پروژه اول درس تحلیل سیستم‌های انرژی

امیرحسین زابلی(۹۹۲۶۸۴۳)

محمدسروش رضایی(۹۹۲۶۵۶۳)

امیر جهانگرد(۹۹۲۵۲۹۳)

تعیین روش محاسبه

ابتدا روش حل توسط کاربر تعیین می‌شود. کد مربوط به این بخش در فایل choose_the_method نوشته شده است. روش کار به این صورت است که مقداری از ۱ تا ۳ از کاربر گرفته می‌شود. عدد ۱ مربوط به روش گوس-سایدل، عدد ۲ مربوط به روش نیوتن-رفسون و نهایتا عدد ۳ مربوط به روش مجزای سریع می‌باشد. برنامه با دریافت عددی خارج از این محدوده به کاربر خطا نشان می‌دهد و منتظر دریافت مجدد ورودی می‌شود:

```
branch_and_bus_data
Ybus

method = input('Enter 1 for Gauss-Seidel, 2 for Newton-Raphson, 3 for Fast
Decoupled: ');
while method ~= 1 && method ~= 2 && method ~= 3
    fprintf('Invalid Input\n');
    method = input('Enter 1 for Gauss-Seidel, 2 for Newton-Raphson, 3 for
Fast Decoupled: ');
end

if method == 1
    maingauss
elseif method == 2
    mainnewton
else
    maindecouple
end
```

*از آنجا که در هر زیر برنامه نیاز به ماتریس‌های دو زیر برنامه branch_and_bus_data و ybus داریم، آن‌ها را همان ابتدا وارد می‌کنیم. (این ماتریس‌ها در ادامه توضیح داده می‌شوند.)

داده‌های شینه‌ها و خطوط

در فایل جداگانه‌ای با اسم branch_and_bus_data.m اطلاعات داده شده در صورت سوال را در دو ماتریس جداگانه وارد می‌کنیم:

```
% bus data
%      ----bus----    Voltage   Angle     -Load-      -------Generator-------
%       bus_i  type    Mag.      Degree    Pd    Qd     PG    QG    Min Q  Max Q
bus = [
        1      1      1.04         0       0     0      72.3  0    -300     300 ;
        2      2      1.025        0       0     0      163   0    -300     300 ;
        3      2      1.025        0       0     0      85    0    -300     300 ;
        4      0      1            0       0     0      0     0     0       0   ;
        5      0      1            0       90    30     0     0     0       0   ;
        6      0      1            0       0     0      0     0     0       0   ;
        7      0      1            0       100   35     0     0     0       0   ;
        8      0      1            0       0     0      0     0     0       0   ;
        9      0      1            0       125   50     0     0     0       0
      ];


% branch data
%          fbus     tbus        RL(pu)     XL(pu)
branch = [
            1        4          0          0.0576 ;
            4        5          0.017      0.092  ;
            5        6          0.039      0.17   ;
            3        6          0          0.0586 ;
            6        7          0.0119     0.1008 ;
            7        8          0.0085     0.072  ;
            8        2          0          0.0625 ;
            8        9          0.032      0.161  ;
            9        4          0.01       0.085
         ];
```

همانطور که مشاهده می‌شود، ماتریس bus به تعداد شینه‌ها سطر دارد که این تعداد برابر ۹ است. علاوه بر این، شامل ۱۰ ستون نیز می‌باشد که مقادیر موجود در آن، بالای همان ستون نوشته شده:

۱- bus_i: شماره هر یک از شینه‌هاست. از آنجا که تعداد شینه‌ها برابر ۹ است، این ستون اعداد ۱ تا ۹ را شامل می‌شود.

۲- type: شین مرجع با عدد ۱، شین تنظیم شده با عدد ۲ و شین بار با عدد صفر نشان داده شده است.

۳-Mag: اندازه ولتاژ هر شین بر حسب پریونیت و با مبنای 345KV در آن قرار داده شده.برای شین‌هایی که مقادیر آن‌ها داده نشده، مقدار 1 را قرار می‌دهیم.

۴-Angle Degree: فاز ولتاژ شینه‌ها در این ستون قرار دارد.فاز شین‌هایی که مقدار آن‌ها در صورت سوال تعیین نشده را برابر صفر قرار می‌دهیم.

۵- Pd و Qd: توان اکتیو(یکای MW) و راکتیو(یکای MVAR) شینه‌ها در این ستون قرار دارند.
٦- چهار ستون انتهایی مربوط به شین‌های ژنراتوری می‌باشند. برای باس‌های بار تمام این مقادیر را برابر صفر قرار می‌دهیم. همچنین QG (توان راکتیو) که برای هیچکدام از شینه‌ها داده نشده را مساوی صفر قرار می‌دهیم.

ماتریس branch اطلاعات مربوط به خطوط را در خود نگه می‌دارد. این ماتریس به تعداد خطوط به شینه‌ها دارای سطر بوده و همچنین، ۴ ستون دارد:
۱- fbus و tbus: ستون اول مربوط به شماره شینه‌ای است که خط از آن شروع می‌شود. ستون دوم نیز شماره شینه‌ای است که خط به آن وارد می‌شود.
۲- RL و XL: به ترتیب نشان دهنده مقاومت و راکتانس هر خط می‌باشند.

