جزوه آزمایشگاه سیستم‌های قدرت

مقدمه: آموزش مقدماتی نرم‌افزار Matlab

۱- معرفی انواع پنجره

الف) پنجره فرمان (Command Window): در این محیط می‌توان دستورات مختلف را تایپ کرده و با فشاردادن کلید Enter اجرا نمود. در ضمن محیط مناسبی برای انجام عملیات ریاضی می‌باشد. لازم به ذکر است که این محیط مکان مناسبی برای نوشتن برنامه مورد نظر نیست. برای این کار باید از محیط برنامه‌نویسی M-File یا Script استفاده نمود. خروجی تمامی برنامه‌های نوشته شده در پنجره فرمان نمایش داده می‌شود. بنابراین این پنجره به نوعی صفحه نمایش خروجی است.

ب) پنجره فضای کاری (Work Space): در این محیط تمام متغیرهای تعریف شده در طول برنامه مورد نظر به همراه مقادیر و ابعاد آنها نشان داده می‌شود.

ج) پنجره تاریخچه دستورات (Command History): در این پنجره تمامی دستورات نوشته شده در محیط Command Window نشان داده می‌شود.

د) پنجره مسیر جاری (Current Directory): در این محیط اسامی تمامی فایل‌های موجود در پوشه یا فولدری که برنامه مورد نظر در آنجا ذخیره شده است، نشان داده می‌شود.

نکته: در بین چهار نوع پنجره مذکور، پنجره فرمان، پنجره اصلی بوده و می‌توان با بستن بقیه پنجره‌ها اندازه این پنجره را تا حد امکان بزرگ نمود. برای بازگرداندن پنجره‌ها باید گزینه Default را که در پنجره‌های فوقانی نرم‌افزار در بخش Desktop - Desktop-layout قرار دارد، انتخاب نمود.

تمرین ۱: تمام پنجره‌های مذکور در محیط Matlab را به جزء پنجره فرمان بسته و سپس به حالت اول باز گردانید.

۲- منوی help نرم‌افزار

دسترسی به help نرم‌افزار از طریق کلیک بر روی پنجره help – product help امکان‌پذیر است. با تایپ‌کردن دستور و یا کلید واژه مورد نظر می‌توان جزئیات مربوط به آن را در این محیط جستجو کرد. منوی help نرم‌افزار Matlab کاملترین و بهترین راه برای یادگیری نرم‌افزار است؛ استفاده از آن اکیداً توصیه می‌گردد.

تمرین ۲: در منوی help نرم‌افزار طرز عملکرد دستور clc را جویا شوید.

۳- نحوه ساختن M-File یا Script

با کلیک‌کردن بر روی آیکون سفید رنگ در قسمت بالا و چپ نرم‌افزار و یا انتخاب گزینهNew M-File در پنجره New – File می‌توان وارد محیط برنامه‌نویسی شد. با انجام این کار یک پنجره سفید رنگ با عنوان untitled باز می‌شود. با نوشتن اولین دستور در این محیط عنوان untitled ستاره‌دار می‌شود. پیدایش علامت ستاره هشداری به برنامه‌نویس جهت انتخاب اسم و ذخیره فایل مورد نظر است. برای این کار می‌توان بر روی آیکون آبی رنگ save کلیک کرده و نام برنامه و محل ذخیره آن را تعیین کرد. برای فراخوانی فایل برنامه نوشته شده از قبل باید بر روی آیکون open که آیکونی زرد رنگ موجود در منوی بالای صفحه اصلی نرم‌افزار و یا صفحه برنامه M-File جدید است، کلیک کرده و با انتخاب آدرس فایل مورد نظر و نام آن، آن فایل را باز نمود.

نکته: پسوند فایل‌های M-File ، m است؛ به طور مثال test.m

تمرین ۳: یک M-File جدید به نام کوچک خودتان بر روی Desktop بسازید. سپس آن فایل را بسته و دوباره فراخوانی کنید.

۴- دستورات clc، clear all و close all

این سه دستور در ابتدای تمامی برنامه‌ها در M-Fileها باید نوشته شود.

- دستور clc صفحه نمایش خروجی (پنجره فرمان) را پاک می‌کند.
- دستور clear all حافظه کامپیوتر اختصاص داده شده به نرم‌افزار Matlab را پاک می‌کند. به عبارت دیگر تمامی متغیرها و یا پنجره کاری (Work Space) را پاک می‌کند.
- دستور close all تمامی نمودارهایی (figures) را که مربوط به برنامه‌های قبلی است می‌بندد.

