k(y_1) & \cdots & & g_k(y_n) \end{bmatrix}$$

باشد. در اینصورت بنا به معادلهٔ (۲ – ۷) داریم:

$$w=fg=\begin{bmatrix} \sum_{j=1}^{k} f_j(x_1)g_j(y_1) & \sum_{j=1}^{k} f_j(x_1)g_j(y_2) & \cdots & \sum_{j=1}^{k} f_j(x_1)g_j(y_n) \\ \sum_{j=1}^{k} f_j(x_2)g_j(y_1) & \sum_{j=1}^{k} f_j(x_2)g_j(y_2) & & \\ \vdots & & \ddots & \vdots \\ \sum_{j=1}^{k} f_j(x_m)g_j(y_1) & \cdots & & \sum_{j=1}^{k} f_j(x_m)g_j(y_n) \end{bmatrix} \qquad (۸-۲)$$

به بیان دیگر عملیات ضرب ماتریسی، مجموع سریها را با هر ترکیبی از مقادیر x و y محاسبه می‌کند. این موضوع ابزار مناسبی را جهت جمع مقادیر سریها، در هر نقطه ای از شبکه که بوسیلهٔ هر ترکیبی از عناصر بردار x و y تعریف شده باشد، فراهم خواهد کرد.

اکنون سه مورد خاص را در مورد ضرب عمومی ماتریسها در نظر می گیریم؛

- ضرب بردار سطری در ستونی
- ضرب بردار ستونی در سطری
- ضرب بردار سطری در ماتریس

در سه مورد فوق می توان از ویژگی خلاصه‌نویسی مطلب در روشهای حل ماتریسی مسائل مهندسی استفاده کرد.

ضرب بردار سطری در ستونی: فرض کنید که a یک بردار سطری باشد؛

$$a = [a_1\, a_2 \ldots a_k]$$

که مرتبهٔ آن $(1 \times k)$ می باشد و b بردار ستونی باشد؛

$$b = [b_1\, b_2 \ldots b_k]'$$

که دارای مرتبهٔ $(k \times 1)$ است در این صورت حاصل ضرب $d = ab$ اسکالر خواهد بود؛

$$d = ab = [a_1\ a_2\ \ldots\ a_k]\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_k \end{bmatrix} = \left[\sum_{j=1}^{k} a_j b_j\right] = \sum_{j=1}^{k} a_j b_j \qquad (۹-۲)$$

زیرا ضرب مرتبه‌های دو بردار به شکل $(1 \times 1) = (1 \times k)(k \times 1)$ می باشد. لذا این عملیات جبری، ضرب نقطه‌ای دو بردار نامیده می‌شود. دستور مطلب برای ضرب ماتریسی دو بردار همانگونه که در بالا بیان شد، به صورت:

$d = a * b$

و یا

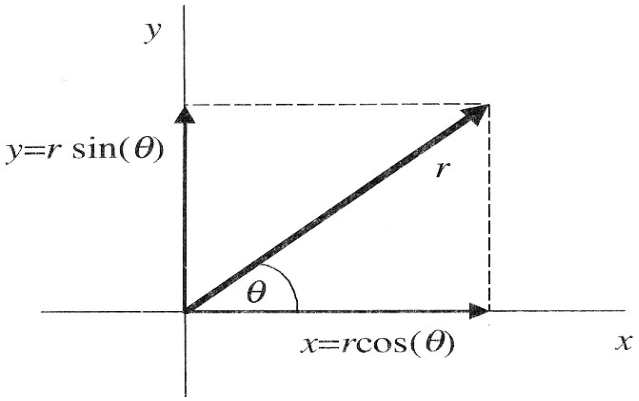
$d = dot(a,b)$

می باشد. اگر c ماتریسی از مرتبهٔ $(n\times n)$ باشد و x بردار ستونی از مرتبهٔ $(n\times 1)$ باشد، در این صورت؛

$$f = x'cx = \sum_{i=1}^{n}\sum_{j=1}^{n} x_i c_{ij} x_j \qquad (۱۰-۲)$$

که باز هم یک اسکالر خواهد بود زیرا ضرب مرتبه هایشان به صورت $(1\times 1)=(1\times n)(n\times n)(n\times 1)$ می‌باشد. دستور مطلب برای معادلهٔ (۲-۱۰) به $f = x' * c * x$

ترکیب عملیات بدین شکل در حل مسائل مقدار ویژه بسیار مفید می‌باشد. (فصل ۹ را ملاحظه کنید)



شکل ۲-۲) تبدیل از دستگاه مختصات قطبی به کارتزین

ضرب بردار ستونی در سطری: فرض کنید b بردار ستونی از مرتبهٔ $(m\times 1)$ و a برداری سطری از مرتبهٔ $(1\times n)$ باشد. در آن صورت حاصلضرب $h = ba$ به صورت زیر خواهد بود:

$$h = ba = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \begin{bmatrix} a_1 & a_2 & \dots & a_n \end{bmatrix} = \begin{bmatrix} b_1a_1 & b_1a_2 & \dots & b_1a_n \\ b_2a_1 & b_2a_2 & \dots & \vdots \\ \vdots & & \ddots & \\ b_ma_1 & \dots & & b_ma_n \end{bmatrix} \qquad (۱۱-۲)$$

که ماتریسی از مرتبهٔ $(m \times n)$ می‌باشد. زیرا ضرب مرتبه هایشان به صورت $(m \times n) = (m \times 1)(1 \times n)$ خواهد بود. بنابر این عناصر ماتریس h (که به صورت $h_{ij} = b_i a_j$ هستند) حاصلضرب تک به تک تمامی ترکیب‌های عناصر b و a خواهند بود.

به عنوان مثالی از کاربردهای این رابطه می‌توان به تبدیل دستگاه مختصات قطبی به دستگاه مختصات کارتزین همانطور که در شکل ۲ – ۲ نشان داده شده است، اشاره نمود. یعنی؛

$$x = r\cos(\theta)$$

$$y = r\,sin(\theta)$$

اگر دو بردار متشکل از مقادیر شعاع ها $r = [r_1\ r_2 \ldots r_m]$ و زاویه ها $\theta = [\theta_1\ \theta_2 \ldots \theta_n]$ را در نظر بگیریم، در این صورت مختصه های کارتزین متناظر به صورت زیر خواهند بود[1]:

$$x = r' * \cos(\theta) = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \end{bmatrix} \begin{bmatrix} \cos\theta_1 & \cos\theta_2 & \ldots & \cos\theta_n \end{bmatrix}$$

$$= \begin{bmatrix} r_1\cos\theta_1 & r_1\cos\theta_2 & \ldots & r_1\cos\theta_n \\ r_2\cos\theta_1 & r_2\cos\theta_2 & & \\ \vdots & & \ddots & \vdots \\ r_m\cos\theta_1 & \cdots & & r_m\cos\theta_n \end{bmatrix} \qquad (۲ - ۱۲ - \text{الف})$$

و

$$x = r' * \sin(\theta) = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \end{bmatrix} \begin{bmatrix} \sin\theta_1 & \sin\theta_2 & \ldots & \sin\theta_n \end{bmatrix}$$

[1] *This conversion can also be Performed with pol 2cart; however. this function is restricted to the case when m=n.*

$$= \begin{bmatrix} r_1 \sin\theta_1 & r_1 \sin\theta_2 & \dots & r_1 \sin\theta_n \\ r_2 \sin\theta_1 & r_2 \sin\theta_2 & & \\ \vdots & & \ddots & \vdots \\ r_m \sin\theta_1 & \dots & & r_m \sin\theta_n \end{bmatrix} \qquad (۲-۱۲-ب)$$

لذا مختصه‌های دستگاه مختصات قطبی را به جای مختصه‌های متناظرشان در مختصات کارتزین قرار داده‌ایم. این کار در هنگام کشیدن نمودار نتایج، همانطور که در مثال زیر آمده است، بسیار مفید خواهد بود.

مثال ۲ - ۱ شکل مد یک عضو دایره‌ای

شکل مد زیر را برای یک غشاء استوانه‌ای صلب که در راستای مرز خارجی اش $r = 1$ کاملاً مقیود شده است، در نظر بگیرید:

$$z(r,\phi) = J_1(3.8316r)\cos(\phi)$$

که در آن $J_1(x)$ تابع بسل[1] نوع اول ازمرتبهٔ یک و (r,ϕ) مختصاتهای قطبی نقاط واقع بر روی غشاء می‌باشند. تابع بسل توسط عبارت زیر تعریف می‌شود.

$$besselj(n,x)$$

که در آن n مرتبه و x آرگومان آن می‌باشد. مبدأ دستگاه مختصات در مرکز غشاء استوانه‌ای واقع شده است. مقدار 3.8316 یکی از ضرایب فرکانس طبیعی این غشاء می‌باشد. این شکل مد را می‌توان توسط تابع زیر ترسیم نمود:

$$mesh(x,y,z)$$

که در آن (x,y) مختصات نقاط روی سطح $z(x,y)$ می باشند. تابع $mesh$ با جزئیات بیشتر در بخش ۷-۲ معرفی شده است. بنابراین اگر سطح را با فاصلهٔ $\Delta r = 0.05$ و هر $\theta = \frac{\pi}{20}$ رسم نماییم، در این صورت برنامهٔ آن به شکل زیر خواهد بود:

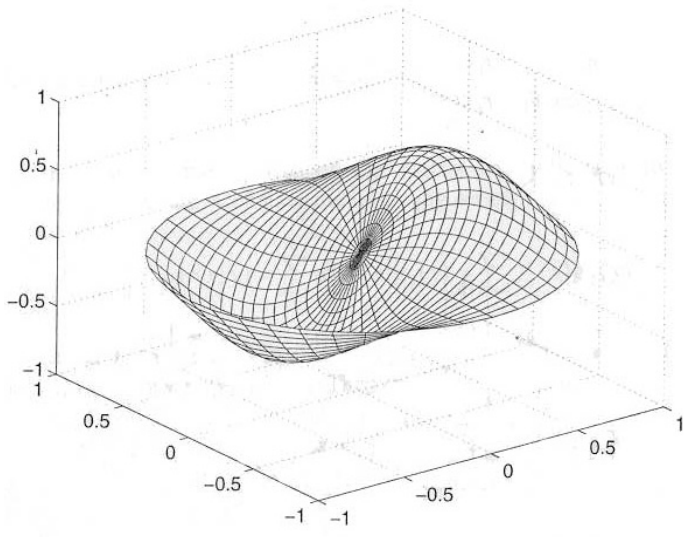
---

[1] *See,for example,F.B.Hildebrand,Advanced Calculus for Applications,Prentice - Hall, Saddle River,NJ,1976*

```
r = [0 : 0.05 : 1]';                          % (21×1)
phi = 0 : pi / 20 : 2 * pi;                   % (1×41)
x = r * cos(phi);                             % (21×41)
y = r * sin(phi);                             % (21×41)
z = besselj(1,3.8316 * r) * cos(phi);         % (21×41)
mesh(x, y, z)
```

در این مورد *phi* ماتریسی (41×1) و *r* ماتریسی (1×21) می‌باشد. بنابراین با توجه به معادلهٔ (۲ - ۱۲) x، y و z هر کدام ماتریسی (41×21) خواهند بود. تبدیل‌های دستگاه مختصات جهت ترسیم سطوح در دستگاه مختصات کارتزین مورد نیاز می‌باشند. توجه کنید که توابع *cos* ، *sin* و *besselj* قادرند بردارها را به عنوان آرگومانشان بپذیرند و بردارهایی را با همان اندازه به عنوان خروجی ارائه دهند؛ به همین دلیل روش های فوق دارای کاربرد فراوان هستند.

اجرای برنامهٔ فوق در شکل ۲ - ۳ آمده است؛



شکل ۲-۳) شکل مد یک غشاء استوانه ای صلب مقید شده

مثال ۲ – ۲ روشی برای حل معادلهٔ لاپلاس

حل معادلهٔ لاپلاس که دارای شرایط مرزی $u(0,\eta)=u(1,\eta)=u(\xi,1)=0$ و $u(\xi,0)=\xi(\xi-1)$ می باشد، عبارت است از:

$$u(\xi,\eta)=4\sum_{n=1}^{N\to\infty}\frac{1-\cos n\pi}{(n\pi)^3}e^{-n\pi\xi}\sin n\pi\eta$$

که در آن $0\le\eta\le 1$ و $\xi\ge 0$ می باشد. حال می‌خواهیم سطح $u(\xi,\eta)$ را با استفاده از دستور $mesh(\eta,\xi,u(\xi,\eta))$ به ازای $N=25$ و با افزایش $\Delta\eta=0.025$ و $\Delta\xi=0.05$ تا $\xi_{max}=0.7$ رسم کنیم. برنامهٔ آن به شرح زیر است:

```
n=(1:25)*pi;                                  % (1×25)
eta=0:0.025:1;                                % (1×41)
xi=0:0.05:0.7;                                % (1×15)
tempc=meshgrid((1-cos(n))./n.^3,xi);          % (15×25)
tempe=exp(-n'*xi)';                           % (15×25)
tempec=tempc.*tempe;                          % (15×25)
temps=sin(n'*eta);                            % (25×41)
z=4*tempec*temps;                             % (15×41)
mesh(eta,xi,z)
```

که نتیجهٔ آن در شکل ۲–٤ نشان داده شده است. ضرب ماتریسی $tempec*temps$ مجموع سریها را به ازای تمام مقادیر n و تمام ترکیبهای eta و xi، همانطور که در معادلهٔ (۲–۸) نشان داده شده است، ایجاد می‌کند. در دستور $mesh$ کاربر می‌تواند دو آرگومان اول این دستور eta و xi را بصورت بردارهایی با طول یکسان z انتخاب نماید. توضیحات مربوط به تابع $mesh$ را در فایل راهنمای[1] مربوط به آن ملاحظه کنید. این فایل متنی را می‌توان به شکل فشرده تر مطابق زیر نوشت:

```
n=(1:25)*pi;
eta=0:0.025:1;
xi=0:0.05:0.7;
z=4*meshgrid((1-cos(n))./n.^3,xi).*exp(-n'*xi)'*sin(n'*eta);
```

---

[1] *Matlab Help File*

```
mesh(eta,xi,z)
```

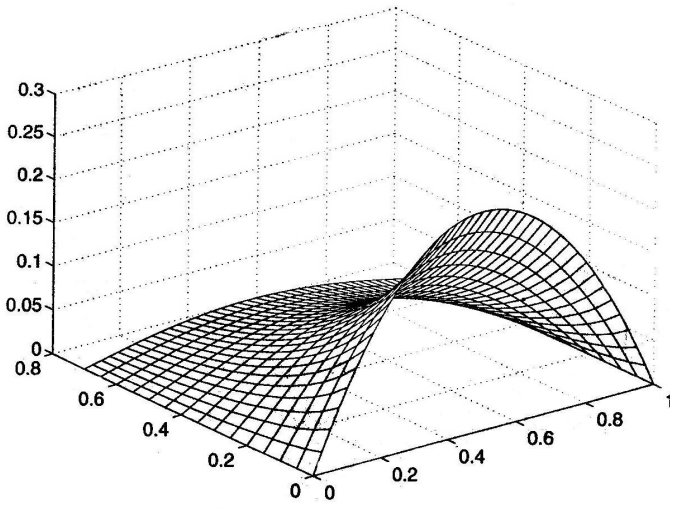
زیرا مطلب عبارات محاسباتی را از سمت چپ به راست انجام می دهد.

مورد سوم - ضرب بردار سطری در ماتریس. فرض کنید b ماتریسی $(m\times n)$ و a برداری سطری از مرتبهٔ $(1\times m)$ باشد. در این صورت حاصل ضرب $g=ab$ به شکل زیر خواهد بود:

$$g=ab=\begin{bmatrix}a_1 & a_2 & \dots & a_m\end{bmatrix}\begin{bmatrix}b_{11} & b_{12} & \dots & b_{1n}\\ b_{21} & b_{22} & \cdots & \\ \vdots & & \ddots & \vdots\\ b_{m1} & \dots & & b_{mn}\end{bmatrix}$$

$$=\begin{bmatrix}\sum_{k=1}^{m}a_k b_{k1} & \sum_{k=1}^{m}a_k b_{k2} & \dots & \sum_{k=1}^{m}a_k b_{kn}\end{bmatrix} \quad (۱۳-۲)$$

که بردار سطری از مرتبهٔ $(1\times n)$ می‌باشد، زیرا که ضرب مرتبه‌های آنها بصورت $(1\times n)=(1\times m)(m\times n)$ می باشد.



شکل ۲-٤) نمایش حل معادلهٔ لاپلاس

این نتایج را می‌توان به صورت زیر تفسیر نمود. سری زیر را در نظر بگیرید[1]:

$$r(x)=\sum_{j=1}^{m} p_j h_j(x) \qquad (۱۴-۲)$$

فرض کنید که علاقمند به محاسبهٔ مقدار $r(x)$ به ازای مقادیر $x_1,x_2,...,x_n$ هستیم. در این صورت می‌توانیم یک عنصر بردار r از مرتبهٔ $(n\times 1)$ را به شکل زیر محاسبه کنیم:

$$r(x_i)=\sum_{j=1}^{m} p_j h_j(x_i) \qquad i=1,2,...,n$$

فرض کنید p برداری از مرتبهٔ $(1\times m)$ شامل عناصر p_i و v ماتریسی $(m\times n)$ به شکل زیر باشد:

$$v=\begin{bmatrix} h_1(x_1) & h_1(x_2) & & h_1(x_n) \\ h_2(x_1) & h_2(x_2) & \cdots & \\ \vdots & & \ddots & \vdots \\ h_m(x_1) & \cdots & & h_m(x_n) \end{bmatrix}$$

در این صورت رابطهٔ $r=pv$، بردار زیر را نتیجه خواهد داد:

$$r=pv=\left[\sum_{j=1}^{m} p_j h_j(x_1) \quad \sum_{j=1}^{m} p_j h_j(x_2) \quad \cdots \quad \sum_{j=1}^{m} p_j h_j(x_n)\right]$$

چند مثال زیر جهت آشنایی بیشتر با چگونگی استفاده از این نتایج آورده شده است.

مثال ۲ - ۳ جمع یک سری فوریه

نمایش یک سری فوریه با پالس مربعی دارای دوام d و پریود τ در زیر آمده است[2]:

---

[1] *Seris of this result from the solution of differention equations with certain boundary and from Fourier expansions of periodic functions.*

[2] *See, for example, H. P. Hsu, Applied Fourier Analysis, Harcourt Brace Jovanovich, San Diego, CA, 1984.*

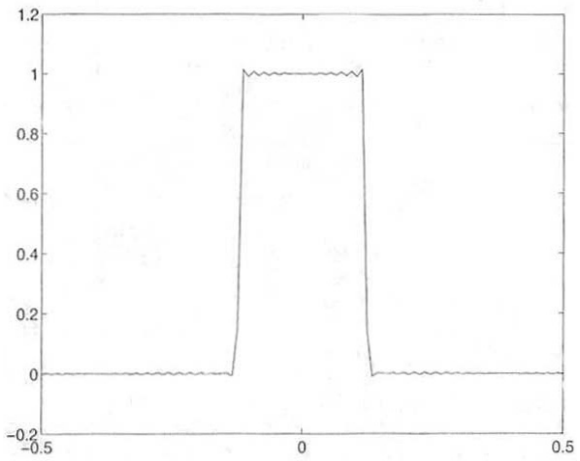
$$f(\tau)=\frac{d}{T}\left[1+2\sum_{n=1}^{\infty}\frac{sin(n\pi d/T)}{n\pi d/T}cos(2\pi n\tau)\right]$$

که در آن $\tau=\frac{t}{T}$. متوجه می‌شویم که این معادله همشکل معادلهٔ (۲ - ١٤) می‌باشد. اکنون می‌خواهیم 150 جملهٔ $f(t)$ را جمع کرده و حاصل را در بازهٔ $-1/2\le\tau\le 1/2$ در حالی که $d/T=0.25$ است، رسم کنیم. برای این کار از دستور $plot(x,y)$ استفاده می‌کنیم.

که در آن $x=\tau$ و $y=f(\tau)$ می‌باشد. (بحث راجع به تابع *plot* را در قسمت ۲ - ٦ ببینید.) فایل متنی مربوط به عملیات فوق به صورت زیر می‌باشد:

```
n=1:150;                                %(1×150)
tau=linspace(-.5,.5,100);               %(1×100)
sn=sin(pi*n/4)./(pi*n/4);               %(1×150)
cntau=cos(2*pi*n'*tau);                 %(150 ×100)
f=0.25*(1+2*sn*cntau);                  %(1×100)
plot(tau,f)
```

می‌بینیم که جهت تعیین f باید از ضرب برداری و تقسیم نقطه ای استفاده نمود. حاصل تقسیم نقطه‌ای جهت ایجاد *sn*، برداری از مرتبهٔ (1×150) می‌باشد،و ضرب برداری جهت ایجاد *cntau* ماتریسی از مرتبهٔ (150×100) ایجاد می‌کند که در معادلهٔ (۲-۱۱) نشان داده شده است. بر اساس معادلهٔ (۲-۱۳)، *sn*cntau* برداری سطری می‌باشد که عناصرش مجموع سریها به ازای هر مقدار t می‌باشند. اجرای این فایل متنی در شکل۲- ٥ نشان داده شده است.



شکل ۲-٥) سری فوریهٔ مجموع یک پالس پریودیک

مثال ۲ - ۴ تابع توزیع انباشتی نرمال

یک تقریب برای تابع احتمال توزیع انباشتی نرمال (گوسی)، که احتمال p را در بازهٔ $0 \le X \le x$ تخمین می زند، به صورت زیر می باشد[1]:

$$p(x) = p(X \le x) \cong 1 - \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \sum_{m=1}^{5} b_m (1 + 0.2316419x)^{-m} \qquad 0 \le x \le \infty$$

که $0.5 \le p(x) \le 1$ می باشد، و:

$$b_1 = 0.319381530$$
$$b_2 = -0.356563782$$
$$b_3 = 1.781477937$$
$$b_4 = -1.821255978$$
$$b_5 = 1.330274429$$

محدودهٔ $-\infty \le x \le 0$ را می توان با محاسبهٔ $1 - p(|x|)$ بدست آورد، که در آن $0 \le 1 - p(|x|) \le 0.5$ است.

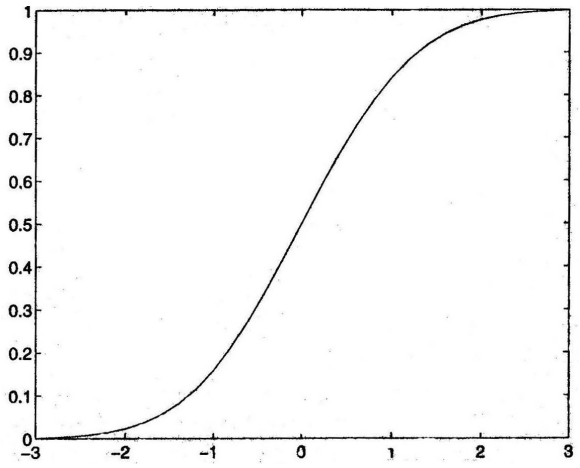
هدف ما محاسبه و رسم توزیع انباشتی برای $-3 \le x \le 3$ در هر $\Delta x = 0.2$ می‌باشد. فایل متنی آن به صورت زیر می‌باشد:

---

[1] *M. Abramowitz and I.A.Stegun, Handbook Of Mathematical Functions, National Bureau Of Standard, Applied Mathematics Series55,U.S.Government Printing Office, Washingtom D.C.,1964,P.932*

```
b = [0.319381530 -0.356563782 1.781477937 -1.821255978 1.330274429];   % (1×5)
m = 1 : length(b);                                                      % (1×5)
x = 0 : 0.2 : 3;                                                        % (1×16)
mm = meshgrid(m,x);                                                     % (16×5)
bb = meshgrid(b,x);                                                     % (16×5)
temp = (1./(1+0.231641* x))';                                           % (16×1)
zxx = meshgrid(temp,m)';                                                % (16×5)
bzm = sum((bb.*(zxx.^mm))');                                            % sum[(16×5)'] → sum[(5×16)]
px = 1 - bzm.* exp(-0.5* x.^2) / sqrt(pi* 2);                           % (1×16)
plot(x, px,'k', -fliplr(1 - px),'k')
```

طول x، 16 و طول m، 5 است. لذا مرتبهٔ سه ماتریس bb، mm و zxx که توسط تابع *meshgrid* ایجاد می شوند، (5×16) است. کمیت $(bb.*(zxx.^{mm}))'$ ، ماتریسی از مرتبهٔ (16×5) می‌باشد، که در معادلهٔ (۲-۵) نشان داده شده است. $Sum((bb.*(zxx.^{mm}))'$ برداری از مرتبهٔ (16×1) می‌باشد. به علاوه $exp(-0.5*x.^2)$، برداری از مرتبهٔ (16×1) می‌باشد. در نتیجه $(bzm.*exp(-0.5*x.^2)$، برداری از مرتبهٔ (16×1) خواهد بود. هنگامیکه هر عنصر این بردار از 1 کم شود، هرکدام از عناصر منتجه $p(x)$ را به ازای 16 مقدار x نشان خواهد داد. تابع *plot*، که به طور مفصلتر در بخش ٦-٢ معرفی خواهد شد از دو مجموعه به عنوان آرگومانهایش استفاده می‌کند. عبارت سوم هر مجموعه سه تایی مربوط به شیوهٔ ترسیم است، که در این قسمت به معنای استفاده از رنگ یکسان (در اینجا سیاه) برای هر قطعه خط می باشد. اولین جفت عبارت هر مجموعهٔ سه تایی مقادیر x و y را که باید نمایش داده شوند، نشان می دهد. عبارت $-fliplr(x)$ مشابه ایجاد بردار جدید $z = -3 : 0.2 : 0$ می‌باشد.



شکل ۲-٦) تابع احتمال توزیع انباشتی نرمال

عبارت $fliplr(1-px)$، ترتیب عناصر بردار $1-px$ را معکوس می کند. و برداری که عناصرش متناظر با مقادیر منفی x ایجاد شده توسط تابع $fliplr(x)-$ می باشند را ایجاد خواهد کرد. اجرای این فایل متنی در شکل ۲-٦ آمده است.

مثال ۲ - ۵ ارزیابی سریها توسط جفت سیگماها

جفت سیگمایی مقابل در نظر بگیرید:

$$w(x_i,y_i,t)=\sum_{n}^{N}\sum_{m}^{M}f_n(x_i)g_m(y_j)h_{nm}(t) \qquad i=1,2,...,s \quad j=1,2,...,p$$

که در آن $x=\begin{bmatrix}x_1 & x_2 & \dots & x_s\end{bmatrix}$ و $y=\begin{bmatrix}y_1 & y_2 & \dots & y_p\end{bmatrix}$ می باشد. اگر فرض کنیم که $N=M=k$ باشد و قرار دهیم $t=t_0$، که مقدار (عددی) نسبت داده شده می باشد، در اینصورت قادر خواهیم بود که این سری را به شیوهٔ زیر محاسبه کنیم. توجه کنید که اگر ماتریس $(m\times n)$، داده شده توسط معادلهٔ (۲- ۷) را در نظر بگیریم و آن را در ماتریس d از مرتبهٔ $(n\times p)$ و شامل عناصر d_{ij} ضرب کنیم، در اینصورت نتیجهٔ آن ماتریس $(m\times p)$ زیر خواهد بود:

$$w=cd=abd=\begin{bmatrix}\sum_{l=1}^{n}\sum_{j=1}^{k}a_{1j}b_{jl}d_{l1} & \sum_{l=1}^{n}\sum_{j=1}^{k}a_{1j}b_{jl}d_{l2} & \dots & \sum_{l=1}^{n}\sum_{j=1}^{k}a_{1j}b_{jl}d_{lp}\\ \sum_{l=1}^{n}\sum_{j=1}^{k}a_{2j}b_{jl}d_{l1} & \sum_{l=1}^{n}\sum_{j=1}^{k}a_{2j}b_{jl}d_{l2} & & \vdots\\ \vdots & & \ddots & \\ \sum_{l=1}^{n}\sum_{j=1}^{k}a_{mj}b_{jl}d_{l1} & \dots & & \sum_{l=1}^{n}\sum_{j=1}^{k}a_{mj}b_{jl}d_{lp}\end{bmatrix} \quad (۱۵-۲)$$

حال $f_n(x)$ را به صورت ماتریس زیر از مرتبهٔ $(k\times s)$

$$f=\begin{bmatrix}f_1(x_1) & f_2(x_1) & & f_s(x_1)\\ f_1(x_2) & f_2(x_2) & \dots & \\ \vdots & & \ddots & \vdots\\ f_1(x_k) & \dots & & f_s(x_k)\end{bmatrix}$$

و $g_m(y)$ را به صورت ماتریس زیر از مرتبهٔ $(k \times p)$

$$g = \begin{bmatrix} g_1(y_1) & g_1(y_2) & \cdots & g_1(y_p) \\ g_2(y_1) & g_2(y_2) & & \\ \vdots & & \ddots & \vdots \\ g_k(y_1) & \cdots & & g_k(y_p) \end{bmatrix}$$

و $h_{nm}(t_0)$ را به صورت ماتریس زیر از مرتبهٔ $(k \times k)$

$$h = \begin{bmatrix} h_{11}(t_0) & h_{12}(t_0) & \cdots & h_{1k}(t_0) \\ h_{21}(t_0) & h_{22}(t_0) & \cdots & \\ \vdots & & \ddots & \vdots \\ h_{k1}(t_0) & \cdots & & h_{kk}(t_0) \end{bmatrix}$$

در نظر می‌گیریم. از آنجاییکه در معادلهٔ (۲- ۵) $n = k$ می باشد، لذا ضرب $f'gh$ به صورت ماتریس زیر از مرتبهٔ $(s \times p)$ خواهد بود.

$$w = \begin{bmatrix} w(x_1, y_1, t_0) & w(x_1, y_2, t_0) & & w(x_1, y_p, t_0) \\ w(x_2, y_1, t_0) & w(x_2, y_2, t_0) & \cdots & \\ \vdots & & \ddots & \vdots \\ w(x_s, y_1, t_0) & \cdots & & w(x_s, y_p, t_0) \end{bmatrix}$$

که در آن

$$w(x_i, y_i, t_0) = \sum_{n=1}^{k} \sum_{m=1}^{k} f_n(x_i) h_{nm}(t_0) g_m(y_i)$$

جهت روشنتر شدن این روند، سری زیر را در نظر بگیرید[1]، که پاسخ جابجایی بدون بعد و نرمالیزه شدهٔ یک پوستهٔ مربعی که دارای تغییر مکان اولیهٔ $w(\eta, \xi, 0) = \eta\xi(\eta - 1)(\xi - 1)$ می‌باشد را نشان می‌دهد:

[1] *H. P. Hsu, ibid., P.202.*

$$w(\eta,\xi,\tau)=\sum_{n=0}^{k}\sum_{m=0}^{k}\frac{sin\big((2n+1)\pi\eta\big)sin\big((2m+1)\pi\xi\big)}{(2n+1)^3+(2m+1)^3}cos\left(\pi\tau\sqrt{(2n+1)^2+(2m+1)^2}\right)$$

که در آن $\eta=\frac{x}{b}$، $\xi=\frac{y}{b}$ ، b طول هر یک از اضلاع مربع بوده، $\tau=\frac{ct_0}{b}$ ، c سرعت انتشار موج در پوسته و t زمان می باشد.

فایل متنی جهت محاسبهٔ این سری در $\tau=0$ و $k=10$ به صورت زیر می باشد.

```
tau=o;n=0:10;m=n;xi=0:0.1;eta=xi;
nden=meshgrid((2*n+1).^3,xi);
mden=meshgrid((2*m+1).^3,eta);
snx=sin(pi*(2*n+1)'*xi);
sme=sin(pi*(2*m+1)'eta);
xterm=snx./eden';
yterm=sme./mden';
[n2,m2]=meshgrid(2*n+1,2*m+1,1);
w=xterm'*cos(pi*tau*sqrt(n2.^2+m2.^2))*yterm;
mesh(xi,eta,w)
```

بردارهای xi، n، m وeta همگی از مرتبهٔ (11×1) می باشند. دو تابع اول *meshgrid* ،دو ماتریس (11×11) برای *ndem* و *mden* ایجاد می کنند. ضربهای برداری که *snx* و *sme* را ایجاد می‌کنند توسط تقسیم‌های نقطه‌ای، عبارتهای *xterm* و *yterm* را تشکیل می‌دهند. سومین تابع *meshgrid* دو ماتریس از مرتبهٔ (11×11) را برای n_2 و m_2 ایجاد می‌کند. بنابراین w یک ماتریس از مرتبهٔ (11×11) می‌باشد. تابع *mesh* شبیه تابع *surf* است یکی از تفاوتهای این دو تابع در شکل جواب، این است که رنگ قطعه‌ها سفید می‌باشد. (بخش ۷-۲ را ملاحظه کنید.) هنگامیکه برنامه فوق اجرا می‌شود، سطحی مطابق آنچه در شکل۲- ۷ نشان داده شده است، ایجاد خواهد شد.

۲ - ۶ - ۳ ماتریس معکوس

معکوس ماتریس مربعی a عملیاتی مطابق زیر است:

$$a^{-1}a=aa^{-1}=I$$

که در آن a منفرد نمی‌باشد، یعنی دترمینانش (بخش ۲-٦-٤ راملاحظه کنید.) مخالف صفر است $\left(|a|\neq0\right)$.اندیس بالایی $"-1"$ مبین ماتریس معکوس است. دستور به دست آوردن معکوس ماتریس a در مطلب به صورت زیر است ؛

```
inv(a)
```

و یا

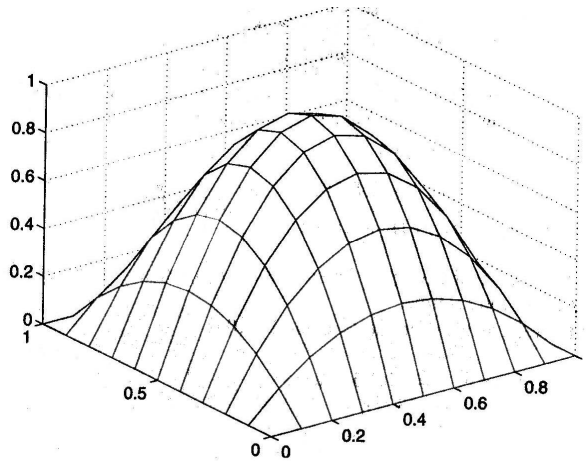
```
a^-1
```

نکتهٔ مهم اینست که در $1/a \neq a\text{^}(-1)$، تایپ عبارت $1/a$ موجب ایجاد خطا خواهد شد. معکوس یک ماتریس را می‌توان با استفاده از عملگر تقسیم وارون همانطور که در بخش ۲ - ٦ - ٥ نشان داده شده است، نیز محاسبه نمود.

۲ - ۶ - ۴ دترمینان

دترمینان ماتریس مرتبهٔ n به صورت زیر تعریف می شود:

$$|a| = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & & \vdots \\ \vdots & & \ddots & \\ a_{n1} & & \dots & a_{nn} \end{bmatrix}$$



شکل ۲-۷) شکل یک عضو مربعی در $\tau = 0$

برای $n = 2$:

$$|a| = a_{11}a_{22} - a_{12}a_{21}$$

برای $n = 3$:

$$|a| = a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} - a_{13}a_{22}a_{31} - a_{11}a_{23}a_{32} - a_{12}a_{21}a_{33}$$

دستور مطلب برای محاسبهٔ دترمینان به صورت زیر می‌باشد:

```
det(a)
```

به عنوان مثال اگر a به شکل زیر تعریف شود؛

$$a = \begin{bmatrix} 1 & 3 \\ 4 & 2 \end{bmatrix}$$

در این صورت؛

```
a=[1 3; 4 2];
d=det(a)
```

پس از اجرا $d = -10$ را ارائه خواهد کرد.

دسته ای از مسائل که در کاربردهای متعدد مهندسی اتفاق می افتند در نهایت به دترمینانی به فرم زیر تبدیل می شوند:

$$|a - \lambda b| = 0$$

که در آن a و b ماتریسهای $(n \times n)$ و λ_iها به ازای $j=1,2,.....n$ ، ریشه‌های این معادله می‌باشند. (گاهاً به این ریشه ها مقادیر ویژه نیز می‌گویند.) حل این معادلهٔ چند جمله‌ای توسط دستور زیر بدست می‌آید:

```
lambda=eig(a,b)
```

مثال ۲ – ۶ تبدیل یک چند جمله ای

چند جمله ای به شکل ؛

$$ax^2 + by^2 + cz^2 + 2dxy + 2exz + 2gyz$$

که در آن a، b، c، d، e و g اعداد حقیقی می‌باشند را می‌توان به شکل قطری اصلی مطابق زیر تبدیل نمود:

$$r_1x'^2 + r_2y'^2 + r_3z'^2$$

که در آن x'، y' و z' مختصه‌های دستگاه مختصات جدید می‌باشند که مبدأاش در $(0,0,0)$ و $r_1 \geq r_2 \geq r_3$, ریشه های دترمینان زیر هستند؛

$$|A - rI| = 0$$

و A ماتریس متقارن حقیقی مطابق زیر می‌باشد:

$$A = \begin{bmatrix} a & d & e \\ d & b & g \\ e & g & c \end{bmatrix}$$

چند جمله‌ای زیر را در نظر بگیرید:

$$4x^2 + 3y^2 - z^2 - 12xy + 4exz - 8gyz$$

بنابراین ؛

$$A = \begin{bmatrix} 4 & -6 & 2 \\ -6 & 3 & -4 \\ 2 & -4 & -1 \end{bmatrix}$$

جهت تعیین ریشه های r_i داریم:

```
r=eig([4 -6  2;-6  3 -4;2 -4 -1])
```

که پس از اجرا بردار ستونی $r = [-4.0000 \ -1.0000 \ 11.0000]'$ بدست می‌آید. دقت کنید که ریشه ها به ترتیب خاصی ندارند. جهت مرتب کردن آنها به شیوهٔ دلخواه، از تابع *sort* استفاده می کنیم. اما تابع *sort* این مقادیر را به ترتیب صعودی مرتب می کند (از منفی ترین به مثبت ترین). لذا می‌توان هر کدام از عبارات زیر را که برای هر ترکیب مثبت و منفی مقادیر حقیقی قابل اجراست، جهت مرتب کردن مقادیر به ترتیب نزولی به کار برد.

شکل #۱

```
r=-sort(-r)
```

که خروجی آن به شکل r=[11.0000 -1.0000 -4.0000]' می باشد؛

شکل #۲

```
r=flipud(sort(r))
```

که خروجی آن مجدداً به صورت *r=[-11/0000 -1/0000- 4/0000]'* می‌باشد. اگر *r* یک بردار سطری باشد، به جای دستور *flipud* می توان، از دستور *fliplr* استفاده کرد.

بنابراین شکل قطری حقیقی این چند جمله ای به صورت زیر می باشد :

$$11x'^2 - y'^2 - 4z'^2$$

می‌توان این عبارات را به شکل فشرده تر مطابق زیر نوشت:

```
r=-sort(-eig([4 -6 2;-6 3 -4;2 -4 -1]))
```

۲ – ۶ – ۵ حل یک دستگاه معادلات

سیستم زیر متشکل از n معادله و n مجهول x_k، $k = 1,2,...,n$ را در نظر بگیرید:

$$\begin{aligned}
&a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n = b_1 \\
&a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n = b_2 \\
&\vdots \\
&a_{n1}x_1 + a_{n2}x_2 + ... + a_{nn}x_n = b_n
\end{aligned}$$

می توانیم این دستگاه معادلات را به شکل ماتریسی زیر بنویسیم:

```
ax=b
```

که در آن a ماتریسی $(n \times n)$ می باشد؛

$$a=\begin{bmatrix} a_{11} & a_{12} & & a_{1n} \\ a_{21} & a_{22} & \cdots & \vdots \\ \vdots & & \ddots & \\ a_{n1} & & \cdots & a_{nn} \end{bmatrix}$$

و x و b بردارهایی ستونی از مرتبهٔ $(n\times 1)$ هستند؛

$$x=\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad b=\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

حل این معادله با ضرب طرفین معادله، در a^{-1} بدست می آید. بنابراین؛

$$a^{-1}ax=a^{-1}b$$
$$x=a^{-1}b$$

از آنجایی که $a^{-1}a=I$ و $Ix=x$ می باشد، دستور حل این دستگاه معادلات مطابق زیر خواهد بود[1]:

```
x=a\b
```

که در آن عملگر تقسیم وارون، بیانگر تقسیم ماتریسی می‌باشد که در مطلب مبین تقسیم ماتریس سمت چپ می‌باشد. تقسیم وارون روندی را در محاسبات طی می‌کند که در مقایسه با معادله‌های، زیر از پایداری عددی بیشتری برخوردار است.

```
x=a^-1*b
```

و

---

[1] دستور $a\backslash b$ می تواند حتی هنگامیکه a ماتریس مربعی نباشد، استفاده شود. در حالیکه دستور $inv(a)$ تنها هنگامیکه a ماتریس مربعی می‌باشد، مورد استفاده قرار می‌گیرد. یعنی اگر a ماتریسی $(m\times n)$، x برداری $(n\times 1)$ و b برداری $(m\times 1)$ باشد اگر داشته باشیم $ax=b$ آنگاه تقسیم وارون $a\backslash b$ مقادیر $x=cb$ را که در آن $c=(a'a)^{-1}a'$، معکوس مجازی a، خواهد یافت.

```
x=inv(a)*b
```

برای مثال معادلات زیر را در نظر بگیرید:

$$8x_1 + x_2 + 6x_3 = 7.5$$
$$3x_1 + 5x_2 + 7x_3 = 4$$
$$4x_1 + 9x_2 + 2x_3 = 12$$

که فرم ماتریسی آن به صورت زیر است:

$$\begin{bmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7.5 \\ 4 \\ 12 \end{bmatrix}$$

حل معادله توسط برنامهٔ زیر حاصل خواهد شد:

```
a = [8 1 6 ;3 5 7 ;4 9 2 ];
b = [ 7.5 4 12 ]';
x = a\b
```

که خروجی آن به صورت $x=[1.2931 \quad 0.8972 \quad -0.6236]'$ می باشد. این برنامه را می توان به شکل فشرده تر زیر نوشت:

```
x = [8 1 6; 3 5 7; 4 9 2]\[7.5 4 12]'
```

برای امتحان کردن درستی پاسخ، در صورتی که:

```
z = a*x
```

خواهیم دید که از لحاظ عددی $z = b$ است.

مثال ۲-۷ حل یک دستگاه معادلات کوپله شده

جهت تعیین حل خیز استاتیکی یک صفحه مربعی که هر چهار لبه‌اش کاملاً مقیود و بارگذاری یکنواختی روی سطح آن اعمال شده است، ابتدا باید ثوابت E_m را از طریق برش دستگاه معادلات نامحدود زیر به دست آورد[1]:

$$a_i E_i + \sum_{m=1,3,\ldots} b_{im} E_m = c_i \qquad i = 1,3,5,\ldots$$

که در آن

$$a_i = \frac{1}{i}\left(\tanh\alpha_i + \frac{\alpha_i}{\cosh^2\alpha_i} \right)$$

$$b_{im} = 8i\left(\pi m^3 \left(1 + \frac{i^2}{m^2} \right)^2 \right)^{-1}$$

$$c_i = \frac{4}{\pi^3 i^4}\left(\frac{\alpha_i}{\cosh^2\alpha_i} - \tanh\alpha_i \right)$$

و $\alpha_i = \frac{i\pi}{2}$ می باشد. اگر تنها چهار جملهٔ اول این سیستم را در نظر بگیریم،در این صورت داریم:

$$\begin{bmatrix} a_1 + b_{11} & b_{13} & b_{15} & b_{17} \\ b_{31} & a_3 + b_{33} & b_{35} & b_{37} \\ b_{51} & b_{53} & a_5 + b_{55} & b_{57} \\ b_{71} & b_{73} & b_{75} & a_7 + b_{77} \end{bmatrix} \begin{bmatrix} E_1 \\ E_3 \\ E_5 \\ E_7 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_3 \\ c_5 \\ c_7 \end{bmatrix}$$

که حل این دستگاه چهار معادله، چهار مجهول توسط برنامه زیر بدست خواهد آمد:

```
m=1:2:7;i=m;alp=m*pi/2;
ai=(tanh(alp)+alp./(cosh(alp).^2))./I;          %(1×4)
```

---

[1] *S.Timoshenko and S. Woinowsky - Kriger. Theory of Plates and Shells, McGraw-Hill New York 1959,PP.197-202.*

```
ci=4.*(alp./(cosh(alp).^2)-tanh(alp))./((pi^3)*i.^4);      %(1×4)
[ii,mm]=meshgrid(i,m);                                     %(4×4)
bim=(8/pi)*ii./(((1+(ii.^2)./(mm.^2)).^2).*mm.^30);        %(4×4)
format longe
ee=(diag(ai)+bim)\ci'
format short
```

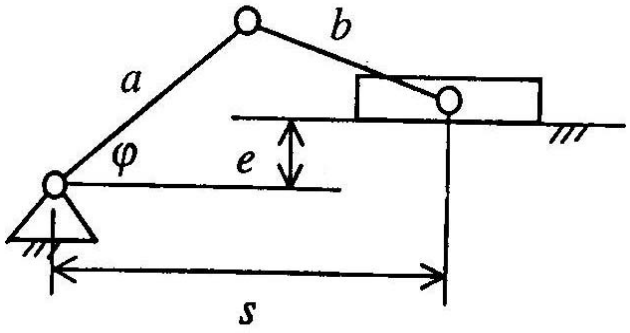
تابع *meshgrid* دو ماتریس (4×4) را به شیوه‌ای که در معادله (۲-۳) نشان داده شد، ایجاد می‌کند، که برای تعیین *bim* استفاده می شوند.(معادلهٔ (۲ - ٤) را ملاحظه کنید.)

حتی اگر اندیسها اعداد فرد باشند، طول هر بردار *4* خواهد بود. اجرای این برنامه منجر به ایجاد یک بردار ستونی می‌شود که مقادیر عددی زیر جهت وضوح بیشتر آورده شده است:

$$ee \rightarrow [-0.048000 \quad 0.004903 \quad 0.002296 \quad 0.001111]'$$

بنابراین:

$$E_1 = ee(1,1) = -0.048000;\ E_3 = ee(2,1) = 0.004903;\ E_5 = ee(3,1) = 0.002296;$$
$$E_7 = ee(4,1) = 0.001111;$$



شکل ۲-۸) مکانیزم لغزنده - لنگ

تمرین ها

بخش ۲-۳

۲ - ۱ دو بردار که یکی از آنها دارای عناصر $2n-1$ و دیگری دارای عناصر $2n+1$ و $n = 0,1,...,7$ باشد بسازید، اولی را a و دومی را b بنامید.

الف) مجموع a و b چیست ؟

ب) تفاضل a و b چیست ؟

پ) حاصلضرب $a'b$ چیست و حاصلضرب دترمینانش چقدر است ؟

ت) حاصلضرب ab' چیست ؟

۲ - ۲ بردار $x=[17\ -3\ -47\ 5\ 29\ -37\ 51\ -7\ 19]$ داده شده است. برنامه‌ای بنویسید که عناصر x را به صورت بردار زیر مرتب کند:

```
Y=[-3 -7 -37 -47 51 29 19 17 5]
```

این برنامه باید به گونه‌ای نوشته شود که قابل اعمال روی یک بردار با طول دلخواه باشد. مقدار صفر را با علامت منفی به بردار کلی نسبت دهید: یعنی هنگامیکه 0 یک عنصر بردار باشد، باید y به عنوان اولین عنصر بردار قرار داده شود.

بخش ۲-۴

۲ - ۳ فرض کنید $z = magic(5)$ باشد.

الف) عملیات زیر را در مورد z به ترتیب آورده شده در زیر انجام دهید:

۱) ستون دوم را بر $\sqrt{3}$ تقسیم کنید.

۲) عناصر سطر سوم را با عناصر سطر پنجم جمع کنید. (سطر سوم بدون تغییر باقی خواهد ماند).

۳) ستون اول را در ستون چهارم ضرب کنید و نتیجه را در ستون اول قرار دهید.

٤) عناصر قطر اصلی را 2 قرار دهید.

ب) نتیجهٔ نهائی بدست آمده در قسمت (الف) را q بنامید، سپس عناصر قطری qq' را نمایش دهید. [پاسخ: $[\,111024\ \ 44522\ \ 7300\ \ 104189\ \ 486\,]'$.]

پ) مربع هر عضو ماتریس q را نمایش دهید.

بخش ۲ – ۵

۲ – ۴ جابجایی لغزنده در مکانیزم لغزنده – لنگ نشان داده شده در شکل ۲– ۸، توسط رابطهٔ زیر بدست می آید:

$$s = a\cos(\phi) + \sqrt{b^2 - (a\sin(\phi) - e)^2}$$

جابجائی s را بصورت تابعی از زاویه (برحسب درجه) که در آن $e = 0.3,\ b = 1.5,\ a = 1$ می‌باشد، در بازهٔ $0 \le \phi \le 360$ ترسیم کنید. در اینجا از تابع $plot(\phi, s)$ استفاده کنید.

۲ – ۵ درصدی از توان کل p در سری تناوبی متشکل از پالسهای مستطیلی به صورت تابعی از تعداد عبارات، N_H، در سری گسترش یافته اش به صورت زیر می باشد:

$$p = 100\,p_o / p_T\ \%$$

که در آن p_T توان کلی بدون بعد سیگنال می باشد؛

$$p_o = 1 + 2\sum_{n=1}^{N_H} \frac{\sin^2(n\pi\tau_0 / T)}{(n\pi\tau_0 / T)^2}$$

و $\frac{\tau_0}{T}$ نسبت دوام پالس به پریودش است. اگر فرض کنیم $\frac{\tau_0}{T} = \frac{1}{\sqrt{19}}$، در این صورت $P_T \cong 4.3589$ می باشد. درصد توان کلی را به عنوان تابعی از N_H در بازهٔ $2 \le N_H \le 25$ رسم کنید.

۲ – ۶ حاصلضرب زیر را در نظر بگیرید[1]؛

---

[1] *L.B.W, Jolly, ibid.*

$$S_N = \prod_{n=1}^{N}\left(1-\frac{x^2}{n^2-a^2}\right)$$

که در آن وقتیکه $N \to \infty$ داریم :

$$S_\infty = \frac{a}{\sin \pi a\sqrt{a^2+x^2}} \sin\left(\pi\sqrt{a^2+x^2}\right)$$

درصد خطا به شکل زیر تعریف می شود:

$$e_N = 100\frac{S_N - S_\infty}{S_\infty}\ \%$$

اگر x از 1 تا 5 با مقدار افزایش 0.5 تغییر کند و $a=\sqrt{2.8}$ باشد، در اینصورت درصد خطای مقادیر x را هنگامیکه $N=100$ است، محاسبه کنید. (از تابع $prod$ استفاده کنید.)

{پاسخ: $e_{100}=[1.0001\ \ 2.2643\ \ 4.0610\ \ 6.4176\ \ 9.3707\ \ 129670\ \ 17.2642\ \ 223330\ \ 28.2588]$.}

۲- ۷ یکی از ابزارهای بدست آوردن تقریبی از پارامتر δ که در تابع چگالی احتمال وایبول ظاهر می‌شود (قسمت ١٤ - ٢ - ٢ و تمرین ٥ - ٢ (١) را ملاحظه کنید) رابطهٔ زیر می باشد:

$$\delta = \left[\frac{1}{n}\sum_{i=1}^{n} x_i^{\ \beta}\right]^{1/\beta}$$

که در آن x_i ها نمونه‌های تصادفی به اندازهٔ n و β پارامتر دیگری می‌باشد. اگر

$$x=[72\ \ 82\ \ 97\ \ 103\ \ 113\ \ 117\ \ 126\ \ 127\ \ 127\ \ 139\ \ 154\ \ 159\ \ 199\ \ 207]$$

و $\beta = 3.644$ باشد، در اینصورت مقدار δ را تعیین کنید. { پاسخ: $\delta = 144.2741$.}

۲ - ۸ تبدیل از دستگاه کروی به دستگاه کارتزین توسط روابط زیر مشخص می شود؛

$$x = b\sin\phi\cos\theta$$

$$y = b\sin\phi\sin\theta$$

$$z = b\cos\phi$$

10 مقدار با فاصلهٔ یکسان برای متغیر ϕ در بازهٔ $0 \le \phi \le 90$ و 24 مقدار دیگر با فاصلهٔ یکسان برای θ در بازهٔ $0 \le \theta \le 360$ انتخاب کنید و نیمکره‌ای را با استفاده از دستور *mesh(x,y,z)* هنگامیکه $b=2$ می باشد، رسم کنید.

۲ – ۹ حاصل سری زیر را برای $n=25$ و 5 مقدار با فاصلهٔ یکسان در بازهٔ $0.1 \le x \le 1$ بدست آورید.

$$\sum_{N\to-\infty}^{N\to\infty} \frac{1}{n^4+x^4} = \frac{2\pi^4}{y^3}\frac{\sinh y+\sin y}{\cosh y-\cos y} \qquad y=\pi x\sqrt{2}$$

این مقادیر را با مقادیر حقیقی‌شان مقایسه کنید.

۲ – ۱۰ نمایش سری تابع بسل نوع اول از مرتبهٔ n به صورت زیر می باشد:

$$J_n(x) = \sum_{k=0}^{k\to\infty} \frac{(-1)^k (x/2)^{2k+n}}{k!\,\Gamma(k+1+n)}$$

اگر $k=25$ باشد آنگاه برداری متشکل از مقادیر $J_n(x)$ برای $n=2$ و نیز 6 مقدار با فاصلهٔ یکسان در بازهٔ $1 \le x \le 6$، ایجاد کنید. تابع گاما Γ و فاکتوریل، هر دو توسط تابع *gamma* قابل دستیابی می باشند. پاسخ خود را با مقادیر به دست آمده از تابع داخلی *besselj (n,x)* مقایسه کنید.

بخش ۲ – ۶ – ۲

۲ – ۱۱ به ماتریسی، ماتریس اگر در ماتریسی داشته باشیم:

$$x'x=I$$

در اینصورت به آن ماترس ارتوگونال گفته می‌شود و در نتیجه $(x'x)^{-1} = I$ خواهد بود. نشان دهید که هر یک از ماتریس‌های زیر ارتوگونال هستند. (بخش ٤ – ۳ – ۱ را نیز ملاحظه کنید.)

$$w = \frac{1}{2}\begin{bmatrix} -1 & -1 \\ 1 & -1 \\ -1 & 1 \\ 1 & 1 \end{bmatrix}$$

$$q=\frac{1}{2}\begin{bmatrix}1 & -1 & -1 & 1\\ 1 & 1 & -1 & -1\\ 1 & -1 & 1 & -1\\ 1 & 1 & 1 & 1\end{bmatrix}$$

۲ -۱۲ بازوی سه درجه آزادی صفحه‌ای شکل ۲ – ۹ را در نظر بگیرید. مکان و جهت قرارگیری نقطهٔ o_3، با توجه به دستگاه مختصات لخت o_0 به صورت زیر می باشد:

$$T_3 = A_1A_2A_3$$

که Aj و به صورت زیر می باشند ؛

$$A_j=\begin{bmatrix}\cos\theta_j & -\sin\theta_j & 0 & a_j\cos\theta_j\\ \sin\theta_j & \cos\theta_j & 0 & a_j\sin\theta_j\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \qquad j=1,2,3$$

و

$$T_3=\begin{bmatrix}u_x & v_x & 0 & q_x\\ u_y & v_y & 0 & q_y\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

مؤلفه‌های q_x و q_y، مختصه های x و y نقطهٔ o_3 نسبت به دستگاه مختصات قرار گرفته در o_o می‌باشند. اگر $\theta=30^\circ$ ، $j=1,2,3$ و $a_1=1$، $a_2=2$ و $a_3=3$ باشد، در این صورت مکان نقطهٔ o_3 نسبت به دستگاه مختصات قرار گرفته در o_o و جهت قرارگیری محورهای x_3 و y_3 را بیان کنید. { پاسخ: $q_x=1.8660$، $q_y=5.2321$، x_3 موازی y_0 و y_3 موازی x_0 ، اما در خلاف جهت می‌باشند.}

۲ – ۱۳ در تحلیل میانیابی چند خطی، مقدار ماتریس زیر دارای چند کاربرد می‌باشد (مثال ۱۴– ۱۳ را ملاحظه کنید).

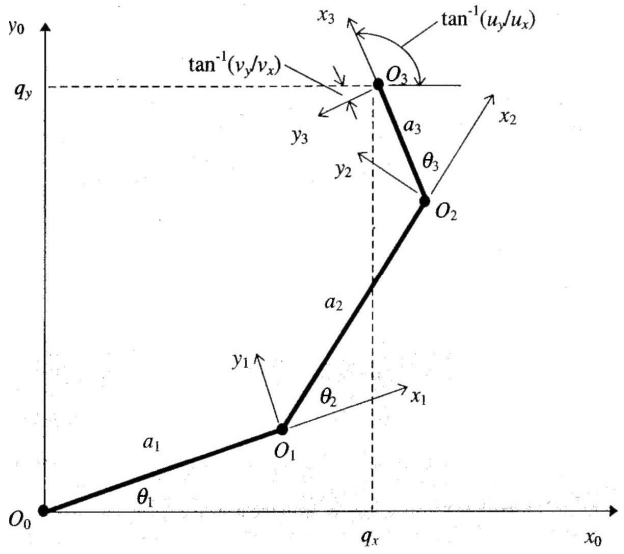
$$H=X(X'X)^{-1}X'$$

اگرداشته باشیم :

$$X=\begin{bmatrix} 17 & 31 & 5 \\ 6 & 5 & 4 \\ 19 & 28 & 9 \\ 12 & 11 & 10 \end{bmatrix}$$

آنگاه عناصر قطری H را تعیین کنید.

{ پاسخ: $[0.7294 \quad 0.9041 \quad 0.4477 \quad 0.9188]'$ =diagonalH.}



شکل ۲- ۹) بازوی سه درجه آزادی صفحه‌ای

۲ – ۱۴ هر یک از سریهای آورده شده در زیر[1] را در بازهٔ تعیین شدهٔ τ رسم کنید. در صورتی که تعداد جمله‌ها تعیین نشده، 200 جملهٔ اول را جهت جمع کردن در مورد هر سری به کار ببندید. برای بدست آوردن پاسخ ها از دستور sum استفاده نکنید.

الف) موج مربعی

$$f(\tau)=\frac{4}{\pi}\sum_{n=1,3,5,\ldots}^{105}\frac{1}{n}sin(2n\pi\tau) \qquad -\frac{1}{2}\le\tau\le\frac{1}{2}$$

[1] *H.P. Hsu,ibid.*

ب) موج دندان اره‌ای

$$f(\tau)=\frac{1}{2}+\frac{1}{\pi}\sum_{n=1}^{100}\frac{1}{n}\sin(2n\pi\tau) \qquad -1\le\tau\le 1$$

پ) موج دندان اره‌ای

$$f(\tau)=\frac{1}{2}-\frac{1}{\pi}\sum_{n=1}^{100}\frac{1}{n}\sin(2n\pi\tau) \qquad -1\le\tau\le 1$$

ت) موج مثلثی

$$f(\tau)=\frac{\pi}{2}-\frac{4}{\pi}\sum_{n=1}^{100}\frac{1}{(2n-1)^2}\cos((2n-1)\pi\tau) \qquad -1\le\tau\le 1$$

ث) موج سینوسی یکسو شده

$$f(\tau)=\frac{\pi}{2}+\frac{4}{\pi}\sum_{n=1}^{100}\frac{1}{1-4n^2}\cos(2n\pi\tau) \qquad -1\le\tau\le 1$$

ج) موج نیم سینوسی

$$f(\tau)=\frac{1}{\pi}+\frac{1}{2}\sin\pi\tau-\frac{2}{\pi}\sum_{n=2,4,6,\ldots}^{106}\frac{\cos n\pi\tau}{n^2-1} \qquad -2\le\tau\le 2$$

چ) موج توانی

$$f(\tau)=\frac{e^{2\pi}-1}{\pi}\left[\frac{1}{2}+\sum_{n=1}^{250}\frac{1}{1+n^2}(\cos n\tau-n\sin n\tau)\right] \qquad 0\le\tau\le 4\pi$$

350 مقدار τ را جهت نمایش استفاده کنید.

ح) موج ذوزنقه ای

$$f(\tau)=\frac{4}{\alpha^2}\sum_{n=1,3,5,\ldots}^{105}\frac{\sin n\pi\alpha}{(n\pi)^2}\sin n\pi\tau \qquad -2\le\tau\le 2$$

$\alpha=0.25$ در نظر بگیرید.

۲ – ۱۵ دو سری زیر را در نظر بگیرید[1]:

$$S_{1N} = \sum_{n=1}^{N} \frac{cos(n\theta)}{n^2 + a^2} \qquad 0 \prec \theta \prec \pi$$

$$S_{2N} = \sum_{n=1}^{N} \frac{n\,sin(n\theta)}{n^2 + a^2} \qquad 0 \prec \theta \prec 2\pi$$

که در آن وقتی $N \to \infty$ ، داریم:

$$S_{1\infty} = \frac{\pi\, cosh[a(\pi - \theta)]}{2a\, sinh\, \pi a} - \frac{1}{2a^2} \qquad 0 \prec \theta \prec \pi$$

$$S_{2\infty} = \frac{\pi\, sinh[a(\pi - \theta)]}{2\, sinh\, \pi a} \qquad 0 \prec \theta \prec 2\pi$$

درصد خطا به شکل زیر تعریف می شود:

$$e_{jN} = 100 \frac{S_{jN} - S_{j\infty}}{S_{j\infty}}\ \% \qquad j = 1,2$$

اگر θ از 10° تا 80° با نرخ 10° افزایش زیاد یابد و $a = \sqrt{3}$ باشد، در اینصورت درصد خطا را برای این دو سری در این هشت مقدار θ هنگامیکه $N=25$ می باشد، محاسبه نمایید.

(مسئله را بدون استفاده از تابع *sum* حل کنید.)

پاسخ: $e_1 = [-1.2435 \quad 0.8565 \quad 0.8728 \quad -1.9417 \quad -0.9579 \quad -8.1206 \quad 0.7239 \quad 1.1661]$ }

و $\{\ e_2 = [8.0538 \quad 10.4192 \quad -8.9135 \quad -5.4994 \quad 12.9734 \quad -0.5090 \quad -17.2259 \quad 11.2961]$

۲ – ۱۶ توزیع دمایی حالت پایدار بی بعد صفحه ای مستطیلی که لبه های آن دارای دمای ثابت $\eta = 1$ هستند توسط سری زیر بیان می شود[2]:

$$T(x,y) = \frac{4}{\pi} \sum_{n=1,3,5,\ldots}^{\infty} \frac{sinh(n\pi\alpha\eta)}{n\, sinh(n\pi\alpha)} sin(n\pi\alpha)$$

که در آن $\eta = \frac{x}{d}$، $\xi = \frac{y}{b}$، d و b طولهای صفحه به ترتیب درجهات x، y و $\alpha = \frac{d}{b}$، $0 \le \eta \le 1$ و $0 \le \xi \le 1$ می باشد.

---

[1] *L.B.W.Jolly, ibid.*
[2] *H.P.Hsu,ibid.*

توزیع دمایی صفحه را هنگامیکه $\alpha = 2$ می باشد، با استفاده از تابع *mesh* نمایش دهید.
فرض کنید $\Delta\eta = \Delta\xi = \frac{1}{14}$ باشد.(برای حل مسئله از *sum* استفاده نکنید.)

۲ – ۱۷ موجی که در یک ریسمان منتشر می‌شود ، دارای سرعت اولیهٔ صفر و جابجائی اولیهٔ؛

$$u(\eta,0) = \frac{\eta}{a} \qquad 0 \le \eta \le a$$

$$u(\eta,0) = \frac{1-\eta}{1-a} \qquad a \le \eta \le 1$$

می باشد؛ جابجایی آن توسط سری زیر محاسبه می شود:

$$u(\eta,0) = \frac{2}{a\pi(1-a)} \sum_{n=1}^{N\to\infty} \frac{\sin n\pi a}{n^2} \sin(n\pi\eta)\cos(n\pi\tau)$$

$u(\eta,\tau)$ را هنگامیکه $N = 50$، $a = 0.25$، $\Delta\eta = 0.05$ و $\Delta\tau = 0.05$ می باشد، در بازهٔ $0 \le \tau \le 2$ نمایش دهید. (برای حل این مسئله از تابع *sum* استفاده نکنید.)

نتیجهٔ این حل در شکل ۲–۱۰ که پس از استفاده از گزینه *rotate* در پنجرهٔ *figure* بدست می آید، نشان داده شده است.

بخش ۲ – ۶ – ۵

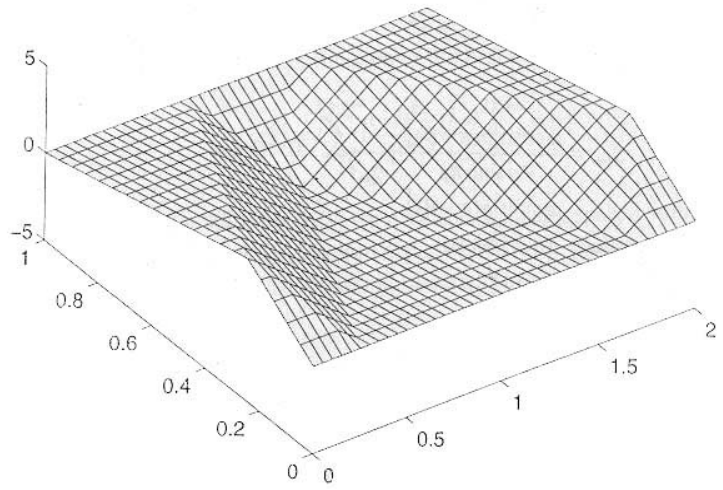
۲ – ۱۸ دستگاه معادلات زیر داده شده است ؛

$$16s + 32u + 33p + 13w = 91$$
$$5s + 11u + 10p + 8w = 16$$
$$9s + 7u + 6p + 12w = 5$$
$$34s + 14u + 15p + w = 43$$

مقادیر s، u، p و w را تعیین کرده و مقدار دترمینان ضرایب را مشخص کنید؟

{پاسخ: $s = -0.1258$، $u = -8.7133$، $p = 11.2875$، $w = -0.0500$ و

$determinant = 7.680$}



شکل ۲-۱۰) انتشار یک جابجایی اولیه در یک طناب

۲ - ۱۹ دو استوانهٔ بلند با دو جنس متفاوت که یکی در داخل دیگری جا داده شده است، را در نظر بگیرید. شعاع داخلی استوانهٔ کوچک تر a و شعاع خارجی آن b می باشد. همچنین شعاع داخلی استوانهٔ بزرگتر b و شعاع خارجی آن c می باشد. مدول یانگ و ضریب پواسان استوانهٔ داخلی به ترتیب، E_1 و ν_1 در مورد استوانهٔ خارجی E_2 و ν_2 می باشد. تنش شعاعی σ_{rr} ، تنش هوپ $\sigma_{\theta\theta}$ و جابجایی u_r به ترتیب توسط روابط زیر داده شده‌اند:

$$\sigma_{rri}(r)=\frac{A_i}{r^2}+B_i$$

$$\sigma_{\theta\theta i}(r)=\frac{-A_i}{r^2}+B_i$$

$$u_{ri}(r)=\frac{-(1+\upsilon_i)}{rE_i}A_i+\frac{(1-\upsilon_i)}{E_i}rB_i$$

که در آن $i=1$ مربوط به استوانهٔ داخلی و $i=2$ مربوط به استوانهٔ خارجی می باشد.

اگر سطح خارجی استوانهٔ خارجی در معرض جابجایی شعاعی فشاری U_0، و سطح داخلی استوانهٔ داخلی فاقد تنش شعاعی باشد، در اینصورت چهار شرط مرزی زیر می تواند جهت تعیین A_i و B_i، $i=1,2$ استفاده گردد:

$$\sigma_{rr1}(a)=0$$

$$\sigma_{rr1}(b) = \sigma_{rr2}(b) \qquad (b)$$

$$u_{r1}(b) = u_{r2}(b)$$

$$u_{r2}(c) = -U_0$$

قرار دادن معادلات (الف) در (ب) منجر به دستگاه معادلات زیر خواهد شد:

$$\begin{bmatrix} 1 & a^2 & 0 & 0 \\ 1 & b^2 & -1 & -b^2 \\ -(1+\nu_1) & (1-\nu_1)b^2 & (1+\nu_2)E_1/E_2 & -(1-\nu_2)b^2E_1/E_2 \\ 0 & 0 & -(1+\nu_2) & (1-\nu_2)c^2 \end{bmatrix} \begin{bmatrix} A_1 \\ B_1 \\ A_2 \\ B_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -U_0E_2c \end{bmatrix}$$

تنش هوپ در استوانه های داخلی و خارجی در $r = b$، هنگامیکه $\nu_1 = \nu_2 = 0.4$،

$b = 0.25\ inch$ ،$a = 0.192\ inch$ ،$U_0 = 0.01 inch$ ،$E_2 = 3.5 \times 10^4\ psi$ ،$E_1 = 3 \times 10^5\ psi$

و $c = 0.312\ inch$ می‌باشد را محاسبه کنید.

فصل سوم

وارد کردن داده ها و نمایش نتایج

در این فصل شیوه‌های نمایش خروجیها همراه با توضیحاتی که در پنجرهٔ فرمان مطلب نمایش داده می‌شود و نحوهٔ ذخیره کردن و فراخوانی داده ها، نشان داده شده است.

۳-۱ چگونگی ایجاد رشته ها (حروف) و نمایش خروجیها همراه با توضیحات

مطلب اجازهٔ ایجاد و ذخیرهٔ ماتریسها و اعمال تغییرات روی آنها را با هر ترکیبی از حروف، اعداد و کاراکترهای خاص بعنوان کمیاتی که رشته نامیده می‌شوند، به کاربر می‌دهد. رشته ها به شیوه‌ای مشابه بردارها و ماتریسها تعریف می شوند، با این تفاوت که یک جفت کوتیشن ('. . .')، آنها را در بر می‌گیرد. هر کاراکتر موجود در رشته، یک عنصر بردار و یا ماتریس محسوب می‌شود.

مثال زیر را در نظر بگیرید. s را برابر با رشتهٔ $'testing123'$ قرار دهید. دستور مطلب جهت تعریف این رشته به صورت بردار

$s = 'testing123'$

یا

$s = ['testing123']$

می باشد. که در آن هر کاراکتر موجود در کوتیشن نشان دهندهٔ یک موقعیت در بردار s است.
بنابراین،

$s(7) \rightarrow g$

$s(3:6) \rightarrow stin$

$fliplr(s) \rightarrow 32\lg nitset$

و $length(s) = 10$ می باشد. اساس کار با رشته ها مشابه اعداد است. بنابراین عبارت،

$sc = [sfliplr\ (s)]$

رشتهٔ زیر را ایجاد می کند:

$testing123321gnitset$

که در آن،

$scs = [s; fliplr(s)]$

ماتریسی با ویژگیهای زیر می باشد:

$scs(1,:) \rightarrow testing\ 123$

$scs(2,:) \rightarrow 32\lg nitset$

توجه کنید که هر دو سطر scs دارای تعداد مشابهی کاراکتر (ستون) می باشند.
اگر در هر سطر ماتریس یک رشته قرار دهیم، روشی ساده جهت دستیابی به عبارات رشته‌ای در اختیار خواهیم داشت. البته رعایت این نکته که هر سطر ماتریس باید شامل تعداد یکسانی کاراکتر باشد ضروری است. این ضرورت را می‌توان با بکارگیری فضاهای خالی جهت یکسان نمودن تعداد کاراکترها هنگامی که عبارات رشته‌ها دارای طولهای نامساوی هستند، بر طرف نمود.

بنابراین اگر:

$lab = [' first':'last':'middle']$

در این صورت :

$lab(1,:) \rightarrow firstb$
$lab(2.:) \rightarrow lastbb$
$lab(3,:) \rightarrow middle$

خواهد بود. که در آن b مبین یک فضای خالی می‌باشد. خوشبختانه، مطلب، شیوه‌ای را برای برطرف کردن این مشکل توسط دستور زیر ارائه می دهد:

$str2mat$

بنابراین عبارات فوق را می‌توان با عبارت زیر که راحت تر قابل استفاده است جایگزین نمود:

$lab = str2mat('first','last','middle')$

که هر عبارت رشته‌ای یک سطر ماتریس lab با اندازهٔ (3×6) می‌باشد. فضاهای خالی را می‌توان توسط دستور زیر از میان برداشت:

$deblank$

جهت تبدیل یک مقدار عددی به یک رشته از دستور زیر استفاده می‌کنیم:

$num2str$

فرض کنید num یک عدد یا ماتریسی از اعداد باشد. در این صورت جهت تبدیل آن به یک رشته خواهیم داشت:

$z = num2str(num)$

که در آن z یک متغیر رشته‌ای می‌باشد.

تابع $num2str$ اغلب جهت قرار دادن خروجی‌های عددی مشخص در پنجرهٔ فرمان مطلب و یا روی یک شکل بکار می‌رود. یک ساختار متداول این دستور مقادیر عددی را با رشتهٔ متناظرشان جایگزین می کند. بنابراین اگر num متغیری عددی، مبین وزن بر حسب کیلوگرم باشد، که باید به صورت زیر نشان داده شود، در این صورت جهت نمایش آن در پنجرهٔ فرمان مطلب، از تابع $disp$ به صورت زیر استفاده می کنیم.

$num = 12.567;$
$z = num2str(num);$
$disp(['Product$

weight ='z'kg'])

یا به شکل فشرده تر از دستورات زیر استفاده می‌شود:

num = 12.567;

disp(['Product weight ='num2str(num)'kg'])

هنگامیکه دستورات فوق اجرا می شود، خروجی زیر را خواهیم داشت:

Product weight = 12.567kg

که در پنجرهٔ فرمان مطلب نمایش داده می‌شود. این رشته برداری با طول *26* می‌باشد. توجه کنید که فضاهای خالی به عنوان کاراکترهای رشته‌ای قابل قبول در نظر گرفته می‌شوند.

اگر *num* بردار وزن باشد، در این صورت

num = [12.567 3.458 9.111];

disp(['Product weight ='num2str(num)'kg'])

خروجی زیر را به دنبال خواهد داشت:

product weight = 12.567 3.458 9.111 kg

جهت ایجاد نشانه برای هر عنصر بردار *num*، از برنامهٔ زیر استفاده می کنیم:

num = [12.567 3.458 9.111];

n = length(num);

disp([repmat('Product weight =',n,1) num2str(num') repmat('kg',n,1)])

که پس از اجرا خروجی زیر را به همراه خواهد داشت:

product weight = 12.567 kg

product weight = 3.458 kg

product weight = 9.111 kg

اگر کاربر بخواهد بردار *num* را بدون برچسب نمایش دهد، در اینصورت:

```
num = [12.567 3.458 9.111];
disp(num)
```

که خروجی زیر را در پنجرهٔ فرمان مطلب نشان می دهد.

```
12.5670   3.4580   9.1110
```

در صورتیکه فایل متنی زیر:

```
num = [12.567 3.458 9.111];
disp(num')
```

نتیجهٔ زیر را بدنبال خواهد داشت:

```
12.5670
3.4580
9.1110
```

مطلب همچنین به کاربر اجازه می‌دهد تا تعداد ارقام عددی را که باید به رشته تبدیل شود، مطابق زیر تعیین نماید:

```
num2str(a,N)
```

که در آن a، عددی است که باید به رشته تبدیل شود و N تعداد ارقام آن می باشد. اگر تعداد ارقام تعیین شده کمتر از تعداد ارقام سمت چپ نقطه اعشار باشد، در این صورت مطلب عدد را به نمایش توانی آن با تعداد ارقام کافی (برابرN) تبدیل خواهد کرد. مثالهای زیر، که در تمامی آنها $a = 1000\pi = 3141.592653589$ می باشد را در نظر بگیرید:

```
num2str(a,1)  →  3e+003
num2str(a,3)  →  3.14e+003
num2str(a,4)  →  3142
num2str(a,5)  →  3141.5
num2str(a,8)  →  3141.5927
```

توجه کنید که نقطه اعشار (.) به عنوان یک رقم در نظر گرفته نمی‌شود.

یک تابع جایگزین که می‌تواند جهت نمایش داده های دارای فرمت در پنجرهٔ فرمان مطلب نمایش داده شود تابع *fprintf* می باشد که دارای برتری اندکی نسبت به تابع *disp* می باشد چون قادر است فرمت مقادیر عددی خروجی را بهتر کنترل کند.

شکل کلی استفاده از دستور *fprintf* جهت نمایش خروجی ها درپنجرهٔ فرمان مطلب بشکل زیر می باشد:

```
fprintf(1,'%....',variables)
```

که در آن اولین آرگومان ،'1'، تعیین می کند که خروجی‌ها باید در پنجرهٔ فرمان مطلب نمایش داده شوند. در آرگومان دوم دستوری که داخل کوتیشن قرار می‌گیرد فرمت متغیرهای خروجی را تعیین خواهد کرد. هنگامیکه متغیرها یک بردار و یا یک ماتریس باشند نسبت دهی فرمت بصورت ستون به ستون در قسمت اختصاص دهی فرمت انجام می‌شود. مرتبهٔ نسبت‌دهی فرمت متناظر با مرتبهٔ متغیرها می‌باشد. علامت % قبل از هر تعیین فرمت بکار برده می‌شود. یک فرمت متداول که اغلب مورد استفاده واقع می‌شود. مطابق شکل زیر می‌باشد:

```
x.yf
```

که در آن *f* یکی از انواع فرمت‌های متداول می‌باشد. برای آشنایی با فرمتهای دیگر نمایش خروجی ها، به فایلهای کمکی مطلب *(Matlab Help Files)* در مورد دستور *fprintf* مراجعه کنید. همچنین *x* یک عدد صحیح می‌باشد که تعداد کل ارقام عدد را تعیین می‌کند و *y* تعداد ارقامی که در سمت راست نقطه اعشار قرار می‌گیرند را معین می سازد. ما نحوهٔ استفاده از دستور *fprintf* را با نمایش بردار:

```
num=[12 -14 3.458 0.11167];
```

به شیوه‌های مختلف، نشان خواهیم داد.

جهت نمایش این بردار در یک خط با استفاده از دستور *fprintf* داریم:

```
num=[12 -14 3.458 0.11167];
fprintf(1,'%5.3f',num)
```

که نتیجهٔ زیر را در بر خواهد داشت:

```
>> 12.000 -14.000 3.458 0.112
```

توجه کنید که *num(1)* و *num(2)* دارای سه صفر اضافه شده به طرف راست نقطه اعشار می‌باشند، در حالیکه *num(4)* به سه رقم پس از نقطه اعشار گرد شده است. همچنین نشانهٔ "»" مبین این است

که در همان خطی که داده های خروجی نمایش داده شده‌اند، می‌توانیم عبارت مطلب دیگری را اجرا کنیم. برای رفتن به خط بعد کافیست تنها کلید *Enter* فشرده شود. اگر بخواهیم این مقادیر را به صورت ستونی دارای چهار عضو نمایش دهیم، از نشانهٔ "\n" که نوعی محدود کننده می‌باشد، مطابق زیر استفاده می کنیم.

```
num = [12  -14  3.458  0.11167];
fprintf(1,'%5.3f\n',num)
```

که خروجی زیر را نتیجه خواهد داد:

```
12.000
-14.000
3.458
0.112
```

جهت تولید مجدد چهار عدد با همان دقتی که اعداد دارا بوده‌اند، از دستورات زیر استفاده می‌کنیم.

```
num = [12  -14  3.458  0.11167];
fprintf(1,'%2.0f  %2.0f  %5.3f  %5.5f',num)
```

که در آن بین هر *f* و علامت % از دو فاصله استفاده نموده‌ایم. اجرای دستورات فوق خروجی زیر را نتیجه خواهد داد:

```
>> 12  -14  3.458  0.11167
```

جهت نشانه‌گذاری هر عدد، از دستورات زیر استفاده می‌کنیم:

```
num = [12  -14  3.458  0.11167];
fprintf(1,'weight = %2.0f kg  pressure = %2.0f  pa time = %5.3f s
length = %5.5f  m\n',num)
```

که عبارت زیر را نتیجه می‌دهد:

```
weight = 12 kg  pressure = -14  pa  time = 3.458  s  length = 0.11167  m
```

جهت نمایش مقادیر بصورت ستونی، از فایل متنی زیر استفاده می‌کنیم:

```
num = [12  -14  3.458  0.11167];
fprintf(1,'weight=%2.0f kg\n  pressure=%2.0f pa\n time=%5.3f s\n
 length=%5.5f  m\n',num)
```

که نتیجهٔ آن مطابق زیر خواهد بود:

```
weight = 12 kg
pressure = -14  pa
time = 3.458 s
length = 0.11167  m
```

اگر بخواهیم تمامی اعداد با فرمت یکسانی نمایش داده شوند، در این‌صورت می‌توانیم اختصاصات فرمت را ساده‌تر کرده و باز هم برای خروجی‌ها از نشانه استفاده کنیم، البته به شیوه‌ای خلاصه‌تر، مطابق زیر:

```
num = [12  -14  3.458  0.11167];
nn = 1 : length ( num );
fprint f(1,'x(%1.0f ) = %7.5f \n',[nn;num])
```

که نتیجهٔ زیر را به دنبال خواهد داشت:

```
x(1) = 12.00000
x(2) = -14.00000
x(3) = 3.45800
x(4) = 0.11167
```

۳-۲ وارد کردن داده ها توسط دستور *input*

آرایه‌ای از داده‌ها ممکن است توسط یک تابع و یک فایل متنی درخواست شود و سپس کاربر آنها را توسط دستور *input* وارد نماید. به علاوه، دستور *input* می تواند پیغامی را که توسط کاربر جهت تعیین اینکه چه چیزی باید وارد شود، در پنجرهٔ فرمان مطلب نشان دهد. شکل حقیقی داده ها بـه ایـن

موضوع که داده ها، اسکالر، بردار یا ماتریس و اینکه این مقادیر عدد و یا رشته هستند بستگی خواهد داشت. اکنون قصد داریم تا این موارد مختلف را نمایش دهیم. سایر روشهای ورود داده‌ها در بخش ۳ – ۳ و ۵ – ۲ مورد بررسی قرار خواهد گرفت.