ماتریس ادمیتانس خطوط

برای محاسبه ماتریس ادمیتانس خطوط از ماتریس branch استفاده می‌کنیم. از دو ستون آخر این ماتریس مقدار امپدانس خطوط را ذخیره کرده و نهایتا مقادیر ادمیتانس ($Y = \frac{1}{Z}$) را محاسبه می‌کنیم. ماتریس ادمیتانس به تعداد سطرهای ماتریس branch که همان تعداد خطوط است، سطر دارد و تعداد ستون‌های نیز برابر یک است. بنابراین ابتدا یک ماتریس با چنین اندازه‌ای را با دستور ones (همه مقادیر ماتریس برابر یک خواهند بود) می‌سازیم. بعد از ذخیره مقادیر امپدانس در یک ماتریس Z، ماتریس یک را بر ماتریس Z تقسیم می‌کنیم تا مقادیر ادمیتانس خطوط محاسبه شوند. در ادامه باید ماتریس ادمیتانس را که یک ماتریس مربعی که تعداد سطر و ستون‌های آن برابر با تعداد شینه‌های سیستم است، پیدا کنیم. متغیر nbr تعداد شینه‌ها را در خود دارد. ماتریسی nbr*nbr ایجاد می‌کنیم و با حلقه مقدار هر درایه را تعیین می‌کنیم. از آنجا که درایه‌هایی که روی قطر اصلی این ماتریس نیستند برابر با قرینه ادمیتانس بین دو شاخه می باشند، کافیست ادمیتانس خطوط را که در مرحله قبل به دست آوردیم، از درایه متناظر با آن شاخه در ماتریس ادمیتانس کم کنیم. علاوه بر این، از آنجا که ماتریس ادمیتانس نسبت به قطر اصلی آن دارای تقارن است، بنابراین مقدار هر درایه‌ای را که محاسبه کردیم، در درایه قرینه شده نسبت به قطر اصلی نیز قرار می‌دهیم.برای محاسبه درایه‌های قطر اصلی، هر ادمیتانس که یکی از دو سر آن، شین مدنظر ما است را با درایه متناظر آن شین در ماتریس Y جمع می‌کنیم. کد مربوط به این بخش به صورت زیر است:

```
nl = branch (:,1);
nr = branch(:,2);
R = branch(:,3);
X = branch(:,4);
```

```
nbr = length( branch(:,1)); % number of branches
nbus =  max(max(nl), max(nr));
Z = R + 1j * X ; % branch impedance

y = ones(nbr, 1)./Z; % branch admittance
Y = zeros(nbus, nbus); % initialize Y to zero
for k = 1 : nbr % formation of the off diagonal elements
    if nl(k) > 0 && nr(k) > 0
        Y(nl(k), nr(k)) = Y(nl(k), nr(k)) - y( k );
        Y(nr(k), nl(k)) = Y(nl(k), nr(k));
    end
end
for n = 1 : nbus % formation of the diagonal elements
    for k = 1 : nbr
        if nl(k) == n || nr(k) == n
            Y(n, n) = Y(n, n) + y(k);
        else
        end
    end
end
```

مقادیر این ماتریس به صورت زیر است:

```
Columns 1 through 8

 0.0000 -17.3611i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 +17.3611i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
 0.0000 + 0.0000i   0.0000 -16.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 +16.0000i
 0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 -17.0648i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 +17.0648i   0.0000 + 0.0000i   0.0000 + 0.0000i
 0.0000 +17.3611i   0.0000 + 0.0000i   0.0000 + 0.0000i   3.3074 -39.4759i  -1.9422 +10.5107i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
 0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i  -1.9422 +10.5107i   3.2242 -16.0989i  -1.2820 + 5.5882i   0.0000 + 0.0000i   0.0000 + 0.0000i
 0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 +17.0648i   0.0000 + 0.0000i  -1.2820 + 5.5882i   2.4371 -32.4374i  -1.1551 + 9.7843i   0.0000 + 0.0000i
 0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i  -1.1551 + 9.7843i   2.7722 -23.4822i  -1.6171 +13.6980i
 0.0000 + 0.0000i   0.0000 +16.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i  -1.6171 +13.6980i   2.8047 -35.6731i
 0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i  -1.3652 +11.6041i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i  -1.1876 + 5.9751i

Column 9

 0.0000 + 0.0000i
 0.0000 + 0.0000i
 0.0000 + 0.0000i
-1.3652 +11.6041i
 0.0000 + 0.0000i
 0.0000 + 0.0000i
 0.0000 + 0.0000i
-1.1876 + 5.9751i
 2.5528 -17.5792i
```

خروجی شین‌ها و خطوط

برای نمایش خروجی شین‌ها (اندازه و فاز ولتاژ، توان‌های اکتیو و راکتیو و ...) فایلی را با نام busout.m ایجاد می‌کنیم. هنگام نمایش خروجی، ابتدا دقت جواب و تعداد تکرارهای محاسبات را (بسته به نوع محاسباتی که کاربر در ابتدا انتخاب کرده) را نمایش می‌دهیم. این مقادیر به ترتیب در متغیرهای maxerror و iter ذخیره می‌شوند. در ادامه با نوشتن یک حلقه، مقادیر شماره شین، اندازه ولتاژ، فاز ولتاژ، توان اکتیو بار، توان راکتیو بار، توان اکتیو ژنراتوری و توان راکتیو ژانراتوری مربوط به هر شین را نمایش می‌دهیم. کد این قسمت به صورت زیر است:

```
disp(tech)
```

```
fprintf('                      Maximum Power Mismatch = %g \n', maxerror)
fprintf('                            No. of Iterations = %g \n\n', iter)
head =['    Bus  Voltage  Angle    ------Load------    ---Generation---'
       '    No.  Mag.     Degree     MW       Mvar       MW       Mvar '
       '                                                                '];
disp(head)
for n=1:nbus
     fprintf(' %5g', n)
     fprintf(' %7.3f', Vm(n))
     fprintf(' %8.3f', deltad(n))
     fprintf(' %9.3f', Pd(n))
     fprintf(' %9.3f', Qd(n))
     fprintf(' %9.3f', Pg(n))
     fprintf(' %9.3f\n', Qg(n))
end
    fprintf('      \n')
    fprintf('    Total              ')
    fprintf(' %9.3f', Pdt)
    fprintf(' %9.3f', Qdt)
    fprintf(' %9.3f', Pgt)
    fprintf(' %9.3f', Qgt)
```

برای نمایش خروجی خطوط، فایلی با نام lineflow.m ایجاد می‌کنیم. توان‌های اکتیو و راکتیو هر شین و پخش توان‌ها و تلفات اکتیو و راکتیو خطوط را با استفاده از دو حلقه به دست می‌آوریم. در حلقه بیرونی، یک شین انتخاب می‌شود و توان‌ها و پخش‌توان آن شین محاسبه می‌شود. در حلقه درونی، توان اتلافی بین خطوط و پخش توان آن شین به شین‌های دیگر را مشخص می‌کنیم. نهایتا جمع توان‌های اکتیو و راکتیو را نمایش می‌دهیم. کد مربوط به این قسمت در زیر قابل رویت است:

```
SLT = 0;
fprintf('\n\n\n')
fprintf('                            Line Flow and Losses \n')
fprintf('     --Line--  Power at bus & line flow    --Line loss--\n')
fprintf('     from  to    MW      Mvar     MVA       MW      Mvar\n')

for n = 1:nbus
busprt = 0;
    for L = 1:nbr
        if busprt == 0
        fprintf('   \n')
        fprintf('%6g', n)
        fprintf('      %9.3f', P(n)*basemva)
        fprintf('%9.3f', Q(n)*basemva)
        fprintf('%9.3f\n', abs(S(n)*basemva))

        busprt = 1;
        else
```

```
        end
        if nl(L)==n
            k = nr(L);
            In = (V(n) - V(k))*y(L);
            Ik = (V(k) - V(n))*y(L);
            Snk = V(n)*conj(In)*basemva;
            Skn = V(k)*conj(Ik)*basemva;
            SL  = Snk + Skn;
            SLT = SLT + SL;
        elseif nr(L) == n
            k = nl(L);
            In = (V(n) - V(k))*y(L);
            Ik = (V(k) - V(n))*y(L);
            Snk = V(n)*conj(In)*basemva;
            Skn = V(k)*conj(Ik)*basemva;
            SL  = Snk + Skn;
            SLT = SLT + SL;
            else
        end
        if nl(L)== n || nr(L)== n
            fprintf('%12g', k)
            fprintf('%9.3f', real(Snk))
            fprintf('%9.3f', imag(Snk))
            fprintf('%9.3f', abs(Snk))
            fprintf('%9.3f', real(SL))
        if nl(L) ==n
            fprintf('%9.3f\n', imag(SL))
        else
            fprintf('%9.3f\n', imag(SL))
        end
        else
        end
    end
end
SLT = SLT/2;
fprintf('   \n')
fprintf('    Total loss                          ')
fprintf('%9.3f', real(SLT))
fprintf('%9.3f\n', imag(SLT))
clear Ik In SL SLT Skn Snk
```