تمرین ۴:

الف) در پنجره فرمان تایپ کنید: $x = 2$ سپس کلید Enter را فشار دهید، چه اتفاقی می‌افتد؟

ب) حال در همان پنجره دستور clc را نوشته و اجرا کنید، نتیجه کار چیست؟ آیا مقدار متغیر x پاک شده است؟ تایپ کنید x و سپس کلید Enter را فشار دهید، چه نتیجه‌ای می‌گیرید؟

ج) حال در ادامه کار در پنجره فرمان دستور clear all را نوشته و اجرا کنید، چه اتفاقی می‌افتد؟ صفحه نمایش خروجی پاک می‌شود و یا متغیر x؟ جواب سوأل را چگونه می‌توان پی برد؟

د) حال سه دستور مذکور را در M-File جدید خود به ترتیب در سه سطر تایپ کرده و آیکون اجرا (یک فلش سبز رنگ در منوی بالای M-File) را کلیک کنید، چه پیغامی ظاهر می‌شود؟ گزینه نخست را انتخاب کنید و پنجره فرمان را مشاهده کنید؛ چه نتیجه‌ای می‌گیرید؟

۵- عملیات ریاضی

در نرم‌افزار Matlab برای انجام عملیات ریاضی جمع، تفریق، ضرب، تقسیم و توان به ترتیب از کاراکترهای + ، - ، * ، / و ^ می‌توان استفاده کرد. اولویت این عملگرها به ترتیب زیر است:

پرانتز، توان، ضرب یا تقسیم (هر کدام اول نوشته شود.) و نهایتاً جمع یا تفریق (هر کدام اول نوشته شود.)

نکته: عملگر جذر در این نرم‌افزار با دستور sqrt(x) تعریف شده است. این دستور ریشه دوم x را محاسبه می‌کند. همچنین به جای آن می‌توان از دستور $x\wedge(1/2)$ استفاده کرد. برای محاسبه عبارت $\sqrt[m]{x^n}$ از دستور $x\wedge(n/m)$ استفاده می‌شود.

تمرین ۵: در محیط M-File یکی یکی روابط زیر را تایپ کرده و آنها را اجرا کند، چه نتیجه‌ای می‌گیرید؟

الف) $x=8/4+4-0.5*2\wedge 2$

ب) $x=8/4+4-0.5*2\wedge 2;$

ج) $x=8/(4+4)-(0.5*2)\wedge 2$

د) $x=8*8/(5+4-0.5*2)\wedge 2$

ه) $x=\text{sqrt}(8\wedge -1/2+2.5-2\wedge \text{sqrt}(4))$

۶- کمیت‌های مختلط

در نرم‌افزار Matlab کمیت‌های مختلط در دستگاه دکارتی (مستطیلی) نمایش داده می‌شوند و قسمت‌های حقیقی و مختلط آن با حرف i تفکیک می‌شوند. برای تعریف یک کمیت مختلط به ۶ طریق می‌توان عمل کرد. این روش‌ها در غالب مثال زیر نشان داده شده است:

$x=4+\text{j}*3$ or $x=4+\text{i}*3$

$x=4+3*\text{j}$ or $x=4+3*\text{i}$

$x=4+3\text{j}$ or $x=4+3\text{i}$

تمرین ۶: در محیط M-File ابتدا کمیت مختلط z را به صورت زیر تعریف کرده و سپس یکی یکی روابط زیر را تایپ کرده و آنها را اجرا کند، چه نتیجه‌ای می‌گیرید؟

$z=4+\text{j}*3;$

الف) $\text{x}=\text{real}(z)$

ب) $\text{y}=\text{imag}(z)$

پ) $\text{q}=\text{conj}(z)$

ت) $k1 = \mathrm{sqrt}(x \wedge 2 + y \wedge 2)$

ث) $k2 = \mathrm{abs}(z)$

ج) $r1 = \mathrm{angle}(z)$

چ) $r2 = \mathrm{angle}(z) * (180 / \mathrm{pi})$

ح) $\mathrm{teta1} = \sin(\mathrm{r1})$

خ) $\mathrm{teta2} = \sin(\mathrm{r2} * \mathrm{pi} / 180)$

خ) $n1 = \mathrm{a}\sin(\mathrm{teta1})$

خ) $n2 = \mathrm{a}\sin(\mathrm{teta1}) * 180/\mathrm{pi}$

۷- دستور if و متعلقات آن

فرمت دستور if به طور کلی به صورت زیر است:

```
if (condition1)
   body1
elseif (condition2)
   body2
.
.
elseif (conditionN)
   body N
else
   body N +1
end
```

در صورت برقراری هر کدام از شرط‌های 1 تا N تنها دستورات نوشته شده در body مربوطه اجرا می‌شود و دستور if پایان می‌پذیرد. در صورتی که هیچ کدام از شرط‌ها برقرار نبود، دستورات نوشته شده در قسمت else اجرا می‌شود.

مثال ۱:

```
x = 2;
if (x > 1)
   y = 1;
else
   y = 3;
end
y
```

مثال۲:

```
x = 2;
if (x > 1)
   y = 1;
elseif (x < 1)
   y = 2;
else
   y = 3;
end
y
```

تمرین ۷: نتیجه برنامه زیر را به دو روش تحلیلی و برنامه‌نویسی به دست آورید.

```
x = 2;
if (x > 1 & x < 5)
   y = 1;
elseif (x < 1 || x > 10)
   y = 2;
else
   y = 20;
end
y
```

نکته مهم: برای اجرای برنامه‌ها به دو روش می‌توان عمل کرد:

۱- با کلیک‌کردن بر روی آیکون اجرا (آیکون سبز رنگ در منوی اصلی)

۲- با روش ردیابی (Trace): برای این کار در کنار اولین خط برنامه (سمت چپ) کلیک کنید. با انجام این کار یک دایره توپر قرمز رنگ کنار دستوری که مورد نظر ماست قرار داده می‌شود. پس از اجرای برنامه،

یک فلش سبز رنگ کنار دستوری که این دایره قرار دارد ظاهر می‌گردد. آیکون سبز رنگ اجرا در منوی اصلی غیرفعال شده و به جای آن چندین آیکون دیگر (سفید رنگ) که تاکنون غیرفعال بوده، فعال می‌گردد. با کلیک کردن بر روی آنها می‌توان خط به خط برنامه را دنبال نمود.