۳ – ۲ – ۱ وارد کردن یک اسکالر

برای وارد کردن یک کمیت عددی از دستور زیر استفاده می کنیم.

```
InputData = input(' Enter the temperature in deg rees C :' );
```

که پس از اجرا عبارت زیر را در پنجرهٔ فرمان مطلب نشان خواهد داد.

```
Enter the temperature in deg rees C : 121.7
```

که عدد *121.7* توسط کاربر وارد شده است. علامت نقطه ویرگول به کار رفته در انتهای دستور فوق مانع از تکرار مقادیر وارد شده می‌شود.

همچنین کاربر می‌تواند تغییراتی را بر روی مقادیر وارد شده، در همان عبارت محاسباتی اعمال کند. برای مثال:

```
InputData= input(' Enter the starting angle in deg rees:' )* pi / 180;
```

که عبارت زیر را نشان می دهد:

```
Enterthe startinganglein degrees: 45
```

که در آن مقدار *45* توسط کاربر وارد شده است. به هر حال، مقدار متغیر *Input Data*، *0.7854* $(=45\pi/180)$ می باشد.

به عنوان مثال دیگری، به تبدیل درجهٔ حرارت از درجهٔ سانتیگراد $(^\circ C)$ به درجهٔ فارنهایت $(^\circ F)$ توجه کنید.

```
InputData= 1.8* input(' Enter the temperature in degrees C :' )+32;
```

که عبارت زیر را نمایش خواهد داد:

```
Enter the temperature in degrees C : 100
```

مقدار *100* توسط کاربر وارد شده است. مقدار متغیر *Input Data*، *212* خواهد بود.

ممکن است کاربر بخواهد پیغامی را در چند خط در پنجرهٔ فرمان مطلب، نشان دهد. در اینصورت باید از نشانهٔ "\n" در موقعیتهای مناسب استفاده نماید. بنابراین،

```
InputData=input('Enterthe startingangle\n in degrees:')*pi/180;
```

خروجی زیر را به دنبال خواهد داشت.

```
Enter the starting angle\n in degrees: 45
```

۳ -۲ -۲ وارد کردن یک رشته

جهت وارد کردن یک کمیت رشته ای از دستور زیر استفاده می‌کنیم:

```
InputData = input(' Enter  file name, including its extension :','s' );
```

که عبارت زیر را در پنجرهٔ فرمان مطلب نشان خواهد داد.

```
Enter  file name, including its extension : DataSet3.txt
```

که در آن رشتهٔ *Data Set3.txt* توسط کاربر وارد شده است. توجه کنید که نیازی به استفاده از علامت کوتیشن نمی‌باشد. که این مورد به خاطر استفاده از علامت 's' به عنوان آرگومان دوم دستور *input* میسر شده است.

۳ -۲ -۳ وارد کردن یک بردار

جهت وارد کردن یک بردار شامل مقادیر عددی از دستور زیر استفاده می کنیم.

```
InputData=input('Enter the temperatures in degrees C :');
```

که عبارت زیر را در پنجرهٔ فرمان مطلب نشان خواهد داد.

```
Enterthe temperatures in degrees C : [120 141 169 201]
```

که در آن بردار شامل اعداد *[120 141 169 201]* توسط کاربر وارد شده است.

استفاده از قلاب در اینجا ضروری می باشد. اگر قرار باشد که یک بردار ستونی وارد شود، در اینصورت کاربر باید عبارت زیر را تایپ کند.

```
[120  141  169  201]'
```

۳ -۲ -۴ وارد کردن یک ماتریس

جهت وارد کردن یک ماتریس شامل مقادیر عددی از دستور زیر استفاده می‌کنیم.

```
InputData= input('Enter the three temperatures in degrees C\nat...
levels1 and 2:');
```

که عبارت زیر را در پنجرهٔ فرمان مطلب نشان خواهد داد.

```
Enter the three temperatures in degrees C\nat levels 1 and 2:
[67 35 91;44 51 103]
```

که در آن آرایهٔ *[103 51 44; 91 35 67]* توسط کاربر وارد شده است. متغیر *Input Data* اکنون یک ماتریس (3×2) می باشد.

۳ -۳ فایلهای دادهٔ ورودی/ خروجی

همانطور که در بخش قبل نشان داده شد، یک روش جهت وارد کردن داده‌ها جهت اجرا استفاده از دستور *input* می‌باشد. روش دوم تعریف داده‌ها به صورت یک فایل متنی توسط شیوه‌هایی است که در بخشهای ۲ - ۳ و ۲ - ٤ توضیح داده شد. همچنین عبارتهایی که منجربه ایجاد داده می شوند، می توانند در یک تابع نیز ظاهر شوند، که در فصل پنجم بحث خواهد شد. در حقیقت، کاربر می تواند یک تابع را به گونه ای تعریف کند که تنها شامل داده باشد. بخش ٥- ۲ را برای دیدن یک مثال، مطالعه نمایید.

اما هنوز یک روش دیگر جهت وارد کردن داده‌ها باقی مانده است و آن عبارت است از قرار دادن داده‌ها در یک فایل متنی *ASCII* و سپس استفاده از دستور زیر:

```
load
```

تابع *Load* داده‌ها را به صورت سطر به سطر می‌خواند، که هر سطر با استفاده از کلید *Enter* از سطر بعدی جدا می‌شود و هر داده در یک سطر نیز با استفاده از یک جای خالی و یا علامت کاما از داده‌های دیگر جدا شده است. تعداد ستونهای داده در هر سطر و همچنین تعداد سطرهای داده در هر ستون باید یکسان و برابر باشد. این الزامات مشابه الزاماتی است که در هنگام ساخت ماتریس‌ها باید مورد توجه قرار گیرد. در اینجا کلید *Enter* مشابه نشانهٔ نقطه ویرگول عمل می کند. هنگام ایجاد یک

بردار سطری کاربر، داده‌ها را بدون استفاده از کلید *Enter* وارد می‌کند، در صورتیکه هنگام ساخت یک بردار ستونی، پس از هر دادهٔ عددی یکبار از کلید *Enter* استفاده می‌شود.

اجازه دهید دو شیوهٔ استفاده از دستور *Load* را نشان دهیم. فرض کنیم که داده ها در فایل *Data Section 33.txt* بصورت زیر قرار گرفته باشند.

```
11  12  13
21  22  23
31  32  33
41  42  43
```

یکی از قابلیتهای مفید تابع *Load* این است که اسم فایل بدون پسوند *(".txt")* به عنوان نام متغیری که عناصر ماتریس و یا بردارش، داده‌هایی هستند که در فایل مربوطه آمده‌اند، قلمداد می‌شوند. بنابراین، در فایل متنی، متغیر *Data Section 33* که ماتریسی (3×4) شامل اعداد می‌باشد، استفاده خواهد شد.

این دستور به شکل

```
load  DataSection33.txt
```

یا

```
load  'DataSection33.txt'
```

و یا

```
load('DataSection33.txt')
```

می‌باشد فرض می‌شود که مسیر این فایل قبلاً تعیین شده است. در غیر اینصورت می‌توانید از دستور *cd* استفاده کنید و یا از منوی پائین کشیدنی *File* گزینهٔ *Path* را انتخاب کرده و سپس مسیر را تعیین کنید. این سه شکل، زمانی استفاده می‌شوند که نام فایل در زمان ساخت فایل متنی مشخص است و تغییر نخواهد کرد. بنابراین اگر بخواهیم تا هر عنصر ماتریس را به توان *2* برسانیم، در اینصورت داریم:

```
load  'DataSection33.txt'
y = DataSection33.^2
```

که نتیجهٔ زیر را در بر خواهد داشت:

```
 121   144   169
 441   484   529
 961  1024  1089
1681  1764  1849
```

از طرف دیگر، اگر کاربر بخواهد تا عملیاتی را روی داده ها در فایلهای مختلف، که هر یک دارای نام متفاوتی هستند اجرا کند، در این صورت باید از تکنیکهای دیگری استفاده شود. در این حالت کاربر اسم فایل را که مانند مثال قبل قرار است عناصر موجود در ماتریس به توان 2 رسانده شوند، را وارد خواهد نمود. هنگامیکه فایل متنی اجرا می‌شود، تمامی داده‌های موجود در فایل که نامشان پس از اجرای فایل معین خواهد شد، به توان 2 می‌رسند.

فایل متنی به شکل زیر می باشد:

```
FileName1 = input('Enter  file name containing data (including suffix) :','s' );
load( FileName1);
m = findstr( FileName1,'.' );
data1 = eval( FileName1(1 : m – 1));
y = data1.^2
```

تابع *findstr*، مکان اولین باری که علامت داخل کوتیشن، در اینجا (.)، در رشته ای از کاراکترها وارد می‌شود را شناسایی کرده و موقعیت آنرا به صورت یک عدد باز می‌گرداند. ما از این تابع جهت محدود کردن رشته‌ای از کاراکترها که شامل *FileName1* می‌شود، اما شامل خود علامت داخل کوتیشن و در اینجا (.)، نمی‌شود، استفاده کرده‌ایم. به همین خاطر از عبارت *(1: m−1)FileName1* استفاده شده است. بنابراین پسوند فایل را از نام فایل حذف نموده‌ایم. از آنجا که شکل حذف شدهٔ پسوند *File Name1* برای سایر دستورات موجود در فایل متنی شناخته شده نمی‌باشد، باید آنرا به یک کمیت عددی تبدیل نمود. این کار با استفاده از دستور *eval* صورت می پذیرد که مقدار رشته‌ای را که بعنوان آرگومانش قرار می‌گیرد ارزیابی می‌کند.

اگر کاربر بخواهد تا مقادیر عددی را که از یک فایل متنی پس از اجرا به دست می‌آید را در فایلی ذخیره کند، باید از دستور زیر استفاده کند:

```
save
```

فرض کنید بخواهیم مقدار مربع هر عدد را در فایل *Data Section 33.txt* به عنوان یک متن *ASCII* ذخیره کنیم. در اینصورت از فایل متنی زیر استفاده خواهیم کرد:

```
load 'DataSectio n33.txt'
y = DataSectio n33.^2;
save 'SavedDataSection33.txt' y -ascii
```

که نتیجه آن بصورت زیر خواهد بود:

```
1.2100000e+002   1.4400000e+002   1.6900000e+002
4.4100000e+002   4.8400000e+002   5.2900000e+002
9.6100000e+002   1.0240000e+003   1.0890000e+003
1.6810000e+003   1.7640000e+003   1.8490000e+003
```

هنگامیکه فقط اسم فایل داده می شود و مطلب فایل را در دایرکتوری پیش فرض قرار می‌دهد. جهت قرار دادن فایل در یک دایرکتوری خاص، باید آدرس کامل مسیر داده شود. برای مثال:

```
load 'DataSection33.txt'
y = DataSection33.^2;
save 'c:\Matlab mfiles\Matlab results\SavedDataSection33.txt' y -ascii
```

توجه کنید ما شکلی از دستور که از علامت کوتیشن در دو طرف آدرس مسیر و اسم فایل بهره می‌برد، را استفاده نموده‌ایم. این مسئله بخاطر ظاهر شدن علامت ویرگول و فضاهای خالی موجود در آدرس مسیر ضروری می‌باشد. همچنین اگر بخواهیم تا مقادیر دیگری را نیز در این فایل ذخیره کنیم، باید نام متغیرهای متناظرشان را مطابق زیر در دستور *save* وارد کنیم. فرض کنید که فایل متنی بالا ریشهٔ دوم مقادیر موجود در *Data Section 33.txt* را نیز محاسبه کند. در اینصورت،

```
load 'DataSection33.txt'
y = DataSection33.^2;
z = sqrt(DataSection33);
save 'c:\Matlab mfiles\Matlab results\SavedDataSection33.txt' y z -ascii
```

داده های موجود در فایل، اکنون شامل یک ماتریس دارای هشت سطر که هر سطر دارای سه ستون است، خواهد بود.

داده‌های موجود در چهار سطر اول مربوط به y و داده های چهار سطر آخر مربوط به z می‌باشند.

روند ذخیره کردن شکلها در بخش ٦ -١ آمده است.

تمرین ها

۳ - ۱ اعداد فیبوناتچی می توانند با استفاده از رابطهٔ زیر ایجاد شوند:

$$F_n = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right] \qquad n = 0,1,2,...$$

شانزده مقدار اول این اعداد را ایجاد کرده و سپس با استفاده از هر دو دستور *fprintf* و *disp* آنها را مطابق ساختار زیر در پنجرهٔ فرمان مطلب نمایش دهید:

$$F0 = 0$$
$$F1 = 1$$
$$F2 = 1$$
$$F3 = 2$$
$$\vdots$$
$$F15 = 610$$

فصل چهارم

کنترل روند برنامه نویسی

در این فصل ابزارهای گوناگونی که برای کنترل ترتیب اجرای عبارت‌های یک برنامه استفاده می‌شوند، به همراه نمونه ای از مجموعهٔ عملگرهای انتسابی و منطقی که عملیات کنترل را تکمیل می‌کنند، ارائه شده است.

۴- ۱ مقدمه

کنترل ترتیبی که عبارت های یک برنامه مورد ارزیابی قرار می‌گیرند، به وسیلهٔ چهار ساختار کنترلی روند برنامه صورت می پذیرد: *while، if ، for* و *switch* . هرگاه یکی از این دستورها در برنامه ظاهر شود، بدنبال آن در قسمت دیگری از برنامه دستور *end* ظاهر می‌شود. تمام عباراتی که در بین دستور ساختار کنترلی و دستور *end* ظاهر می شوند، تا وقتی که تمام شرایط ساختار تامین شود، اجرا می گردند.

هر یک از این دستورهای ساختار کنترلی می‌توانند بارها، بر حسب ضرورت در بین یکدیگر و یا در میان خودشان ظاهر شوند. در این شرایط به آن ها ساختارهای تو در تو می‌گویند.

ساختارهای کنترلی غالباً به عملگرهای منطقی و انتسابی وابسته هستند تا برقراری شرط را بررسی کنند. وقتی این شرط برقرار بود، ساختار، اجرا را به قسمت مشخصی از آن هدایت می کند تا یک یا چند دستور را اجرا نماید. چند عملگر منطقی و انتسابی مطلب در جدول ٤-١ داده شده است.

هنگام استفاده از ساختارهای کنترلی توصیه می‌شود که دستورات پس از هر ساختار کنترلی تا دستور *end* (اما نه خود دستور *end*)، کمی با فاصله از سمت چپ نوشته شود (حاشیه گذاری شود). این روش میزان خوانایی برنامه یا تابع را افزایش می‌دهد. وقتی که ساختارها به صورت متداخل هستند، کل ساختار متداخل حاشیه گذاری می‌شود. ضمن اینکه حاشیه گذاری در هر حلقهٔ متداخل نیز صورت می پذیرد.

هنگام استفاده از محیط ویرایشگر یا اشکال زدای مطلب، این عمل به طور خودکار انجام می‌شود.

می‌توان عملگرهای منطقی موجود در جدول ٤-١ را برای ایجاد یک تابع منطقی که در آن اگر عملگر منطقی درست باشد خروجی آن *1* و اگر غلط باشد خروجی آن *0* است، استفاده کرد. فرض کنید می‌خواهیم تابع *g(x)* را بصورت زیر تعریف کنیم.

$$g(x) = f(x) \qquad a \le x \prec b$$
$$= 0 \qquad otherwise$$

عملگر منطقی به صورت زیر ایجاد می شود:

$$y = (a <= x \,\&\, x < b);$$

که در آن a و b قبل از دستور فوق توسط مقادیر عددی مقدار دهی شده اند و،

$$(a <= x \,\&\, x < b)$$

جدول ۴-۱) چند عملگر منطقی و انتسابی

عملگرهای انتسابی		
مساوی	=	= =
مخالف	≠	~ =
کمتر	<	<
بیشتر	>	>
کمتر مساوی	≤	<=
بیشتر مساوی	≥	> =

عملگرهای منطقی		
و	AND	&
یا	OR	|
نه	NOT	~

عملگر منطقی ایست که هنگامی که $x \geq a$ و $x < b$ باشد مقدار آن 1 (درست) و برای تمام مقادیر دیگر x مقدار آن 0 (غلط) است. بنابراین اگر a=-1 ، b=2 ، $f(x) = e^{x/2}$ و $x = [-4 \ -1 \ 1 \ 4]$ باشد در این‌صورت برنامه ای که از این عملگر منطقی استفاده می‌کند، به صورت زیر خواهد بود:

```
a = -1; b = 2;
x = [-4  -1  1  4];
gofx = exp(x / 2).*(a <= x & x < b)
```

که پس از اجرا مقدار $gofx$ برابر $gofx = [0 \ 0 \ 0.6065 \ 1.6487 \ 0]$ خواهد بود.

۴-۲ کنترل جریان برنامه

۴-۲-۱ حلقهٔ *while*

حلقهٔ *while* یک یا چند دستور را به دفعات نامعین تکرار می‌کند و خروج از حلقه تنها زمانی صورت می‌گیرد که شرط تعیین شده برقرار نباشد. شکل کلی آن به صورت زیر می‌باشد:

```
while condition
statements
end
```

که در آن عبارت بیان کنندهٔ *condition* می‌تواند دارای یک یا چند متغیر باشد که توسط *statements* مقدار می‌گیرند.

در اینجا دو مثال از حلقهٔ *while* ارائه می کنیم. مثال اول برای تضمین اینکه اطلاعات داده شده توسط کاربر در محدودهٔ تعیین شده می‌باشند، استفاده می‌شود و مثال دوم هنگامی که محک همگرایی خاصی باید برقرار باشد، استفاده می‌شود.

مثال ۴- ۱ وارد کردن صحیح اطلاعات

عبارات ذیل که قسمتی از یک برنامهٔ مطلب می‌باشد از کاربر می‌خواهد عددی را از *1* تا *8* وارد کند عمل درخواست از کاربر تا زمانی ادامه می‌یابد که ورودی در بازهٔ تعیین شده قرار داشته باشد. تابع *input* پیغام داخل کوتیشن را در پنجرهٔ نمایش مطلب نمایش می‌دهد و منتظر می‌ماند تا کاربر مقداری را وارد کند، که این برنامه آن مقدار را برابر *nfnum* قرار می دهد. علامت "|" نشان دهندهٔ عملگر منطقی *OR* می‌باشد.

```
nfnum = 0;
while (nfnum < 1)|(nfnum > 8)
nfnum = input('Enter a number from 1 to 8 :');
end
```

در ابتدا مقدار *nfnum* برابر مقداری دلخواه می‌باشد (در این مورد صفر) که باعث می‌شود تا دستور *while* شرط *(nfnum<1)|(nfnum>8)* را امتحان کند تا وارد ساختار *while* شود. بعد از رسیدن به آخرین عبارت قبل از دستور *end* ، برنامه به عبارت شرط *while* باز می گردد تا بررسی کند که آیا شرط برقرار است یا نه، اگر شرط برقرار نباشد، خط بعدی ساختار اجرا می شود. در غیر این صورت به سراغ دستور بعد از عبارت *end* خواهد رفت.

توجه کنید که ما عمل جایگذاری مقدار اولیهٔ ساختار - در اینجا *nfnum* - را درست قبل از ورود به ساختار به کار برده‌ایم، که در مورد سایر موارد نیز اینکار توصیه می‌شود.

مثال ۴ -۲ همگرایی سری ها

ابتدا تعداد عباراتی را که برای سری زیر:

$$S_N = \sum_{n=1}^{N} \frac{1}{n^2}$$

جهت همگرایی آن در حدود 0.01% مقدار دقیقیش که $S_\infty = \pi^2/6$ می باشد، لازم است را تعیین کرده و نمایش می‌دهیم. برنامه به صورت زیر خواهد بود.

```
series=1;k=pi^2/6;
while abs ((series-exact)/exact)>=1e -4
series=series+1/k^2;
k=k+1;
end
disp(['# terms='num2str(k-1)])
```

مقدار سری‌ها در ابتدا برابر مقداری (در این مورد *1*) قرار داده می‌شود که باعث می‌شود دستور *while* شرط *abs((series-exact)/exact)* را برای ورود به ساختار *while* امتحان کند. وقتی که به آخرین دستور قبل از *end* رسید، برنامه به عبارت شرطی *while* باز می‌گردد، تا اینکه بررسی کند که آیا این شرط برقرار است یا نه. اگر برقرار نباشد، خط بعدی ساختار را اجرا می‌کند. در غیر این صورت دستور بعد از دستور *end* را پردازش می‌کند که در این حالت حاصل را در پنجرهٔ فرمان مطلب نشان می‌دهد.

در اینجا از قدر مطلق عبارت شرط *(series-exact)/exact* استفاده کرده‌ایم تا از هرگونه موردی که ممکن است وقتی تفاوت *series -exact* منفی شود پیش بیاید، جلوگیری کنیم. چون ممکن است مقدار این عبارت کمتر از 10^{-4} می‌باشد (مثلاً این اختلاف منفی باشد.)، ولی اندازهٔ آن کمتر از 10^{-4} نباشد. این کار لزوم پیش بینی قبلی در این باره که آیا اندازهٔ *series* به مقدار واقعی از بالا یا از پایین نزدیک می‌شود جلوگیری خواهد کرد. باید کاربر هنگام ایجاد محک مناسب برای خروج از حلقهٔ *while* خیلی دقت کند، برای اینکه اگر به صورت نادرست و یا ضعیفی انتخاب شود، ممکن است حلقه به صورت نامتناهی اجرا شود، یعنی تا وقتی که کلیدهای *control* و *c* با هم فشار داده شوند.

۴-۲-۲ دستور *if*

شکل کلی دستور *if* به صورت زیر است.

```
if  condition #1
  expressions#1
elseif  condition #2
  expressions#2
else
  expressions #3
end
```

وقتی که *condition#1* برقرار باشد , *expressions#1* ، سپس دستورات بعد از بیرونی ترین *end* اجرا می‌شوند. وقتی که *condition#1* برقرار نباشد آنگاه *condition#2* مورد بررسی قرار می‌گیرد. اگر برقرار باشد، آنگاه *expresstions#2* اجرا می‌گردند و دستورات بعد از بیرونی‌ترین *end* اجرا می‌شوند. اگر هیچ کدام از شرایط *1* یا *2* برقرار نباشند، آنگاه *expressions#3* اجرا می‌شوند و دستورات بعد از بیرونی ترین *end* دنبال می‌شوند.

استفاده از دستورات *else* و *else if* اختیاری می‌باشد. همچنین می توان از بیش از یک دستور *else if* استفاده نمود. برنامهٔ زیر کاربرد دستور *if* را نشان می‌دهد. *j* ، *x* و *nnum* دارای مقادیر عددی می باشند که به آنها نسبت داده شده است و یا توسط یک عملیات محاسبه ای در برنامه تعیین شده‌اند.

if j == 1

z = sin(x); ← تنها در صورتی اجرا می شود که *j=1* باشد

if nnum<=4 ← با این دستور *if* فقط وقتی مواجه می‌شوید که *j=1* باشد

```
nr = 1;
nc = 1;
```
← این دستورات تنها در صورتی اجـرا مـی شـود کـه $j = 1$ و $nnum \le 4$ باشد

```
else
nr = 1;
nc = 2;
```
← این دستورات تنها در صورتی اجرا می شود که $j=1$و $nnum>4$ باشد

```
end
else
nr = 2;
nc = 1;
```
← این دستورات تنها در صورتی اجرا می شود که $j=1$ باشد

```
end
```

همان طور که ملاحظه می‌کنید ما در اینجا یک دستور *if* تو در تو داریم. بنابراین به دستور*end* دیگری نیز نیاز داریم. همچنین تعیین کردن چگونگی حاشیه گذاری دستورات در داخل حلقه های تو در توی مختلف باعث خوانایی بیشتر برنامه می‌شود.

مثال ۴-۳ فاکتورهای مقاومت خستگی

به روابط زیر که ضرایب تصحیح را جهت تخمین مقاومت خستگی فلزات ارائه می‌دهند , توجه کنید. یک قسمت از متن برنامه که برای اندازه گیری این فاکتور ها می‌تواند مورد استفاده قرار گیرد در زیر داده شده است. مقادیر *lode*، *d* و *temp* قبلاً توسط مقادیر عددی نسبت دهی و یا در برنامه محاسبه شده‌اند. کمیت *lode* از نوع رشته می‌باشد.

ضریب	بازه	تصحیح
بارگذاری	خمشی	$C_{Load} = 1$
	محوری	$C_{Load} = 0.70$
اندازه	$d \le 8$ mm	$C_{Load} = 1$
	$8 \le d \le 250$ mm	$C_{Load} = 1$
دما	$T\lambda 450^{\circ}C$	$C_{Load} = 1$
$C_{temp} = 1 - 0.0032\ (T - 840\ )450\ \lambda T^{\circ}C$		

```
If Load == 'bending'
  cload=1;
else
  cload=0.7;
end
if  d<=8
  csize=1;
else
  csize=1.189*d^(-0.097);
end
if  temp<=450
  ctemp=1;
else
  ctemp=1-0.0032*(T-840);
end
```

همچنین می توانیم این برنامه را به صورت زیر بنویسیم:

```
if  lode=='bending' , cload=1;, else, cload=0.7;, end
if  d<=8,  csize=1;, else, csize=1.189*d^(-0.097);, end
if   temp<=450,  ctemp=1;,  ctemp=1;,  else,   ctemp=1-0.0032*(T-840);
end
```

همان طور که در قسمت ۱- ۸ نشان دادیم، می توان برای جداکردن دستورات در یک خط از کاما (,) یا نقطه ویرگول (;) استفاده نمود.

۴ - ۲ -۳ حلقهٔ *for*

حلقهٔ *for* یک سری از دستورات را به تعداد دفعات مشخص تکرار می‌کند. شکل کلی آن به صورت زیر است:

```
for variable=expression
   statements
end
```

که یک یا چند تا از دستورات می‌توانند تابعی از متغیرها باشند.

مثال ۴-۴ بهرهٔ کلی وام دهی

می‌خواهیم بهرهٔ کلی وام دهی را هنگامیکه مقدار وام L و مدت زمان باز پرداخت آن m ماه و مقدار بهرهٔ سالیانهٔ آن I_a باشد، را حساب کنیم. اقساط ماهیانه به صورت زیر خواهد بود:

$$p_{mon} = \frac{iL}{1-(1+i)^{-m}}$$

که در آن:

$$i = I_a / 1200$$

میزان بهرهٔ ماهیانه است که به صورت یک عدد اعشاری نشان داده می‌شود. هر ماه، هنگامیکه وام به طور کامل پرداخت شود، قسمتی از قسط برای پرداخت بهره استفاده می‌شود و باقیماندهٔ آن برای مقدار وام پرداخت نشده استفاده می شود. مقدار پرداخت نشدهٔ وام بعد از پرداخت هر قسط باقیماندهٔ حساب نامیده می‌شود. این روابط را در ریاضیات به شکل زیر نشان می‌دهیم. اگر $b_0 = L$ باشد، در این صورت داریم:

$$i_n = ib_{n-1}$$

$$P_n = p_{mon} - i_n \qquad n = 1,2,3,...,m$$

$$b_n = b_{n-1} - P_n$$

که در آن i_n قسمتی از p_{mon} است که بهرهٔ قسط را نشان می‌دهد و p_n قسمتی از قسط که جهت کاهش باقیماندهٔ b_n ، که مقدار لازم برای پرداخت کامل وام است، می‌باشد.

بهرهٔ مطلق که در پایان مدت زمان وام پرداخت می‌شود به شکل زیر خواهد بود:

$$i_T = \sum_{j=1}^{m} i_j$$

برنامهٔ محاسبهٔ i_T به شکل زیر می‌باشد:

ورودی ←

```
loan = input('Enter loan amount :');
durat = input('Enter term of loan months :');
int = input('Enter annual interest rate :')/1200;
```

```
int s = zeros( durat,1);
prins = int s;
bals = int s;
pmon = (loan * int) /(1 – (1 + int)^(–durat));
bals(1) = loan;
```

← مقداردهی اولیه

```
for  m = 2 : durat + 1
  int s( m) = int* bals( m – 1);
  prins( m) = pmon – int s( m);
  bals( m) = bals( m – 1) – prins( m);
end
```

← محاسبات

```
fpr int f (1,'Total  int erest = $%8.2 f \ n' ,sum(int s))
```

← خروجی

همان طور که در بخش ۱ – ۲ گفته شد، این فایل متنی یک ساختار معمول برنامه را دنبال می‌کند: ورودی، مقدار دهی اولیه ، محاسبات و خروجی. که در اینجا نتایج در پنجرهٔ فرمان مطلب نشان داده می‌شود. اجرای این برنامه خروجی زیر را نتیجه می‌دهد:

```
Enter  loan  amount : 100000

Enter term of  loan  months : 360

Enter  annual  int erest  rate : 8

Total  int erest = $164155.25
```

سه خط اول، پاسخ کاربر به پرسشهای متوالی نمایش داده شده در پنجرهٔ فرمان را نشان می‌دهد. که در آن کاربر پس از هر پرسش یک عدد را وارد می‌کند. و خط آخر پاسخ برنامه می‌باشد. نرم افزار مطلب توصیه می‌کند قبل از ورود به حلقهٔ *for* یا حلقه های *for* تو در تو، که در آنها بردارها و ماتریسها با تعیین اندیسهایشان ایجاد می‌شوند کاربر باید ماتریس را توسط تابع *fzero* ایجاد کند. این عمل زمان اجرای این قسمت از برنامه را بسیار کاهش می‌دهد، زیرا در غیر این صورت ممکن است مطلب در حین اجرای حلقهٔ *for* حافظهٔ خود را به صورت پویا اختصاص دهد.

مثال ۴- ۵ برنامهٔ معادل دستور *find*

فرض کنیم بردار g متشکل از اعداد مثبت و منفی و طول اختیاری داده شده است. هدف ایجاد برنامه‌ای است که مشابه دستور زیر عمل کند.

```
indx = find(g>a)
```

که در آن a توسط کار بر تعیین می‌شود. برنامه را با مقادیر $a=4$ و بردار $g=[4\ 4\ 7\ 10\ -6\ 42\ 1\ 0]$ چک خواهیم کرد:

```
g = [4  4  7  10  -6  42  1  0]; a = 4;
k = 0;
idxx = [];
for  n = 1 : length( g )
  if  g( n ) > a
  k = k + 1;
  indx( k ) = n;
 end
end
disp(['Element  locations  for  g( n ) >' num2str( a )':' num2str(indx )])
```

که پس از اجرای برنامه، خروجی زیر در پنجرهٔ فرمان مطلب نشان داده می‌شود.

```
Element  locations  for  g(n) > 4 : 3  4  6
```

یک راه حل دیگر برای بدست آوردن *indx* به صورت زیر است:

```
g = [4  4  7  10  -6  42  1  0]; a = 4;
k = 0;
idxx = [];
```

```
for  n = 1 : length( g )
  if  g( n ) > a
  indx = [ indx  n ];
 end
end
disp([' Element  locations  for  g( n ) >' num2str( a )' :' num2str( indx )])
```

مثال۴-۶ برنامهٔ معادل دستور *cumsum*

می‌خواهیم برای بردار داده شده که دارای طول دلخواه می‌باشد برنامه‌ای بنویسیم که نتایج آن مشابه نتایج تابع زیر باشد:

```
Csum=cumsum(c)
```

برنامه را با بردار $c=[4\ 4\ 7\ 10\ -6\ 42\ 1\ 0]$ چک خواهیم کرد برنامه به صورت زیر خواهد بود:

```
c=[4 4 7 10 -6 42 1 0];
Csum(1)=c(1);
for k=2:length(c)
   Csum(k)=Csum(k-1)+c(k);
end
disp([Cumsum of c=num2str(Csum)])
```

که پس از اجرا خروجی زیر را در پنجرهٔ فرمان مطلب نشان خواهد داد:

```
Cumsum of c =4 8 15 19 61 62 62
```

یک راه دیگر بدست آوردن *Csum* در زیر نشان داده شده است:

```
c=[4 4 7 10 -6 42 1 0];
```

```
Csum(1)=c(1);
for k =2:length(c)
  Csum=[Csum Csum(k-1)+c(k)];
end
disp(['Cumsum of c=' num2str(Csum)])
```

مثال ۴-۷ برنامهٔ معادل دستور *diag*

برای ماتریس b از مرتبهٔ $(n\times n)$ یک برنامه شامل دو قسمت که نتایج مشابهی با نتایج دستور زیر؛

```
v=diag(b)
```

در قسمت اول و دستور زیر؛

```
d =diag(v)
```

در قسمت دوم ایجاد می‌کند خواهیم نوشت. در اینجا نتایج را با استفاده از دستور *b=magic(4)* بررسی کرده و فرض می‌کنیم که مجاز به استفاده از نماد ":" و دستور *zero* نمی‌باشیم. برنامه به صورت زیر خواهد بود:

```
b=magic(4);[r c]=size(b);
fork=1:r
  v(k)=b(k,k);
end
disp(['Diagonal elements of b='])
disp(num2str(v))
for n=1:r
 for m=1:r
  if n==m
   d(n,m)=v(n);
  else
   d(n,m)=0.0;
   end
  end
```

```
end
disp(['Diagonal matrix d='])
disp(num2str(d))
```

که پس از اجرا، خروجی زیر را در پنجرهٔ فرمان مطلب نشان می‌دهد:

```
Diagonal elements of b =
16 11 6 1
Diagonal matrix d =
16  0  0  0
0   11 0  0
0   0   6  0
0   0   0  1
```

اگر مجاز به استفاده از نماد ":" می‌بودیم، برنامه به صورت زیر کوتاه تر می‌شد:

```
b=magic(4);[r c]=size(b);
for k=1:r
  v(k)=b(k,k);
end
disp(['Diagonal elements of b='])
disp(num2str(v))
d(1:r,1:c)=0.0;
for n=1:r
  d(n,n)=v(n);
end
disp(['Diagonal matrix d='])
disp(num2str(d))
```

۴-۲-۴ خروج فوری از حلقهٔ *for* و یا *while*

تابع *break* برای پایان دهی فوری به حلقه های *for* و یا *while* استفاده می‌شود. اگر تابع *break* در بین حلقه های *for* یا *while* تو در تو قرار گرفته باشد، آنگاه اجرا در آن حلقهٔ *for* یا *while* داخلی متوقف خواهد شد. اکنون به قسمتی از یک برنامه توجه کنید:

```
for j=1:14
  ⋮
```

```
b = 1
while   b < 25
  ⋮
  if  n < 0
    break
  end
   ⋮
end
⋮

end
```

وقتی که $n<0$ باشد، از حلقه *if* خارج می شود و برنامه از دستور بعد از این دستور(*end*) دنبال خواهد شد.

برای پایان دادن به برنامه (یا تابع: به فصل ۵ مراجعه کنید)، وقتی که یک شرط معین برقرار نباشد، می‌توان از تابع *error* استفاده کرد. تابع *error* معمولاً برای این استفاده می‌شود تا حصول نتایج با معنی را در استفاده از یک برنامه شامل مقادیر عددی، تضمین کند.

وقتی یک برنامه با تابع *error* مواجه می شود، پیامی را در پنجرهٔ فرمان مطلب نشان خواهد داد که در بردارندهٔ آن است. بعد از نشان دادن این پیام، اجرای برنامه یا تابع متوقف می‌شود و کنترل به خط فرمان در پنجرهٔ فرمان مطلب باز می‌گردد. از تابع *error* می‌توان در هر قسمتی از برنامه یا تابع استفاده کرد و محدود به ساختارهای *for* و *while* نیست.

۴-۲- ۵ دستور *switch*

شکل کلی دستور *switch* به صورت زیر می‌باشد.

```
switch_expression   switch
    case    case_expression#1
          statement#1
    case    case_expression#2
          statement#2
              ⋮
    case    case_expression#n
          statement#n
```

```
    otherwise
        statement#n
end
```

اولین *case_expression#j* که در آن $j=1,2,...,n$ می‌باشد و زمانی که *case_expression#j=switch_expression* برقرار باشد با آن مواجه می‌شویم، باعث خواهد شد که *statements#j* اجرا شود. فقط یک *case* اجرا می‌شود. بعد از اجرای *statements#j* ، عملیات اجرا به عبارت بعد از دستور *end* انتقال خواهد یافت. اگر هیچ یک از عبارات *case_expression#j* برقرار نباشد آنگاه *statements#n+1* اجرا خواهد شد. استفاده از دستور *otherwise* اختیاری است. اگر دستور *otherwise* حذف شود و عبارت *case_expression#j* به ازای هر *j* ای با *switch_expression* برابر نباشد، آنگاه دستور بعد از دستور *end* اجرا خواهد شد. دستور *switch* ، جایگزینی برای ساختارهای *if - else - elseif - end* می باشد. ساختار *switch* در زیر همان طور که در بالا گفته شد عمل می کند. مقدار *k* تعیین شده است و یا قبل از اینکه با این ساختار روبرو شویم محاسبه شده است.

```
a = 3;
switch  k
case  1
  disp(' case  1' )          ← این دستور فقط وقتی k=1 باشد اجرا می شود.
case { 2,3 }
  disp(' case  2   or  3' )  ← این دستور فقط وقتی k=2 باشد اجرا می شود.
case  a^2
  disp(' case  9' )          ← این دستور فقط وقتی k=9 باشد اجرا می شود.
otherwise
  disp(' otherwise' )        ← این دستور فقط وقتی k≠1,2,3,4 باشد اجرا می شود.
end
```

مثال ۴-۸ رسم چهار نمای یک سطح

تصور کنید که کاربر چهار نما از رویهٔ *z(x,y)* را در چهار مربع جدا از هم در یک پنجره می‌کشد. فایل متنی در زیر داده شده است. تابع *subplot* پنجره را به یک شبکه تقسیم می‌کند که در این حالت این شبکه 2×2 است.

شاخص k، مشخص کنندهٔ مکان عدد در شبکه است. مثلاً $k=1$ موقعیت بالا سمت چپ، $k=2$ موقعیت بالا سمت راست، $k=3$ موقعیت پایین سمت چپ، $k=4$ موقعیت پایین سمت راست می‌باشد. (فصل ٦ و ٧ را برای بدست آوردن جزئیات بیشتر در این مورد و توابع گرافیکی دیگر ملاحظه کنید). تابع *surf* یک پرسپکتیو سه بعدی از آرایهٔ مقادیر z به عنوان تابعی از x و y ترسیم می‌کند. همچنین تابع *view* زوایای سطح را تعیین می‌کند. مقادیر x، y و z قبلاً تعیین شده‌اند.

```
for k=1:4
  subplot(2,2,k)
  surf (x,y,z)
switch k
    case 1
      view(-37.5,30)
    case 2
      view(-70,30)
    case 3
      view(-37.5,50)
    case 4
      view(-70,50)
  end
end
```

۴-۳ دو کاربرد ساختارهای کنترلی برنامه

اکنون به بررسی دو برنامه که از ترکیبات ساختارهای کنترلی برنامه نویسی استفاده کرده‌اند، می‌پردازیم.

۴-۳-۱ تشکیل یک جدول 2^K فاکتوریل

معمولاً استقراها روی سیستم‌های چند متغیره‌ای که دارای کاربردهای وسیعی می‌باشند اعمال می‌شوند. به همین دلیل این فرآیندها نیاز دارند که از تمامی ترکیبهای ممکن هر متغیر برای هر بار اجرای کامل استقرا استفاده کنند. چنین استقرایی را استقرای فاکتوریل می‌نامند.

اگر استقرا شامل k فاکتور و هر فاکتور دارای دو حالت باشد، در این صورت استقرایی وابسته به 2^k فاکتور خواهیم داشت. در اینجا قرار داد می‌کنیم که حالت اول فاکتورها را با علامت "+" و حالت دوم فاکتورها را با علامت "-" نشان دهیم. فاکتورها برای $k=2$ با A و B و برای $k=3$ با A، B و C و برای $k=4$ با A، B، C و D مشخص شده‌اند. به جزئیات جدول ٤ - ۲ توجه کنید.

علامت های + و- در هر ستون مبین $+1$ و -1 می‌باشند. ستونهایی که تحت عناوین A، B، C و D نامگذاری شده‌اند فاکتورهای اولیه نامیده می‌شوند. مابقی ستون‌ها، نشان دهندهٔ عبارت‌های ترکیبی می‌باشند که توسط ضرب علامتهای متناظر در ستون‌هایی از فاکتورهای اولیه در ستونهایی که قبلاً محاسبه شده‌اند بدست می‌آیند. بنابراین علامت ستون‌ها که ترکیب ABC را به وجود می‌آورند به وسیلهٔ ضرب علامت‌های ستون های تحت عناوین A، B و C و یا AB و C یا A و BC بدست آمده‌اند. مثلاً در ردیف 7 $(m=7)$ $A=-1$، $B=+1$، $C=+1$ می‌باشد. بنابراین علامت در ردیف هفدهم از ستون ABC , -1 است $[=(-1)(+1)(+1)]$. به علاوه برای 2^2 استقرا سه ستون اول و ردیف های $m=1,2,...4$ استفاده می‌شوند. برای 2^3 استقرا، 7 ستون اول و ردیف‌های $m=1,2,...8$ استفاده می‌شوند و برای 2^4 استقرا همهٔ 15 ستون و ردیف‌های $m=1,2,...16$ مورد استفاده قرار می‌گیرند. همچنین به خاطر بسپارید که $2^{(k-1)}$ ردیف اول $(k>2)$ برای تشکیل ردیف‌های زیر خطوط افقی در جدول ٤-۲ همان طور که تعداد فاکتورها افزایش می‌یابد، تکرار می‌شوند.

نتایج ارائه شده در جدول ٤-۲ جهت تحلیل اطلاعات بدست آمده از یک استقرای وابسته به 2^k فاکتور ضروری می‌باشد. بنابراین برنامهٔ زیر را جهت ایجاد مقادیر ± 1 که در این جدول برای $k = 2$، $k = 3$ و $k = 4$ آمده است، ایجاد می‌کنیم:

```
k=input('Enter number of factors(=2,3,or4)');
s=ones(2^k,2^k-1);
for r=1:2:2^k
  s(r,1)=-1;
end
for c=2:k
e=2^(c-1);
for r=1:e
  s(r,e)=-1;
end
eorr=e+1:2^(k)
  s(r,2^(c-2))=s(r-e,2^(c-2));
```

```
  end
end
for m=2:k
  e=2^(m-1);
  for j=1:e-1
    s(:,e+j)=s(:,j).*s(:,e);
   end
  end
disp(s)
```

جدول ۴ - ۲ فاکتور نشانه 2^k

فاکتورها و روابط بین آنها

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
$k=2$			$k=3$				$k=4$								
A	B	AB	C	AC	BC	ABC	D	AD	BD	ABD	CD	ACD	BCD	ABCD	m
−	−	+	−	+	+	−	−	+	+	−	+	−	−	+	1
+	−	−	−	−	+	+	−	−	+	+	+	+	−	−	2
−	+	−	−	+	−	+	−	+	−	+	+	−	+	−	3
+	+	+	−	−	−	−	−	−	−	−	+	+	+	+	$4=2^2$
−	−	+	+	−	−	+	−	+	+	−	−	+	+	−	5
+	−	−	+	+	−	−	−	−	+	+	−	−	+	+	6
−	+	−	+	−	+	−	−	+	−	+	−	+	−	+	7
+	+	+	+	+	+	+	−	−	−	−	−	−	−	−	$8=2^3$
−	−	+	−	+	+	−	+	+	−	+	−	+	+	−	9
+	−	−	−	−	+	+	+	−	−	−	−	−	+	+	10
−	+	−	−	+	−	+	+	+	+	−	−	+	−	+	11
+	+	+	−	−	−	−	+	−	+	+	−	−	−	−	12
−	−	+	+	−	−	+	+	+	−	+	+	−	−	+	13
+	−	−	+	+	−	−	+	−	−	−	+	+	−	−	14
−	+	−	+	−	+	−	+	+	+	−	+	−	+	−	15
+	+	+	+	+	+	+	+	−	+	+	+	+	+	+	$16=2^4$

اجرای این برنامه خروجی زیر را نتیجه می‌دهد:

Enter number of factors (= 2,3, or 4) 3

$$\begin{matrix} -1 & -1 & 1 & -1 & 1 & 1 & -1 \\ 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{matrix}$$

آخرین عدد خط اول (3) ورودی داده شده توسط کاربر است.

توجه کنید که اگر $z = s/2^{k/2}$ آنگاه ماتریس z، یک ماتریس اورتوگونال است که دارای خاصیت زیر می باشد: (تمرین ۲-۱۱ را بیاد بیاورید)

$$I = z'z$$

که I در این حالت یک ماتریس یکانی (7×7) می‌باشد.

۴ - ۳ - ۲ یافتن ریشه های مختلف یک تابع به روش نصف کردن بازه ها[۳۲]

مطلب دارای تابعی به نام *fzero* که جهت تعیین ریشه های تابع $f(x)$ استفاده می‌شود، می‌باشد. (بخش ۵- ۵- ۱ را ملاحظه کنید). اما با استفاده از این تابع هر بار فقط یک ریشه پیدا می‌شود و برای یافتن ریشه های دیگر، کاربر باید *fzero* را دوباره با حدس اولیهٔ دیگری که نزدیک ریشهٔ بعدی است بکار برد.

فرض کنید که می‌خواهیم بطور همزمان یک سری مقادیر مثبت x، $(x_1, x_2, ...)$ که منجر به برقراری رابطهٔ $f(x)=0$ می‌شوند را بیابیم. در اینجا فرض شده است که $f(x)$ دارای چندین ریشه است و بنابراین علامتش با افزایش x تغییر می‌کند.

---

۱ - برای مثال به مرجع زیر مراجعه کنید.

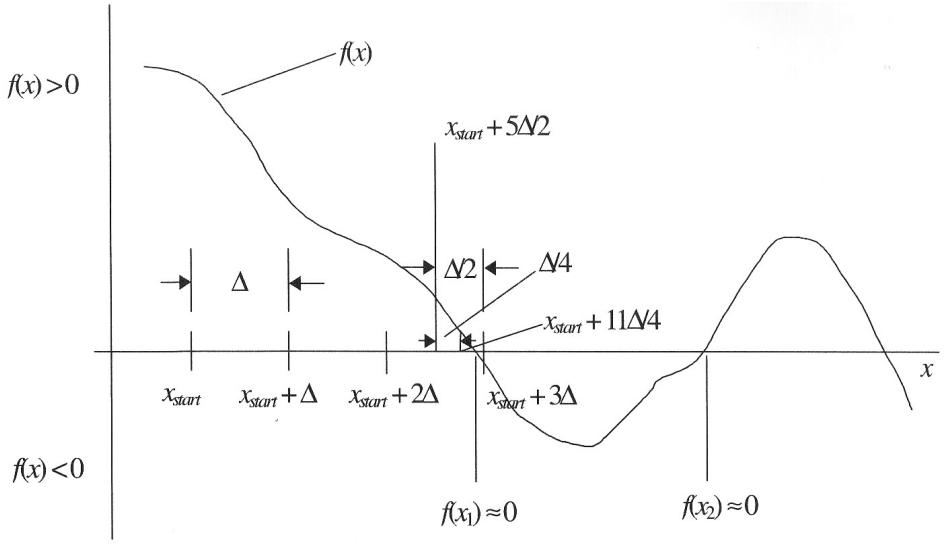
S. C. Chapra and R. P. Canale, Numerical Methods for Engineers, 2nd ed., McGraw-Hill, NewYork, 1988, p. 128ff.

یک تکنیک برای یافتن ریشه های یک تابع نصف کردن بازه ها می‌باشد[۳۳]. با توجه به شکل (٤ - ١) این روش به صورت زیر عمل می‌کند:

ابتدا به متغیر مستقل x مقدار اولیهٔ $x = x_{start}$ نسبت داده می‌شود، سپس علامت $f(x_{start})$ تعیین می‌گردد. آنگاه متغیر x به اندازهٔ Δ افزایش می‌یابد و علامت $f(x_{start} + \Delta)$ تعیین می‌شود. علامت این دو مقدار مقایسه می‌شود و اگر علامت ها یکسان باشند (همانطور که در شکل (٤- ١) نشان داده شده است) آنگاه x دوباره به اندازهٔ Δ افزایش می‌یابد و علامت $f(x_{start} + 2\Delta)$ اندازه گیری می‌شود و آنرا با $f(x_{start})$ مقایسه می‌کنیم. اگر علامت ها مخالف باشد آنگاه مقدار x با نصف کردن بازه به اندازهٔ $\Delta/2$ کاهش می‌یابد. در شکل (٤-١) می‌بینیم که تغییر علامت در $x = x_{start} + 3\Delta$ اتفاق می‌افتد. بنابراین بعد از اینکه تغییر علامت مشخص شد مقدار بعدی x، $x = x_{start} + 5\Delta/2$ خواهد بود. سپس علامت $f(x_{start} + 5\Delta/2)$ با علامت $f(x_{start})$ مقایسه می‌شود، اگر یکسان باشد، نصف این مقدار به x اضافه می‌شود، در غیر اینصورت این مقدار از x کم خواهد شد. در این مثال علامت $f(x_{start})$ با علامت $f(x_{start} + 5\Delta/2)$ یکسان است، پس نقطهٔ بعدی که $f(x)$ در آن اندازه گیری می‌شود، $x = x_{start} + 11\Delta/4$ خواهد بود. این عمل تا وقتی تکرار می‌شود که تغییر افزایشی x تقسیم بر مقدار فعلی x کمتر از تلرانس (t_0) باشد. یعنی: $\Delta_{current} / x_{current} < t_0$. زمانی که این رابطه برقرار شده باشد، آنگاه $x = x_{current}$ ریشه است. این عمل تا وقتی که عدد مطلوب x_j تعیین شود ادامه دارد. بعد از اینکه هر x_j بدست آمد، $x = 1.05x_j$ و Δ را برابر مقدار اولیه اش قرار می‌دهیم و این فرایند تکرار خواهد شد.

هدف این است که برنامه‌ای بنویسیم که از روش نصف کردن بازه برای تعیین پنج مقدار اولیهٔ x که رابطهٔ زیر را برقرار می‌کند، استفاده کند.

۱- اگرچه این روش ریشه‌ها را خواهد یافت اما بهترین شیوه برای انجام این کار نمی‌باشد. چون روشهایی که از تابع *fzero* جهت یافتن ریشه های یک تابع با دقتی خاص استفاده می‌کنند، نیاز به تکرار کمتری جهت یافتن ریشه ها خواهند داشت.



شکل ٤-۱) نمایش روش نصف کردن بازه‌ها

که a یک مقدار ثابت است. مقادیر x_j این تساوی را هنگامیکه کاهش تغییرات x تقسیم بر x کمتر از $t_0 = 10^{-6}$ باشد، برقرار می‌کند. همچنین رابطهٔ $0 \le x_{start} \le x_j$ به ازای جمیع مقادیر $j = 1,2,...,n$ برقرار می باشد.

بنابراین به طور کلی، ورودیهای قسمت ریشه یابی برنامه n، x_{start}، t_0 و Δ است و همچنین برای یک تابع خاص مقدار a نیز جزء ورودی‌ها است.

در فصل پنجم جهت تعیین ریشه‌های یک $f(x)$ دلخواه، این برنامه را به صورت تابعی بازنویسی خواهیم کرد. شکل ٥-۱ را مشاهده کنید. بنابراین $n=5$ و $t_0 = 10^{-6}$ می‌باشد. و اگر $x_{start}=0.2$، $\Delta=0.3$ و $a=\pi$ قرار دهیم، متن برنامه به شکل زیر خواهد بود:

```
n = 5; a = pi; increment = 0.3; tolerance = 1e-6; xstart = 0.2;
x = xstart;
dx = increment;
for m = 1 : n
s1 = sign(cos(a * x));
while dx / x > tolerance
if s1 ~= sign(cos(a * (x + dx)))
dx = dx / 2;
```

```
else
x = x + dx;
end
end
route( m ) = x;
dx = increment;
x = 1.05* x;
end
disp( route )
```

پس از اجرا خروجی به صورت زیر خواهد بود.

0.5000 1.5000 2.5000 3.5000 4.5000

تمرین ها

۴-۱ برنامه‌ای بنویسید که عمل معادل عملگر منطقی که در بخش ٤-۱ معرفی شد را به ازای هر بردار شامل مقادیر h انجام دهد. بطوریکه بردار خروجی v ایجاد شده توسط عملگر منطقی نشان دهد که کدامیک از عناصر، شرط $h>a$ و$h<b$ را برقرار می‌کنند. برنامهٔ خود را با $h=[1\ 3\ 6\ -7\ -45\ 12\ 17\ 9]$، $a=3$ و$b=13$ امتحان کنید.

{پاسخ: $v=[0\ 0\ 1\ 0\ 0\ 1\ 0\ 1]$.}

۴ ۲ واریانس n تا x_j به صورت زیر تعیین می‌شود.

$$s_n^{\,2} = \frac{1}{n-1}\left[\sum_{j=1}^{n} x_j^{\,2} - n\bar{x}_n^{\,2}\right] \qquad n > 1$$

که در آن:

$$\bar{x}_n = \frac{1}{n}\sum_{j=1}^{n} x_j$$

تخمینی از مقدار متوسط است. واریانس توسط تابع var تعیین می‌شود.

برنامه‌ای بنویسید که s_n^2 را به صورت تابعی از n برای $n>1$ و داده‌های $x=[45\ 38\ 47\ 41\ 35\ 43]$ تعیین کند.

{پاسخ: $[24.5000\quad 22.3333\quad 16.2500\quad 24.2000\quad 19.9000]$.}

۴ - ۳ برای هر $a>0$ رابطهٔ زیر، مقدار مثبت $\sqrt{a}$ را با تلرانس t_0 و مقدار آغازین $x_0>0$ تعیین خواهد کرد.

$$x_{n+1}=\frac{1}{2}\left(x_n+\frac{a}{x_n}\right)\qquad n=0,1,2,...$$

که در آن $x_{n+1}\cong\sqrt{a}$ می باشد. (وقتی که $x_0<0$ باشد، منفی ریشه بدست می‌آید).

برنامه‌ای بنویسید که $\sqrt{a}$ را به ازای تلرانس 10^{-6} برای $a=7$ تعیین کند. چند تکرار لازم است اگر الف) $x_0=3$ و ب) $x_0=100$ باشد. تکرار اول تعیین کنندهٔ x_1 است. {تذکر: دقت کنید که رابطهٔ بالا تابعی صریح از n نمی‌باشد. n صرفاً شاخصی است که مبین اندیس x می‌باشد. x_{n+1} تابعی از x_n است. بنابراین هر بار در حلقه مقدارهای x_n و x_{n+1} تغییر می‌کند. در نتیجه باید از مقدار n با خبر باشیم تا تعداد دفعاتی که این رابطه باید استفاده شود تا رابطهٔ بالا همگرا شود را تعیین کنیم.}.

{پاسخ: (الف) $n_{iterations}=4$؛ (ب) $n_{iterations}=10$.}

۴ - ۴ با توجه به رابطهٔ زیر :

$$x_{n+1}=x_n^{\,2}+0.25\qquad n=0,1,2,...,N$$

برای $x=0$ دو برنامه بنویسید که مقادیر x_n را برای $n=0,\ 5,\ 10,\ ...,\ 200$ نشان دهد. در برنامهٔ اول از حلقهٔ for و در برنامهٔ دوم از حلقهٔ $while$ استفاده کنید. x_n در چه مقادیری همگرا خواهد شد ؟ برای سومین آرگومان تابع $plot$ از $plot(...,...,'ks')$ استفاده کنید. که مقادیر x_n را به صورت مربع‌هایی نشان خواهد داد. مقادیر محور x، مبین مقادیر n هستند و مقادیر محور y، مبین مقادیر x_n هستند. توجه کنید که همهٔ x_nها برای $n=0,\ 1,\ 2,\ ...,\ 200$ باید حساب شوند ولی از هر 5 تا x_n فقط یکی کشیده خواهد شد. تفاوت این تمرین با تمرین ٤-۳ این است که مقادیر x_n باید به عنوان عناصر بردار ذخیره شود تا اینکه عناصر صحیح نشان داده شوند.

۴ - ۵ روش آماری $chi-square$ توسط رابطهٔ زیر تعیین می‌شود:

$$X^2=\sum_{i=1}^{k}\frac{(x_i-e_i)^2}{e_i}$$

که e_i و x_i متغیرهای مستقلی به طول k هستند.

اگر $e_i < 5$ باشد، آنگاه e_i و x_i باید با مقادیر متناظر e_{i+1} و x_{i+1} ترکیب شوند.

اگر مجموع $e_i + e_{i+1}$ همچنان کمتر از 5 باشد، آنگاه e_{i+2} به مجموع $e_i + e_{i+1}$ اضافه خواهد شد. این عمل تا وقتی که این حاصل جمع بزرگتر از 5 شود تکرار خواهد شد. وقتیکه که $e_i \geq 5$ باشد و مجموع سایر مقادیر e_{i+1}، e_{i+2}،...، e_k کمتر از 5 باشد، در این صورت این مقادیر به مقدار e_i افزوده خواهد شد. برنامه‌ای بنویسید که X^2 را به شیوه‌ای که در بالا توضیح دادیم، حساب کند. نتایج خود را با بردارهای زیر چک کنید که سه حالت مختلف را بیان می‌کند.

الف) $x = [1\ 7\ 8\ 6\ 5\ 7\ 3\ 5\ 4]$ و $e = [2\ 6\ 10\ 4\ 3\ 6\ 1\ 2\ 3]$.

ب) $x = [7\ 11\ 13\ 6]$ و $e = [6\ 10\ 15\ 7]$.

پ) $x = [3\ 14\ 20\ 25\ 14\ 2\ 0\ 1\ 0]$ و $e = [4\ 12\ 19\ 19\ 14\ 8\ 4\ 2\ 1\ 1]$.

این برنامه را برای استفاده در تمرین ۱۴- ۶- ب ذخیره کنید. {راهنمایی: بیشتر برنامه های فشرده به وسیلهٔ انجام آزمایش روی عناصر $cumsum(e)$ بدست می‌آیند، که طول e همانطور که مقداریابی انجام می‌شود، تغییر می کند.

{پاسخ:

(الف) $e_{modified} = [8\ 10\ 7\ 6\ 6]$، $x_{modified} = [8\ 8\ 11\ 7\ 12]$ و $X^2 = 8.8524$.

(ب) $e_{modified} = [6\ 10\ 15\ 7]$، $x_{modified} = [7\ 11\ 13\ 6]$ و $X^2 = 0.6762$.

(پ) $e_{modified} = [16\ 19\ 19\ 14\ 8\ 8]$، $x_{modified} = [17\ 20\ 25\ 14\ 6\ 3]$ و $X^2 = 5.6349$.

۴ - ۶ دو چند جمله ای به شکل زیر را در نظر بگیرید:

$$y(x) = p_1 x^n + p_2 x^{n-1} + \ldots + p_n x + p_{n+1}$$

$$z(x) = s_1 x^m + s_2 x^{m-1} + \ldots + s_m x + s_{m+1}$$

برنامه‌ای بنویسید که آنها را با هم جمع کند؛ وقتی که $m = n$، $m < n$ یا $m > n$ باشد، $h(x) = y(x) + z(x)$ خواهد بود. جمع چند جمله‌ایها توسط جمع کردن عناصر با توان یکسان انجام می‌شود. فرض کنید که ورودی برنامه بردارهای $p = [\,p_1\ p_2\ \ldots\ p_n\ \ p_{n+1}\,]$ و $s = [\,s_1\ s_2\ \ldots\ s_m\ \ s_{m+1}\,]$ باشد.

برنامهٔ خود را با اطلاعات زیر چک کنید.

(الف) $p=[1\ \ 2\ \ 3\ \ 4]$ و $s=[10\ \ 20\ \ 30\ \ 40]$.

(ب) $p=[11\ \ 12\ \ 13\ \ 14]$ و $s=[101\ \ 102]$.

(پ) $p=[43\ \ 54\ \ 55]$ و $s=[77\ \ 66\ \ 88\ \ 44\ \ 33]$.

{پاسخ: (الف) $h=[11\ \ 22\ \ 33\ \ 44]$؛ (ب) $h=[11\ \ 22\ \ 114\ \ 116]$؛

(پ) $h=[77\ \ 66\ \ 131\ \ 98\ \ 88]$.}

۴ - ۷ برنامه‌ای بنویسید که روزهای هفته را برای سال های *1999* و *2000*، حساب کند، که ورودی آن به صورت *month/day/year – xx/xx/xxxx.* می‌باشد. خروجی برنامه را به صورت زیر در آورید:

The date 5/31/2000 is the 152 day of the year and falls on wednesday.

توابع *str2num* و *deblank* و *findstr* مفید خواهند بود.

فصل پنجم

توابع

۵-۱ مقدمه

۵-۱-۱ چرا از توابع استفاده می کنیم؟

۵-۱-۲ نامگذاری توابع

۵-۱-۳ طول توابع

۵-۱-٤ غلط یابی توابع

۵-۲ فایل تابع

۵-۲-۱ **شکل اول**: آرگومانهای ورودی به صورت جداگانه شناخته می شوند و یک متغیر خروجی داریم.

۵-۲-۲ **شکل دوم**: آرگومانهای ورودی به صورت یک بردار نشان داده می شوند و یک متغیر خروجی داریم.

۵-۲-۳ **شکل سوم**: آرگومانهای ورودی به صورت یک بردار نشان داده می شوند و هر متغیرخروجی به صورت جداگانه شناخته می شود.

۵-۲-٤ دو مورد خاص

۵-۳ تابع *INLINE*

۵-٤ ایجاد توابعی که از تابع *FEVAL* استفاده می کنند (تابعی از توابع)

۵-۵ توابعی که از تابع *FEVAL* استفاده می کنند

۵-۵-۱ صفر های توابع - توابع *FZERO*، و *ROOTS/POLY*

۵-۵-۲ انتگرالهای عددی - توابع *QUAD8*، *TRAPZ*, *POLYAREA*

۵-۵-۳ مینیمم محلی یک تابع - تابع *FMINBND*

در این فصل چگونگی ایجاد توابع و کاربرد متنوع آنها در نرم افزار مطلب مورد بررسی قرار می‌گیرد. همچنین چندین تابع داخلی مطلب که در به دست آوردن راه‌حل‌های عددی در مسائل مهندسی کاربرد فراوان دارند معرفی می‌شود.

۵-۱ مقدمه

یکی از اشکال *m file* برنامه و شکل دیگر آن، تابع می‌باشد. توابع، برنامه هایی هستند که فضای کاری مستقل و محلی خود را در محیط مطلب ایجاد می‌کنند. تمامی متغیرهایی که در یک تابع تعریف می‌شوند برای آن تابع، محلی محسوب می‌شوند و با متغیرهای مشابه همنام دیگری که در یک برنامه و یا تابع دیگری استفاده می‌شوند هیچگونه ارتباطی ندارند. تمامی توابع داخلی مطلب از این نوع هستند. اولین خط غیر دستوری یک تابع باید از یک فرمت مشخص که در بخش ۵-۲ بیان شده است تبعیت کند. برنامه های مطلب که توسط یک کاربر ایجاد می‌شود شامل تعدادی دستورات می‌باشد که آنها نیز به نوبهٔ خود دارای تعداد دلخواهی از توابع ساخته شده توسط کاربر و توابع داخلی مطلب می‌باشد.

۵-۱-۱ لزوم استفاده از تابع

در حالیکه مطلب دارای تعداد بیشماری تابع داخلی می‌باشد دلایل بسیاری برای ایجاد تابع در این نرم افزار، وجود دارد. (بخش ۵ -۵ را ببینید) که عبارتند از:

پرهیز از نوشتن برنامه های تکراری.

محدود کردن اثر تغییرات به بخش‌های خاصی از برنامه.

ارتقاء برنامه برای استفاده های مجدد.

کاهش پیچیدگی برنامهٔ اصلی از طریق خواناتر شدن و همچنین افزایش قابلیت کنترل، کاهش پیچیدگی برنامه برای خوانا تر شدن آن.

تفکیک شدن عملیات پیچیده.

بهبود انتقال برنامه.

غلط یابی و تفکیک خطاها.

بهبود عملکرد از طریق بهینه سازی شیوه‌ها.

تقسیم برنامهٔ اصلی به چند بخش مجزا می‌تواند با استفاده از توابع فراهم شود. این مسئله باعث به حداقل رسیدن استفادهٔ بی‌مورد از داده‌ها می‌شود، چرا که داده‌های مورد نیاز هر تابع بر اساس ضرورت استفاده از آن داده تعیین می‌شود.

۵-۱-۲ نامگذاری توابع

نامگذاری توابع باید به گونه‌ای صورت گیرد که اولاً دارای معنا و مفهوم بوده و ثانیاً بیانگر وظیفه‌ای که تابع بر عهده دارد، باشد. طول متداول برای نامگذاری توابع بین *9* تا *20* کاراکتر می‌باشد و باید استانداردها و قراردادهای موجود را رعایت کند.

برای مثال، نام تمامی برنامه می تواند با *scr* شروع شود، در صورتیکه در فایلهای تابع نمی‌توان از این پیشوند استفاده کرد. انتخاب صحیح اسم توابع زمانی میسر خواهد بود که استفاده از توضیحات راجع به تابع در نام آن به حداقل رسانده شود. توصیه می‌شود که قراردادها و اصول مربوط به نامگذاری توابع را که در بخش ۱- ۳ بیان شده، مرور نمایید.

۵-۱-۳ طول توابع

طول یک تابع می‌تواند از دو خط تا بیشتر از صدها خط تغییر کند. اما باید به مسئلهٔ خوانا بودن آن نیز توجه کرد . برای مثال، تابع *sin (x)*، تابعی کاملاً خوانا است. در صورتیکه تابع*sin cos tan (x)*، که می‌تواند *sin*، *cos* و *tan* را محاسبه کند، به دلیل اینکه سه عمل مجزا را که هیچگونه ارتباطی به یکدیگر ندارند، انجام می‌دهد، از درجهٔ خوانایی کمتری برخوردار است. یک تابع می‌تواند جهت ایجاد توابع خوانای دیگر مورد استفاده قرار گیرد . یک مزیت دیگر ساخت توابع خوانا قابلیت اعتماد بیشتر

آنها است که دلیل آن نرخ خطای کمتر آنهاست. هنگامی که توابع به حد کافی خوانا نباشند، اغلب مشکلاتی در تفکیک خطاها به وجود می‌آید .

۵-۱-۴ غلط یابی توابع

در هنگام ساخت و ایجاد توابع، برنامه‌ها باید بصورت مستقل پس از نوشتن هر چند خط، اصلاح شود تا کاربر از کارکرد درست برنامه اطمینان یابد. کارکرد مطلب در این زمینه بسیار ساده و چشمگیر است. با برداشتن علامت نقطه ویرگول از انتهای دستورات می‌توان به این هدف دست یافت. به علاوه با حذف علامت نقطه ویرگول در برنامه‌هایی که از بردارها و ماتریسهای بزرگ و تکنیکهای حل حلقهٔ تکرار در آنها استفاده نمی‌شود، کاربر زمان کمتری را جهت غلط یابی صرف خواهد نمود. غلط‌یابی باید به همراه چند نوع محاسبه یا تخمین مستقل انجام پذیرد. در هنگام انجام مرحلهٔ غلط یابی و اصلاح، خطوطی از برنامه که جهت نمایش خروجی های میانی استفاده می‌شوند، باید تا زمان اصلاح تمام تابع و اجرای صحیح آن توسط علامت % از جمع دستورات اجرایی مطلب خارج شده و پاک نشوند، در صورت لزوم، تنها هنگامی باید به سراغ بهبود تابع و کاهش زمان اجرای برنامه رفت که یک تابع به درستی اجرا شود. ساخت و ایجاد برنامه‌هایی که به درستی اجرا می‌شوند همواره اولین هدف کاربر خواهد بود.

۵-۲ فایل تابع

یک تابع حداقل دارای دو خط برنامه می‌باشد. خط اول آن دارای فرمت خاصی است که از سوی نرم‌افزار مطلب تعیین می‌شود. لزومی به کاربرد عبارت یا کاراکتر پایانی نظیر عبارت *end* که برای دستوراتی نظیر *if،while،for* و ساختار *switch* ضروری می‌باشد، نیست. علاوه بر این، نام *m file* نیز باید با نام تابع یکسان باشد مگر اینکه اسم فایل دارای پسوند *m*. باشد.

تعداد و نوع (اسکالر، بردار یا ماتریس) متغیرهای ورودی و خروجی تابع توسط اولین خط غیردستوری تابع تعیین می‌شود که در زیر شکل کلی آن آمده است:

```
function  OutputVariable = FunctionName( InputVariables )

% comments

expression( s )
```

توضیحاتی که بلافاصله پس از اولین خط غیر دستوری تابع آورده می‌شود، توسط مطلب جهت ایجاد اطلاعات کمکی در مورد نحوهٔ استفاده از آن تابع مطرح می‌شود. هنگامی که کاربر عبارت زیر را در پنجرهٔ فرمان مطلب تایپ می‌کند ؛

help FunctionName

تمام اطلاعات اولیه ضروری در مورد آن تابع در پنجرهٔ فرمان مطلب ظاهر می‌شود. توضیحاتی که قبل از عبارت *function* آورده شود جزء عبارت‌های کمکی آن تابع نیست. توضیحات به همان شیوه‌ای که برای یک برنامه نوشته می‌شوند، بیان خواهند شد، بجز اینکه هر متغیری که در تابع استفاده می‌شود باید توسط نام‌هایی که برای متغیرهای ورودی استفاده می‌شود و یا توسط یکی از عبارت‌های تابع تعریف شود. متغیرهای خروجی می‌توانند اسکالر، بردار یا ماتریس و حتی متشکل از مقادیر عددی و یا رشته‌ها باشند. ممکن است تابع دارای هر تعداد از متغیرهای ورودی باشد، که هر یک توسط علامت کاما از مابقی جدا شده است. متغیرهای ورودی نیز ممکن است اسکالر، بردار و یا ماتریس و حتی می‌توانند متشکل از مقادیر عددی یا رشته‌ها باشند. بنابراین اگر سه متغیر ورودی a، b و c را داشته باشیم، در این‌صورت اولین خط غیر دستوری تابع به صورت زیر خواهد بود:

$$function\ \ OutputVariable = FunctionName(a,b,c)$$

چندین مفهوم جهت ایجاد صحیح توابع وجود دارد.

اولین نکته این است که لزومی در تطابق نام متغیرهایی که در تابع استفاده می‌شوند با نام‌های متناظری که هنگام فراخوانی تابع از طریق پنجرهٔ فرمان مطلب، برنامه و یا توابع دیگر مورد استفاده قرار می‌گیرند، وجود ندارد. یعنی این، مکان متغیرهای ورودی در لیست آرگومان‌های داخل پرانتز است که چگونگی انتقال اطلاعات را تعیین می‌کند. بدین صورت که اولین آرگومان در عبارت فراخوانی، مقدار خود را به اولین آرگومان در خط تعریف (اولین خط غیر دستوری) تابع انتقال می‌دهد و در مورد سایر آرگومانها فرآیند مشابهی طی می‌شود.

نکتهٔ دوم اینکه نام‌هایی که برای هر آرگومان تابع انتخاب می‌شوند برای برنامهٔ آن تابع بصورت محلی بوده و تنها زمانی دارای معنا خواهند بود که در متن برنامهٔ تابع مورد استفاده قرار گیرند. اسامی مشابه می‌توانند در یک برنامهٔ مجزا که این تابع را فرا می‌خواند و یا در تابع دیگری که توسط این تابع استفاده می‌شود، بکار برده شود. اما اسامی که برای هر متغیر ورودی در عبارتهای فراخوانی تابع استفاده می‌شود، باید از همان نوعی (اسکالر، بردار، ماتریس و یا رشته) باشد که در برنامهٔ تابع بکار رفته است، تا دستورات تابع همانگونه که کاربر انتظار دارد اجرا شوند.

برای مثال هنگام ضرب دو بردار سطری ممکن است در صورت عدم تطابق مرتبه‌های دو بردار پیغام خطایی ظاهر شود. به علاوه، نام‌هایی که برای متغیرهای ورودی توابع استفاده می‌شوند معادل مقادیر ظاهر شده در طرف چپ علامت تساوی هستند. بنابراین در مثال فوق متغیرهای ورودی *(a,b,c)* معادل عبارات:, ...=a ; ....=b ; ....=c می‌باشند. در نهایت، چندین حالت برای متغیرهای خروجی *(out put Variable)* وجود دارد که در ادامه معرفی شده‌اند.

فایل تابع می‌تواند در هر دایرکتوری که از سوی کاربر تعیین خواهد شد، و دارای نام *Function Name.m* است ذخیره شود. برای نشان دادن چگونگی ایجاد یک تابع، مثالی می‌آوریم و سه حالتی که می‌توان برای آن پیش بینی کرد را نشان خواهیم داد.

۵ -۲-۱ شکل اول: آرگومانهای ورودی بصورت جداگانه شناخته می شوند و یک متغیر خروجی داریم.

معادلات زیر را که باید از طریق یک تابع محاسبه شوند در نظر بگیرید:

$$x = cos(at) + b$$

$$y = |x| + c$$

مقادیر x، y باید توسط تابع باز گردانده شوند. اکنون تابعی با نام *ComputeXY* ایجاد می‌کنیم که بعنوان یک فایل با نام *ComputeXY.m* ذخیره می‌شود، تا این کمیت‌ها را محاسبه نماید[۱].

```
function  zanswer = ComputeXY(t,a,b,c)
% Computation of -
% x = cos(at) + b
% y = |x| + c
```

---

۱ - توضیحات تابع در این مثال آورده شده تا کاربرد آن نشان داده شود. در شمار زیادی از برنامه ها و توابع نشان داده شده در این کتاب، توضیحات مربوط به آنها جهت افزایش خوانایی حذف شده است. به هر حال در اغلب موارد، موضوعات مهم برنامه ها در داخل متنهایی که به همراه توابع بکار برده می شود، بحث شده است و یا این موضوعات با توجه به متن توابع و دستورات آنها واضح می باشند.

```
% Scalars : a,b,c
% Vector : t
% Matrix : zanswer
x = cos(a*t)+b;
zanswer = [x;abs(x)+c];
```

در صورتیکه کاربر دستور زیر را در پنجرهٔ فرمان مطلب تایپ می‌کند

```
help ComputeXY
```

عبارت زیر نمایش داده خواهد شد:

```
Computation of –
x = cos(at)+b
y = |x|+c
Scalars : a,b,c
Vector : t
Matrix : zanswer
```

مطلب توابع را از طریق اسم فایل و نه توسط خصوصیات متغیرهای ورودی و خروجی شناسایی می‌کند. بنابراین کاربر با توجه به اینکه متغیرهای ورودی و خروجی چگونه توسط تابع مورد استفاده قرار می‌گیرند، باید از تعداد و نوع متغیر اطمینان حاصل کند. این محدودیت‌ها باید در توضیحات تابع درج شود، تا در هنگام استفاده از فرمان *help* ، بتوان به این اطلاعات دست یافت. در این مورد ما فرض خواهیم کرد که t یک اسکالر و یا یک بردار می‌باشد و همچنین a، b و c عدد می‌باشد. بنابراین هنگامیکه t یک بردار با طول n_t باشد تابع یک ماتریس $(2 \times n_t)$ را با نام *Zanswer* باز می‌گرداند که در آن *Zanswer (1,:)=x(:)* و *Zanswer(2,:) =y(:)* می‌باشد.

اکنون توجه شما را به قسمتی از برنامه که وظیفهٔ فراخوانی این تابع را بر عهده دارد معطوف خواهیم ساخت؛ یکی از صور ممکن مطابق زیر می‌باشد:

```
vick = ComputeXY(0 : pi / 4 : pi,1.4,2,0.75);
```

با مقایسهٔ موقعیت آرگومانها با موقعیت آنها در تابع مقادیر زیر حاصل می‌شود:
$t=0,\pi/4,\pi/2,3\pi/4,\pi$ ، $a=1.4$ ، $b=2$ و $c=0.75$ می باشد. پس از اجرای این عبارات:

$$vick(1,:) \rightarrow [3.0000 \quad 2.4540 \quad 1.4122 \quad 1.0123 \quad 1.6910]$$

$$vick(2,:) \rightarrow [3.7500 \quad 3.2040 \quad 2.1622 \quad 1.7623 \quad 2.4410]$$

که در آن x=vick(1,:) و y=vick(2,:) می‌باشد. تذکر این نکته لازم است که اگر تابع *Zanswer* به شکل زیر نوشته می‌شد.

```
zanwer = [ x  abs( x ) + c ];
```

در آن صورت اجرای تابع *ComputeXY* منجر به ایجاد برداری $(1\times 2n_t)$ به شکل زیر می‌شد:

```
vick → [ 3.0000  2.4540  1.4122  1.0123  1.6910  3.7500  3.2040  2.1622
         1.7623  2.4410 ]
```

که در این مورد *x=vick(1:5)* و *y=vick(6:10)* می‌بود.

۵-۲-۲ شکل دوم : آرگومانهای ورودی به صورت یک بردار نشان داده می‌شوند و یک متغیر خروجی داریم.

برنامهٔ متناظر

```
vick = ComputeXY ( 0 : pi / 4 : pi,
[ 1.4  2  0.75 ]);
```

برنامهٔ تابع

```
function  zanswer = ComputeXY(t,w)

    x = cos( w(1) * t ) + w( 2 );

    zanswer  = [x; abs (x) + w(3)];
```

با مقایسهٔ برنامهٔ متناظر و برنامهٔ تابع در می‌یابیم که:

$w(1)=a=1.4$ ، $w(2)=b=2$ ، $w(3)=c=0.75$ که در آن a ، b و c به سه کمیت تعریف شده در شکل اول باز می‌گردند.

۵-۲-۳ شکل سوم : آرگومانهای ورودی به صورت یک بردار نشان داده می‌شوند و هر متغیر خروجی به صورت جداگانه شناخته می‌شود.

برنامهٔ متناظر **برنامهٔ تابع**

```
[u,v] = ComputeXY (0 : pi / 4 : pi,
[1.4  2  0.75]);
```

```
function [x,y] = ComputeXY(t,w)
x = cos(w(1)*t) + w(2);
y = abs(x) + w(3);
```

در عبارت *function[x,y]*، وجود علامت ویرگول ضروری می‌باشد، ولی در عبارت *[u,v] = ...*، علامت ویرگول می‌تواند با یک جای خالی جایگزین شود. در برنامهٔ فراخوانی تابع *ComputeXY* اگر از شکل *q = ComputeXY (...)* استفاده شود، در این‌صورت تنها اولین مقدار به *q* نسبت داده خواهدشد، یعنی *q* متناظر با بردار *x* می‌شود. بنابراین، هنگامی که عبارت:

```
[u,v] = ComputeXY(0 : pi / 4 : pi,[1.4  2  0.75]);
```

اجرا می‌شود، خواهیم داشت:

$$u \rightarrow [3.0000 \quad 2.4540 \quad 1.4122 \quad 1.0123 \quad 1.6910]$$

$$v \rightarrow [3.7500 \quad 3.2040 \quad 2.1622 \quad 1.7623 \quad 2.4410]$$

در صورتی که وقتی عبارت؛

```
q = ComputeXY(o : pi / 4 : pi,[1.4  2  0.75]);
```

اجرا می‌شود، داریم:

$$q \rightarrow [3.0000 \quad 2.4540 \quad 1.4122 \quad 1.0123 \quad 1.6910]$$

و مقادیر متناظر با *y* از طریق برنامه قابل دسترسی نخواهد بود.

در نتیجه شکل اخیر فراخوانی تابع در اکثر اوقات با مقادیر متناظر در دستور تعریف تابع هم خوانی دارد. بسیاری از توابع داخلی مطلب از این ویژگی به صورت مستقل از متن آن استفاده می‌کنند. برای مثال فایل کمکی مربوط به دستور *polyval* را مشاهده کنید.

از آنجا که نقش آرگومان‌ها در تعریف تابع، اختصاص مکانی برای مقادیر عددی است باید هنگام اجرای تابع، در جای خاص خود قرار گیرند، لذا کاربر می‌تواند هر عبارتی را که از لحاظ ساختاری صحیح باشد بعنوان آرگومانهای تابع قرار دهد.