در ادامه به سه الگوریتم حل می‌پردازیم. برای هر یک از این روش‌ها، دو فایل ایجاد می‌کنیم؛ در یک فایل اطلاعات کلی روش مربوطه و در دیگر الگوریتم پیاده‌سازی را کدنویسی می‌کنیم:

روش گوس-سایدل

در دو فایل mainguass.m و ifgauss.m پیاده‌سازی شده:

فایل maingauss: در این فایل چهار پارامتر مورد نیاز برای استفاده از این روش تعریف شده است: مبنای سیستم (یکای MVA)، دقت عدم تطابق توان، ضریب تسریع و حداکثر تعداد تکرارها. این مقادیر به ترتیب در متغیرهای basemva، accuracy، accel، maxiter ذخیره شده‌اند. مطابق با صورت سوال، مقدار basemva را برابر 100 قرار می‌دهیم. برای ضریب تسریع نیز مقداری دلخواه در نظر می‌گیریم: 1.3. همچنین تعداد تکرارها را برابر 100 در نظر می‌گیریم. برای دقت عدم تطابق نیز مقدار 5.10^{-5} در نظر گرفته شده. (دقت تا چهار رقم اعشار.) برای محاسبه پیچیدگی روش هم از دو دستور tic و toc استفاده می‌کنیم تا مدت زمانی که انجام محاسبات طول می‌کشد را بیابیم. بنابراین دستور tic را قبل از ifgauss و دستور toc را بعد از آن قرار می‌دهیم. همچنین در انتهای کد، فایل‌های busout و lineflow را برای نمایش خروجی فراخوانی می‌کنیم:

```
basemva = 100;
accuracy = 0.00005;
accel = 1.3;
maxiter = 100;

tic
ifgauss
toc
busout
lineflow
```

فایل ifgauss.m: در این فایل محاسبات اصلی روش گوس-سایدل را پیاده‌سازی می‌کنیم. در این روش داریم:

$$v_i^{(k+1)} = \frac{\frac{P_i^{sch} - jQ_i^{sch}}{V_i^{*(k)}} - \sum_{j \neq i} Y_{ij} V_j^{(k)}}{Y_{ii}}$$

$$P_i^{(k+1)} = Re\{V_i^{*(k)}\left[V_i^{(k)} Y_{ii} + \sum_{j \neq i} Y_{ij} V_j^{(k)}\right]\}$$

$$Q_i^{(k+1)} = -I\{V_i^{*(k)}\left[V_i^{(k)} Y_{ii} + \sum_{j \neq i} Y_{ij} V_j^{(k)}\right]\}$$

در این روش برای ولتاژهای مجهول مقدار 1.0 + j0.0 در نظر گرفته می‌شود. در شین‌های بار که توان‌های اکتیو و راکتیو P_i^{sch} و Q_i^{sch} معلوم هستند، با تخمین اولیه مذکور و با کمک معادله اول می‌توان ولتاژ این شین‌ها را تعیین کرد.

در شین‌های تنظیم شده که P_i^{sch} و $|V_i|$ معلوم هستند، ابتدا به کمک سومین معادله مقدار Q_i^{sch} را به دست آورده و سپس به کمک معادله اول، مقدار $V_i^{(k+1)}$ را محاسبه می‌کنیم. از طرفی با معلوم بودن $|V_i|$ در این نوع شین، تنها بخش موهومی $V_i^{(k+1)}$ ذخیره می‌شود و برای بخش حقیقی آن داریم:

$$e_i^{(k+1)} = \sqrt{|V_i|^2 - \left(f_i^{(k+1)^2}\right)}$$

در معادله بالا، خروجی بخش حقیقی $V_i^{(k+1)}$ است و $f_i^{(k+1)^2}$ بخش موهومی آن می‌باشد. بنابراین، ولتاژ شین‌ها را از رابطه زیر حساب می‌کنیم که در آن آلفا همان ضریب تسریع است:

$$V_i^{(k+1)} = V_i^{(k)} + \alpha\left(V_{cal}^{(k)} - V_i^{(k)}\right)$$

برای استفاده از روش گوس-سایدل در متلب لازم است تا داده‌های ماتریس اطلاعات شین‌ها که در ماتریس bus هستند را بازیابی کنیم. بنابراین از یک حلقه استفاده می‌کنیم. لازم به ذکر است که در هر مرحله این حلقه، زاویه شین‌ها را بر حسب رادیان محاسبه می‌کنیم و سپس فازور ولتاژ را به کمک اندازه و زاویه به دست می‌آوریم. از سوی دیگر، توان‌های اکتیو و راکتیو هر شین را که از ماتریس bus بازیابی می‌شود را به کمک مقدار مبنا برحسب پریونیت می‌نویسیم و بنابراین می‌توانیم ماتریس توان‌های مختلط را تشکیل دهیم. در نهایت، روش گوس-سایدل را با استفاده از یک حلقه پیاده‌سازی می‌کنیم و آن را به دفعات مدنظرمان تکرار می‌کنیم. در صورت همگرا نشدن پاسخ، واگرایی را به کاربر اطلاع می‌دهیم. همچنین، تعداد دفعات انجام فرآیند تا رسیدن به دقت مدنظر را ذخیره می‌کنیم.در نهایت توان‌های محاسبه شده را از حالت پریونیت به مقدار واقعی و زوایا را به درجه برمی‌گردانیم تا توان‌های تلفات خط و ... را محاسبه کنیم. کد مربوط به این بخش به صورت زیر است:

```
Vm = 0;
delta = 0;
yload = 0;
deltad = 0;
nbus = length(bus(:,1));
```

```
for k = 1:nbus
    n = bus(k,1);
    kb(n) = bus(k,2);
    Vm(n) = bus(k,3);
    delta(n) = bus(k,4);
    Pd(n) = bus(k,5);
    Qd(n) = bus(k,6);
    Pg(n) = bus(k,7);
    Qg(n) = bus(k,8);
    Qmin(n) = bus(k,9);
    Qmax(n) = bus(k,10);
    if Vm(n) <= 0
        Vm(n) = 1.0;
        V(n) = 1 + 1j*0;
    else
        delta(n) = delta(n)*pi/180;
        V(n) = Vm(n)*(cos(delta(n)) + 1j*sin(delta(n)));
        P(n) = (Pg(n)-Pd(n))/basemva;
        Q(n) = (Qg(n)-Qd(n))/basemva;
        S(n) = P(n) + 1j*Q(n);
    end
    DV(n) = 0;
end
num = 0;
AcurBus = 0;
converge = 1;
Vc = zeros(nbus,1)+1j*zeros(nbus,1);
Sc = zeros(nbus,1)+1j*zeros(nbus,1);

iter = 0;
maxerror = 10;
while maxerror >= accuracy && iter <= maxiter
    iter = iter+1;
    for n = 1:nbus
        YV = 0+1j*0;
    for L = 1:nbr
        if nl(L) == n
            k = nr(L);
            YV = YV + Y(n,k)*V(k);
        elseif nr(L) == n
            k = nl(L);
            YV = YV + Y(n,k)*V(k);
        end
    end
    Sc = conj(V(n))*(Y(n,n)*V(n) + YV) ;
    Sc = conj(Sc);
    DP(n) = P(n) - real(Sc);
    DQ(n) = Q(n) - imag(Sc);
    if kb(n) == 1
```

```
        S(n) = Sc;
        P(n) = real(Sc);
        Q(n) = imag(Sc);
        DP(n) = 0;
        DQ(n) = 0;
        Vc(n) = V(n);
    elseif kb(n) == 2
        Q(n) = imag(Sc);
        S(n) = P(n) + 1j*Q(n);
        if Qmax(n) ~= 0
            Qgc = Q(n)*basemva + Qd(n);
                if abs(DQ(n)) <= 0.005 && iter >= 10
                    if DV(n) <= 0.045
                        if Qgc < Qmin(n)
                            Vm(n) = Vm(n) + 0.005;
                            DV(n) = DV(n)+0.005;
                        elseif Qgc > Qmax(n)
                            Vm(n) = Vm(n) - 0.005;
                            DV(n) = DV(n)+0.005;
                        end
                    else
                    end
                else
                end
        else
        end
    end
    if kb(n) ~= 1
        Vc(n) = (conj(S(n))/conj(V(n)) - YV )/ Y(n,n);
    else
    end
    if kb(n) == 0
        V(n) = V(n) + accel*(Vc(n)-V(n));
    elseif kb(n) == 2
        VcI = imag(Vc(n));
        VcR = sqrt(Vm(n)^2 - VcI^2);
        Vc(n) = VcR + 1j*VcI;
        V(n) = V(n) + accel*(Vc(n) -V(n));
    end
    end
    maxerror = max( max(abs(real(DP))), max(abs(imag(DQ))) );
    if iter == maxiter && maxerror > accuracy
        fprintf('\nWARNING: Iterative solution did not converged after ')
        fprintf('%g', iter), fprintf(' iterations.\n\n')
        fprintf('Press Enter to terminate the iterations and print the
results \n')
        converge = 0;
        pause
    else
    end
```

```
end
if converge ~= 1
   tech= ('                        ITERATIVE SOLUTION DID NOT CONVERGE');
else
   tech=('                     Power Flow Solution by Gauss-Seidel Method');
end
k=0;
for n = 1:nbus
     Vm(n) = abs(V(n));
     deltad(n) = angle(V(n))*180/pi;
     if kb(n) == 1
        S(n) = P(n)+1j*Q(n);
        Pg(n) = P(n)*basemva + Pd(n);
        Qg(n) = Q(n)*basemva + Qd(n);
        k = k+1;
        Pgg(k) = Pg(n);
     elseif  kb(n) == 2
        k = k+1;
        Pgg(k) = Pg(n);
        S(n) = P(n)+1j*Q(n);
        Qg(n) = Q(n)*basemva + Qd(n);
     end
     yload(n) = (Pd(n)- 1j*Qd(n))/(basemva*Vm(n)^2);
end
Pgt = sum(Pg);
Qgt = sum(Qg);
Pdt = sum(Pd);
Qdt = sum(Qd);
bus(:,3) = Vm';
bus(:,4) = deltad';
clear  AcurBus  DP  DQ  DV  L Sc Vc VcI VcR YV converge delta
```

خـروجی روش گـوس-سـایدل: تـصاویر زیر بـه تـرتیب خـروجی‌هـای busout و lineflow فـرآیند را نـشان می‌دهند:

Maximum Power Mismatch = 4.95821e-05
No. of Iterations = 48

Bus No.	Voltage Mag.	Angle Degree	------Load------ MW	Mvar	---Generation--- MW	Mvar
1	1.040	0.000	0.000	0.000	72.349	81.367
2	1.025	9.692	0.000	0.000	163.000	53.411
3	1.025	4.880	0.000	0.000	85.000	38.554
4	0.996	-2.306	0.000	0.000	0.000	0.000
5	0.966	-3.738	90.000	30.000	0.000	0.000
6	1.004	2.107	0.000	0.000	0.000	0.000
7	0.979	0.836	100.000	35.000	0.000	0.000
8	0.997	3.973	0.000	0.000	0.000	0.000
9	0.951	-4.139	125.000	50.000	0.000	0.000
Total			315.000	115.000	320.349	173.332

Line Flow and Losses

--Line-- from	to	Power at bus & line flow MW	Mvar	MVA	--Line loss-- MW	Mvar
1		72.349	81.367	108.880		
	4	72.348	81.367	108.880	-0.000	6.313
2		163.000	53.411	171.528		
	8	162.998	53.411	171.526	0.000	17.502
3		85.000	38.554	93.335		
	6	84.997	38.554	93.333	0.000	4.859
4		0.000	0.000	0.000		
	1	-72.348	-75.054	104.246	-0.000	6.313
	5	31.029	26.554	40.840	0.286	1.548
	9	41.317	48.500	63.713	0.409	3.480
5		-90.000	-30.000	94.868		
	4	-30.743	-25.006	39.629	0.286	1.548
	6	-59.257	-4.993	59.467	1.477	6.440
6		0.000	0.000	0.000		
	5	60.735	11.433	61.801	1.477	6.440
	3	-84.997	-33.695	91.433	0.000	4.859
	7	24.262	22.262	32.928	0.128	1.084
7		-100.000	-35.000	105.948		
	6	-24.134	-21.178	32.109	0.128	1.084
	8	-75.867	-13.821	77.116	0.527	4.466
8		0.000	0.000	0.000		
	7	76.394	18.287	78.553	0.527	4.466
	2	-162.998	-35.909	166.906	0.000	17.502
	9	86.605	17.622	88.379	2.513	12.641
9		-125.000	-50.000	134.629		
	8	-84.092	-4.980	84.239	2.513	12.641
	4	-40.908	-45.020	60.829	0.409	3.480
Total loss					5.340	58.332

روش نیوتن-رفسون

فایل mainnewton.m: مشابه فایل maingauss.m می‌باشد با این تفاوت که در این روش ضریب تسریع نداریم و همچنین maxiter را برابر 100 در نظر می‌گیریم.