تمرین ۸: مثال‌های ۱ و ۲ و تمرین ۷ را به روش ردیابی اجرا کنید. به نظر شما حسن استفاده از روش ردیابی چیست؟

تمرین ۹: فرق میان آیکون‌های اجرای سفید رنگ step، step in، step out، و continue در چیست؟

۸- دستور for

فرمت دستور for به صورت زیر است:

```
for (k = m : n : p)
   body
end
```

هنگامی این دستور را به کار می‌بریم که بخواهیم یک عمل را به دفعات تکرار کنیم. برای این کار مقدار k برابر مقدار m قرار داده می‌شود. سپس با قدم‌های n، به مقدار k اضافه می‌شود تا زمانی که $k \le p$ شود. تعداد تکرارهای برابر $\left[\frac{p-m}{n}\right]+1$ می‌شود (در این رابطه، [] بیانگر براکت است.) ثابت کنید!

تمرین ۱۰: نتیجه برنامه زیر را به روش ردیابی دنبال کنید.

```
x = 0
for (k = 1 : 2 : 6)
   k
   x = x + 1
end
k
x
```

تمرین ۱۱: نتیجه برنامه زیر را به روش ردیابی دنبال کنید.

```
x = 0
for (k = 1:1:5)
    k
   x = x + k
end
x
```

تمرین ۱۲: نتیجه برنامه زیر را به روش ردیابی دنبال کنید.

```
x = 0
m = 10
n = −1
p = 7
for (k = m : n : p)
    k
   x = x + k
end
x
```

۹- دستور while

فرمت دستور while به صورت زیر است:

```
while (condition)
   body
end
```

مشابه دستور for برای انجام یک عمل تکراری به کار می‌رود با این تفاوت که در دستور for تعداد تکرارها مشخص است ولی در دستور while تعداد تکرارها مشخص نیست. در حقیقت، تا زمانی که شرط مقابل این دستور برقرار باشد، حلقه تکرار می‌گردد.

تمرین ۱۳: نتیجه برنامه زیر را به روش ردیابی دنبال کنید.

```
x = 0
k = 2
while (k > 0)
   k
   x = x + 1
   k = k - 1
end
k
x
```

تمرین ۱۴: نتیجه برنامه زیر را به روش ردیابی دنبال کنید.

```
x = 0
k = 2
while (k > 0)
   k
   x = x + 1
   k = k + 1
end
k
x
```

نکته: اگر در نرم‌افزار Matlab برنامه‌ای به اشتباه نوشته شود و اجرای برنامه در حلقه بینهایت قرار بگیرد، می‌توان با کلیک‌کردن در پنجره فرمان و به طور همزمان زدن کلیدهای ctlr و c، جلوی اجرای برنامه را گرفت.

۱۰- تعریف بردار

بردارها و ماتریس‌ها نقش مهمی در نرم‌افزار Matlab بازی می‌کنند. تعریف، کاربرد و نحوه دسترسی به درایه‌های آنها حائز اهمیت است. در مثال‌ها و تمرین‌های زیر به این موارد پرداخته می‌شود.

مثال۳: تشکیل یک بردار سطری با سه درایه ۱، ۲ و ۳ با قرار دادن این اعداد در داخل کروشه و قراردادن space بین آنها:

```
x = [1 2 3]
```

مثال۴: تشکیل یک بردار ستونی با سه درایه ۱، ۲ و ۳ با قرار دادن این اعداد در داخل کروشه و قراردادن علامت نقطه-ویرگول (;) بین آنها:

```
x = [1; 2; 3]
```

و یا با زدن کلید Enter بین آنها:

```
x = [1
     2
     3]
```

مثال۵: تشکیل یک ماتریس دو در سه با درایه‌های مساوی ۱:

```
x = [1 1 1;1 1 1]
```

مثال۶: تشکیل یک ماتریس سه در دو با درایه‌های مساوی ۱:

```
x = [1 1;1 1;1 1]
```

نکته: عملگر ترانهاده (Transpose) در نرم‌افزار Matlab علامت quotation به صورت ' مشخص می‌گردد.

مثال۷: تبدیل یک ماتریس سه در دو با درایه‌های مساوی ۱ به ماتریس دو در سه:

```
x = [1 1;1 1;1 1]'
```

مثال۸: دسترسی به درایه‌های یک بردار:

```
x = [1 2 5]
y = x(3)
```

و یا:

```
x = [1; 2; 5]
y = x(1)
```

مثال۹: دسترسی به درایه‌های یک ماتریس:

```
x = [1 2 5;8 6 7]
y = x(2,3)
z = x(1,2)
d = x(2,2)
```

مثال ۱۰: دسترسی به سطر و یا ستون یک ماتریس:

$x = [1\ \ 2\ \ 5; 8\ \ 6\ \ 7]$
$y = x(1,:)$
$z = x(:,1)$
$d = x(2,:)$

تمرین ۱۵: خروجی‌های برنامه زیر را به دست آورید.