برای روشن‌تر شدن این مسئله، برنامهٔ زیر را که برای تابع از نوع شکل سوم نوشته شده است در نظر بگیرید:

```
n = 4;
c = linspace(1,1.4,n);
for k = 1:n
[u,v] = ComputeXY(o, pi/4:pi,[c(k) sqrt(1.8/(1+k)^3)1/0.85]);
⋮
```

بعنوان آخرین نکته به موردی اشاره می‌کنیم، موردی که در آن نتایج خروجی تابع بصورت یک بردار است و در برنامه به صورت سطری از ماتریس دوباره تعریف می‌شود. برای سادگی فرض می‌کنیم که تنها مقادیر x را می‌خواهیم و این مقادیر را باکمک حلقهٔ تکرار *for* باز می‌گردانیم. بنابراین، بخشی از برنامه می تواند به صورت زیر نوشته شود:

```
n = 4;
c = linspace(1,1.4,n);
p = zeros(4,n);
for k = 1:4
p(k,:) = ComputeXY(o, pi/4:pi,[c(k) sqrt(1.8/(1+k)^3) 1/0.85]);
```

روشن است که در این مورد $p=(=u=cos(at)+b)$ یک ماتریس (5×4) می باشد؛ که به این دلیل است که $k=1,2,3,4$ و طول بردار t برابر با 5 می‌باشد. به یاد می‌آوریم که شیوهٔ نوشتن $p(k,:)$ به معنای عناصر سطر k ام ماتریس p است که باید عناصر ستونهایش را با مقادیر عددی متناظر عناصر ستونهای بردار سطری ناشی از اجرای تابع *ComputeXY* مقدار دهی کند.

۵-۲-۴ دو مورد خاص

توابع می‌توانند بگونه ای ساخته شوند که یا فقط نتایج را در پنجرهٔ فرمان مطلب نشان دهند و یا آنها را در فایل دیگری ذخیره سازند. در این دو حالت، هیچ مقداری به برنامه‌ای که تابع را فراخوانی کرده است، باز گردانده نخواهد شد. در این صورت شکل خط اول تابع به صورت زیر ساده می‌شود:

```
function  FunctionName( InputVariables )
```

هنگامی که از یک تابع تنها جهت ذخیرهٔ داده‌ها به شیوه ای خاص استفاده می‌شود، تابع نیازمند هیچ آرگومانی نخواهد بود. در این حالت خط اول تابع به صورت :

```
function  OutputVariables = FunctionName
```

و یا

```
function  [ a,b,...] = FunctionName
```

خواهد بود.

توابع معمولاً هنگامی که به آخرین عبارتشان می‌رسند به برنامهٔ اصلی باز می‌گردند.

اگر بخواهیم این بازگشت، سریع‌تر صورت گیرد از دستور زیر استفاده می‌کنیم:

```
return
```

مثال زیر که نشان دهندهٔ بخشی از یک تابع می‌باشد، را در نظر بگیرید:

```
 c = FunctionName( x,a,b )  function
if   length( x ) == 1 | nargin ~= 3
   c = x
   return
end
⋮
```

که در آن دستور *nargin* تعداد متغیرهای ورودی تابع را نشان می‌دهد.

۵-۳ تابع *inline*

کاربر می‌تواند با استفاده از دستور *inline* ، یک تابع محلی در پنجرهٔ فرمان مطلب، در یک برنامه و یا در یک تابع دیگر ایجاد کند مزیت این تابع آن است که لزومی به ذخیرهٔ آن در فایل جداگانه‌ای نمی‌باشد. اما این تابع دارای چند محدودیت است. مثلاً قادر به فراخوانی تابع *inline* دیگری نمی‌باشد. همچنین تابع *inline* تنها می‌تواند از یک عبارت مطلب تشکیل شود و این تابع تنها می‌تواند یک متغیر خروجی داشته باشد. یعنی استفاده از شکل سوم توابع مجاز نخواهد بود. بنابراین هر تابعی که نیاز به عملیات منطقی و یا چندین عملیات مختلف جهت دستیابی به نتایج نهایی داشته باشد، نمی‌تواند از تابع *inline* استفاده کند.

شکل کلی تابع *inline* به صورت زیر است ؛

$$FunctionNa\,me = inline('exp\,ression','p1','p2',...)$$

که در آن *expression* می‌تواند هر عبارت معتبری در مطلب باشد و *p1*، *p2* و . . . نام تمامی متغیرهای ظاهر شده در عبارت *expression* هستند. در مثال زیر دستور *inline* را توضیح می‌دهیم:

اجازه دهید تابع f بر حسب x را به شکل زیر تعریف می‌کنیم :

$$f(x) = x^2\cos(ax) - b$$

که در آن a و b اسکالر می‌باشند. در این صورت دستور ؛

$$FofX = inline('x.^2.*\cos(a*x) - b','x','a','b')$$

خروجی زیر را در پنجرهٔ فرمان مطلب نمایش خواهد داد.

$$FofX =$$

$$inline\ \ function:$$

$$FofX(x,a,b) = x.^2.*\cos(a*x) - b$$

اگر در انتهای عبارت فوق از علامت نقطه ویرگول استفاده کرده بودیم، در این‌صورت خروجی در پنجرهٔ فرمان مطلب نمایش داده نمی‌شد.

بنابراین هنگامیکه عبارت زیر را در پنجرهٔ فرمان مطلب وارد کنیم،

$$g = FofX([pi/3 \quad pi/3.5],4,1)$$

پاسخ سیستم به صورت $g=[-1.5483 \quad -1.7259]$ خواهد بود.

استفاده از شکل *inline* برای توابع و در بسیاری از دستورات مطلب دارای مزیت می‌باشد. که لازم است کاربر تابعی ایجاد کند که در آن دستورات *inline* به طور متوالی اجرا شوند. چندین مثال در مورد چگونگی استفاده از این دستور در بخش ۵ - ۵ آورده خواهد شد.

۵ -۴ ایجاد توابعی که از تابع *feval* استفاده می کنند (تابعی از توابع)

تعداد زیادی از توابع داخلی مطلب وجود دارند که کاربر را ملزم به ایجاد توابع خاص خود که دارای همان فرمت توابع داخلی هستند، می‌کنند. چندین تابع از این قبیل در بخش ۵- ۵ بررسی خواهد شد. به علاوه ممکن است موقعیتی برای کاربر پیش بیاید که کاربر ترجیح دهد از توابع خاص خود استفاده کند. مطلب ابزار مورد نیز این کار را توسط دستور زیر در اختیار کاربر قرار می‌دهد.

feval

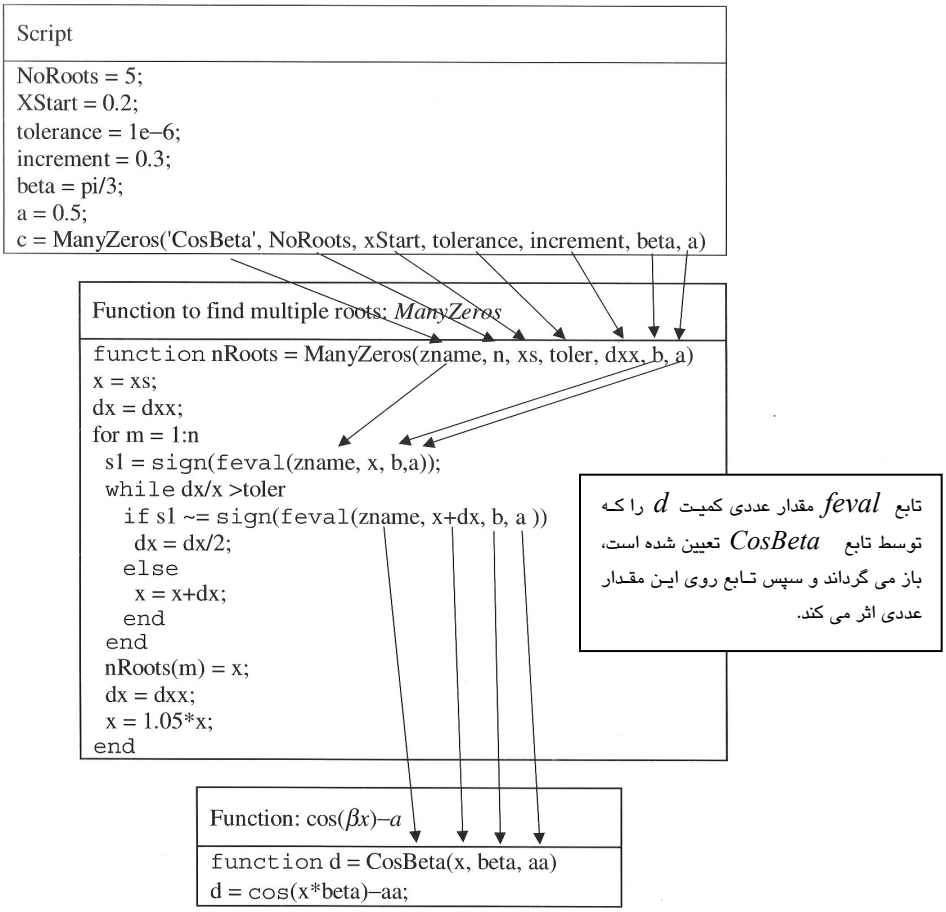
ما چگونگی روند این کار را در قالب یک مثال بر اساس نتایج بخش ٤-۳ -۲ که یک تابع ریشه یاب جهت تعیین m ریشهٔ کوچکتر معادلهٔ $f(x)=0$ بود، بیان خواهیم کرد. اکنون این برنامه را به تابعی با نام *Manyzeros* که بصورت فایلی با نام *Manyzeros.m* می‌باشد، تبدیل می‌کنیم. تابع *f(x)* و نامش کاملاً اختیاری می‌باشند. به علاوه فرض می‌شود که در تابع *f(x)*، تمامی پارامترهای موجود بجز x مقدار دهی شده باشند. به علاوه به خاطر می‌آوریم که برنامهٔ ریشه یاب به چهار ورودی نیازمند بود: m تعداد کوچکترین ریشه های مورد نظر، x_s مقدار اولیه جهت جستجو، t تلرانس محاسباتی که مبین درجه نزدیکی $f(x_{root})$ به صفر می‌باشد و Δ مقدار افزایش اولیه.

برای این مثال *f(x)* را به صورت زیر در نظر می‌گیریم:

$$f(x) = cos(\beta x) - a \qquad a \le 1$$

بنابراین ناچاریم تا دو مقدار β و a را به تابع ایجاد شده توسط کاربر انتقال دهیم. این تابعِ کاربر را، *CosBeta* می‌نامیم و آنرا به صورت فایلی با نام *CosBeta.m* ذخیره می‌کنیم. جهت تکمیل کردن این توابع به یک برنامه نیز نیاز خواهیم داشت. توابع و برنامه در شکل ۵-۱ نشان داده شده‌اند بر اساس فایل کمکی مطلب اولین آرگومان تابع *feval*، اسم تابع و سایر آرگومانها پارامترهایی هستند که باید به تابع انتقال داده شوند. تابع *Manyzeros* به متغیر های ورودی زیر نیاز دارد: (۱) نام تابعی که $f(x)$ را تعریف می‌کند که متغیر رشته ای *zname(='CosBeta');* می‌باشد و (۲) شش پارامتر، که چهار تای اول آن متناظر با m، x_s، t و Δ و دو تای دیگر β و a پارامترهایی هستند که باید به تابع *CosBeta* انتقال داده شوند. با توجه به مقدار m، نتیجهٔ c می‌تواند یک اسکالر *(m=1)* و یا یک بردار به طول m باشد. جهت دستیابی به تابع *CosBeta* و بازگرداندن مقادیر عددی آن، از تابع *feval* استفاده شده است. سپس تابع *sign*، علامت مقدار عددیی که توسط

تابع *feval* باز گردانده شده است را معین کرده است. همچنین توجه کنید که متغیرهایی که بعنوان متغیرهای ورودی تابع در برنامه تعریف شده‌اند و متغیر های مربوط به دو فایل تابع دیگر متفاوتند. اینکار را انجام دادیم تا نشان دهیم تنها موقعیت و توالی استفاده از آرگومانها دارای اهمیت می‌باشد. زیرا متغیرها نسبت به توابع نظیرشان محلی می‌باشند.



تابع *feval* مقدار عددی کمیت d را که توسط تابع *CosBeta* تعیین شده است، باز می‌گرداند و سپس تابع روی این مقدار عددی اثر می‌کند.

شکل ۵–۱) نمایش گرافیکی چگونگی استفاده از تابع *feval* در توابع ایجاد شده توسط کاربر

۵-۵ توابع مطلبی که از تابع *feval* استفاده می‌کنند

مطلب چندین تابع را جهت محاسبهٔ توابع ایجاد شده توسط کاربر ارائه می‌دهد. توابعی که ما در این قسمت آنها را معرفی خواهیم کرد، عبارتند از:

fzero – یک ریشهٔ معادلهٔ $f(x)=0$ را می‌یابد.

roots – صفر های یک چند جمله ای را می‌یابد.

quad8 – به صورت عددی انتگرال $f(x)$ را در بازه ای خاص می‌یابد.

trapz – به صورت عددی انتگرال $f(x)$ را در بازه ای خاص می‌یابد.

polyarea – مساحت محدود بین منحنی های بسته را محاسبه می‌کند.

fminbnd – یک مینیمم محلی تابع $f(x)$ را در بازه ای خاص می‌یابد.

ode45 – به صورت عددی معادلات دیفرانسیل معمولی را حل می‌کند.

fsolve – به صورت عددی یک دستگاه معادلات غیر خطی را حل می‌کند.

هنگام استفاده از این توابع داخلی، آرگومانهای توابع ایجاد شده توسط کاربر و خروجی آنها باید از برخی جنبه‌ها با یکدیگر مطابقت داشته باشد. این الزامات گوناگون در بخشهای زیر نشان داده شده‌اند و همچنین به وضوح در فایل کمکی آن تابع نیز بیان شده است.

۵-۵-۱ صفرهای توابع - توابع *fzero* و *roots/poly*

تابع داخلی *fzero* یک حل معادلهٔ *f(x)* را با تلرانس t_0 در همسایگی x_0 یا در بازهٔ $[x_1 \;\; x_2]$ تعیین می‌نماید. این تابع همچنین می‌تواند پارامترهای p_j را به تابعی که *f(x)* را تعریف می‌کند، انتقال دهد. شکل کلی آن بصورت زیر می‌باشد.

fzero(FunctionName,x0,options, p1, p2,...)

که در آن *FunctionName*، یا نام فایل تابع است که در میان علامت کوتیشن قرار می‌گیرد و شامل پسوند."*m* " نمی‌شود و یا هنگامیکه با استفاده از دستور *inline* ساخته شود، نام متغیر تابع خواهد بود که نیازی به علامت کوتیشن ندارد.

و $x0 = x_0$ یا $x0 = [x_1 \;\; x_2]$ و غیره، پارامتر های p_j می‌باشند و *options* توسط بکار بردن دستور زیر تعیین می‌شود.

optimset

تابع داخلی*optimset* یک تابع تنظیم پارامتر مطلب می‌باشد که توسط چندین تابع مطلب (بیشتر مربوط به جعبه ابزار بهینه‌سازی) استفاده می‌شود. حداقل توصیه می‌شود که از تابع *optimset* جهت غیر فعال کردن نمایش استفاده شود.

بنابراین دستور :

```
options = optimset('display','off');
```

قبل از هر بار استفاده از تابع *fzero* بکار خواهد رفت. فایل کمکی مربوط به تابع *optimset* را جهت دیدن چگونگی انواع نسبت دادنها، که می تواند با توجه به تابع تغییر کند را ببینید.

اولین عبارت تابع بصورت زیر می‌باشد:

```
function  z = FunctionName(x,p1,p2,...)
```

که در آن x متغیر مستقلی است که تابع *fzero* جهت یافتن مقداری که منجر به $f(x=z)\cong 0$ می‌شود. آنرا تغییر می‌دهد. متغیر مستقل باید همواره در این موقعیت قرار داشته باشد. این الزامات برای اکثر توابعی که توسط کاربر جهت محاسبه بوسیله توابع مطلب، ساخته می شوند، صحیح می باشد و همچنین برای تمامی توابعی که در این فصل معرفی شدند رعایت این اصول لازم خواهند بود. اکنون نحوهٔ استفاده از دستور *fzero* را بیان خواهیم کرد.

تابع *f(x)* می تواند یک تابع داخلی مطلب و یا تابع ایجاد شده توسط کاربر باشد. فرض کنید می‌خواهیم یک ریشهٔ *cos(x)* را در نزدیکی *x=6*، تعیین کنیم.

در این صورت عبارت زیر:

```
options = optimset('display','off');
w = fzero('cos',2*pi,options)/pi
```

منجر به نمایش *w=1.5000* خواهد شد که بدلیل اینست که $cos(1.5\pi)=0$ می‌باشد.

و عبارت زیر:

```
options = optimset('display','off');
w = fzero('cos',2.04*pi,options)/pi
```

نتیجهٔ *w=2.5000* را به همراه خواهد داشت و در حالیکه عبارت:

```
options = optimset('display','off');
w = fzero('cos',2.03*pi,options)/pi
```

$w=1.5000$ را نتیجه می‌دهد.

بنابراین برای توابع چند مقداری کاربر باید از شکل $x0=[x_1 \quad x_2]$ استفاده کند و محدوده را به شیوه‌ای صریح تعیین نماید. اما در صورتیکه علامت $f(x1)$ با علامت $f(x2)$ تفاوتی نداشته باشد خطایی رخ خواهد داد. بنابراین عبارت :

```
options = optimset('display','off');
w = fzero('cos',[0 2*pi],options)/pi
```

منجر به ایجاد خطا می‌شود، در صورتیکه عبارت:

```
options = optimset('display','off');
w = fzero('cos',[0.6*pi 2*pi],options)/pi
```

مانند گذشته $w=1.5000$ را نتیجه خواهد داد. از اینرو برای توابع چند مقداری که مشخصاتشان برای کاربر معلوم نمی‌باشد، کاربر باید ابتدا تابع را رسم کند تا به محل تقریبی ریشه‌ها دست یابد.

از طرف دیگر، اگر ریشهٔ معادلهٔ $J_1(x)=0$ (تابع بسل[1] نوع اول از مرتبهٔ 1) در نزدیکی عدد 3 مطلوب باشد، در این صورت عبارت :

```
options = optimset('display','off');
w = fzero('besselj',3,options)
```

آنچنان که خواسته شده است، اجرا نخواهد شد، بخاطر اینکه آرگومانهای تابع *besselj*، *(n,x)* *besselj* می‌باشند، که در آن n مرتبه (در اینجا $n=1$) و x متغیر مستقل می‌باشد. بنابراین n اولین آرگومان خواهد بود و نه x. از اینرو، مجبور به ایجاد تابع جدیدی مطابق زیر خواهیم بود :

```
function v = besseljx(x,n)
```

---

۱ – برای مثال بـه منبـع زیـر مراجعـه شـود. *F.B. Hildbrand، Advanced Calculus for Application، Saddle River، Prentice -Hall، NJ,1976.*

```
v = besselj(n,x);
```

و سپس تابع *fzero* را به شکل زیر استفاده خواهیم کرد :

```
options = optimset('display','off');
a = fzero('besseljx',3,options,1)
```

که نتیجهٔ صحیح *a=3.8317* را باز خواهد گردانید. توجه کنید که بجای انتقال دادن پارامترها به تابع *besseljx* ، مجبور به قرار دادن مقدار *1* در مکان چهارم تابع *fzero* شدیم.

به عنوان مثال دیگری معادلهٔ زیر که بیان کنندهٔ ضرایب فرکانسهای طبیعی یک تیر دو سر گیردار می‌باشد را در نظر بگیرید :

$$f(x) = \cos(x)\cosh(x) - 1$$

ابتدا تابعی با نام *ccbeam* مطابق زیر ایجاد خواهیم نمود.

```
function  s = ccbeam(x)
s = cos(x)*cosh(x)-1;
```

در اینصورت عبارت زیر :

```
options = optimset('display','off');
q = fzero('ccbeam',4,options)
```

مقدار *q=4.7300* را باز می‌گرداند. تابع *f(x)* در اشکال ٦ - ٨ و ٦ - ٩ ترسیم شده است.

می‌توانستیم این ریشه را با استفاده از دستور *inline* ، به هر یک از دو شیوهٔ زیر بدست آوریم.

شکل اول

```
qcc = inline('cos(x)*cosh(x)-1','x');
options = optimset('display','off');
q = fzero(qcc,4,options)
```

شکل دوم

```
options = optimset('display','off');
```

$q = fzero(inline('cos(x)*cosh(x)-1','x'),4,options)$

همانگونه که قبلاً ذکر شد، هنگامیکه *qcc* توسط تابع *inline* تعریف می‌شود، نیازی به استفاده از کوتیشن در اطراف آن نخواهد بود. این مورد برای تمامی توابع داخلی مطلب، که نیاز به توابع ایجاد شده توسط کاربر دارند، صادق می‌باشد. هنگامیکه قرار است از تابعی بیش از یک بار در یک برنامه استفاده شود، شکل اول در بالا استفاده می‌شود زیرا در شکل دوم تابع به گونه‌ای تعریف نشده است تا برای دستورات دیگر موجود در فایل متنی، قابل دسترسی باشد. استفاده از شکل اول عموماً ترجیح داده می‌شود اکنون پنج ریشهٔ کوچکتر عبارت *f(x)* را (بجز ریشهٔ واضح *x=0*) تعیین خواهیم کرد.

$f(x) = cos(x)cosh(x) - 1$

ابتدا باید تابع را در بازه‌ای از x رسم کنیم تا مکان تقریبی x ها را که رابطهٔ *f(x)=0* را ارضاء می‌کنند بیابیم. سپس از این اطلاعات جهت بدست آوردن پنج بازهٔ شامل صفر استفاده خواهیم کرد. برنامه مطابق زیر می باشد:

```
qcc = inline('cos(x).*cosh(x)-1','x');
options = optimset('display','off');
%   x = linspace(0,20);
%   plot(x,qcc(x))
%   axis([0  20  -10  10])
xo = [3  5];
for   n = 1:5
   q(n) = fzero(qcc,xo,options);
   xo = [1.05*q(n)  q(n)+4];
end
disp(['Lowest  five  roots  are:' num2str(q)])
```

که پس از اجرا خروجی زیر را در پنجرهٔ فرمان مطلب نمایش خواهد داد.

```
Lowest  five roots are : 4.73004  7.8532  10.9956  14.1372  17.2788
```

جهت ترسیم ساده تابع *qcc* از عملیات نقطه‌ای در تعریف تابع استفاده کرده‌ایم. این عملیات نقطه‌ای هنگام استفاده از دستور *fzero* لزومی ندارد. ثانیاً، بدلیل اینکه اندازهٔ *qcc* در یک بازهٔ وسیعی از مقادیر مثبت و منفی تغییر می‌کند، از دستور *axis* جهت محدود کردن مقادیر نشان داده شده جهت افزایش کیفیت گرافیکی نمودار استفاده کرده‌ایم. در این مورد محور y را محدودهٔ بازهٔ *[10 10-]* تعریف کرده‌ایم.

بخش ٦-٢ را مشاهده نمایید. از روی نمودار متوجه می‌شویم که منطقه جستجو برای x می‌تواند از مقداری که 5% بزرگتر از مکان ریشه قبلی است تا مکان ریشه قبلی به اضافهٔ 4 انتخاب شود. که همیشه حد بالای منطقه جستجو را که کمتر از مکان صفر بعدی می‌باشد، مورد بررسی قرار می‌دهد. در حقیقت شش دستور انتهایی برنامه تا زمانی که پنج دستور ابتدایی آن اجرا نشوند، نوشته نخواهند شد. بنابراین هنگامی که این اطلاعات معلوم شود، دیگر نیازی به آنها نخواهد بود و عبارت در خروجی نمایش داده خواهند شد.

اگر برای تابع *f(x)* یک مقدار ثابت (در اینجا 4) برای افزایش مکان ریشه‌ها وجود نداشت و به عوض، چندین مقدار مختلف C_j و $j = 1,2,3,4$ نیاز بود، برنامه فوق به شکل زیر می‌بود:

```
qcc= in line ('cos(x). *cosh(x)-1','x');
options= optimset('display','off');
C=[ 4 5 6 5 0 ] ;
xo= [ 3 5 ] ;
for n = 1: 5
q( n ) = fzero( qcc، xo، options ) ;
xo = [ 1.05*q(n) q(n) + C (n) ] ;
end
disp(['Lowest five roots are: ' num2str(q)])
```

به بردار C، عنصر پنجم صفر افزوده شده است که توسط تابع *fzero* استفاده نمی‌شود ولی به خاطر مقدار اندیس C مورد نیاز می‌باشد.

به عنوان مثال نهایی اجازه دهید مقدار a را که معادله زیر را ارضا می‌کند؛

$$\sum_{j=1}^{1000} \frac{1}{j^2 - a} = 0$$

و پاسخ را در پنجره فرمان مطلب با هشت رقم نشان می‌دهد، تعیین کنیم. ابتدا تابعی به نام *suma* مطابق زیر ایجاد می‌کنیم.

```
function z = suma (a)
z = sum (1. / ([1:1000].^2-a));
```

سپس با حدس اولیهٔ $\pi/2$ خواهیم داشت:

```
options= optimset('display','off');
fofa=f zero ('suma,pi/2,options);
disp(['The value of a is ' num2str (fofa,8)'.'])
```

که پس از اجرا، خروجی زیر را نمایش خواهد داد.

```
The value of a is 2.046576.
```

اگر از تابع *inline* استفاده شود، می توان مقدار a را توسط دستورات زیر محاسبه نمود؛

```
options= optimset('display','off');
fafa= f zero ( in line ('sum(1. / ([1:1000]. ^2 -a )) '،'a'),pi /
2,options);
disp(['The value of a is ' num2str (fofa,8)'.'])
```

هنگامی که *f(x)* یک چند جمله‌ای به شکل زیر باشد:

$$f(x) = c_1 x^n + c_2 x^{n-1} + ... + c_n x + c_{n+1}$$

ریشه‌های آن را می توان بسادگی توسط دستور زیر یافت:

```
roots(c)
```

که در آن $c = [\, c_1\ c_2\ ...\ c_{n+1}\,]$ می‌باشد. برای مثال اگر :

$$f(x) = x^4 - 10x^3 + 35x^2 - 50x + 24$$

در این صورت دستور زیر :

```
r=roots([1 -10  35  -50  24 ])
```

خروجی *r=[4 3 2 1]'* را نمایش خواهد داد. تابع برعکس *roots* به شکل زیر می‌باشد؛

```
c= poly (rts)
```

که *c*، ضرایب چند جمله‌ای را باز می گرداند، و *rts* بردار ریشه‌ها می‌باشد.

در حالت کلی، *rts* برداری متشکل از مقادیر حقیقی و موهومی می‌باشد. چند جمله‌ای‌ها را می‌توان توسط دستور زیر در زیر ضرب نمود.

```
conv(a,b)
```

که در آن *b* و *a* بردارهایی شامل ضرایب چند جمله‌ای‌های مربوطه می باشند. برای مثال، فرض کنید که چند جمله ای دیگری مطابق زیر داشته باشیم:

$$g(x) = x^2 - 4$$

در این صورت ضرایب *h(x)=g(x)f(x)* توسط دستور زیر محاسبه می‌شود.

```
h= conv([1 0 -4 ],[1- 10  35 -50  24 ])
```

که نتیجهٔ زیر به دنبال خواهد داشت:

$$h \rightarrow [1 \quad -10 \quad 31 \quad -10 \quad -116 \quad 200 \quad -96]$$

بنابراین چند جمله ای به شکل زیر خواهد بود:

$$h(x) = x^6 - 10x^5 + 31x^4 - 10x^3 - 116x^2 + 200x - 96$$

ریشه‌های این چند جمله ای باید ریشه‌های $f(x)$ و ریشه‌های $g(x)$، ± 2 باشند. بنابراین عبارت زیر :

```
rh =roots([1 -10  31 -10  -116  200   -96 ])
```

خروج زیر را نشان خواهد داد:

$rh \rightarrow$ `[ 4.0000  -2.0000  3.0000  2.0000  2.0000  1.0000]'`

توجه کنیم که نتایج تابع *roots* به ترتیبی قراردادی ظاهر شده‌اند.

۵-۵-۲ انتگرالهای عددی - توابع *quad8*، *trapz* و *poly area*

تابع داخلی *quad8* انتگرال *f(x)* را به صورت عددی از حد پایین a تا حد بالای b با تلرانس t_0 محاسبه می‌کند. این تابع همچنین می‌تواند پارامترهای p_j را به تابعی که *f(x)* را تعریف می‌کند، انتقال دهد.

شکل کلی استفاده از تابع *quad8* به صورت زیر می‌باشد.

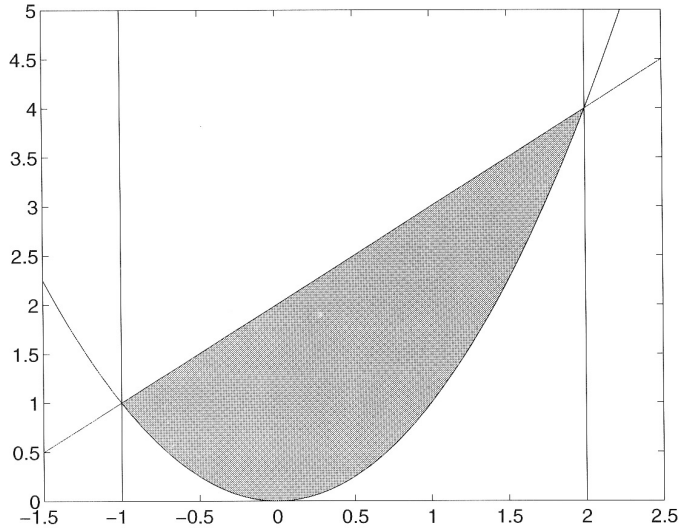
```
quad8(FunctionName,a,b,t0,tc,p1,p2,…)
```

که در آن *FunctionName* نام فایل تابع در میان کوتیشن بدون پسوند *".m"* و یا هنگامی که توسط دستورات *inline* ساخته می‌شود، نام متغیر تابع بدون استفاده از کوتیشن می‌باشد. $t0=t_0$،b=b،$\partial=a$ (هنگامیکه t_0، از دستور حذف شود از مقدار پیش فرض آن استفاده خواهد شد. ، p_1، p_2 و ...، پارامترهای p_j می باشند. هنگامیکه $tc\neq[\,]$، تابع *quad8*، خروجی‌های میانی را نشان خواهد داد.

اولین خط غیر دستوری این تابع به شکل زیر می‌باشد[۱]:

```
function z= FunctionName (x,p1,p2,…)
```

۱- انتگرال دوگانه تابع *g(x,y)* توسط دستور *dblquad* بدست می آید. اما کاربر نمی تواند هر پارامتری را به آن انتقال دهد.



شکل ۵ – ۲) تعیین مرکز سطح هاشور خورده

که در آن x متغیر مستقلی است که تابع *quad8*، عمل انتگرال گیری را بر اساس آن انجام می‌دهد. متغیر مستقل در این تابع، همواره باید در این محل قرار گیرد. اکنون چگونگی استفاده از دستور *quad8* را توضیح خواهیم داد. دو کمیتی که در مکانیک همواره دارای اهمیت بوده است مساحت اشکال دو بعدی و مکان مرکزی سطحشان می‌باشد. اجازه دهید فرض کنیم که دو منحنی با ضابطهٔ $y_j = f_j(x)$ و $j = 1,2$ داریم و محل تلاقی این دو منحنی x_1 و x_2 می‌باشد. در اینصورت مساحت سطح محصور بین این دو منحنی به صورت زیر خواهد بود،

$$A = \int dA = \int_{x_1}^{x_2} (y_2 - y_1)dx$$

و موقعیت مرکز سطح این سطح نسبت به مبدأ به صورت زیر به دست می آید؛

$$x_c = \frac{1}{A}\int x dA = \frac{1}{A}\int_{x_1}^{x_2} x(y_2 - y_1)dx$$

$$x_c = \frac{1}{A}\int y dA = \frac{1}{A}\int_{x_1}^{x_2} \frac{1}{2}(y_2 + y_1)dx = \frac{1}{2A}\int_{x_1}^{x_2} \frac{1}{2}({y_2}^2 - {y_1}^2)dx$$

فرض کنید $y_1 = x^2$ و $y_2 = x + 2$ باشد، که در شکل ۵-۲ نشان داده شده اند. می‌بینیم که محل تلاقی این دو منحنی در $x_1 = -1$ و $x_2 = 2$ می‌باشد. انتگرالهای ذکر شده در قبل منجر به نتایج

$A=4.5$، $x_c = 0.5$ و $y_c = 1.6$ خواهند شد. اکنون این محاسبات را به شیوهٔ عددی انجام می‌دهیم.

از آنجا که ضابطهٔ این منحنی ها نسبتاً ساده می‌باشد، ما از توابع *inline* و *quad8* جهت بدست آوردن نتایج استفاده می‌کنیم. برنامهٔ آن مطابق زیر خواهد بود.

$$area = quad8(inline('x+2','x'),-1,2) - quad8(inline('x.^2','x'),-1,2)$$

$$xc = quad8(inline('x.*((x+2)-x.^2)','x'),-1,2)/area$$

$$yc = quad8(inline('((x+2).^2-x.^4)/2','x'),-1,2)/area$$

اجرای این برنامه خروجی‌های $area=4.5000$، $x_c = 0.5000$ و $y_c = 1.6000$ را ارائه خواهد کرد.

روش دیگر جهت حل تقریبی انتگرالهای یگانه استفاده از دستور

$$trapz(x,y)$$

می‌باشد. در این مورد، کاربر مقادیر x و مقادیر متناظر y را در قالب آرایه‌هایی وارد می‌کند. در اینصورت تابع، مجموع حاصلضرب متوسط مقادیر y و مقادیر متناظر x آنها را محاسبه خواهد کرد. اگرچه ممکن است این تابع دقت تابع *quad 8* را نداشته باشد ولی بدلیل اینکه مانند تابع *quad 8* نیازمند داشتن ضابطهٔ تابع نمی‌باشد، ساده‌تر می‌باشد. بنابراین زمانی از تابع *trapz(x,y)* استفاده می‌کنیم که آرگومانهای انتگرال تنها به صورت آرایه‌ای از اعداد باشند. برای حل مثال قبل از طریق دستور *trapz*، برنامهٔ زیر را خواهیم داشت:

```
x = linspace(-1,2,150);
y = x+2-x.^2;
area = trapz(x,y)
xc = trapz(x,x.*y)/area
yc = trapz(x,(x+2).^2-x.^4)/2/area
```

هنگامی که برنامهٔ فوق اجرا شود، نتایج تا پنج رقم با معنی به صورت $area=4.4998$، $x_c = 0.5000$ و $y_c = 1.6000$ خواهند بود.

مساحت منحنی‌های بسته را می توان توسط دستور :

$polyarea\ (x, y)$

محاسبه نمود که در آن (x,y) مختصه‌های خطوط مستقیم به هم متصلی هستند که این منحنی‌های بسته را به صورت تقریبی تشکیل می‌دهند. در مثال خاص ما، یکی از منحنی‌ها، خط مستقیمی می‌باشد که دارای مختصه‌های انتهایی $(-1,1)$ و $(2,4)$ می‌باشد. برنامهٔ محاسبه این سطح محصور توسط دستور $polyarea$ به صورت زیر خواهد بود:

$x = linspace(-1,2);$

$y = x.^2;$

$x = [-1 \quad 2 \quad x];$

$y = [1 \quad 4 \quad y];$

$area = polyarea(x, y)$

که پس از اجرا نتیجهٔ $area = 4.4995$ را بدنبال خواهد داشت.

انتگرال زیر را که طول یک خط در فضا را محاسبه می‌کند، در نظر بگیرید:

$$L = \int_a^b \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2 + \left(\frac{dz}{dt}\right)^2}\, dt \approx \sum_{i=1}^{N} \sqrt{(\Delta_i x)^2 + (\Delta_i y)^2 + (\Delta_i z)^2}$$

که در آن :

$$\Delta_i x = x(t_{i+1}) - x(t_i)$$

$$\Delta_i y = y(t_{i+1}) - y(t_i)$$

$$\Delta_i z = z(t_{i+1}) - z(t_i)$$

می‌باشد. و $t_1 = a$ و $t_{N+1} = b$ خواهد بود. کمیتهای $\Delta_i x$، $\Delta_i y$ و $\Delta_i z$ توسط دستور زیر محاسبه می‌شوند.

$diff$

که بردار q را از بردار $x = [\, x_1 \;\; x_2 \;\; \ldots \;\; x_n \,]$ ، شامل $n\text{-}1$ عنصر به شکل زیر می سازد:

$$q = [x_2 - x_1, x_3 - x_2, \ldots, x_n - x_{n-1}]$$

برای برداری مثل x، تابع $diff$ می تواند به شکل ساده تر زیر تعریف شود:

$q = x(2:end) - x(1:end-1);$

که در آن *end=length(x)* یک تابع مطلب می‌باشد.

برای نشان دادن چگونگی ارائهٔ تقریبی برای L، فرض می‌کنیم؛

$x = 2t$

$y = t^2$

$z = \ln t$

که در آن $1 \le t \le 2$ و *n=25* می‌باشد. برنامهٔ محاسبه L به صورت زیر می‌باشد:

$t = linspace(1,2,25);$

$L = sum(sqrt(diff(2*t).^2 + diff(t.^2).^2 + diff(\log(t)).^2))$

که پس از اجرا *L=3.6931* را ارائه خواهد کرد.

۵ – ۵ – ۳ مینیمم محلی یک تابع ـ تابع *fminbnd*

تابع *fminbnd*، که متعلق به جعبه ابزار بهینه‌سازی می‌باشد، مینیمم محلی تابع *f(x)* را در بازه $a \le x \le b$ با تلرانس t_0 می‌یابد. این تابع همچنین می‌تواند پارامترهای p_j را به تابع تعریف کنندهٔ *f(x)* انتقال دهد. شکل کلی دستور *fminbnd* به صورت زیر می‌باشد.

```
fminbnd(FunctionName, a, b, options, p1,p2,…)
```

که در آن *FunctionName*، نام فایل تابع در میان علامت کوتیشن بدون پسوند *".m"* و یا هنگامیکه توسط دستور *inline* ساخته شود، نام متغیر تابع بدون استفاده از کوتیشن می‌باشد، $\partial = a$، $b=b$ و متغیر *options* یک بردار اختیاری که پارامترهایش توسط دستور *optimset* تعیین می‌شوند، می‌باشد. (به فایل کمکی مطلب برای تابع *optimset* رجوع شود) و p_1، p_2 و ... پارامترهای p_j می‌باشند.

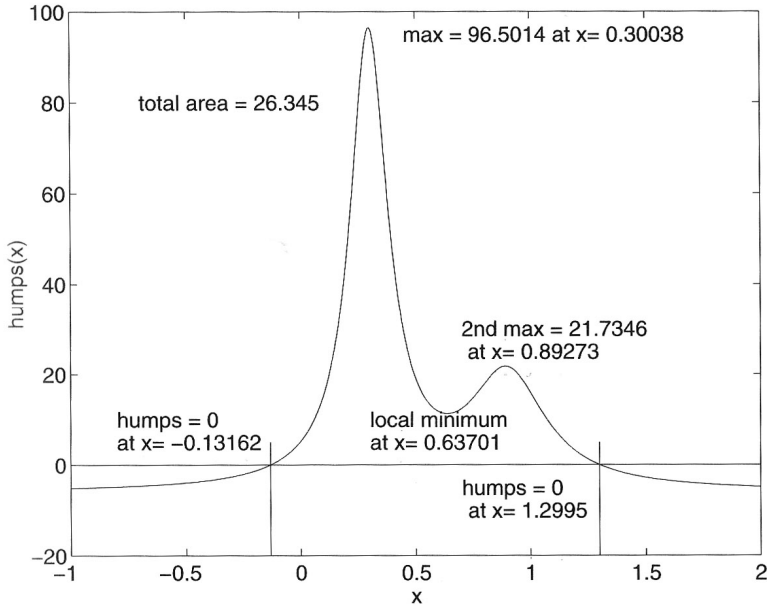
اولین خط غیر دستوری این تابع به شکل زیر می‌باشد؛

```
function  z= FunctionName (x,p1,p2,…)
```

که در آن x متغیر مستقلی است که توسط تابع *fminbnd* جهت یافتن مینیمم *f(x)* تغییر می‌کند. متغیر مستقل باید همواره دراین موقعیت ظاهر شود. اکنون چگونگی استفاده از دستور *fminbnd* را تشریح خواهیم کرد.

تابع *humps* را که توسط مطلب در شکل ۵ -۳ نشان داده شده است، در نظر بگیرید برای یافتن مقدار مینیمم تابع در بازهٔ $0 \le x \le 1$ از برنامهٔ زیر استفاده می‌کنیم:

```
 options = optimset( 'display',off);
xmin=fminbnd('humps',0,1,options)
```



شکل ۵-۳) ویژگیهای نمایش تابع *humps* در مطلب

که پس از اجرا, خروجی آن به صورت $xmin = x_{min} = 0.6370$ خواهد بود. از طرف دیگر، اگر بخواهیم مقدار ماکزیمم تابع *humps* را در این بازه بیابیم، باید تشخیص دهیم که تابع *fminbnd* را روی مقدار منفی و یا معکوس تابع *humps* اثر دهیم. بنابراین با استفاده از منفی مقدار تابع *humps* ، فایل زیر را می سازیم.

```
 options = optimset( 'display','off');
w=fminbnd(inline ('- humps(x)','x')0,1,options)
```

که مقدار $w = 0.300376$ را پس اجرا ارائه می‌کند. مقدار تابع *humps* در مکان این ماکزیمم توسط دستور زیر به دست می‌آید:

```
wmax =humps(0.300376)
```

که نتیجه $w_{max} = 96.5014$ را ارائه می‌کند.

کمیتهای دیگر نشان داده شده در شکل ۵ -۳ می‌توانند با استفاده از تکنیکهای بحث شده در بخشهای ۵-۵-۱ و ۵-۵- ۲ تغییر کنند.

۵-۵-۴ حل عددی معادلات دیفرانسیل معمولی - تابع *ode 45*

تابع *ode45* یک سیستم شامل n معادلهٔ دیفرانسیل معمولی مرتبهٔ اول

$$\frac{dy_i}{dt} = f_i(t, y_1, y_2, ...y_n) \qquad j = 1,2,...,n$$

را مطابق زیر در بازهٔ $t_o \le t \le t_f$ با شرایط اولیهٔ $y_j(t_0) = a_j$ و $j = 1,2,...,n$ حل می‌کند که در آن a_j ها مقادیر ثابتی می‌باشند. آرگومانهای تابع *ode45* به صورت زیر می‌باشند.

$$[t, y] = ode45(FunctionName, [t_0 \quad t_f], [a_1 \quad a_2 \quad ... \quad a_n]', options, p_1, p_2, ...)$$

که در آن خروجی t، یک بردار ستونی از مرتبهٔ $t_o \le t \le t_f$ می‌باشد که توسط تابع *ode 45* تعریف می‌شود. خروجی y ماتریس جوابها می‌باشد که تعداد سطرهای آن متناظر با مرتبه t و تعداد ستونهایش متناظر با حل های زیر می‌باشند:

$$y(:,1) = y_1(t)$$

$$y(:,2) = y_2(t)$$

$$y(:,n) = y_n(t)$$

اولین آرگومان تابع *ode 45*، *Function Name* می‌باشد، که نام فایل تابع در میان علامت کوتیشن بدون پسوند *"m."* و یا زمانیکه توسط دستور *inline* ساخته شود، نام متغیر بدون نیاز به استفاده از علامت کوتیشن خواهد بود. شکل کلی دستور باید مطابق زیر باشد؛

```
function yprime=Function Name(t,y,flag,p1,p2,……)
```

که در آن t متغیر مستقل، y برداری است که عناصرش متناظر با y_j می باشند، $flag$ یک رشته متشکل از حروف کوچک که در این کتاب استفاده نشده است ولی باید نمایش داده شود (فایل کمکی تابع $odefile$ را مشاهده کنید.) p_1 ، p_2 و . . . پارامترهای نسبت داده شده به $FunctionName$ می باشند، و $yprime$ یک بردار ستونی (سطری) با طول n که عناصرش $f_j(t,y_1,y_2,...y_n)$ و $j=1,2,...n$ هستند، می‌باشد، یعنی؛

$$yprime = [f_1; f_2; ...; f_n]$$

متغیر های $yprime$، $FunctionName$ و ... توسط کاربر نسبت داده می‌شوند. آرگومان دوم $ode45$ یک بردار دو عنصری که شامل زمان شروع و پایان جهت انجام حل عددی است، می باشد. این کمیت می‌تواند برداری شامل زمانهای $[t_0\ t_1\ t_2 \ldots t_f]$ که در آنها حل صورت می‌پذیرد، باشد. آرگومان سوم برداری شامل شرایط اولیهٔ $y_j(t_0) = a_j$ باشد. آرگومان چهارم، $options$، معمولاً تهی می‌باشد (فایل کمکی $ode\ 45$ را ببینید). سایر آرگومانها به تابع $Function\ Name$ انتقال داده خواهند شد.

چهار حل کنندهٔ معادلات دیفرانسیل معمولی دیگر در مطلب وجود دارد که هر یک از آنها دارای مزایای خاص خود، بسته به ویژگیها و شرایط مربوطه می‌باشند. که این چهار تابع $ode\ 23$، $ode113$، $ode15S$، $ode23S$ می باشند. نحوهٔ استفاده از این توابع مانند تابع $ode\ 45$ می باشد. به راهنمای کاربران در مطلب و فایلهای کمکی مربوطه برای دستیابی به جزئیات بیشتر رجوع کنید.

اکنون نحوهٔ استفاده از این تابع را با در نظر گرفتن معادلهٔ دیفرانسیل معمولی مرتبهٔ دوم زیر توضیح می‌دهیم؛

$$\frac{d^2y}{dt^2} + 2\xi\frac{dy}{dt} + y = h(t) \qquad (۱-۵)$$

این معادله را می‌توان با احتساب تغییر متغیرهای زیر به صورت دستگاهی متشکل از دو معادلهٔ دیفرانسیل معمولی مرتبهٔ اول در نظر گرفت.

$$y_1 = y$$

$$y_2 = \frac{dy}{dt}$$

در این صورت داریم:

$$\frac{dy_1}{dt} = y_2$$

$$\frac{dy_2}{dt} = -2\xi y_2 - y_1 + h$$

اجازه دهید سه مورد زیر که در همهٔ آنها $\xi = 0.15$، $t_0 = 0$، $t_f = 35$ می باشد را در نظر بگیریم.

مورد اول

$$y_1(0) = 0$$

$$y_2(0) = 0$$

$$h(t) = u(t)$$

که $u(t)$ مبین تابع پله ای واحد می‌باشد.

مورد دوم

$$y_1(0) = 1$$

$$y_2(0) = 0$$

$$h(t) = 0$$

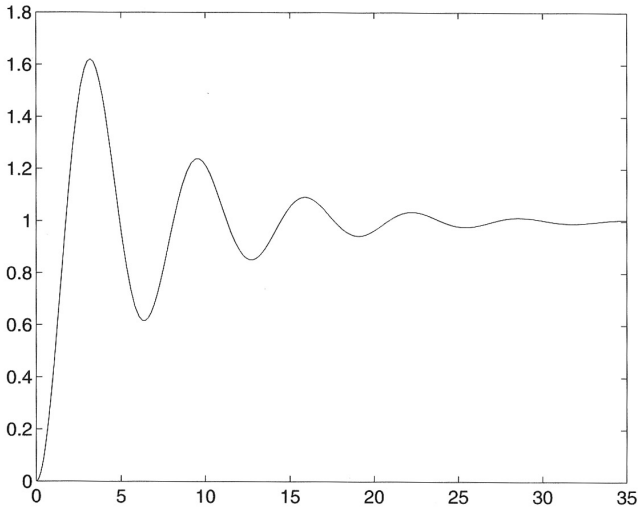
مورد سوم

$$y_1(0) = 0$$

$$y_2(0) = 0$$

$$h(t) = \sin(\pi t/5) \qquad t \le 5$$

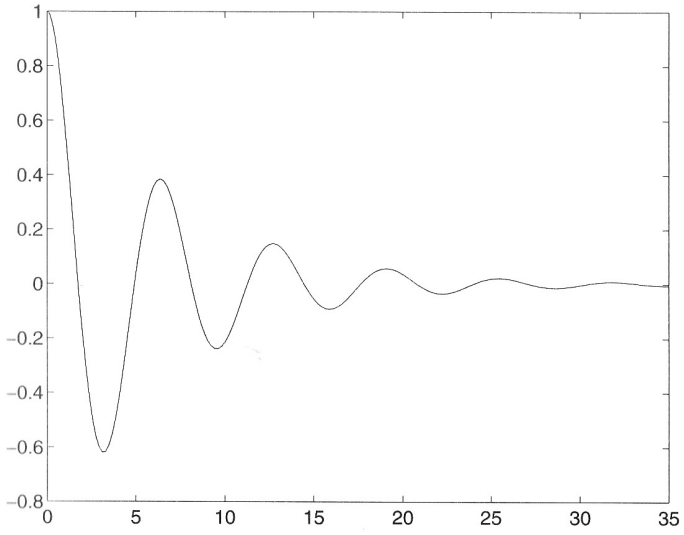
$$= 0 \qquad t > 5$$



شکل۵–٤) پاسخ معادلهٔ (۵–۱) به ورودی تابع پله

در ابتدا تابعی جهت ادارهٔ این سه مورد می‌سازیم و خروجی را به شکلی که توسط دستور *ode45* مورد نیاز است، نشان خواهیم داد.

```
function  s = ForcingFunction(t,w,flag,x,c)

switch  c
 case  1
  s = [w(2);-2*x*w(2)-w(1)+1];
 case  2
  s = [w(2);-2*x*w(2)-w(1)];
 case  3
  h = sin(pi*t/5).*(t<=5);
  s = [w(2);-2*x*w(2)-w(1)+h];
end
```



شکل ٥- ٥) پاسخ معادلهٔ (٥-١) به یک شرط اولیه

که در آن $w(1)=y_1(t)$، $w(2)=y_2(t)$ و $x=\xi$ می باشد.

مورد اول: $y_1(0)=0$، $y_2(0)=0$ و $h(t)=u(t)$ است. برنامه در این حالت به صورت زیر می‌باشد:

```
[tt,yy]=ode45('ForcingFunction',[0 35],[0 0]',[],0.15,1);
plot(tt,yy(:,1))
```

که شکل ٥-٤ را نتیجه خواهد داد. بنابراین؛

$$yy(:,1)=y_1(t)=y(t)$$

$$yy(:,2)=y_2(t)=dy/dt$$

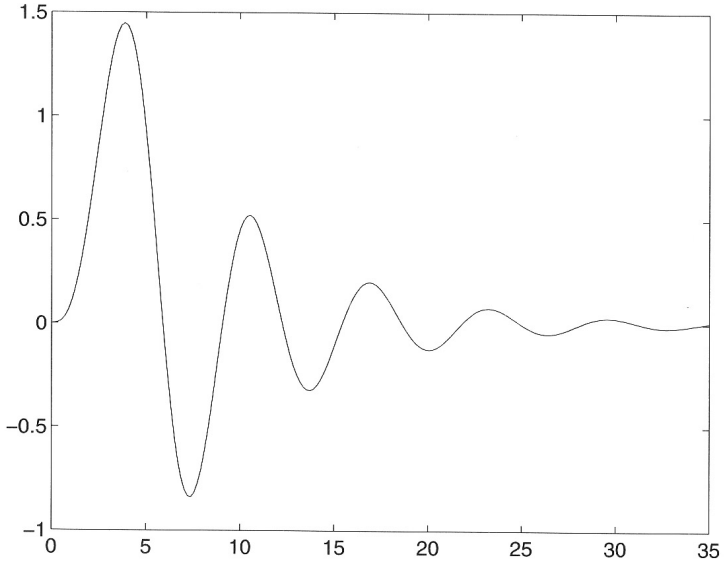
خواهد بود. شیوهٔ دیگر جهت بدست آوردن پاسخ پلهٔ مربوط به یک معادلهٔ دیفرانسیل معمولی استفاده از دستور

step

می باشد. که در جعبه ابزار کنترل مطلب موجود می باشد. فصلهای ۹ و ۱۰ را مشاهده نمایید.

مورد دوم: $y_1(0)=1$، $y_2(0)=0$ و $h(t)=0$ است. برنامه در این حالت به صورت زیر می‌باشد:

```
[tt,yy]=ode45('ForcingFunction',[0 35],[1 0]',[],0.15,2);
plot(tt,yy(:,1))
```



شکل ۵-۶) پاسخ معادلهٔ (۵-۱) به یک موج نیم سینوسی

که شکل ۵-۵ را نتیجه خواهد داد.

مورد سوم: $y_1(0)=0$، $y_2(0)=0$، $h(t)=\sin(\pi t/5),\ t\le 5$ و $h(t)=0,\ t>5$ است. در این مورد مقدار t را توسط جایگذاری آرگومان دوم با بردار زمان، تعیین می کنیم. برنامهٔ آن به شکل زیر خواهد بود.

```
[tt,yy]=ode45('ForcingFunction',linspace(0,35,200),[0 0]',[],0.15,3);
plot(tt,yy(:,1))
```

که پس از اجرا، شکل ۵-۶ را نتیجه می‌دهد.

اکنون چندین مثال دیگر را جهت توسعهٔ استفادهٔ اساسی از دستور *ode 45* بررسی خواهیم کرد.

مثال ۵-۱ جابجایی آزاد در طول یک صفحهٔ عمودی گرم

معادلات توصیف کنندهٔ جابجایی آزاد در طول یک صفحهٔ عمودی گرم، که با یک سیال سردتر در تماس است توسط روابط زیر داده می‌شود (بخش ۱۲-۳-۲ را مشاهده کنید.)

$$\frac{d^3 f}{d\eta^3} + 3f\frac{d^2 f}{d\eta^2} - 2\left(\frac{df}{d\eta}\right)^2 + T^* = 0$$

$$\frac{d^{2^*} T^*}{d\eta^2} + 3pr\ f\frac{dT^*}{d\eta} = 0$$

که در آن $pr = 0.7$ و شرایط مرزی در $\eta = 0$ به صورت زیر می‌باشند:

$$f = 0 \qquad \frac{df}{d\eta} = 0 \qquad \frac{d^2 f}{d\eta^2} = 0.68$$

$$T^* = 1 \qquad \frac{dT^*}{d\eta} = -0.5$$

این دستگاه را می‌توان توسط معرفی مجموعهٔ متغیرهای وابستهٔ زیر به دستگاهی شامل پنج معادلهٔ دیفرانسیل مرتبهٔ اول تبدیل نمود.

$$y_1 = f \qquad y_4 = T^*$$

$$y_2 = \frac{df}{d\eta} \qquad y_5 = \frac{dT^*}{d\eta}$$

$$y_3 = \frac{d^2 f}{d\eta^2}$$

که در آن y_1 تابع جریان، y_2 مبین سرعت، y_3 مبین برش، y_4 نشان دهنده درجهٔ حرارت و y_5 شدت گرما می‌باشد. در اینصورت معادلات دیفرانسیل بر حسب متغیرهای جدید به صورت زیر می‌باشند:

$$\frac{dy_1}{d\eta} = y_2 \qquad \frac{dy_4}{d\eta} = y_5$$

$$\frac{dy_2}{d\eta} = y_3 \qquad \frac{dy_5}{d\eta} = -3pr\ y_1 y_5$$

$$\frac{dy_1}{d\eta} = 2y_2{}^2 - 3y_1 y_2 - y_4$$

که شرایط مرزی متناظر در $\eta = 0$ به صورت زیر خواهد بود.

$$y_1(0) = 0 \qquad y_4(0) = 1$$

$$y_2(0) = 0 \qquad y_5(0) = -0.50$$

$$y_3(0) = 0.68$$

جهت حل این دستگاه معادلات ابتدا باید تابع زیر را جهت تعیین بردار ستونی نشان‌دهندهٔ طرف راست تساویهای پنج معادلهٔ فوق، یعنی *fi*ها، تشکیل دهیم.

```
function  ff = NaturalConv( x, y, flag, pr )
ff =[y(2); y(3);-3* y(1)* y(3)+2* y(2)^2-y(4); y(5);-3* pr* y(1)* y(5)];]
```

برنامهٔ آن به صورت زیر می‌باشد.

```
y0 = [0 0 0.68 1 -0.50];
pr = 0.7;
[ eta  ff ] = ode45('NaturalConv',[0 20], y0,[], pr );
```

نتایج در شکل ۱۲-۱٤ نشان داده می‌شوند.

مثال ۵-۲ آونگ معکوس

پاندول معکوس متصل شده به یک دیسک را مطابق شکل ۱۰ - ۳۵ در نظر بگیرید.

معادلات حرکت خطی شده آن به صورت زیر می‌باشند.

$$ml^2 \frac{d^2\theta}{dt^2} + mrl\frac{d^2\psi}{dt^2} = mgl\theta + b_1 \frac{d\theta}{dt}$$

$$mrl\frac{d^2\theta}{dt^2}+(J+mr^2)\frac{d^2\psi}{dt^2}=b_2\frac{d\psi}{dt}+\tau_m$$

که در آن m جرم وزنه، L طول پاندول، r شعاع دیسک که شامل وزنهٔ متصل نیز می‌باشد، d ضخامت دیسک، $J=\rho\pi dr^4/4$ اینرسی دیسک، b_1 ضریب اصطکاک اتصال مفصلی در پاندول، b_2 ضریب اصطکاک اتصال مفصلی در دیسک، τ_m گشتاور اعمال شده توسط موتور متصل به پایهٔ دیسک می باشد.

اگر مجموعه متغیرهای وابسته زیر را تعریف کنیم،

$$x_1(t)=\theta(t) \qquad x_3(t)=\frac{d\theta}{dt}$$

$$x_2(t)=\psi(t) \qquad x_4(t)=\frac{d\psi}{dt}$$

در این‌صورت معادلات حاکم بر سیستم به فرم ماتریسی مطابق زیر خواهند بود:

$$M\dot{x}=Qx+W$$

که در آن :

$$M=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&ml^2&mlr\\0&0&mlr&J+mr^2\end{bmatrix} \qquad Q=\begin{bmatrix}0&0&1&0\\0&0&0&1\\mgl&0&b_1&0\\0&0&0&b_2\end{bmatrix}$$

$$W=\begin{bmatrix}0\\0\\0\\\tau_m\end{bmatrix} \qquad x=\begin{bmatrix}x_1\\x_2\\x_3\\x_4\end{bmatrix} \qquad \dot{x}=\begin{bmatrix}dx_1/dt\\dx_2/dt\\dx_3/dt\\dx_4/dt\end{bmatrix}$$

جهت بدست آوردن چهار معادله دیفرانسیل معمولی مرتبه اول که توسط تابع *ode45* مورد نیاز می‌باشد، باید این مجموعه معادلات را برای بدست آوردن dx_i/dt حل نماییم. بنابراین،

$$\dot{x}=M^{-1}Qx+M^{-1}W$$

که زمانی عبارت فوق دارای معنی می‌باشد که دترمینان M مخالف صفر باشد. در این مورد $det(M) = ml^2 J \neq 0$ می‌باشد.

ابتدا تابع زیر را جهت تعیین طرف راست چهار معادلهٔ دیفرانسیل معمولی مرتبهٔ اول مطابق زیر ایجاد می‌کنیم.

```
function p=InvPend(t,x,flag,taum,m,r,L,d,g,rho,b1,b2)
J=0.25*pi*rho*d*r^4;
M=[1 0 0 0; 0 1 0 0; 0 0 m*L^2  m*r*L; 0 0 m*r*L  J+m*R^2];
 Q=[0 0 1 0 ; 0 0 0 1; m*g*L 0 b1 0; 0 0 0 b2];
W=[0;0;0;taum];
p=inv(M)*Q*x+inv(M)*W;
```

که در آن p، برداری ستونی می‌باشد و کاربر بایستی از تابع *InvPend* جهت یافتن پاسخ سیستم در برابر گشتاورهای ناگهانی وارد بر مکانیزم و یا مجموعه‌ای متشکل از شرایط اولیه، استفاده کند. بدبختانه این سیستم ناپایدار می باشد و حل تحلیلی این مسئله منجر به اطلاعات مفیدی نخواهد شد. به علاوه این حل به سرعت منطقه‌ای که در آن فرضهای خطی سازی معادلات معتبر می باشند، را ترک خواهد کرد. به هر حال، این سیستم را می توان با استفاده از یک کنترلر مناسب همانگونه که در بخش ۱۰ –۵ –۳ شرح داده شده است، پایدار نمود.

مثال ۵–۳ شرایط مرزی تعیین شده در دو انتهای بازهٔ عمل

معادلهٔ دیفرانسیل معمولی زیر را در نظر بگیرید؛

$$\frac{d^2y}{dx^2} + ky = x \qquad 0 \le x \le 1$$

که دارای شرایط مرزی زیر می‌باشد:

$y(0)=0 \qquad y(1)=0$

از آنجا که تابع *ode 45*، تنها نیازمند تعیین شرایط مرزی (اولیه) در $x=0$ می‌باشد، باید از روشی جهت تعیین مقدار *dy(0)/dx*، که به ازای آن $y(1) \to 0$ میل کند، استفاده کنیم. این روند نیازمند یک

فرایند تکراری می‌باشد که در آن مقدار *dy(0)/dx*، آنقدر تغییر می‌کند تا مقدار *y(1)* به اندازهٔ کافی به صفر نزدیک شود[۱].

از آنجا که تنها دارای یک مجهول، *dy(0)/dx*، می‌باشیم، برای یافتن این مقدار می‌توان از تابع *fzero* استفاده نمود. برای معادلات دیفرانسیل مرتبه‌های بالاتر از دستور *fsolve* بجای *fzero* استفاده می‌کنیم.

بخشهای ۸ -۲- ۱ و ۱۲ -۳ -۱ را در این زمینه مشاهده کنید.

در ابتدا معادله را به یک جفت معادلهٔ دیفرانسیل مرتبهٔ اول، همانگونه که قبلاً نشان داده شد، تبدیل می‌کنیم، در این صورت دو معادله، به صورت زیر خواهیم داشت:

$$\frac{dy_1}{dx} = y_2$$

$$\frac{dy_2}{dx} = x - ky_1$$

تابع نشان‌دهندهٔ این معادلات در قالبی که توسط دستور *ode 45* مورد نیاز است، به صورت زیر می‌باشد:

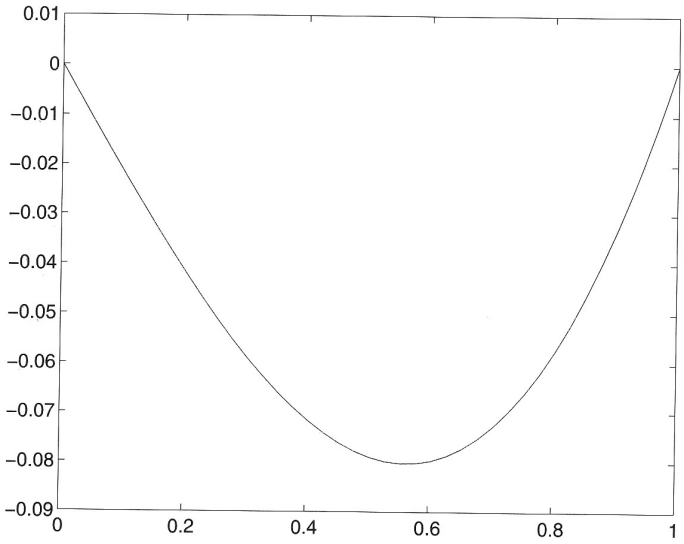
```
function f=ExampleODE(x,y,flag,k)
f=[y(2);x-k*y(1)];
```

سپس تابعی را جهت استفاده توسط تابع *fzero* برای یافتن مقدار *dy(0)/dx*، مطابق زیر ایجاد می‌کنیم.

```
function z=Slope(s,k)
[x,y]=ode45('ExampleODE',[0 1],[0 s]',[],k);
z= y(end,1);
```

که در آن *y(end,1)=y(1)* و *s=dy(0)/dx* می‌باشد. توجه کنید که *fzero* مقدار *s* را آنقدر تغییر خواهد داد تا $z \to 0$ میل کند.

۴- این روش که مقادیر مرزی را به مقادیر اولیه تبدیل می کند، روش شکار نامیده می شود.



شکل ۵-۷) منحنی $y(x)$ برای مثال ۵ - ۳

برنامه جهت تعیین $y(x)$ هنگامیکه $k=2$ می باشد به صورت زیر خواهد بود:

```
options= optimset('display','off');
k=2; guess= -0.1;
bc=fzero('Slope'guess,options,k);
[x,y]=ode45('ExampleODE',[0 1],[0 bc]',[],k);
disp(['Slope at x=0 is ' num2str(bc)])
plot(x,y(:,1))
```

که پس از اجرا خروجی زیر را در پنجرهٔ فرمان مطلب نمایش خواهد داد و نتایج را در شکل ۵-۷ نشان می‌دهد. این مطلب باید دانسته شود، که در برنامه $bc=y(1,2)$ می‌باشد. بعنوان کاربرد دیگری از این روش، بخش ۱۱ -۳ -۲ را مشاهده کنید.

```
Slop at x=0 is -0.21586
```

۵-۵-۵ حل های عددی معادلات غیر خطی - تابع *fsolve*

تابع داخلی *fsolve* که در جعبه ابزار بهینه سازی مطلب قراردارد، حل عددی یک سیستم شامل n معادلهٔ غیر خطی $f_n(x_1,x_2,...,x_n)=0$ که در آن x_n ها مجهولات دستگاه می‌باشند را با استفاده

از یک حدس اولیهٔ $x_s = [x_{s1}\ x_{s2} \ldots x_{sn}]$ ارائه می‌دهد. این تابع همچنین می‌تواند پارامترهای p_j را به تابع تعریف کنندهٔ $f_n(x)$ انتقال دهد. شکل کلی دستور *fsolve* به صورت زیر می‌باشد:

```
fsolve(FunctionName,xs,options,p1,p2,….)
```

که در آن *FunctionName*، نام فایل تابع در میان علامت کوتیشن و بدون پسوند ".m" و یا هنگامیکه توسط دستور *inline* ایجاد می‌شود، نام متغیر تابع بدون استفاده از کوتیشن می‌باشد، *options*، یک بردار اختیاری است که پارامترهایش توسط دستور *optimset* تعیین می‌شوند (به فایل کمکی تابع *optimset* رجوع شود) و p_1 ، p_2 و ... پارامترهای p_j می‌باشند.

اولین خط غیر دستوری این تابع به شکل زیر خواهد بود:

```
function z= FunctionName(x,p1,p2,…)
Z=[f1;f2;…;fn];
```

که در آن x برداری شامل n کمیت است که باید توسط مقادیر x_n ، تعیین شود و Z برداری ستونی مشتمل بر n عبارت مطلب، مبین n معادله غیر خطی $f_n(x_1,x_2,...,x_n)=0$ بر حسب عناصر x می‌باشد.

دستگاه معادلات زیر که از یک مرحلهٔ میانی تحلیل مکانیزم سه درجهٔ آزادی نشان داده شده در شکل ۲-۷ بدست آمده است را در نظر بگیرید:

$$r_1 - a_1 cos(\theta_1) - a_2 cos(\theta_1 + \theta_2) = 0$$

$$r_2 - a_1 sin(\theta_1) - a_2 sin(\theta_1 + \theta_2) = 0$$

جهت حل این دستگاه معادلات، ابتدا تابع *kinematics*، را تشکیل می‌دهیم، که معادلات فوق را به شکلی که توسط تابع *fsolve* مورد نیاز است تبدیل می‌کند، بنابراین:

```
function  w = kinematics(theta,a1,a2,r1,r2)
w = [a1*cos(theta(1)) + a2*cos(theta(1)+theta(2)) - r1;...
     a1*sin(theta(1)) + a2*sin(theta(1)+theta(2)) - r2];
```

که در آن $theta(1) = \theta_1$ و $theta(2) = \theta_2$، می‌باشد.

حال $r_1 = 1.8$، $r_2 = 2.1$، $a_1 = 1$ و $a_2 = 2$ و حدسهای اولیه برای θ_1 و θ_2 را برابر $\pi/6$ قرار می‌دهیم. در این صورت برنامه به صورت زیر خواهد بود:

```
options= optimset('display','off');
z=fsolve(' kinematics ',[pi/6  pi/6],options,1,2,1.8,2.1)*180/pi
```

که پس از اجرا نتایج $\theta_1 = z(1) = 16.6026^\circ$ و $\theta_2 = z(2) = 48.5095^\circ$ را به همراه خواهد داشت. مجموعهٔ دیگری از زوایا را می‌توان با قراردادن مقادیر اولیه به صورت، $\theta_1 = \theta_2 = \pi$ یافت. بنابراین، تابع *fsolve* را با احتیاط باید استفاده نمود بخصوص در مواردی که دستگاه دارای بیشتر از یک جواب می‌باشد.

به عنوان مثال دیگری از دستگاه معادلات چند مجهولی، به محل تلاقی بیضی با معادلهٔ

$$g(x,y) = x^2/4 + y^2 - 1$$

با سهمی به معادلهٔ :

$$f(x,y) = y - 4x^2 + 3$$

توجه کنید. نمودار این دو تابع مبین این است که آنها در چهار نقطه یکدیگر را قطع می‌کنند.

بنابراین مقداری که توسط تابع *fsolve* بدست می‌آید، حساس به حدس اولیه خواهد بود. این مسئله را با ایجاد تابع زیر، نشان خواهیم داد.

```
function w=fgsolve(xy)
w=[0.25*xy(1).^2+xy(2).^2-1;xy(2)-4*xy(1).^2+3];
```

که در آن x = *xy(1)* و y=*xy(2)* می‌باشد. برنامهٔ تعیین حل با حدس اولیهٔ $x = 0.5$ و $y = -0.5$ به صورت زیر می‌باشد:

```
 options= optimset('display','off');
xy=fsolve('fgsolve',[0.5,-0.5])
```

که پس از اجرا $x = xy(1) = 0.7188$ و $y = xy(2) = -0.9332$ خواهند بود. اگر حدس اولیه را به صورت $x = -0.5$ و $y = 0.5$ انتخاب می‌کردیم، نتایج $x=xy(1)=-0.9837$ و $y=xy(2)=0.870$ را بدست می‌آوردیم.

۵-۶ مثالهایی از چند تابع دیگر مطلب

تعداد زیادی از توابع دیگر در مطلب وجود دارند که برای بدست آوردن حل‌های عددی مسائل مهندسی بکار می‌روند، که چند تای آنها را در اینجا به همراه کاربردهایشان معرفی خواهیم کرد.

۵ - ۶ - ۱ برازش داده ها با چند جمله ایها - تابع *polyfit/polyval*

یکی از پارامترهای حساس در مورد فلزات، q، ضریب حساسیت به شکاف می‌باشد، که بر حسب ثابت نیوبر، $\sqrt{a}$ ،و شعاع شیار،r ،به شکل زیر محاسبه می‌شود:

$$q = \left(1 + \frac{\sqrt{a}}{\sqrt{r}} \right)^{-1}$$

مقدار $\sqrt{a}$ برای فلزات مختلف، متفاوت می‌باشد. و تابعی از مقاومت نهایی ماده، S_u ، می‌باشد. که آنرا می‌توان به وسیلهٔ برازش یک چند جمله‌ای، برای داده‌های بدست آمده از کار آزمایشگاهی بر حسب تابعی از S_u برای فلزی مشخص، محاسبه نمود. پس از بدست آوردن این چند جمله‌ای می‌توانیم مقدار q را برای هر r و S_u داده شده، تعیین کنیم.

جدول ۵-۱ ثوابت برای فولاد

$S_u(ksi)$	$\sqrt{a}\left(\sqrt{in}\right)$	$S_u(ksi)$	$\sqrt{a}\left(\sqrt{in}\right)$
50	0.130	170	0.028
70	0.092	190	0.020
90	0.072	210	0.015
110	0.057	230	0.010
130	0.046	250	0.007

150 0.037

داده‌های داده شده در جدول ۵-۱ را برای فولاد، در نظر بگیرید. با استفاده از این داده‌ها، ابتدا ضرایب یک چند جمله‌ای درجهٔ چهار را که مبین $\sqrt{a}$ می‌باشد، به صورت تابعی از S_u، تعیین می‌کنیم. وسپس از این چند جمله ای برای بدست آوردن q به ازای یک مقدار r و S_u معلوم، استفاده خواهیم کرد.

جهت برازش یک چند جمله ای با مجموعه ای از داده ها از تابع؛

```
polyfit
```

برای بدست آوردن ضرایب چند جمله ای استفاده خواهیم کرد و همچنین از تابع؛

```
polyval
```

جهت یافتن مقادیر تابع، بهره خواهیم برد. در حالت کلی یک چند جمله ای بصورت:

$$(۲-۵)\quad y(x) = p_1 x^n + p_2 x^{n-1} + \ldots + p_n x + p_{n+1}$$

می‌باشد. و دستور بدست آوردن ضرایب آن به صورت؛

```
p=polyfit(x,y,n)
```

می‌باشد، که در آن n شمارهٔ چند جمله‌ای، $p = [\, p_1\ p_2 \ldots\ p_n\ p_{n+1}\,]$ برداری به طول $n+1$ مبین ضرایب چند جمله‌ای در معادلهٔ (۵-۲)، و x و y هر یک بردارهایی به طول $m \geq n+1$ می‌باشند. همچنین داده‌هایی وجود دارند که چند جمله‌ایها بر اساس آنها برازش می‌شوند. که در آنها x ورودی و y خروجی می‌باشد.

جهت به دست آوردن معادلهٔ ۵-۲، هنگامیکه بردار p را داشته باشیم از دستور؛

```
y=polyval(p,x)
```

استفاده می‌کنیم. که در آن p برداری به طول $n+1$ که توسط تابع *polyfit* بدست آمده است و x کمیتی اسکالر و یا برداری متشکل از نقاطی که چند جمله‌ای بر اساس آنها باید محاسبه شود، می‌باشد. بطور کلی، مقادیر p در تابع *polyval* بطور قراردادی مشخص می‌شوند. در مورد جاری، علاقمند به استفاده از برداری هستیم که توسط تابع، *polyfit* محاسبه شده است.

جهت برازش داده‌های داده شده در جدول ۵-۱ و بدست آوردن مقداری برای $\sqrt{a}$در حوزهٔ $50 \le S_u \le 250$ و $0 < r < 0.2$ از برنامهٔ زیر استفاده می‌کنیم.

برای سادگی فرض می‌کنیم در هر دفعه یک مقدار S_u و r را وارد کنیم. به علاوه داده های جدول ۵-۱ را در تابعی با نام *NeuberData* قرار خواهیم داد. مجموعه‌ای از نمودارها که مقادیر q را بر حسب بازه ای از داده‌ها نمایش می دهد در شکل ٦-۱۹-ب نشان داده است.

تابع داده ها به صورت زیر می باشد:

```
function   nd = NeuberData
nd = [ 50,0.13;70,0.092;90,0.072;110,0.057;130,0.046;150,0.037;...
      170,0.028;190,0.020;210,0.015;230,0.010;250,0.007 ];
```

که در آن $nd(:,1) = Su$ و $nd(:,2) = r$ می‌باشد. برنامهٔ آن مطابق زیر است:

```
ncs = NeuberData
p = polyfit( ncs(:,1),ncs(:,2),4 );
r = input(' Enter notch radius (0 < r < 0.2)' );
Su = input(' Enter ultimate strength of  material ( 50 < Su < 250)' );
q = 1/(1+ polyval( p,Su ) / sqrt( r ));
disp([' Notch sensitivity =' num2str( q )])
```

که پس از اجرا نتیجهٔ زیر را در بر خواهد داشت:

```
Enter notch radius (0 < r < 0.2)  0.1
Enter ultimate strength of  material ( 50 < Su < 250)   135
Notch sensitivity = 0.87999
```

که برنامه پس از اجرا، خروجی‌های فوق را به ترتیب نشان می‌دهد و کاربر اعداد 1 و 135 را به ترتیب، پس از اینکه هر خط نشان داده شد، وارد می‌کند. سپس سیستم مقدار q را محاسبه کرده و در خط سوم نمایش می‌دهد.

۵ – ۶ – ۲ میانیابی داده ها ـ تابع *interp1*

اجازه دهید تا به حل عددی معادلهٔ ۵–۱ با ورودی پله که در شکل ۵–٤ نشان داده شد، بازگردیم. در اینجا علاقمند به تعیین مقدار درصد فراجهش خروجی سیستم یعنی $p_{overshoot}$ و زمان صعود مربوطه یعنی t_r می‌باشیم. زمان صعود به زمانی اطلاق می‌شود که خروجی سیستم از 10% تا 90% پاسخ حالت پایدارش می‌رسد، که توسط y_{ss} نشان داده می‌شود. هنگامیکه $\xi > 0$ باشد و هنگامیکه ورودی سیستم یک تابع پله باشد، داریم:

$$t_r = t_h - t_l$$

که در آن t_h و t_l به ترتیب از روابط زیر تعیین می‌شوند.

$$y(t_h) = 0.9 y_{ss}$$

$$y(t_l) = 0.1 y_{ss}$$

در صد فراجهش به صورت زیر تعریف می‌شود. اگر ماکزیمم مقدار پاسخ خروجی y_{max} و مقدار حالت پایدار سیستم y_{ss} باشد، در اینصورت داریم:

$$p_{overshoot} = 100(y_{max} - y_{ss}) / y_{ss}$$

از آنجا که $y(t)$ به صورت عددی بدست آمده است، از توابع *fzero* و *fminbnd* نمی‌توان استفاده کرد. یک ابزار جهت غلبه بر این محدودیت در بخش ۵–٦–۳ داده شده است. بنابراین جهت یافتن این کمیت‌ها از تابع :

```
interp1
```

برای تعیین t_h و t_l و از تابع :

```
max
```

جهت تعیین موقعیت y_{max} استفاده می‌شود. حالت کلی دستور *interp1* به صورت؛

```
V=interp1(u,v,U)
```

می‌باشد. که در آن v همان $v(u)$، u و v بردارهایی با طول یکسان، و U کمیتی اسکالر و یا برداری شامل مقادیر u، هنگامیکه V مورد نیاز باشد، می باشد V دارای طول مشابه U می‌باشد. بنابراین برنامه زیر :

```
[t,y]=ode45('ForcingFunction',[0 35],[0 0]',[],0.15,1);
[ymax,tmax]=max(y(:,1));
tr=interp1(y(1:tmax),t(1:tmax),[0.1 0.9]);
povershoot=100*(ymax-y(end,1))/y(end,1);
disp(['Percentage overshoot=' num2str(povershoot,4)'%'])
disp(['Rise time=' num2str(tr(2)-tr(1),4)'second'])
```

پس از اجرا نتیجهٔ زیر را در پنجرهٔ فرمان مطلب نمایش خواهد داد.

```
Percentage overshoot=61.23%

Rise time=1.153 seconds
```

خط اول برنامه در بخش ٥-٥-٤ مورد بررسی قرار گرفت. بیاد آورید که y یک بردار دو ستونی می‌باشد، که اولین ستونش متناظر با $y(t)$ و ستون دوم آن متناظر با dy/dt می‌باشد. خط دوم شکل دیگری از ورودی برای تابع *max* می‌باشد، که متناظر با شکل سوم در بخش ٥-٢ می‌باشد. (فایل کمکی تابع *max* را ببینید.) دومین خروجی تابع، یعنی *tmax*، اندیس آرایهٔ $y(:,1)$، در هنگامیکه ماکزیمم مقدار $y(:,1)$ اتفاق می‌افتد، می‌باشد. در این مورد، ماکزیمم مقدار عنصر 39ام آرایه می‌باشد ($tmax=39$). بنابراین $y(39,1)=y_{max}=1.6207$ خواهد بود. تابع *interp1* نیاز دارد که اولین آرگومانش یکنواخت باشد. بنابراین، باید ارایه $y(:,1)$ را در مقدار ماکزیمم‌اش *(index=tmax)* خاتمه دهیم، زیرا پس از این نقطه، $y(:,1)$ شروع به کاهش می‌کند.

٥ - ٦ - ٣ برازش داده ها با تابع درجهٔ سه - تابع *spline*

در این بخش ابتدا تعدادی دادهٔ نوسانی ومیرای نمایی تولید کرده و سپس این داده‌ها را با یک سری منحنی های درجهٔ سه توسط تابع :

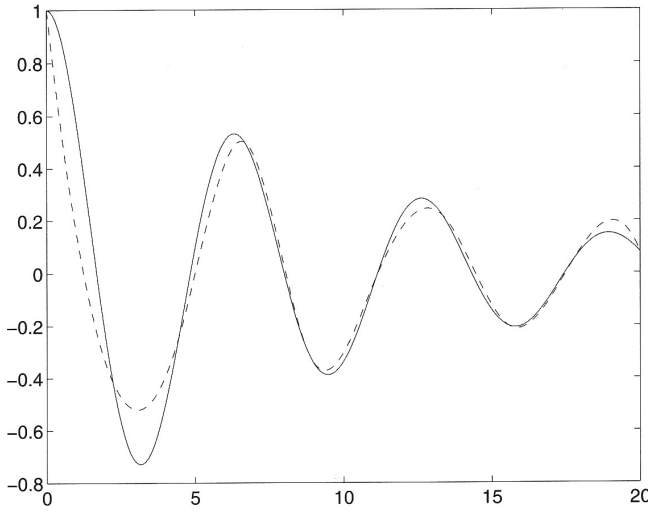
```
spline
```

برازش می‌کنیم. همچنین پریود یکی از نوساناتش را تعیین کرده و میرایی آنرا توسط روش کاهۀ لگاریتمی محاسبه می‌کنیم.