فایل ifnewton.m: در این روش داریم:

$$P_i = \sum_{j=1}^{n} |V_i||V_j||Y_{ij}|\cos\left(\theta_{ij} - \delta_i + \delta_j\right)$$

$$Q_i = -\sum_{j=1}^{n} |V_i||V_j||Y_{ij}|\sin\left(\theta_{ij} - \delta_i + \delta_j\right)$$

مشابه قبل، زوایا بر حسب پریونیت بوده و زاویه فازها بر حسب رادیان می‌باشد. همچنین در این روش ماتریسی با نام ژاکوبین نیز تعریف می‌شود:

$$\begin{bmatrix} \Delta P_2^{(k)} \\ \vdots \\ \Delta P_n^{(k)} \\ \hline \Delta Q_2^{(k)} \\ \vdots \\ \Delta Q_n^{(k)} \end{bmatrix} = \left[\begin{array}{ccc|ccc} \frac{\partial P_2}{\partial \delta_2}^{(k)} & \cdots & \frac{\partial P_2}{\partial \delta_n}^{(k)} & \frac{\partial P_2}{\partial |V_2|}^{(k)} & \cdots & \frac{\partial P_2}{\partial |V_n|}^{(k)} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial P_n}{\partial \delta_2}^{(k)} & \cdots & \frac{\partial P_n}{\partial \delta_n}^{(k)} & \frac{\partial P_n}{\partial |V_2|}^{(k)} & \cdots & \frac{\partial P_n}{\partial |V_n|}^{(k)} \\ \hline \frac{\partial Q_2}{\partial \delta_2}^{(k)} & \cdots & \frac{\partial Q_2}{\partial \delta_n}^{(k)} & \frac{\partial Q_2}{\partial |V_2|}^{(k)} & \cdots & \frac{\partial Q_2}{\partial |V_n|}^{(k)} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial Q_n}{\partial \delta_2}^{(k)} & \cdots & \frac{\partial Q_n}{\partial \delta_n}^{(k)} & \frac{\partial Q_n}{\partial |V_2|}^{(k)} & \cdots & \frac{\partial Q_n}{\partial |V_n|}^{(k)} \end{array}\right] \begin{bmatrix} \Delta \delta_2^{(k)} \\ \vdots \\ \Delta \delta_n^{(k)} \\ \hline \Delta |V_2^{(k)}| \\ \vdots \\ \Delta |V_n^{(k)}| \end{bmatrix}$$

که به طور خلاصه به صورت زیر نیز نمایش داده می‌شود:

$$\begin{bmatrix} \Delta P \\ \Delta Q \end{bmatrix} = \begin{bmatrix} J_1 & J_2 \\ J_3 & J_4 \end{bmatrix} \begin{bmatrix} \Delta \delta \\ \Delta |V| \end{bmatrix}$$

برای عناصر قطری و غیر قطری ماتریس J_1 داریم:

$$\frac{\partial P_i}{\partial \delta_i} = \sum_{j \neq i}^{n} |V_i||V_j||Y_{ij}| \sin\left(\theta_{ij} - \delta_i + \delta_j\right)$$

$$\frac{\partial P_i}{\partial \delta_j} = -|V_i||V_j||Y_{ij}| \sin\left(\theta_{ij} - \delta_i + \delta_j\right) \quad j \neq i$$

برای عناصر قطری و غیر قطری J_2 داریم:

$$\frac{\partial P_i}{\partial |V_i|} = 2|V_i||Y_{ii}| \cos(\theta_{ii}) + \sum_{j \neq i}^{n} |V_i||V_j||Y_{ij}| \cos\left(\theta_{ij} - \delta_i + \delta_j\right)$$

$$\frac{\partial P_i}{\partial |V_j|} = |V_i||Y_{ij}| \cos\left(\theta_{ij} - \delta_i + \delta_j\right) \quad j \neq i$$

برای عناصر قطری و غیر قطری J_3 داریم:

$$\frac{\partial Q_i}{\partial \delta_i} = \sum_{j \neq i}^{n} |V_i||V_j||Y_{ij}| \cos\left(\theta_{ij} - \delta_i + \delta_j\right)$$

$$\frac{\partial Q_i}{\partial \delta_j} = -|V_i||V_j||Y_{ij}|\cos\left(\theta_{ij} - \delta_i + \delta_j\right) \quad j \neq i$$

عناصر قطری و غیر قطری ماتریس J_4 به صورت زیر تعیین می‌شوند:

$$\frac{\partial Q_i}{\partial |V_i|} = -2|V_i||Y_{ii}|\sin(\theta_{ii}) - \sum_{j\neq i}^{n}|V_i||V_j||Y_{ij}|\sin\left(\theta_{ij} - \delta_i + \delta_j\right)$$

$$\frac{\partial Q_i}{\partial |V_j|} = -|V_i||Y_{ij}|\sin\left(\theta_{ij} - \delta_i + \delta_j\right) \quad j \neq i$$

همچنین معادلات زیر را داریم:

$$\Delta P_i^{(k)} = P_i^{sch} - P_i^{(k)}$$
$$\Delta Q_i^{(k)} = Q_i^{sch} - Q_i^{(k)}$$
$$\delta_i^{(k+1)} = \delta_i^{sch} + \Delta\delta_i^{(k)}$$
$$|V_i^{(k+1)}| = |V_i^{(k)}| + \Delta|V_i^{(k)}|$$