$x = [2\ \ 20; 30\ \ 3; 40\ \ 45]$
$z = x(1,:)$
$r = x(3,2)$
$y = x'$
$s = y(2,3)$
$f = y(2,:)$
$t = y'$
$e = t(1,:) + t(2,:)$

مثال ۱۱: نحوه تشکیل یک بردار با عملگر دو نقطه (:) به صورت زیر است:

$x = [0:0.5:3]$
$y = [10:-2:6]$
$z = [10:-2:5]'$

مثال۱۲: دسترسی به بعد یک ماتریس به کمک دستور size:

$x = [1\ \ 2\ \ 5; 8\ \ 6\ \ 7]$
$y = size(x)$

و یا:

$x = [1\ \ 2\ \ 5; 8\ \ 6\ \ 7]$
$[m\ \ n] = size(x)$

به این ترتیب می‌توان تعداد سطر و ستون ماتریس x را به ترتیب در متغیرهای m و n ذخیره کرد.

مثال۱۳: محاسبه دترمینان یک ماتریس مربعی به کمک دستورdet:

$x = [1\ \ 2; 3\ \ 8]$
$y = \det(x)$

مثال۱۴: تشکیل ماتریس معکوس یک ماتریس مربعی به کمک دستورinv:

$x=[1\ \ 2;3\ \ 8]$
$y=inv(x)$

مثال۱۵: محاسبه مقادیر ویژه (eigenvalue) یک ماتریس مربعی به کمک دستورeig:

$x=[1\ \ 2;3\ \ 8]$
$y=eig(x)$

مثال۱۶: محاسبه مقادیر ویژه و بردار ویژه (eigenvector) یک ماتریس مربعی به کمک دستورeig:

$x=[1\ \ 0;0\ \ 2]$
$[V\ \ D]=eig(x)$

به این ترتیب می‌توان بردارهای ویژه ماتریس x را به صورت ستونی در ماتریس V و مقادیر ویژه ماتریس x را در ماتریس قطری D به تفکیک به دست آورد.

تمرین ۱۶: خروجی‌های برنامه زیر را به دست آورید.

$x=[(0:1:5)'\ \ (5:1:10)']$
$[w\ e]=size(x')$
$y=[6\ 0\ 0;0\ 5\ 0;0\ 0\ 4]$
$c=eig(y)$
$s=\det(y)$
$[a\ \ b]=eig(y)$

۱۱- حل یک معادله جبری چند جمله‌ای

برای به دست آوردن ریشه‌های یک معادله جبری چند جمله‌ای (محل برخورد منحنی با محور xها) که در حالت کلی به شکل $k_n y^n + k_{n-1} y^{n-1} + ... + k_2 y^2 + k_1 y + k_0 = 0$ است، باید ابتدا ضرایب این چند جمله‌ای را در یک بردار به صورت $x=[k_n \quad k_{n-1} \ ... \ k_2 \quad k_1 \quad k_0]$ قرار داد و سپس به کمک دستور roots(x) ریشه‌های آن را به دست آورد.

مثال۱۷: ریشه‌های معادله $y^5+5y^3+2y-9=0$ را به دست آورید.

$x=[1\ \ 0\ \ 5\ \ 0\ \ 2\ \ -9]$
$y=roots(x)$

تمرین ۱۷: ریشه معادله‌های جبری زیر را به دست آورید.

$y^5 + 9y = 0$
$y^{10} - y^3 + y - 1 = 0$
$(y^4 + 2)(y^2 + 2)(y + 2) = 0$

۱۲- رسم نمودار

برای رسم نمودار باید از دستور plot استفاده کرد. فرمت کلی آن به صورت زیر است:

```
plot(x, y, 'LineStyle','--','LineWidth', 2,'color','r')
xlabel('x','FontSize',16)
ylabel('y','FontSize',16)
```

که در آن x و y باید دو بردار با ابعاد مساوی باشند. به جای -- می‌توان - یا .- یا * یا : و به جای r (رنگ قرمز (red)) می‌توان k برای رنگ سیاه (black)، b برای رنگ آبی (blue)، y برای رنگ زرد (yellow)، g برای رنگ سبز (green)، w برای رنگ سفید (white)، m برای رنگ بنفش رو به صورتی (magenta)، c برای رنگ سبز رو به آبی یا فیروزه‌ای (cyan) می‌توان استفاده کرد. پهنای خط را هم می‌توان با تغییر عدد 2 تغییر داد. دستورات *xlabel* و *ylabel* برای مشخص کردن نوع متغیرهای x و y در کنار محورهای آنها می‌باشد.