شکل کلی تابع *spline* به صورت زیر می‌باشد.

```
Y=spline(x,y,X)
```

که در آن y همان $y(x)$ و x و y بردارهایی با طول یکسان که جهت ایجاد رابطهٔ تابعی $y(x)$ به کار می‌روند، می‌باشند، و X یک اسکالر و یا بردار، هنگامی که $Y = y(X)$ مورد نیاز است، می‌باشد.



شکل ۵-۸) مقایسهٔ یک موج سینوسی میرا (خط پیوسته) با یک منحنی تقریبی (خط چین) که توسط یک تابع اسپیلاین با استفاده از 10 نقطه با فاصلهٔ یکسان در بازهٔ $0 \le \tau \le 20$ بدست آمده است.

داده ها توسط مقداری در تابع[۱]. زیر برای بازه ای از مقادیر بی بعد زمان، τ، به ازای یک مقدار مشخص میرایی، $\xi < 1.0$، ایجاد می‌شوند.

$$f(\tau,\xi) = \frac{e^{-\xi\tau}}{\cos\alpha}\cos\left(\tau\sqrt{1-\xi^2}+\alpha\right) \qquad (۵-۳)$$

که در آن :

$$\alpha = \tan^{-1}\frac{-\xi}{\sqrt{1-\xi^2}}$$

۱- برای مثال بـه کتـاب ارتعاشـات مکـانیکی *S.S.Rao,Addison -Wesley,reading,MA,1986,pp.80 -82* رجوع شود.

می‌باشد. اکنون این معادله را توسط تابعی با نام *DampedSineWave* مطابق زیر مقداریابی می‌کنیم:

```
function  f = DampedSineWave (tau,xi)
alpha=atan(-xi/sqrt(1-xi^2));
f=exp(-xi*tau).*cos(tau*sqrt(1-xi^2)+alpha)/cos(alpha);
```

اجازه دهید 10 نقطه با فاصلهٔ یکسان برای تابع $f(\tau,\xi)$ در بازهٔ $0 \le \tau \le 20$ در نظر بگیریم و چند جمله‌ای قطعه‌ای حاصل را به همراه موج اصلی به ازای $\xi = 0.1$ رسم کنیم. برنامه به صورت زیر می‌باشد؛

```
n=10; xi=0.1;
tau=linspace(0,20,n);
data= DampedSineWave (tau,xi);
newdata=linspace(0,20,200);
plot(newdata,spline(tau,data,newdata),'k--',newdata,…
              DampedSineWave (newdata,xi),'k-')
```

که پس از اجرا، شکل ۵-۸ را ایجاد خواهد کرد. این مسئله به وضوح دیده می شود که نتایج نسبتاً خوب می باشند. منحنی های فوق هنگامیکه تعداد نقاط نمونه به 15 افزایش یابد، کاملاً بر یکدیگر منطبق خواهند شد.

به هر حال، در مرحله بعدی آزمایش این داده های برازش شده، از مقدار $n=40$ برای دستیابی به تطابق عددی عالی میان مقادیر تخمین زده شده توسط کاربر و مقادیر دقیق، استفاده خواهیم کرد. در مورد جزئیات دستور *plot* در بخش ٦ -٢ توضیحاتی آورده شده است.

کاهۀ لگاریتمی مبین میزان کاهش دامنه ارتعاشات آزاد میرا می‌باشد و به صورت زیر تعریف می‌شود[1]:

$$\Delta = ln\frac{x_1}{x_2} = \frac{2\pi\xi}{\sqrt{1-\xi^2}}$$

[1] *S.S.Rao,ibid.*

که در آن؛

$$x_1 = f(\tau_1, \xi)$$

$$x_2 = f(\tau_1 + T, \xi)$$

توسط رابطهٔ ٥-۲ بدست می‌آیند و T، پریود ارتعاشات میرای سیستم می‌باشد. جهت تعیین ξ پس از تعیین x_1 و x_2 از روند زیر استفاده خواهیم کرد. در اینجا 40 نقطهٔ نمونه با فاصله یکسان را توسط تابع *spline* در بازهٔ $0 \le \tau \le 20$ برازش خواهیم کرد. سپس برداری متشکل از 200 دادهٔ زمانی در این بازهٔ ایجاد کرده و از آنها جهت محاسبهٔ مقادیر موج سینوسی میرا شده توسط تابع برازش شده بهره خواهیم برد. از بردار به دست آمده می‌توانیم اندیس مقداری که مقدار مینیمم تقریبی تابع $f(\tau, \xi)$ در آن رخ می‌دهد را بیابیم. سپس دو مرتبه از این مقدار مینیمم به عنوان حد بالای تخمین در تابع *fminbnd*، که به طور دقیق تر زمانی که مقدار مینیمم در آن رخ می دهد را تخمین می زند، استفاده می کنیم. که این زمان مینیمم همان زمان نوسان،T خواهد بود. سپس نسبت لگاریتمی دامنه‌ها در $0.05T$ و $1.05T$ را محاسبه می‌کنیم که همان مقدار Δ خواهد بود. و از دستور *fzero* جهت حل معادله فوق برای یافتن مقدار ξ استفاده خواهیم کرد. برنامه‌ای که این عملیات را انجام می‌دهد مطابق زیر خواهد بود؛

```
n=40; xi=0.1;
tau=linspace(0,20,n);
data= DampedSineWave (tau,xi);
tx=linspace(0,20,200);
datafit=inline('spline(tau,data,tx)','tx','tau','data');
[datamin imin]=min(datafit(tx,tau,data));
options= optimset('display','off');
periodT=2*fminbnd(datafit,0,2*tx(imin),options,tau,data);
delta=log(datafit(0.05*periodT,tau,data)/datafit(1.05*periodT,tau,da
ta));
logdec=inline('s/sqrt(1-s^2)-d/2/pi','s','d');
xiEst=fzero(logdec,[0.01.999],options,delta);
dif=100*(xiEst-xi)/xi;
disp(['Estimated xi='num2str(xiEst)'  Exact xi='num2str(xi)…
                  'Difference =' num2str(dif) '%'])
```

اجرای این برنامه برای $\xi = 0.1$ خروجی زیر را در پنجرهٔ فرمان مطلب نمایش خواهد داد؛

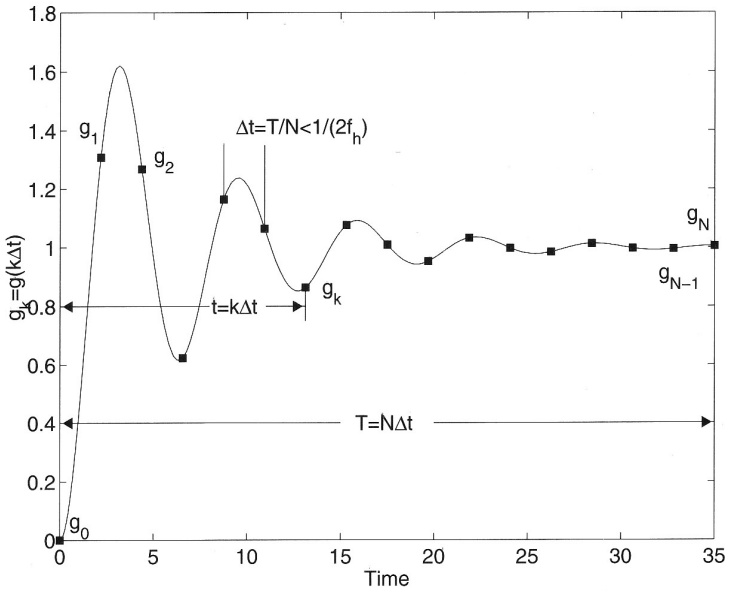
```
Estimated xi=0.10006  Exact xi=0.1  Difference=0.063776%
```

اگر $\xi = 0.95$ باشد، در اینصورت خروجی زیر را خواهیم داشت:

```
Estimated xi=0.94383  Exact xi=0.95  Difference=-0.64952%
```

با قرار دادن $n=100$ می‌توانیم خطا ها را در مورد اخیر به -0.26% کاهش دهیم.

۵ – ۶ – ۴ پردازش سیگنال دیجیتال – توابع *fft* و *ifft*



شکل ۵–۹) شکل موج مثال زده شده

تبدیل فوریهٔ گسسته : تبدیل فوریهٔ تابع حقیقی $g(t)$ که با فواصل Δt در بازهٔ $0 \le t \le T$ ، مقدار دهی می‌شود. می‌تواند توسط تبدیل فوریهٔ گسستهٔ زیر تقریب زده شود؛

$$G_n = G(n\Delta f) = \Delta t \sum_{k=0}^{N-1} g_k e^{-j2\pi nk/N} \qquad n = 0,1,...,N-1$$

که در آن $g_k = g(k\Delta t)$ ، $\Delta f = 1/T$ ، $T = N\Delta t$ و N تعداد نمونه‌ها می‌باشد

به شکل ۵-۹ رجوع کنید. به طور کلی، G_n یک کمیت مرکب می‌باشد. محدودیت موجود در مورد Δt بصورت؛

$$\alpha\Delta t \prec \frac{1}{f_h}$$

می‌باشد، که در آن f_h بالاترین فرکانس $g(t)$ ، هنگامیکه $\alpha \geq 2$ است، می‌باشد. کمیت G_n دامنهٔ چگالی $g(t)$ می‌باشد و دارای واحد دامنه - ثانیه و یا دامنه بر هرتز می‌باشد.

تبدیل فوریهٔ معکوس توسط رابطهٔ ؛

$$g_k = \Delta f \sum_{n=0}^{N-1} G_n e^{j2\pi nk/N} \qquad k = 0,1,...,N-1$$

محاسبه می‌شود. جهت تخمین بزرگی دامنهٔ A_n ، متناظر با مقادیر G_n در فرکانسهای متناظر $n\Delta f$، کاربر می‌تواند G_n را در Δf ضرب کند. بنابراین:

$$A_n = \Delta f G_n$$

و لذا داریم:

$$A_n = \frac{1}{N} \sum_{k=0}^{N-1} g_k e^{-j2\pi nk/N} \qquad n = 0,1,...,N-1$$

که در آن $\Delta f \Delta t = 1/N$ قرار داده شده است. متوسط توان سیگنال به صورت :

$$p_{ave} = \sum_{n=0}^{N-1} |A_n|^2$$

خواهد بود. همچنین کاربر می‌تواند $|A_n|$ را به صورت تابعی از $n\Delta f$ جهت بدست آوردن نمودار پاسخ فرکانسی ترسیم نماید. در این مورد می دانیم که[۱]:

$$|A_n|_s = 2|A_n| \qquad n = 0,1,...,N/2-1$$

۱- برای نمونه به مثال زیر رجوع کنید.

J.S.Bendat and A. G. PIERSOL, Engineering Applications of Correlation and Spectral Analysis, John Wiley &Sons, New York، 1980.

این عبارات می‌توانند به بهترین نحو توسط تبدیل فوریهٔ سریع، ارزیابی شوند. که یک الگوریتم بسیار کارآمد برای ارزیابی تبدیل فوریهٔ گسسته می‌باشد.

این روش به خصوص هنگامیکه تعداد داده های نمونه، به صورت توانی از 2 می‌باشد، یعنی $N = 2^m$ که m عدد حقیقی مثبتی می‌باشد، کار آمدتر خواهد بود. مطلب این الگوریتم را توسط تابع؛

```
fft(g,N)
```

کامل می کند و معکوس آنرا توسط تابع؛

```
Ifft(g,N)
```

انجام می‌دهد. که در آن تابع *fft*مقدار $G_n/\Delta t$ و *ifft* مقدار $g_k/\Delta f$ را باز می‌گرداند.

توابع وزنی : موقعیتهای گوناگونی وجود دارد که وزن دادن به تابع $g(t)$ توسط یک تابع مناسب، جهت ایجاد دقت بیشتر و سایر خواص، در حوزهٔ تبدیل فرکانسی مطلوب می‌باشد. روند کار به اینصورت است که سیگنال مرجع، قبل از اجرای تبدیل فوریهٔ گسسته، اصلاح می‌شود. در چنین روشی تأثیر تغییرات ایجاد شده با باز کردن تابع، روی مقدار متوسط سیگنال و توان متوسط سیگنال، از بین خواهد رفت. بنابراین اگر مقادیر تعیین شده برای تابع وزنی، $w_n = w(n\Delta t)$ باشند، در اینصورت سیگنال تصحیح شدهٔ g_{cn} توسط رابطه زیر[1]:

$$g_{cn} = k_2 w_n (g_n - k_1) \qquad n = 0,1,\ldots,N-1$$

محاسبه می شود، که در آن؛

$$k_1 = \sum_{n=0}^{N-1} w_n g_n \Big/ \sum_{n=0}^{N-1} w_n$$

مقدار متوسط تابع و؛

$$k_2 = \left[N \Big/ \sum_{n=0}^{N-1} {w_n}^2 \right]^{1/2}$$

---

[1] *E. C. Ifeachor and B. W. Jervis, Digital Signal Processing: A Practical Approach, Addison-Wesley, Harlow, England, 1993, p. 593.*

مقدار متوسط توان تابع را تصحیح می‌کند. سپس کاربر می‌تواند تبدیل فوریهٔ گسستهٔ g_{cn} را محاسبه کند.

جعبه ابزار پردازش سیگنال دیجیتال مطلب شامل هشت تابع وزنی متداول می‌باشد.

همبستگی عرضی *(cross correlation)* : همبستگی عرضی دو سیگنال قابل محاسبه $x(t)$ و $y(t)$ با زمان محدود توسط رابطهٔ ؛

$$R_{xy}(\tau)=\int_{-\infty}^{\infty}x(t)y(t+\tau)dt \qquad -\infty \prec \tau \prec \infty$$

محاسبه می‌شود. اکنون می‌خواهیم به عنوان تمرین مقدار این عبارت را با استفاده از تبدیل فوریهٔ معکوس تابع چگالی $S_{xy}(w)$ محاسبه می‌کنیم، بنابراین:

$$R_{xy}(\tau)=F^{-1}\left[S_{xy}(\omega)\right]$$

که در آن $F^{-1}(...)$ مبین تبدیل فوریهٔ معکوس و:

$$S_{xy}(\omega)=X(\omega)Y^{*}(\omega)$$

می‌باشد. که در آن مقادیر $X(\omega)$ و $Y(\omega)$، تبدیلات فوریهٔ توابع $x(t)$ و $y(t)$ می‌باشند و علامت ستاره مبین مزدوج مختلط می‌باشد. جهت تبدیل $R_{xy}(\tau)$ به واحدهای مناسبش باید $S_{xy}(\omega)$ را در $\Delta t=T/N$ ضرب کنیم.

این روابط را بوسیلهٔ دو مثال زیر نشان خواهیم داد.

مثال ۵ -۴ تبدیل فوریهٔ موج سینوسی

فرض کنید:

$$g(t)=A_0\sin(2\pi f_0 t) \qquad 0\le t\le T=2^K/f_0 \qquad K=0,1,2,...$$

باشد بنابراین:

$$m-K\succ 1 \quad or \quad \Delta t\prec 1/(2f_0)$$

و چون ؛

$$f_h = f_0 = 2^K / T$$

و

$$\Delta t = 2^{-m} T$$

می‌باشد. فرض می‌کنیم که تابع $g(t)$ توسط تابع همینگ *(Hamming)* مطابق زیر وزن دهی شده است:

$$w(t) = 0.54 - 0.46\cos(2\pi t/T) \qquad 0 \le t \le T$$
$$= 0 \quad otherwise$$

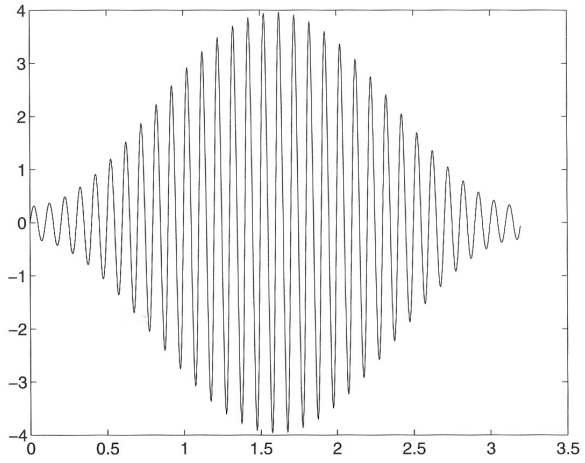
برنامه و جهت محاسبه و رسم سیگنال وزن دهی اصلاح شده $g_c(t)$ ودامنه A_n و همچنین نمایش توان متوسط سیگنال که بصورت $p_{ave} = A_0^{\,2}/2$ می‌باشد، مطابق زیر است. فرض می‌کنیم که $A_0 = 2.5$ ، $f_0 = 10\ Hz$ ، $k = 5$ و $m = 10$ $(n = 1024)$ باشد.

```
k=5;m=10;fo=10;Ao=2.5;
N=2^m; T=2^k/fo;
ts=(0:N-1)*T/N;
df=(0:N/2-1)/T;
whamm=0.54-0.46*cos(2*pi*ts/T);
sampledSignal=Ao*sin(2*pi*fo*ts);
k1=sum(whamm.*SampledSignal)/sum(whamm);
k2=sqrt(N/sum(whamm.^2));
correctedSignal=whamm.*(SampledSignal-k1)*k2;
figure(1)
plot(ts,CorrectedSignal,N))/N;
plot(df,2*An(1:N/2))
disp(['Average power='num2str(sum(An.^2))])
```

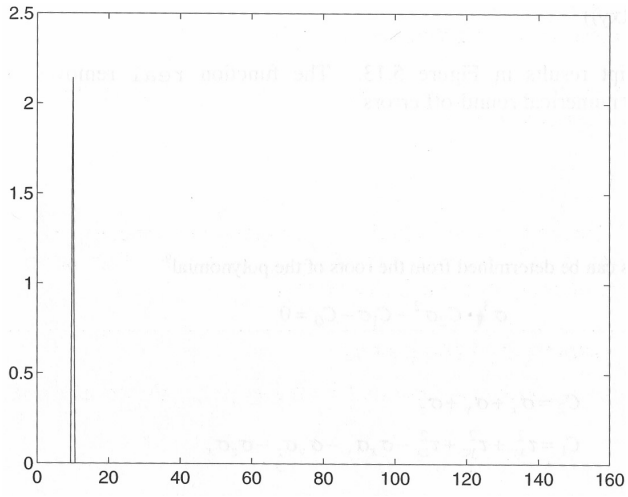
اجرای این برنامه منجربه اشکال ۵-۱۰ و۵-۱۱ می‌شود و پیغام زیر در پنجره فرمان مطلب نشان داده خواهد شد:

```
Average power =3.125
```

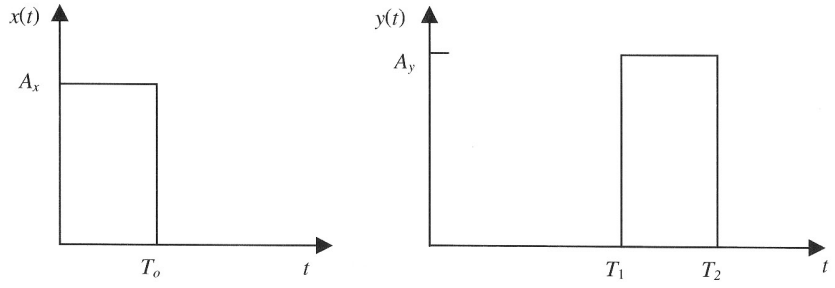
تابع مطلب *figure*، جهت فراهم کردن دو پنجرهٔ شکل مجزا استفاده می‌شود.



شکل ۵-۱۰) موج سینوسی اصلاح شده توسط تابع وزن دهی همینگ



شکل ۵- ۱۱) نمودار دامنهٔ یک موج سینوسی با استفاده از تابع وزن دهی همینگ



شکل ۵-۱۲) دو پالس نمونه

بخش ٦-١ را مشاهده کنید. توجه کنید که دامنه موج سینوسی مساوی 2.5 نمی باشد. اما این مقدار را می توان با حذف تابع وزنی بدست آورد.

مثال ۵-۵ همبستگی عرضی دو پالس

اکنون تابع همبستگی عرضی دو پالس نشان داده شده در شکل ۵-۱۲ را که به صورت زیر بیان می‌شوند، محاسبه خواهیم کرد.

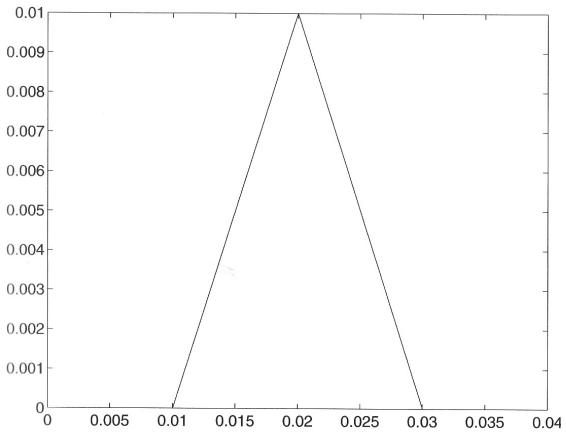
$$x(t) = A_x[u(t) - u(t - T_0)] \qquad t \geq 0$$

$$y(t) = A_y[u(t - T_1) - u(t - T_1 - T_2)] \qquad t \geq 0$$

که در آن $u(t)$ تابع پله واحد می‌باشد. در اینجا فرض می‌کنیم که $A_x = A_y = 1$ ، $T_0 = 0.01s$، $T_1 = 2T_0$، $T_2 = T_1 + T_0$، $T = T_2 + T_0$ و $N = 2^{10}$ باشد. در اینصورت برنامه، بصورت زیر خواهد بود:

```
To = 0.01;T1 = 2*T0;T2 = T1+To;Tend = T2+To;
N = 2^10;deltaT = Tend / N;amp1 = 1;
t = linspace(0,Tend,N);
PulseCrossCorr = inline('amp1*((t-Ts>=0)-(t-Te>0))','t','Ts','Te','amp1');
x = PulseCrossCorr(t,0,To,amp1);
y = PulseCrossCorr(r,T1,T2,amp1);
Y = conj(fft(y,N));
```

```
Rxy = ifft( X.*Y*deltaT,N );
plot(t,real(Rxy))
```



شکل ۵ – ۱۳) تابع همبستگی عرضی دو پالس با دوام یکسان

اجرای برنامهٔ فوق در شکل ۵-۱۳ نشان داده شده است. تابع *real* قسمتهای موهومی باقیماندهٔ مربوط به خطاهای عددی را حذف می‌کند.

تمرین ها

بخش ۵- ۵- ۱

۵-۱ تنشهای اصلی را می‌توان از طریق یافتن ریشه‌های چند جمله‌ای[۱]

$$\sigma^3 - C_2\sigma^2 - C_1\sigma - C_0 = 0$$

یافت، که در آن :

$$C_2 = \sigma_x + \sigma_y + \sigma_z$$

۹- برای نمونه به کتاب زیر رجوع کنید.

J. E. Shigley and C. R. Mischke, Mechanical Engineering Design, 5th ed., McGraw-Hill, New York, 1989.

$$C_1 = \tau_{xy}^{\ 2} + \tau_{yz}^{\ 2} + \tau_{zx}^{\ 2} - \sigma_x\sigma_y - \sigma_y\sigma_z - \sigma_z\sigma_x$$

$$C_0 = \sigma_x\sigma_y\sigma_z + 2\tau_{xy}\tau_{yz}\tau_{zx} - \sigma_x\tau_{yz}^{\ 2} - \sigma_y\tau_{zx}^{\ 2} - \sigma_z\tau_{xy}^{\ 2}$$

می‌باشد. و σ_x، σ_y و σ_z ، تنشهای عمودی اعمال شده و τ_{xy} ، τ_{yz} و τ_{zx} تنشهای برشی اعمال شده می‌باشند. اگر ریشه های معادله σ_1، σ_2 و σ_3 (سه تنش اصلی) باشند که $\sigma_1 > \sigma_2 > \sigma_3$ است، در این صورت تنشهای برشی اصلی به شکل زیر خواهند بود.

$$\tau_{12} = (\sigma_1 - \sigma_2)/2 \qquad \tau_{23} = (\sigma_2 - \sigma_3)/2 \qquad \tau_{13} = (\sigma_1 - \sigma_3)/2$$

که در آن $\tau_{max} = \tau_{13}$ می باشد.

تنشهای اصلی نرمال و تنشهای اصلی برشی متناظرشان را هنگامیکه ؛

$$\sigma_x = 100 \qquad \tau_{xy} = -40$$

$$\sigma_y = -60 \qquad \tau_{yz} = 50$$

$$\sigma_z = 80 \qquad \tau_{zx} = 70$$

می‌باشد، معین کنید. تابع ریشه یاب قادر به مرتب کردن ریشه ها نمی‌باشد. جهت انجام عملیات مرتب سازی از تابع *sort* به شیوه‌ای که در انتهای مثال ۲-۲ مربوط به بخش ۲-۵-٤ آورده شده، استفاده نمایید. {پاسخ: $\sigma_1 = 160.7444$، $\sigma_2 = 54.8980$، $\sigma_3 = -95.6424$ $\tau_{12} = 52.9232$، $\tau_{13} = 128.19349$، $\tau_{23} = 75.2702$.}

۲-۵ در معادلات زیر در مواردی که تعداد ریشه‌های مورد نیاز، در صورت مسئله تعیین نشده است، پنج ریشهٔ مثبت کوچکتر معادلات را با استفاده از تابع *fzero* تعیین کنید. از شکلی از تابع *fzero* استفاده کنید که تابع را ملزم به جستجو در یک بازهٔ محدود $[x_0 \quad x_1]$ نماید. هر تابع را قبل از یافتن ریشه ها ترسیم نمایید.

جهت وضوح بیشتر نمودارها از تابع *axis* استفاده کنید.

الف) معادلهٔ زیر در هنگام مطالعهٔ ارتعاشات طنابها بدست می‌آید[1]:

$$\tan x = x$$

[1] *E.B.Magrab vibration of Elastic Structural Members Sijthoff Noordhoff The Netherlands 1979 p*

ب) معادلهٔ زیر در هنگام مطالعهٔ جریان حرارت در ورقها بدست می‌آید[1]: ریشه‌های این معادله را برای دو حالت مجزای $p=0.1$ و $p=1$ تعیین کنید.

$$2\cot x = \frac{x}{p} - \frac{p}{x}$$

پ) معادلهٔ زیر[2] در هنگام ارتعاشات اعضای توخالی بدست می‌آید. فرض کنید $b=2$ باشد.

$$J_0(x)Y_0(xb) - J_0(xb)Y_0(x) = 0$$

از توابع *besselj* و *bessely* به ترتیب برای $J_0(x)$ و $Y_0(x)$ که توابع بسل نوع اول و دوم از مرتبهٔ صفر نامیده می‌شوند، استفاده کنید.

ت) معادلهٔ زیر[3] هنگام بررسی ارتعاشات یک تیر یک سرگیردار و یک سر آزاد که جرمی به اندازه M_0 را در انتهای آزادش تحمل می‌کند، بدست می‌آید. ریشه‌های آن را برای سه حالت $\frac{M_0}{m_0} = 0$، $\frac{M_0}{m_0} = 0.2$ و $\frac{M_0}{m_0} = 1$ بیابید.

$$(M_0/m_0)\Omega[\cos(\Omega)\sinh(\Omega) - \sin(\Omega)\cosh(\Omega)] + \cos(\Omega)\cosh(\Omega) + 1 = 0$$

ث) معادلهٔ زیر[4] هنگام بررسی ارتعاشات یک تیر که در یک انتها گیردار و در انتهای دیگر دارای تکیه‌گاه ساده می باشد، بدست می‌آید.

$$\tanh(\Omega) - \tan(\Omega) = 0$$

ج) معادلهٔ زیر[5] هنگام بررسی ارتعاشات یک صفحهٔ دایروی صلب که مرز خارجی‌اش کاملاً مقیود شده است، بدست می‌آید.

$$J_m(\Omega)I_{m+1}(\Omega) + I_m(\Omega)J_{m+1}(\Omega) = 0$$

که در آن $J_m(x)$، تابع بسل نوع اول از مرتبه m و $I_m(x)$ تابع بسل اصلاح شدهٔ نوع اول از مرتبهٔ m می باشد. از توابع *besselj* و *besseli* به ترتیب برای $J_m(x)$ و $I_m(x)$ استفاده کنید. سه

---

[1] *M,N.Ozisik Heat Conduction 2md ed. John Wiley Sons New York 1993 p. 47.*
[2] *E. B. Magrab,ibid., p. 83.*
[3] *E. B. Magrab,ibid., p. 130.*
[4] *E. B. Magrab,ibid., p. 130.*
[5] *E. B. Magrab,ibid., p. 252.*

ریشهٔ کوچکتر معادله را برای $m=0$، $m=1$ و $m=2$ بیابید، برنامه را جهت استفاده در تمرین ۷-٤ ذخیره نمایید.

چ) معادلهٔ زیر[1] هنگام تعیین مدهای متقارن صفحه ای یک کابل معلق استفاده می شود. کمترین ریشه را هنگامیکه $\lambda^2 = 2\pi^2$، $\lambda^2 = 4\pi^2$ و $\lambda^2 = 8\pi^2$ می باشد، بیابید. این حل باید ابتدا با رسم معادله آغاز شود. از دستور *axis* جهت محدود کردن محور قائم در بازهٔ -10 تا 20 استفاده کنید. بخش ٦ -۲ را ببینید.

$$\tan\Omega = \Omega - \frac{4\Omega^3}{\lambda^2}$$

ح) در هنگام بررسی جریان یکنواخت در یک کانال باز با سطح مقطع ذوزنقه‌ای نسبت عمق مایع به ارتفاع گرادیان انرژی، x، توسط رابطهٔ زیر[2] بدست می‌آید:

$$(1+c_0 x)^2\left(x^2 - x^3\right) = c_1$$

که در آن $0 \le C_0 \le 11$ و $0.005 \le C_1 \le 12.3$ توابعی از هندسه و نرخ جریان می‌باشند. به هر حال، همه ترکیبهای C_0 و C_1 مناسب نمی‌باشند. جفت مقادیر حقیقی x را در بازهٔ $[1 \quad 0]$، که این معادله را ارضا می‌کنند، برای؛

الف) $C_0 = 0.4$ و $C_1 = 0.2$

ب) $C_0 = 7.0$ و $C_1 = 4.0$

بیابید. از دو روش تابع *fzero* و *roots* استفاده کنید. جهت استفاده از تابع *roots* معادله به صورت زیر باز نویسی می‌شود :

$$-c_0{}^2 x^5 + \left(c_0{}^2 - 2c_0\right)x^4 + (2c_0 - 1)x^3 + x^2 - c_1 = 0$$

خ) زاویهٔ موج β $\left(0 < \beta \le \pi/2\right)$ مربوط به یک موج متلاطم در سطح مایع یک کانال باز که سرعت مایع از سرعت موج بیشتر است توسط رابطهٔ زیر[3] تعیین می‌شود :

---

[1] *M.Irvine, Cable Structures، Dover publications,Inc.,New York، 1981,p.95.*
[2] *H.W.King,Handbook of Hydraulics، 4th ed.,McGraw -Hill,New York، 1954,p.8-1.*
[3] *N. H. C. Hwang and C. E. Hita, Fundamentals of Hydraulic Engineering Systems، 2nd ed., Prentice Hall,Englewood Cliffs,NJ,1987,P.222.*

$$2N_F{}^2 \sin^2(\beta)\tan^2(\beta-\theta) = \tan(\beta)\tan(\beta-\theta) + \tan^2(\beta) \qquad \beta > \theta$$

که در آن θ تغییر شکل زاویه‌ای دیوار و $1 \le N_F \le 12$ عدد فراد می‌باشد. مقدار β را بر حسب درجه هنگامیکه $\theta = 35^\circ$ و $N_F = 5$ می‌باشد، تعیین کنید.

ابتدا تابع را رسم کنید.

د) نرخ داخلی بازگشت i_{rr} یک ابزار اندازه گیری حسابداری می‌باشد، که درصد تمایل بدست آمده از یک سرمایه گذاری متعادل را مشخص می‌کند. که توسط رابطهٔ[1]

$$\sum_{k=0}^{n} F_k (1+i_{rr})^{-k} = 0$$

محاسبه می شود. که در آن n تعداد بازه‌ها، i_{rr} نرخ داخلی می‌باشد که به صورت یک عدد اعشاری بیان می‌شود و F_k جریان پول در هر پریود می‌باشد.

جریان پول مثبت به معنای این است که پول دریافت می‌شود و جریان پول منفی به معنای اینست که پول تحویل می شود.

مقدار i_{rr} را هنگامی که $F_0 = -\$1000$، $F_1 = -\$800$، $F_2 = \$500$، $F_3 = \$500$، $F_4 = \$500$، $F_0 = \$1200$ می‌باشد بیابید.

ذ) اگر کسی مبلغ p را سرمایه گذاری کند و مبلغ A را در هر بازهٔ زمانی از این سرمایه گذاری سود ببرد. در اینصورت تعداد پریودهای n برای برگشت کامل سرمایهٔ p با یک نرخ سود i در هر بازه (بصورت یک تابع اعشاری بیان می شود.) توسط رابطهٔ زیر[2] تعیین می‌شود؛

$$\frac{A}{P} = \frac{i(1+i)^n}{(1+i)^n - 1}$$

اگر $i=12\%$ برای هر سال باشد و $A/P = 0.16$ باشد، در اینصورت n را که تعداد سالهای مورد نیاز برای بازگشت سرمایه می‌باشد را بیابید.

---

[1] *G. J.Theusen and W.J. Fabrycky,Engineering Economy, 8th ed.,prentice-Hall, Englewood Cliffs، NJ,1993,P. 176,.*

[2] *G.J.Theusen and W.J.Fabrycky, ibid.,p.188.*

ر) یک تخمین از پارامتر β که در تابع چگالی احتمال وایبول ظاهر می‌شود (بخش ۱۴ -۲ -۲ را ببینید.) نیازمند حل معادلهٔ زیر می باشد[1].

$$\beta = \left[\frac{\sum_{i=1}^{n} x_i^{\beta} \ln(x_i)}{\sum_{i=1}^{n} x_i^{\beta}} - \frac{1}{n} \sum_{i=1}^{n} \ln(x_i) \right]^{-1}$$

که در آن x_i نمونهٔ تصادفی با اندازهٔ n می باشد. اگر؛

```
x=[72 82 97 103 113 117 126 127 127 139 154 159 199 207]
```

باشد، در اینصورت مقدار β را تعیین کنید.

ز) در تعیین تنش برشی تماسی سطح بین یک کره و یک صفحه، که مدلی از تأثیرات متقابل یاتاقان در برابر سطح می باشد، مقدار نسبت x از رابطهٔ[2]

$$x \ln\left(\sqrt{x^2 - 1} + x\right) - \sqrt{x^2 - 1} - Cx = 0$$

تعیین می‌شود. که در آن $x > 1$ و $C < 1$ می‌باشد. برای $C = 0.5$ مقدار x را تعیین کنید.

۵ - ۳ سه ریشهٔ حقیقی معادلهٔ[3] ؛

$$x^4 = 2^x$$

را تعیین کنید. {راهنمایی: ابتدا تابع را روی دو بازهٔ متفاوت $-1 \le x \le 2$ و $2 \le x \le 17$ رسم کنید.}

۵ - ۴ فرمول محاسباتی برای معادلهٔ تعمیم یافتهٔ ضریب تراکم پذیری z یک گاز توسط رابطهٔ زیر داده می‌شود[1]:

---

[1] *D. C. Montgomery and G. C.Runger,Applied Statistics and Probability for Engineers، John Wiley & Sons، New York, 1994,p.299.*

[2] *W. Changsen, Analtsis of Rolling Element Bearings، Mechanical Engineering Publishers، London، 1991، p.80*

[3] *Problem suggested by Prof. Jeffery M. Cooper, Department of Mathematics, University of Maryland, College Park, MD.*

$$Z(r,\tau)=1+r\sum_{i=1}^{6}A_i\tau^{i-1}+r^2\sum_{i=7}^{10}A_i\tau^{i-1}+r^3\sum_{i=11}^{13}A_i\tau^{i-11}+r^4A_{14}\tau$$
$$+r^5\left(A_{15}\tau^2+A_{16}\tau^3\right)+r^6A_{17}\tau^2+r^7\left(A_{18}\tau+A_{19}\tau^3\right)+r^8A_{20}\tau^3$$
$$+r^2e^{-0.0588r^2}\left[A_{21}\tau^3+A_{22}\tau^4+r^2\left(A_{23}\tau^3+A_{24}\tau^5\right)+r^4\left(A_{25}\tau^3+A_{26}\tau^4\right)\right.$$
$$\left.+r^6\left(A_{27}\tau^3+A_{28}\tau^5\right)+r^8\left(A_{29}\tau^3+A_{30}\tau^4\right)+r^{10}\left(A_{31}\tau^3+A_{32}\tau^4+A_{33}\tau^5\right)\right]$$

که درآن $\tau=T_c/T$ $(0.4\le\tau\le 1)$، $r=RT_c/p_cv$، R ثابت گاز با دیمانسیون $(Mpa-m^3)/(kg-K)$، T درجه حرارت در مقیاس کلوین (k)، p فشار برحسب Mpa، v حجم مخصوص با دیمانسیون m^3/kg، T_c و p_c به ترتیب دما و فشار بحرانی می‌باشند و مقادیر 33 ثابت در جدول ۵ -۲ داده شده است.

الف) تابعی جهت تعیین *z(r,t)* ایجاد کنید. تابع خود را با استفاده از دستور *forma tlong e* با مقادیر زیر تست کنید.

```
(i )   Z(1,1)=0.70242396927
(ii)   Z(1/0.3,1)=0.29999999980
(iii) Z(2.5,0.5)=0.99221853928
```

این تابع و توابعی را که در قسمت (ب) و (پ) ایجاد می‌کنید جهت استفاده در تمرین ٦-١٢ ذخیره نمایید.

ب) کمیت فوق در فرمول زیر استفاده می‌شود؛

$$Z(r,\tau)=\frac{p\tau}{r}=\frac{Pv}{RT}$$

که درآن $p=P/P_c$ $(1\le P\le 6)$ می‌باشد. مقدار r و $Z(r,\tau)$ را با استفاده از معادلهٔ *(a)* برای

(i) $P=0.6$ و $\tau=1/1.05$

(ii) $P=2.18$ و $\tau=1/1.2$

تعیین کنید. {پاسخ: *(i)* $z=0.8013$ در $r=0.7131$ و *(ii)* $z=0.5412$ در $r=3.3567$.}

---

[1] *W. C. Reynolds, "Thermodynamic Properties in SI," Department of Mechanical Engineering, Stanford University, Stanford, CA, 1979.*

جدول ۵-۲ (ثوابت در فرمول تعمیم یافته Z)					
j	A_j	j	A_j	j	A_j
1	0.062432384	12	-0.000727155024313	23	-0.0845194493813
2	0.12721477	13	-0.00452454652610	24	-0.00340931311928
3	-0.93633233	14	0.00130468724100	25	-0.00195127049901
4	0.70184411	15	-0.000222165128409	26	$4.93899910978 \times 10^{-5}$
5	-0.35160896	16	-0.00198140535656	27	$-4.93264612930 \times 10^{-5}$
6	0.056450032	17	$5.97573972921 \times 10^{-5}$	28	$8.85666572382 \times 10^{-7}$
7	0.0299561469907	18	$-3.64135349702 \times 10^{-6}$	29	$5.34788029553 \times 10^{-8}$
8	-0.0318174367647	19	$8.41364845386 \times 10^{-6}$	30	$-5.93420559192 \times 10^{-8}$
9	-0.0168211055517	20	$-9.82868858822 \times 10^{-9}$	31	$-9.06813326929 \times 10^{-9}$
10	1.60204060081	21	-1.57683056810	32	$1.61822407265 \times 10^{-9}$
11	-0.00109996740746	22	0.0400728988908	33	$-3.32044793915 \times 10^{-10}$

پ) از معادلهٔ (a) جهت تعیین مقادیر τ و $Z(r,\tau)$، هنگامی که؛

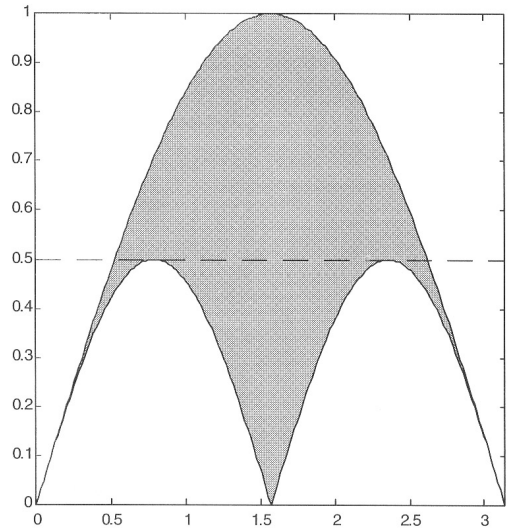
$(i)\ r = 1/1.4$ و $p = 0.6$

$(ii)\ r = 1/0.6$ و $p = 2.18$

می‌باشد استفاده کنید. {پاسخ: (i) در $\tau = 0.9532$ مقدار $z = 0.8007$ و (ii) در $\tau = 0.6505$ مقدار $z = 0.8508$ می باشد.}

۵ – ۵ افت فشار یک سیال روان در داخل لوله تابعی از ضریب اصطکاک لوله λ می باشد که می‌توان آنرا توسط فرمول کولبروک تخمین زد[1]،

[1] *N. H. C. Hwang and C. E. Hita, ibid., p.68.*



شکل ۵ – ١٤) شکل مربوط به تمرین ۵- ٦

$$\lambda = \left[-2\log_{10}\left(\frac{2.51}{R_e\sqrt{\lambda}} + \frac{0.27}{d/k} \right) \right]^{-2} \qquad R_e \geq 4000$$

که در آن Re عدد رینولدز، d قطر لوله، k زبری سطحی می‌باشد. برای لوله‌های صاف ($k \cong 0$ یا $d/k > 100000$) می‌باشد.

$$\lambda = \left[-2\log_{10}\left(\frac{R_e\sqrt{\lambda}}{2.51} \right) \right]^{-2} \qquad R_e \geq 4000$$

برای جریان متلاطم کاملاً توسعه یافته ضریب اصطکاک توسط رابطه زیر داده می‌شود:

$$\lambda = \left[2\log_{10}\left(3.7\frac{d}{k} \right) \right]^{-2}$$

که مستقل از Re (عدد رینولدز) می‌باشد. فرمول آخری یک مورد خاص از فرمول کلی کلبروک می‌باشد. و می‌تواند که جهت بدست آوردن حدس اولیه در مسائل کمّی و همانگونه که در بخش ۱۱ – ۲ –۳ گفته خواهد شد، بکار رود.

اگر مقدار λ در بازهٔ 0.008 تا 0.08 تغییر کند، در اینصورت مقدار کمیت λ را هنگامیکه $Re=10^5$ و $d/k=200$ (1) و $k=0$ (2) است بیابید. تابع و برنامه را جهت استفاده در تمرین ٦-١١ ذخیره نمایید. {پاسخ: (1) $\lambda=0.0313$ و (2) $\lambda=0.0180$}

بخش ۵-۵-۲

۵ - ۶ مساحت بین دو منحنی سینوسی نشان داده شده در شکل ٥-١٤ را با استفاده از توابع *quad8* و *trapz* بیابید.

۵ - ۷ در تعیین توزیع بارگذاری در یاتاقانهای با اتکاء جانبی تحت یک بار خارج از مرکز، انتگرال زیر باید محاسبه شود[1].

$$I_m(\varepsilon)=\frac{1}{2\pi}\int_{-a}^{a}\left[1-\left(1-\cos(x)\right)/2\varepsilon\right]^{c}\cos(mx)dx$$

که در آن $\varepsilon>0$، $m=0$ و یا $m=1$ می باشد،

$$a=\cos^{-1}(1-2\varepsilon)$$

و برای یاتاقان ساچمه‌ای $c=1.5$ و برای یاتاقان رولر $c=1.1$ می‌باشد مقدار $I_1(0.6)$ را برای یاتاقان ساچمه‌ای محاسبه کنید. {پاسخ: $I_1(0.6)=0.2416$.}

۵ - ۸ معادلهٔ زیر؛

$$\int_0^{\infty}E_{\lambda,b}(\lambda,T)d\lambda=\sigma T^4$$

داده شده است که در آن :

$$E_{\lambda,b}(\lambda,T)=\frac{C_1}{\lambda^5\left[\exp(C_2/\lambda T)-1\right]}$$

---

[1] *W.Changsen, ibid.,p.92.*

و λ طول موج بر اساس μm ، T دما بر حسب درجهٔ کلوین، $c_1 = 3.742 \times 10^8 W.\mu m^4 / m^2$ ، $c_1 = 3.742 \times 10^8 W.\mu m^4 / m^2$ و $\sigma = 5.667 \times 10^{-8} W / m^2.k^4$ ثابت استفان - بولتزمن می باشد. این انتگرال را به صورت عددی برای $T = 300^\circ k$ ، $T = 400^\circ k$ ، $T = 500^\circ k$ انجام‌داده و درصد خطای نتایج تقریبی را در مقایسه با مقادیر دقیق محاسبه کنید. نکته‌ای که در اینجا وجود دارد اینست که هر دو حد انتگرال، کاربر را در هنگام انتگرال گیری عددی، دچار مشکل می‌کنند. بنابراین مقدار انتگرال را با تخمین حد پایینی $1\mu m$ و حد بالایی $150\mu m$، حل نمایید. این حدود با استفاده از نمودار $E_{\lambda,b}$ در سه دما و مقادیر حدود انتگرال بدست آمده اند و می توانند بدون ایجاد پیغام خطا توسط تابع $quad8$ بدست آیند. {پاسخ: $error_{300} = -0.145\%$، $error_{400} = -0.061\%$، $error_{500} = -0.030\%$}. (همچنین تمرین ۱۲- ٦ را مشاهده کنید.)

بخش ۵-۵-۳

۵ - ۹ رابطهٔ بین زاویهٔ پیشروی یک چرخدندهٔ حلزونی λ،و نسبت $\beta = \frac{N_1}{N_2}$، هنگامی که N_1 تعداد دندانه‌های چرخدنده حلزونی و N_2 تعداد دندانه های چرخدنده محرّک است و C فاصله بین میل محورها و p_{dn} گام قطری قائم می باشد، بصورت زیر می باشد[1]:

$$K = \frac{2 p_{dn} C}{N_2} = \frac{\beta}{\sin\lambda} + \frac{1}{\cos\lambda}$$

بازهٔ عملی مطلوب برای k و λ بصورت $1 \le k \le 2$، $1^\circ \le \lambda \le 40^\circ$، و $0.02 \le \beta < 0.3$ می‌باشد. برای ترکیبهای مشخص، مقادیر λ می‌تواند دارای یک مقدار، دو مقدار و یا هیچ مقداری نباشد.

الف) مقدار λ را منجر به ایجاد یک مینیمم می‌شود، هنگامی که $\beta = 0.02, 0.05, 0.08, 0.11, 0.15, 0.18, 0.23, 0.30$ می‌باشد بیابید. برنامه را برای استفاده در تمرین ٦ -۸ ذخیره نمایید.

ب) برای $k = 1.5$ و $\beta = 0.16$ مقدار (مقادیر) λ را بیابید.

۵- ۱۰ معادلهٔ ۵-۱ و حل عددی آنرا با ورودی پله در نظر بگیرید. مقدار ζ را که مینیمم کمیت زیر را ارائه می‌دهد، بدست آورید:

---

[1] *M. F. Spotts and T.E.Shoup،Design of Machine Elements, Prentice Hall,Upper Saddle River، NJ,1998,P.613*

$$f(\xi)=\sum_{n=1}^{N}\left(y(t_n)-1\right)^2$$

اجازه دهید ξ در بازه 0.05 تا 1.5 با افزایش مقدار 0.05 تغییر کند.

این مسئله باید دانسته شود که تابع *fminbnd* نمی‌تواند مورد استفاده قرار گیرد. بخاطر اینکه $f(\xi)$ آرایه‌ای از مقادیر عددی می‌باشد؛ از تابع *min* جهت بدست آوردن اندیس ξ و تعیین مقدار $f(\xi)$ استفاده نمایید.

۵ - ۱۱ در تمرین ۱-۷ نرخ جریان جرمی گاز خروجی از یک مخزن در فشار p_0 و تحت شرایط آدیاباتیک باز گشت پذیر متناسب است با ؛

$$\psi=\sqrt{\frac{k}{k-1}}\sqrt{\left(\frac{p_e}{p_0}\right)^{2/k}-\left(\frac{p_e}{p_0}\right)^{(k+1)/k}}$$

که در آن p_e ، فشار بیرونی خروجی تانک و k ثابت گاز، در فرایند آدیاباتیک بازگشت پذیر می‌باشد. ماکزیمم مقدار تابع در؛

$$\frac{p_e}{p_0}=\left(\frac{2}{k+1}\right)^{\frac{k}{k-1}}$$

رخ می دهد. مقدار ماکزیمم را بصورت عددی برای $k=1.4$ با استفاده از توابع *fminbnd* و *min* با 200 مقدار با فاصلهٔ یکسان در بازهٔ $0\le p_e/p_0\le 1$ تغییر دهید.

بخش ۵-۵-۴

۵ - ۱۲ حرکت یک موشک که نقطهٔ $(0,0)$ را با سرعت اولیهٔ v_0 و زاویه با محور افقی α ترک می‌کند را در نظر بگیرید. اگر موشک در موقعیت (x_e,y_e) فرود بیاید و در مواجه با نیروی درگ (اصطکاکی) که متناسب با مجذور سرعتش می باشد، قرار داشته باشد، در اینصورت چهار معادلهٔ درجهٔه اول که حاکم برحرکت این موشک می باشند بصورت زیر خواهند بود[1]:

[1] *H. B. Wilson and L. H. Turcotte,Advanced Mathematics and Mechanics Applications Using MATLAB,2nd ed.,CRC Press,Boca Raton, FL,1997,P.,249*

$$\frac{dv_x}{dx} = -c_d v \qquad \frac{dv_y}{dx} = \frac{-\left(g + c_d v v_y\right)}{v_x} \qquad \frac{dy}{dx} = \frac{v_y}{v_x} \qquad \frac{dt}{dx} = \frac{1}{v_x}$$

که در آن y، ارتفاع عمودی موشک، x فاصلهٔ افقی طی شده، t زمان، v_x و v_y به ترتیب مولفه‌های افقی و عمودی سرعت v، c_d ضریب درگ، g ثابت گرانش زمین، و

$$v = \sqrt{{v_x}^2 + {v_y}^2}$$

می‌باشد. این معادلات تنها زمانی معتبر هستند که v_0 به اندازه کافی بزرگ باشد تا هنگامی که آن به موقعیت x_e رسید v_x بزرگتر از صفر باشد. آزمون این شرایط می‌تواند بصورت $|v_x| > v_0 \times 10^{-6}$ انجام شود. اگر این شرایط ارضا نشود، در اینصورت اجرای برنامه باید متوقف شود. از تابع *error* جهت ایجاد توقف استفاده کنید. این محک در ابتدای تابعی که توسط *ode45* فراخوانی می‌شود، قرار می‌گیرد. شرایط اولیه بصورت؛

$$v_{ox} = v_0 \cos(\alpha) \qquad v_{oy} = v_0 \sin(\alpha) \qquad y = 0 \qquad t = 0$$

می‌باشند. جهت نوشتن معادلات فوق در مطلب فرض کنید $y_1(x) = v_x$، $y_2(x) = v_y$، $y_3(x) = y$، $y_4(x) = t$ باشند.

الف) مسیر حرکت موشک را به ازای $v_0 = 600\,fps$، $c_d = 0.002$ و $\alpha = 45^\circ$ تا هنگامی که موشک به زمین می‌رسد ($y_e = 0$)، رسم کنید. یعنی فقط نقاط را برای هنگامیکه $y_e > 0$ می‌باشد، رسم کنید. در تابع *ode45*، $x_{final} = 1000\,ft$ قرار می‌دهیم.

ب) ماکزیمم مقدار صعود موشک و مکان این مقدار ماکزیمم را بیابید. از تابع *fminbnd* و *spline* جهت بدست آوردن این مقادیر استفاده کنید. {پاسخ: $y_{max} = 474.8285\,ft$ که در $x = 648.1205\,ft$ اتفاق می‌افتد.}

پ) مقدار x_e را هنگامی که $y_e = 0$ است و همچنین مدت زمان لازم جهت رسیدن موشک به این نقطه را تعیین کنید. از تابع *interp1* جهت تعیین این مقادیر استفاده کنید. {پاسخ: $x_e = 975.3240$ و مدت زمانیکه موشک در حال پرواز است $10.6246\,s$ می‌باشد.}

۵ – ۱۳ ورزشکاری که علاقمند به پریدن از ارتفاع است، در حال آماده کردن خود برای یک پرش از داخل یک بالون که دارای ارتفاع زیادی از سطح زمین است می‌باشد. برای اینکار از طناب مخصوص کشسانی به طول l، استفاده می‌کند. جهت تعیین امنیت ورزشکار ماکزیمم شتاب، سرعت و مسافتی که ورزشکار در هوا طی می‌کند باید پیش بینی شود چون نیرویی که توسط شخص به طناب وارد

می‌شود خیلی بزرگ نمی‌باشد و فاصلهٔ بالون تا سطح زمین به اندازهٔ کافی زیاد است و ورزشکار به زمین برخورد نخواهد کرد. با در نظر گرفتن نیروهای اصطکاکی آیرودینامیکی وارده بر شخص، معادلهٔ حاکم بر مسئله به صورت[۱] ؛

$$\frac{d^2x}{dt^2} + c_0 signum(dx/dt)\left(\frac{dx}{dt}\right)^2 + \frac{k}{m_J}(x-L)u(x-L) = g$$

می‌باشد. که در آن $g = 9.8\,m/s^2$ شتاب جاذبهٔ زمین، c_d متناسب با ضریب اصطکاک و دارای واحد m^{-1} ، k ثابت فنریت طناب کشسان با واحد N/m ، m_J جرم ورزشکار، و u_z تابع پله ای واحد می‌باشد. یعنی هنگامیکه $z \le 0$ باشد $u(z)=0$ و هنگامیکه $z > 0$ باشد، $u(z)=1$ خواهد بود. برنامه‌نویسی این مسئله در صورت استفاده از اپراتورهای منطقی، که در بخش ٤-١ تشریح شد، بسیار ساده خواهد شد.

اگر $c_0 = 0.00324m^{-1}$ ، $k = 10\,N/m$ ، $m_J = 70kg$, $L=150\ m$ باشد و شرایط مرزی صفر باشند، در اینصورت نشان دهید که:

(1) ماکزیمم مقدار x، $-308.47m$ است که در $t = 11.47s$ رخ می‌دهد.

(2) ورزشکار در مدت زمان $5.988s$، $150m$ را طی کرده و به سرعت $-43.48\,m/s$ می‌رسد.

(3) ماکزیمم مقدار شتاب $-12.82\,m/s^2$ $(-1.308g)$ خواهد بود که در $t = 11.18s$ رخ می‌دهد.

همچنین نمودارهای جابجایی، سرعت و شتاب را رسم کنید. شتاب را می‌توانید با تقریب زدن آن با مشتق سرعت با استفاده از تابع *diff* بدست آورید. نتایج عددی بیان شده در بالا توسط استفاده از تابع *spline*، و استفاده از نتایج خروجی تابع *ode 45* بدست آمده‌اند.

۵-۱۴ یک پاندول معکوس که متشکل از میلهٔ صلب بدون وزنی به طول L می‌باشد و جرم m و یک فنر خطی با ثابت K به سر آزاد آن متصل شده‌اند، را در نظر بگیرید. پاندول در ابتدا به صورت قائم می‌باشد. طول آزاد فنر L می‌باشد. مفصل چرخشی پاندول دارای میرایی c می‌باشد و پاندول توسط

۱- برای نمونه به مثال زیر توجه نمایید.

D. M. Etter, Engineering Problems Solving with MATLAB, Prentice Hall, Upper Saddle River, NJ, 1997, pp. 220-221.

گشتاور محرک $M(t)$ به حرکت واداشته می‌شود. معادلهٔ حاکم بر سیستم که توصیف کنندهٔ حرکت دایره‌ای مجموعه می‌باشد بصورت زیر خواهد بود[1] ؛

$$\frac{d^2\theta}{d\tau^2}+\alpha\frac{d\theta}{d\tau}-\sin\theta+\beta\left(1-\frac{1}{\sqrt{5-4\cos\theta}}\right)\sin\theta=P(t)$$

که در آن؛

$$\beta=\frac{2kL}{mg}\qquad P=\frac{M}{mgL}\qquad \tau=t\sqrt{\frac{g}{L}}\qquad \alpha=(c/m)\sqrt{L/g}$$

و t مبین زمان می‌باشد.

اگر $M=0$، $\beta=10$، $\alpha=0.1$، $\theta(0)=\pi/4$، $d\theta(0)/d\tau=0$ باشد، نمودار تغییرات زاویه‌ای θ را به صورت تابعی از τ برای 1000 مقدار با فاصلهٔ یکسان در بازهٔ $0\le\tau\le 50$ رسم کنید، و در شکلی جداگانه، نمودار $\theta(\tau)$ را بر حسب $d\theta(\tau)/d\tau$ رسم نمایید.

۵ - ۱۵ یک کابل یکنواخت ناکشسان به طول L_0 و وزن واحد طول w که دو انتهایش در نقاط $x=0$ و $x=L$ بسته شده است و $L<L_0$ می باشد را در نظر بگیرید. اگر کابل دارای صلبیت خمشی نباشد و بتواند تنها نیروهای کششی را تحمل کند، در اینصورت معادلهٔ تغییر شکل بدون بعد $z(\eta)$ در کابل به شکل زیر خواهد بود[2] ؛

$$\frac{d^2z}{d\eta^2}=\beta\sqrt{1+\left(\frac{dz}{d\eta}\right)^2}$$

که در آن $\eta=x/L$، $\beta=wL/H$، H مولفهٔ افقیT و z منفی، به معنای تغییر شکل به سمت پایین می‌باشد. طول متناظر L_0 کابل از رابطهٔ زیر بدست می‌آید:

$$L_0=L\int_0^1\sqrt{1+\left(\frac{dz}{d\eta}\right)^2}\,d\eta$$

---

[1] *H. B. Wilson and L. H. Turcotte, ibid.,p.279.*
[2] *M. Irvine,ibid.,p. 4.*

که با داشتن آن کاربر می‌تواند مقدار β و در نتیجه H را هنگامیکه w، L و L_0 مشخص باشند، تعیین نماید. شرایط مرزی به شکل زیر می‌باشند:

$$z(0)=0 \qquad z(1)=0$$

مقدار β و شیب $dz(0)/d\eta$ را هنگامیکه $L_0/L=1.2$ می‌باشد، تعیین کنید. روش حل، نیازمند استفاده از حلقه‌های تکرار تو در تو می‌باشد. داخلی ترین حلقه مقدار شیب $dz(0)/d\eta$ را به ازای $z(1)=0$ تعیین می‌کند.

همچنین خارجی ترین حلقه مقدار β را که انتگرال فوق را ارضا می‌کند ارائه می‌دهد. تو در تو کردن حلقه ها در اینجا الزامی می‌باشد، زیرا جهت تعیین شیب $dz(0)/d\eta$ نیازمند دانستن مقدار β می‌باشیم. عملیات انتگرال گیری با استفاده از تابع *trapz* انجام شود.

{پاسخ: $\beta=2.1284$ و $dz(0)/d\eta=-1.2768$.}

۵ – ۱۶ نوسانات ارتفاع z مربوط به اختلاف سطح مایع در دو منبع منشوری با قاعده مستطیلی که توسط یک خط لولهٔ بلند به یکدیگر مرتبط شده‌اند را می‌توان توسط رابطه[1] ؛

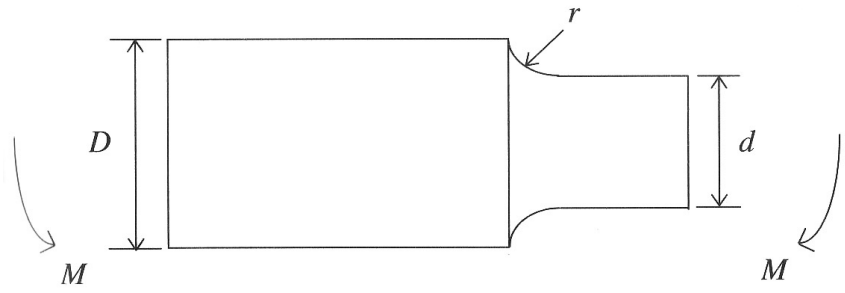
$$\frac{d^2Z}{dt^2}+signum(dZ/dt)p\left(\frac{dZ}{dt}\right)^2+qZ=0$$

تعیین کرد. اگر $p=0.375m^{-1}$، $q=7.4\times10^{-4}s^{-2}$ و شرایط اولیه، $Z(0)=Z_nm$ و $dZ(0)/dt=0\,m/s$ باشد، در اینصورت اولین مقدار t_n، $n=1,2$ را برای اینکه $Z(t_n)=0$ شود، هنگامیکه $Z_1=10m$ و $Z_2=50m$ می‌باشد، تعیین کنید. برای تعیین t_n از تابع *interp1* استفاده کنید. علامت کمیت را می توان با استفاده از تابع *sign* تعیین نمود.

پیشنهاد : نتایج را برای یک مقدار Z_n رسم کنید و سپس با توجه به مشخصات منحنی یک ترکیب مناسب از توابع *min* و *find* را جهت تعیین اندیس میانی حد پایین مقادیر که تابع *interp1* عملیات میانیابی را انجام می‌دهد، ایجاد کنید. {پاسخ: $t_1=114.2692s$ و $t_2=276.1428s$.}

---

[1] *D. N. Roy,Applide Fluid Mechanics،Ellis Horwood Limited,Chichester,England, 1988,pp.290-293.*



شکل ۵ - ۱۵ هندسه و نحوهٔ بارگذاری برای ضریب تمرکز تنش

بخش ۵-۵-۵

۵ - ۱۷ الف) از تابع *fsolve* جهت یافتن مقادیر θ بر حسب درجه و k که معادلات زیر را هنگامیکه $a=1$ و $b=3$ می‌باشد، ارضا می‌کند، استفاده کنید.

$$b = k(1 - cos\,\theta)$$

$$a = k(\theta - sin\,\theta)$$

ب) دو معادلهٔ داده شده در قسمت (الف) می توانند با ترکیب به معادلهٔ زیر تبدیل شوند :

$$b(\theta - sin\,\theta) - a(1 - cos\,\theta) = 0$$

از تابع *fzero* جهت تعیین مقدار θ ، هنگامیکه $a=1$ و $b=3$ می باشد استفاده کنید و سپس از یکی از معادلات قسمت (الف) جهت تعیین k استفاده کنید. {پاسخ: $k = 6.9189$ و $\theta = 55.4999^\circ$.}

۵ - ۱۸ الف) از تابع *fsolve* جهت تعیین مقادیر Q، T_A و T_B هنگامیکه $\sigma = 5.667 \times 10^{-8}$، $T_1 = 373k^\circ$ و $T_2 = 293k^\circ$ می‌باشد، استفاده کنید.

$$T_1^{\,4} - T_A^{\,4} = Q/\sigma$$

$$T_A^{\,4} - T_B^{\,4} = Q/\sigma$$

$$T_B^{\,4} - T_2^{\,4} = Q/\sigma$$

ب) معادلات قسمت (الف) را می‌توان به صورت زیر نوشت:

$$\begin{bmatrix} 1 & 0 & 1/\sigma \\ 1 & -1 & -1/\sigma \\ 0 & 1 & -1/\sigma \end{bmatrix} \begin{Bmatrix} x \\ y \\ Q \end{Bmatrix} = \begin{Bmatrix} T_1^4 \\ 0 \\ T_2^4 \end{Bmatrix}$$

که در آن $x = T_A^4$ و $y = T_B^4$ می‌باشد. مقادیر Q، T_A و T_B را از این دستگاه معادلات با استفاده از تقسیم وارون (\)، بدست آورید. {پاسخ: $T_A = 352.052$، $T_B = 326.5116$ و $Q = 226.4312$.}

بخش ۵ -۶ -۱

۵-۱۹ ضریب تمرکز تنش برای یک شفت پله‌ای نشان داده شده در شکل ۵-۱۵ با عبارت زیر تخمین زده می‌شود[1]:

$$K_t = c\left(\frac{D-d}{2d}\right)^{-a}$$

که در آن c و a در جدول ۵-۳ داده شده‌اند. دو عبارت، یکی برای c و دیگری برای a بصورت تابعی از D/d ، به دو شیوه بدست آورید: *(1)* با یک چند جمله‌ای درجهٔ پنج *(2)* با یک اسپیلاین. در هر دو شیوه مقادیر k_t بدست آمده توسط دو مجموعه مقادیر برازش شده را با مقادیر اصلی داده‌شده در جدول ۵-۳ مقایسه کنید. کدام شیوه در این مورد بهتر است؟

بخش ۵ -۶ -۴

۵ - ۲۰ سیگنال زیر را در نظر بگیرید:

$$f(t) = \sum_{n=1}^{4} H_n e^{-\xi_n \omega_n t} \sin\left(\sqrt{1-\xi_n^2}\,\omega_n t\right) \qquad 0 \le t \le T$$

جدول ۵-۳ (ثوابت ضریب تمرکز تنش)		
D/d	c	a
00.6	0.88	0.33

[1] *R. L. Notron Machine Design,An Integrated Approach,Prentice Hall,Upper Saddle River,NJ,1996,P.1,005ff*

3.00	0.89	0.31
2.00	0.91	0.29
1.50	0.94	0.26
1.20	0.97	0.22
1.10	0.95	0.24
1.07	0.98	0.21
1.05	0.98	0.20
1.03	0.98	0.18
1.01	0.92	0.17

جدول ۵ – ۴ (ثوابت تعریف کنندهٔ سیگنال در تمرین ۵ – ۲۰)

n	$\varpi_n / 2\pi$	ξ_n	H_n
1	5	0.1	1
2	9	0.04	1.3
3	9.4	0.04	1.3
4	20	0.03	1.8

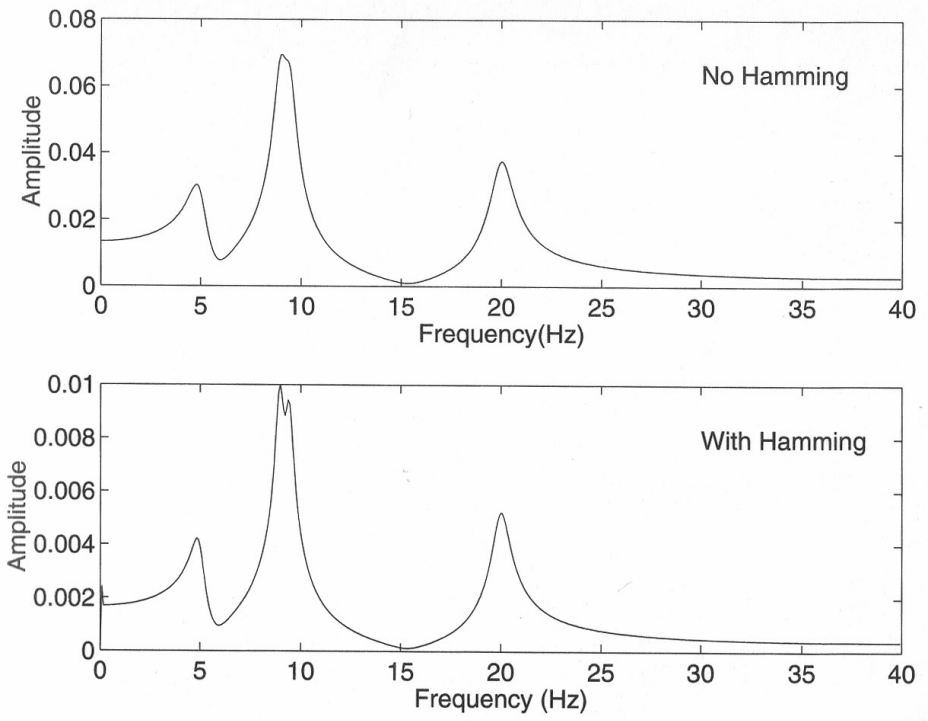
که در آن مقادیر ثابتها در جدول ٥-٤ داده شده است. برای $N = 2^{10}$ و $\Delta t = 2\pi/(4\omega_4)$؛

الف) نمودار دامنه را برای این سیگنال با استفاده از تابع همینگ و بدون استفاده از آن رسم کنید. نتایج باید شبیه اشکال نشان داده شده در شکل ٥ -١٦ باشد.

ب) فرکانسهایی را که نقاط ماکزیمم در آنها رخ می‌دهد را تعیین کنید. {راهنمایی: از چند کاربرد تابع *find* و *diff* استفاده کنید.}

{پاسخ: بدون تابع همینگ: Hz $[4.84375 \quad 9.14063 \quad 20.0781]$

با تابع همینگ: Hz $[4.92188 \quad 9.0625 \quad 9.45313 \quad 20.0781]$.}



شکل ۵ – ۱۶) نتایج بدست آمده از حل مثال ۵ – ۲۰ – الف

فصل ششم

گرافیک دو بعدی

در این فصل با قابلیتهای متنوع ترسیمات دو بعدی در نرم افزار مطلب آشنا خواهیم شد.

۶- ۱ مقدمه

نرم افزار مطلب طیف وسیعی از دستورات قابل انعطاف و تعمیم پذیر را در فضای دو بعدی و سه بعدی در اختیار کاربر قرار می‌دهد. توابع ترسیمی را می‌توان در سه گروه طبقه بندی نمود : مدیریت گرافیکی، ایجاد سطح و منحنی، برچسب‌گذاری مشخصات نمودار. اگر چه تعداد دستورات ترسیمی مطلب زیاد نمی‌باشد، در اغلب موارد شکل استفاده از این توابع و نحوهٔ بر چسب‌گذاری آنها یکسان می‌باشد. توابعی که موارد استفاده‌شان در این فصل و فصل بعدی توضیح داده خواهد شد، عبارتند از:

مدیریت	ایجاد	ویژگی ها و نشانه گذاری
figure	2- D	xlabel
subplot	plot	ylabel
zoom	polar	zlabel(3D only)
hold	fill	text
view (3D only)	plotyy	text(3D only)
rotate3d (3D only)		title

3- D	legend(2D only)
Plot3	box
surf,surfc	set
mesh,meshz	grid
contour,contour3,	axis,axis equal,
contourf	axis off

waterfall	colorbar(3D only)
cylinder	clabel
	colormap(3D only)

برخی از توابع ترسیم خاص، مثل *bar* و *hist*، در فصل‌های بعدی بکار خواهند رفت.

هنگام ایجاد نمودارها، کاربر باید تمام تلاش خود را جهت اینکه (۱) نمودارها مبین اهداف خواسته شده باشند و (۲) دارای وضوح کامل همراه با برچسب گذاریهای لازم، محورهای نشانه‌گذاری شده، عنوان شکلها، مشخصات شکلها (هنگامیکه بیش از یک شکل موجود است.) و نمایش مقادیر عددی مهم باشند، قرار دهد. هر ابزاری جهت کامل کردن شکل، مثل رنگ، نوع خط، نشانه ها، و متنها ، باید موجب گمراه شدن خواننده نشود. دستورات معمولی ایجاد توابع، متشکل از توابع مدیریتی اضافی می‌باشند. به هر حال، به جز در مورد توابع مدیریتی، ترتیب قرارگیری این توابع، در اغلب موارد، کاملاً دلخواه است. همچنین بکارگیری بر چسب گذاری و توابع مشخصهٔ نمودار نیز اختیاری می‌باشد. مطلب محورها را مدرج کرده و اندازهٔ محورها را نیز بصورت بر چسب‌هایی، حتی اگر بیش از یک سری داده ترسیم شود، روی محورها قرار می‌دهد. بنابراین، کاربر می‌تواند در صورت استفادهٔ صحیح از توابع نمودارهایی با برچسب گذاری لازم ایجاد کند.

نمودارها در پنجرهٔ شکل ترسیم می‌شوند، که این پنجره توسط مطلب در هنگام اجرا، در صورتیکه هر یک از توابع مدیریت، ایجاد و برچسب گذاری نمودار فراخوانی شود، بوجود می‌آید. هنگامیکه یک برنامه (تابع و یا فایل متنی) ، از بیش از یک تابع ایجاد نمودار استفاده کند، مطلب پنجرهٔ شکل جدیدی ایجاد خواهد کرد و هر پنجرهٔ شکلی که قبلاً ایجاد شده است پیش از ایجاد پنجرهٔ شکل جدید، از بین خواهد رفت. جهت ایجاد هر نمودار در پنجرهٔ شکل خاص خودش، کاربر باید از دستور زیر استفاده کند:

```
figure(n)
```

که در آن *n*، عددی صحیح می‌باشد. اگر آرگومان دستور *figure* حذف شود، مطلب به صورت خودکار مقدار صحیح بعدی را به عنوان آرگومان آن در نظر می‌گیرد.

کاربر همچنین می‌تواند چندین نمودار ساخته شدهٔ مستقل را با استفاده از دستور زیر در یک پنجرهٔ شکل قرار دهد:

```
subplot(i,j,k)
```

دو آرگومان اول، پنجرهٔ را به بخشهایی تقسیم می‌کنند (به صورت ستونی و سطری)، و اندیس سوم نشان دهندهٔ بخشی است که نمودار در آن بخش ترسیم می‌شود. مقدار *1* برای این آرگومان مبین گوشهٔ بالا، سمت چپ می‌باشد و مجموع تعداد سطرها و ستونها مبین گوشهٔ پایین، سمت راست می‌باشد. با افزایش این عدد، پنجره از چپ به راست حرکت خواهد کرد. که اینکار از سطر بالا شروع می‌شود. هر تابع مدیریتی و یا برچسب گذاری که پس از دستور *figure/subplot* در برنامه ظاهر شود، تنها در مورد بخشی که توسط آرگومان سوم دستور *subplot* تعیین می‌شود، اعمال خواهد شد. در هر بخش هر مجموعهٔ سازگاری از توابع ایجاد نمودارهای دو بعدی و یا سه بعدی می‌تواند مورد استفاده واقع شود. برای دیدن چگونگی استفاده از توابع *figure* و *subplot* در قالب چند مثال، به شکل ٦-١ مراجعه کنید. تذکر این نکته در اینجا ضروری است که اگر تنها نیاز به یک پنجرهٔ شکل داشته باشیم، دستور *figure* می تواند حتی در صورت استفاده از دستور *subplot* حذف شود.

از آنجا که هر تابع ایجاد شکل یک پنجرهٔ شکل جدید باز می‌کند[1]، جهت ترسیم بیش از یک منحنی، سطح و یا خط (و یا ترکیبی از اینها) روی یک نمودار، کاربر باید از دستور زیر استفاده نماید.

```
hold on
```

که پنجرهٔ کنونی را فعال نگاه می‌دارد. اما، کاربر تنها مجاز به استفاده از توابع ایجاد نمودار همساز مثال *surf* و *plot3* یا *plot* و *fill* می‌باشد. همهٔ اشکالی که ایجاد شده‌اند می‌توانند با استفاده از انتخاب گزینهٔ *copy figure* در منوی پایین کشیدنی *Edit* در هر جای دلخواه در محیط ویندوز کپی شوند. نرم افزار مطلب ابزارهایی را جهت تبدیل یک شکل به فرمت سازگار با بسیاری از پرینترهای متداول ارائه می‌دهد. برای مثال، اگر کاربر بخواهد که نتایج گرافیکی ظاهر شده در پنجرهٔ شکل فعال را به صورت *encapsulated postscript file* با سطح -٢ جهت استفاده از پرینترهای سیاه و سفید و تحت نام *File Name* ذخیره کند، در اینصورت از دستور زیر استفاده خواهد کرد.

```
print -deps2 'c:\path\FileName.eps'
```

که در آن *path* نام شاخه و یا زیر شاخه‌ای که فایل در آن قرار خواهد گرفت را نشان می‌دهد. برای سایر گزینه‌ها در دستور *print* به فایل کمکی آن مراجعه کنید. از طرفی، اگر کاربر بخواهد که یک

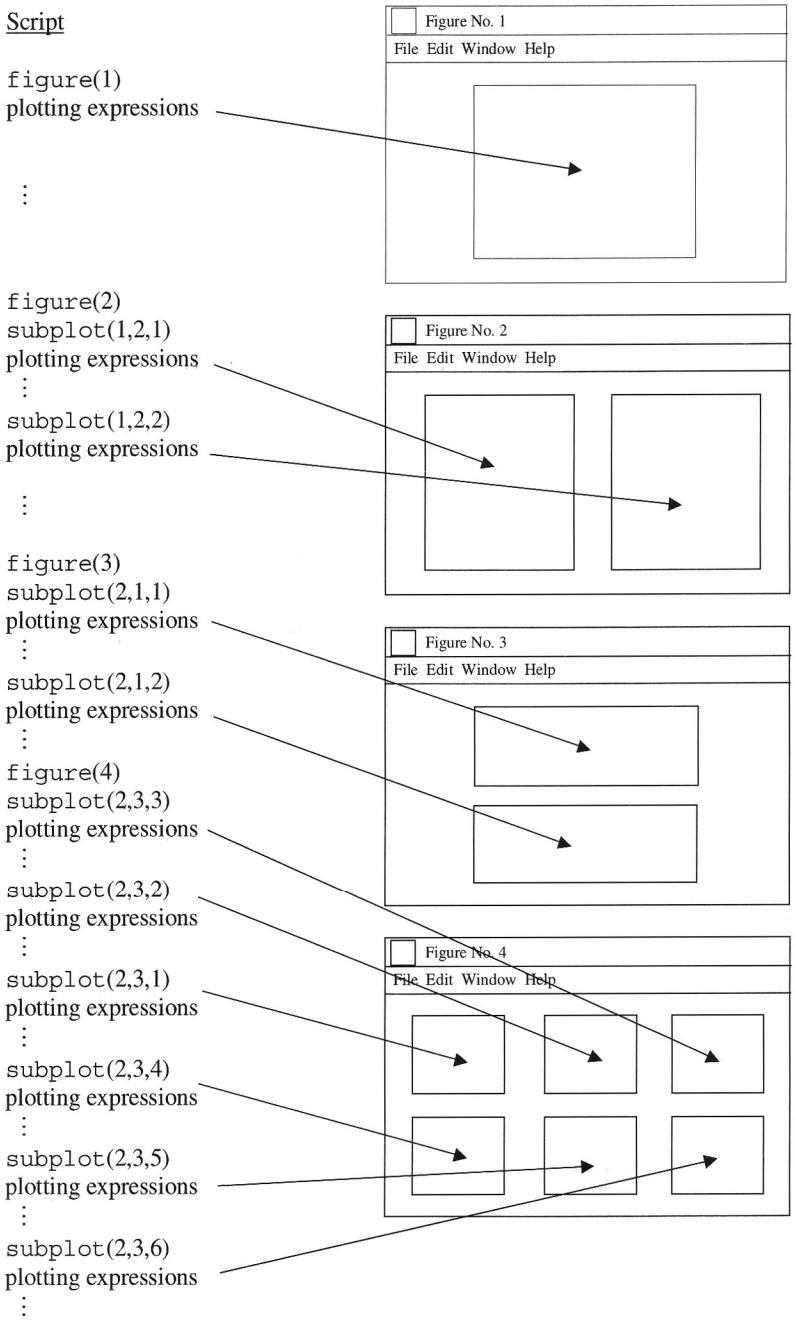
---

١ - شکل پنجره مطلب، چگونگی ادارهٔ آن و تعاریف مربوط به مدیریت فایلها، به محیط ویندوز مربوط می‌شود. این مسئله در هنگام کار با سایر سیستم عامل دیگر نیز وجود خواهد داشت.

فایل *encapsulated postscript file* را در *Ms word document* قرار دهد که در آن پیش نمایی از شکل، نمایش داده شود، باید از دستور زیر استفاده کند[۱].

```
print -deps2 -tiff 'c:\path\FileName.eps'
```

۲ – جهت استفادهٔ این فایل در *Ms word*، فیلترهای مناسب *encapsulated postscript* باید نصب شود. که بخشهایی از *Ms word* می‌باشند. اما همیشه با نصب *Ms word*، نصب نمی‌شوند. در این مورد باید به گزینهٔ نصب *Ms word* رجوع کرده و فیلترهای دلخواه را نصب نمود.