مشابه قبل، داده لازم را از ماتریس bus بازیابی می‌کنیم. مقادیر فاز را نیز به رادیان تبدیل می‌کنیم. ماتریس توان مختلط را نیز از مقادیر توان در ماتریس bus ایجاد می‌کنیم. بعد از انجام محاسبات، مقادیر زاویه را به درجه و مقادیر توان را از پریونیت به مقدار واقعی می‌رسانیم. کد این قسمت به صورت زیر می‌باشد:

```
ns = 0;
ng = 0;
Vm = 0;
delta = 0;
yload = 0;
deltad = 0;
nbus = length(bus(:,1));
for k = 1:nbus
    n = bus(k,1);
    kb(n) = bus(k,2);
    Vm(n) = bus(k,3);
    delta(n) = bus(k, 4);
    Pd(n) = bus(k,5);
    Qd(n) = bus(k,6);
    Pg(n) = bus(k,7);
    Qg(n) = bus(k,8);
    Qmin(n) = bus(k, 9);
    Qmax(n) = bus(k, 10);
    if Vm(n) <= 0
        Vm(n) = 1.0;
        V(n) = 1 + 1j*0;
    else delta(n) = delta(n)*pi/180;
        V(n) = Vm(n)*(cos(delta(n))+1j*sin(delta(n)));
        P(n) = (Pg(n)-Pd(n))/basemva;
```

```
        Q(n) = (Qg(n)-Qd(n))/basemva;
        S(n) = P(n)+1j*Q(n);
    end
end
for k = 1:nbus
    if kb(k) == 1
        ns = ns+1;
    else
    end
    if kb(k) == 2
        ng = ng+1;
    else
    end
    ngs(k) = ng;
    nss(k) = ns;
end
Ym = abs(Y);
t = angle(Y);
m = 2*nbus-ng-2*ns;
maxerror = 1;
converge = 1;
iter = 0;
% Start of iterations
clear A  DC   J  DX
while maxerror >= accuracy && iter <= maxiter
    for i = 1:m
        for k = 1:m
            A(i,k) = 0;
        end
    end
    iter = iter+1;
    for n = 1:nbus
        nn = n-nss(n);
        lm = nbus+n-ngs(n)-nss(n)-ns;
        J1 = 0;
        J2 = 0;
        J3 = 0;
        J4 = 0;
    for i = 1:nbr
        if nl(i) == n || nr(i) == n
            if nl(i) == n
                l = nr(i);
            end
            if nr(i) == n
                l = nl(i);
            end
            J1 = J1+ Vm(n)*Vm(l)*Ym(n,l)*sin(t(n,l)- delta(n) + delta(l));
            J3 = J3+ Vm(n)*Vm(l)*Ym(n,l)*cos(t(n,l)- delta(n) + delta(l));
        if kb(n)~= 1
```

```
            J2 = J2+ Vm(l)*Ym(n,l)*cos(t(n,l)- delta(n) + delta(l));
            J4 = J4+ Vm(l)*Ym(n,l)*sin(t(n,l)- delta(n) + delta(l));
        else
        end
        if kb(n) ~= 1  && kb(l) ~=1
            lk = nbus+l-ngs(l)-nss(l)-ns;
            ll = l -nss(l);

            A(nn, ll) = -Vm(n)*Vm(l)*Ym(n,l)*sin(t(n,l)- delta(n) +
delta(l));
            if kb(l) == 0
                A(nn, lk) = Vm(n)*Ym(n,l)*cos(t(n,l)- delta(n) + delta(l));
            end
            if kb(n) == 0
                A(lm, ll) = -Vm(n)*Vm(l)*Ym(n,l)*cos(t(n,l)- delta(n)
+delta(l));
            end
            if kb(n) == 0 && kb(l) == 0
                A(lm, lk) = -Vm(n)*Ym(n,l)*sin(t(n,l)- delta(n) + delta(l));
            end
        else
        end
        else
        end
    end
    Pk = Vm(n)^2*Ym(n,n)*cos(t(n,n))+J3;
    Qk = -Vm(n)^2*Ym(n,n)*sin(t(n,n))-J1;
    if kb(n) == 1
        P(n)= Pk;
        Q(n) = Qk;
    end
    if kb(n) == 2
        Q(n)= Qk;
        if Qmax(n) ~= 0
            Qgc = Q(n)*basemva + Qd(n);
            if iter <= 7
                if iter > 2
                    if Qgc  < Qmin(n)
                        Vm(n) = Vm(n) + 0.01;
                    elseif Qgc  > Qmax(n)
                        Vm(n) = Vm(n) - 0.01;
                    end
                else
                end
            else
            end
        else
        end
    end
    if kb(n) ~= 1
```

```
        A(nn,nn) = J1;
        DC(nn) = P(n)-Pk;
    end
    if kb(n) == 0
        A(nn,lm) = 2*Vm(n)*Ym(n,n)*cos(t(n,n))+J2;
        A(lm,nn)= J3;
        A(lm,lm) = -2*Vm(n)*Ym(n,n)*sin(t(n,n))-J4;
        DC(lm) = Q(n)-Qk;
    end
    end
    DX = A\DC';
    for n = 1:nbus
        nn = n-nss(n);
        lm = nbus+n-ngs(n)-nss(n)-ns;
        if kb(n) ~= 1
            delta(n) = delta(n)+DX(nn);
        end
        if kb(n) == 0
            Vm(n) = Vm(n)+DX(lm);
        end
    end
    maxerror = max(abs(DC));
    if iter == maxiter && maxerror > accuracy
        fprintf('\nWARNING: Iterative solution did not converged after ')
        fprintf('%g', iter)
        fprintf(' iterations.\n\n')
        fprintf('Press Enter to terminate the iterations and print the
results \n')
        converge = 0;
        pause
    else
    end

end

if converge ~= 1
    tech = ('                        ITERATIVE SOLUTION DID NOT CONVERGE');
else
    tech = ('                    Power Flow Solution by Newton-Raphson
Method');
end
V = Vm.*cos(delta)+1j*Vm.*sin(delta);
deltad = delta*180/pi;
k=0;
for n = 1:nbus
    if kb(n) == 1
        k = k+1;
        S(n)= P(n)+1j*Q(n);
        Pg(n) = P(n)*basemva + Pd(n);
        Qg(n) = Q(n)*basemva + Qd(n);
```

```
        Pgg(k) = Pg(n);
        Qgg(k) = Qg(n);
    elseif  kb(n) ==2
        k = k+1;
        S(n) = P(n)+1j*Q(n);
        Qg(n) = Q(n)*basemva + Qd(n);
        Pgg(k) = Pg(n);
        Qgg(k) = Qg(n);
    end
    yload(n) = (Pd(n)- 1j*Qd(n))/(basemva*Vm(n)^2);
end
busdata(:,3)=Vm';
busdata(:,4)=deltad';
Pgt = sum(Pg);
Qgt = sum(Qg);
Pdt = sum(Pd);
Qdt = sum(Qd);
```

خروجی روش نیوتن-رفسون: تصاویر زیر به ترتیب خروجی‌های busout و lineflow را نمایش می‌دهند.