مثال ۱۸: نمودار معادله $y = x^5 + 5x^3 + 2x - 9$ را رسم کنید.

```
x = [-10 : 0.1 : 10];
y = x.^5 + 5*x.^3 + 2*x - 9;
plot(x, y, 'LineStyle',':','LineWidth', 4,'color','g')
xlabel('x','FontSize',20)
ylabel('y','FontSize',20)
```

توجه: به کاربرد نقطه بعد از متغیر x در مثال فوق برای تشکیل بردار y دقت کنید. راه دیگر تشکیل بردار y و رسم نمودار معادله فوق به قرار زیر است:

```
x=[-10:0.1:10]';
n=length(x);
for(k=1:1:n)
    y(k)=x(k)^5+5*x(k)^3+2*x(k)-9;
end
plot(x,y,'LineStyle',':','LineWidth',4,'color','g')
xlabel('x','FontSize',20)
ylabel('y','FontSize',20)
```

توجه: به کاربرد دستور *length* در برنامه فوق دقت کنید.

تمرین ۱۸: فرق بین دو دستور *size* و *length* را با ارائه مثال‌های مختلف آزمایش کنید.

تمرین ۱۹: نمودار معادله‌های جبری $y^5+9y=0$ و $y^{10}-y^3+y-1=0$ را به هر دو روش فوق رسم کنید.

به کارگیری نرم‌افزار Matlab در مسائل مربوط به سیستم‌های قدرت

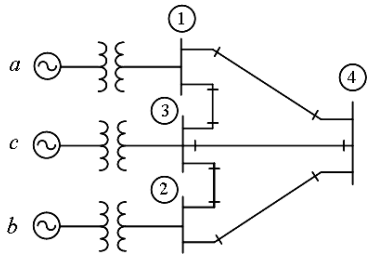
۱۳- برنامه تشکیل ماتریس ادمیتانس شبکه یا Y_{Bus}

برای تشکیل ماتریس ادمیتانس باید به دو نکته زیر توجه کرد:

۱- عنصر iام از قطر اصلی این ماتریس برابر مجموع ادمیتانس‌های متصل به باس بار iام است.

۲- عنصر سطر iام و ستون j ام این ماتریس برابر منفی ادمیتانس موجود بین باس بار iام و j ام است.

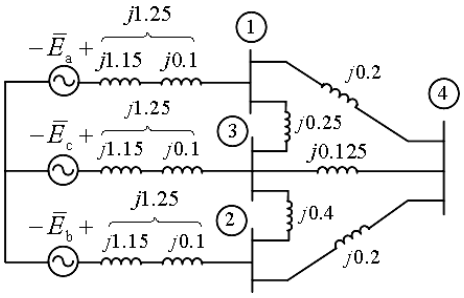
مثال ۱۹: ماتریس ادمیتانس شبکه قدرت زیر را به دو روش دستی و برنامه‌نویسی کامپیوتری به دست آورید.



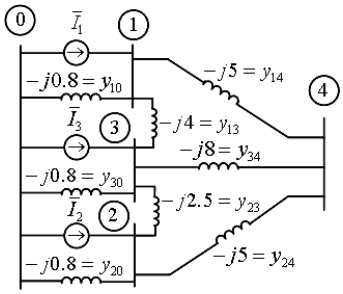
$$z_{G_a}=z_{G_b}=z_{G_c}=j1.15\ ,\ z_{T_a}=z_{T_b}=z_{T_c}=j0.1$$
$$z_{13}=j0.25\ ,\ z_{14}=j0.2\ ,\ z_{34}=j0.125$$
$$z_{23}=j0.4\ ,\ z_{24}=j0.2$$

حل: الف) به روش دستی

مدار معادل امپدانسی پریونیتی سیستم:



مدار معادل ادمیتانسی پریونیتی سیستم:



$$Y_{BUS} = \begin{bmatrix} y_{10}+y_{13}+y_{14} & 0 & -y_{13} & -y_{14} \\ 0 & y_{20}+y_{23}+y_{24} & -y_{23} & -y_{24} \\ -y_{13} & -y_{23} & y_{30}+y_{31}+y_{32}+y_{34} & -y_{34} \\ -y_{14} & -y_{24} & -y_{34} & y_{41}+y_{42}+y_{43} \end{bmatrix}$$

$$\Rightarrow Y_{BUS} = \begin{bmatrix} -j9.8 & 0 & j4 & j5 \\ 0 & -j8.3 & j2.5 & j5 \\ j4 & j2.5 & -j15.3 & j8 \\ j5 & j5 & j8 & -j18 \end{bmatrix}$$

ب) به روش برنامه‌نویسی کامپیوتری

```
ZLL = [ 1  3  0.25j
        1  4  0.2j
        3  4  0.125j
        2  3  0.4j
        2  4  0.2j
      ]                          % the impedances connected between two busbars
zb = [1.25j;1.25j;1.25j;0]       % the impedances connected to the busbars
n = length(zb)                   % number of buses
```

```
b1 = ZLL(:,1)
b2 = ZLL(:,2)
z  = ZLL(:,3)
L  = length(b1)                    %  number of lines
y = zeros(n)
for (i=1:1:L)
   i
   if (z(i) ~= 0)
      y(b1(i),b1(i))=y(b1(i),b1(i))+1/z(i)
      y(b2(i),b2(i))=y(b2(i),b2(i))+1/z(i)
      y(b1(i),b2(i))=y(b1(i),b2(i))-1/z(i)
      y(b2(i),b1(i))=y(b2(i),b1(i))-1/z(i)
   end
end
for (i=1:1:n)
   if (zb(i) ~=0)
      y(i,i)=y(i,i)+1/zb(i)
   end
end
y
```