شکل ٦-١) مثالهایی از چگونگی استفاده از ترکیبهای مختلف توابع *figure* و *subplot*

۶ -۲ دستورات اساسی ترسیمات دو بعدی

شکل کلی دستور ترسیمات دو بعدی بصورت زیر می باشد:

```
plot(u1,v1,c1,u2,v2,c2,…)
```

که در آن u_i و v_i به ترتیب مختصه های x و y یک نقطه و یا مجموعه ای از نقاط می باشند. آنها می توانند جفتهایی از اعداد، بردارهایی با طول یکسان، ماتریسهایی هم مرتبه، و یا عبارتهای محاسباتی که پس از محاسبه به یکی از سه شکل فوق تبدیل می شوند، باشند. کمیت c_j رشته ای از کاراکترها می باشد. یکی از کاراکترها رنگ خط/ نقطه را تعیین می کند، کاراکتر دیگر نوع نقاطی را که باید استفاده شوند، تعیین می‌کند. و حداکثر از دو کاراکتر می‌توان جهت تعیین مشخصات خط استفاده نمود. هنگامیکه لازم است تا مجموعه ای از نقاط رسم شود، یکی از کاراکترهای C_j می‌تواند برای مثال، 's' جهت رسم مربع یا '*' جهت ترسیم ستاره باشد. هنگامی که مجموعه ای از نقاط را بخواهیم با خط مستقیم به هم وصل کنیم، کاراکتر c_j می‌تواند علامت '-' برای خط توپر و '--' برای خط مقطع باشد. هنگامی که قرار است که خطوط و نقاط با رنگ یکسانی رسم شوند، کاراکتر c_j شامل هر دو تعریف کننده خواهد بود. برای مثال اگر بخواهیم از خطوط مقطع آبی که از به هم پیوستن نشانه‌های الماسی آبی رنگ (مربوط به نقاط) بوجود آمده است، استفاده کنیم، در اینصورت کاراکتر c_j به صورت *'b--d'* خواهد بود. ترتیب قرار گرفتن سه مجموعهٔ کاراکتر در میان علامت کوتیشن دارای اهمیت نمی‌باشد. هنگامیکه لازم است که خطوط و نقاط، هر دو، رسم شوند اما تعداد نقاطی که خط را ایجاد می‌کنند با تعداد نقاطی که باید رسم شوند متفاوت است، این حالت c_1 نشانه‌ای خواهد بود که قرار است بعنوان نوع خط بکار رود و c_2 نشانه‌ای مبین نوع نقطه خواهد بود و بر عکس. به فایل کمکی دستور *plot* جهت دستیابی به لیستی از رنگها و نوع خطها و نقطه ها مراجعه کنید. اگر کاراکتر c_j حذف شود، در اینصورت مقادیر پیش فرض سیستم استفاده خواهند شد. اگر بیش از یک منحنی ترسیم شود، در اینصورت رنگ خطوط به ترتیبی که پیش فرض مطلب است، تغییر خواهد کرد.

اکنون دستورات ایجاد، نقاط، خطوط، دوایر، عبارات محاسباتی، خانواده‌ای از منحنی‌ها و منحنی‌های توصیف شده توسط توابع چند مقداری را توضیح خواهیم داد.

۶ -۲ -۱ نقاط

جهت قرار دادن یک نشانهٔ ستارهٔ قرمز در مکان *(2,4)*، از ساختار دستور زیر استفاده می‌کنیم:

```
plot(2,4,'r*')
```

۶ -۲ -۲ خطوط

جهت ترسیم خطی که نقاط $(0,0)$ و $(1,2)$ را توسط نوع خط پیش فرض به یکدیگر وصل می کند (خط توپر یا رنگ آبی)، دستور زیر را استفاده می‌کنیم:

```
plot([0 1],[0 2])
```

توجه کنید که اولین بردار دو عضوی $[1\ 0]$ مبین مقادیر مختصهٔ x و بردار دو عضوی بعدی $[2\ 0]$ مبین مقادیر مختصهٔ y می‌باشد. بنابراین اولین عنصر هر بردار، مختصهٔ یک نقطهٔ انتهایی خط (x,y) را تعیین می‌کند، و عنصر دوم این بردار ها مختصه های نقطهٔ انتهایی دیگر خط می‌باشند.

فرض کنید که می‌خواهیم مجموعه ای از n خط منقطع که نقاط انتهایی آنها (x_{1n},y_{1n}) و (x_{2n},y_{2n}) می‌باشند، را ترسیم کنیم. جهت انجام این کار، چهار بردار زیر را ایجاد خواهیم کرد.

$$x_j = [\,x_{j1}\ \ x_{j2}\ \ \ldots\ \ x_{jn}\,]$$

$$y_j = [\,y_{j1}\ \ y_{j2}\ \ \ldots\ \ y_{jn}\,] \qquad j = 1,2$$

در اینصورت دستور ترسیم به شکل زیر خواهد بود[۱]:

```
plot([x1;x2],[y1;y2])
```

که در آن $[x1;x2]$ و $[y1;y2]$ هر یک ماتریسهای $(n\times 2)$ می باشند. جهت روشنتر کردن این مطلب، اجازه دهید چهار خط عمودی را از $y=0$ تا $y = cos(\pi x/20)$، در حالیکه $x=2,4,6,8$ می‌باشد ترسیم کنیم. برنامه به صورت زیر خواهد بود:

```
x=2:2:8;
plot([x;x],[zeros(1,length(x));cos(pi*x/20)],'k')
```

چون $x_1 = x_2$ می‌باشد. از آنجا که رنگ تعیین شده است، بنابراین تمام خطوط دارای رنگ یکسانی (در این مورد سیاه) خواهند بود. تابع *zeros* جهت ایجاد برداری شامل مقادیر صفر، با طولی یکسان با طول x استفاده می شود. نتایج در شکل ٦ -۲ - الف نشان داده شده است. بدلیل اینکه مطلب

---

۱ - تابع *plot* از بعضی جهات، تعمیم تابع *stem* می باشد، که فرض می کند $x_1 = x_2$ و $y_1 = 0$ می‌باشد.

دارای قابلیت اندازه دهی خودکار محورها می‌باشد، پنجرهٔ شکل ایجاد شده، با توجه به خطوط ابتدا و انتها اندازه دهی می‌شوند. بنابراین جهت قابل رویت کردن این خطوط ابتدا و انتهایی باید از دستور زیر استفاده شود:

```
axis ([xmin xmax ymin ymax])
```

که در آن x_{min}، x_{max}، y_{min} و y_{max} به ترتیب مقادیر مینیمم و ماکزیمم محورهای x و y می‌باشند. بنابراین فایل متنی اصلاح شده به صورت زیر خواهد بود :

```
x=2:2:8;
plot([x; x],[zeros(1,length(x));cos(pi*x/20)],'k')
axis ([1 9 0 1 ])
```

شکل اصلاح شده در شکل ٦ – ٢ – ب نشان داده شده است. دستیابی به مقادیر حدود محورها و تعریف مجدد یک یا چندتای آنها در صورت لزوم، موجب ایجاد انعطاف پذیری بیشتر خواهد شد. حدود را می‌توان با استفاده از دستور زیر بدست آورد.

```
v=axis;
```

که در آن v برداری متشکل از چهار عنصر مطابق زیر، خواهد بود:

$$v(1) = x_{min} \qquad v(3) = y_{min}$$

$$v(2) = x_{max} \qquad v(4) = y_{max}$$

بنابراین جهت دستیابی به شکل اصلاح شده، شکل ٦-٢ ب، برنامه را باید به شکل زیر اصلاح کنیم ؛

$$x = 2 : 2 : 8;$$

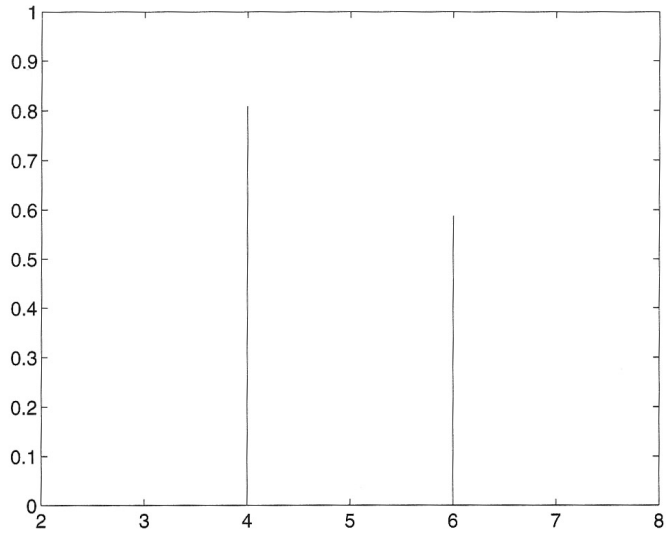
$$plot([x;x],[zeros(1,length(x));cos(pi*x/20)],'k')$$

$$v = axis;$$

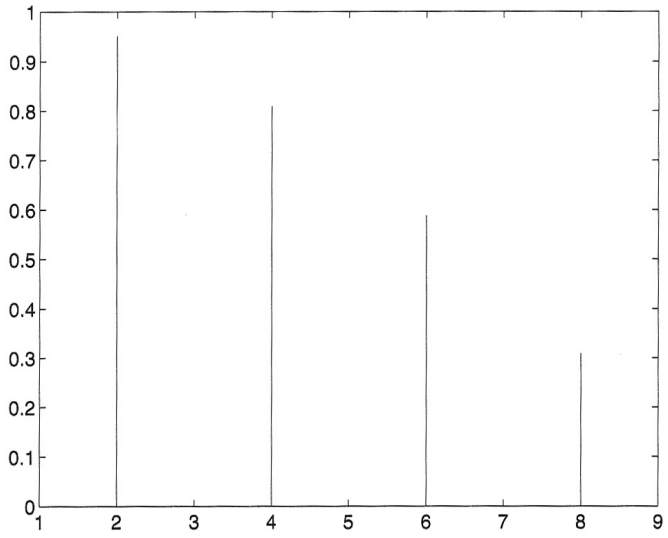
$$v(1) = 1;$$

$$v(2) = 9;$$

$$axis(v)$$



(الف)



(ب)

شکل ٦ – ٢) (الف) حالتیکه چهارچوب شکل خطوط را مخفی می کند؛ (ب) استفاده از تابع *axis* جهت تعیین حدود محورها به گونه ای که تمامی خطوط دیده شوند.

۶-۲-۳ دوایر

جهت ترسیم دایره‌ای به شعاع r که مرکز آن در نقطه ای به مختصات (a,b) در دستگاه مختصات کارتزین واقع شده است، کاربر باید ابتدا آنرا توسط روابط زیر ؛

$$x = a + r\cos(\theta)$$

$$y = b + r\sin(\theta)$$

به دستگاه مختصات قطبی (شکل ۲-۲ را بیاد بیاورید) تبدیل کند. که در آن $0 \le \theta \le \theta_1 \le 2\pi$ می‌باشد. هنگامیکه $\theta_1 \le 2\pi$ باشد در اینصورت کمانی از دایره خواهیم داشت. اگر فرض کنیم $\theta_1 = 2\pi$، $a = 1$ ، $b = 2$ و $r = 0.5$ ، در اینصورت برنامهٔ مورد نیاز برای ترسیم دایره بصورت زیر خواهد بود.

```
theta = linspace(0.2* pi);
plot(1+0.5*cos(theta),2+0.5*sin(theta))
axis  equal
```

تابع *axis equal* نمودار را متناسب می‌کند، لذا دایرهٔ مورد نیاز، به شکل دایره دیده خواهد شد نه بیضی.

برنامهٔ ترسیم خانواده‌ای متشکل از شش دایرهٔ هم مرکز که شعاع اولیه آن 0.5 بوده و با نرخ افزایش 0.25 زیاد می‌شود و مراکزشان توسط علامت '+' مشخص می‌شود، بصورت زیر می‌باشد:

```
theta = linspace(0.2* pi,50)        %  (1×50)
rad = 0.5 : 0.25 : 1.75;            %  (1×6)
x = 1+cos(theta)'*rad;              %  (50×6)
y = 2 + sin(theta)'*rad;            %  (50×6)
plot(x,y,'k',1,2,'k+')
```

axis equal مقادیر ماتریس‌ها را به صورت ستون به ستون نشان خواهد داد. از آنجا که می‌خواهیم تمامی 50 مقدار *theta* برای هر مقدار *rad* رسم شود. آنها را به صورت ماتریسهایی (50×6) تشکیل خواهیم داد. اگر رشتهٔ *'k'* حذف شود، در اینصورت هر دایره با رنگی متفاوت ترسیم خواهد شد. اجرای این برنامه شکل ٦-۳ را نتیجه خواهد داد.

۶-۲-۴ تابعی بر حسب تابع دیگر

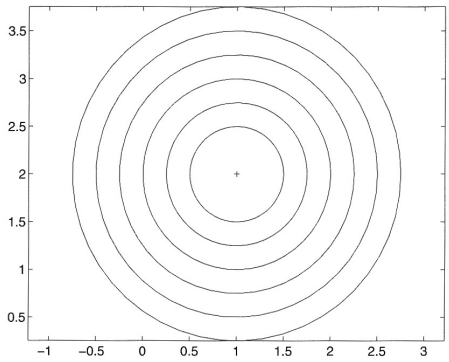
هنگامیکه $sin(n\theta)$ بر حسب $sin(m\theta+\theta_1)$ ترسیم می‌شود که در آن m و n اعداد مثبتی بوده، $0 \le \theta \le 2\pi$ و $0 \le \theta_1 < 2\pi$ می‌باشد، نموداری مطابق شکل ٦ -٤ - الف خواهیم داشت. اجازه دهید موردی را در نظر بگیریم که $n=1$، $m=2$ و $\theta_1 = \pi/4\ (45^\circ)$ می‌باشد. اگر 101 مقدار θ با فاصلهٔ یکسان را در نظر بگیریم، در اینصورت برنامه به صورت زیر خواهد بود :

```
theta=linspace(0,2*pi,101);
plot(sin(theta),sin(2*theta+pi/4))
```

اجرای این برنامه در شکل ٦ -٤ - الف نشان داده شده است. می توانیم این شکل را بدون نمایش محورهای مختصات با افزودن دستور زیر به برنامه ایجاد کنیم:

```
axis off
```

نتیجهٔ افزودن دستور فوق به برنامه در شکل ٦ -٤ - ب نشان داده شده است.



شکل ٦ - ۳) دوایر هم مرکز

۶ -۲-۵ خانواده ای از منحنی ها

یک شیوهٔ ترسیم خانواده‌ای از منحنی‌ها در هنگام ترسیم شش دایرهٔ هم مرکز در قسمت قبلی نشان داده شد. بطور کلی، مطلب به کاربر اجازه می‌دهد که یک محور را با بردار و محور دیگر را با یک ماتریس تعریف کند. در این صورت منحنی‌ها توسط ترسیم بردارها در مقابل سطرها و یا ستونهای ماتریس، بسته به اینکه طول بردار با تعداد کدامیک (سطرها یا ستونها) هم خوانی داشته باشد، بدست خواهد آمد. به عنوان مثال به ترسیم خانواده ای از سهمی ها با ضابطهٔ زیر ؛

$$y = a^2 - x^2$$

در بازهٔ $-5 \le x \le 5$ و $a = 1,2,...,5$ توجه کنید. برنامهٔ آن بصورت زیر خواهد بود:

```
x=-5:0.2:5;
a=1:5;
[xx,aa] =meshgrid(x.^2,a.^2);
plot(x,aa-xx,'k')
```

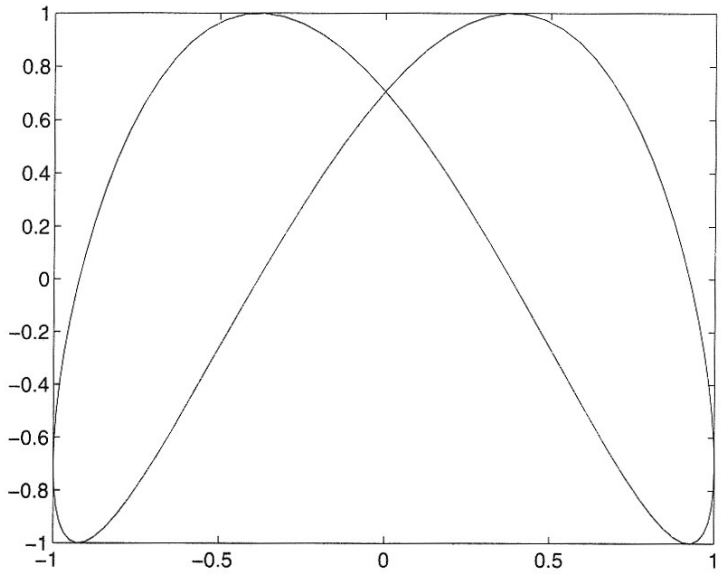
که شکل ٦ -٥ را نتیجه خواهد داد.

اکنون می خواهیم، همگرایی سری زیر را بصورت شهودی و با استفاده از نمودار ؛

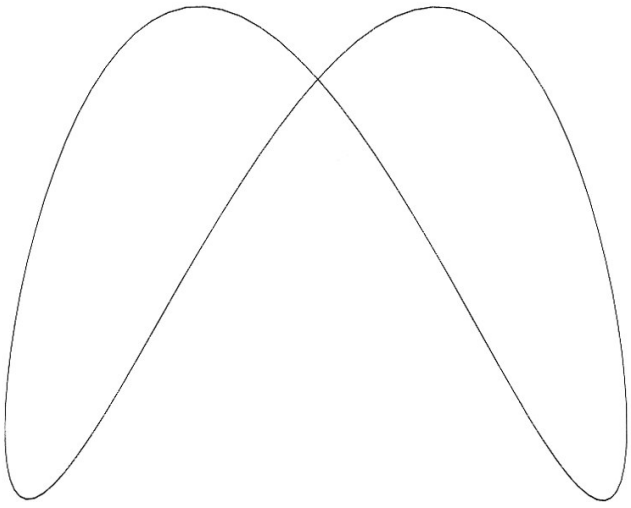
$$S_N = \sum_{j=1}^{N} \frac{1}{(a+j)^2}$$

به ازای $N=1,2,.....,10$ و $a=1,2,3$ نشان دهیم. در این مورد از دستور *cumsum* (بخش ۲ -٥ را بیاد بیاورید) در قالب برنامهٔ زیر استفاده خواهیم نمود:

```
aa=1:3;                    %(1×3)
```



(الف)



(ب)

شکل ٦ – ٤) (الف) منحنی پروانه ای (ب) منحنی پروانه ای با استفاده از دستور *axis off*

```
N=1:10;                          %(1×10)
[a,k]=meshgrid(aa,N);            %(10×3)
S=cumsum(1./(a+k).^2);           %(10×3)
plot(N,S,'k')
```

که پس از اجرا شکل ٦ -٦ را نتیجه خواهد داد.

۶ -۲ -۶ چند تابع روی یک شکل[۱]

سه تابع زیر را در نظر بگیرید:

$$g_1(x) = 0.1x^2$$

$$g_2(y) = cos^2\, y$$

$$g_3(z) = e^{-0.3z}$$

که در آن $0 \le x = y = z \le 3.5$ می باشد. می‌توانیم این سه تابع را روی یک شکل به هر یک از سه شیوهٔ ذیل ترسیم کنیم:

```
x=linspace(0,3.5);
plot(x,[0.1*x.^2;cos(x).^2;exp(-0.3*x)],'k')
```

یا

```
x=linspace(0,3.5);
plot(x,0.1*x.^2,'k',x,cos(x).^2,'k',x,exp(-0.3*x)],'k')
```

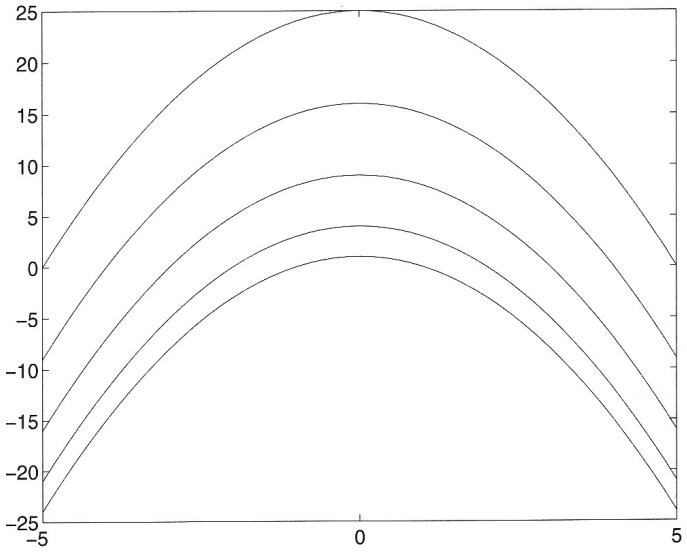
و یا

```
x=linspace(0,3.5);
plot(x,[0.1*x.^2,'k',x,cos(x).^2,'k',x,exp(-0.3*x)],'k')
x=linspace(0,3.5);
```

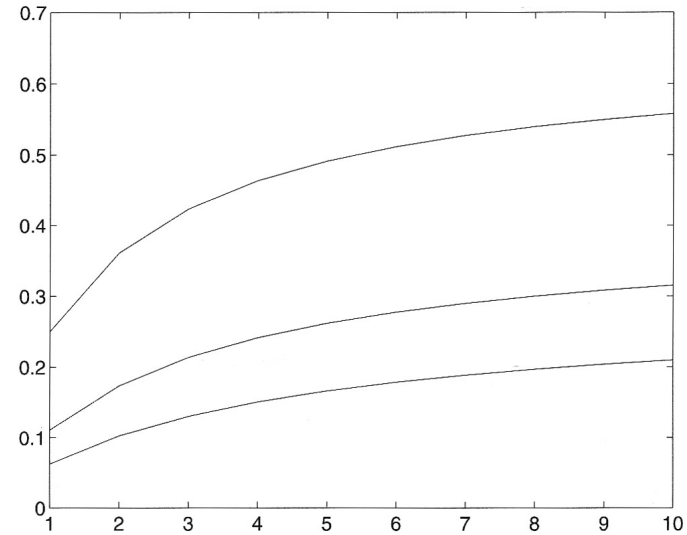
---

۱ - جهت ترسیم دو نوع مختلف نمودار با دو دستگاه مختصات متفاوت از دستور *plotyy* استفاده نمایید. که این مطلب در قسمت ٦-۳-٦ نشان داده شده است. همچنین به برنامه ای که شکل ١٤-١ را ایجاد کرده است، توجه کنید.

```
plot(x,[0.1*x.^2,'k')
hold on
plot(x,cos(x).^2,'k')
plot(x,exp(-0.3*x),'k'))
```



شکل ۶ – ۵) خانواده ای از سهمی ها



شکل ۶ – ۶) دیدن همگرایی یک سری

اجرای هر یک از برنامه‌ها منجربه ایجاد شکل ٦-٧- الف خواهد شد. که در آن تمامی منحنی‌ها دارای رنگ یکسان (در اینجا سیاه) می‌باشند.

از طرف دیگر، اگر دامنهٔ کاری متغیرهای مستقل در مورد هر یک از این توابع متفاوت باشد، در اینصورت تنها از برنامه‌های دوم و سوم می‌توانیم استفاده کنیم. برای مثال اگر $0 \le x \le 3$، $1 \le y \le 4$ و $2 \le z \le 5$ باشد، در اینصورت برنامهٔ دوم در بالا به صورت زیر نوشته خواهد شد:

```
x=linspace(0,3,45);
y=linspace(1,4,55);
z=linspace(2,5,65);
plot(x,0.1*x.^2,'k-',y,cos(y).^2,'k--',z,exp(-0.3*z),'k-.')
```

که پس از اجرا شکل ٦-٧- ب را نتیجه خواهد داد. توجه کنید در اینجا هر تابع با نوع خط و تعداد نقاط متفاوتی رسم شده است.

۶-۳ بر چسب گذاری روی نمودار و بالا بردن قابلیتهای گرافیکی

۶-۳-۱ بر چسب گذاری محورها و منحنی ها، عنوان دهی شکلها، اختصارات، نوشتن متن، و سایر نشانه ها.

اکنون در قالب یک مثال چگونگی افزودن قسمتهای مورد نیاز یک شکل را در قالب :

بر چسب گذاری محورها، عناوین شکلها، برچسب گذاری منحنی‌ها، عنوان دهی شکلها، اختصارات، پرکردن سطحها و قراردادن متنها در جاهای مورد نظر.

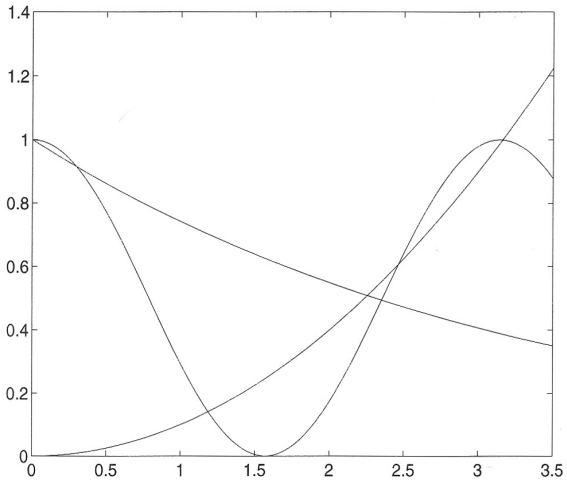
تغییر دادن محورها، خطوط منحنی و متنها.

استفاده از حروف یونانی، نشانه‌های ریاضیاتی و زیرنویس و بالا نویس توضیح خواهیم داد.

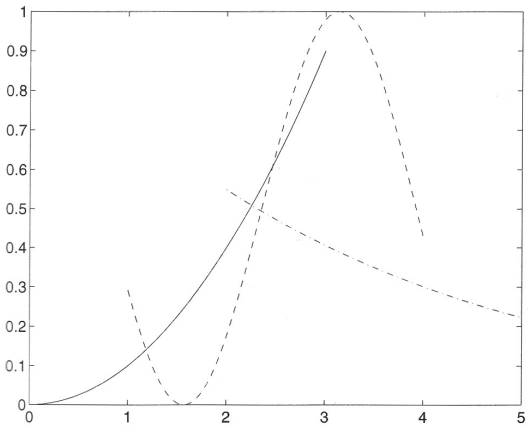
اجازه دهید دو منحنی متقاطع $cos(x)$ و $1/cosh(x)$ را در بازهٔ $0 \le x \le 6$ ترسیم کرده سپس آن را برچسب گذاری و عنوان دهی کرده و رابطهٔ دو منحنی را نشان دهیم. در این بازه، این دو منحنی در نقطهٔ $x = 4.73$ یکدیگر را قطع خواهند کرد. مثال بخش ٥-٥-١ را بخاطر بیاورید. همچنین خطی

عمودی گذرنده از نقطهٔ تلاقی رسم کرده و مقدار x نزدیک به این نقطهٔ تلاقی را نشان خواهیم داد. برنامه ای که موارد فوق را شامل می شود در زیر آورده شده است :

```
x=0:.05:6;
plot(x,cos(x),'k',x,1./cosh(x),
xlabel('x')
ylabel('Value of functions')
title('Visualizaion of two intersecting curves')
text(4.8,-1,'x=4.73')
text(2.1,.3,'1/cosh(x)')
```

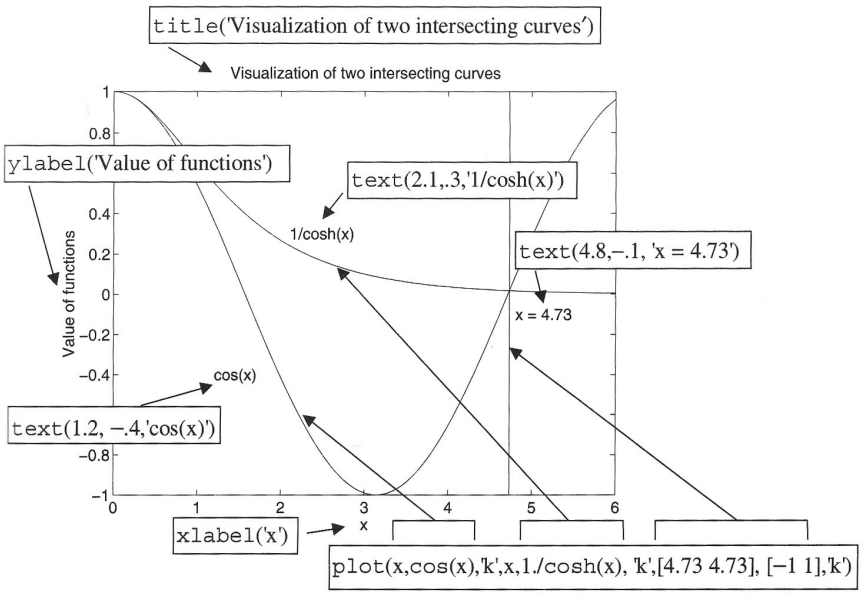


(الف)



(ب)

شکل ٦- ٧) (الف) سه تابع مختلف در یک بازهٔ یکسان (ب) سه تابع مختلف در سه بازهٔ مختلف



شکل ٦ - ٨) عباراتی که نمودار فوق را به همراه نشانه گذاریهای مربوط به آن ایجاد می کنند.

```
text(1.2,-.4,'cos(x)')
```

اجرای این برنامه شکل ٦-٨ را نتیجه خواهد داد. مقادیر مولفه‌های تعیین مکان متنهای مختلف پس از اجرای تابع *plot*، انتخاب می شوند. یعنی پس از اینکه در خط ابتدایی برنامه نوشته شد و شکل حاصل بدست آمد. سپس توابع *text* افزوده خواهند شد.

می‌توانیم نتایج بدست آمده را تغییر دهیم و فضای بین دو منحنی را در بازهٔ $0 \le x \le 4.73$ با رنگ کبود پر کنیم. جهت پرکردن فضای بین دو منحنی از دستور ؛

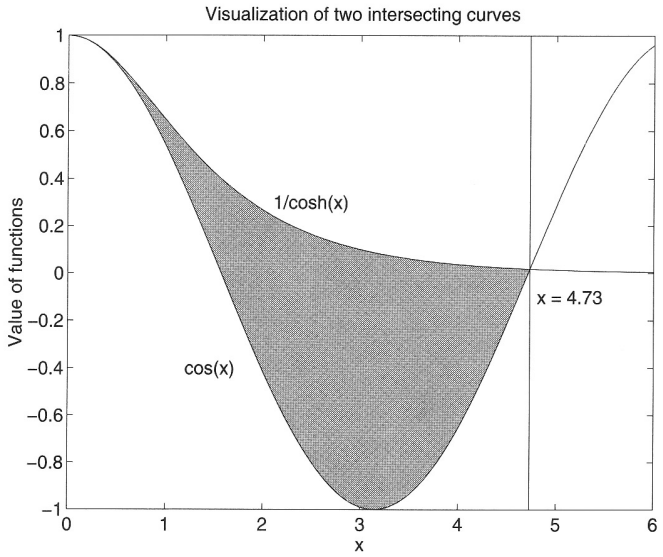
```
fill
```

که در آن لازم است تا دو منحنی به صورت یک چند ضلعی بسته در آیند. جهت انجام این کار، باید بازهٔ جدیدی برای x یعنی $0 \le x \le 4.73$، ایجاد کنیم.

برای اجرای این موارد باید دستورات زیر به برنامهٔ فوق اضافه شود :

```
xn=linspace(0,4.73,50);
hold on
fill([xn fliplr(xn)],[1./cosh(xn)fliplr(cos(xn))],'c');
```

نتایج افزودن دستورات جدید به قسمتهای قبل در شکل ٦-۹ نشان داده شده است. چند ضلعی متصل توسط تشکیل بردار [1./cosh(xn) fliplr(cos(xn))] ایجاد می شود، که الحاق منحنی بالا 1/cosh(x)، و معکوس بردار cosx که منحنی پایینی می‌باشد، خواهد بود. بردار محور x جدید [xhfliplr(xn)] متناظر با مقادیر بردار جدید می‌باشد که توسط الحاق مقادیر جدید x و مقادیر معکوس آن تشکیل می‌شود.



شکل ٦ – ۹) اصلاح شکل ٦ – ۸ که در آن سطح بین منحنی ها هاشور خورده است.

اگر بجای رنگ کردن کامل ناحیهٔ تلاقی دو منحنی، این منطقه با 20 خط عمودی با فاصلهٔ یکسان مشخص شود، در اینصورت برنامهٔ زیر را خواهیم داشت:

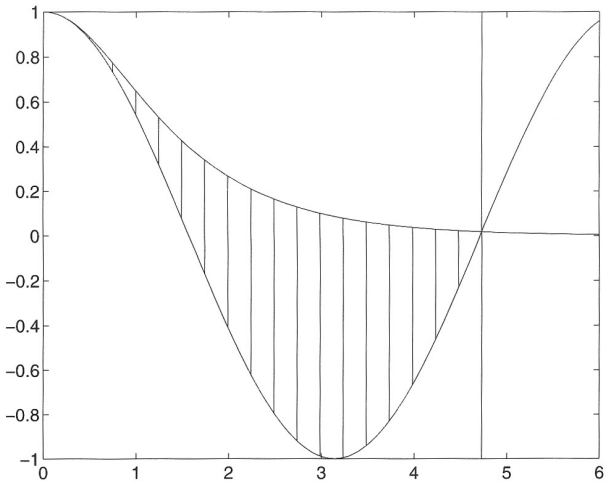
```
x=0:.05:6;
plot(x,cos(x),'k',x,1./cosh(x),'k',[4.73 4.73],[-1 1],'k')
```

```
hold on
xx=linspace(0,4.73,20);
plot([xx;xx],[cos(xx);1./cosh(xx)],'k-')
```

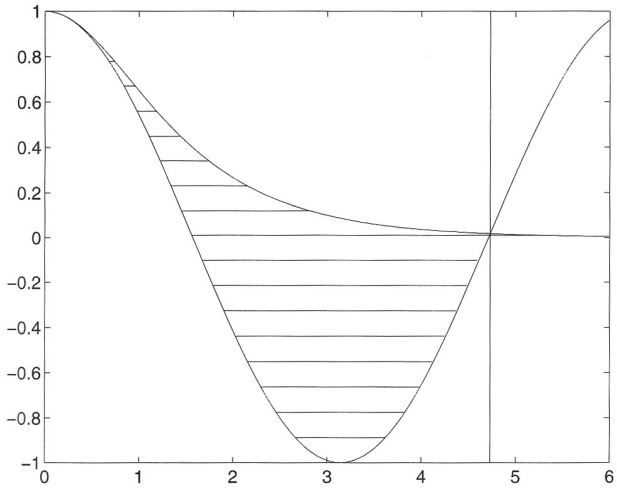
که پس از اجرا، نتیجهٔ مطابق شکل ٦-١٠- الف را خواهد بود.

جهت ایجاد 20 خط افقی با فاصلهٔ یکسان، باید از توابع معکوس $cos^{-1}(x)$ و $cosh^{-1}(x)$ استفاده کنیم. در این مورد فایل متنی به صورت زیر خواهد بود:

```
x=0:.05:6;
plot(x,cos(x),'k',x,1./cosh(x),'k',[4.73 3.73],[-1 1],'k')
hold  on
```

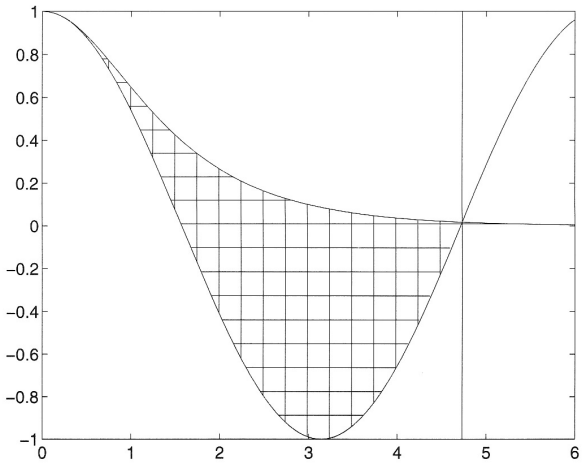


(الف)



(ب)

شکل ٦ – ١٠) سطح بین دو منحنی متقاطع هاشور خورده با خطوط با فاصلهٔ یکسان (الف) عمودی (ب) افقی



شکل ٦ – ١١) سطح بین دو منحنی متقاطع هاشور خورده توسط خطوط عمودی و افقی

```
y1=linspace(1,0.01,10);
plot([acos(y1);acosh(1./y1)],[y1;y1],'k-')
y2=linspace(0.01,-1,10);
```

```
plot([acos(y2);pi+fliplr(acos(y1))],[y2;y2],'k-')
```

که پس از اجرا شکل ٦ -۱۰- ب را نمایش خواهد داد. این دو برنامه را می توان جهت ایجاد هاشور، همانطور که در شکل ٦-۱۱ نشان داده شده است، با یکدیگر ترکیب نمود.

روش دیگری برای مشخص کردن منحنی‌ها در شکل ٦-۹ وجود دارد که استفاده از تابع *legend* می‌باشد.

تابع *text* را می‌توان به دفعات زیاد مورد استفاده قرارداد، در صورتیکه تنها یکبار می‌توان استفاده نمود. تعداد آرگونهای تابع *legend* برابر تعداد خطوط متفاوتی است که توسط تابع *plot* ترسیم می‌شوند. همچنین یک آرگومان دیگر (آرگومان انتهایی) نیز وجود دارد که استفاده از آن کاملاً اختیاری می‌باشد. هر آرگومان (بجز آرگومان نهایی) یک متغیر رشته‌ای متشکل از حروف و اعداد برای هر خط می‌باشد. مقدار آرگومان اختیاری (*1*، *2*، *3* یا *4*)، اختصارات *(legend)* را در یکی از چهار گوشهٔ شکل قرار خواهد داد. همچنین مقدار *(-1)* برای این آرگومان اختصارات را در سمت راست شکل و مقدار *(0)* آنرا در بهترین موقعیت، یعنی جایی که کمترین داده‌ها روی شکل وجود دارد، قرار می‌دهد. هنگامیکه آخرین آرگومان حذف شود، اختصارات در محل پیش فرض که گوشهٔ بالا سمت راست می‌باشد قرار خواهند گرفت. حتی کاربر می‌تواند این آرگومان را حذف کرده و از ماوس جهت تعیین مکان قرارگیری اختصارات استفاده کند. به سادگی می‌توان روی اختصارات کلیک کرده و آنرا به مکان دلخواه انتقال داد. اما این قرارگیری اختصارات توسط ماوس کاملاً موقتی می‌باشد، یعنی هنگامیکه شکل بسته می‌شود و دوباره ایجاد می‌شود اختصارات دوباره در مکان پیش فرض قرار خواهند گرفت و دوباره باید توسط ماوس تغییر مکان داده شوند.

در اینجا طریقهٔ استفاده از تابع *legend* را با دوباره امتحان کردن یک سری تغییرات در ترسیم کمیتهای ظاهر شده در برنامه جهت ایجاد شکل ٦-۸ نشان خواهیم داد. عبارات زیر را در نظر بگیرید:

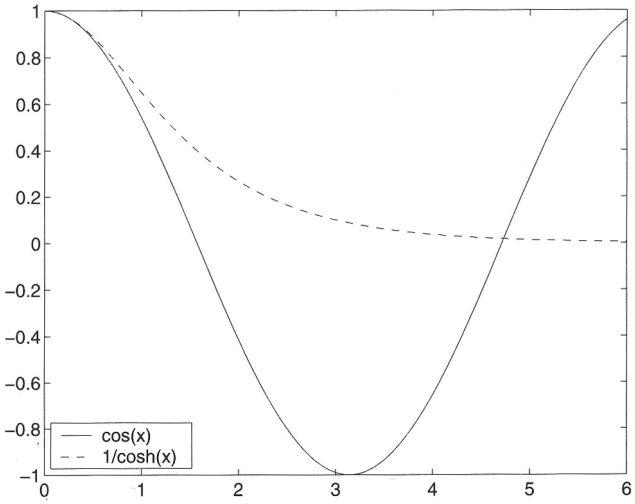
```
x=0:.05:6;
plot(x,cos(x),'k-',x,1./cosh(x),'k--')
legend ('cos (x)','1/cosh(x)',3)
```

برنامهٔ فوق شکل ٦-۱۲ را ایجاد خواهد کرد. آرگومان سوم هر مجموعهٔ سه تایی در تابع *plot*، نشان دهندهٔ این است که خطوط باید با رنگ مشکی ترسیم شوند که در اینجا تابع *cos(x)* با خطوط ممتد و تابع *1/cosh(x)* به صورت خطوط مقطع ترسیم شده است. آرگومانهای تابع *legend* به ترتیب آورده شده اند. اولین آرگومان مربوط به اولین منحنی ترسیم شده، دومین آرگومان مربوط

به دومین منحنی و می‌باشد. اگر از چندین تابع *plot* استفاده شود، در اینصورت این ترتیب با اولین آرگومان تابع *plot* دوم تا آخرین رشتهٔ تعیین کنندهٔ آخرین منحنی رسم شده در دستور *plot* قبلی ادامه خواهد یافت. به ازای هر دستور *figure* یا *subplot* تنها می‌توان یک دستور *legend* استفاده نمود. عدد *3* در تابع *legend*، اختصارات را در گوشهٔ پایین، سمت چپ شکل قرارخواهد داد.

شکل ٦-۱۲ را همچنین می توان توسط برنامهٔ زیر نیز ایجاد نمود:

```
x=0:.05:6;
plot(x,cos(x),'k-')
hold on
plot(x,1./cosh(x),'k--')
legend('cos(x)','1/cosh(x)',3)
```



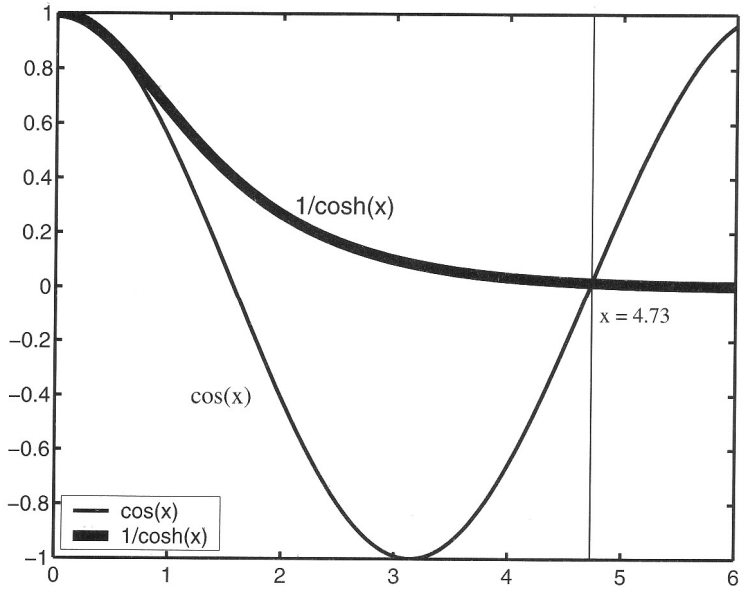
شکل ٦ – ۱۲) چگونگی استفاده از تابع *legend*

که در آن نوع و رنگ خطوط در تابع *plot* تعیین نشده است. تابع *legend* از نوع خط پیش فرض (پیوسته) و توالی پیش فرض رنگها استفاده خواهد کرد. یعنی خطوط موجود در قسمت اختصارات خطوطی پیوسته با رنگهای متفاوت خواهند بود. نرم افزار مطلب امکان اعمال تغییرات روی همهٔ قسمتهای تشکیل دهندهٔ یک نمودار را فراهم می‌کند. چند مورد از این تغییرات شامل، نوع و اندازهٔ فونت، اندازهٔ اعداد نشان داده شده روی محورها، به همراه ضخامت خطوط محورها و منحنی‌ها در این بخش مطلب موجود خواهد بود. مقدار پیش فرض ضخامت خطوط در مطلب *0.5* و همچنین

مقدار پیش فرض اندازهٔ فونت متن و بر چسب محورها *10* و نام فونت مربوط به هر دو *Helvetica* می‌باشد. اجازه دهید به برنامه‌ای که شکل ٦ –۸ را ایجاد نمود و در زیر آمده است، رجوع کنیم ؛

```
x=0:.05:6;
plot(x,cos(x),'k',x,1./cosh(x),'k',[4.73  4.73],[-1 1],'k')
text(4.8,-.1,'x=4.73')
text(2.1,.3,'1/cosh(x)')
text(1.2,-4,'cos(x)')
```

اکنون برنامهٔ فوق را بگونه‌ای اصلاح می‌کنیم که ضخامت خطوط کمی بیشتر شود، منحنی‌ها با خطوطی دارای ضخامت متفاوت ترسیم شوند، اختصارات در گوشهٔ پایین سمت چپ نمودار قرارگیرد، و سه متن درج شده روی شکل دارای دو نوع فونت متفاوت و سه اندازهٔ نایکسان باشند. برنامهٔ اصلاح شده به شکل ذیل خواهد بود:



شکل ٦ – ۱۳) تغییر اندازهٔ فونت، نوع فونت، و ضخامت خط

$$x = 0 : .05 : 6;$$

$$h = plot(x,cos(x),'k',x,1./cosh(x),'k',[4.73\ \ 4.73],[-1\ 1],'k');$$

```
text(4.8,-0.1,'x = 4.73',' fontname','times',' fontsize',14)
text(2.1,0.3,'1/cosh(x)',' fontsize',16)
text(1.2,-0.4,'cos(x)',' fontsize',16,' fontname','times')
set(gca,' fontsize',14,'linewidth',2)
PropertyName={'LineWidth','LineWidth','LineWidth'};
PropertyValue={2.5,2.5,2.5;7,7,7;1,1,1};
set(h,PropertyName,PropertyValue)
[legendhandle objecthandle]=legend('cos(x)','1/cosh(x)',3);
set(objecthandle(1),' fontsize',14,'color','r')
```

که پس از اجرا، شکل ٦ -۱۳ را نتیجه خواهد داد. دستور h=plot(…) در بردار ستونی h، سه مقدار قرار خواهد داد، که به آنها دستگیره گفته می‌شود، که مطلب را قادر می کند تا به نمودارهای نشان داده شده در شکل ها دسترسی پیدا کند. در اینجا سه مقدار برای بردار h وجود دارد، چون در این مورد سه نمودار متفاوت $cos(x)$، $1/cosh(x)$ و خط عمودی در $x = 4.73$ ترسیم شده است. با دانستن شمارهٔ مشخصهٔ هر منحنی که به ترتیب ایجاد منحنی‌ها به آنها نسبت داده می‌شود، از آخرین تابع *set* استفاده شده در برنامه، جهت تغیر ضخامت هر خط به مقدار تعیین شده استفاده می‌شود. همچنین اولین تابع *set* در برنامه از تابع *gca* (قرار دادن دستگیره روی محور فعلی) که برای محور مورد نظر دستگیره‌ای قرار می‌دهد. استفاده می‌کند. در این مورد، دو مشخصه را در مورد محورها که عبارتند از : ضخامت خط و اندازهٔ فونت را تغییر داده‌ایم. در مورد برچسب‌ها دستگیره‌های گوناگونی روی خطوط *(legendhandle)* و متن‌ها *(object handle)* قرار می‌گیرد. توسط *objecthandle(1)* می‌توان به مشخصات متن‌ها در بخش اختصارات دسترسی پیدا کرد، در اینجا ما اندازهٔ فونت حروف بخش اختصارات را تغییر داده‌ایم، همچنین رنگ آنها را نیز قرمز تعیین نموده‌ایم.

تغییراتی که قرار است روی مشخصات متن‌ها، منحنی‌ها و محورها اعمال شود را می‌توان مستقیماً در پنجرهٔ شکل با استفاده از آیکن‌های مناسب در بالای پنجره و یا انتخاب عملیات مناسب از منوی *tools*، انجام داد. پس از اعمال تغییرات می‌توان شکل را ذخیره نمود. اما، اگر دوباره این برنامه‌ها و توابع که جهت ایجاد شکل نوشته شده‌اند، اجرا شوند، این تغییرات و مطالب افزوده شده، از بین خواهد رفت و دوباره باید ایجاد شوند.

قابلیت دیگری که می‌تواند در دستور *text* گنجانیده شود، *rotation* می‌باشد که متن مورد نظر را به اندازهٔ زاویهٔ θ بر حسب درجه نسبت به محور افقی خواهد چرخاند.

این مطلب در شکل ٦ -٢٠ از بخش ٦-٣-٦ نشان داده شده است.

همچنین می توان نمودار را با استفاده از حروف کوچک و یا بزرگ یونانی زیر نویس و بالا نویس و نشانه‌های ریاضیاتی نسبتاً زیادی، برچسب گذاری نمود. این برچسب گذاریها می‌توانند توسط دستورات *xlabel، ylabel، zlabel، text، legend* و *title* انجام شوند. اصول فرمت دهی که بدنبال زبان *Latex*[۱] می‌آیند و جز دستور ویرایش متن مطلب نمی‌باشند، قبلاً مورد بررسی قرار گرفته‌اند.

زیر نویسها با استفاده از علامت (_) و بالا نویسها با استفاده از علامت توان (^) ایجاد می‌شوند. ایجاد حروف یونانی توسط حجی کردن حرف و قرار دادن علامت تقسیم وارون (\) قبل از آن، همانگونه که در جدول ٦ -١ نشان داده شده است، میسر می باشد. جهت دستیابی به حروف بزرگ یونانی کافیست که کاراکتر اول هجی آن حرف را با حرف بزرگ بنویسیم. از آنجا که بسیاری از حروف بزرگ یونانی مانند حروف بزرگ انگلیسی نوشته می‌شود، تنها آن دسته از حروفی که شیوهٔ نوشتنشان به شیوهٔ یونانی با انگلیسی متفاوت است در جدول ٦ -١ آورده شده اند. مابقی حروف بزرگ یونانی به سادگی با استفاده از حروف بزرگ متناظر انگلیسی شان قابل دستیابی خواهد بود.

ایجاد نشانه های ریاضیاتی از طریق هجی کردن آنها بعد از علامت تقسیم وارون (\) میسر می‌باشد. تعدادی از نشانه‌های متداول تر مطلب در جدول ٦-١ آورده شده است. در حالت کلی باید، مجموعهٔ دستورات الحاق شده در میان علامت کوتیشن قرار بگیرند. هنگامیکه مجموعه‌های مشخصی از نشانه‌ها باید در کنار یکدیگر قرار گیرند، مثل وقتی که کاراکتریهای در کنار هم بعنوان توان استفاده می‌شوند، این نشانه ها باید در میان یک جفت قلاب ({ }) قرارگیرند. اکنون این مطلب را در قالب مثال زیر نشان می دهیم:

---

۵ - برای مثال به کتاب

L. Lamport, LaTeX: A Document Preparation System, Addison - Wesley, Reading, MA, 1987.

مراجعه کنید.

جدول ۶-۱ حروف بزرگ و کوچک یونانی و برخی نشانه‌های ریاضی

حروف کوچک		حروف بزرگ		ریاضی	
نشانه	شیوهٔ نوشتن	نشانه	شیوهٔ نوشتن	نشانه	شیوهٔ نوشتن
α	\alpha	Γ	\Gamma	$\leq$	\leq
β	\beta	Δ	\Delta	$\geq$	\geq
γ	\gamma	Θ	\Theta	$\neq$	\neq
δ	\delta	Λ	\Lambda	$\pm$	\pm
ε	\epsilon	Ξ	\Xi	$\times$	\times
ς	\zeta	Π	\Pi	∞	\inf ty
η	\eta	Σ	\Sigma	$\sum$	\sum
θ	\theta	Y	\Upsilon	$\int$	\int
ι	\iota	Φ	\Phi	$\div$	\div
κ	\kappa	Ψ	\Psi	$\sim$	\sim
λ	\lambda	Ω	\Omega	$\leftarrow$	\leftarrow
μ				$\uparrow$	\uparrow
ν	\nu			$\circ$	\circ
ξ	\xi			$<<$	\ll
o	o			$>>$	\gg
π				$'$	\prime
ρ	\rho			$\Leftarrow$	\Leftarrow
σ	\sigma			$\angle$	\angle
τ	\tau			$\surd$	\surd
υ	\upsilon			#	\#

\$	$	\phi	ϕ
\%	%	\chi	χ
\&	&	\psi	ψ

می‌خواهیم تابع زیر را به ازا $\beta = 3$ در بازهٔ $1 \le \Omega_1 \le 2$ رسم کرده محورها را بطور مناسب برچسب گذاری کنیم:

$$g_2 = 1 + e^{-\Omega_1^{\beta}}$$

برنامه به صورت زیر خواهد بود:

```
Omega1 =linspace(1,2);beta=3;
plot(omega1,1+exp(-omega1.^beta),'k')
title('plot of g-2verus \omega1-1for \beta=3')
ylable('g_2')
xlable('\omega-1')
text(1.2,1.2,'g-2=1+e^{-\omega-1^\beta}','fontsize',16)
```

نتایج اجرای این برنامه در شکل ٦ -١٤ آمده است.

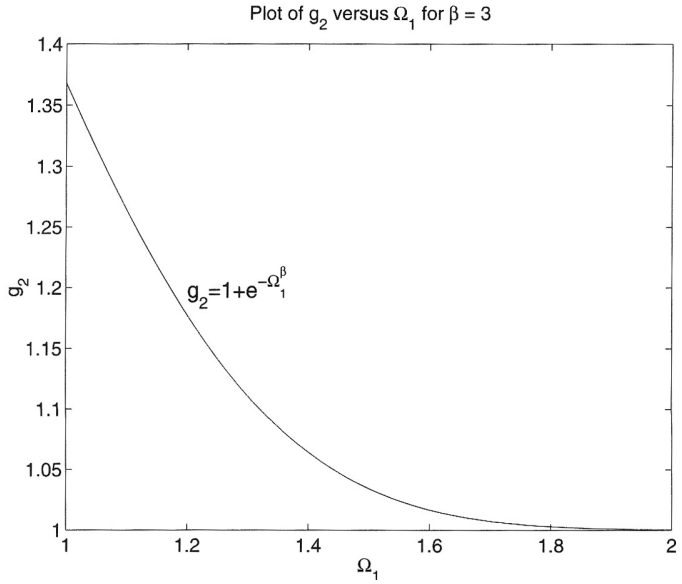
۶ -۳ -۲ تکرار کردن منحنی ها: نمایش *cot(x)* در بازهٔ $0 \le x \le m\pi$

در اینجا تابع *cot(x)* را در بازهٔ $0 \le x \le m\pi$ که در آن m یکی از اعداد 2، 3، ... یا 6 می‌باشد و توسط کاربر تعیین می‌شود، رسم خواهیم کرد. در حالیکه $m=2$ انتخاب شود، نتیجه مشابه شکل ٦- ١٥ خواهد بود. حدود محورy، ± 8 انتخاب شده است، و حدود محور x از 0 تا $m\pi$ متغیر می‌باشد. از آنجا که $cot(0) = \infty$ و $cot(\pi) = -\infty$ می‌باشد، بازهٔ ترسیم شامل مقادیر فوق نخواهد بود. از طرفی از آنجا که تابع کتانژانت خودش را در هر $m\pi$ تکرار می‌کند، لذا لازم است تا آنرا یکبار در بازهٔ $0 < x < \pi$ محاسبه نموده سپس از این مقادیر برای ترسیم در نواحی $(m-1)\pi < x < m\pi$ و $m > 1$ براحتی با افزودن محور x به اندازهٔ $x+(m-1)\pi$، استفاده نماییم.

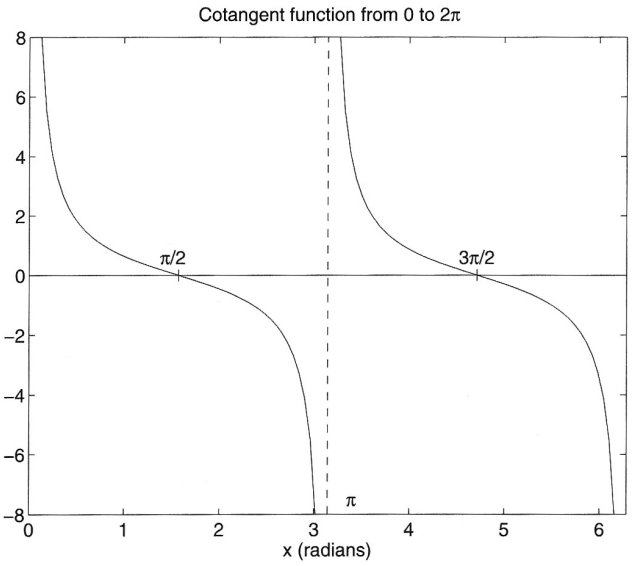
برنامه به صورت زیر خواهد بود:

```
m=input('enter number of repetitions of cot functions (integer from
2 to 6)=');
the =linspace(0.12,pi-0.12,50);
```

```
ct=cot(the);
hold on
for n=1:m
 plot([(2*n-1)*pi/2(2*n-10*pi/2],[-8/40 8/40 ],'k-')
 if n==1
text((2*n-1)*pi/2-pi/(8*m),.6,'\pi/2')
  text(n*pi+pi/(8*m);-7.5,'\pi')
 else
  text((2*n-1)*pi/2-pi/(8*m),0.6,[num2str((2*n-1),2) '\pi/2'])
  if n<m
  text(n* pi+pi/(8*m);-7.5,[num2str(n,1) '\pi'])
 end
end
if n==m
plot(the+(n-1)*pi,ct,'k-')
else
 plot( the+(n-1)*pi,ct,'k-',[n*pi nn*pi],[-8 8],'k—')
end
end
plot ([0m*pi],[0 0],'k-')
axis([0m*pi-8 8])
xlabel('x(radians)')
ylabel('cot(x)')
title(['cotangant function from 0 to ' num2str(m,1),'\pi'])
box on
```



شکل ٦ – ١٤) نشانه گذاری توسط اندیسهای بالا، پایین و حروف یونانی



شکل ۶ – ۱۵) رسم تابع $cot(x)$ در بازهٔ 0 تا 2π

همانطور که در برنامهٔ فوق دیده می‌شود تمامی دستورات برنامه بجز دو تای آنها که جهت محاسبهٔ *cot(x)* استفاده شده‌اند، برای برچسب گذاری شکل بهره برداری شده‌اند.

۶ -۳- ۳ ترسیم نمودارهای قطبی: الگوی میدان تشعشعی یک منبع صدا

فشار صوت نرمالیزه شده در یک فاصلهٔ دور از مرکز یک پیستون دایروی که در داخل یک بَفِل نامحدود با فرکانس fدر حال نوسان است، توسط رابطهٔ زیر داده می‌شود:

$$p(r,\theta)=\left|\frac{J_1(ka\theta)}{ka\theta}\right| \qquad ka^2 << r \quad and \quad a << r$$

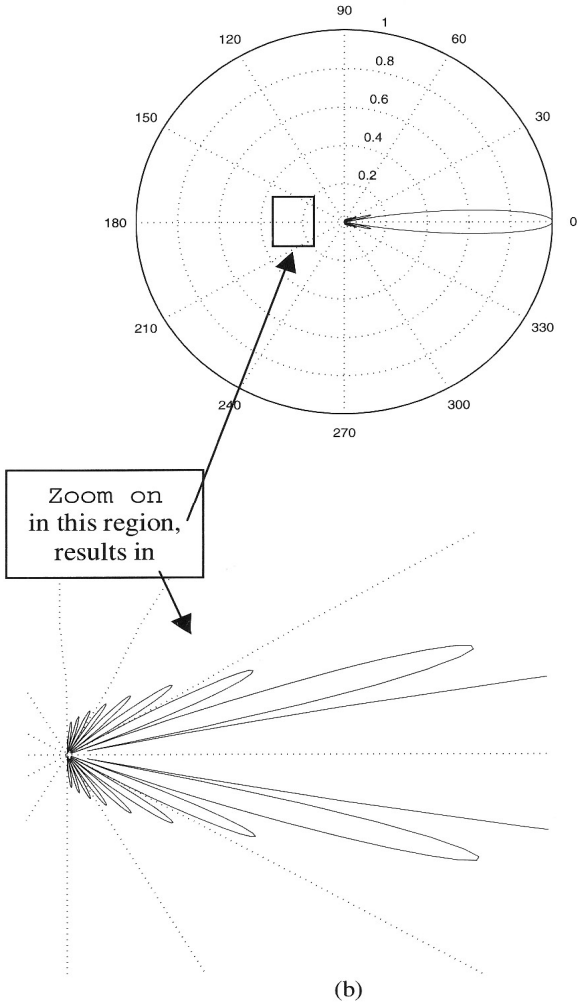
که در آن r فاصلهٔ شعاعی از مرکز پیستون، θ زاویهٔ r نسبت به صفحهٔ بَفِل، k شمارهٔ موج، a شعاع پیستون و $J_1(x)$ تابع بسل نوع اول از مرتبهٔ 1 می‌باشد. شمارهٔ موج معکوس، طول موج صوت در فرکانس f می‌باشد. بنابراین ka کمیتی بدون بعد خواهد بود. این مدل یک تقریب نسبتاً خوب جهت پراکندگی زاویه ای صدا از یک بلند گو می‌باشد.

اکنون می‌خواهیم یک نمودار قطبی مربوط به الگوی تشعشع نرمالیزه شده به ازای $ka=6\pi$، هنگامیکه $-\pi/2<\theta<\pi/2$ می‌باشد، ایجاد کنیم. همچنین از این حل جهت نشان دادن چگونگی استفاده از تابع *zoom* استفاده خواهیم کرد. تابع *zoom* در این مورد مناطقی از نمودار که با شیوهٔ نمایش معمولی مطلب قابل دیدن نیستند را رویت پذیر می‌سازد. برنامه به صورت زیر می‌باشد:

```
theta=linspace(-pi/2,pi/2,300);
rad=abs(bessrlj(1,6*pi*theta)./(6*pi*theta));
polar(theta,rad/max(rad))
zoom  on
```

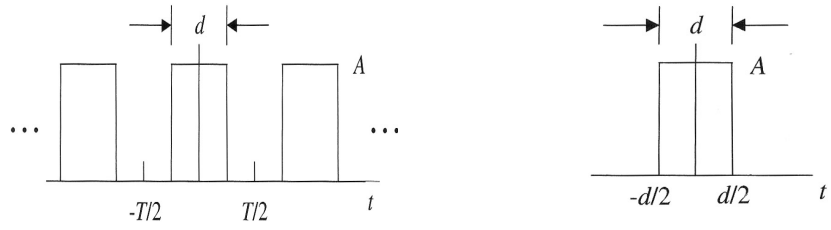
پس از اجرای برنامهٔ فوق نتیجهٔ نشان داده شده در شکل ٦-١٦- الف نشان داده خواهد شد. توجه کنید که مقادیر θ بگونه‌ای هستند که $ka\theta\neq 0$ می‌باشد. تابع max، ماکزیمم مقدار بردارrad را بدست می‌آورد، لذا نسبت rad/max(rad) الگوی تشعشی نرمالیزه شده می‌باشد که دارای مقدار ماکزیمم 1 می‌باشد. تابع zoom on به کاربر اجازه می‌دهد تا از کلیک چپ ماوس جهت تعریف یک محدودهٔ مستطیلی، که پس از رها کردن کلیک ماوس، تمام فضای ترسیم را پر خواهد کرد، استفاده کند. این محدوده توسط مستطیل نشان داده شده در شکل ٦-١٦- الف تعیین شده است و پس از رها شدن کلیک ماوس منجر به نمایش شکل ٦-١٦- ب خواهد شد. جهت بازگرداندن شکل به اندازهٔ

اصلی‌اش کلیک راست ماوس باید چند بار در حالیکه مکان نما در پنجرهٔ شکل قراردارد، فشرده شود. همچنین کاربر می‌تواند، فرمان بزرگنمایی را با تایپ کردن zoom off و یا zoom غیر فعال کند. این فرمان را می‌توان با استفاده از آیکن pan and zoom در پنجرهٔ شکل بجای تایپ دستور zoom on، اجرا نمود.



(b)

شکل ٦ – ١٦) (الف) نمایش قطبی یک الگوی تشعشعی؛ (ب) محدودهٔ بزرگ شده



شکل ٦ – ١٧) (الف) پالس مستطیلی پریودیک پیوسته؛ (ب) پالس مستطیلی منفرد.

۶ – ۳ – ۴ شکل های چند تایی: ترسیم طیفی یک پالس پریودیک و یک پالس منفرد.

سری متناوب زیر مربوط به پالس مستطیلی نشان داده شده در شکل ٦–٧– الف که دوام پالس آن d و پریودش t می‌باشد را در نظر بگیرید. این سیگنال را می‌توان بصورت زیر نمایش داد (مثال ٢–٣ را بیاد بیاورید):

$$g(t)=\frac{f(t)T}{Ad}=c_0+2\sum_{n=1}^{\infty}c_n\cos(n\omega_0 t)$$

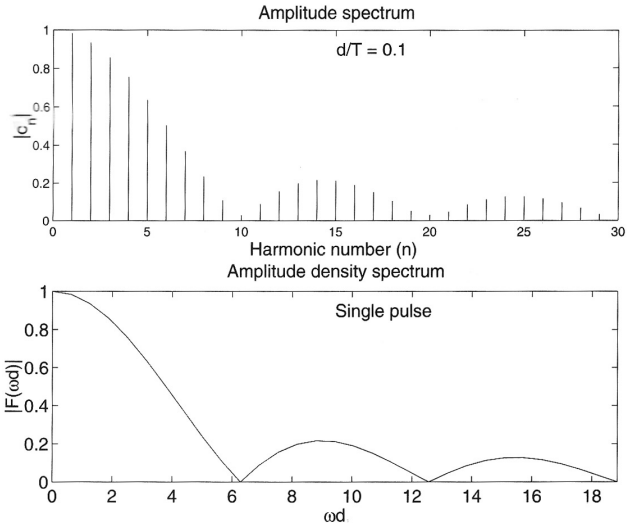
که در آن $\omega_0=2\pi/T$ و

$$c_0=1$$

$$c_n=\frac{\sin(n\pi d/T)}{(n\pi d/T)} \qquad n=1,2,...$$

می‌باشد. در رابطهٔ فوق c_n ها مبین دامنه‌های نرمالیزه شدهٔ هر تناوبی که شامل سیگنال می‌شود، می‌باشد. هنگامیکه کاربر $|c_n|$ را به صورت تابعی از n ترسیم می‌کند، نمودار حاصل، طیف دامنه‌ای سیگنال نامیده می‌شود. نمودار طیفی تنها در فرکانسهای $n\omega_0$، دارای محتوای فرکانسی است و در بقیهٔ جاها، دارای مقدار صفر می‌باشد. این مطلب به وضوح مشخص است که هرگاه رابطهٔ زیر ؛

$$\frac{n\pi d}{T}=m\pi \qquad or \qquad n=\frac{m}{(d/T)}$$



شکل ٦ – ١٨) طیف فرکانسی (الف) پالس پریودیک نشان داده شده در شکل ٦ – ١٧ – الف؛ (ب) پالس مستطیلی منفرد نشان داده شده در شکل ٦ – ١٧ – ب

برقرار باشد، $c_n = 0$ خواهد بود. از طرف دیگر، اگر پالسی منفرد، همانگونه که در شکل ٦-١٧-ب ترسیم شده است، داشته باشیم، طیف فرکانسی آن بصورت ؛

$$F(0) = 1$$

$$F(\omega) = \frac{G(\omega)}{Ad} = \frac{sin(\omega d/2)}{(\omega d/2)} \qquad \omega > 0$$

خواهد بود. مقدار قدرمطلق $G(w)$ ، نمودار چگالی دامنه نامیده می شود.

اکنون می خواهیم دو نمودار ایجاد کنیم، یکی $|c_n|$ برحسب n، که در آن $n=1,2,...,30$ و $d/T = 0.1$ می‌باشد و دومی $|F(\omega d)|$ برحسب ωd، که در آن $0 \le \omega d \le 6\pi$ می‌باشد. جهت ایجاد نمودار اول از تکنیک بیان شده در بخش ٦-٢-٢ جهت ترسیم مجموعه‌ای از خطوط مقطع مستقیم استفاده خواهیم کرد. این دو شکل بگونه‌ای که یکی بالای دیگری قرار گیرد، نشان داده خواهند شد. دقت کنید که ملاحظات خاصی جهت جلوگیری از پیش آمدن حالت تقسیم بر صفر باید لحاظ گردد. نتایج برنامهٔ زیر در شکل ٦-١٨ نشان داده شده است.

```
n=1:30;
cn=[1abs(sin(0.1*pi*n)./(0.1*pi*n))];
n=[0 n];
subplot(2,1,1)
plot([n;n],[zeros(1,length(cn));cn],'k')
xlabel(Harmonic number(n)')
ylabel('|c_n|')
text(15,.9,'d/T=0.1')
title('Amplitude spectrum')
w=pi/5:pi/5:6*pi;
subplot (2,1,2)
plot ([0 w],[1abs(sin(w/2)./(w/2))],'k')
axis([06*pi  0  1])
xlabel('\omegad')
ylabel('|F(\omegad)|')
text(3*pi,0.9,'Single pulse')
title('Amplitude density spectrum')
```

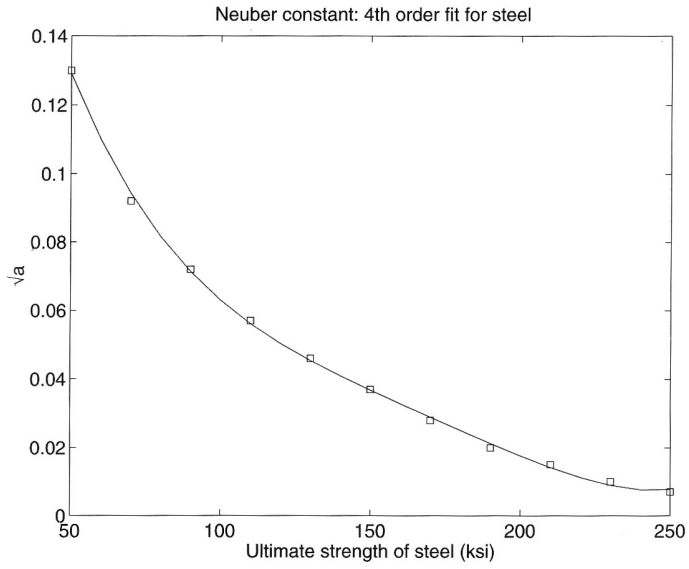
جهت جلوگیری از حالت تقسیم بر صفر، تکنیک زیر استفاده خواهیم کرد. در ابتدا بردار اعداد هارمونیک بصورت n=1:30 تعریف می شود. بنابراین براحتی قادر خواهیم بود $|c_n|$ را در این بازه با استفاده از تقسیم نقطه ای محاسبه کنیم. سپس به ترتیب دو بردار شامل n=0 و $c_0 = 1$، همانگونه که در خطوط دوم و سوم نشان داده شده است، ایجاد می‌کنیم. روش مشابهی جهت محاسبهٔ $|F(\omega)|$، استفاده می‌شود. توجه داشته باشید که در این روش نیازی به برنامه‌نویسی پیچیده نخواهد بود.

۶-۳- ۵ منحنی های چند تایی: حساسیت شکاف برای فولاد

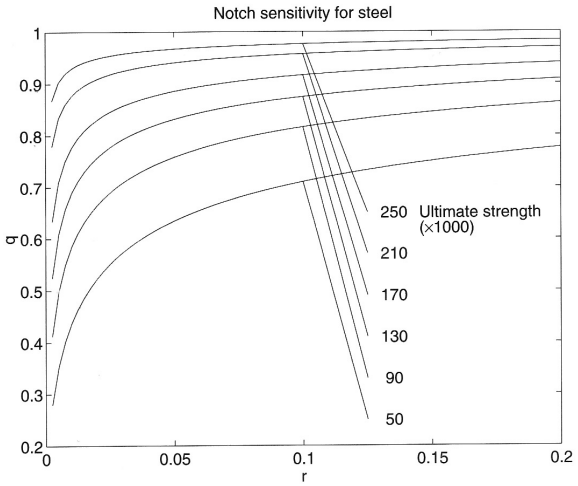
اکنون به مثال داده شده در بخش ٥-٦-١ باز می‌گردیم و ضریب حساسیت شکاف، q را در بازهٔ $50 \le S_u \le 250$ و $0 < r < 0.2$ رسم می کنیم. جهت خواناتر کردن برنامه، تابعی برای داده هایی که باید بردازش شوند، ایجاد می کنیم، بنابراین ؛

```
function ns = Data Neuber
ns=[50,.13,70,092;90,.072;110,.057;130,.046;150,.037;…
       170,.028 ;190,.020;210,.015;230,.010;250,.007];
```

برنامهٔ زیر شامل دو قسمت می‌شود. اولین قسمت ضرایب چند جمله ای درجهٔ چهار که جهت برازش این داده‌ها استفاده می‌شود را بدست می‌آورد و سپس نقاط متناظر با این داده‌ها و چند جمله‌ای که این نقاط را برازش می‌کند، را ترسیم می‌نماید. قسمت دوم از چند جمله‌ای‌های بدست آمده در قسمت اول جهت ایجاد خانواده‌ای از منحنی‌ها که نشاند هندهٔ حساسیت شکاف ، q ، برحسب شعاع شکاف برای چندین مقدار از مقاومت نهایی فولاد، S_u ، می‌باشند، استفاده می‌کند. اجرای این برنامه در شکل ٦-١٩- الف و ٦-١٩- ب نشان داده شده است.

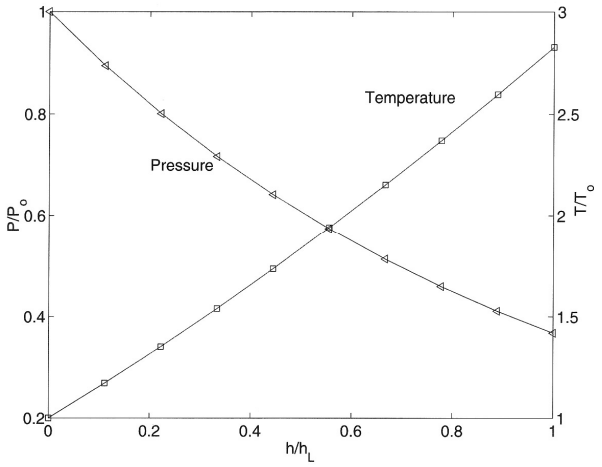


(الف)



(ب)

شکل ٦ – ۱۹) (الف) ثابت نیوبر برای فولاد (ب) حساسیت شکاف برای فولاد بصورت تابعی از شعاع شکاف r



شکل ٦ – ۲۰) دو کمیت متفاوت رسم شده توسط تابع *plotyy*

```
st=50:10:250;skip=1:2:1 1;loc=0.25:0.08:0.65:
ncs=DataNeuber;
p=polyfit(ncs(:,1),ncs(:,2),4);
figure(1)
plot(st,polyval(p,st),'k',ncs(:,1),ncs(:,2),'ks')
title(Neuber constant: 4th order fit for steel')
xlabel('Ultimate strength of steel (ksi)')
ylabel(\surda')
figure(2)
[s,r]=meshgrid(ncs(skip,1),0.0025:0.0025:0.2);
notch=inline('1./(1+polyval(p,s)./sqrt(r))','p','s','r');
plot(r,notch(p,s,r),'k')
hold on
plot([repmat(0.125,1,6);repmat(0.1,1,6)],[loc;notch(p,ncs(skip,1)'.0
.1)],'k')
text(repmat(0.13,1,6),loc,num2str(ncs(skip,1)))
text(0.145,0.65,'ultimate strength')
text(0.145,0.62,(\times1000)')
xlabel('r')
ylabel('q')
```

```
title('Notch sensitivity for steel')
```

تابع meshgrid دو ماتریس با ابعاد (6×80) ایجاد می‌کند که سطرهای آن همان مقادیر r و ستونهایش مقادیر S_u می‌باشند. از آنجا که دستورات در تابع inline تحت عنوان notch با استفاده از علامت نقطه نوشته شده اند، می توانیم این ماتریسها را بعنوان آرگومانهای مناسب آنها وارد کنیم. ترسیم خطوط مشخص کنندهٔ منحنی‌ها، توسط آخرین تابع plot انجام می‌شود. در اینجا از تکنیک مشابهی که قبلاً جهت کشیدن یک سری از خطوط مستقیم مقطع استفاده گردید، استفاده می‌کنیم. اما توجه کنید که مجبوریم تا بردار ncs(:,1) را از شکل ستونی به سطری تبدیل می‌کنیم. جهت قرار دادن متن در انتهای این خطوط، برداری متشکل از مختصه های x، y، متناظر با متن داده شده توسط تبدیل کردن ستون اول ncs(:,1) به یک رشته با استفاده از تابع num2str، ایجاد می‌کنیم. هر عنصر دیگر ncs با افزایش زیر نویس با گام 2 قابل دستیابی خواهد بود.

۶ -۳ -۶ منحنی های چند تایی با محورهای y متفاوت : تابع Plotyy

دو کمیت بدون بعد، نسبت فشار و نسبت دما که هر یک تابعی از نسبت ارتفاع می‌باشد را در نظر بگیرید. فرض کنید این نسبت ها توسط روابط زیر محاسبه شوند:

$$\frac{T}{T_0}=\left(1+\frac{h}{h_L}\right)^{1.5} \quad and \quad \frac{p}{p_0}=e^{-h/h_L}$$

برنامهٔ زیر نشان می‌دهد که چگونه می‌توان این دو نسبت را در بازهٔ $0 \le h/h_L \le 1$ روی یک نمودار بر چسب گذاری شده توسط تابع *plotyy* ترسیم نمود. پس از اجرای برنامه شکل ٦ -۲۰ نشان داده خواهد شد.

```
hoverhL=linspace(0,1,10);
ToverTo=(1+hoverhL).^1.5;
PoverPo=exp(-hoverhL);
[ax,h1,h2]=plotyy(hoverhL, PoverPo, hoverhL, ToverTo);
xlabel('h/h-L')
ylabel(p/p-o)
v=axis;
text(v(2)*1.06,v(3)+v(4)-v(3))/2,'T/T-O','rotation',90)
text(v(1)+v(2)-v(1))/5,v(3)+v(4)-v(30)/1.6,'pressure')
text (v(2)/1.6,v(4)/1.2,'Temperature')
```

```
set(h2,'marker','s')
set(h1,'marker','<')
```

تابع plotyy تک تک منحنی‌ها را رسم کرده و مقادیر اندازه شدهٔ صحیح برای هر یک از آنها را باز می گرداند، که این کار توسط نسبت دادن محور عمودی سمت چپ به اولین جفت متغیرها در آرگومانهای تابع plotyy و نسبت دادن مقادیر محور عمودی سمت راست به دومین جفت از متغیرها انجام می‌شود. به هر حال کاربر می‌تواند محور عمودی سمت چپ را تنها با استفاده از تابع ylabel بر چسب گذاری کند. همچنین تابع plotyy به کاربر اجازه نمی‌دهد تا نوع خطوط را برای هر یک از منحنی‌ها انتخاب کند. لذا جهت بر چسب گذاری محور عمودی سمت راست و اینکه بتوان به هر منحنی ویژگیهای خاص خودش را نسبت داد، از شکلی از تابع plotyy که در برنامه استفاده شده است. بهره برده‌ایم. کمیت ax(1) و ax(2)، به ترتیب دستگیره‌های محورهای سمت راست و چپ می‌باشند. کمیتهای $h1$ و $h2$ همانطوری که به ترتیب ظاهر شدنشان در دستور plotyy آمده‌اند، دستگیره‌های منحنی‌های اول و دوم می‌باشند. تابع set در اینجا جهت تعیین مشخصات Marker که مبین خصوصیات دستگیره می‌باشد، استفاده می‌شود. اولین تابع set، نقاط ترسیم شده را بصورت مربع‌هایی نشان می‌دهد. در دومین تابع set، نقاط ترسیم شده بصورت مثلث‌هایی که نوک آنها به طرف چپ است، نشان داده می‌شوند. عبارت rotation در اولین دستور Text، متن را به اندازهٔ 90 درجه می‌چرخاند.

۶-۳-۷ خواندن مقادیر عددی از روی نمودارها: تابع *ginput*

مطلب دارای این قابلیت است که می تواند مولفه‌های *(x,y)* را مستقیماً از روی یک نمودار توسط تابع *ginput*، ذخیره کند. این تابع می‌تواند مستقیماً در پنجرهٔ فرمان مطلب تایپ شود و یا بخشی از یک برنامه باشد. در اینجا شیوهٔ استفاده از این تابع را در قالب برنامه‌ای که از تابع *Damp sin wave* استفاده می‌کند و قبلاً در بخش ۵ -۶ -۳ ساخته شد، نشان خواهیم داد. هدف ما تعیین دورهٔ تناوب میرا شده از روی نمودار با استفاده از میانگین نقاط متوالی مربوط به ماکزیمم محلی دامنه و میانگین نقاط متوالی صفرهای متلاقی تابع خواهد بود. سپس این مقادیر متوسط بدست آمده، با مقادیر تحلیلی بدست آمده در ذیل با پریود T مقایسه خواهند شد.

$$T = \frac{2\pi}{\sqrt{1-\xi^2}}$$

که در آن ٤ ضریب میرایی بدون بعد می‌باشد.