Maximum Power Mismatch = 6.86262e-07
No. of Iterations = 4

Bus No.	Voltage Mag.	Angle Degree	------Load------ MW	Mvar	---Generation--- MW	Mvar
1	1.040	0.000	0.000	0.000	72.341	81.367
2	1.025	9.693	0.000	0.000	163.000	53.412
3	1.025	4.881	0.000	0.000	85.000	38.554
4	0.996	-2.306	0.000	0.000	0.000	0.000
5	0.966	-3.737	90.000	30.000	0.000	0.000
6	1.004	2.107	0.000	0.000	0.000	0.000
7	0.979	0.836	100.000	35.000	0.000	0.000
8	0.997	3.974	0.000	0.000	0.000	0.000
9	0.951	-4.138	125.000	50.000	0.000	0.000
Total			315.000	115.000	320.341	173.333

Line Flow and Losses

--Line-- from	to	Power at bus & line flow MW	Mvar	MVA	--Line loss-- MW	Mvar
1		72.341	81.367	108.875		
	4	72.341	81.368	108.875	-0.000	6.313
2		163.000	53.412	171.528		
	8	163.000	53.412	171.528	0.000	17.503
3		85.000	38.554	93.335		
	6	85.000	38.554	93.335	0.000	4.859
4		0.000	0.000	0.000		
	1	-72.341	-75.055	104.242	-0.000	6.313
	5	31.026	26.555	40.838	0.286	1.547
	9	41.315	48.500	63.712	0.409	3.480
5		-90.000	-30.000	94.868		
	4	-30.740	-25.007	39.627	0.286	1.547
	6	-59.260	-4.993	59.470	1.477	6.440
6		0.000	0.000	0.000		
	5	60.737	11.433	61.804	1.477	6.440
	3	-85.000	-33.695	91.435	0.000	4.859
	7	24.263	22.262	32.928	0.128	1.084
7		-100.000	-35.000	105.948		
	6	-24.135	-21.178	32.109	0.128	1.084
	8	-75.865	-13.822	77.114	0.527	4.466
8		0.000	0.000	0.000		
	7	76.393	18.287	78.551	0.527	4.466
	2	-163.000	-35.909	166.909	0.000	17.503
	9	86.607	17.622	88.382	2.513	12.642
9		-125.000	-50.000	134.629		
	8	-84.095	-4.980	84.242	2.513	12.642
	4	-40.905	-45.020	60.828	0.409	3.480
Total loss					5.341	58.334

روش مجزای سریع:

فایل maindecouple: مشابه فایل mainnewton.m می‌باشد.

فایل decouple.m: می‌دانیم در این روش ماتریس‌های J_2 و J_3 صفر در نظر گرفته می‌شوند. تعریف می‌کنیم:

$$B_{ii} = |Y_{ii}| \sin(\theta_{ii})$$

برای عناصر قطری و غیر قطری J_1 داریم:

$$\frac{\partial P_i}{\partial \delta_i} = -|V_i| B_{ii}$$

$$\frac{\partial P_i}{\partial \delta_j} = -|V_i| B_{ij}$$

برای عناصر قطری و غیر قطری J_4 نیز داریم:

$$\frac{\partial Q_i}{\partial |V_i|} = -|V_i||Y_{ii}\sin(\theta_{ii}) + Q_i$$

$$\frac{\partial Q_i}{\partial |V_i|} = -|V_i|B_{ij}$$

همچنین برای تغییرات متوالی اندازه ولتاژ و زاویه فاز داریم:

$$\Delta\delta = -[B']^{-1}\frac{\Delta P}{|V|}$$

$$\Delta|V| = -[B'']^{-1}\frac{\Delta Q}{|V|}$$

درنهایت تمام مراحلی که در روش‌های قبلی طی می‌کردیم را انجام می‌دهیم. با این تفاوت که باید ماتریس‌های B1 و B2 را که همان B' و B'' هستند را از هم جدا کنیم. کد مرتبط با این روش در ادامه قابل مشاهده است:

```
ns = 0;
Vm = 0;
delta = 0;
yload = 0;
deltad = 0;
nbus = length(bus(:,1));

for k = 1:nbus
    n = bus(k,1);
    kb(n) = bus(k,2);
    Vm(n) = bus(k,3);
    delta(n) = bus(k, 4);
    Pd(n) = bus(k,5);
    Qd(n) = bus(k,6);
    Pg(n) = bus(k,7);
    Qg(n) = bus(k,8);
    Qmin(n) = bus(k, 9);
    Qmax(n) = bus(k, 10);
    if Vm(n) <= 0
        Vm(n) = 1.0;
        V(n) = 1 + 1j*0;
    else delta(n) = pi/180*delta(n);
        V(n) = Vm(n)*(cos(delta(n))+1j*sin(delta(n)));
        P(n) = (Pg(n)-Pd(n))/basemva;
        Q(n) = (Qg(n)-Qd(n))/basemva;
```

```
        S(n) = P(n) + 1j*Q(n);
    end
    if kb(n) == 1
        ns = ns+1;
    else
    end
    nss(n) = ns;
end
Ym = abs(Y);
t = angle(Y);
ii = 0;
for ib = 1:nbus
     if kb(ib) == 0 || kb(ib) == 2
        ii = ii+1;
        jj = 0;
        for jb = 1:nbus
            if kb(jb) == 0 || kb(jb) == 2
            jj = jj+1;
            B1(ii,jj) = imag(Y(ib,jb));
            else
            end
        end
     else
     end
end

ii = 0;
for ib = 1:nbus
     if kb(ib) == 0
        ii = ii+1;
        jj = 0;
        for jb = 1:nbus
            if kb(jb) == 0
                jj = jj+1;
                B2(ii,jj) = imag(Y(ib,jb));
            else
            end
        end
     else
     end
end
B1inv = inv(B1);
B2inv = inv(B2);
maxerror = 1;
converge = 1;
iter = 0;
% Start of iterations
while maxerror >= accuracy && iter <= maxiter
    iter = iter+1;
```

```
    id = 0;
    iv = 0;
    for n = 1:nbus
        nn = n-nss(n);
        J11 = 0;
        J33 = 0;
        for i = 1:nbr
            if nl(i) == n || nr(i) == n
                if nl(i) == n
                    l = nr(i);
                end
                if nr(i) == n
                    l = nl(i);
                end
                J11 = J11+ Vm(n)*Vm(l)*Ym(n,l)*sin(t(n,l)- delta(n) +
delta(l));
                J33 = J33+ Vm(n)*Vm(l)*Ym(n,l)*cos(t(n,l)- delta(n) +
delta(l));
            else
            end
        end
        Pk = Vm(n)^2*Ym(n,n)*cos(t(n,n))+J33;
        Qk = -Vm(n)^2*Ym(n,n)*sin(t(n,n))-J11;
        if kb(n) == 1
            P(n) = Pk;
            Q(n) = Qk;
        end
        if kb(n) == 2
            Q(n) = Qk;
            Qgc = Q(n)*basemva + Qd(n);
            if Qmax(n) ~= 0
                if iter <= 20
                    if iter >= 10
                        if Qgc  < Qmin(n)
                            Vm(n) = Vm(n) + 0.005;
                        elseif Qgc > Qmax(n)
                            Vm(n) = Vm(n) - 0.005;
                        end
                    else
                    end
                else
                end
            else
            end
        end
        if kb(n) ~= 1
            id = id+1;
            DP(id) = P(n)-Pk;
            DPV(id) = (P(n)-Pk)/Vm(n);
        end
```

```
        if kb(n) == 0
            iv = iv+1;
            DQ(iv) = Q(n)-Qk;
            DQV(iv) = (Q(n)-Qk)/Vm(n);
        end
    end
    Dd = -B1\DPV';
    DV = -B2\DQV';
    id = 0;
    iv = 0;
    for n = 1:nbus
        if kb(n) ~= 1
            id = id+1;
            delta(n) = delta(n)+Dd(id);
        end
    if kb(n) == 0
        iv = iv+1;
        Vm(n) = Vm(n)+DV(iv);
    end
    end
    maxerror = max(max(abs(DP)),max(abs(DQ)));
    if iter == maxiter && maxerror > accuracy
        fprintf('\nWARNING: Iterative solution did not converged after ')
        fprintf('%g', iter)
        fprintf(' iterations.\n\n')
        fprintf('Press Enter to terminate the iterations and print the
results \n')
        converge = 0;
        pause
    else
    end

end
if converge ~= 1
    tech= ('                        ITERATIVE SOLUTION DID NOT CONVERGE');
else
    tech=('                    Power Flow Solution by Fast Decoupled Method');
end
k = 0;
V = Vm.*cos(delta)+1j*Vm.*sin(delta);
deltad = 180/pi*delta;

clear A DC DX

for n = 1:nbus
     if kb(n) == 1
        S(n)=P(n)+1j*Q(n);
        Pg(n) = P(n)*basemva + Pd(n);
        Qg(n) = Q(n)*basemva + Qd(n);
        k = k+1;
```

```
        Pgg(k) = Pg(n);
    elseif  kb(n) == 2
        S(n) = P(n)+1j*Q(n);
        Qg(n) = Q(n)*basemva + Qd(n);
        k = k+1;
        Pgg(k) = Pg(n);
    end
    yload(n) = (Pd(n)- 1j*Qd(n))/(basemva*Vm(n)^2);
end
bus(:,3) = Vm';
bus(:,4) = deltad';
Pgt = sum(Pg);
Qgt = sum(Qg);
Pdt = sum(Pd);
Qdt = sum(Qd);
clear Pk Qk  DP DQ J11 J33 B1 B1inv B2 B2inv DPV  DQV Dd delta ib id ii iv jb
jj
```

خروجی روش مجزای سریع: تصاویر زیر به ترتیب خروجی‌های busout و lineflow را نمایش می‌دهند:

Maximum Power Mismatch = 2.10374e-05
No. of Iterations = 7

Bus No.	Voltage Mag.	Angle Degree	------Load------ MW	Mvar	---Generation--- MW	Mvar
1	1.040	0.000	0.000	0.000	72.340	81.367
2	1.025	9.693	0.000	0.000	163.000	53.411
3	1.025	4.881	0.000	0.000	85.000	38.554
4	0.996	-2.306	0.000	0.000	0.000	0.000
5	0.966	-3.737	90.000	30.000	0.000	0.000
6	1.004	2.107	0.000	0.000	0.000	0.000
7	0.979	0.836	100.000	35.000	0.000	0.000
8	0.997	3.974	0.000	0.000	0.000	0.000
9	0.951	-4.138	125.000	50.000	0.000	0.000
Total			315.000	115.000	320.340	173.332

Line Flow and Losses

--Line-- from	to	Power at bus & line flow MW	Mvar	MVA	--Line loss-- MW	Mvar
1		72.340	81.367	108.874		
	4	72.341	81.367	108.875	0.000	6.313
2		163.000	53.411	171.528		
	8	163.000	53.412	171.528	0.000	17.503
3		85.000	38.554	93.335		
	6	85.000	38.554	93.335	0.000	4.859
4		0.000	0.000	0.000		
	1	-72.341	-75.055	104.242	0.000	6.313
	5	31.026	26.555	40.838	0.286	1.548
	9	41.315	48.500	63.712	0.409	3.480
5		-90.000	-30.000	94.868		
	4	-30.740	-25.008	39.627	0.286	1.548
	6	-59.260	-4.993	59.470	1.477	6.440
6		0.000	0.000	0.000		
	5	60.737	11.433	61.804	1.477	6.440
	3	-85.000	-33.695	91.435	0.000	4.859
	7	24.263	22.262	32.928	0.128	1.084
7		-100.000	-35.000	105.948		
	6	-24.135	-21.178	32.109	0.128	1.084
	8	-75.865	-13.822	77.114	0.527	4.466
8		0.000	0.000	0.000		
	7	76.393	18.287	78.551	0.527	4.466
	2	-163.000	-35.909	166.908	0.000	17.503
	9	86.607	17.622	88.382	2.513	12.642
9		-125.000	-50.000	134.629		
	8	-84.095	-4.980	84.242	2.513	12.642
	4	-40.905	-45.020	60.828	0.409	3.480
Total loss					5.341	58.334

**مقایسه خروجی سه روش نشان می‌دهد که نتایج نهایی تقریبا یکسان هستند.

مقایسه پیچیدگی روش‌ها

مقایسه تعداد تکرارها: روش گوس-سایدل بعد از ۴۸ تکرار به نتیجه رسیده و این تعداد نیز با استفاده از ضریب تسریع ۱.۳ رخ داده. روش نیوتن-فسون بعد از ۴ مرحله به تکرار رسیده که بین این سه روش کم‌ترین تعداد تکرار می‌باشد. روش مجزای سریع نیز بعد از هفت تکرار به نتیجه رسیده.

زمان فرآیند محاسبات: در سیستم مورد بررسی ما، روش گوس-سایدل تقریبا بعد از 0.0394 ثانیه، روش مجزای سریع در 0.437 ثانیه و روش نیوتن-رفسون بعد از 0.999 به جواب نهایی رسیده.

منبع: کتاب بررسی سیستم‌های قدرت دکتر هادی سعادت