۱۴- ساختن زیربرنامه یا function

در برنامه‌نویسی حرفه‌ای، در زمانی که ممکن است از یک قسمت برنامه چندین بار در طول برنامه استفاده شود، از ساختار زیربرنامه استفاده می‌شود. در این ساختار یک M-file جداگانه ساخته و در همان مکانی که فایل اصلی برنامه ذخیره شده است، ذخیره می‌کنیم. در این M-file جدید آن قسمت از برنامه را که قرار است چندین بار در طول برنامه اصلی تکرار شود، قرار می‌دهیم و در M-file برنامه اصلی هر کجا که به آن بخش از برنامه نیاز بود آن را فراخوانی می‌کنیم.

مثال ۲۰: می‌توان برنامه ماتریس شبکه را در یک زیربرنامه به صورت زیر قرار داد. در برنامه M-file اصلی می‌نویسیم:

```
clc
clear all
close all

ZLL = [ 1  3  0.25j
        1  4  0.2j
        3  4  0.125j
        2  3  0.4j
        2  4  0.2j
       ]                           %  the impedances connected between two busbars
zb = [1.25j;1.25j;1.25j;0]         %  the impedances connected to the busbars
```

```
y=ybusgeneration(ZLL,zb)
```

توجه: سطر آخر نحوه فراخوانی زیربرنامه‌ای به اسم ybusgeneration است. ورودی برای این زیربرنامه دو ماتریس ZLL و zb است و خروجی این زیربرنامه در متغیری به اسم y قرار داده شده است.

حال باید یک فایل M-file جدید به اسم ybusgeneration و در همان مکانی که فایل برنامه اصلی فوق ذخیره شده، آن را ایجاد و ذخیره کنیم.

نکته: اسم فایل زیربرنامه و اسمی که آن را فراخوانی می‌کنیم باید یکسان باشد. زیربرنامه ybusgeneration به صورت زیر ساخته می‌شود:

```
function y = ybusgeneration(ZLL,zb)

n = length(zb)                    %  number of buses
b1 = ZLL(:,1)
b2 = ZLL(:,2)
z  = ZLL(:,3)
L  = length(b1)                   %  number of lines
y = zeros(n)
for (i=1:1:L)
   i
   if (z(i) ~= 0)
      y(b1(i),b1(i))=y(b1(i),b1(i))+1/z(i)
      y(b2(i),b2(i))=y(b2(i),b2(i))+1/z(i)
      y(b1(i),b2(i))=y(b1(i),b2(i))-1/z(i)
      y(b2(i),b1(i))=y(b2(i),b1(i))-1/z(i)
   end
end
for (i=1:1:n)
   if (zb(i) ~=0)
      y(i,i)=y(i,i)+1/zb(i)
   end
end
```

توجه: شروع زیر برنامه‌ها با کلمه function است که به صورت اتوماتیک پس از نوشتن آن در اولین خط زیربرنامه رنگ آن آبی می‌شود. می‌توان در پایان زیربرنامه‌ها از کلمه end استفاده کرد که البته اگر آن را ننویسیم خطایی در برنامه ایجاد نمی‌شود.

نکته: عبارت پس از کلمه function در زیربرنامه درست همانند عبارتی است که در برنامه اصلی این زیربرنامه را فراخوانی می‌کنیم.

نکته: برای اجرای برنامه، کاری با زیربرنامه‌ها نداریم و تنها بایستی برنامه اصلی را اجرا کنیم. زیربرنامه‌ها در هنگام فراخوانی در فایل اصلی برنامه خود به خود اجرا می‌شوند.

تمرین ۲۱: برنامه مثال ۲۰ را در نرم‌افزار Matlab نوشته و آن را در برنامه اصلی ردیابی (trace) کنید. فرق میان آیکون‌های اجرای سفید رنگ step، step in، step out، و continue در چیست؟ تمامی آنها را امتحان کنید.

تمرین ۲۲: برنامه‌های نوشته شده در تمرین ۱۹ را به زیربرنامه تبدیل کنید. راهنمایی: ضرایب معادلات جبری را در یک بردار مانند x قرار دهید و به عنوان ورودی زیربرنامه انتخاب کنید.

تمرین ۲۳: زیربرنامه‌ای بنویسید که به عنوان ورودی یک ماتریس را گرفته و در صورتی که آن ماتریس مربعی باشد دترمینان ماتریس، معکوس ماتریس و مقادیر ویژه آن را محاسبه کرده و به عنوان خروجی به برنامه اصلی باز گرداند. در صورتی که ماتریس مربعی نباشد این محاسبات را انجام ندهد و تنها ابعاد ماتریس را باز گرداند.