این تابع معمولاً به شکل زیر استفاده می‌شود ؛

[x,y]=ginput

که در آن x و y بردارهایی شامل مقادیر متناظر با محورهای مختصات مکان نما (مرکز علامت ضربدر) روی شکل در مکانی که کلیلک چپ ماوس فشرده شوند، خواهند بود. مکان نما می‌تواند به تعداد دفعات دلخواه اطلاعات مختصاتی نقاط را دریافت نماید. پس از اینکه امکان آخرین نقطه ثبت شد، کاربر باید کلید Enter را فشار دهد. جهت برخورداری از کیفیت خوب گرافیکی در هنگام استفاده از این روش کاربر باید پنجرهٔ گرافیکی را قبل از اجرای این عملیات به حداکثر برساند. اگر $\xi = 0.1$ و $0 \le t \le 30$ باشد، در اینصورت برنامهٔ زیر را خواهیم داشت:

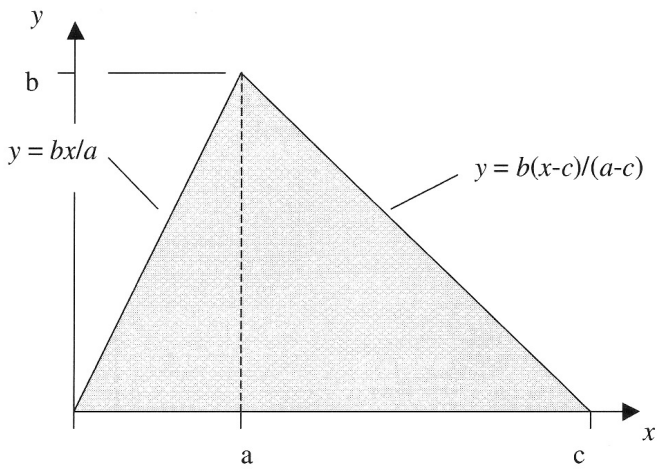
```
tau=linspace(0,30,200);xi=0.10;
plot(tau,DampedSine Wave(tau,xi),'k')
grid on
[time,amp]=ginput;
disp('   tau     ampl')
disp(num2str([time ampl))
disp(['Analytically determined period='num2str(2*pi/sqrt(1-xi^2))])
disp(['Averagen                  priod                  graphically
obtained='num2str(mean(diff(tim)))])
disp(['Standard deviation='num2str(mean(diff(tim)))])
```

پس از اجرای برنامهٔ فوق و بدست آوردن پنج دادهٔ مربوط به نقاط تلاقی صفر خواهیم داشت:

```
tau             ampl
1.68539    -0.00137931
7.99358    -0.00137931
14.3018    -0.00137931
20.61        -0.00137931
26.9181    -0.00137931
Analytically determined period =6.3148
Average period graphically obtained=6.3082
'Standard deviation=2.2933e-015
```

پس از بدست آوردن پنج دادهٔ مربوط به نقاط ماکزیمم محلی، خروجی زیر را بدست می‌آوریم:

```
   tau             ampl
    0             0.995862
6.30819         0.528276
12.5682         0.28
18.9246         0.147586
25.2327         0.0772414
Analytically determined period =6.3148
```



شکل ٦ – ۲۱) مثلثی که مساحتش توسط اعداد تصادفی تقریب زده می شود.

```
Average period graphically obtained=6.3082
'Standard deviation=0.039318
```

۶ – ۳ – ۸ پر کردن مساحت ها توسط اعداد تصادفی

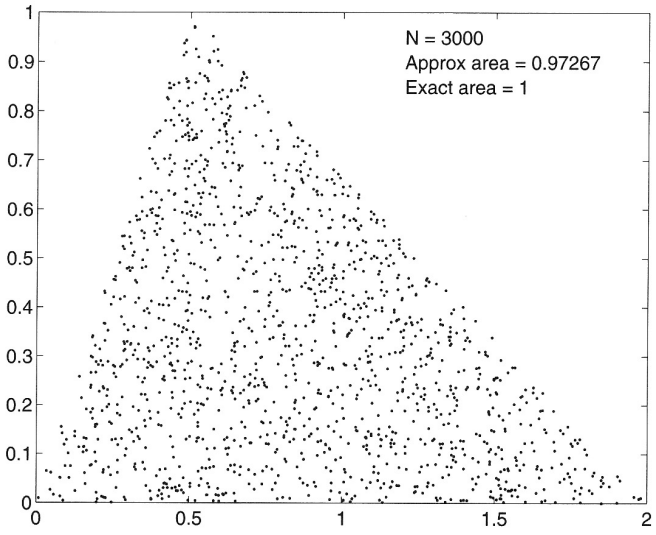
در این قسمت مساحت تقریبی یک مثلث را با استفاده از مختصه‌های تصادفی ایجاد شده و سپس پرکردن مساحت محدود شده توسط مثلث، و نمایش نقاطی که داخل محدودهٔ مثلث قرار می‌گیرند،تقریب خواهیم زد. اعداد تصادفی توسط دستور زیر ساخته می‌شوند:

```
r=unifrnd(al,a2,m,n)
```

که ماتریسی $(n\times m)$ از اعداد دارای توزیع احتمال یکنواخت (مساوی) را در بازهٔ مشخص شدهٔ (a_1,a_2)، ایجاد خواهد کرد.

مثلث نشان داده شده در شکل ٦ -٢١ را در نظر بگیرید. سطح این مثلث بصورت $A_E = D.Sbc$ خواهد بود. محاسبهٔ مساحت تقریبی این سطح توسط محاسبهٔ نسبت نقاط تصادفی ایجاد شده که در داخل مرزها قرار می‌گیرند، ضربدر مساحت ممکن کل (فضای نمونه) یعنی bc، میسر خواهد بود. اگر فرض کنیم b=1، c=2 و a=c/4 باشد و 3000 جفت نقطه را مورد آزمون قرار دهیم، در اینصورت برنامهٔ مورد نیاز جهت محاسبهٔ این عملیات بصورت زیر خواهد بود:

```
b=1;c=2;a=0.25*c;
exactarea=0.5*b*c;
N=3000;
x=unifrnd (0,c,1,N);
y=unifrnd (0,b,1,N);
indL=find (x<=a);
```



شکل ٦ – ٢٢) مثلثی که سطحش توسط نقاط ایجاد شدهٔ تصادفی پر شده است.

```
nxL=x(indL);
nyL=y(indL);
indyL=find(nyL<=b*nxL/a);
indR=find(x>a&x<=c);
```

```
nxR=x(indR);
nyR=y(indR);
indyR=find(nyR<=b/(a-c)*(nxR-c));
approxA=b*c*(length(indyL)+length(indyR))/N;
plot(nxL(indyL),nyL(indyL),'k',nxR(indyR),nyR(indyR),'k.')
text(0.6*c,0.95,['N='num2str(N)])
text(0.6*c,0.90,['Approx area='num2str(approxA)])
text(0.6*c,0.85,['Exact area='num2str(exactarea)])
```

نتیجهٔ اجرای این برنامه در شکل ٦-۲۲ نشان داده شده است.

تمرین ها

توجه: ترسیمات در همهٔ تمرینها را با استفاده از بردارها و عملیات نقطه‌ای و تابع *meshgrid* می‌توان انجام داد. از ساختار *for* و تنها جهت افزایش پارامترها به شیوهٔ مناسب استفاده کنید.

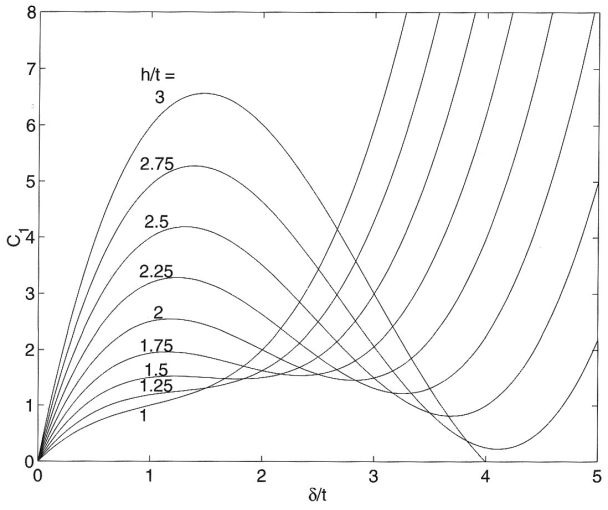
۶-۱ نیرو در یک فنر بشقابی (به شکل ٥-۳۳ توجه شود) متناسب با ضریب C_1 است که به صورت:

$$C_1 = 0.5d_t^{\,3} - 1.5h_t d_t^{\,2} + \left(1 + h_t^{\,2}\right)d_t$$

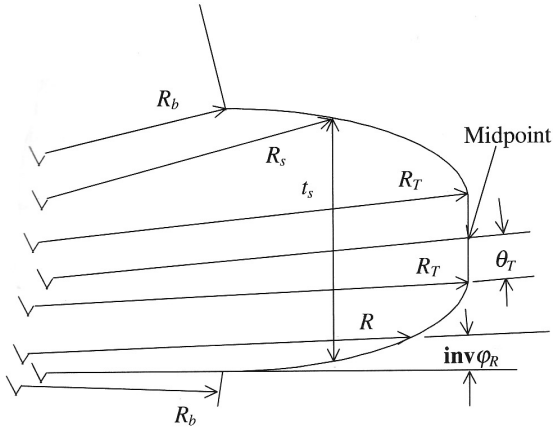
می‌باشد و $h_t = h/t$ ،$d_t = \delta/t$ و δ تغییر شکل فنری می‌باشد. C_1 را به صورت تابعی از d_t در حالی که h_t از 1 تا 3 با نسبت افزایش 0.25 و d_t از 0 تا 5 تغییر می‌کند را رسم کنید. منحنی ها را نشانه گذاری کنید و محور y را تا 8 محدود کنید. نتیجه باید به صورت شکل ٦-۲۳ باشد.

٦-۲ دندانهٔ دندهٔ نشان داده شده درشکل٦-۲٤ را در نظر بگیرید. اگر دنده شامل n دندانه باشد در این صورت در هر $2\pi/n$ رادیان یک دندانه قرار خواهد داشت فرض کنید R_b شعاع دایرهٔ مبنا و R_t شعاع سر دنده و $R(R_b \le R \le R_T)$ شعاع یک نقطه از پروفیل دندانه باشد مختصه‌های قطبی پروفیل دندانه (R,ψ) که شامل فاصلهٔ خالی میان دندانه های متوالی نیز می شود در جدول ٦-۲ داده شده است. در جدول ٦-۲ φ_s زاویهٔ فشار دنده، می‌باشد.

جدول ٦- ٢ (تعاریف بخشهای مختلف دندانهٔ نشان داده شده در شکل ٦ - ٢٤)		
تعاریف	**Ψ**	**R**
$\varphi(R)=\cos^{-1}(R_b/R)$	$inv(\varphi(R))$	$R_b \le R \le R_T$
$inv(x)=tan(x)-x$		
$\theta_T = 0.5t_s/R_s + inv(\varphi_s) - inv(\varphi(R_T))$	$inv(\varphi(R_T)) \le \psi \le inv(\varphi(R_T)) + 2\theta_T$	R_T
	$2[\theta_T + inv(\varphi(R_T))] - inv(\varphi(R))$	$R_b \le R \le R_T$
	$2[\theta_T + inv(\varphi(R_T))] \le \psi \le 2\pi/n$	R_b



شکل ٦ – ٢٣) ضریب C_1 مربوط به یک فنر بشقابی



شکل ٦ – ٢٤) ابعاد نشان داده شده مربوط به یک دندانهٔ چرخ دنده

زاویه فشار دنده می تواند 14.5°، 20° و یا 25° باشد. $R_s = nm/2$ شعاع گام استاندارد، m مدول دندهٔ و t_s ضخامت دندانه در R_s می باشد. اگر دنده‌ای شامل 24 دندانه با زاویهٔ فشار 20°، مدول $10mm$، ضخامت دندانهٔ $14.022mm$، شعاع مبنای $90.21mm$ وشعاع سر دنده $106mm$ باشد در اینصورت دنده را با استفاده از دو تابع $polar$ و $plot$ ترسیم نمایید.

۳-۶ بازده یک پیچ انتقال قدرت بر حسب درصد هنگامیکه از اصطکاک لغزنده صرفنظر شود به صورت زیر خواهد بود ؛

$$e = 100\frac{cos(\alpha) - \mu tan(\lambda)}{cos(\alpha) + \mu cot(\lambda)}\%$$

که در آن μ ضریب اصطکاک، λ زاویهٔ پیشروی پیچ و α زاویهٔ دندانه می‌باشد بازده را بصورت تابعی از λ در محدودهٔ $0 < \lambda < 90^\circ$ و $\mu = 0.02,0.05,0.10,0.15,0.20,0.25$ برای دو زاویهٔ دندانهٔ $\alpha = 7^\circ$ و $\alpha = 14.5^\circ$ ترسیم کنید. شکل و منحنی های منفرد را نشانه گذاری کنید و از تابع $axis$ جهت محدود کردن بازده از 0 تا 100% استفاده کنید. نتایج باید مشابه نتایج نشان داده شده در شکل ٦-٢٥ باشد.

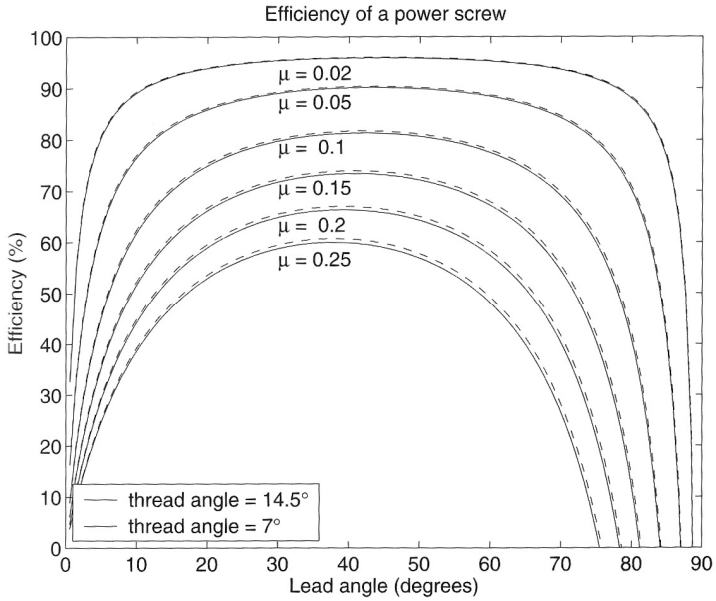
٦-٤ با استفاده از نتایج تمرین ١-١ پارامترهای σ_x/p_{max}، σ_y/p_{max} و σ_z/p_{max} $\tau_{xz}/p_{max} = \tau_{yz}/p_{max} = 0.5(\sigma_x/p_{max} - \sigma_z/p_{max})$ را به صورت تابعی از z/a برای $\upsilon_1 = 0.3$ ترسیم کنید. نمودار را نشانه گذاری کرده و منحنی را مشخص کنید.

۶-۵ با استفاده از نتایج تمرین ۱-۲ پارامترهای σ_x/p_{max}، σ_y/p_{max}، σ_z/p_{max} و τ_{yz}/p_{max} را بصورت تابعی از z/b برای $\upsilon = 0.3$ ترسیم کنید. نمودار را نشانه گذاری کرده و منحنی را مشخص کنید.

۶-۶ منحنی‌های زیر را ترسیم کنید[1]. در ترسیم تمامی این نمودارها باید از تابع *axis equal* استفاده شود.

سیکلوییید $(-\pi \le \varphi \le 3\pi \ ; \ r_a = 0.5, 1, 1.5)$

$$x = r_a\varphi - \sin\varphi$$



شکل ۶ – ۲۵) بازده یک پیچ انتقال قدرت

$$y = r_a - \cos\varphi$$

لیمسنکات (پروانه ای) $(-\pi/4 \le \varphi \le \pi/4)$

[1] D. von Seggern, *CRC Standard Curves and Surfaces*, CRC Press, Inc., Boca Raton, FL, 1993.

$$x = \cos\varphi\sqrt{2\cos(2\varphi)}$$
$$y = \sin\varphi\sqrt{2\cos(2\varphi)}$$

مارپیچ $(0 \le \varphi \le 6\pi)$

ارشمیدس

$$x = \varphi\cos\varphi$$
$$y = \varphi\sin\varphi$$

لگاریتمی $(k = 0.1)$

$$x = e^{k\varphi}\cos\varphi$$
$$y = e^{k\varphi}\sin\varphi$$

کاردیوئید $(0 \le \varphi \le 2\pi)$

$$x = 2\cos\varphi - \cos 2\varphi$$
$$y = 2\sin\varphi - \sin 2\varphi$$

ستاره $(0 \le \varphi \le 2\pi)$

$$x = 4\cos^3\varphi$$
$$y = 4\sin^3\varphi$$

اپی سیکلوئید $(R_r = 3, a_r = 0.5, 1 \text{ or } 2, \text{ and } 0 \le \varphi \le 2\pi;\ R_r = 2.5, a_r = 2, \text{ and } 0 \le \varphi \le 6\pi)$

$$x = (R_r + 1)\cos\varphi - a_r\cos(\varphi(R_r + 1))$$
$$y = (R_r + 1)\sin\varphi - a_r\sin(\varphi(R_r + 1))$$

هیپوسیکلوئید $(R_r = 3, a_r = 0.5, 1 \text{ or } 2, \text{ and } 0 \le \varphi \le 2\pi)$

$$x = (R_r - 1)\cos\varphi + a_r\cos(\varphi(R_r - 1))$$
$$y = (R_r - 1)\sin\varphi - a_r\sin(\varphi(R_r - 1))$$

جدول ۶ - ۳ (ثوابت مورد نیاز جهت تعیین μ)			
عدد SAE	j	A_j	B_i
10	1	9.1209	3.5605
20	2	9.1067	3.5385
30	3	8.9939	3.4777
40	4	8.9133	3.4292
50	5	8.5194	3.2621
60	6	8.3666	3.1884

۶-۷ ویسکوزیتهٔ مطلق روغن در واحد $(lb-s/in^2)$ $\mu reyn$ را می توان با خطای $\pm 10\%$ از رابطهٔ ؛

$$\mu = 10^{C-1}$$

بدست آورد. که در آن :

$$C = 10^{A_j - B_j \log_{10} T_0}$$

$T_0 = 255.2 + 5/9T\ ^\circ K$ ، T دمای روغن برحسب $^\circ F$ و A_j و B_j در جدول ۶-۲ بصورت تابعی از عدد SAE روغن داده شده است. منحنی $\log_{10}\log_{10}(10\mu)$ را بصورت تابعی از $\log_{10}(T_0)$ و μ را بصورت تابعی از T_0 برای شش روغن داده شده در جدول ۶-۳ ترسیم کنید. (بخش ۶-۸ را نیز در این باره مشاهده کنید.)

۶-۸ رابطهٔ میان λ ، زاویهٔ پیشروی یک چرخدندهٔ حلزونی، نسبت $\beta = N_1/N_2$ که N_1 تعداد دندانه های چرخ حلزونی و N_2 تعداد دندانه‌های حلزون متحرک می‌باشد، C ، فاصلهٔ مراکز میان شافت ها و P_{dn} ، گام قطری عمودی به صورت ؛

$$K = \frac{2p_{dn}C}{N_2} = \frac{\beta}{\sin\lambda} + \frac{1}{\cos\lambda}$$

می‌باشد.نمودار K برحسب λ رابرای $1^\circ \le \lambda \le 40^\circ$ $\beta = 0.02, 0.05, 0.08, 0.11, 0.15, 0.18, 0.23, 0.30$ ترسیم کرده و شکل منحنی ها را نشانه گذاری کنید. بازهٔ تغییرات محور y ها را از 1 تا 2 قرار دهید. در همان شکل نتایج تمرین ۶-۹-۵ الف را توسط رسم کردن خطی که مقادیر مینیمم منحنی‌ها را به یکدیگر وصل می‌کند، نشان

دهید. اینکار را توسط الحاق کردن توابع مناسب و به کار بردن قسمتهایی از برنامهٔ تمرین ۵-۹-الف، در برنامهٔ نوشته شده برای این تمرین انجام دهید. نتایج باید شبیه شکل ٦-٢٦ باشد.

٦-٩ برنامه ای بنویسید که سه و یا تعدادی بیشتری دایره را حول یک دایرهٔ مرکزی به شعاع $r_b = 1.5$ ، همانگونه که در شکل ٦-٢٧ برای n=5 دایره نشان داده شده است، ترسیم کند. شعاع r_s دایره های بیرونی توسط رابطهٔ ؛

$$r_s = \frac{r_b \sin(\pi/n)}{1-\sin(\pi/n)}$$

قابل محاسبه می‌باشد. این برنامه باید تعداد دایره‌ها را از کاربر بپرسد. این برنامه را می‌توان بدون استفاده از حلقهٔ *for* تشکیل داد.

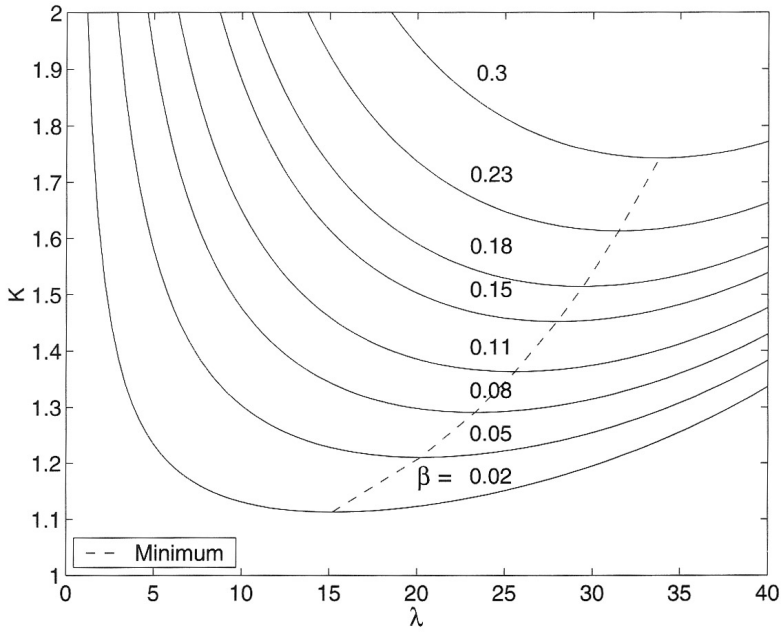
۶-۱۰ اغلب در فرآیند بهینه سازی رسم تابعی (objective function) که قرار است در محدودهٔ مشخصی (قیود) بهینه شود، بسیار سودمند خواهد بود. فرض کنید می‌خواهیم تابع

$$f(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 1)^2$$

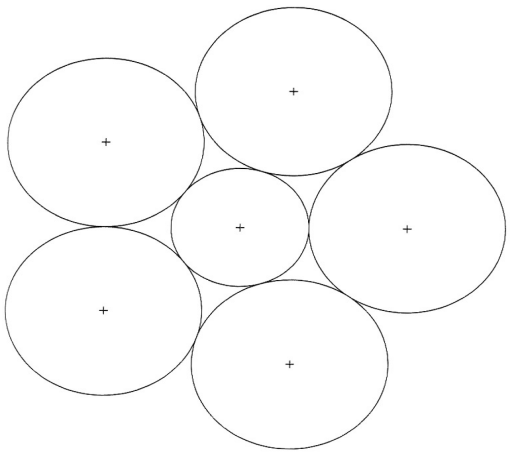
را در مواجهه با قیود :

$$g_1 = (x_1 - 3)^2 + (x_2 - 1)^2 - 1 \le 0$$
$$g_2 = 2x_1 - x_2 - 5 \le 0$$

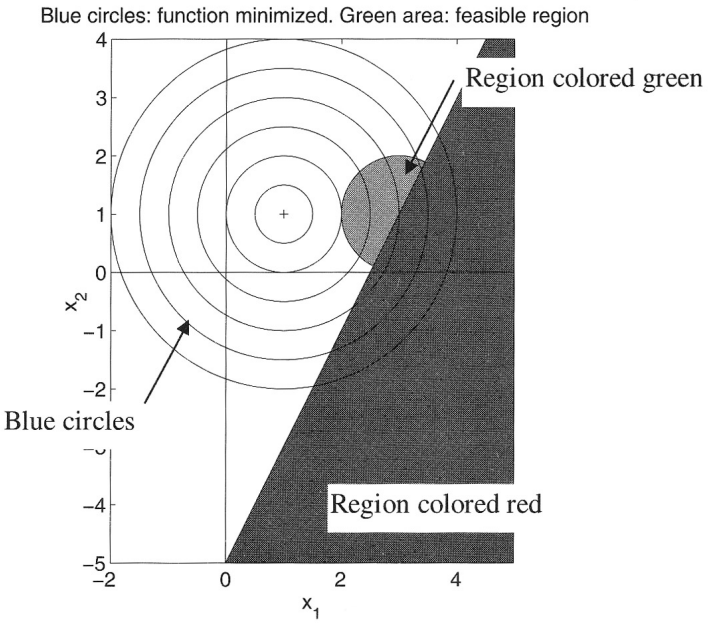
مینیمم کنیم. بنابراین پاسخ x_{m1} و x_{m2} باید روی دایرهٔ $f(x_1, x_2)$ و در محدودهٔ مشخص شده توسط g_1 و g_2 قرار داشته باشد.



شکل ٦ – ۲٦) زاویهٔ پیشروی یک دندهٔ حلزونی



شکل ۶ – ۲۷) پنج دایرهٔ محیط شده روی یک دایره



شکل ٦ – ۲۸) حلی برای تمرین ٦ – ۱۰

تابع هدف فوق (دوایر) را به همراه محدوده‌هایی که جواب باید در آنها قرار داشته باشد، ترسیم کنید. نتیجه باید مشابه شکل ٦–۲۸ باشد. جهت بدست آوردن این نتیجه تابع fill باید به ترتیب مناسب استفاده و اعمال شود.

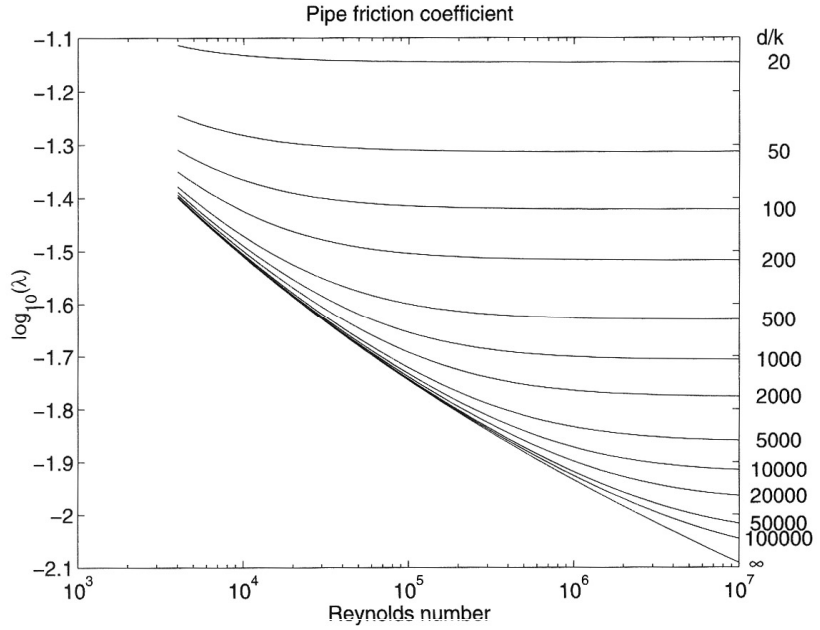
٦–۱۱ در تمرین ٥–٥ فرمول کلبروک را که در آن λ ، ضریب اصطکاک لوله، را تخمین می‌زند، به شکل زیر؛

$$\lambda = \left[-2log_{10}\left(\frac{2.51}{R_e\sqrt{\lambda}} + \frac{0.27}{d/k} \right) \right]^{-2} \qquad R_e \geq 4000$$

نمایش دادیم. که در آن R_e عدد رینولدز، d قطر لوله، k زبری سطح می باشد. برای لوله‌های صاف λ $(k \cong 0; d/k > 100{,}000)$ به صورت زیر ؛

$$\lambda = \left[-2log_{10}\left(\frac{R_e\sqrt{\lambda}}{2.51} \right) \right]^{-2} \qquad R_e \geq 4000$$

محاسبه می‌شود. نمودار $log_{10}(\lambda)$ را به صورت تابعی از $log_{10}(R_e)$، در بازهٔ $4\times10^3 \le R_e \le 10^7$ برای $d/k = 20, 50, 100, 200, 500, 1000, 2000$، $d/k = 5000, 10000, 20000, 50000, 100000$ و $d/k = \infty (k = 0)$ رسم کنید. از تابع semilogx بجای استفاده از دستور plot بهره ببرید. شکل و منحنی ها را نشانه گذاری کنید. نشانه‌های منحنی ها را در طرف راست $R_e = 10^7$ یعنی خارج از شکل، سمت راست محور عمودی قرار دهید. شکل حاصل دیاگرام بدست آمده باید شبیه شکل ٦-۲۹ باشد.



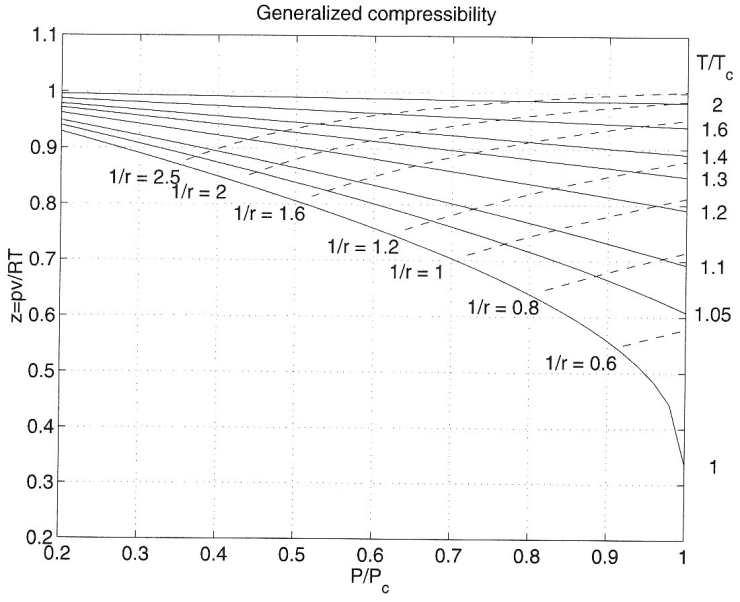
شکل ٦ – ۲۹) دیاگرام مودی

۶-۱۲ تمرین ٥-٤ که در آن معادلهٔ تراکم پذیری زیر برای گازها معرفی شد را به یاد بیاورید:

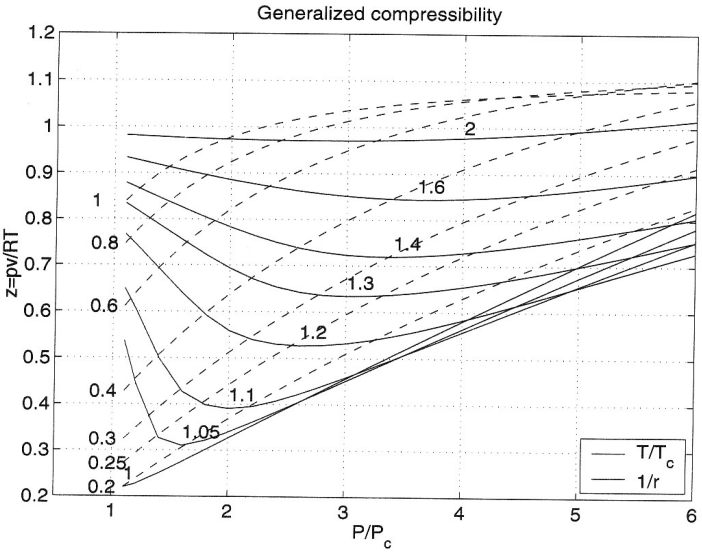
$$Z(r,\tau) = \frac{p\tau}{r} = \frac{p\upsilon}{RT}$$

که در آن $Z(r,\tau)$ در تمرین ٥-٤ تعریف گردید. از روشهای استفاده شده در تمرین های ٥-٤- ب و ٥-٤- پ جهت ایجاد اشکال ٦-۳۰- الف و ٦-۳۰- ب استفاده کنید. در شکل ٦-۳۰- الف، هنگامیکه τ ثابت است در تابع fzero، محدودهٔ r را بصورت $0.05 \le r \le 3.4$ و هنگامیکه r ثابت است، محدودهٔ τ بصورت $0.4 \le \tau \le 2.5$ ، قرار دهید. جهت ایجاد منحنی هایی با خطوط مقطع، از تابع find جهت ساده تر شدن برنامه استفاده کنید. در شکل ٦-۳۰- ب هنگامیکه τ ثابت و کوچکتر

از $1/1.3$ است از حدس اولیه r=4 در تابع fzero استفاده کنید. هنگامیکه $\tau \geq 1/1.3$ و $p < 3$ می‌باشد، از حدس اولیه r=2 و در غیر اینصورت از r=3.3 استفاده کنید. هنگامیکه r ثابت است از حدس اولیهٔ $\tau = 0.6$ استفاده نمایید.

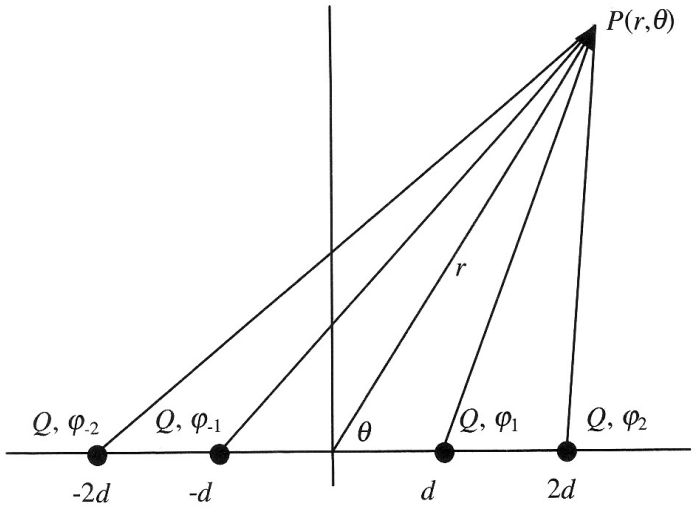


(الف)



(b)

شکل ٦-۳۰) ضریب تراکم پذیری تعمیم یافته؛ (الف) $0 \le p/p_c \le 1$ (ب) $0 \le p/p_c \le 6$



شکل ٦ – ۳۱) آرایش خطی منابع آکوستیک

۱۳-٦ چند جمله ای زیر را در بازهٔ $-12 \le x \le 7$ در نظر بگیرید ؛

$$y = 0.001x^5 + 0.01x^4 + 0.2x^3 + x^2 + 4x - 5$$

تنها مقادیر مثبت آنرا رسم کنید. مقدار y را در نقاط ابتدایی و انتهایی هر قسمت تقریباً برابر با صفر قرار دهید. {راهنمایی: از تابع *find* استفاده کنید.}

۱٤-٦ برنامه ای بنویسید که شکل ٥-۱٤ مربوط به تمرین ٥-٦ را ایجاد نماید.

۱٥-٦ردیفی خطی مشکل از N جفت منبع آکوستیک که در شکل ٦-۱۳ رسم شده است و با فرکانس ω و دامنهٔ Q در حال ارتعاش هستند را در نظر بگیرید.

فشار کلی صوت در فاصلهٔ r از ردیف منبع ها به صورت ؛

$$p(r,\theta) = Z_0\left[\sum_{m=1}^{N}\left(1+\frac{md}{r}\cos\theta\right)exp\left[j\left(-\varphi_m + mdk\cos\theta\right)\right]\right]$$
$$+\sum_{m=1}^{N}\left(1-\frac{md}{r}\cos\theta\right)exp\left[j\left(-\varphi_{-m} - mdk\cos\theta\right)\right]$$

می‌باشد. که در آن ؛

$$Z_0 = \frac{j\rho ckQ}{4\pi r}e^{j(\omega t - kr)}$$

و S چگالی متوسط، C سرعت متوسط موج، $k = 2\pi\omega/c$ شمارهٔ موج، $\varphi_{\pm m}$ زوایای فاز می‌باشد. ضمناً $\varphi_1 = 0$ ، $d/r << 1$ و $dk << 1$ می‌باشد.

برای حالتیکه $N=1$ ، $\varphi_1 = 0$ و $\varphi_{-1} = \pi$ می باشد، معادلهٔ فوق به شکل زیر ؛

$$P(r,\theta) = 2Z_0\left[jdk + \frac{d}{r}\right]\cos\theta$$

ساده می‌شود، که از آن جهت محاسبهٔ فشار یک منبع صوتی دی پل در نقاط مختلف میدان استفاده می‌شود.

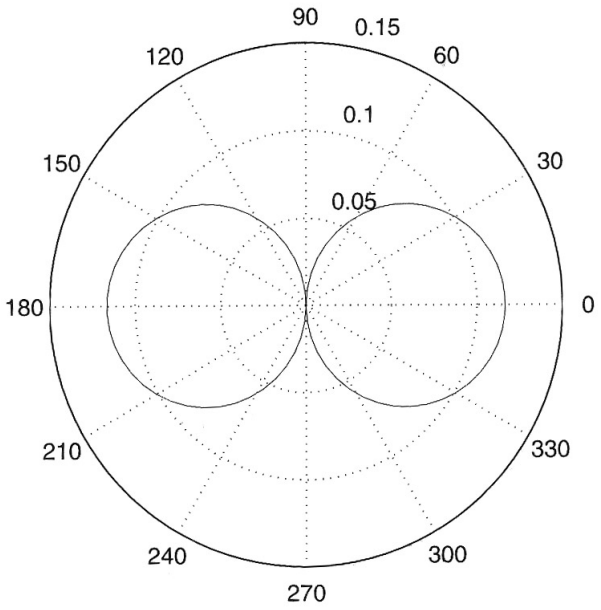
مقدار $|P(r,\theta)/Z_0|$ را بصورت تابعی از θ محاسبه کرده و نتایج را با استفاده از تابع polar، در بازهٔ $0 \le \theta \le 2\pi$ ، برای $d/r = 0.05$ ، $dk = 0.03$ ، $\varphi_{-1} = \pi$ و $\varphi_1 = 0$ رسم نمایید. نتیجه باید مشابه شکل ٦-۳۲ باشد. از فرمول حالت کلی برای ردیف خطی، استفاده کنید. همچنین درصد اختلاف میان حل دقیق و حل عددی کاهیده شده را برای دی پل رسم نمایید. نقطهٔ اول یعنی $\theta = 0$ را از این ترسیم حذف کنید. تابعی جهت محاسبهٔ اندازهٔ فشار به صورت تابعی از اعداد جفت شدهٔ منبع‌ها، روابط مربوط به فاز و فواصلشان و مکان میدان دور در فضا بنویسید.

٦-۱٦ تابع چگالی احتمال یک سیگنال متغیر با زمان جهت محاسبهٔ احتمال اینکه دامنهٔ سیگنال در پریود زمانیT دارای مقداری بین x و x+dx باشد، استفاده می‌شود. به بیان دیگر، از این تابع جهت تعیین درصدی از زمان که سیگنال در بازهٔ دامنه مطلوب قرار دارد، بهره‌برداری می‌شود. تابع چگالی احتمال را می‌توان توسط رابطهٔ زیر ؛

$$P(x) = \lim_{\substack{\Delta x \to 0 \\ T \to \infty}} \left[\sum_i \Delta t_i \Big/ T\Delta x \right]$$

تخمین زد. که پارامترهای آن برای یک پریود موج سینوسی در شکل ٦-۳۳ نشان داده شده است. تابع چگالی احتمال p(x) مربوط به یک موج سینوسی با دامنهٔ A_0 توسط رابطهٔ ؛

$$P(x) = \frac{1}{\pi\sqrt{A_0{}^2 - x^2}} \qquad |x| \le A_0$$
$$= 0 \qquad |x| > A_0$$



شکل ٦ – ۳۲) الگوی تشعشعی یک دو قطبی آکوستیک

داده می شود. تابع چگالی احتمال را برای تابع ؛

$$y = A_0 \sin t$$

در بازهٔ $-\pi \le t \le \pi$ تخمین بزنید و نتایج را با مقادیر دقیقشان مقایسه کنید. فرض کنید $A_0 = 2$ و تعداد داده‌ها در این بازهٔ زمانی، 400 باشد. مقادیر تقریبی $p(x)$ و مقادیر دقیق آنرا ترسیم کنید. نتایج باید شبیه نتایج نشان داده شده در شکل ٦-۳٤ باشد، که در آن اختصارات به صورت دستی جابجا شده است.

۱۷-۶ مستطیل نشان داده شده در شکل ٦-۳۵ را در نظر بگیرید. روابط ؛

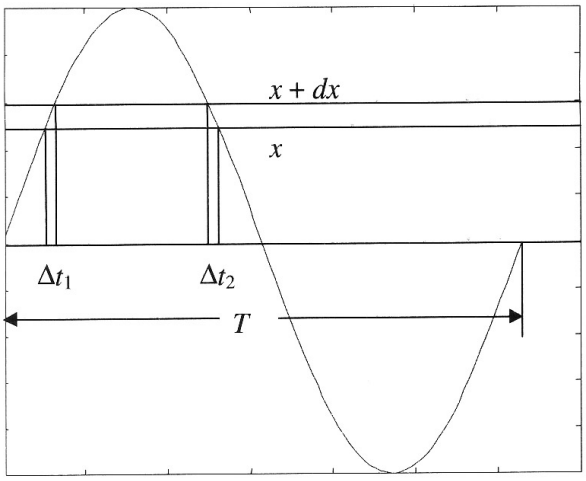
$$r_1 = \sqrt{d^2 + (W/2)^2} \qquad \alpha = tan^{-1}(W/2d)$$

$$r_2 = \sqrt{(d+L)^2 + (W/2)^2} \qquad \beta = tan^{-1}(W/2(d+L))$$

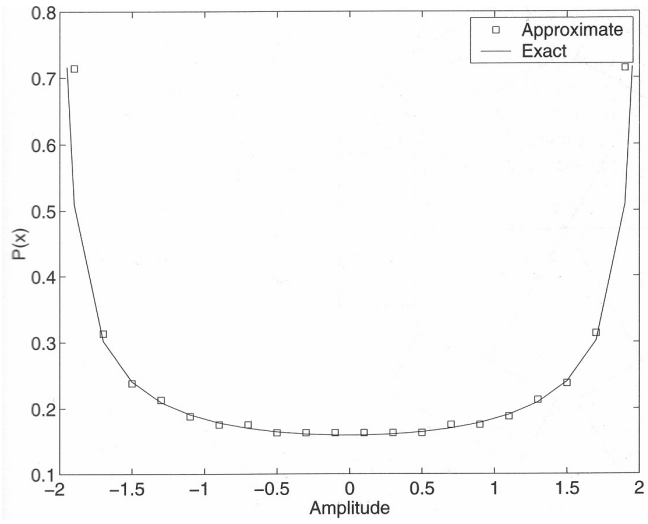
به وضوح مشخصند. اگر مقادیر L، W و d داده شوند، برنامه‌ای بنویسید که ماکزیمم تعداد مستطیلهای غیر متقاطع را همانطور که در شکل ٦-۳٦ نشان داده شده است، رسم نماید.

مقادیر $L = 1$، $W = 2$ و $d = 2$ جهت ایجاد شکل ٦-٣٦ استفاده شده‌اند. ماکزیمم تعداد مستطیلها را می‌توان توسط تابع $floor(\pi/\alpha)$ تعیین نمود.

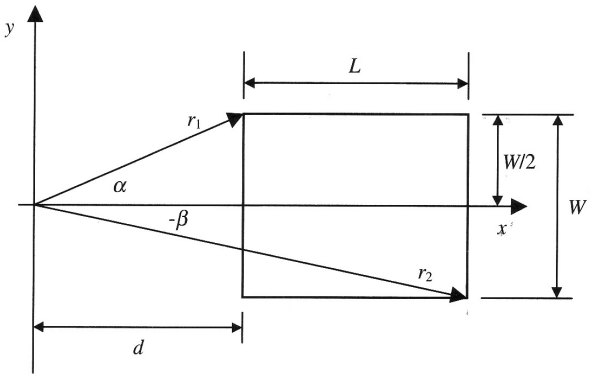
این برنامه را می توان بدون استفاده از حلقهٔ for نوشت.



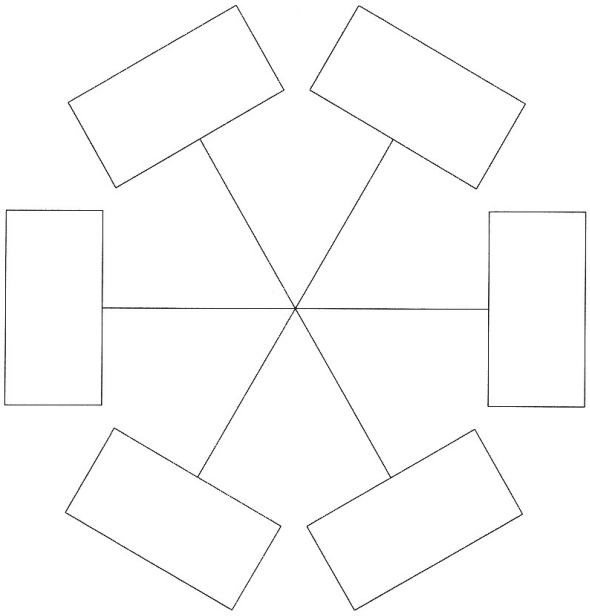
شکل ٦-٣٣) تعیین مدت زمانیکه یک موج سینوسی فاصله بین $x + dx$, x را می‌پیماید



شکل ٦- ٣٤) تابع چگالی احتمال یک موج سینوسی



شکل ٦ – ٣٥) توصیف یک مستطیل برای تمرین ٦– ١٧



شکل ٦ – ٣٦) تکرار مستطیلهای غیر متقاطع

فصل هفتم

گرافیک سه بعدی

۷-۱ خطوط در فضای سه بعدی

۷-۲ سطوح

تمرین‌ها

در این فصل چگونگی استفاده از توابع گرافیک سه بعدی به همراه شیوهٔ استفاده از آنها بیان خواهد شد.

۷ - ۱ خطوط در فضای سه بعدی

شکل سه بعدی دستور *plot* به صورت زیر می‌باشد:

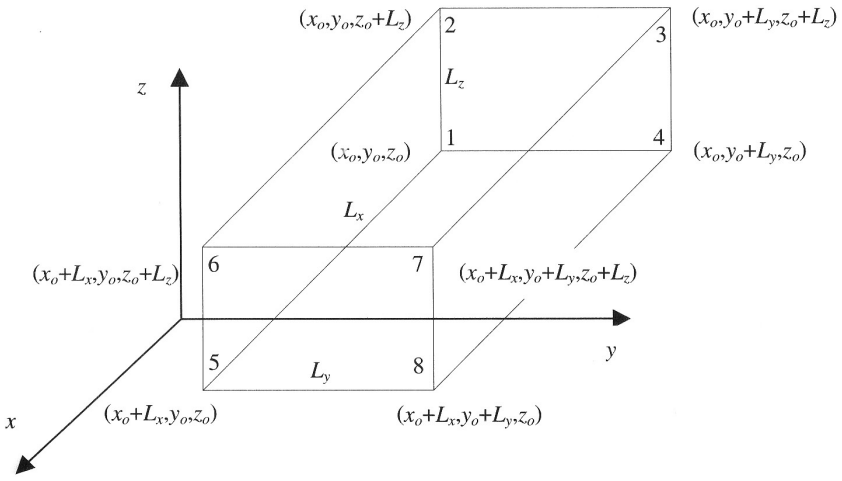
plot3(u1,v1,w1,c1,u2,v2,w2,c2,...)

که u_i و v_i و w_i به ترتیب مختصه های *(x,y,z)* یک نقطه و یا مجموعه‌ای از نقاط می‌باشند. این مختصه‌ها می‌توانند مجموعه‌های مرتب سه تایی از اعداد، بردارهایی با طول یکسان، ماتریسهایی با مرتبهٔ یکسان و یا عبارتهای محاسباتی که پس از ارزیابی منجر به یکی از سه شکل ذکر شده می‌شوند، باشند. کمیت c_j رشته‌ای از کاراکترها می‌باشد، که کاراکتر اول نشان دهندهٔ رنگ، کاراکتر دوم مبین ویژگیهای نقطه، و مابقی کاراکترها (حداکثر دو کاراکتر) تعیین کنندهٔ نوع خط خواهند بود. به بحث مقدماتی انجام شده در مورد تابع *plot* در بخش ٦ - ۲ توجه کنید.

فرض کنید می‌خواهیم n خط جداگانه را که نقاط انتهایی آنها (x_{1n}, y_{1n}, z_{1n}) و (x_{2n}, y_{2n}, z_{2n}) می‌باشند، رسم کنیم. برای انجام این کار شش بردار را به شکل زیر تشکیل می‌دهیم:

$$x_j = [\, x_{j1}\ x_{j2}\ \ldots\ x_{jn}\,]$$

$$y_j = [\, y_{j1}\ y_{j2}\ \ldots\ y_{jn}\,] \qquad j = 1,2$$



شکل ۷ – ۱) نمایش مختصه های یک مکعب

$$z_j = [\, z_{j1}\ z_{j2} \ldots z_{jn} \,]$$

در این صورت شکل کلی تابع *plot3* به صورت زیر خواهد بود :

plot3([x1;x2],[y1;y2],[z1;z2])

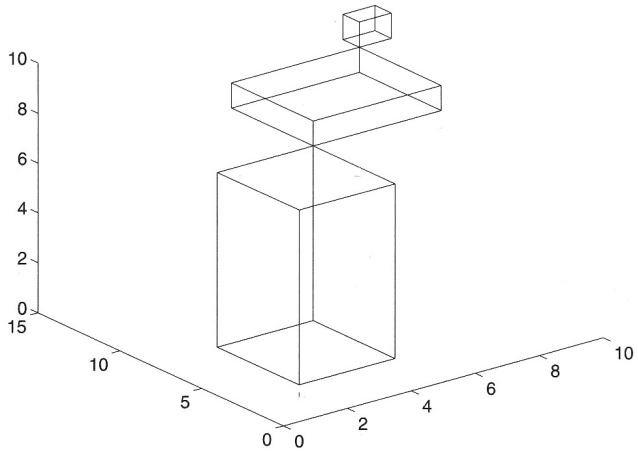
که در آن *[x1;x2]*، *[y1;y2]* و *[z1;z2]* ماتریسهایی از مرتبهٔ *(n×2)* می‌باشند. که تعمیمی از شکل کلی تابع *plot* در فضای سه بعدی می‌باشد.

تمامی شیوه‌های برچسب گذاری که برای ترسیمات دو بعدی در بخش ٦ – ۲ مطرح گردید، در مورد توابع ایجاد سطح و ایجاد منحنی‌های سه بعدی نیز صادق خواهند بود، البته به جز تابع *text3* که بجای تابع *text* استفاده می‌شود و تابع *zlabel* که از آن جهت نشانه گذاری محور *z* استفاده می‌شود.

مثال ۷– ۱ مکعبی در قالب شکل سیمی (متشکل از مجموعه ای خطوط)

مکعبی با ابعاد $L_x \times L_y \times L_z$ را که در شکل ۷–۱ نشان داده شده است، در نظر بگیرید. تابع *Boxplot3* را برای کشیدن چهار یال هر یک از شش وجه مکعب ایجاد می‌کنیم. موقعیت و جهت

مکعب با استفاده از مختصاتهای دو گوشهٔ مقابل $P(x_0,y_0,z_0)$ و $P(x_0\pm L_x,y_0\pm L_y,z_0\pm L_z)$ که روی قطرهای اصلی مکعب واقعند، تعیین می‌شود.



شکل ۷ – ۲) مکعبها در قالب شکل سیمی

```
function BoxPlot3 (x0,y0,z0,Lx,Ly,Lz)
x=[x0       x0       x0       x0       x0+Lx       x0+Lx       x0+Lx
x0+Lx];          y=[y0      y0     y0+Ly   y0+Ly     y0              y0
y0+Ly        y0+Ly];
z=[z0      z0+Lz    z0+Lz     z0         z0          z0+Lz       z0+Lz
z0      ];
index=zeros(6,5);
index=(1,:)=[1 2 3 4 1];
index=(2,:)=[5 6 7 8 5];
index=(3,:)=[1 2 6 5 1];
index=(4,:)=[4 3 7 8 4];
index=(5,:)=[2 6 7 3 2];
index=(6,:)=[15 8 4 1];
for k=1:6
  plot3(x(index(k,:)),y(index(k,:)),z(index(k,:)))
  hold on
end
```

برنامهٔ زیر که سه مکعب با ابعاد داده شده و مکان یکی از گوشه ها ایجاد می کند را در نظر بگیرید:

```
Box#1
        Size: 3×5×7
        Location: (1,1,1)
Box#2
        Size: 4×5×1
        Location: (3,4,5)
Box#3
         Size: 1×1×1
         Location: (4.5,5.5,6)
```

برنامهٔ لازم جهت تشکیل و نمایش این قالب سیمی جعبه ها به صورت زیر می باشد:

```
BoxPlot3(1,1,1,3,5,7)
BoxPlot3(4,6,8,4,5,1)
BoxPlot3(8,11,9,1,1,1)
```

که پس از اجرا منجر به نمایش شکل ۷-۲ خواهد شد.

۷-۲ سطوح

توابع رسم ترسیمات سه بعدی که سطوح، کانتورها، حجم‌ها و ترکیبهایی از این اشکال را به وجود می‌آورند دارای قدرت نسبتاً زیادی می‌باشند. توابع مهم و عمدهٔ ترسیم سطوح، در زیر آورده شده است.

```
surf
```

و

```
mesh
```

تابع *surf* سطحی متشکل از قطعات رنگی را ترسیم می‌کند، در صورتیکه از تابع *mesh*، جهت ترسیم قطعات سفید رنگ که توسط حدودشان تعیین می‌شوند، استفاده می‌شود. در تابع *surf*، رنگ قطعات

توسط مقدار z تعیین می‌شود.در حالیکه در تابع *mesh* ، مقدار z تعیین کنندهٔ رنگ خطوط خواهد بود. یک سطح را می‌توان توسط عبارت *z=f(x,y)* نمایش داد. که در آن x و y ، مختصه‌های دستگاه مختصات در صفحهٔ xy و z ارتفاع حاصله می‌باشد. روند ایجاد سطح با استفاده از دستور ؛

```
surf(x,y,z)
```

کامل خواهد شد. در اینجا از تابع *surf* جهت ایجاد شکل ۲-۳ و از تابع *mesh* جهت ایجاد شکل ۲-٦ استفاده شده است. اکنون نحوهٔ استفاده از این توابع را با رسم سطح ؛

$$z(x_1,x_2) = x_1^4 + 3x_1^2 + x_2^2 - 2x_1 - 2x_2 - 2x_1^2 x_2 + 6$$

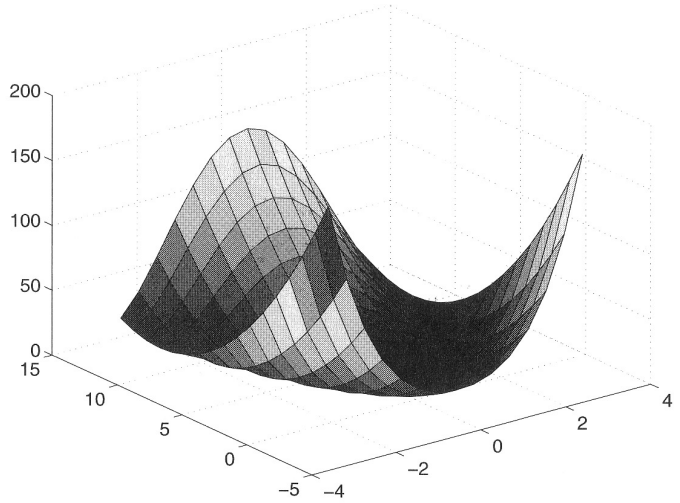
در بازهٔ $-3 < x_1 < 3$ و $-3 < x_2 < 13$ بیشتر توضیح می‌دهیم.

برنامهٔ مورد نیاز جهت رسم این سطح مطابق زیر می باشد:

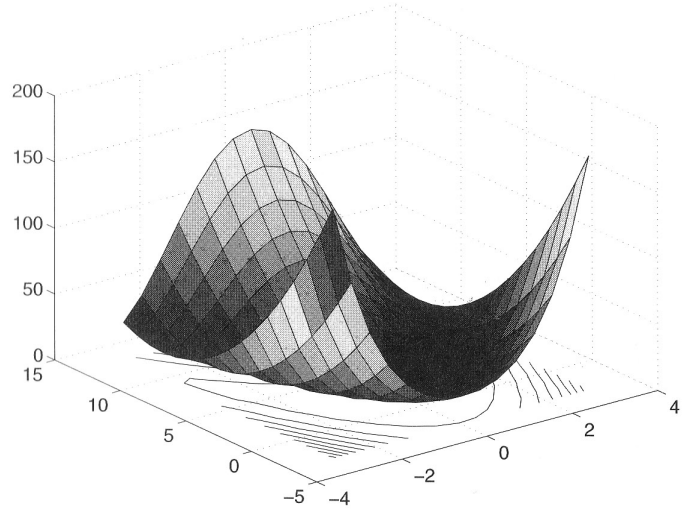
```
xx1=linspace(-3,3,15);                          %(1×15)

xx2=linspace(-3,13,17);                         %(1×17)

[x1,x2]=meshgrid(xx1,xx2);                       %(17×15)

z=x1.^4+3*x1^2-2*x1+6-2*x2.*x1.^2+x2.^2-2*x2;   %(17×15)

surf(x1,x2,z)
```

تابع *meshgrid* دو ماتریس *(17×15)* ایجاد خواهد کرد. بنابراین z می‌تواند در تمامی ترکیب‌های x_1 و x_2 محاسبه شود. این محاسبات با استفاده از تمامی ترکیبهای x_1 و x_2 توسط عملگرهای نقطه‌ای نشان داده شده در عبارت $z = z(x_1,x_2)$ صورت می‌پذیرد.

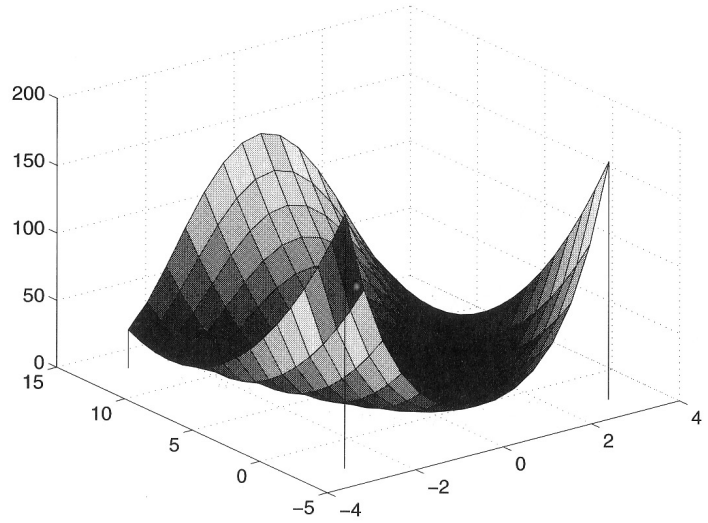
اجرای این برنامه شکل ۷ -۳ را نتیجه خواهد داد. جهت دیدن تصویر چند کانتور از این سطح که روی صفحهٔ *z=0* تصویر شده است، بجای استفاده از تابع *surf*، از تابع *surfc* استفاده خواهیم کرد. نتیجه در شکل ۷ - ٤ نشان داده شده است. آرگومانهای تابع *surfc* مانند آرگومانهای تابع *surf* خواهند بود.



شکل ۷ – ۳) سطح ایجاد شده توسط دستور *surf*



شکل ۷ – ٤) سطح و کانتورهای ایجاد شده توسط دستور *surfc*



شکل ۷ – ۵) سطح با خطوط رسم شده به گوشه هایش

جهت ترسیم خطوطی از چهار گوشهٔ سطح به صفحهٔ $z=0$ ، ابتدا یاد آوری می‌کنیم که مختصات این نقاط به صورت ؛

$[-3,-3,z(-3,-3)]$

$[-3,13,z(-3,13)]$

$[3,13,z(3,13)]$

$[3,-3,z(3,-3)]$

می‌باشد. برنامه به صورت زیر خواهد بود:

```
xx1 = linspace(-3,3,15);
xx2 = linspace(-3,13,17);
[x1,x2] = meshgrid(xx1,xx2);
z = x1.^4 + 3*x1.^2 - 2*x1 + 6 - 2*x2.*x1.^2 + x2.^2 - 2*x2;
surf(x1,x2,z)
```

```
x11=[-3 -3 3 3];
x22=[-3 13 13 -3];
z2=x11.^4+3*x11.^2-2*x11+6-2*x22.*x11.^2+x22.^2-2*x22;
hold on
plot3([x11;x11],[x22;x22],[zeros(1,4);z2],'b')
```

نتیجهٔ اجرای این برنامه در شکل ۷ – ۵ نمایش داده شده است. خط چهارم در شکل دیده نمی‌شود.

همانگونه که قبلاً گفته شد، رنگ قطعات مختلف ایجاد شده، توسط دستور *surf* به صورت خودکار با توجه به مقادیر z آنها ایجاد خواهد شد. به طور مشابه، رنگ خطوط ایجاد شده توسط دستور *mesh* نیز وابسته به مقادیر z خواهد بود. رنگ قطعات و یا خطوط را می‌توان توسط استفاده از تابع ؛

```
colormap(c)
```

به یک رنگ مشخص تغییر داد. که در آن c، برداری سه عنصره که مقادیرش بین 0 و 1 تغییر می‌کند، می‌باشد. عنصر اول متناظر با شدت رنگ قرمز، عنصر دوم متناظر با شدت رنگ سبز و عنصر سوم متناظر با شدت رنگ آبی خواهد بود. به فایل کمکی تابع *colormap* مراجعه کنید. برخی از اشکال متداول ترکیب این رنگها مطابق ؛

$c=[0\ 0\ 0] \rightarrow black$

$c=[1\ 1\ 1] \rightarrow white$

$c=[1\ 0\ 0] \rightarrow red$

$c=[0\ 1\ 0] \rightarrow green$

$c=[0\ 0\ 1] \rightarrow blue$

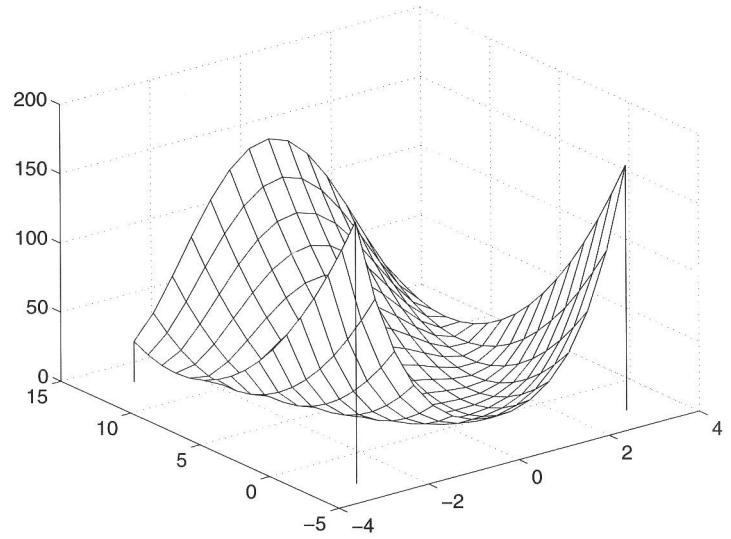
$c=[1\ 1\ 0] \rightarrow yellow$

$c=[1\ 0\ 1] \rightarrow magenta$

$c=[0\ 1\ 1] \rightarrow cyan$

$c=[0.5\ 0.5\ 0.5] \rightarrow gray$

می‌باشند. جهت نشان دادن تمایز بین توابع *surf* و *mesh* و روشن کردن چگونگی استفاده از تابع *colormap*، بجای استفاده از تابع *surf* در برنامهٔ قبل از دستورات ؛



شکل ۷ – ٦) سطح ایجاد شده توسط دستور *mesh* که رنگهایش

توسط دستور *colormap* تغییر می کند.

```
mesh(x1,x2,z)
colormap([0 0 1])
```

استفاده خواهیم کرد. در اینصورت شکل ۷ – ٦ را خواهیم داشت که در آن تمامی خطوط و سطوح به رنگ آبی می باشند.

تابع ؛

```
grid off
```

خطوط شبکه را در این شکلها حذف خواهد کرد. جهت نمایش مجدد این خطوط می‌توان از تابع ؛

```
grid on
```

استفاده نمود. به علاوه می توانیم با استفاده از تابع ؛

```
box on
```

یک مستطیل دور شکل ایجاد کنیم. بنابراین شکل ۷ -٦ را توسط حذف خطوط شبکه و قراردادن مستطیلی دور آن، اصلاح خواهیم کرد. برنامه مطابق زیر خواهد بود :

```
xx1 = linspace(-3,3,15);
xx2 = linspace(-3,13,17);
[x1,x2] = meshgrid(xx1,xx2);
z = x1.^4 + 3*x1.^2 - 2*x1 + 6 - 2*x2.*x1.^2 + x2.^2 - 2*x2;
mesh(x1,x2,z)
colormap([0 0 1])
x11 = [-3 -3 3 3];
x22 = [-3 13 13 -3];
z2 = x11.^4 + x11.^2 - 2*x11 + 6 - 2*x22.*x11.^2 + x22.^2 - 2*x22;
hold on
plot3([x11;x11],[x22;x22],[zeros(1,4);z2],'b')
grid off
box on
```

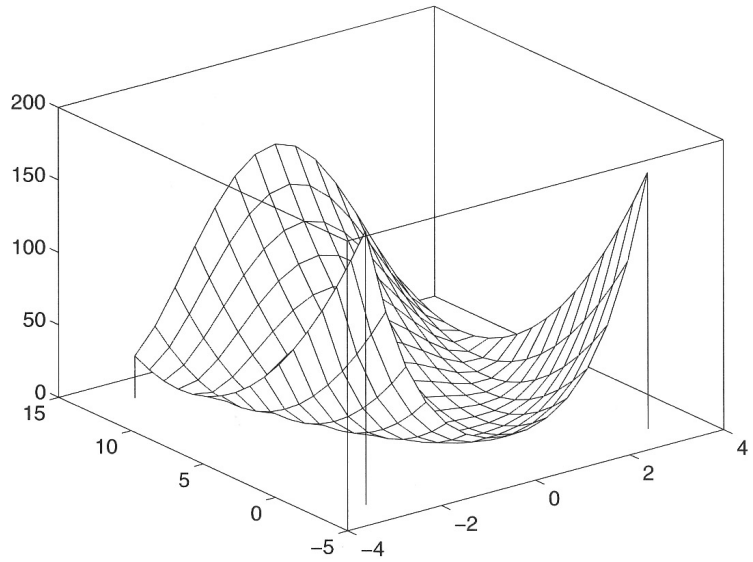
اجرای این برنامه شکل ۷ -۷ را نتیجه می دهد. از طرف دیگر اگر خطوط شبکه را با استفاده از تابع *grid off* و مستطیل دور شکل و محورها را به ترتیب توسط دستورات ؛

```
box off
```

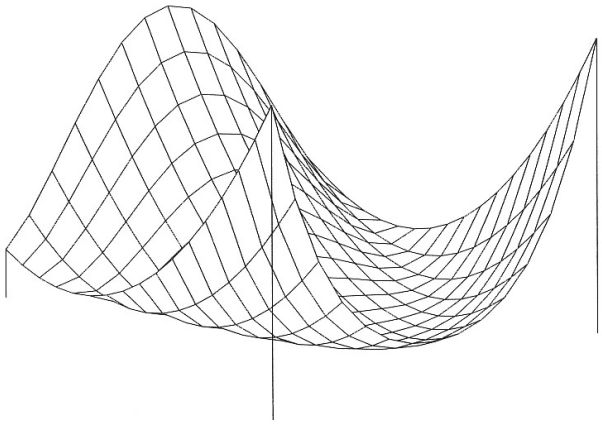
و

```
axis off
```

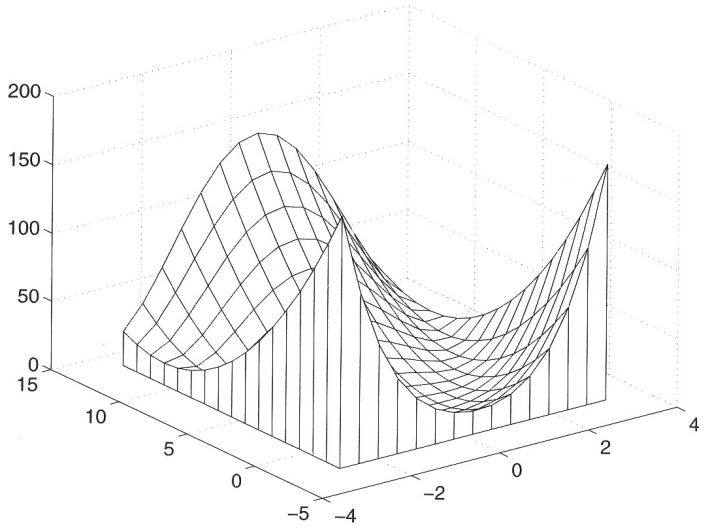
حذف کنیم، شکل ۷ -۸ را خواهیم داشت.



شکل ۷ – ۷) شکل ۷ – ٦ به همراه دستورات *grid off* و *box on*



شکل ۷ – ۸) شکل ۷ – ٦ به همراه دستورات *grid off* ، *box off* و *axis off*



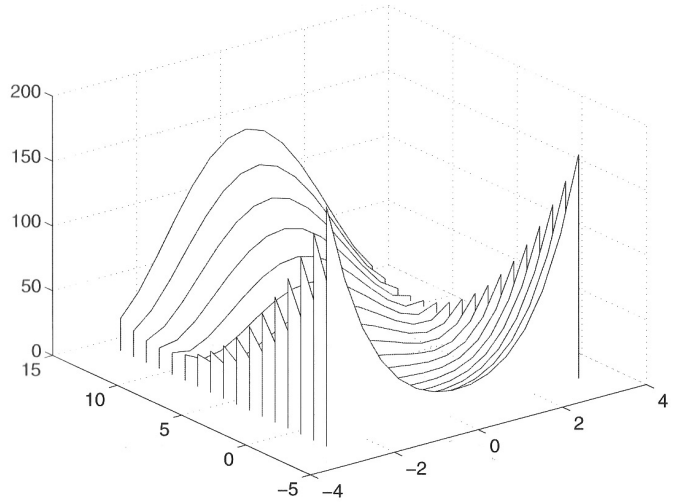
شکل ۷ – ۹) سطح ایجاد شده توسط دستور *meshz*

روشهای متعدد دیگری جهت تغییر شیوه‌های نمایش سطوح ظاهر شده در اشکال ۷-۳ تا ۷-۸ وجود دارد. این شیوه‌ها به طور مختصر در اشکال ۷-۹ تا ۷-۱۳ نشان داده شده است. شکل ۷-۹ با استفاده از برنامه های زیر ایجاد می‌شود :

```
xx1 = linspace(−3,3,15);
xx2 = linspace(−3,13,17);
[x1,x2] = meshgrid(xx1,xx2);
z = x1.^4 + 3*x1.^2 − 2*x1 + 6 − 2*x2.*x1.^2 + x2.^2 − 2*x2;
meshz(x1,x2,z)
colormap([0 0 1])
```

شکل ۷ –۱۰ را می‌توان با تعویض *meshz* در برنامهٔ فوق توسط عبارت ؛

```
waterfall
```



شکل ۷–۱۰) سطح ایجاد شده توسط دستور *waterfall*

ایجاد نمود. آرگومانهای تابع *waterfall* مشابه آرگومانهای تابع *meshz* خواهند بود.

همچنین سطوح را می توان با استفاده از تابع ؛

contour

برای ترسیم کانتورهای دو بعدی و تابع ؛

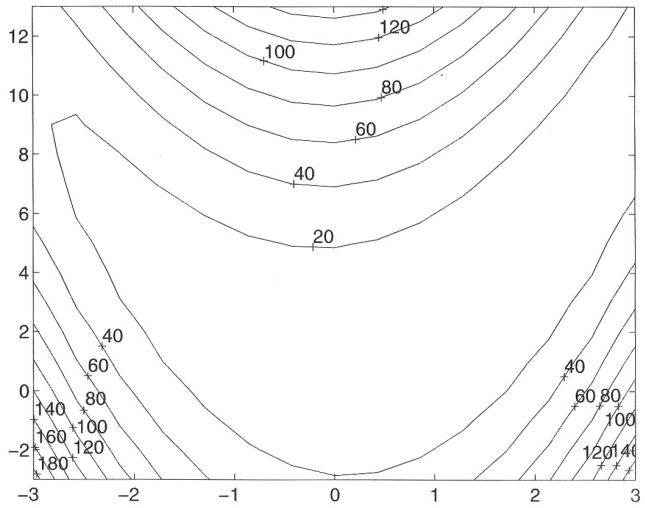
contour3

برای ترسیم کانتورهای سه بعدی و تابع ؛

contourf

برای ترسیم کانتورهای هاشور خورده، ایجاد نمود. توابع *contour* و *contour3* را می‌توان توسط دستور ؛

clabel



شکل ۷-۱۱) کانتور نشانه گذاری شده توسط دستورات *contour* و *clabel*

نشانه گذاری نمود. تابع *contourf* توسط دستور زیر

colorbar

نشانه گذاری می‌شود. که ستونی متشکل از رنگها و مقادیر عددی متناظرشان را در نزدیکی شکل قرار می‌دهد. اکنون کاربرد این توابع را نشان خواهیم داد.

جهت بدست آوردن کانتورهای دو بعدی سطوح نشان داده شد در شکل ۷ -٤، برنامهٔ زیر را خواهیم داشت:

```
xx1 = linspace(−3,3,15);
xx2 = linspace(−3,13,17);
[x1,x2] = meshgrid(xx1,xx2);
z = x1.^4 + 3*x1.^2 − 2*x1 + 6 − 2*x2.*x1.^2 + x2.^2 − 2*x2;
h = contour(x1,x2,z)
clabel(h)
```

اجرای این برنامه منجر به شکل ۷-۱۱ خواهد شد. استفاده از عبارت $h=contour(\ldots)$ در هنگام استفاده از تابع *clabel* ضروری می‌باشد. اگر از تابع *clabel* استفاده نشود - یعنی خطوط کانتور بدون مقادیر عددیشان نمایش داده شوند- در اینصورت تنها عبارت $contour(\ldots)$ استفاده می‌شود. اگر تابع *contour* را با تابع *contour3* جایگزین کنیم، شکل ۷ -۱۲ را خواهیم داشت. آرگومانهای تابع *conour*، *contour3* یکسان می‌باشند.

جهت بدست آوردن یک نقشهٔ کانتور رنگی هاشور خورده از تابع *contourf* و تابع *colorbar* مطابق زیر استفاده خواهیم کرد:

```
xx1 = linspace(-3,3,15);
xx2 = linspace(-3,13,17);
[x1,x2] = meshgrid(xx1,xx2);
z = x1.^4 + 3*x1.^2 - 2*x1 + 6 - 2*x2.*x1.^2 + x2.^2 - 2*x2;
contourf(x1,x2,z);
colorbar('vert')
```

که در آن *vert* در تابع *colorbar* جهت تعیین شیوهٔ نمایش به صورت ستونی بکار رفته است. اجرای این برنامه نتایج نشان داده شده در شکل ۷ -۱۳ را نمایش خواهد داد.

کاربر همچنین می تواند با استفاده از دستور ؛

```
cylinder
```

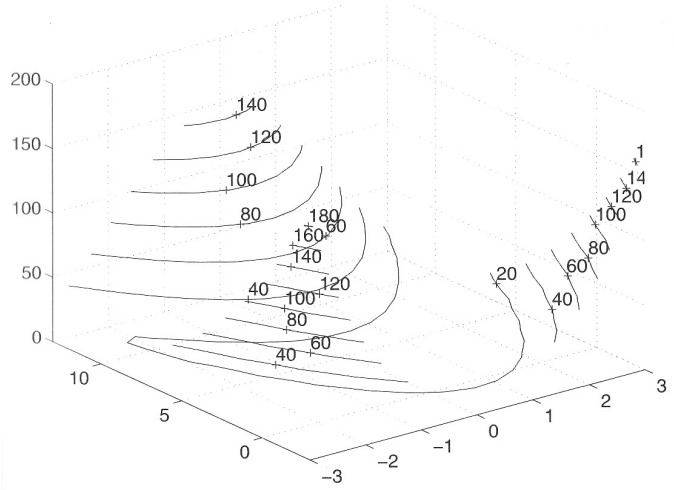
از منحنی‌های دو بعدی به عنوان مولدی برای ایجاد سطوح احجام دوران داده شده استفاده نماید. جهت روشنتر شدن مطلب منحنی ؛

$$y = 1.1 + \sin(z) \qquad 0 \le z \le 2\pi$$

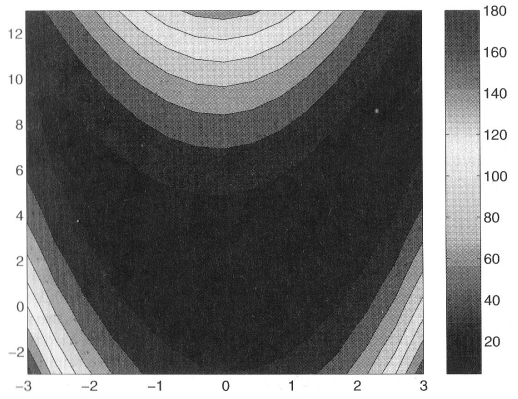
را در نظر بگیرید که حول محور z به اندازهٔ 360° چرخیده است. برنامه بصورت زیر می‌باشد:

```
[x,y,z] = cylinder(1.1+sin(0:0.25:2*pi),16);
surf(x,y,z)
axis off
```

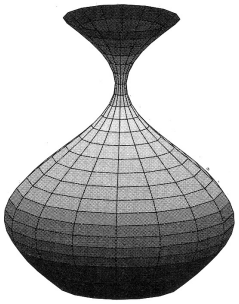
که پس از اجرا نتایج داده شده در شکل ۷ -۱٤ را نمایش خواهد داد.



شکل ۷ – ۱۲) کانتور سه بعدی نشانه گذاری شده توسط دستورات *contour3* و *clabel*



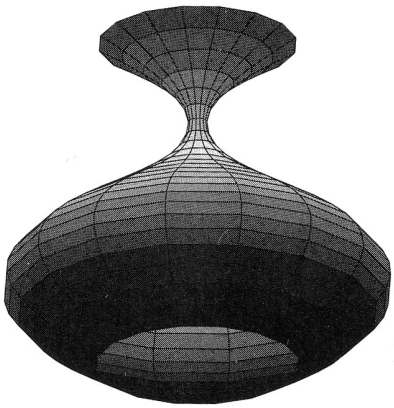
شکل ۷ – ۱۳) کاربرد دستورات *contourf* و *colorbar*



شکل ۷ – ۱٤) کاربرد دستور *cylinder*

در شکل ۷-۱٤، زوایای دید، مقادیر پیش فرض می‌باشند. مواردی وجود دارد که کاربر تمایل به تغییر زوایای دید پیش فرض بدلایل (۱) قسمتهایی از سطح که مورد نظر هستند، دیده نمی‌شوند. (۲) نشان دادن چند نمای مختلف از سطح با استفاده از تابع *subplot* مطلوب باشد. (۳) کاربر بخواهد قبل از تصمیم گیری در مورد نمای قطعی، نماهای مختلف شکل را مشاهده کند. مطلب دارای روشی مستقیم با استفاده از توابع *rotate3d* و *view* جهت تعیین نماهای دید می‌باشد. در شکل ۷-۱۵، زوایای دید به شیوهٔ زیر تعیین شده‌اند. در پنجرهٔ فرمان مطلب عبارت *rotate3d* تایپ می‌شود و یا آیکن *rotate* در پنجرهٔ شکل کلیک می‌شود. که اینکار کلیک چپ ماوس را فعال خواهد کرد. در پنجرهٔ فرمان، کلیک چپ ماوس فشرده شده و تا هنگامی که محورها به موقعیت جدیدی که نمای شکل بهتر شود، انتقال یابند، فشرده می‌ماند. سپس کلیک ماوس رها شده و ترسیم سه بعدی ایجاد می‌شود. اگر نما، رضایت بخش نباشد، در اینصورت این روند تا هنگامیکه نمای مناسبی ایجاد شود، ادامه خواهد یافت. پس از اینکه نمای مناسبی بدست آمد، به پنجرهٔ فرمان مطلب باز گشته و عبارت ؛

$$[az, elev] = view$$



شکل ۷ – ۱۵) شکل ۷ - ۱٤ پس از استفاده از دستورات *rotate3d* و *view*

را تایپ می‌کنیم. در اینصورت دو عدد نمایش داده می‌شوند، عدد اول برای *az* و عدد دوم برای *elev* (هر نام دیگری را می‌توان برای متغیرها بکار برد.) می‌باشد.