۱۵- حل مسأله پخش بار (Load Flow) به روش گوس - سایدل

برای حل مسأله پخش بار در یک سیستم قدرت باید ابتدا سیستم را به دستگاه پریونیت برده و سپس ماتریس ادمیتانس شبکه را تشکیل داد. در ادامه باید نوع باس بارها ($slack$ ، $P-Q$ ، $P-|V|$) را تشخیص داده و جدول مقادیر اولیه ولتاژها را تنظیم کنیم. برای باس بارهای $P-Q$ از رابطه زیر مقدار جدید فازور ولتاژ را به دست آورده و هم اندازه و هم فاز آن را وارد جدول می‌کنیم:

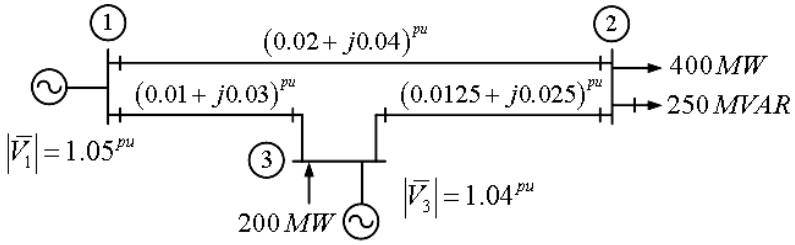
$$\overline{V}_{\mathrm{i}}=\frac{1}{Y_{\mathrm{ii}}}\left[\frac{P_{\mathrm{i}}-jQ_{\mathrm{i}}}{\overline{V}_{\mathrm{i}}^{*}}-\sum_{\substack{\mathrm{k}=1\\ \mathrm{k}\neq\mathrm{i}}}^{\mathrm{N}}Y_{\mathrm{ik}}\overline{V}_{\mathrm{k}}\right]$$

در این رابطه توان‌های اکتیو و راکتیو باس بارها، توان‌های اکتیو و راکتیو خالص باس بار (تولید منهای مصرف) هستند. برای باس بارهای $P-|V|$ ابتدا از رابطه زیر مقدار توان راکتیو (Q) جدید را پیدا کرده و سپس از رابطه فوق مقدار جدید فازور ولتاژ را به دست آورده و تنها فاز آن را وارد جدول می‌کنیم:

$$Q_{\mathrm{i}}=-\mathrm{Im}\left[\overline{V}_{\mathrm{i}}^{*}\left(\sum_{\mathrm{k}=1}^{\mathrm{N}}Y_{\mathrm{ik}}\overline{V}_{\mathrm{k}}\right)\right]$$

اگر نتیجه محاسبات با مرحله قبل با تقریب معینی برابر باشد، به جواب رسیده‌ایم، در غیر این صورت محاسبات با استفاده از جدول جدید از اول تکرار می‌گردد.

مثال ۲۱: در این مثال می‌خواهیم مسأله پخش بار سیستم قدرت نمونه زیر را به روش گوس – سایدل در نرم‌افزار Matlab برنامه‌نویسی کنیم. در این شبکه توان نامی مبنا 100^{MVA} انتخاب شده است:



محاسبه مقادیر توان اکتیو و راکتیو خالص باس بارها و ماتریس ادمیتانس شبکه به روش دستی:

$$\bar{S}_2^{\text{pu}} = P_2^{\text{pu}} + jQ_2^{\text{pu}} = 0 - \frac{400 + j250}{100} = (-4 - j2.5)^{pu} \;\; , \;\; P_3^{\text{pu}} = \frac{200}{100} = 2^{pu}$$

$$y_{12} = \frac{1}{0.02 + j0.04} = 10 - j20 \;\; , \;\; y_{13} = \frac{1}{0.01 + j0.03} = 10 - j30 \;\; , \;\; y_{23} = \frac{1}{0.0125 + j0.025} = 16 - j32$$

$$\Rightarrow Y_{\text{BUS}} = \begin{bmatrix} 20 - j50 & -10 + j20 & -10 + j30 \\ -10 + j20 & 26 - j52 & -16 + j32 \\ -10 + j30 & -16 + j32 & 26 - j62 \end{bmatrix}$$

با توجه به اطلاعات داده شده، باس بار 1 از نوع *slack*، باس بار 2 از نوع $P-Q$ و باس بار 3 از نوع $P-|V|$ می‌باشد. جدول مقادیر اولیه ولتاژها به صورت زیر است:

i		$\lvert\bar{V}_i\rvert$	δ_i
slack	1	1.05^{pu}	$\delta_1 = 0^\circ$
$P-Q$	2	1^{pu}	$\delta_2 = 0^\circ$
$P-\lvert V\rvert$	3	1.04^{pu}	$\delta_3 = 0^\circ$

مقادیری که دور آنها خط کشیده شده مقادیر حدس‌های اولیه می‌باشند که در روش گوس – سایدل طبق روابط ذکر شده، در هر مرحله بازسازی جدول باید به روز رسانی شوند. به صورت دستی داریم:

$$\Rightarrow \bar{V}_2^{\text{new}_1} = \frac{1}{Y_{22}}\left[\frac{P_2 - jQ_2}{\bar{V}_2^{\,*}} - \left(Y_{21}\bar{V}_1 + Y_{23}\bar{V}_3\right)\right]$$

$$= \frac{1}{26 - j52}\left[\frac{-4 - j(-2.5)}{1\angle 0^\circ} - \left((-10 + j20)(1.05\angle 0^\circ) + (-16 + j32)(1.04\angle 0^\circ)\right)\right]$$

$$= \underbrace{0.97553^{pu}} \;\; \angle \;\; \underbrace{-2.486^\circ}$$

وارد جدول وارد جدول

$$Q_3^{\text{pu}} = -\text{Im}\left[\bar{V}_3^{*}\left(Y_{31}\bar{V}_1 + Y_{32}\bar{V}_2 + Y_{33}\bar{V}_3\right)\right] =$$

$$-\text{Im}\left\{1.04\angle 0^{\circ}\left[(-10+j30)(1.05\angle 0^{\circ}) + (-16+j32)(0.97553\angle -2.486^{\circ}) + (26-j62)(1.04\angle 0^{\circ})\right]\right\}$$

$$= 1.16^{pu}$$

$$\Rightarrow \bar{V}_3^{\text{new}_1} = \frac{1}{Y_{33}}\left[\frac{P_3 - jQ_3}{\bar{V}_3^{*}} - \left(Y_{31}\bar{V}_1 + Y_{32}\bar{V}_2\right)\right]$$

$$= \frac{1}{26-j62}\left[\frac{2-j1.16}{1.04\angle 0^{\circ}} - \left((-10+j30)(1.05\angle 0^{\circ}) + (-16+j32)(0.97553\angle -2.486^{\circ}\right)\right]$$

$$= 1.0378^{pu}\angle \underbrace{-0.2854^{\circ}}$$

وارد جدول

اگر دقت مورد نظر $5\times10^{-5}\ pu$ باشد، پس از هفت بار تکرار محاسبات فوق (هر بار با مقادیر جدید جدول) به نتایج زیر می‌رسیم:

$$\bar{V}_2^{\text{final}} = 0.97168^{pu}\angle -2.6951^{\circ}\ ,\ \bar{V}_3^{\text{final}} = 1.04^{pu}\angle -0.4975^{\circ}\ ,\ Q_3^{\text{final}} = 1.4617^{pu}$$

برنامه کامپیوتری روش گوس – سایدل برای مثال فوق به قرار زیر است:

```
ZLL = [1 2 0.02+0.04j
       1 3 0.01+0.03j
       2 3 0.0125+0.025j
      ];                    % the impedances connected between two busbars
zb = [0;0;0];               % the impedances connected to the busbar
n = length(zb);             % number of buses
v = [ 1.05 ; 1 ; 1.04 ];    % busbar magnitude voltage matrix , busbar number 1 = slack
delta = zeros(n,1);         % busbar angle voltage matrix
Pg = [ 0 ; 2 ];             % busbar gerarating active power matrix
Qg = [ 0 ];                 % busbar gerarating reactive power matrix
Pc = [ 4 ; 0 ];             % busbar consuming active power matrix
Qc = [ 2.5 ];               % busbar consuming reactive power matrix
Pnet = Pg-Pc;
Qnet = Qg-Qc;
n1=length(Qnet);            % n1 = number of P-Q buses
n2=n-n1-1;                  % n2 = number of P-|v| buses
%********************* YBUS GENERATION *****************
b1 = ZLL(:,1);
b2 = ZLL(:,2);
z  = ZLL(:,3);
L  = length(b1);            % number of lines
y = zeros(n);
for (i=1:1:L)
   if (z(i) ~= 0)
      y(b1(i),b1(i))=y(b1(i),b1(i))+1/z(i);
      y(b2(i),b2(i))=y(b2(i),b2(i))+1/z(i);
      y(b1(i),b2(i))=y(b1(i),b2(i))-1/z(i);
```

```
        y(b2(i),b1(i))=y(b2(i),b1(i))-1/z(i);
    end
end
for (i=1:1:n)
    if (zb(i) ~=0)
        y(i,i)=y(i,i)+1/zb(i);
    end
end
%********************* MAIN LOOP ********************
emax=1;
accuracy = 1e-5;                    % accuracy
step=0;
while (emax > accuracy & step < 100)
    m=1;
    vold=v;
    for (i=2:1:n)
        if (Pnet(i-1) <= 0)
            sigma=0;
            for(k=1:1:n)
                if (k ~= i)
                    sigma=sigma+y(i,k)*v(k);
                end
            end
            v(i)=(1/y(i,i))*((Pnet(i-1)-j*Qnet(i-1))/conj(v(i))-sigma);
            abs(v);
            angle(v)*180/pi;
        else
            sigma=0;
            for (k=1:1:n)
                sigma=sigma+y(i,k)*v(k);
            end
            Q(m)=-imag(conj(v(i))*sigma);
            sigma=0;
            for(k=1:1:n)
                if (k ~= i)
                    sigma=sigma+y(i,k)*v(k);
                end
            end
            v(i)=(1/y(i,i))*((Pnet(i-1)-j*Q(m))/conj(v(i))-sigma);
            abs(v);
            angle(v)*180/pi;
            v(i)=vold(i)*(cos(angle(v(i)))+j*sin(angle(v(i))));
            abs(v);
            angle(v)*180/pi;
            m=m+1;
        end
```

```
    end
    deltav=abs(abs(vold)-abs(v));
    emax=max(deltav);
    step=step+1;
end
step
abs(v)
angle(v)*180/pi
```