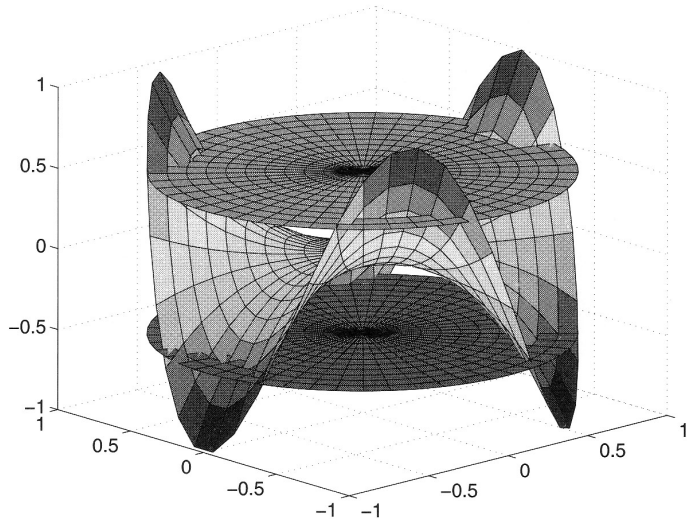
، کاربر پس از تابع ایجاد سطح در برنامه دستور ؛

$$view(n, m)$$

را وارد می‌کند. که در آن *n*، مقدار عددی برای *az* و *m*، مقدار عددی برای *elev* می باشد. با انجام موارد فوق مقادیر *az=-35.5* و *elev=-34* بدست خواهند آمد. بنابراین برنامهٔ اصلاح شده، به صورت زیر خواهد بود :

```
[x,y,z] = cylinder(1+sin(0:0.25:2*pi),16);
surf(x,y,z)
view(-35.5,-34)
axis off
```

که پس از اجرا، شکل ۷-۱۵ را نتیجه خواهد داد.



شکل ۷ – ۱۶) تلاقی دو دیسک با یک صفحه

همچنین می توان چندین سطح را در یک شکل ایجاد نمود. سطح زیر به معادلهٔ ؛

$$z(r,\theta) = r^3 \cos(3\theta)$$

را هنگامی که $0 \le r \le 1$ و $0 \le \theta \le 2\pi$ می‌باشد، در نظر بگیرید. فرض می‌شود که این سطح دو دیسک با شعاع *1* را که در *z=10.5* واقعند، قطع می‌کند. برنامهٔ ایجاد این سطوح به صورت زیر می‌باشد :

```
nr = 12; nth = 50;
```

```
r = linspace(0,1,nr);
theta = linspace(0,2*pi,nth);
[R,T] = meshgrid(r,theta);
x = cos(theta')*r;
y = sin(theta')*r;
surf(x,y,R.^3.*cos(3*T))
hold on
z0 = repmat(0.5,size(x));
surf(x,y,z0)
surf(x,y,-z0)
view(-42.5,20)
```

که پس از اجرا، شکل ۷-۱۶ را نتیجه خواهد داد.

مثال ۷-۲ ایجاد صفحات

فرض کنید سه نقطهٔ $P_0(x_0,y_0,z_0)$ ، $P_1(x_1,y_1,z_1)$ و $P_2(x_2,y_2,z_2)$ در فضا مشخص شده باشد، نمایش پارامتریک هر نقطه ای در صفحهٔ شامل این نقاط توسط رابطهٔ ؛

$$P = P_0 + sv + tw$$

تعیین می‌شود. که در آن ؛

$$P = xi + yj + zk$$

$$P_0 = x_0 i + y_0 j + z_0 k$$

$$v = v_1 i + v_2 j + v_3 k = (x_1 - x_0)i + (y_1 - y_0)j + (z_1 - z_0)k$$

$$w = w_1 i + w_2 j + w_3 k = (x_2 - x_0)i + (y_2 - y_0)j + (z_2 - z_0)k$$

و $0 \le s \le 1$ و $0 \le t \le 1$ می باشد. بنابراین ،

$$x = x_0 + sv_1 + tw_1 = x_0 + s(x_1 - x_0) + t(x_2 - x_0)$$

$$y = y_0 + sv_2 + tw_2 = y_0 + s(y_1 - y_0) + t(y_2 - y_0)$$

$$z = z_0 + sv_3 + tw_3 = z_0 + s(z_1 - z_0) + t(z_2 - z_0)$$

خواهد بود. اگر فرض شود که می‌توان صفحه را بصورت سطحی متشکل از شبکه ای 5×5 از قطعات در نظر گرفت، در اینصورت می‌توانیم تابعی با نام *PlanarSurface* جهت ایجاد و نمایش سطح صفحه‌ای ایجاد کنیم.

```
function  PlanarSurface(P0,P1,P2)
v = P1 – P0;
w = P2 – P0;
S = 0 : 0.2 : 1;
[s,t] = meshgrid(S,S);
xx = P0(1) + s * v(1) + t * w(1);
yy = P0(2) + s * v(2) + t * w(2);
zz = P0(3) + s * v(3) + t * w(3);
surf(xx,yy,zz)
hold  on
```

که در آن P_0 ، P_1 و P_2 هر یک بردارهایی سه عنصره شامل مختصه‌های نقاط روی صفحه می‌باشند. بنابراین با وارد کردن عبارت زیر در پنجرهٔ فرمان مطلب ؛

```
PlanarSurface([0 0 0],[2 6 3],[7 1 5])
```

نتایج بدست آمده در شکل ۷–۱۷ را خواهیم داشت.

جهت تصویر کردن سطح رو سه صفحهٔ اصلی عمود بر هم، از علامت ضرب نقطه‌ای استفاده می‌کنیم. لذا برای تصویر نمودن سطح روی صفحهٔ xy داریم :

$$P.(i + j + 0k)$$

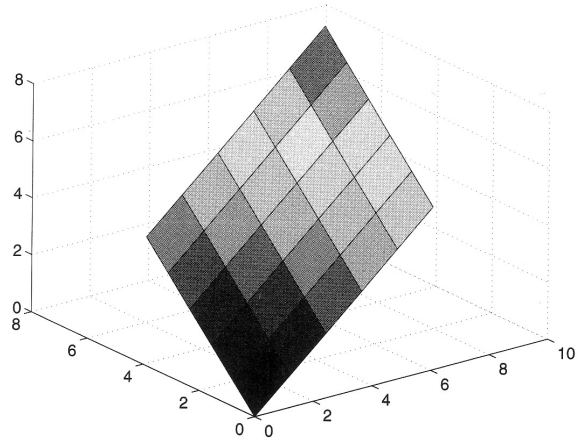
که معادل با مینیمم کردن مختصه‌های z نقاط روی صفحه می‌باشد. بطور مشابه برای تصویر کردن سطح روی صفحهٔ yz داریم :

$$P.(0i+j+k)$$

که معادل با مینیمم کردن مختصه‌های x نقاط روی صفحه می‌باشد. جهت تصویر کردن سطح روی صفحهٔ xz داریم:

$$P.(i+0j+k)$$

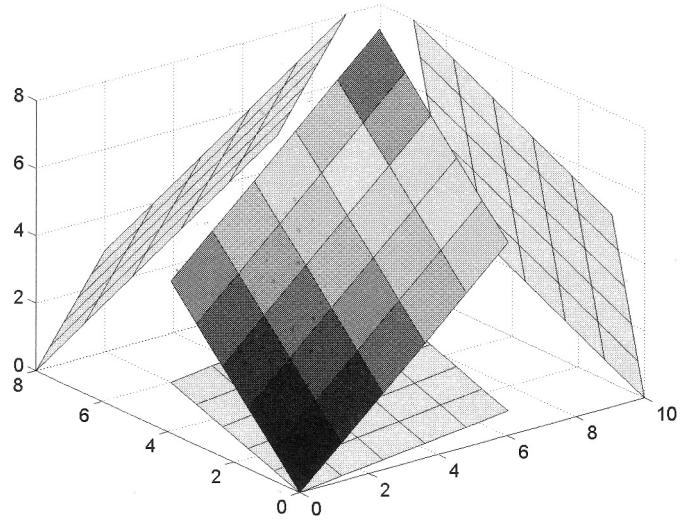
که معادل با مینیمم کردن مختصه‌های y نقاط روی صفحه می‌باشد. لذا می‌توان تابع *SurfacePlanar* را بگونه‌ای اصلاح کرد، که شامل گزینه‌ای اختیاری جهت نمایش این تصاویر باشد.



شکل ۷ – ۱۷) ایجاد یک سطح صفحه‌ای

```
function PlanarSurface(P0,P1,P2,projection)
v = P1 – P0;
w = P2 – P0;
S = 0 : 0.2 : 1; L = length(S);
[s,t] = meshgrid(S,S);
```

```
xx = P0(1) + s*v(1) + t*w(1);
yy = P0(2) + s*v(2) + t*w(2);
zz = P0(3) + s*v(3) + t*w(3);
surf(xx,yy,zz)
hold on
if nargin > 3
a = axis;
c(1:L,1:L,1:3) = zeros(L,L,3);
c(:,:,1) = 1;
c(:,:,2) = 1;
c(:,:,3) = 0;
surf(xx,yy,a(5)*ones(L,L),c)
```



شکل ۷ – ۱۸) تصویر یک صفحه روی صفحات مرجع مختصات

```
surf(xx,a(4)*ones(L,L),zz,c)
surf(a(2)*ones(L,L),yy,zz,c)
end
```

که در آن *projection* می‌تواند هر عددی باشد. آرایهٔ c بگونه‌ای تعریف می‌شود که تصاویر به رنگ زرد نمایش داده می‌شوند. دو اندیس اول آرایهٔ c باید دارای مرتبهٔ یکسان با xx، yy و zz باشند. اندیس آخری باید دقیقاً سه عنصر را نشان بدهد، که هر یک می‌توانند دارای مقداری بین 0 و 1 باشد. این سه عنصر آخری رنگ هر قطعه را در هر ترکیبی از دو اندیس اول تعیین می‌کنند. اگر عبارت زیر را در پنجرهٔ فرمان مطلب وارد کنیم ؛

```
PlanarSurface([0  0  0],[2  6  3],[7  1  5],1)
```

نتایج نشان داده شده در شکل ۷-۱۸ را خواهیم داشت. اگرچه تصور می‌رود که تصاویر، در صفحات مورد نظر تصویر نشده‌اند، ولی استفاده از تابع *rotate3d* ما را از این بابت مطمئن می‌سازد.

مثال ۷-۳ ایجاد مکعب ها

اکنون می‌خواهیم از نتایج مثال ۷-۲ جهت ایجاد سه مکعب با ابعاد $L_x \times L_y \times L_z$ استفاده نماییم. با مراجعه به شکل ۷-۱ خواهیم دید که هر جعبه از شش صفحه تشکیل می‌شود، که هر یک شامل سه دسته نقطه جهت تعریف آنها، مطابق زیر می‌باشند :

سطوح عمود بر صفحهٔ yz

$$P_0(x_0,y_0,z_0),\ P_1(x_0,y_0,z_0+L_z),\ P_2(x_0,y_0+L_y,z_0)$$

$$P_0(x_0+L_x,y_0,z_0),\ P_1(x_0+L_x,y_0,z_0+L_z),\ P_2(x_0+L_x,y_0+L_y,z_0)$$

سطوح عمود بر صفحهٔ xz

$$P_0(x_0,y_0,z_0),\ P_1(x_0,y_0,z_0+L_z),\ P_2(x_0+L_x,y_0,z_0)$$

$$P_0(x_0,y_0+L_y,z_0),\ P_1(x_0,y_0+L_y,z_0+L_z),\ P_2(x_0+L_x,y_0+L_y,z_0)$$

سطوح عمود بر صفحهٔ xy

$$P_0(x_0,y_0,z_0),\ P_1(x_0+L_x,y_0,z_0),\ P_2(x_0,y_0+L_y,z_0)$$

$P_0(x_0, y_0, z_0 + L_z),\ P_1(x_0 + L_x, y_0, z_0 + L_z),\ P_2(x_0, y_0 + L_y, z_0 + L_z)$

از روابط فوق در تابع زیر که *Box Surface* نامیده می‌شود، استفاده خواهیم کرد:

```
function  BoxSurface( P0,L )
PlanarSurface( P0,P0+[0  0  L(3)],P0+[0  L(2)  0])
PlanarSurface( P0+[ L(1)  0  0],P0+[ L(1)  0  L(3)],P0+[L(1)  L(2)  0])
PlanarSurface( P0,P0+[0  0  L(3)],P0+[L(1)  0  0])
PlanarSurface( P0+[0  L(2)  0],P0+[0  L(2)  L(3)],P0+[L(1)  L(2)  0])
PlanarSurface( P0,P0+[ L(1)  0  0],P0+[0  L(2)  0])
PlanarSurface( P0+[0  0  L(3)],P0+[ L(1)  0  L(3)],P0+[0  L(2)  L(3)])
```

که در آن $P_0 = [x_0, y_0, z_0]$ ، $L = [L_x, L_y, L_z]$ و *PlanarSurface* در مثال ۷-۲ داده شده است.

برنامهٔ زیر که سه مکعب با ابعاد مربوطه و مکان یکی از گوشه‌های هر یک از آنها را ایجاد می‌کند، در نظر بگیرید ؛

جعبهٔ #۱

اندازه: $7 \times 5 \times 3$

موقعیت: $(1,1,1)$

جعبهٔ #۲

اندازه: $1 \times 5 \times 4$

موقعیت: $(3,4,5)$

جعبهٔ #۳

اندازه: $1 \times 1 \times 1$

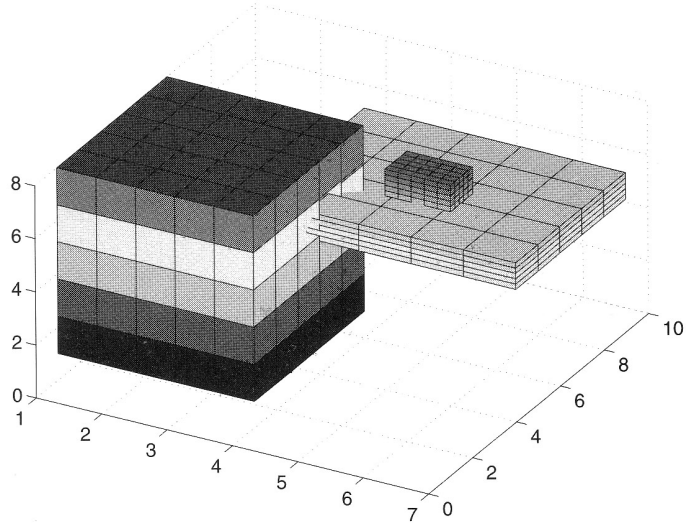
موقعیت: $(4.5,5.5,6)$

برنامهٔ مورد نیاز جهت ایجاد و نمایش این مکعبها بصورت زیر می‌باشد:

```
BoxSurface([1,1,1],[3,5,7])
```

```
BoxSurface([ 3,4,5 ],[ 4,5,1])
BoxSurface([ 4.5,5.5,6 ],[ 1,1,1])
view( 29.5,44 )
```

که پس از اجرا، شکل ۷-۱۹ را نتیجه خواهد داد.



شکل ۷ – ۱۹) سه مکعب

مثال ۷-۴ چرخش و انتقال اجسام سه بعدی: شاسی اتومبیل

چرخش و انتقال نقطهٔ $p(x,y,z)$ به مکان دیگر $P(X,Y,Z)$ توسط روابط ؛

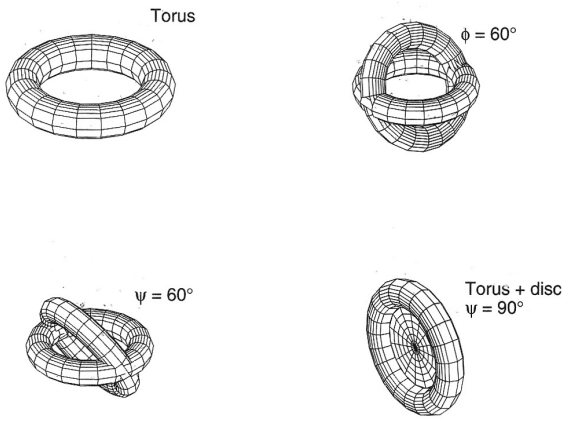
$$X = L_x + a_{11}x + a_{12}y + a_{13}z$$

$$Y = L_y + a_{21}x + a_{22}y + a_{23}z$$

$$Z = L_z + a_{31}x + a_{32}y + a_{33}z$$

تعیین می‌شود. که در آن L_x ، L_y و L_z به ترتیب مختصه‌های (x,y,z) مربوط به انتقال و a_{ij} ها عناصر ماتریس ؛

$$a=\begin{bmatrix} \cos\psi\cos\chi & -\cos\psi\sin\chi & \sin\psi \\ \cos\phi\sin\chi+\sin\phi\sin\psi\cos\chi & \cos\phi\cos\chi-\sin\phi\sin\psi\sin\chi & -\sin\phi\cos\psi \\ \sin\phi\sin\chi-\cos\phi\sin\psi\cos\chi & \sin\phi\cos\chi+\cos\phi\sin\psi\sin\chi & \cos\phi\cos\psi \end{bmatrix}$$



شکل ۷-۲۰) دورانهای یک چمبره و یک چمبره به همراه یک دیسک داخلی

می‌باشند. کمیتهای ϕ ، ψ و χ زوایای چرخش منظم (زوایای اولر) دستگاه مختصات حول مبدا می‌باشند : ϕ زاویهٔ چرخش حول محور x ، ψ زاویهٔ چرخش حول محور y و نهایتاً χ زاویهٔ چرخش حول محور z می‌باشد. بطور کلی *(x,y,z)* ها می‌توانند اسکالر، بردارهایی با طول یکسان، یا ماتریسهایی با مرتبهٔ یکسان باشند.

قبل از اعمال کردن این روابط، تابع زیر را جهت تکمیل آنها ایجاد می‌کنیم:

```
function [ Xrt, Yrt, Zrt ] = EulerAngles( psi,chi, phi,Lx,Ly,Lz,x,y,z )
R = [cos( psi )* cos( chi ),-cos( psi )* sin( chi ),sin( psi );
   cos( phi )* sin( chi )+ sin( phi )* sin( psi )* cos( chi ),
   cos( phi )* cos( chi )- sin( phi )* sin( psi )* cos( chi ),
 - sin( phi )* cos( psi );sin( phi )* sin( chi ),
 - cos( phi )* sin( psi )* cos( chi ),sin( phi )* cos( chi )+
```

$cos(phi) * sin(psi) * sin(chi), cos(phi) * cos(psi)];$

$Xrt = R(1,1) * x + R(1,2) * y + R(1,3) * z + Lx;$

$Yrt = R(2,1) * x + R(2,2) * y + R(2,3) * z + Ly;$

$Zrt = R(3,1) * x + R(3,2) * y + R(3,3) * z + Lz;$

اکنون چگونگی استفاده از این توابع انتقال را با ایجاد تغییراتی در یک چمبره، که جهت نمایش تایر اتومبیل از آن استفاده می‌شود، نشان می‌دهیم معادلهٔ آن بصورت زیر می‌باشد :

$$z = \pm\sqrt{a^2 - \left(\sqrt{x^2 + y^2} - b\right)^2}$$

که در آن ؛

$x = r\cos\theta$

$y = r\sin\theta$

و $b - a \le r \le b + a$ ، $0 \le \theta \le 2\pi$ و $b > a$ خواهد بود.

ابتدا خود چمبره را رسم می‌کنیم. سپس چمبره ای را که 60° درجه حول محور x چرخیده است $(\phi = 60^\circ)$ و سپس 60° درجه حول محور y دوران نموده است $(\psi = 60^\circ)$ را ترسیم نموده و در نهایت دیسکی را که در صفحهٔ $z = 0$ کنار چمبره قرار می‌دهیم و هر دوی آنها را به اندازهٔ 90° درجه حول محور y دوران می‌دهیم $(\psi = 90^\circ)$.

نتایج در شکل ۷-۲۰ نشان داده شده است. قبل از نوشتن برنامه، ابتدا دو تابع اضافی، ایجاد خواهیم نمود. تابع اول هندسهٔ چمبره را توصیف می‌کند.

$function \quad [X, Y, Z] = CarTire(a, b)$

$r = linspace(b - a, b + a, 10);$

$th = linspace(0, 2 * pi, 22);$

$x = r' * cos(th);$

$y = r' * sin(th);$

$z = real(sqrt(a^2 - (sqrt(x.^2 + y.^2) - b).^2));$

$X = [x \quad x];$

```
Y = [ y    y ];
Z = [ z   − z ];
```

که در آن تابع *real* جهت بخشهای موهومی کوچک ایجاد شده توسط خطاهای عددی بکار می‌رود. تابع دوم جهت ایجاد دیسک بکار برده می‌شود.

```
function   [ XD, YD, ZD ] = CarDisk( a,b )
diskr = linspace( 0,b − a,7 );
th = linspace( 0,0.2 * pi,16 );
XD = discr' * cos( th );
YD = discr' * sin( th );
ZD = zeros( 7,16 );
```

به ازای *z=0.2* و *b=0.8* برنامه به صورت زیر خواهد بود:

```
a = 0.2; b = 0.8;
[ X   Y   Z ] = CarTire( a,b );
[ XD   YD   ZD ] = CarDisk( a,b );
Lx = 0; Ly = 0; Lz = 0;
for   k = 1 : 4
subplot( 2,2,k )
switch   k
case  1
mesh( X ,Y ,Z )
v = axis;
axis([ v( 1)   v( 2)   v( 3)   v( 4)   −1   1])
```

```
text(0.5,–0.5,1,'Torus' )
case  2
psi = 0; chi = 0; phi = pi / 3;
[ Xr   Yr   Zr ] = EulerAngles( psi,chi, phi,Lx,Ly,Lz,X,Y,Z );
mesh( X,Y,Z )
hold   on
mesh( Xr,Yr,Zr )
text(0.5,–0.5,1,'\phi = 60 \circ' )
case  3
psi = pi / 3; chi = 0; phi = 0;
[ Xr   Yr   Zr ] = EulerAngles( psi,chi, phi,Lx,Ly,Lz,X,Y,Z );
mesh( X,Y,Z )
hold   on
mesh( Xr,Yr,Zr )
text(0.5,–0.5,1,'\psi = 60 \circ' )
case  4
psi = pi / 2; chi = 0; phi = 0;
[ Xr   Yr   Zr ] = EulerAngles( psi,chi, phi,Lx,Ly,Lz,X,Y,Z );
[ Xd   Yd   Zd ] = EulerAngles( psi,chi, phi,Lx,Ly,Lz,XD,YD,ZD );
mesh( Xd,Yd,Zd )
hold   on
mesh( Xr,Yr,Zr )
text(0.5,–0.5,0.7,'\psi = 90 \circ' )
```

```
text(0.5,−0.5,1,'Torus + disc' )
end
colormap([0  0  1])
axis  equal
axis  off
grid  off
end
```

اکنون از چمبره و دیسک داخلی استفاده کرده و شاسی ماشین و تایر هایش را همانگونه که در شکل ۷-۲۱ نشان داده شده است، ایجاد خواهیم کرد. جهت انجام اینکار، همچنین نیازمند ایجاد استوانه که به شکل ؛

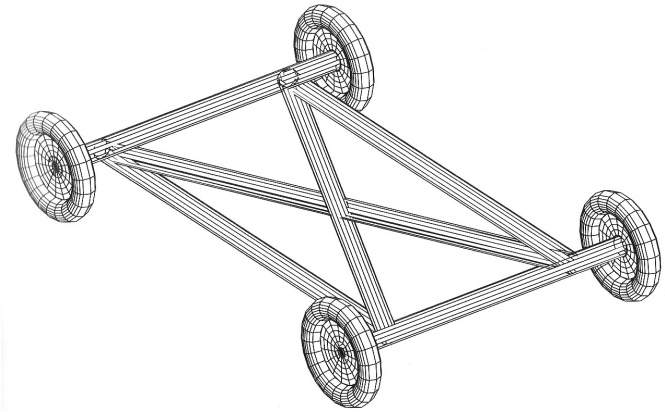
$$x = r\cos\theta$$

$$y = r\sin\theta$$

$$z = z$$

تعریف می‌شود، خواهیم بود. فرض می‌شود که قطر استوانه $0.3(b-a)$ می‌باشد و طول آن بستگی به نوع المانی که نمایش داده می‌شود، خواهد داشت. جهت افزایش خوانایی برنامه، یک تابع اضافی که شفتی به طول $2L_2$ ایجاد می‌کند، خواهیم نوشت ؛

```
function  [ XC,YC,ZC ] = CarShaft( a,b,L2 )
th = linspace(0,2* pi,16 );
[ XC,ZC ] = meshgrid(0.3*(b − a)*cos(th),[−L2   L2]);
YC = meshgrid(0.3*(b − a)*sin(th),[−L2   L2]);
```



شکل ۷ – ۲۱) شاسی اتومبیل

برنامه بصورت زیر خواهد بود:

```
a = 0.2; b = 0.8;

[ X   Y   Z ] = CarTire( a,b );

[ XD   YD   ZD ] = CarDisk( a,b );

psi = [ pi / 2   pi / 2   pi / 2   pi / 2 ]; chi = [ 0   0   0   0]; phi = [ 0   0   0   0];
Lx = [ 3   - 3   3   - 3]; Ly = [0   0   8   8]; Lz = [0   0   0   0];

for   k = 1 : 4

[ Xr   Yr   Zr] = EulerAngles(psi(k),chi(k), phi(k),Lx(k),Ly(k),Lz(k),X,Y,Z);
[ Xd   Yd   Zd] = EulerAngles(psi(k),chi(k), phi(k),Lx(k),Ly(k),Lz(k),XD,YD,ZD);
mesh( Xr,Yr,Zr )

end

psi = [ pi / 2   pi / 2   0   0   atan(0.5)   - atan(0.5)];

chi = [0   0   0   0   0   0];

phi = [0   0   pi/2   pi/2   pi/2   pi/2];

Lx = [0   0   2   - 2   0   0]; Ly = [0   8   4   4   4   4]; Lz = [0   0   0   0   0   0];
for  k = 1 : 6
```

```
switch  k
case  {1,2}
[ XC,YC,ZC ] = CarShaft( a,b,[−3   3]);
case  {3,4}
[ XC,YC,ZC ] = CarShaft( a,b,[−4   4]);
case  {5,6}
[ XC,YC,ZC ] = CarShaft( a,b,[−4.4721   4.4721]);
end
[ Xc   Yc   Zc] = EulerAngle s(psi(k), chi(k), phi(k), Lx(k), Ly(k), Lz(k), XC, YC, ZC);
mesh( Xc,Yc,Zc )
end
colormap([0  0  1])
axis  equal
axis  off
```

تمرینها

۷ – ۱ منحنی‌های سه بعدی زیر را ترسیم کنید. از تابع *axis equal* استفاده کنید.

مارپیچ کروی $(c = 5.0,\ \ 0 \le t \le 10\pi)$

$$x = \sin(t/2c)\cos(t)$$

$$y = \sin(t/2c)\sin(t)$$

$$z = \cos(t/2c)$$

موج سینوسی روی استوانه $(a = 10.0,\ \ b = 1.0,\ \ c = 0.3,\ \ 0 \le t \le 2\pi)$

$$x = b\cos(t)$$

$$y = b\sin(t)$$
$$z = c\cos(at)$$

موج سینوسی روی کره $(a = 10.0,\ b = 1.0,\ c = 0.3,\ 0 \le t \le 2\pi)$

$$x = \cos(t)\sqrt{b^2 - c^2\cos^2(at)}$$
$$y = \sin(t)\sqrt{b^2 - c^2\cos^2(at)}$$
$$z = c\cos(at)$$

مارپیچ فنری $(a = 0.2,\ b = 0.8,\ c = 20.0,\ 0 \le t \le 2\pi)$

$$x = [b + a\sin(ct)]\cos(t)$$
$$y = [b + a\sin(ct)]\sin(t)$$
$$z = a\cos(ct)$$

۲-۷ سطوح مربوط به احجام زیر را ترسیم کنید. از شکل برداری تبدیل مختصات $x = r\cos(\theta)$ و $y = r\sin(\theta)$ یا $x = a\cos(\theta)$ و $y = b\sin(\theta)$ ، هر یک که مناسب است استفاده کنید. r (یا b,a) باید دارای طول 10 و θ برداری به طول 22 باشد. در تمامی این اشکال باید از تابع *axis equal* استفاده نمایید.

کره $(r = 1,\ 0 \le \theta \le 2\pi)$

$$z = \sqrt{1 - x^2 - y^2}$$

بیضیگون $(a = 1.0,\ b = 1.5,\ c = 2.0)$

$$z = c\sqrt{1 - x^2/a^2 - y^2/b^2}$$

کروی دو سر تخت (بیضیگون با پارامترهای $c = 0.5 : a = b = 1.0 : a = b > c$)

کروی دو سر بر جسته (بیضیگون با پارامترهای $c = 1.2 : a = b = 1.0 : a = b < c$)

مخروط $(-2 \le x \le 2, -2 \le y \le 2)$ {دو بار از تابع *rotate3d* جهت بررسی سطح استفاده کنید.}

$$z = \pm\sqrt{x^2 + y^2}$$

که در آن $x = r\cos\theta$ ، $y = r\sin\theta$ ، $0 \le r \le 2$ و $0 \le \theta \le 2\pi$ می‌باشد.

شاخی شکل $(a = 0.3,\ b = 0.5,\ 0 \le u \le 2\pi,\ -3 \le v \le 3)$ {از تابع *rotate3d* جهت بررسی سطح استفاده کنید.}

$$x = e^{bv}\cos(v) + e^{av}\cos(u)\cos(v)$$

$$y = e^{bv}\sin(v) + e^{av}\cos(u)\sin(v)$$

$$z = e^{av}\sin(u)$$

۷ - ۳ سطوح زیر را ترسیم کرده و از تابع *rotate3d* جهت بررسی آنها استفاده کنید.

الف) $(a = b = 1, c = 0.5, -3 \le x \le 3, -3 \le y \le 3)$

$$z = c\left((x/a)^4 \pm (y/b)^4\right)$$

ب) $(a = 3, c = 0.25, -1 \le x \le 1, -1 \le y \le 1)$

$$z = c\sin\left(2\pi a\sqrt{x^2 + y^2}\right)$$

پ) $(a = 3, c = 0.25, -1 \le x \le 1, -1 \le y \le 1)$

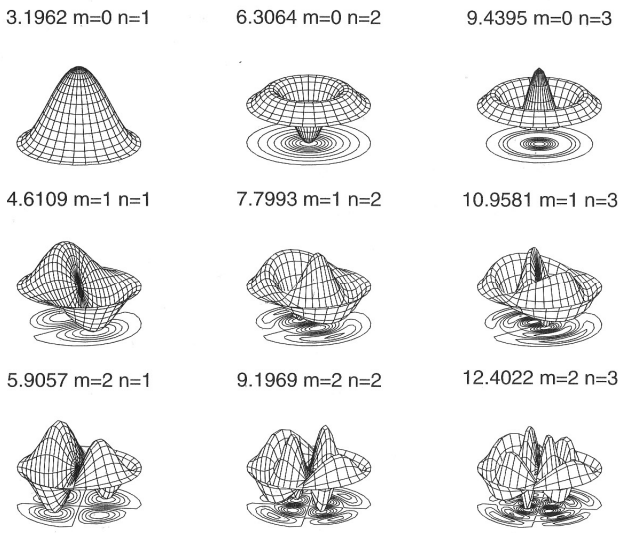
$$z = c\sin(2\pi axy)$$

ت) $(c = 0.2, -1 \le x \le 1, -1 \le y \le 1; x \ne 0, y \ne 0)$

$$z = c\ln(|xy|)$$

ث) $(1 \le u \le 5, 0 \le v \le 2\pi)$

$$x = u\cos(v)$$



شکل ۷ – ۲۲) شکل مدهای یک صفحهٔ دایروی صلب

$$y = u\,sin(v)$$

$$z = cosh^{-1}(u)$$

ج) $(c = 1/2\pi,\ \ -0.5 \le u \le 0.5,\ \ -2\pi \le v \le 2\pi)$

$$x = u\,cos(v)$$

$$y = u\,sin(v)$$

$$z = cv$$

۷ – ٤ شکل مدهای یک صفحهٔ دایروی صلب که دو لبهٔ خارجی اش یعنی r=b کاملاً مقیود شده است به صورت زیر می‌باشد:

$$w_{mn}(r,\theta) = [C_{mn}J_m(\Omega_{mn}r/b) + I_m(\Omega_{mn}r/b)]\,cos(m\theta)$$

که در آن $m = 0,1,2,...,J_m(x)$ تابع بسل نوع اول از مرتبهٔ m و $I_m(x)$ شکل اصلاح شدهٔ تابع بسل نوع اول از مرتبهٔ m می‌باشد، همچنین ؛

$$C_{mn} = -\frac{I_m(\Omega_{mn})}{J_m(\Omega_{mn})}$$

و Ω_{mn} حل های مربوط به معادلهٔ ؛

$$J_m(\Omega_{mn})I_{m+1}(\Omega_{mn}) + I_m(\Omega_{mn})J_{m+1}(\Omega_{mn}) = 0$$

می‌باشند و در تمرین ۵-۲ قسمت (ج) محاسبه شده‌اند.

با استفاده از نتایج تمرین ۵-۲ ج، که در آن ضرایب سه فرکانس طبیعی پایین به ازای $m=0,1,2$ تعیین شدند، نُه شکل مد متناظر را با استفاده از توابع *surfc* و *subplot* روی یک شکل نشان دهید. و در بالای هر شکل مقدار m، n و ضریب فرکانس مربوطه را ثبت کنید. محورهای x و y را مخفی کنید. اولین سطر برای $m=0$، دومین سطر برای $m=1$ و سطر سوم برای $m=3$ می‌باشد. هر شکل مد را با استفاده از تابع *max*، دو مرتبه نرمالیزه نمایید (چون میدان تغییر مکان یک ماتریس می‌باشد) تا اینکه مقدار ماکزیمم 1 باشد. توصیه می‌شود تعداد تقسیمات شعاعی (r/b) ، 15 و تعداد تقسیمات زاویه ای 30 انتخاب شود. نتایج باید مطابق شکل ۷-۲۲ باشد، که توسط استفاده از تابع *meshc* جهت وضوح بیشتر ایجاد شده است.

۷- ۵ قطعه‌ای به ضخامت $2L$ در جهت x و دارای ابعاد بزرگ در جهات y و z را در نظر بگیرید. اگر قطعه که در ابتدا $(t=0)$ دارای دمای یکنواخت و ثابت T_i می باشد، در معرض محیطی با دمای T_∞ قرار گیرد، در اینصورت توزیع دمایی قطعه بصورت تابعی از زمان و مکان بصورت زیر ؛

$$\frac{\theta}{\theta_i} = 2\sum \frac{\sin\delta_n \cos(\delta_n \eta)}{\delta_n + \sin\delta_n \cos(\delta_n)} exp\left(-\delta_n{}^2 \tau\right)$$

خواهد بود. که در آن $\theta=\theta(\eta,\tau)=T-T_\infty$، $T=T(\eta,\tau)$ دمای قطعه، $\eta=x/l$، $\theta_i=T_i-T_\infty$، $\tau=\alpha^2 t/L^2$ زمان بی‌بعد شده (برخی موارد بنام مدول فوریه نیز شناخته می‌شود)، α ضریب نفوذ حرارتی و δ_n پاسخ معادلهٔ ؛

$$\cot\delta_n = \frac{\delta_n}{B_i}$$

می باشد. که در آن $B_i = \bar{h}L/k$ عدد بایوت، $\bar{h}$ ضریب انتقال حرارت متوسط برای جابجایی از کل سطح و k ضریب هدایت گرمایی قطعه می باشد.

20 مقدار ابتدایی δ_n را برای $B_i = 0.7$ بیابید، و از آنها جهت ترسیم سطح $\theta(\eta,\tau)/\theta_i$، هنگامیکه $0 \le \eta \le 1$ و $0 \le \tau \le 2$ است، استفاده نمایید سپس از تابع *rotate3d* جهت یافتن نمای قابل قبولی از سطح استفاده کنید. محورها را نشانه گذاری کرده و شکل را عنوان دهی کنید. همچنین همانگونه که در شکل ۷ –۵ نشان داده شده است، خطوط عمودی را جهت واضح تر شدن سطح به آن بیافزایید.

۷ – ٦ شکل مدهای زیر و کانتورهای مربوطه را برای یک عضو مربعی که در مرز خارجی‌اش با استفاده از 25 نقطه شبکه در هر جهت برای $0 \le x \le 1$ و $0 \le y \le 1$ مقیود شده است، بیابید.

$$w_{23}(x,y) = \sin(2\pi x)\sin(3\pi y)$$

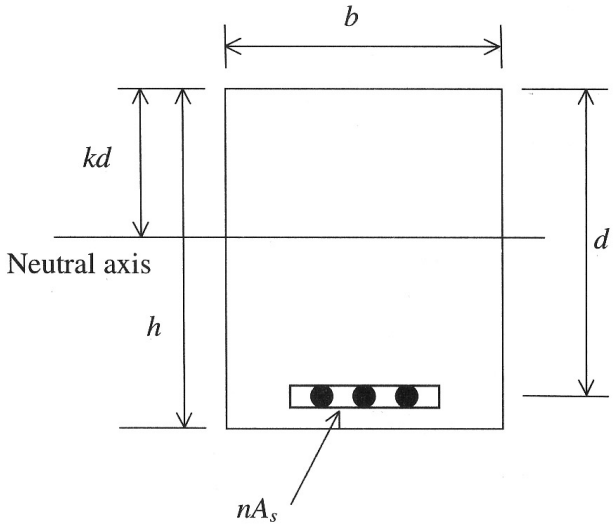
۷ – ۷ عدد نوسلت متوسط برای جریان متلاطم روی یک صفحه به طول L بصورت زیر می‌باشد:

$$Nu = \frac{0.037\,Re^{0.8}\,Pr}{1 + 2.443\,Re^{-0.1}\left(Pr^{2/3} - 1\right)} \qquad 5\times10^5 \le Re \le 10^7 \quad 0.6 \le Pr \le 2000$$

که در آن Re عدد رینولدز و Pr عدد پرانتل می‌باشد. تابع $log_{10}(Nu)$ را به صورت سطحی که تابع $log_{10}(Re)$ و $log_{10}(Pr)$ می‌باشد، در محدودهٔ تعیین شده رسم کنید. خطوط عمودی را از صفحهٔ مرزی شکل به گوشه های سطح، همانگونه که در شکل ۷ – ۵ نشان داده شده است، متصل کنید.

۷ – ۸ مکان تار خنثی در یک ستون بتنی مسلح شده با فولاد، که در شکل ۷ –۲۳ نشان داده شده است توسط پارامتر k که بصورت زیر تعریف می شود، تعیین خواهد شد :

$$k = -\rho n + \sqrt{(\rho n)^2 + 2\rho n}$$



شکل ۷-۲۳) مقطع یک ستون بتنی مسطح شده با فولاد

که درآن $\rho = A_s/bd$ و $n = E_s/E_c$ نسبت مدول یانگ فولاد به بتن می‌باشد. سطح k را به صورت تابعی از n و ρ برای ده مقدار n در بازهٔ $6 \le n \le 12$ و نُه مقدار ρ در بازهٔ $0.001 \le \rho \le 0.009$ و ده مقدار دیگر در بازهٔ $0.01 \le \rho \le 0.1$ ترسیم کنید.

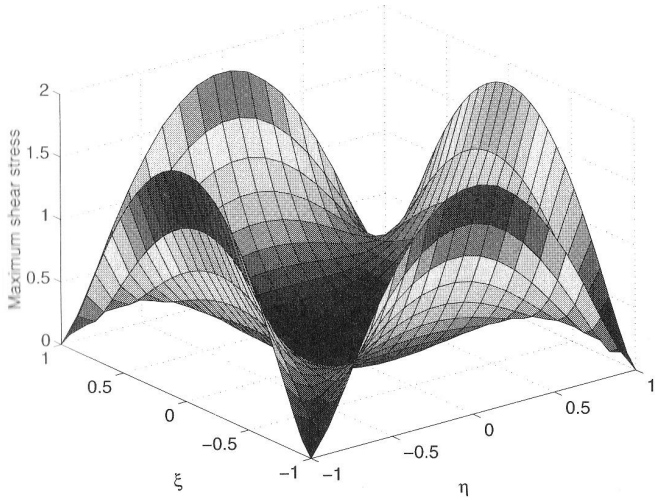
۷ - ۹ ماکزیمم تنش برشی بی بعد τ' در یک تیر مستطیلی که تحت گشتاور پیچشی T قرار دارد توسط رابطهٔ زیر ؛

$$\tau'^2 = \tau'^2_{xz} + \tau'^2_{yz}$$

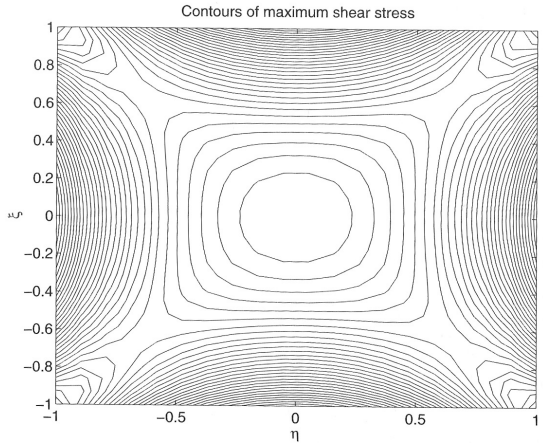
بدست می‌آید. که در آن ؛

$$\tau'_{xz} = \frac{\tau_{xz} J}{Ta} = -\frac{16}{\pi^2} \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)^2} \frac{sinh(k_n \xi)}{cosh(k_n b/a)} sin(k_n \eta)$$

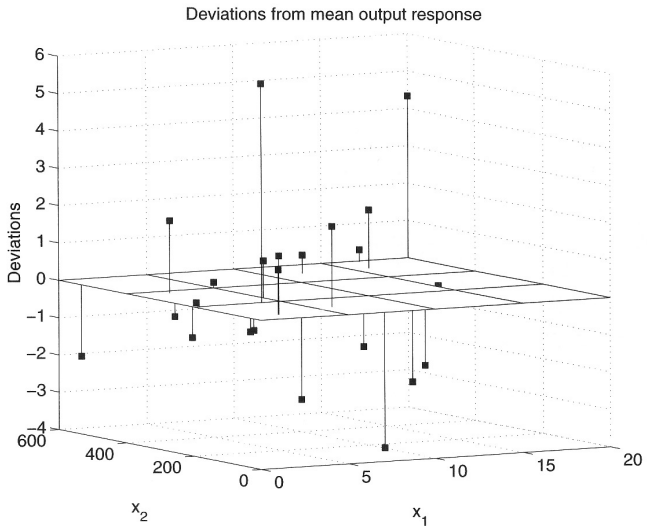
$$\tau'_{yz} = \frac{\tau_{yz} J}{Ta} = 2\eta - \frac{16}{\pi^2} \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)^2} \frac{cosh(k_n \xi)}{cosh(k_n b/a)} sin(k_n \eta)$$



شکل ۷ – ۲٤) مربع ماکزیمم تنش برشی مربوط به پیچش یک تیر با مقطع مستطیلی



شکل ۷ – ۲۵) نمایش کانتور ماکزیمم تنش برشی نشان داده شده در شکل ۷ – ۲٤



شکل ۷ – ۲٦) نمایش انحرافهای ایجاد شده از صفحهٔ $z = 0$ با استفاده از تابع *stem3*

و J ثابت پیچشی، $\eta = x/a(-1 \le \eta \le 1)$، $\xi = y/b(-1 \le \xi \le 1)$ و $k_n = (2n+1)\pi/2$ می‌باشد. موارد ذکر شده در ادامه را رسم کنید. سطحی برای $\tau'^2(\eta,\xi)$ به ازای $b/a = 1$، سطح مقطعی مربعی، و در شکلی جداگانه کانتورهایی از $\tau'^2(\eta,\xi)$ همراه با *30* خط کانتور. از تابع *rotate3d* جهت دیدن نماهای مختلف صفحه استفاده نمایید. نتایج باید مشابه شکلهای ۷-۲٤ و ۷- ۲۵ باشد.

۷- ۱۰ داده های جدول ۷-۱ را در نظر بگیرید. که مقادیر انحراف خروجی یک فرآیند بر اساس مقدار متوسط *z=0* را به ازای ورودیهای x_1 و x_2 نشان می‌دهد. از تابع ترسیم *stem3* جهت بدست آوردن شکل ۷ -۲٦ استفاده کنید. همچنین صفحه‌ای را توسط فرمانی جداگانه در *z=0* ایجاد کنید. از دستور *view(-30,7)* جهت دستیابی به نمای نشان داده شده استفاده نمایید. (شکل دو بعدی تابع *stem3* بصورت *stem* می‌باشد).

جدول ۷-۱ (انحراف خروجی‌های تحلیل نسبت به مقادیر متوسطشان)

x_1	x_2	z	x_1	x_2	Z
2	50	1.5713	2	360	-0.6023
8	110	-1.1460	4	205	5.8409
11	120	-2.2041	4	400	-0.3620
10	550	-1.5968	20	600	4.3341
8	295	-2.8937	1	585	-2.0368
4	200	1.1136	10	540	-1.5415
2	375	1.9297	15	250	0.0302
2	52	1.1962	15	290	-2.1809
9	100	-3.8650	16	510	1.5587
8	300	-0.4763	17	590	0.3222
4	412	-1.3223	6	100	2.1478
11	400	-0.4619	5	400	0.1537
12	500	0.4911			

فصل هشتم

سیمولینک

۸-۱ مقدمه

۸-۱-۱ مفهوم شبیه سازی سیستم های دینامیکی

۸-۱-۲ مفهوم سیگنال و جریان منطقی

۸-۱-۳ چگونگی اتصال بلوک ها

۸-۲ چگونگی دسترسی به کتابخانهٔ *SOURCES AND SINKS*

۸-۳ سیستم های پیوسته و گسسته

۸-٤ عملگرهای غیر خطی

۸-٥ نحوهٔ استفاده از توابع (نوشته شده به صورت *C*،*M* و...)

۸-٦ عملیات ریاضیاتی

۸-۷ انتقال داده و سیگنالها

۸-۸ بهینه کردن ظاهر بصری

۸-۸-۱ استفاده از زیر سیستم ها و ماسک ها

۸-۸-۲ ایجاد نمودن زیر سیستم ها

۸-۸-۳ کمکهای بصری

۸-۹ تعیین نمودن پارامترهای شبیه سازی

۸-۱۰ مفهوم سخت افزار در حلقه

۸-۱۱ چند راهنمایی و ترفند

در این فصل با قسمتهای مختلف جعبه ابزار سیمولینک در مطلب آشنا خواهیم شد. سپس چندین راهنمایی جهت هر چه بهتر استفاده کردن از این جعبه ابزار ارائه می‌شود.

۸ – ۱ مقدمه

امروزه کامپیوترها امکانات ریاضیاتی بسیار وسیعی را در اختیار مهندسین قرار می‌دهند، که می‌توان از آنها جهت شبیه سازی سیستم های دینامیکی استفاده نمود، که این مطلب باعث می شود که بدون نیاز به ایجاد و ساخت سیستم حقیقی، بتوان خواص و پاسخ آنرا به تحریکهای مختلف پیشگویی نمود.

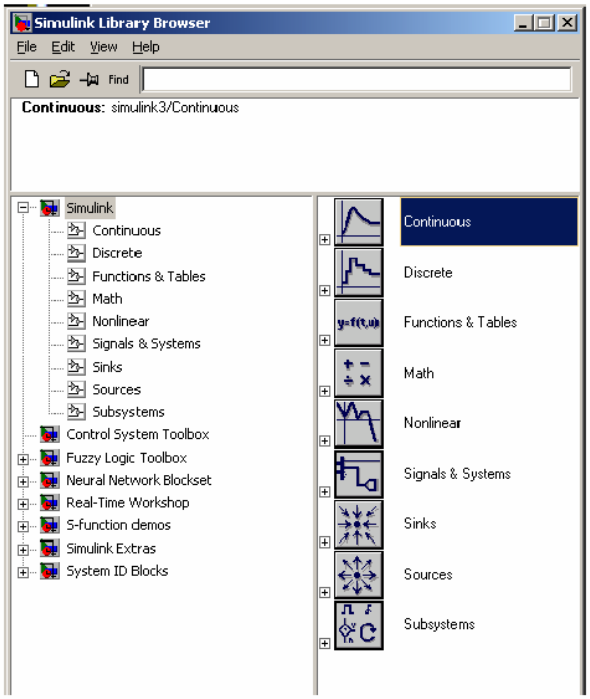
۸ – ۱ – ۱ مفهوم شبیه سازی سیستم های دینامیکی

شبیه سازی سیستم های دینامیکی هنگام طراحی سیستم های کنترل بسیار مفید می‌باشد. زیرا موجب صرفه جویی در زمان و هزینه های مربوط به نمونه سازی سیستم فیزیکی مورد نظر خواهد شد. سیمولینک یک نرم افزار بسیار قدرتمند می‌باشد که به یک نرم افزار مطلب ضمیمه شده و بصورت یکی از جعبه ابزارهای آن در آمده است. نرم افزار مطلب با توجه به قابلیت تحلیلهای عددی وسیعی که دارد، به نرم افزار قدرتمند در این زمینه بدل شده است. جعبه ابزار سیمولینک ابزاری جهت شبیه‌سازی سیستم های دینامیکی (سیستم هایی که شامل چندین معادلهٔ دیفرانسیل حاکم بر سیستم می باشند.) و نمایش نتایج به صورت گرافیکی می‌باشد. هر حلقهٔ منطقی، یا سیستم کنترلی، مربوط به سیستم های دینامیکی را می‌توان با استفاده از بلوکهای آمادهٔ استاندارد موجود در کتابخانهٔ سیمولینک، شبیه‌سازی نمود. در این نرم افزار، جعبه ابزارهای متعددی برای زمینه‌های مختلف وجود دارد، مثل جعبه ابزار منطق فازی، جعبه ابزار شبکه‌های عصبی، جعبه ابزار Dsp، جعبه ابزار آمار و . . . که موجب قدرتمند شدن توانایی پردازش این نرم افزار می‌شوند. مهمترین مزیت این نرم افزار داشتن بلوکهای آماده می‌باشد، که نیاز به نوشتن برنامه را برای تحلیل‌های ریاضیاتی کوتاه از میان می‌برد.

۸ – ۱ – ۲ مفهوم سیگنال و جریان منطقی

در سیمولینک، داده‌ها / اطلاعات از بلوکهای مختلف توسط خطوط رابط به بلوکهای مرتبط فرستاده می‌شود. سیگنالهای استاتیکی و دینامیکی را می‌توان تولید کرده و بعنوان ورودی بلوکها، به آنها انتقال داد. داده‌ها را می‌توان بعنوان ورودی توابع نیز به آنها انتقال داد. پس از انجام عملیات،

می‌توان داده‌های حاصل را به عنوان ورودی توابع sink که می تواند اسکوپها باشند، به آنها انتقال داد و یا این داده‌ها را در فایلی ذخیره نمود.

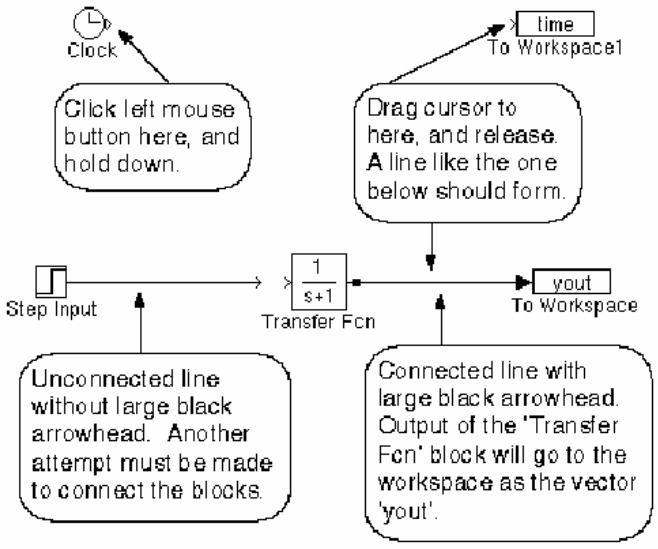


شکل ۸ – ۱) کتابخانهٔ سیمولینک

داده‌ها می‌توانند از یک بلوک به بلوک دیگر متصل و یا شاخه شاخه شوند. در شبیه سازی، داده‌ها پردازش شده و در فواصل زمانی غیر پیوستهٔ انتقال داده می‌شوند که این بدلیل ماهیت غیر پیوسته رایانه‌ها می‌باشد. بنابراین گام زمانی یک فرآیند شبیه سازی، مسئله ای بسیار اساسی و مهم می باشد و انتخاب آن توسط سریعترین دینامیک در سیستم شبیه سازی شده صورت می پذیرد. در بخشهای بعدی، بلوکهای مختلف که در نرم افزار سیمولینک قابل دسترسی هستند، تشریح خواهند شد. شکل ۸ – ۱ نمای کلی کتابخانه های در دسترس سیمولینک را نشان می‌دهد. آخرین نسخهٔ نرم افزار سیمولینک، همراه نرم افزار *MATLAB 6.5 (release 12.1)* ارائه می‌شود.

۸ -۱-۳ چگونگی اتصال بلوکها

جهت اتصال بلوکها، کلیک راست کرده و ماوس را از خروجی یک بلوک به ورودی بلوک دیگر بکشید. شکل ۲- ۸ چگونگی انجام این عملیات را بصورت مرحله به مرحله نشان می‌دهد. چگونگی شاخه دار کردن اتصالات در انتهای این فصل تشریح شده است.



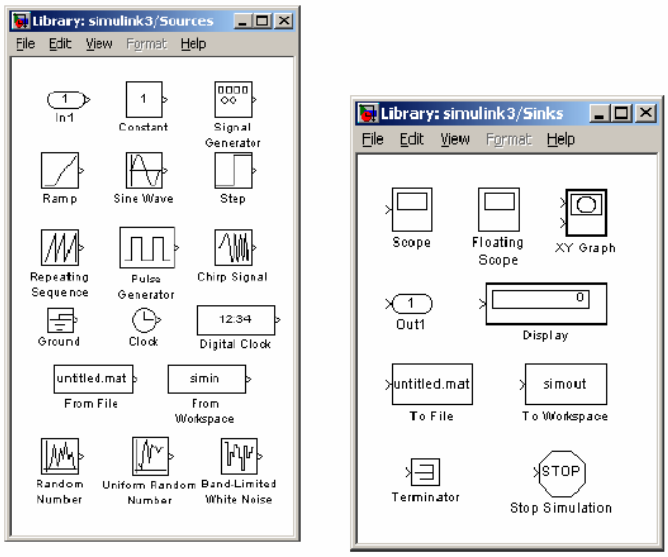
شکل ۸- ۲) چگونگی متصل کردن بلوکها

۸ - ۲ چگونگی دسترسی به کتابخانهٔ *Sources and Sinks*

کتابخانهٔ *sources* شامل منابعی از داده‌ها یا سیگنالها که کاربر جهت شبیه‌سازی سیستم‌های دینامیکی به آنها احتیاج خواهد داشت، می شود. ممکن است کاربر بخواهد تا از ورودی ثابت، موج سینوسی، تابع پله، تابع شیب و . . . استفاده نماید. همچنین ممکن است کاربر بخواهد تا تأثیرات اغتشاشات را روی سیستم بیازماید در این صورت می‌تواند از تابع تولید کنندهٔ سیگنال تصادفی جهت شبیه سازی اغتشاشات استفاده نماید. از آیکن *clock* می‌توان در مواردی که نیاز به زمان گذاری در ترسیمات باشد، استفاده نمود. همچنین جهت جلوگیری از پیغام‌های خطای نشان دهندهٔ پورتهای استفاده نشده، می توان آنها را به آیکن *ground* متصل نمود.

کتابخانهٔ *sinks* شامل بلوکهایی می شود که سیگنالها در آنها به پایان می رسند. در بیشتر موارد می خواهیم تا نتایج را در یک فایل مجزا و یا ماتریسی از متغیرها ذخیره کنیم. داده‌ها می‌توانند نمایش داده شوند و یا در فایلی ذخیره شوند. بلوک *stop* جهت توقف عملیات شبیه سازی در صورتیکه ورودی بلوک مخالف صفر باشد، استفاده می شود. شکل ۸-۳ بلوکهای قابل دسترس در کتابخانهٔ *Sources and sinks* را نشان می‌دهد.

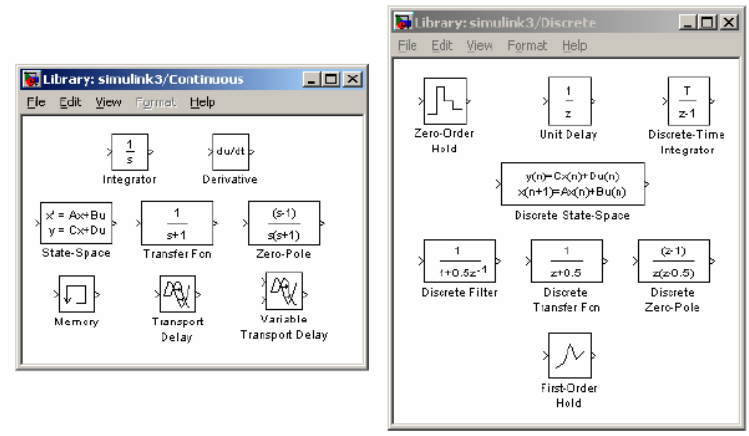
سیگنالهای استفاده نشده باید خاتمه داده شوند، تا از دادن هشدارهایی در مورد سیگنالهای قطع شده، جلوگیری کند.



شکل ۸ - ۳) کتابخانهٔ *Sources* و *Sink*

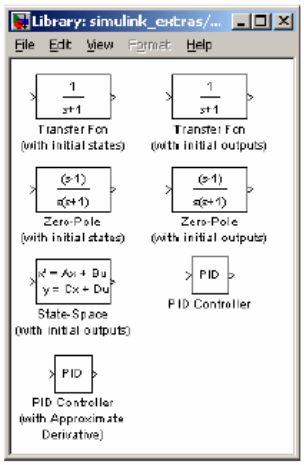
۸-۳ سیستم های پیوسته و گسسته

تمام سیستم های دینامیکی را می‌توان به عنوان سیستم هایی بازمان پیوسته و یا گسسته تحلیل نمود. نرم افزار سیولینک به کاربر اجازه می‌دهد تا این سیستم ها را توسط توابع انتقال، بلوکهای تکاملی، بلوکهای تأخیری و . . . نشان دهد.



شکل ۸ – ٤) سیستمهای گسسته و پیوسته

شکل ۸-٤ بلوکهای سیستم های دینامیکی در دسترس را نشان می‌دهد. سیستم های گسسته را می‌توان در صفحهٔ z که مبین معادلات تفاضل می‌باشد، طراحی نمود. سیستم ها می‌توانند به شکل حالت – فضا نشان داده شوند که این شیوهٔ نمایش در طراحی سیستم‌های کنترل مدرن بسیار سودمند خواهد بود.



شکل ۸– ۵) سیستمهای خطی پیشرفته

شکل ۸-۵ چندین بلوک خطی پیشرفته را که در کتابخانهٔ *Simulink Entras* موجود می‌باشند، نشان می‌دهد. این بلوکها شامل بلوکهای پیشرفتهٔ خاصی، مانند بلوک کنترلی *PID*، توابع انتقال دارای شرایط اولیه و... می‌شوند.

مثال ۸-۱ یک سیستم جرم ـ فنر ـ دمپر

بخش زیر شامل مثالی برای ایجاد یک سیستم جرم ـ فنر ـ دمپر می‌باشد. این سیستم را می‌توان به دو شیوه ساخت :

نمایش حالت ـ فضا، که در تئوری کنترل پیشرفته بکار می رود.

۱. نمایش تابع انتقال کلی. سیستم جرم ـ فنر ـ دمپریک سیستم درجهٔ دو می‌باشد که در اغلب سیستم های دینامیکی با آن مواجه می‌شویم. مدارهای الکتریکی، مقاومت ـ سلف ـ خازن *(RLC)* نیز مشابه این مثال خواهند بود و می توانند بعنوان سیستم‌های مرتبهٔ دوم مدلسازی شوند.

این مثال در شکل ۸-٦ نشان داده شده است. از یک تابع پلهٔ واحد جهت ورودی سیستم کنترلی استفاده شده است. (سیستم آورده شده در این مثال، حلقه باز می باشد). بخش بالایی بلوک شامل چگونگی نمایش تابع انتقال سیستم دینامیکی می‌باشد. در اینجا کاربر می‌تواند تنها خروجی ها را مشاهده کند و قادر به مشاهده و تحلیل حالتهای سیستم نخواهد بود. همچنین در این حالت کاربر قادر به اعمال شرایط اولیهٔ غیر صفر روی سیستم نخواهد بود. (البته با استفاده از بلوک تابع انتقال خاصی که در جعبه ابزار *Extras* موجود می‌باشد می‌توان شرایط اولیهٔ غیر صفر را نیز برای سیستم تعریف نمود.) بخش پایینی دیاگرام سیمولینک، همان سیستم درجه دوم را بصورت نمایش حالت ـ فضا نشان می‌دهد. بالاترین مرتبهٔ مشتق (در این مورد شتاب) بصورت تابعی از ورودیها و حالتهای دیگر نشان داده شده است. از این ورودی جهت تشکیل حالت با مرتبهٔ مشتق کمتر، انتگرال گیری می‌شود. شرایط اولیه برای هر حالت را می‌توان در بلوک انتگرال گیر تعیین نمود. حالتها می‌توانند بصورت کاملاً مجزا رویت شده و یا حتی اصلاح شوند.

یک سیستم جرم ـ فنر ـ دمپر با معادلهٔ دینامیکی زیر را در نظر بگیرید :

$$m\ddot{x} + c\dot{x} + kx = q_i u$$

که در آن x متغیر خروجی، m جرم، c ضریب میرایی، k سختی فنر و u نیروی کنترل شده (در اینجا در ثابت q_i ضرب شده است) می‌باشد. این معادله را می توان در حوزهٔ لاپلاس به صورت زیر نوشت:

$$a\frac{X(s)}{U(s)} = \frac{k\omega_n^{\,2}}{s^2 + 2\varsigma\omega_n s + \omega_n^{\,2}}$$

که در آن $\varsigma = \frac{c}{2\sqrt{km}}$ نسبت میرایی، $\omega_n = \sqrt{\frac{k}{m}}$ فرکانس طبیعی و $k = \frac{q_i}{m}$ دامنهٔ حالت پایدار می‌باشد. در شکل حالت ـ فضا، این سیستم برحسب بالاترین مرتبهٔ مشتقش نشان داده می‌شود ؛

$$m\ddot{x} = (q_i u - c\dot{x} - kx) \Rightarrow \ddot{x} = \frac{1}{m}(q_i u - c\dot{x} - kx)$$

معادلات فوق را می‌توان برحسب میرایی و فرکانس طبیعی سیستم و با فرض $q_i = 1$ به صورت ؛

$$\ddot{x} = \frac{u}{m} - 2\varsigma\omega_n\dot{x} - \omega_n{}^2 x$$

نوشت. در این مثال، با شرایط اولیهٔ صفر، روش تابع انتقال و نمایش حالت ـ فضا منجر به نتایج یکسانی خواهند شد. در حالت کلی استفادهٔ همزمان از دو روش ذکر شده ضرورتی ندارد. مراحل بدست آوردن معادلات حالت ـ فضا در زیر آمده است:

معادلهٔ دیفرانسیل را برای بدست آوردن بالاترین مرتبهٔ مشتق حل می کنیم. اگر معادله نرمالیزه نشده است، بالاترین مرتبهٔ مشتق ممکن است شامل عبارتی به عنوان ضریبش باشد. در اینصورت تمامی مقادیر معادله را بر این عبارت تقسیم خواهیم کرد. اکنون بالاترین مرتبهٔ مشتق با ضریب یک را سمت چپ تساوی و بقیهٔ عبارات را در سمت راست تساوی قرار می دهیم.

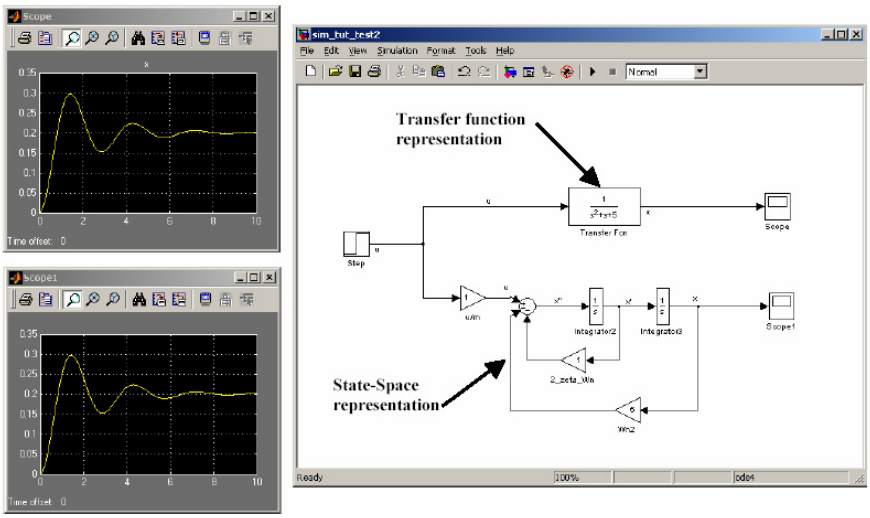
۱. یک بلوک مقایسه کننده در پنجره ایجاد کنید. بلوک باید به تعداد عبارتهای مثبت و منفی سمت راست علامت تساوی، به ترتیب دارای ورودیهای مثبت و منفی باشد (در اینجا در سمت راست علامت تساوی سه عبارت وجود دارد که دو تای آنها منفی می باشد، بنابراین به بلوک مقایسه کننده. دو ورودی منفی و یک ورودی مثبت می‌افزاییم.) خروجی بلوک جمع کننده باید مساوی عبارت شامل بالاترین مرتبهٔ مشتق که در یک ثابت ضرب شده است، باشد. حال می‌توان معادله را در یک ضریب، ضرب و یا تقسیم نمود، تا ضریب بالاترین مرتبهٔ مشتق معادله یک شود.

۲. در این مرحله انتگرالگیرها را می‌افزاییم. تعداد کل انتگرالگیرها باید مساوی مرتبهٔ معادلهٔ دیفرانسیل باشد. برای مثال، اگر یک سیستم مکانیکی با معادلهٔ دیفرانسیل مرتبهٔ دو، حاکم بر آن داشته باشیم، و هدف بیان موقعیت سیستم باشد، نیاز به دو مرتبه انتگرال گیری وجود خواهد داشت. باید در انتها، بلوکی را نیز برای متغیر خروجی در نظر بگیریم.

۳. پس از هر انتگرال گیری، سیگنال را به مکان مناسبش در بلوک مقایسه کننده فیدبک می‌کنیم. در سمت راست هر انتگرال مقداری برابر با انتگرال مقدار سمت چپ آن وجود

دارد. قبل از فیدبک کردن مقدار خروجی هر انتگرال گیر به مکان مناسبش در بلوک مقایسه کننده، (باید) از بلوک اندازه جهت ضرب کردن خروجی در مقدار ضربش استفاده شود.

توجه کنید که در نمایش حالت – فضای سیستمها دسترسی به مشتقات مرتبهٔ پایین تر (حالتهای سیستم) ممکن خواهد بود. این قابلیت، شیوهٔ نمایش حالت – فضا را بصورت روشی بهتر برای توصیف سیستمهای دینامیکی معرفی می کند. بعلاوه، در این شیوه سازگاری سیستم با اجزای غیر خطی، ساده تر خواهد بود. روش ماتریس انتقال روشی ساده تر نسبت به نمایش حالت فضا می‌باشد (زیرا در آن تنها از یک بلوک استفاده می شود)، اما این روش، دارای محدودیتهای بسیاری می‌باشد.

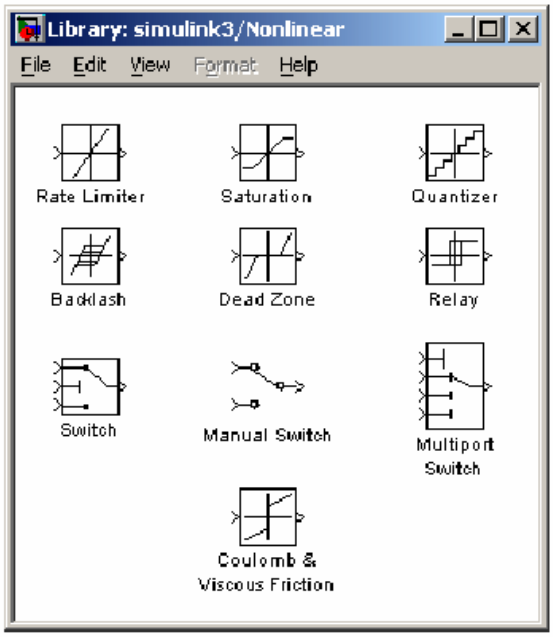


شکل ۸ – ٦) سیستم جرم – فنر – دمپر (مثالی از یک سیستم دینامیکی مرتبهٔ دو)

۸ – ۴ عملگرهای غیر خطی

یک مزیت عمدهٔ استفاده از نرم افزارهایی چون سیمولینک، قابلیت شبیه‌سازی سیستم های غیر خطی و رسیدن به نتایج صحیح بدون انجام حل های تحلیلی می‌باشد. رسیدن به حلهای تحلیلی برای سیستم های دارای موارد غیر خطی مثل تابع *saturation*، تابع *signum*، تابع *limited slew rate* بسیار مشکل می‌باشد. در هنگام شبیه سازی، از آنجا که سیستم ها با استفاده از تکرارها، تحلیل می‌شوند، عوامل غیر خطی مزاحمتی ایجاد نخواهند کرد. شکل ۸–۷ عناصر غیر خطی که می‌توانند در تحلیل سیستم های غیر خطی بکار روند را نشان می‌دهد. یکی از عناصر، بلوک *saturation* می

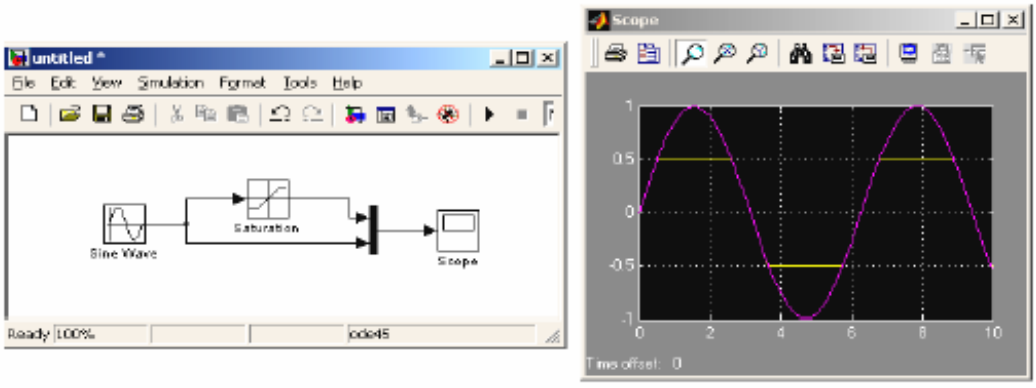
باشد. که نشان دهندهٔ محدودیتی فیزیکی روی پارامترهای خاص می‌باشد. مثل سیگنال ولتاژ در یک موتور الکتریکی و سوییچ ها هنگامیکه شبیه سازی باید برای موارد متعددی انجام گیرد، مفید خواهند بود. سوییچ ها معادلهای منطقی عبارات *IF-THEN* در برنامه نویسی می‌باشند. تابع *slew rates* با استفاده از محدود کنندهٔ نرخی می تواند نرخ تغییرات یک پارامتر فیزیکی، مثل سرعت یک موتور *Dc* را کنترل نماید.



شکل ۸ – ۷) بلوکهای غیر خطی

مثال ۸ – ۲ سیگنالهای اشباع شده و اشباع نشده

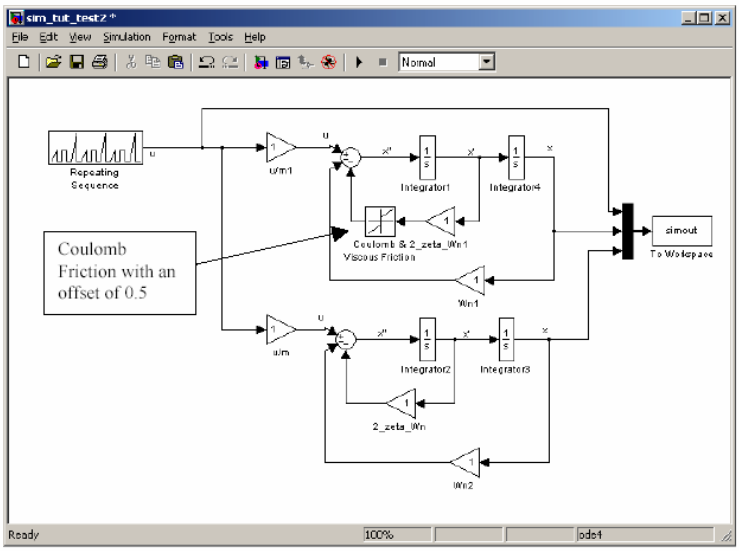
در اینجا مثالی که از یک بلوک غیر خطی استفاده می‌کند، آورده شده است. یک موج سینوسی با دامنهٔ 1 (سیگنال بین $1+$ و $1-$ تغییر می کند) را در نظر بگیرید. از یک بلوک *saturation* جهت محدود کردن دامنهٔ سیگنال خروجی به مقدار 0.5 استفاده می شود. سپس تفاوت سیگنالهای اشباع شده و اشباع نشده نشان داده شده است. نتایج در شکل ۸–۸ نشان داده شده است. سیگنالهای اشباع شده و اشباع نشده به وضوح قابل رویت می باشند.



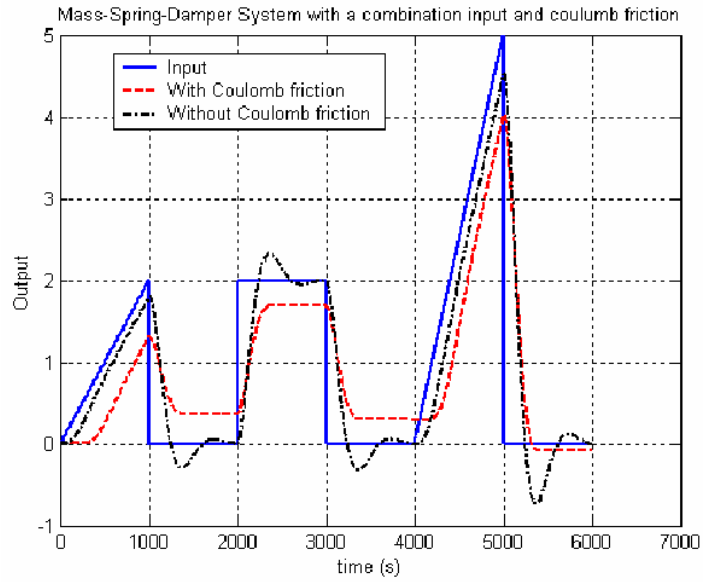
شکل ۸ – ۸) مثالی از یک تابع غیر خطی (اشباع)

مثال ۸ – ۳ یک سیستم جرم – فنر – دمپر

یک سیستم جرم – فنر – دمپر همراه با اصطکاک کولمب، جهت ایجاد نیروی میرا کننده، تشکیل می‌دهیم. اصطکاک کولمب (موجود در کتابخانهٔ توابع غیر خطی) بعنوان آفستی در سرعت صفر نشان داده شده است. در این مسئله مقدار آفست 0.5 (با شیب 1) می‌باشد. چگونگی انجام مراحل در شکل ۸ –۹ نشان داده شده است. خروجی به ازای ترکیبی از ورودیها در شکل ۸ –۱۰ نشان داده شده است. ترکیب ورودیها از طریق بلوک *repeating sequence* در کتابخانهٔ *sources* امکان پذیر خواهد بود. همانطور که انتظار می‌رود، اصطکاک کولمب موجب ایجاد جوابهای نامطلوبی در خروجی سیستم می‌شود.



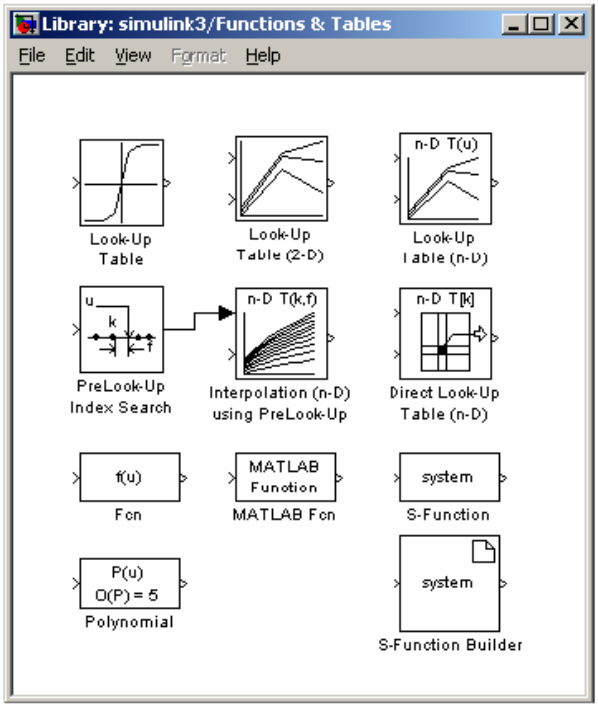
شکل ۸ – ۹) مثالی از یک سیستم جرم – فنر – دمپر به همراه اصطکاک کولمب



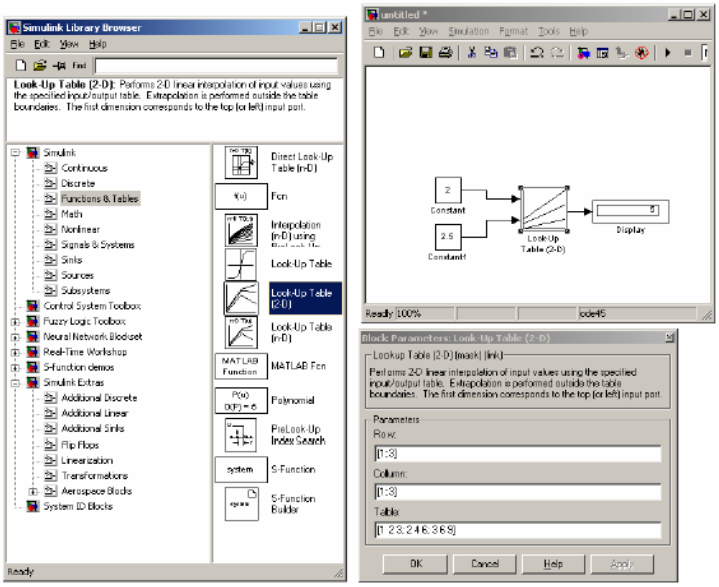
شکل ۸ – ۱۰) خروجی یک سیستم جرم – فنر – دمپر به همراه اصطکاک کولمب

۸ – ۵ نحوهٔ استفاده از توابع (نوشته شده به صورت *C*، *M* و...)

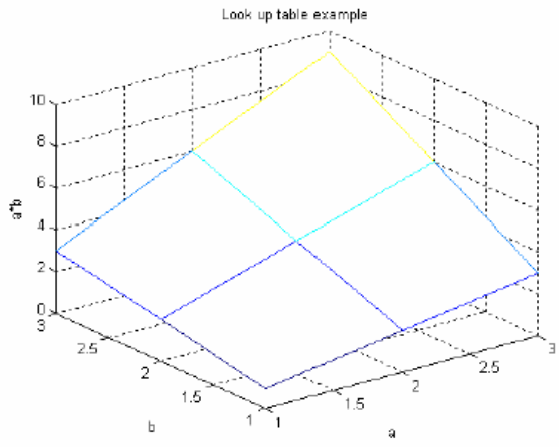
توابعی که در *M* یا در هر زبان برنامه نویسی دیگری مثل *C* یا فرترن نوشته می‌شوند، می‌توانند همراه سیمولینک نیز بکار روند که اینکار موجب افزایش توانایی نرم افزار سیمولینک خواهد شد. برنامه ها، در صورت استفاده، بصورت *'M'* فایل و یا *'C'* فایل در هر مرحله از شبیه سازی مورد ارزیابی واقع می‌شود. *S* توابع، کتابخانه‌های مرتبط شدهٔ دینامیکی *(DLL)* می‌باشند که در زبان برنامه نویسی دیگری مثل *C* توسط کامپایلر مطلب *'MEX'* کامپایل می‌شوند. این مطلب در هنگام شبیه سازی های بزرگ، بسیار مفید می‌باشد، زیرا تابعی که در زبان *'C'* نوشته شود خیلی سریعتر از یک برنامه بصورت *M* تابع، اجرا می‌شود. همچنین برای شبیه سازیهای *REAL-TIME* تنها *S* تابع ها می‌توانند مورد استفاده قرار گیرند، که دلیل آن مجدداً سرعت زیاد عمل پردازش خواهد بود.



شکل ۸ – ۱۱) توابع و جداول



شکل ۸ – ۱۲) مثال جدول *look up* دو بعدی



شکل ۸ – ۱۳) ترسیم جدول *look up* دو بعدی

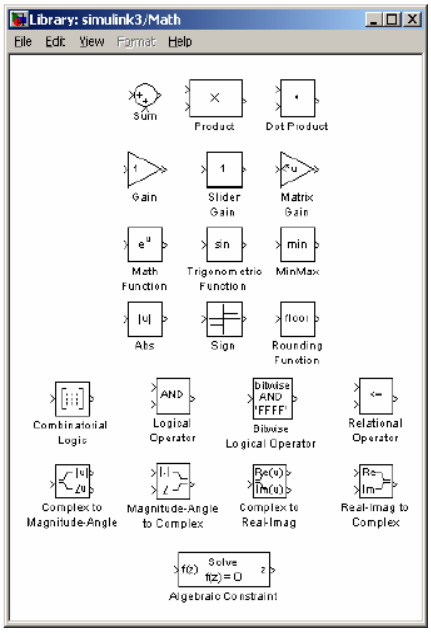
جداول ابزارهای بسیار مهمی در نگاشتن نقاط و توابع گوناگون می‌باشند. شکل ۱۱ توابع گوناگون و جداول استفاده شده در سیمولینک را نشان می‌دهد.

مثال ۸ – ۴ میانیابی خطی با استفاده از جداول

از جداول جهت ایجاد خروجی‌ها بر اساس الگوی ورودیها استفاده می‌شود. اگر الگو شناخته شود، در اینصورت داده‌ها را می توان در یک جدول وارد نمود، و می‌توان از میانیابی خطی جهت ایجاد خروجی ها براساس مجموعهٔ جدید ورودیها استفاده نمود. فرض کنید می‌خواهیم دو ورودی را در یکدیگر ضرب کرده و حاصل را به عنوان خروجی نمایش دهیم. یک جدول دو بعدی در سیمولینک ایجاد شده و مقادیر متناظر با *1*، *2* و *3* بعنوان ورودی ها در بلوک خروجی وارد می‌شود، که این روند در شکل ۱۲ نشان داده شده است. از این بلوک جهت ضرب دو ورودی در یکدیگر استفاده می‌شود و خروجی به صورت زیر نشان داده می‌شود:

$$2 * 2.5 = 5$$

نمایش دو بعدی جدول در شکل ۸ –۱۳ آورده شده است. هر سطح دو بعدی را می‌توان بعنوان جدول در نظر گرفت، البته به شرطی که داده‌های مربوط به نقاط در ورودیها موجود باشد. نمایشهای دو بعدی تا n بعدی جداول درسیمولینک موجود می‌باشند.



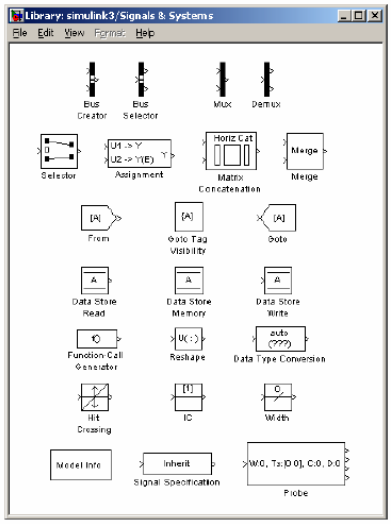
شکل ۸ – ۱٤) ابزارهای ریاضیاتی

۸ – ۶ عملیات ریاضیاتی

عملگرهای ریاضیاتی مثل ضرب، جمع، عملگرهای منطقی مثل *AND*، *OR* و . . . می توانند با استفاده از جریان سیگنال در سیمولینک برنامه نویسی شوند. ضرب ماتریسی با استفاده از ماتریس بلوک اندازه بسیار ساده خواهد شد. توابع مثلثاتی مثل *sin* یا *atan* به سادگی قابل دسترس خواهند بود. عملگرهای رابطه‌ای مثل علامت تساوی، بزرگتر از و... را نیز می‌توان در مدارهای منطقی بکار برد. شکل ۸–۱٤ ابزارهای ریاضیاتی در دسترس را در سیمولینک نشان می‌دهد.

۸ – ۷ انتقال داده ها و سیگنالها

در دیاگرامهای بلوگی پیچیده، ممکن است نیاز به انتقال داده ها از بخشی از دیاگرام بلوک به بخش دیگر باشد. ممکن است این بخش‌ها در زیر سیستم های متفاوتی واقع باشند. در اینصورت می‌توان سیگنال را به عنوان ورودی یک بلوک *GOTO*، که جهت انتقال سیگنالها از یک زیر سیستم به زیر سیستم دیگر استفاده می‌شود، قرار داد. تقسیم مدل به چند زیر سیستم موجب حذف پیچیدگیهای مربوط به اتصال دهنده‌های اضافی شده و نمایش ماتریسی را ساده تر می‌سازد.



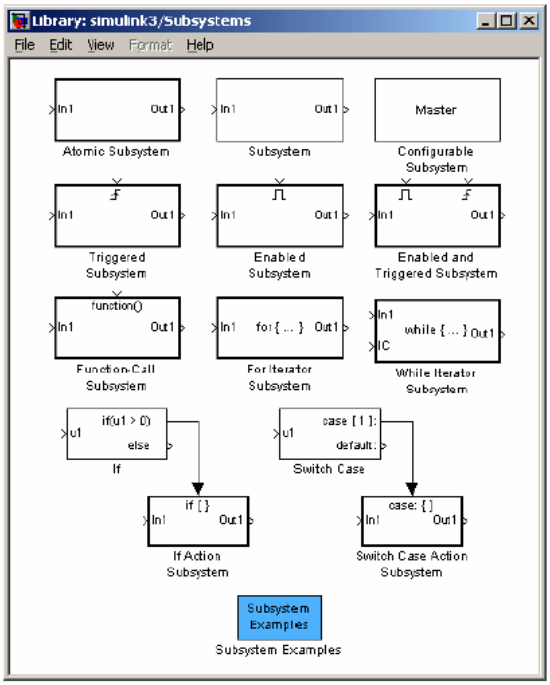
شکل ۸ – ۱۵) انتقال داده ها و سیگنالها

۸ – ۸ بهینه کردن ظاهر بصری

در موارد متعددی، هنگامیکه یک دیاگرام سیمولینک پیچیده ایجاد می‌شود، تعداد اتصال دهنده‌ها و بلوکها در یک سطح خاص، ممکن است مانع از درک درستی از جریان منطقی برنامه شود. در اینگونه موارد کاربر می‌تواند از روند استفاده از زیر سیستم‌ها جهت ساده‌تر و قابل فهم‌تر شدن دیاگرام بلوکی استفاده کند.

۸ – ۸ – ۱ استفاده از زیر سیستم ها و ماسکها

ماسکها رابط های میان عملکرد زیر سیستم‌ها و کاربر می‌باشند. برای مثال، اگر برنامه‌نویس بخواهد که الگوریتم برنامه را از دید کاربران پنهان سازد، در این صورت می‌تواند از یک ماسک استفاده کرده و الگوریتم را پس از قرار دادن در یک زیر سیستم پنهان سازد.

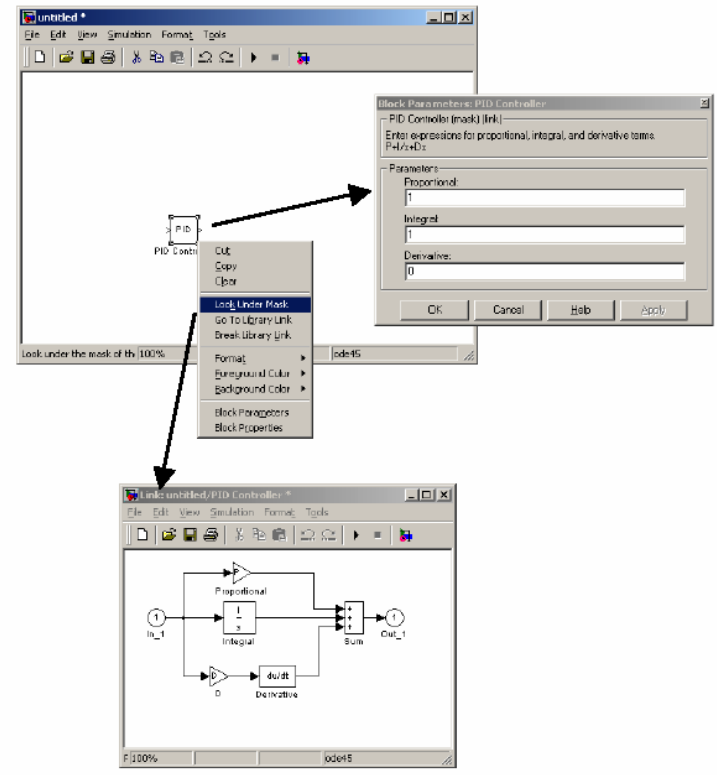


شکل ۸ – ۱۶) زیر سیستمها

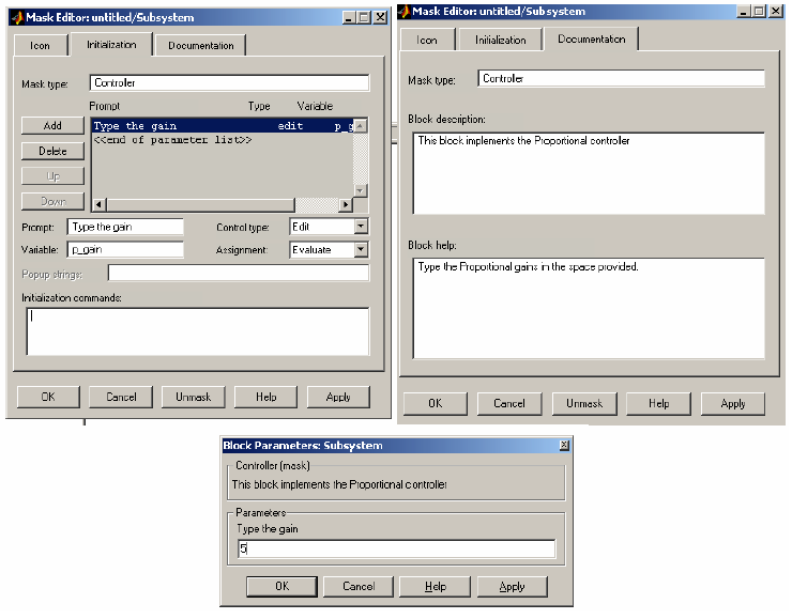
مثال ۸ – ۵ بلوک کنترلی *PID* در سیمولنیک

کتابخانهٔ *Extras* در سیمولنیک، شامل بلوک مربوط به کنترل *PID* می‌باشد. هنگامیکه روی این بلوک دو مرتبه کلیک شود، از کاربر اندازه‌های قسمتهای *P*، *I* و *D* کنترلی را خواهد خواست. اجزای داخل سیستم (که می‌توان از طریق کلیک راست روی بلوک و سپس کلیک روی گزینهٔ *'Look under mask'* به آنها دست یافت) در شکل ۸ –۱۷ نشان داده شده است.

شکل ۸–۱۸ اجزای داخلی ماسک را نشان می‌دهد. فضاهایی جهت تایپ پیغام‌های کمکی، طراحی اشکال برای نمای ظاهری بلوکها، پذیرفتن متغیرها و ایجاد نشانگرها،... در محیط ویرایشگر ماسک در نظر گرفته شده است.



شکل ۸ – ۱۷) مثالی از ایجاد ماسک (بلوک کنترلر *PID*)

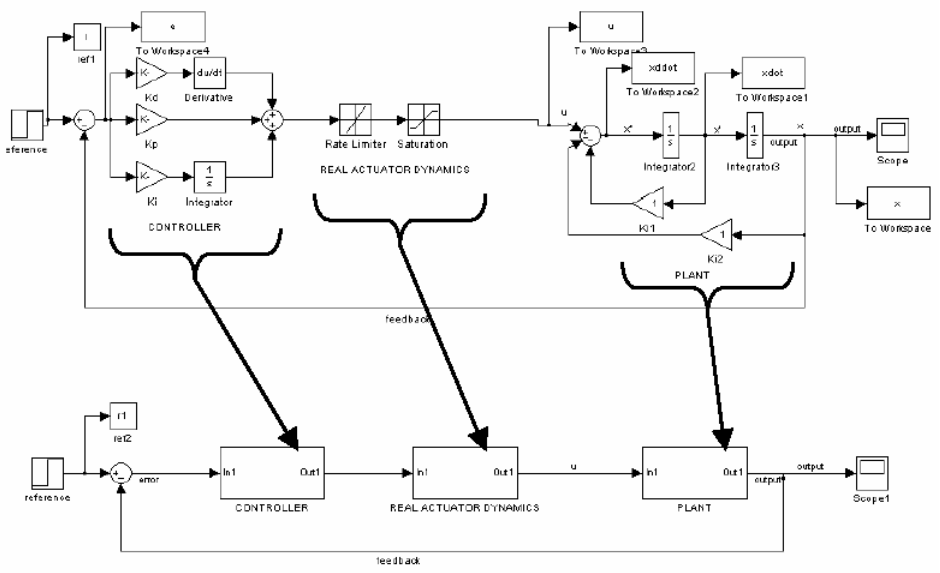


شکل ۸ – ۱۸) چگونگی برنامه نویسی ماسک

مثال ۸ – ۶ چگونگی ساده تر کردن نمودار بلوکی

در مواقعی که با نمودارهای بلوکی پیچیده، سر و کار داریم، قرار دادن قسمتهای پیچیده در بلوکهای مجزا، موجب قابل فهمتر شدن نمودار بلوکی می‌شود. در این گونه موارد، بلوکها را می‌توان از پنجرهٔ اصلی داخل زیر سیستمهایی قرار داد و این زیر سیستمها روند کلی برنامه را تشکیل خواهند داد.

شکل ۸–۱۹ مثالی از یک سیستم دینامیکی متشکل از یک کنترلر و تحریک کنندهٔ دینامیکی را نشان می‌دهد. سه قسمت عمدهٔ نمودار بلوکی که موجب پیچیدگی آن نیز شده‌اند در زیر سیستمهای مربوطه قرار گرفته‌اند. این کار موجب قابل فهم تر شدن نمودار بلوکی شده است.

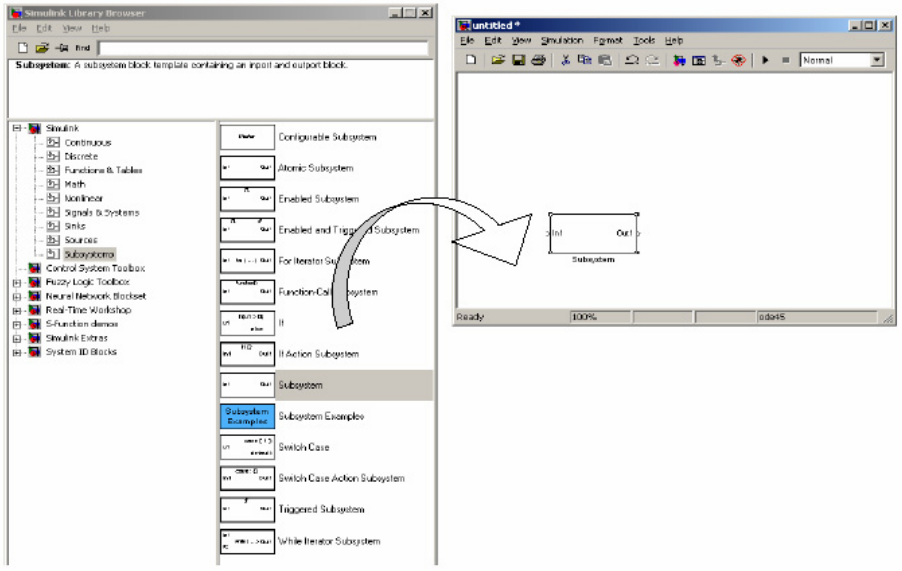


شکل ۸ – ۱۹) ساده سازی سیستمها با استفاده از زیر سیستمها

۸ – ۸ – ۲ ایجاد نمودن زیر سیستم ها

روند زیر بیان کنندهٔ چگونگی ایجاد زیر سیستم ها بعنوان بلوکهای مجزا می‌باشد، که در شکل ۸–۱۹ نیز نشان داده شد.

۱. از کتابخانهٔ سیمولینک یک زیر سیستم انتخاب کرده و آنرا به محیط بلوک فعلی یعنی جاییکه که میواهید از دید کاربر مخفی، باشد کشیده و رها کنید. نوع زیر سیستم بستگی به هدف بلوک خواهد داشت در هنگام انتخاب زیر سیستم ها می‌توان از زیر سیستم‌های استاندارد و یا دیگر زیر سیستم‌ها استفاده نمود. برای مثال، زیر سیستم می‌تواند یک بلوک آغاز گر باشد که در اینصورت تنها یک سیگنال آغاز گر نیز دریافت می‌شود. شکل ۸–۲۰ روند ایجاد زیر سیستم ها را نشان می‌دهد.



شکل ۸ – ۲۰) چگونگی ایجاد یک زیر سیستم

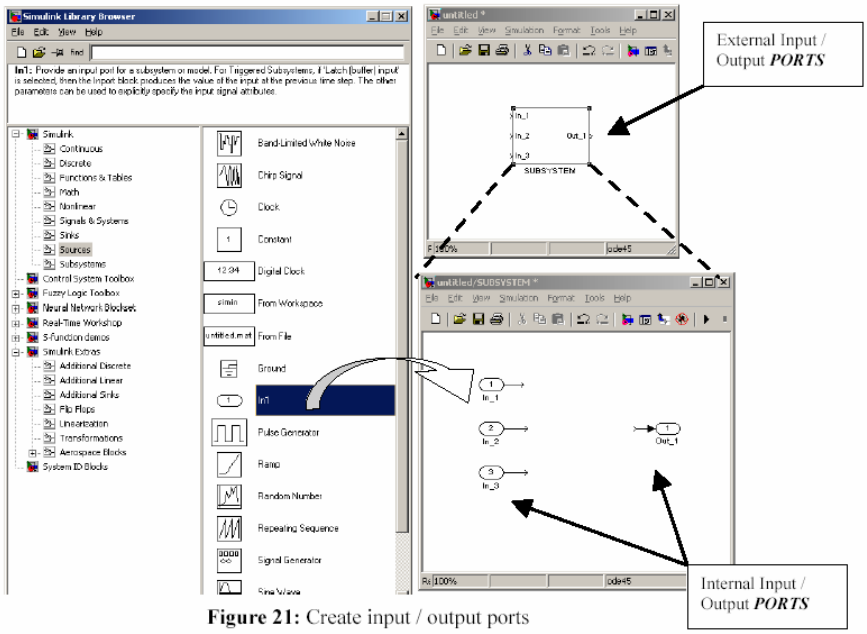
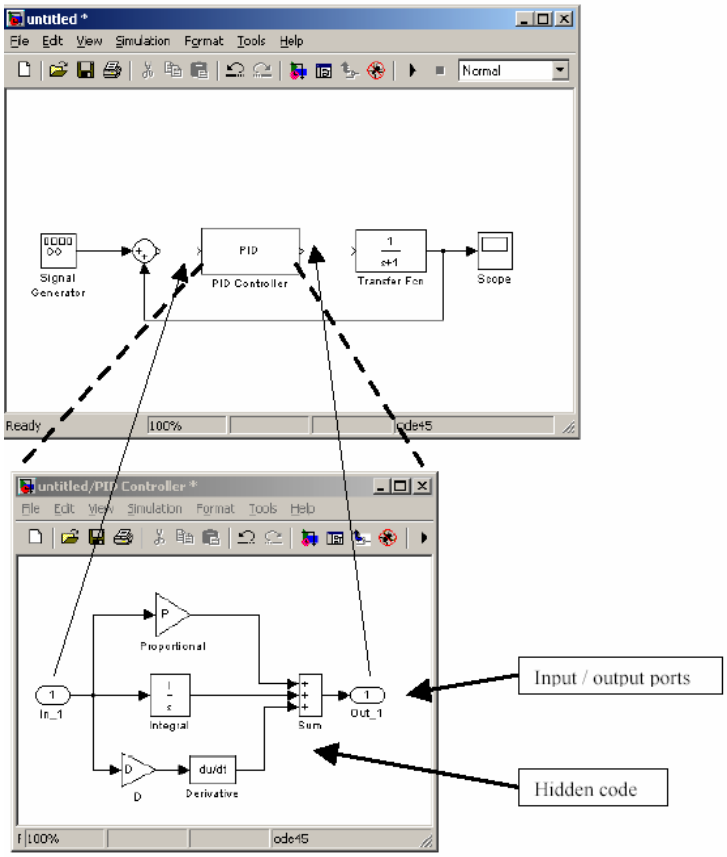


Figure 21: Create input / output ports

شکل ۸ – ۲۱) چگونگی ایجاد پورتهای ورودی/ خروجی

۲. زیر سیستم را با دوبار کلیک کردن روی آن بازکرده و پورتهای ورودی و خروجی زیر سیستم را که سیگنال‌ها را به زیر سیستم وارد و از آن خارج می‌کنند، ایجاد نمایید. پورتهای ورودی و خروجی به ترتیب از کتابخانه‌های *sources*،*sinks* قابل دسترسی خواهد بود. هنگامیکه پورتها در زیر سیستم ایجاد شد، این پورتها به طور خودکار در نمای بلوک اصلی ایجاد خواهد شد. در این صورت اتصال صحیح سیگنالها از بلوک اصلی به زیر سیستم‌ها میسر خواهد بود. شکل ۸ -۲۱ چگونگی ایجاد پورتهای ورودی و خروجی را نشان می‌دهد.



شکل ۸ - ۲۲ ایجاد کدهای مخفی

۳. پس از ایجاد زیر سیستم در ابتدا و انتهای سیستم، پورتهای ورودی و خروجی قرار خواهد گرفت. این مطلب در قسمت پایین شکل ۸ -۲۲ نشان داده شده است. بلوکهای زیر سیستم شامل دیاگرامی می‌باشند که در دیاگرام بلوکی اصلی دیده نمی‌شود. و ارتباط بین

زیر سیستم‌ها با دیاگرام بلوکی اصلی از طریق پورتهای ورودی و خروجی میسر می‌شود. شکل ۸ -۲۲ نشان می‌دهد که چگونه زیر سیستم ها از طریق پورتهای ورودی و خروجی با دیاگرام بلوکی اصلی مرتبط می‌شوند این زیر سیستم ها پس از ایجاد می‌توانند به صورت ماسک در آورده شوند.

۸ – ۸ – ۳ کمکهای بصری

از کمکهای بصری زیر می‌توان جهت ایجاد اطلاعات بیشتر در مورد بلوکهای شبیه‌سازی شده، بهره برد.

- **رنگهای زمان نمونه برداری**

بر اساس نرخ نمونه برداری از سیستم و اجزای اختصاصی آن، رنگها به طور خودکار به سیستم های دارای زمان نمونه برداری متفاوت نسبت داده می شوند.

- **نوع سیگنال**

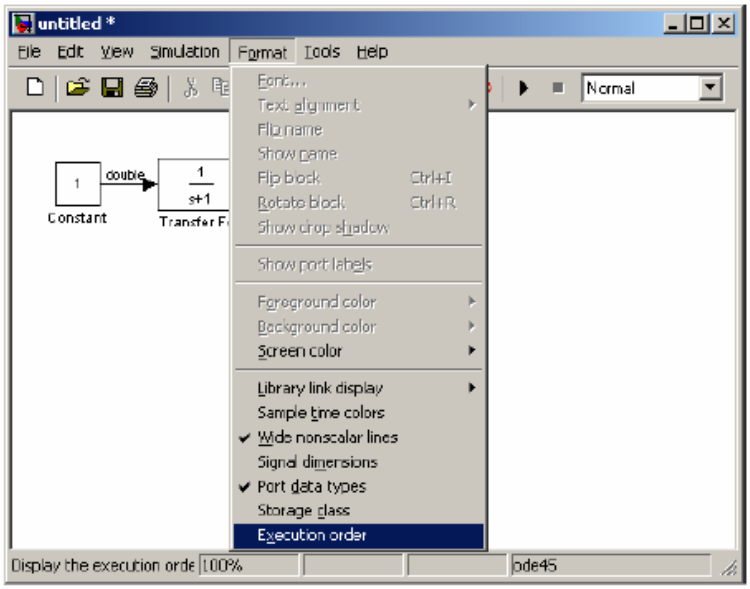
بر اساس نوع سیگنال، که می‌تواند به صورت Boo Lean,double و..... ، سیگنالها می‌توانند نشانه گذاری شوند که این امر ما را در تشخیص اینکه هر سیگنال نشاندهندهٔ چه چیزی می‌باشد، یاری خواهد کرد.

- **ضخامت خطوط**

ضخامت خطوط می‌تواند بر اساس این مطلب که آنها چه چیزی (بردار یا اسکالر) را انتقال می‌دهند، متغیر باشند. خطوط با ضخامت بیشتر مبین انتقال بردارها می‌باشند. ضخامت حقیقی خطوط را می توان در کنار هر خط نمایش داد.

- **ترتیب اجرا**

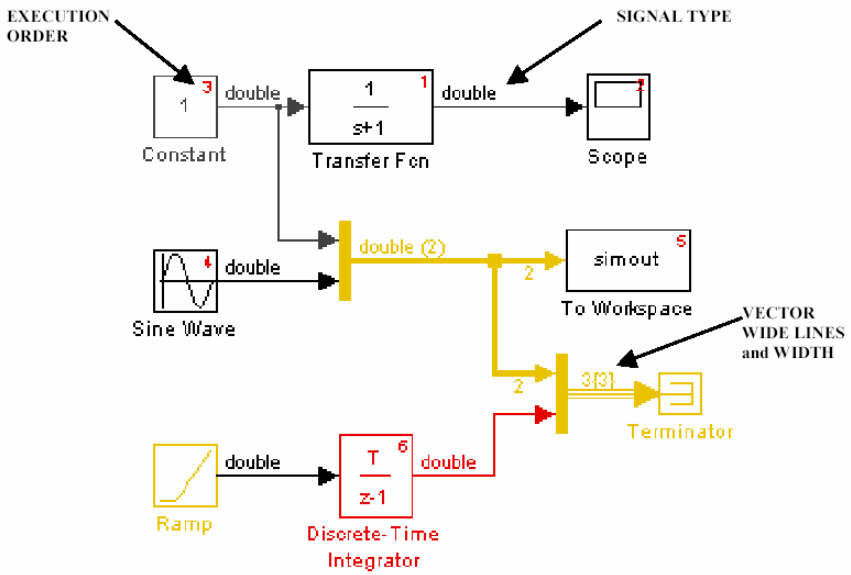
در برخی موارد دانستن اجراء بلوکها در سیمولینک مفید خواهد بود. این فرمان عددی را کنار هر بلوک قرار می‌دهد که نشاندهندهٔ ترتیب بلوک در اجرای برنامه می‌باشد. فرمانهای بیان شده در شکل ۸-۲۳ آورده شده‌اند. یک نمونهٔ عملی جهت درک بهتر این فرامین در شکل ۸-۲٤ نشان داده شده است.



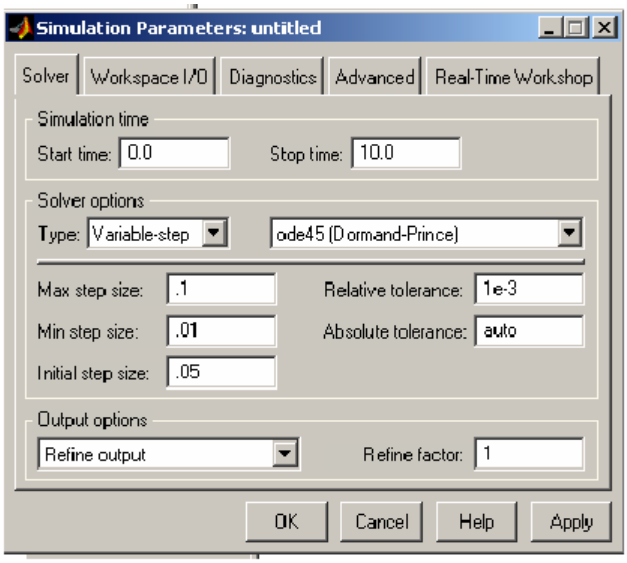
شکل ۸-۲۳) چگونگی تنظیم شیوهٔ نمایش بلوکها

۸ - ۹ تعیین نمودن پارامترهای شبیه سازی

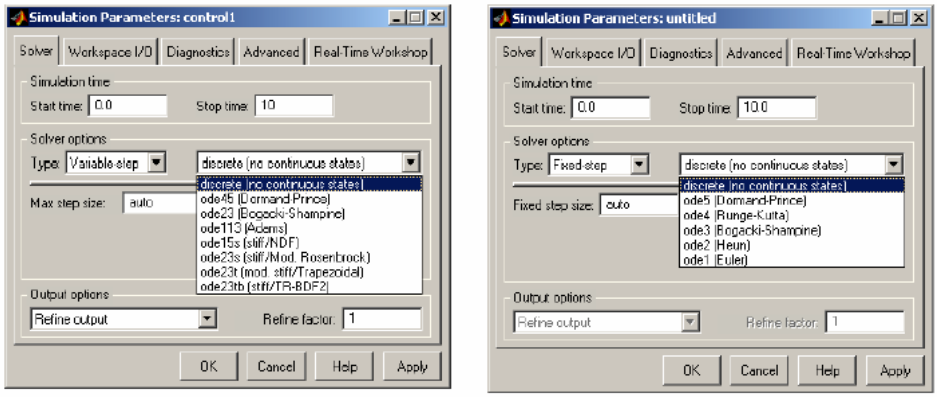
اجرای یک فرایند شبیه سازی در کامپیوتر همواره نیازمند بکار بردن یک شیوهٔ عددی جهت حل معادلهٔ دیفرانسیل مربوطه می‌باشد. سیستم را می‌توان به صورت یک سیستم گسسته و یا پیوسته در نظر گرفت که این مطلب بستگی به محتویات درون بلوکها نیز خواهد داشت. زمان شروع و پایان شبیه سازی می‌تواند تعیین شود. در هنگام تعیین اندازهٔ گام، مقدار کمترین و بیشترین اندازهٔ گام می‌تواند انتخاب شود. البته انتخاب اندازهٔ گام ثابت همواره توصیه می‌شود، زیرا موجب تعیین دقیق تعداد نقاط و در نتیجه کنترل کردن اندازهٔ بردار داده‌ا خواهد شد. انتخاب اندازهٔ گام شبیه سازی باید بر اساس دینامیک سیستم صورت پذیرد. در یک فرایند انتقال حرارتی گام زمانی در حد چند ثانیه می‌باشد، در حالیکه در یک موتور DC گام زمانی در حد چند میلی ثانیه انتخاب می‌شود.



شکل ۸ – ۲٤) مثالی از گزینه های نمایش بلوک



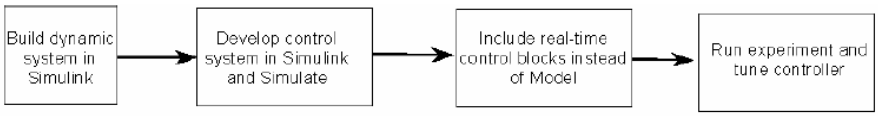
شکل ۸ – ۲۵) تنظیمات شبیه سازی



شکل ۸ – ۲۶) روشهای عددی موجود برای حل معادلات دینامیکی

۸ – ۱۰ مفهوم سخت افزار در حلقه

قسمت *REAL TIME WORK SHOP(RTW)* در نرم افزار سیمولینک امکان اتصال آنرا به هر سخت افزاری میسر می‌سازد، بنابراین امکان کنترل پذیری سیستم توسط یک زبان برنامه‌نویسی سطح بالا نظیر مطلب / سیمولینک، فراهم خواهد شد. این قابلیت که به آن قابلیت سخت افزار در حلقهٔ *(HIL)* گفته می‌شود به طور گسترده ای در کنترل سیستم بکار می رود. مفهوم کنترل *HIL* در شکل ۸ –۲۷ توضیح داده شده است. مثالی در شکل ۸ – ۲۸ در صفحهٔ بعد نشان داده شده است. یک مدل درجهٔ اول توسط سیستم‌های تغذیهٔ اطلاعاتی *DAC* و *ADC* با سخت افزار حقیقی تبادل اطلاعاتی انجام می‌دهند. سیگنال *DAC* به یک تحریک کننده فرستاده می‌شود و سیگنال *ADC* از یک سنسور به دست می‌آید.

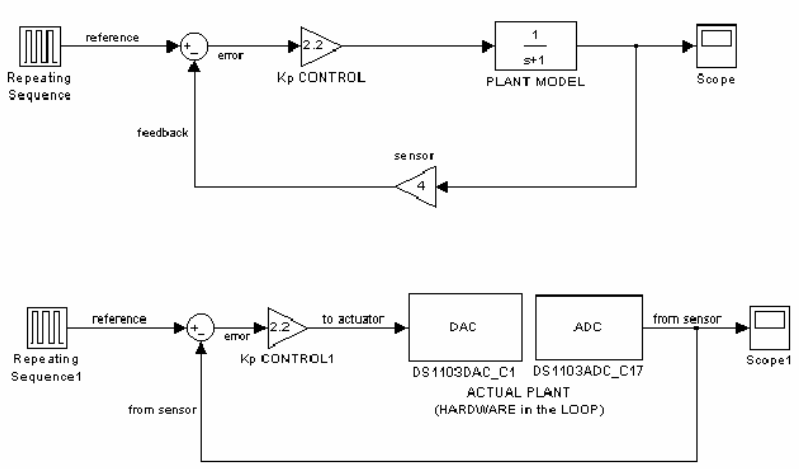


شکل ۸ – ۲۷) مفهوم سخت افزار در حلقه

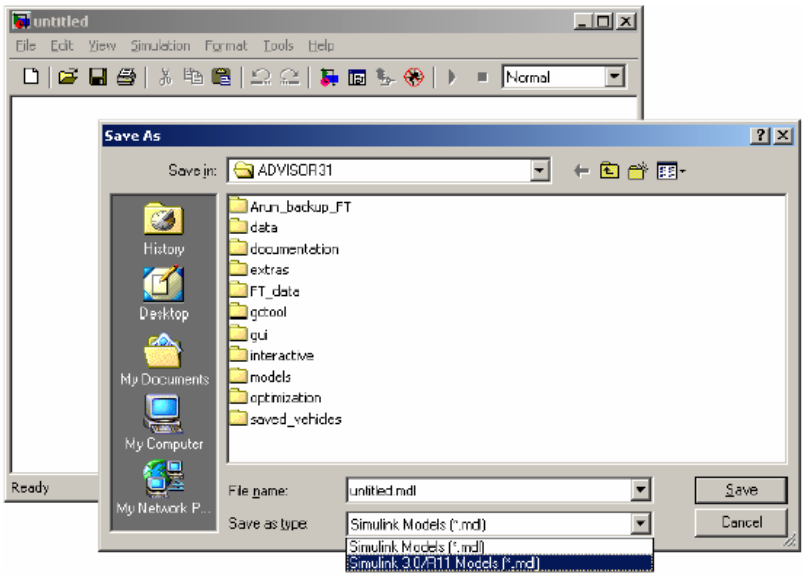
۸ – ۱۱ چند راهنمایی و ترفند

در اینجا چند راهنمایی جهت استفادهٔ هر چه بهتر از سیمولینک آورده شده است.

۱. جهت کپی کردن یک بلوک، روی آن کلیلک راست کرده و بلوک را به پنجرهٔ سیمولینک کشیده و رها کنید



شکل ۸ – ۲۸) مثالی از سخت افزار در حلقه



شکل ۸ – ۲۹) ایجاد سازگاری با نسخه های پیشین سیمولینک

۲. **کتابخانه‌ها** : جهت ایجاد یک بلوک نمونه (جهت استفاده های بعدی)، بلوک مورد نظر را با عنوان کتابخانهٔ بلوک ها ذخیره سازید. بعدها، می توانید بلوک مورد نظر را از کتابخانه به هر بلوک سیمولینک مورد نظر کپی کنید. تغییر دادن ساختار بلوک و یا پارامترهای آن در کتابخانهٔ موجب اعمال این تغییرات روی تمامی بلوکهایی که ممکن است استفاده شوند، خواهند شد. اگر بخواهید ارتباط میان بلوک خاصی را با کتابخانه‌اش قطع کنید، می‌توانید روی بلوک کلیلک راست کرده و گزینهٔ Break library link را انتخاب نمایید.

۳. جهت ایجاد یک شاخه از سیگنال، روی سیگنال مرجع در نقطه‌ای که می‌خواهید شاخه ایجاد شود کلیک راست کرده و آنرا به مکان دلخواه بکشید.

٤. همواره تمامی پورتهای باز را جهت جلوگیری از پیغام خطای مربوط به پورتهای مرتبط شده، به یک دیاگرام بلوکی متصل کنید. بلوک Ground در کتابخانهٔ sources و بلوک terminator در کتابخانهٔ sinks می‌توانند جهت اتصال و مهار پورتهای باز استفاده شوند.

راهنمای سریع توابع مطلب

نرم افزار مطلب مشتمل بر ۲۰ گروه تابع داخلی می باشد. برخی از توابع در مفسر آن ساخته می‌شوند، در حالیکه عمدهٔ این توابع بصورت *M-file* ایجاد می شوند. در ذیل توابع داخلی مطلب به همراه توابع مربوط به جعبه ابزارهای آن آورده شده است.

MATLAB‘s Main Categories of Functions

color	Color control and lighting model functions.
datafun	Data analysis and Fourier transform functions.
demos	Demonstrations and samples.
elfun	Elementary math functions.
elmat	Elementary matrices and matrix manipulation.
funfun	Function functions - nonlinear numerical methods.
general	General purpose commands.
graphics	General purpose graphics functions.
iofun	Low-level file I/O functions.
lang	Language constructs and debugging.
matfun	Matrix functions - numerical linear algebra.
ops	Operators and special characters.
plotxy	Two dimensional graphics.
plotxyz	Three dimensional graphics.
polyfun	Polynomial and interpolation functions.
sparfun	Sparse matrix functions.

specfun	Specialized math functions.
specmat	Specialized matrices.
sounds	Sound processing functions.
strfun	Character string functions.

General Purpose Commands

Managing Commands and Functions

demo	Run demos.
help	Online documentation.
info	Information about MATLAB and The MathWorks.
look for	Keyword search through the help entries.
path	Control MATLAB's search path.
type	List M-file.
what	Directory listing of M-, MAT- and MEX-files.
which	Locate functions and files.

MATLAB Toolbox Quick Reference

Signal Processing Toolbox

Filter Analysis

abs	Absolute value (magnitude).
angle	Phase angle.
freqs	Frequency response of analog filters.
freqspace	Frequency spacing for frequency response.
freqz	Compute the frequency response of digital filters.
freqzplot	Plot frequency response data.
grpdelay	Compute the average filter delay (group delay).
impz	Compute the impulse response of digital filters.
unwrap	Unwrap phase angles.
zplane	Zero-pole plot.

Filter Implementation

conv	Convolution and polynomial multiplication.
conv2	Two-dimensional convolution.
deconv	Deconvolution and polynomial division.
fftfilt	FFT-based FIR filtering using the overlap-add method.

filter	Filter data with a recursive (IIR) or nonrecursive (FIR) filter.
filter2	Two-dimensional digital filtering.
filtfilt	Zero-phase digital filtering.
filtic	Find initial conditions for a transposed direct form II filter implementation.
latcfilt	Lattice and lattice-ladder filter implementation.
medfilt1	One-dimensional median filtering.
sgolayfilt	Savitzky-Golay filtering.
sosfilt	Second-order (biquadratic) IIR digital filtering.
upfirdn	Upsample, apply an FIR filter, and downsample.

FIR Digital Filter Design

convmtx	Convolution matrix.
cremez	Complex and nonlinear-phase equiripple FIR filter design.
fir1	Design a window-based finite impulse response filter.
fir2	Design a frequency sampling-based finite impulse response filter.
fircls	Constrained least square FIR filter design for multiband filters.
fircls1	Constrained least square filter design for lowpass and highpass linear phase FIR filters.
firls	Least square linear-phase FIR filter design.
firrcos	Raised cosine FIR filter design.

intfilt	Interpolation FIR filter design.
kaiserord	Estimate parameters for an FIR filter design with Kaiser window.
remez	Compute the Parks-McClellan optimal FIR filter design.
remezord	Parks-McClellan optimal FIR filter order estimation.
sgolay	Savitzky-Golay filter design.

IIR Digital Filter Design--Classical and Direct

butter	Butterworth analog and digital filter design.
cheby1	Chebyshev type I filter design (passband ripple).
cheby2	Chebyshev type II filter design (stopband ripple).
ellip	Elliptic (Cauer) filter design.
maxflat	Generalized digital Butterworth filter design.
prony	Prony's method for time-domain IIR filter design.
stmcb	Compute a linear model using Steiglitz-McBride iteration.
yulewalk	Recursive digital filter design.

IIR Filter Order Estimation

buttord	Calculate the order and cutoff frequency for a Butterworth filter.
cheb1ord	Calculate the order for a Chebyshev type I filter.
cheb2ord	Calculate the order for a Chebyshev type II filter.
ellipord	Calculate the minimum order for elliptic filters.

Analog Lowpass Filter Prototypes

besselap	Bessel analog lowpass filter prototype.
buttap	Butterworth analog lowpass filter prototype.
cheb1ap	Chebyshev type I analog lowpass filter prototype.
cheb2ap	Chebyshev type II analog lowpass filter prototype.
ellipap	Elliptic analog lowpass filter prototype.

Analog Filter Design

besself	Bessel analog filter design.
butter	Butterworth analog and digital filter design.
cheby1	Chebyshev type I filter design (passband ripple).
cheby2	Chebyshev type II filter design (stopband ripple).
ellip	Elliptic (Cauer) filter design.

Analog Filter Transformation

lp2bp	Transform lowpass analog filters to bandpass.
lp2bs	Transform lowpass analog filters to bandstop.
lp2hp	Transform lowpass analog filters to highpass.
lp2lp	Change the cut -off frequency for a lowpass analog filter.

Filter Discretization

bilinear	Bilinear transformation method for analog-to-digital filter conversion.
impinvar	Impulse invariance method for analog-to-digital filter conversion.

Linear System Transformations

latc2tf	Convert lattice filter parameters to transfer function form.
polystab	Stabilize a polynomial.
polyscale	Scale the roots of a polynomial.
residuez	z-transform partial-fraction expansion.
sos2ss	Convert digital filter second-order section parameters to state-space form.
sos2tf	Convert digital filter second-order section data to transfer function form.
sos2zp	Convert digital filter second-order sections parameters to zero-pole-gain form.
ss2sos	Convert digital filter state-space parameters to second-order sections form.
ss2tf	Convert state-space filter parameters to transfer function form.
ss2zp	Convert state-space filter parameters to zero-polegain form.
tf2latc	Convert transfer function filter parameters to latticefilter form.

tf2sos	Convert digital filter transfer function data to secondorder sections form.
tf2ss	Convert transfer function filter parameters to statespace form.
tf2zp	Convert transfer function filter parameters to zeropole-gain form.
zp2sos	Convert digital filter zero-pole-gain parameters to second-order sections form.
zp2ss	Convert zero-pole-gain filter parameters to statespace form.
zp2tf	Convert zero-pole-gain filter parameters to transfer function form.

Windows

Bartlett	Compute a Bartlett window.
Blackman	Compute a Blackman window.
boxcar	Compute a rectangular window.
chebwin	Compute a Chebyshev window.
hamming	Compute a Hamming window.
hann	Compute the Hann (Hanning) window.
Kaiser	Compute a Kaiser window.
triang	Compute a triangular window.

Transforms

czt	Chirp z-transform.
dct	Discrete cosine transform (DCT).
dftmtx	Discrete Fourier transform matrix.
fft	Compute the one-dimensional fast Fourier transform.
fft2	Compute the two-dimensional fast Fourier transform.
fftshift	Rearrange the outputs of the FFT functions.
Hilbert	Compute the discrete-time analytic signal using the Hilbert transform.
idct	Inverse discrete cosine transform.
ifft	One-dimensional inverse fast Fourier transform.
ifft2	Two-dimensional inverse fast Fourier transform.

Cepstral Analysis

cceps	Complex cepstral analysis.
icceps	Inverse complex cepstrum.
rceps	Real cepstrum and minimum phase reconstruction.

Statistical Signal Processing and Spectral

Analysis

cohere	Estimate magnitude squared coherence function between two signals.
corrcoef	Compute the correlation coefficient matrix.
corrmtx	Compute a data matrix for autocorrelation matrix estimation.
cov	Compute the covariance matrix.
csd	Estimate the cross spectral density (CSD) of two signals.
pburg	Estimate the power spectral density using the Burg method.
pcov	Estimate the power spectral density using the covariance method.
peig	Estimate the pseudospectrum using the eigenvector method.
periodogram	Estimate the power spectral density (PSD) of a signal using a periodogram.
pmcov	Estimate the power spectral density using the modified covariance method.
pmtm	Estimate the power spectral density using the multitaper method (MTM).
pmusic	Estimate the power spectral density using MUSIC algorithm.

psdplot	Plot power spectral density (PSD) data.
pwelch	Estimate the power spectral density (PSD) of a signal using Welch's method.
pyulear	Estimate the power spectral density using the Yule-Walker AR method.
rooteig	Estimate frequency and power content using the eigenvector method.
rootmusic	Estimate frequency and power content using the root MUSIC algorithm.
tfe	Estimate the transfer function from input and output.
xcorr	Estimate the cross-correlation function.
xcorr2	Estimate the two-dimensional cross-correlation.
xcov	Estimate the cross-covariance function (equal to mean-removed cross-correlation).

Parametric Modeling

arburg	Compute an estimate of AR model parameters using the Burg method.
arcov	Compute an estimate of AR model parameters using the covariance method.
armcov	Compute an estimate of AR model parameters using the modified covariance method.
aryule	Compute an estimate of AR model parameters using the Yule-Walker method.
ident	See the System Identification Toolbox documentation.

invfreqs	Identify continuous-time filter parameters from frequency response data.
invfreqz	Identify discrete-time filter parameters from frequency response data.
prony	Prony's method for time domain IIR filter design.
stmcb	Compute a linear model using Steiglitz-McBride iteration.

Linear Prediction

ac2poly	Convert an autocorrelation sequence to prediction polynomial.
ac2rc	Convert an autocorrelation sequence to reflection coefficients.
is2rc	Convert inverse sine parameters to reflection coefficients.
lar2rc	Convert log area ratio parameters to reflection coefficients.
levinson	Compute the Levinson-Durbin recursion.
lpc	Compute linear prediction filter coefficients.
lsf2poly	Convert line spectral frequencies to a prediction filter coefficients.
poly2ac	Convert a prediction filter polynomial to an autocorrelation sequence.
poly2lsf	Convert prediction filter coefficients to line spectral frequencies.
poly2rc	Convert a prediction filter polynomial to reflection coefficients.

rc2ac	Convert reflection coefficients to an autocorrelation sequence.
rc2is	Convert reflection coefficients to inverse sine parameters.
rc2lar	Convert reflection coefficients to log area ratio parameters.
rc2poly	Convert reflection coefficients to a prediction filter polynomial.
rlevinson	Compute the reverse Levinson-Durbin recursion.
schurrc	Compute reflection coefficients from an autocorrelation sequence.

Multirate Signal Processing

decimate	Decrease the sampling rate for a sequence (decimation).
Interp	Increase sampling rat e by an integer factor (interpolation).
interp1	One-dimensional data interpolation (table lookup).
resample	Change sampling rate by any rational factor.

spline

upfirdn	Upsample, apply an FIR filter, and downsample.

Waveform Generation

chirp	Generate a swept-frequency cosine.

diric	Compute the Dirichlet or periodic sinc function.
gauspuls	Generate a Gaussian-modulated sinusoidal pulse.
gmonopuls	Generate a Gaussian monopulse.
pulstran	Generate a pulse train.
rectpuls	Generate a sampled aperiodic rectangle.
sawtooth	Generate a sawtooth or triangle wave.
sinc	Sinc function.
square	Generate a square wave.
tripuls	Generate a sampled aperiodic triangle.
vco	Voltage controlled oscillator.

Specialized Operations

buffer	Buffer a signal vector into a matrix of data frames.
cell2sos	Convert a cell array for second-order sections to a second-order section matrix.
cplxpair	Group complex numbers into complex conjugate pairs.
demod	Demodulation for communications simulation.
dpss	Discrete prolate spheroidal sequences (Slepian sequences).
dpssclear	Remove discrete prolate spheroidal sequences from database.
dpssdir	Discrete prolate spheroidal sequences database directory.
dpssload	Load discrete prolate spheroidal sequences from database.

dpsssave	Save discrete prolate spheroidal sequences in database.
eqtflength	Make the lengths of a transfer function's numerator and denominator equal.
modulate	Modulation for communications simulation.
seqperiod	Compute the period of a sequence.
sos2cell	Convert a second-order section matrix to cell arrays.
specgram	Time-dependent frequency analysis (spectrogram).
stem	Plot discrete sequence data.
strips	Strip plot.
udecode	Decode 2n-level quantized integer inputs to floatingpoint outputs.
uencode	Quantize and encode floating-point inputs to integer outputs.

Graphical User Interfaces

fdatool	Open the Filter Design and Analysis Tool.
sptool	Interactive digital signal processing tool (SPTool).

Image Processing Toolbox

Image Display

colorbar	Display colorbar. (This is a MATLAB function. See the online MATLAB Function Reference for its reference page.)

getimage	Get image data from axes.
image	Create and display image object. (This is a MATLAB function. See the online MATLAB Function Reference for its reference page.)
imagesc	Scale data and display as image. (This is a MATLAB function. See the online MATLAB Function Reference for its reference page.)
immovie	Make movie from multiframe indexed image.
imshow	Display image.
montage	Display multiple image frames as rectangular montage.
subimage	Display multiple images in single figure.
truesize	Adjust display size of image.
warp	Display image as texture-mapped surface.
zoom	Zoom in and out of image or 2-D plot. (This is a MATLAB function. See the online MATLAB Function Reference for its reference page.)

Image File I/O

imfinfo	Return information about image file. (This is a MATLAB function. See the online MATLAB Function Reference for its reference page.)
imread	Read image file. (This is a MATLAB function. See the online MATLAB Function Reference for its reference page.)

imwrite	Write image file. (This is a MATLAB function. See the online MATLAB Function Reference for its reference page.)

Geometric Operations

imcrop	Crop image.
imresize	Resize image.
imrotate	Rotate image.
interp2	2-D data interpolation. (This is a MATLAB function. See the online MATLAB Function Reference for its reference page.)

Pixel Values and Statistics

corr2	Compute 2-D correlation coefficient.
imcontour	Create contour plot of image data.
imfeature	Compute feature measurements for image regions.
imhist	Display histogram of image data.
impixel	Determine pixel color values.
improfile	Compute pixel-value cross-sections along line segments.
mean2	Compute mean of matrix elements.
pixval	Display information about image pixels.
std2	Compute standard deviation of matrix elements.

Image Analysis

edge	Find edges in intensity image.
qtdecomp	Perform quadtree decomposition.
qtgetblk	Get block values in quadtree decomposition.
qtsetblk	Set block values in quadtree decomposition.

Image Enhancement

histeq	Enhance contrast using histogram equalization.
imadjust	Adjust image intensity values or colormap.
imnoise	Add noise to an image.
medfilt2	Perform 2-D median filtering.
ordfilt2	Perform 2-D order-statistic filtering.
wiener2	Perform 2-D adaptive noise-removal filtering.

Linear Filtering

conv2	Perform 2-D convolution. (This is a MATLAB function. See the online MATLAB Function Reference for its reference page.)
convmtx2	Compute 2-D convolution matrix.
convn	Perform N-D convolution. (This is a MATLAB function. See the online MATLAB Function Reference for its reference page.)

filter2	Perform 2-D filtering. (This is a MATLAB function. See the online MATLAB Function Reference for its reference page.)
fspecial	Create predefined filters.

Linear 2-D Filter Design

freqspace	Determine 2-D frequency response spacing. (This is a MATLAB function. See the online MATLAB Function Reference for its reference page.)
freqz2	Compute 2-D frequency response.
fsamp2	Design 2-D FIR filter using frequency sampling.
ftrans2	Design 2-D FIR filter using frequency transformation.
fwind1	Design 2-D FIR filter using 1-D window method.
fwind2	Design 2-D FIR filter using 2-D window method.

Image Transforms

dct2	Compute 2-D discrete cosine transform.
dctmtx	Compute discrete cosine transform matrix.
fft2	Compute 2-D fast Fourier transform. (This is a MATLAB function. See the online MATLAB Function Reference for its reference page.)
fftn	Compute N-D fast Fourier transform. (This is a MATLAB function. See the online MATLAB Function Reference for its reference page.)

fftshift	Reverse quadrants of output of FFT. (This is a MATLAB function. See the online MATLAB Function Reference for its reference page.)
idct2	Compute 2-D inverse discrete cosine transform.
ifft2	Compute 2-D inverse fast Fourier transform. (This is a MATLAB function. See the online MATLAB Function Reference for its reference page.)
ifftn	Compute N-D inverse fast Fourier transform. (This is a MATLAB function. See the online MATLAB Function Reference for its reference page.)
iradon	Compute inverse Radon transform.
phantom	Generate a head phantom image.
radon	Compute Radon transform.

Neighborhood and Block Processing

bestblk	Choose block size for block processing.
blkproc	Implement distinct block processing for image.
col2im	Rearrange matrix columns into blocks.
colfilt	Perform neighborhood operations using columnwise functions.
im2col	Rearrange image blocks into columns.
nlfilter	Perform general sliding-neighborhood operations.

Binary Image Operations

applylut	Perform neighborhood operations using lookup tables.
bwarea	Compute area of objects in binary image.
bweuler	Compute Euler number of binary image.
bwfill	Fill background regions in binary image.
bwlabel	Label connected components in binary image.
bwmorph	Perform morphological operations on binary image.
bwperim	Determine perimeter of objects in binary image.
bwselect	Select objects in binary image.
dilate	Perform dilation on binary image.
erode	Perform erosion on binary image.
makelut	Construct lookup table for use with applylut.

Region-Based Processing

roicolor	Select region of interest, based on color.
roifill	Smoothly interpolate within arbitrary region.
roifilt2	Filter a region of interest.
roipoly	Select polygonal region of interest.

Colormap Manipulation

brighten	Brighten or darken colormap. (This is a MATLAB function. See the online MATLAB Function Reference for its reference page.)
cmpermute	Rearrane colors in colormap.
cmunique	Find unique colormap colors and corresponding image.
colormap	Set or get color lookup table. (This is a MATLAB function. See the online MATLAB Function Reference for its reference page.)
imapprox	Approximate indexed image by one with fewer colors.
rgbplot	Plot RGB colormap components. (This is a MATLAB function. See the online MATLAB Function Reference for its reference page.)

Color Space Conversions

hsv2rgb	Convert HSV values to RGB color space. (This is a MATLAB function. See the online MATLAB Function Reference for its reference page.)
ntsc2rgb	Convert NTSC values to RGB color space.
rgb2hsv	Convert RGB values to HSV color space. (This is a MATLAB function. See the online MATLAB Function Reference for its reference page.)
rgb2ntsc	Convert RGB values to NTSC color space.
rgb2ycbcr	Convert RGB values to YCbCr color space.

ycbcr2rgb	Convert YCbCr values to RGB color space.

Image Types and Type Conversions

dither	Convert image using dithering.
double	Convert data to double precision. (This is a MATLAB function. See the online MATLAB Function Reference for its reference page.)
gray2ind	Convert intensity image to indexed image.
grayslice	Create indexed image from intensity image by thresholding.
im2bw	Convert image to binary image by thresholding.
im2double	Convert image array to double precision.
im2uint16	Convert image array to 16-bit unsigned integers.
im2uint8	Convert image array to 8-bit unsigned integers.
ind2gray	Convert indexed image to intensity image.
ind2rgb	Convert indexed image to RGB image.
isbw	Return true for binary image.
isgray	Return true for intensity image.
isind	Return true for indexed image.
isrgb	Return true for RGB image.
mat2gray	Convert matrix to intensity image.
rgb2gray	Convert RGB image or colormap to grayscale.
rgb2ind	Convert RGB image to indexed image.

uint16	Convert data to unsigned 16-bit integers. (This is a MATLAB function. See the online MATLAB Function Reference for its reference page.)
uint8	Convert data to unsigned 8-bit integers. (This is a MATLAB function. See the online MATLAB Function Reference for its reference page.)

Toolbox Preferences

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

Demos

dctdemo	2-D DCT image compression demo.
edgedemo	Edge detection demo.
firdemo	2-D FIR filtering and filter design demo.
imadjdemo	Intensity adjustment and histogram equalization demo.
nrfiltdemo	Noise reduction filtering demo.
qtdemo	Quadtree decomposition demo.
roidemo	Region-of-interest processing demo.

Slide Shows

ipss001	Region labeling of steel grains.
ipss002	Feature-based logic.

ipss003	Correction of nonuniform illumination.

Neural Network Toolbox

Analysis Functions

errsurf	Error surface of a single input neuron.
maxlinlr	Maximum learning rate for a linear neuron.

Distance Functions

boxdist	Distance between two position vectors.
dist	Euclidean distance weight function.
linkdist	Link distance function.
mandist	Manhattan distance weight function.

Graphical Interface Function

nntool	Neural Network Tool - Graphical User Interface.

Layer Initialization Functions

initnw	Nguyen-Widrow layer initialization function.
initwb	By-weight-and-bias layer initialization function.

Learning Functions

learncon	Conscience bias learning function.
learngd	Gradient descent weight/bias learning function.
learngdm	Grad. descent w/momentum weight/bias learning function.
learnh	Hebb weight learning function.
learnhd	Hebb with decay weight learning rule.
learnis	Instar weight learning function.
learnk	Kohonen weight learning function.
learnlv1	LVQ1 weight learning function.
learnlv2	LVQ2 weight learning function.
learnos	Outstar weight learning function.
learnp	Perceptron weight and bias learning function.
learnpn	Normalized perceptron weight and bias learning function.
learnsom	Self-organizing map weight learning function.
learnwh	Widrow-Hoff weight and bias learning rule.

Line Search Functions

srchbac	One-dim. minimization using backtracking search.
srchbre	One-dim. interval location using Brent's method.
srchcha	One-dim. minimization using Charalambous' method.

srchgol	One-dim. minimization using Golden section search.
srchhyb	One-dim. minimization using Hybrid bisection/cubic search.

Net Input Derivative Functions

dnetprod	Product net input derivative function.
dnetsum	Sum net input derivative function.

Net Input Functions

netprod	Product net input function.
netsum	Sum net input function.

Network Initialization Functions

initlay	Layer-by-layer network initialization function.

Network Use Functions

adapt	Allow a neural network to adapt.
disp	Display a neural network's properties.
display	Display a neural network variable's name and properties.
init	Initialize a neural network.
sim	Simulate a neural network.
train	Train a neural network.

New Networks Functions

network	Create a custom neural network.
newc	Create a competitive layer.
newcf	Create a cascade-forward backpropagation network.
newelm	Create an Elman backpropagation network.
newff	Create a feed-forward backpropagation network.
newfftd	Create a feed-forward input -delay backprop network.
newgrnn	Design a generalized regression neural network.
newhop	Create a Hopfield recurrent network.
newlin	Create a linear layer.
newlind	Design a linear layer.
newlvq	Create a learning vector quantization network.
newp	Create a perceptron.
newpnn	Design a probabilistic neural network.
newrb	Design a radial basis network.
newrbe	Design an exact radial basis network.
newsom	Create a self-organizing map.

Performance Derivative Functions

dmae	Mean absolute error performance derivative function.
dmse	Mean squared error performance derivatives function.

dmsereg	Mean squared error w/reg performance derivative function.
dsse	Sum squared error performance derivative function.

Performance Functions

mae	Mean absolute error performance function.
mse	Mean squared error performance function.
msereg	Mean squared error w/reg performance function.
sse	Sum squared error performance function.

Plotting Functions

hintonw	Hinton graph of weight matrix.
hintonwb	Hinton graph of weight matrix and bias vector.
plotbr	Plot network perf. for Bayesian regularization training.
plotep	Plot weight and bias position on error surface.
plotes	Plot error surface of single input neuron.
plotpc	Plot classification line on perceptron vector plot.
plotperf	Plot network performance.
plotpv	Plot perceptron input target vectors.
plotsom	Plot self-organizing map.
plotv	Plot vectors as lines from the origin.
plotvec	Plot vectors with different colors.

Pre and Post Processing Functions

postmnmx	Unnormalize data which has been norm. by prenmmx.
postreg	Postprocess network response w. linear regression analysis.
poststd	Unnormalize data which has been normalized by prestd.
premnmx	Normalize data for maximum of 1 and minimum of -1.
prepca	Principal component analysis on input data.
prestd	Normalize data for unity standard deviation and zero mean.
tramnmx	Transform data with precalculated minimum and max.
trapca	Transform data with PCA matrix computed by prepca.
trastd	Transform data with precalc. mean & standard deviation.

Simulink Support Function

gensim	Generate a Simulink block for neural network simulation.

Topology Functions

gridtop	Gridtop layer topology function.
hextop	Hexagonal layer topology function.
randtop	Random layer topology function.

Training Functions

trainb	Batch training with weight and bias learning rules.
trainbfg	BFGS quasi-Newton backpropagation.
trainbr	Bayesian regularization.
trainc	Cyclical order incremental update.
traincgb	Powell-Beale conjugate gradient backpropagation.
traincgf	Fletcher-Powell conjugate gradient backpropagation.
traincgp	Polak-Ribiere conjugate gradient backpropagation.
traingd	Gradient descent backpropagation.
traingda	Gradient descent with adaptive lr backpropagation.
traingdm	Gradient descent with momentum backpropagation.
traingdx	Gradient descent with momentum & adaptive lr backprop.
trainlm	Levenberg-Marquardt backpropagation.
trainoss	One step secant backpropagation.
trainr	Random order incremental update.
trainrp	Resilient backpropagation (Rprop).
trains	Sequential order incremental update.
trainscg	Scaled conjugate gradient backpropagation.

Transfer Derivative Functions

dhardlim	Hard limit transfer derivative function.

dhardlms	Symmetric hard limit transfer derivative function.
dlogsig	Log sigmoid transfer derivative function.
dposlin	Positive linear transfer derivative function.
dpurelin	Linear transfer derivative function.
dradbas	Radial basis transfer derivative function.
dsatlin	Saturating linear transfer derivative function.
dsatlins	Symmetric saturating linear transfer derivative function.
dtansig	Hyperbolic tangent sigmoid transfer derivative function.
dtribas	Triangular basis transfer derivative function.

Transfer Functions

compet	Competitive transfer function.
hardlim	Hard limit transfer function.
hardlims	Symmetric hard limit transfer function.
logsig	Log sigmoid transfer function.
poslin	Positive linear transfer function.
purelin	Hard limit transfer function.
radbas	Radial basis transfer function.
satlin	Saturating linear transfer function.
satlins	Symmetric saturating linear transfer function.
softmax	Softmax transfer function.
tansig	Hyperbolic tangent sigmoid transfer function.

tribas	Triangular basis transfer function.

Utility Functions

calca	Calculate network outputs and other signals.
calca1	Calculate network signals for one time step.
calce	Calculate layer errors.
calce1	Calculate layer errors for one time step.
calcgx	Calc. weight and bias perform. gradient as a single vector.
calcjejj	Calculate Jacobian performance vector.
calcjx	Calculate weight and bias performance Jacobian as a single matrix.
calcpd	Calculate delayed network inputs.
calcperf	Calculation network outputs, signals, and performance.
formx	Form bias and weights into single vector.
getx	Get all network weight and bias values as a single vector.
setx	Set all network weight and bias values with a single vector.

Vector Functions

cell2mat	Combine a cell array of matrices into one matrix.
combvec	Create all combinations of vectors.
con2seq	Converts concurrent vectors to sequential vectors.
concur	Create concurrent bias vectors.

ind2vec	Convert indices to vectors.
mat2cell	Break matrix up into cell array of matrices.
minmax	Ranges of matrix rows.
normc	Normalize columns of matrix.
normr	Normalize rows of matrix.
pnormc	Pseudo-normalize columns of matrix.
quant	Discretize value as multiples of a quantity.
seq2con	Convert sequential vectors to concurrent vectors.
sumsqr	Sum squared elements of matrix.
vec2ind	Convert vectors to indices.

Weight and Bias Initialization Functions

initcon	Conscience bias initialization function.
initzero	Zero weight and bias initialization function.
midpoint	Midpoint weight initialization function.
randnc	Normalized column weight initialization function.
randnr	Normalized row weight initialization function.
rands	Symmetric random weight/bias initialization function.
revert	Change ntwk wts. and biases to prev. initialization values.

Weight Derivative Function

ddotprod	Dot product weight derivative function.

Weight Functions

dist	Euclidean distance weight function.
dotprod	Dot product weight function.
mandist	Manhattan distance weight function.
negdist	Negative distance weight function.
normprod	Normalized dot product weight function.

Transfer Function

compet	Competitive transfer function.
hardlim	Hard limit transfer function.
hardlims	Symmetric hard limit transfer function.
logsig	Log sigmoid transfer function.
poslin	Positive linear transfer function.
purelin	Linear transfer function.
radbas	Radial basis transfer function.
satlin	Saturating linear transfer function.
satlins	Symmetric saturating linear transfer function
softmax	Softmax transfer function.
tansig	Hyperbolic tangent sigmoid transfer function.
tribas	Triangular basis transfer function.

Statistics Toolbox

Parameter Estimation

betafit	Parameter estimation for the beta distribution.
betalike	Beta log-likelihood function.
binofit	Parameter estimation for the binomial distribution.
expfit	Parameter estimation for the exponential distribution.
gamfit	Parameter estimation for the gamma distribution.
gemlike	Gamma log-likelihood function.
mle	Maximum likelihood estimation.
normfit	Parameter estimation for the normal distribution.
normlike	Normal log-likelihood function.
poissfit	Parameter estimation for the Poisson distribution.
raylfit	Rayleigh parameter estimation.
unifit	Parameter estimation for the uniform distribution.
weibfit	Weibull parameter estimation.

Cumulative Distribution Functions (cdf)

betacdf	Beta cdf.
binocdf	Binomial cdf.
cdf	Parameterized cdf routine.

chi2cdf	Chi-square cdf.
expcdf	Exponential cdf.
fcdf	F cdf.
gamcdf	Gamma cdf.
geocdf	Geometric cdf.
hygecdf	Hypergeometric cdf.
logncdf	Lognormal cdf.
nbincdf	Negative binomial cdf.
ncfcdf	Noncentral F cdf.
nctcdf	Noncentral t cdf.
ncx2cdf	Noncentral Chi-square cdf.
normcdf	Normal (Gaussian) cdf.
poisscdf	Poisson cdf.
raylcdf	Rayleigh cdf.
tcdf	Student's t cdf.
unidcdf	Discrete uniform cdf.
unifcdf	Continuous uniform cdf.
weibcdf	Weibull cdf.

Probability Density Functions (pdf)

betapdf	Beta pdf.
binopdf	Binomial pdf.

chi2pdf	Chi-square pdf.
exppdf	Exponential pdf.
fpdf	F pdf.
gampdf	Gamma pdf.
geopdf	Geometric pdf.
hygepdf	Hypergeometric pdf.
lognpdf	Lognormal pdf.
nbinpdf	Negative binomial pdf.
ncfpdf	Noncentral F pdf.
nctpdf	Noncentral t pdf.
ncx2pdf	Noncentral Chi-square pdf.
normpdf	Normal (Gaussian) pdf.
pdf	Parameterized pdf routine.
poisspdf	Poisson pdf.
raylpdf	Rayleigh pdf.
tpdf	Student's t pdf.
unidpdf	Discrete uniform pdf.
unifpdf	Continuous uniform pdf.
weibpdf	Weibull pdf.

Inverse Cumulative Distribution Functions

betainv	Beta critical values.

binoinv	Binomial critical values.
chi2inv	Chi-square critical values.
expinv	Exponential critical values.
finv	F critical values.
gaminv	Gamma critical values.
geoinv	Geometric critical values.
hygeinv	Hypergeometric critical values.
icdf	Parameterized inverse distribution routine.
logninv	Lognormal critical values.
nbininv	Negative binomial critical values.
ncfinv	Noncentral F critical values.
nctinv	Noncentral t critical values.
ncx2inv	Noncentral Chi-square critical values.
norminv	Normal (Gaussian) critical values.
poissinv	Poisson critical values.
raylinv	Rayleigh critical values.
tinv	Student's t critical values.
unidinv	Discrete uniform critical values.
unifinv	Continuous uniform critical values.
weibinv	Weibull critical values.

Random Number Generators

betarnd	Beta random numbers.
binornd	Binomial random numbers.
chi2rnd	Chi-square random numbers.
exprnd	Exponential random numbers.
frnd	F random numbers.
gamrnd	Gamma random numbers.
geornd	Geometric random numbers.
hygernd	Hypergeometric random numbers.
lognrnd	Lognormal random numbers.
mvnrnd	Multivariate normal random numbers.
mvtrnd	Multivariate t random numbers.
nbinrnd	Negative binomial random numbers.
ncfrnd	Noncentral F random numbers.
nctrnd	Noncentral t random numbers.
ncx2rnd	Noncentral Chi-square random numbers.
normrnd	Normal (Gaussian) random numbers.
poissrnd	Poisson random numbers.
random	Parameterized random number routine.
raylrnd	Rayleigh random numbers.
trnd	Student's t random numbers.

unidrnd	Discrete uniform random numbers.
unifrnd	Continuous uniform random numbers.
weibrnd	Weibull random numbers.

Moments of Distribution Functions

betastat	Beta mean and variance.
binostat	Binomial mean and variance.
chi2stat	Chi-square mean and variance.
expstat	Exponential mean and variance.
fstat	F mean and variance.
gamstat	Gamma mean and variance.
geostat	Geometric mean and variance.
hygestat	Hypergeometric mean and variance.
lognstat	Lognormal mean and variance.
nbinstat	Negative binomial mean and variance.
ncfstat	Noncentral F mean and variance.
nctstat	Noncentral t mean and variance.
ncx2stat	Noncentral Chi-square mean and variance.
normstat	Normal (Gaussian) mean and variance.
poisstat	Poisson mean and variance.
raylstat	Rayleigh mean and variance.
tstat	Student's t mean and variance.

unidstat	Discrete uniform mean and variance.
unifstat	Continuous uniform mean and variance.
weibstat	Weibull mean and variance.

Descriptive Statistics

bootstrap	Bootstrap statistics for any function.
corrcoef	Correlation coefficients (in MATLAB).
cov	Covariance matrix (in MATLAB).
crosstab	Cross tabulation.
geomean	Geometric mean.
grpstats	Summary statistics by group.
harmmean	Harmonic mean.
iqr	Interquartile range.
kurtosis	Sample kurtosis.
mad	Mean absolute deviation.
mean	Arithmetic average (in MATLAB).
median	50th percentile (in MATLAB).
moment	Central moments of all orders.
nanmax	Maximum ignoring missing data.
nanmean	Average ignoring missing data.
nanmedian	Median ignoring missing data.
nanmin	Minimum ignoring missing data.

nanstd	Standard deviation ignoring missing data.
nansum	Sum ignoring missing data.
prctile	Empirical percentiles of a sample.
range	Sample range.
skewness	Sample skewness.
std	Standard deviation (in MATLAB).
tabulate	Frequency table.
trimmean	Trimmed mean.
var	Variance.

Statistical Plotting

boxplot	Box plots.
cdfplot	Plot of empirical cumulative distribution function.
errorbar	Error bar plot.
fsurfht	Interactive contour plot of a function.
gline	Interactive line drawing.
gname	Interactive point labeling.
gplotmatrix	Matrix of scatter plots grouped by a common variable.
gscatter	Scatter plot of two variables grouped by a third.
lsline	Add least-squares fit line to plotted data.
normplot	Normal probability plots.
pareto	Pareto charts.

qqplot	Quantile-Quantile plots.
rcoplot	Regression case order plot.
refcurve	Reference polynomial.
refline	Reference line.
surfht	Interactive interpolating contour plot.
weibplot	Weibull plotting.

Statistical Process Control

capable	Quality capability indices.
capaplot	Plot of process capability.
ewmaplot	Exponentially weighted moving average plot.
histfit	Histogram and normal density curve.
normspec	Plot normal density between limits.
schart	Time plot of standard deviation.
xbarplot	Time plot of means.

Cluster Analysis

cluster	Create clusters from linkage output.
clusterdata	Create clusters from a dataset.
cophenet	Calculate the cophenetic correlation coefficient.
dendrogram	Plot a hierarchical tree in a dendrogram graph.

inconsistent	Calculate the inconsistency values of objects in a cluster hierarchy tree.
linkage	Link objects in a dataset into a hierarchical tree of binary clusters.
pdist	Calculate the pairwise distance between objects in a dataset.
squareform	Reformat output of pdist function from vector to square matrix.
zscore	Normalize a dataset before calculating the distance.

Linear Models

anova1	One-way Analysis of Variance (ANOVA).
anova2	Two-way Analysis of Variance.
anovan	N-way analysis of variance.
aoctool	Interactive tool for analysis of covariance.
dummyvar	Dummy-variable coding.
friedman	Friedman's test (nonparametric two-way anova).
glmfit	Generalized linear model fitting.
kruskalwallis	Kruskal-Wallis test (nonparametric one-way anova).
leverage	Regression diagnostic.
lscov	Regression given a covariance matrix (in MATLAB).
manova1	One-way multivariate analysis of variance.
manovacluster	Draw clusters of group means for manova1.
multcompare	Multiple comparisons of means and other estimates.

polyconf	Polynomial prediction with confidence intervals.
polyfit	Polynomial fitting (in MATLAB).
polyval	Polynomial prediction (in MATLAB).
rcoplot	Residuals case order plot.
regress	Multiple linear regression.
regstats	Regression diagnostics.
ridge	Ridge regression.
rstool	Response surface tool.
robustfit	Robust regression model fitting.
rstool	Multidimensional response surface visualization (RSM).
stepwise	Stepwise regression GUI.
x2fx	Factor settings matrix (X) to design matrix (D).

Nonlinear Regression

nlinfit	Nonlinear least-squares fitting.
nlintool	Prediction graph for nonlinear fits.
nlparci	Confidence intervals on parameters.
nlpredci	Confidence intervals for prediction.
nnls	Nonnegative least squares (in MATLAB).

Design of Experiments

cordexch	D-optimal design using coordinate exchange.
daugment	D-optimal augmentation of designs.
dcovary	D-optimal design with fixed covariates.
ff2n	Two-level full factorial designs.
fracfact	Two-level fractional factorial design.
fullfact	Mixed level full factorial designs.
hadamard	Hadamard designs (in MATLAB).
rowexch	D-optimal design using row exchange.

Principal Components Analysis

barttest	Bartlett's test.
pcacov	PCA from covariance matrix.
pcares	Residuals from PCA.
princomp	PCA from raw data matrix.

Multivariate Statistics

classify	Linear Discriminant Analysis.
mahal	Mahalanobis distance.
manova1	One-way multivariate analysis of variance.
manovacluster	Draw clusters of group means for manova1.

Hypothesis Tests

ranksum	Wilcoxon rank sum test.
signrank	Wilcoxon signed rank test.
signtest	Sign test for paired samples.
ttest	One sample t -test.
ttest2	Two sample t -test.
ztest	Z-test.

Distribution Testing

jbtest	Jarque-Bera test of normality.
kstest	Kolmogo rov-Smirnov test for one sample.
kstest2	Kolmogorov-Smirnov test for two samples.
lillietest	Lilliefors test of normality.

Nonparametric Testing

friedman	Friedman's test (nonparametric two-way anova).
kruskalwallis	Kruskal-Wallis test (nonparametric one-way anova).
ranksum	Wilcoxon rank sum test (independent samples).
signrank	Wilcoxon sign rank test (paired samples).
signtest	Sign test (paired samples).

File I/O

caseread	Read casenames from a file.
casewrite	Write casenames from a string matrix to a file.
tblread	Retrieve tabular data from the file system.
tblwrite	Write data in tabular form to the file system.
tdfread	Read in text and numeric data from tab-delimited file.

Demonstrations

aoctool	Interactive tool for analysis of covariance.
disttool	Interactive exploration of distribution functions.
glmdemo	Generalized linear model slide show.
randtool	Interactive random number generation.
polytool	Interactive fitting of polynomial models.
rsmdemo	Interactive process experimentation and analysis.
robustdemo	Interactive tool to compare robust and least squares fits.

Data

census.mat	U. S. Population 1790 to 1980.
cities.mat	Names of U.S. metropolitan areas.
discrim.mat	Classification data.
gas.mat	Gasoline prices.

hald.mat	Hald data.
hogg.mat	Bacteria counts from milk shipments.
lawdata.mat	GPA versus LSAT for 15 law schools.
mileage.mat	Mileage data for three car models from two factories.
moore.mat	Five factor - one response regression data
parts.mat	Dimensional runout on 36 circular parts.
popcorn.mat	Data for popcorn example (anova2, friedman).
polydata.mat	Data for polytool demo.
reaction.mat	Reaction kinetics data.
sat.dat	ASCII data for tblread example.

Optimization Toolbox

Minimization

fgoalattain	Multiobjective goal attainment.
fminbnd	Scalar nonlinear minimization with bounds.
fmincon	Constrained nonlinear minimization.
fminimax	Minimax optimization.
fminsearch,fminunc	Unconstrained nonlinear minimization.
fseminf	Semi-infinite minimization.
linprog	Linear programming.
quadprog	Quadratic programming.

Equation Solving

\	Use \ (left division) to solve linear equations. See the Arithmetic Operators reference page.
fsolve	Nonlinear equation solving.
fzero	Scalar nonlinear equation solving.

Least Squares (Curve Fitting)

\	Use \ (left division) for linear least squares with no constraints. See the Arithmetic Operators reference page.
lsqlin	Constrained linear least squares.
lsqcurvefit	Nonlinear curve fitting.
lsqnonlin	Nonlinear least squares.
lsqnonneg	Nonnegative linear least squares.
optimset,optimget	Parameter setting.

Database Toolbox

General

logintimeout	Set or get time allowed to establish database connection.
setdbprefs	Set preferences for database actions for handling NULL values.

Database Connection

clearwarnings	Clear warnings for database connection.
close	Close database connection.
database	Connect to database.
get	Get property of database connection.
isconnection	Detect if database connection is valid.
isreadonly	Detect if database connection is read-only.
ping	Get status information about database connection.
set	Set properties for database connection.
sql2native	Convert JDBC SQL grammar to system's native SQL grammar.

SQL Cursor

close	Close cursor.
exec	Execute SQL statement and open cursor.
get	Get property of cursor object.
querytimeout	Get time allowed for a database SQL query to succeed.
set	Set RowLimit for cursor fetch.

Importing Data into MATLAB

attr	Get attributes of columns in fetched data set.

cols	Get number of columns in fetched data set.
columnnames	Get names of columns in fetched data set.
fetch	Import data into MATLAB cell array.
rows	Get number of rows in fetched data set.
width	Get field size of column in fetched data set.

Exporting Data to a Database

commit	Make database changes permanent.
insert	Export MATLAB cell array data into database table.
rollback	Undo database changes.
update	Replace data in database table with data from MATLAB cell array.

Database Metadata Object

bestrowid	Get database table unique row identifier.
columnprivileges	Get database column privileges.
columns	Get database table column names.
crossreference	Get information about primary and foreign keys.
dmd	Construct database metadata object.
exportedkeys	Get information about exported foreign keys.
get	Get database metadata properties.
importedkeys	Get information about imported foreign keys.

indexinfo	Get indices and statistics for database table.
primarykeys	Get primary key information for database table or schema.
procedurecolumns	Get catalog's stored procedure parameters and result columns.
procedures	Get catalog's stored procedures.
supports	Detect if property is supported by database metadata object.
tableprivileges	Get database table privileges.
tables	Get database table names.
versioncolumns	Get automatically updated table columns.

Driver Object

driver	Construct database driver object.
get	Get database driver properties.
isdriver	Detect if driver is a valid JDBC driver object.
isjdbc	Detect if driver is JDBC-compliant.
isurl	Detect if the database URL is valid.
register	Load database driver.
unregister	Unload database driver.

Drivermanager Object

drivermanager	Construct database drivermanager object.

get	Get database drivermanager properties.
set	Set database drivermanager properties.

Resultset Object

clearwarnings	Clear the warnings for the resultset.
close	Close resultset object.
get	Get resultset properties.
isnullcolumn	Detect if last record read in resultset was NULL.
namecolumn	Map resultset column name to resultset column index.

Resultset Metadata Object

get	Get resultset metadata properties.
rsmd	Construct resultset metadata object.

Visual Query Builder

confds	Configure data source for use with Visual Query Builder (JDBC only).
querybuilder	Start visual SQL query builder.