ECE 31032—ABET Project (due by 12/1)

The purpose of this project is to assess attainment of the ABET Learning Objectives of this course. The project will be graded on a pass/fail basis. To pass the project, you need to:

1. answer all the project questions correctly (as verified by a test script provided on Brightspace),
2. record a 10-minute video in which you explain your code and your solution, and run the test script which confirms that all your solutions are correct,
3. submit via Brightspace by the due date a screenshot of the output of the test script, your recording, and a zip file containing all your code that can reproduce the results shown in your recording.

Failure to complete these steps means that you will get 0 for the project part of this course.

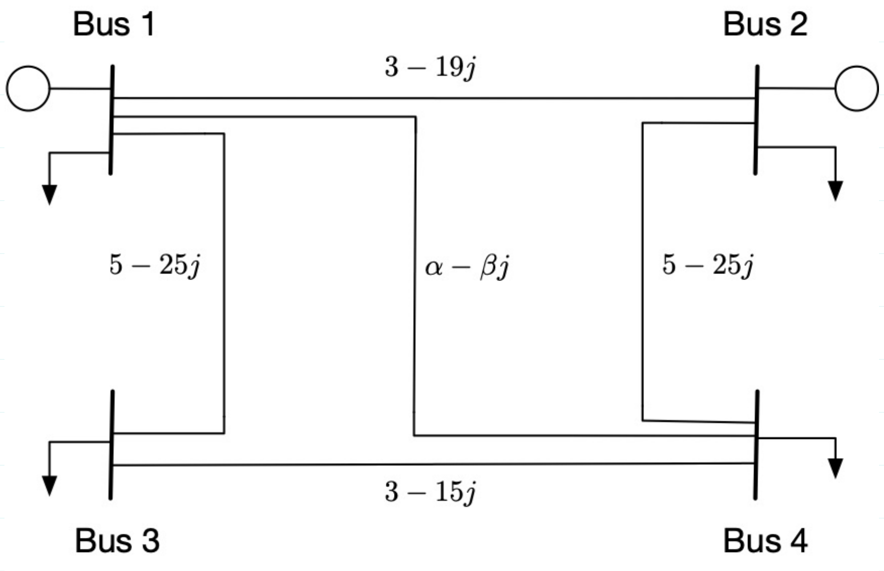
Instructions:

- *Please refer to the following documents to get started on this project: (a) the tips and suggestions part of this pdf, (b) the solution to homework 7, and (c) the sample code provided for the 3-bus system (this will be uploaded to Brightspace after homework 7 is due).*
- *I and the TAs will be available to help you with this project. Please send us an email if you get stuck. We can either set up one-on-one meetings or I may hold extra office hours depending on the demand. The extra office hours will likely be in the weekends.*
- *Please start early. Plan to create your first version of code at least one week ahead of the deadline so you have sufficient time to revise and debug.*
- *You should write your own code for the project. Copying code online or directly use an existing implementation of the Newton-Raphson method is not acceptable.*

Good luck!

Name: ______________________

Consider the power system depicted below, where the values associated with the lines are per unit admittance values. The per unit admittance value for the line connecting bus 1 and bus 4 is $\alpha - \beta j$, where α and β are defined according to the alphabetic order of your initials. For example, for initials "J.Q.", $\alpha = 10$ and $\beta = 17$. Base values for the system are 100 MVA and 230 kV.



The bus data are listed in the table below, where "N.A." indicates that the corresponding information is not available.

	Generation		Load			
Bus	P, MW	Q, MVar	P, MW	Q, MVar	V_i, p.u.	ϕ_i
1 (slack)	N.A.	N.A.	50	30	1	0
2	300	N.A.	80	50	1.02	N.A.
3	0	0	170	105	N.A.	N.A.
4	0	0	200	125	N.A.	N.A.

Write a computer program that implements a Newton-Raphson power flow solution. Your code can be based on modifying the Matlab script (to be uploaded after homework 4 is due) that was presented in class, or you can write your own program from scratch. You can use the language of your choice (e.g., C, Python, etc).

Calculate:

1. The voltage phasor at bus 4 (in pu).
2. The complex power generation at bus 2 (in MW/MVAr).
3. The total real power generation within this system (in MW).
4. The complex power leaving bus 1 towards bus 3 (in MW/MVAr).

Tips and Suggestions

Here we provide some tips and instructions that might be helpful for you to better structure your code and streamline the process for solving the project. You can structure your code differently and **do not have to follow these instructions**. Nevertheless, some information provided here can still be of use for you to cross-check your code and debug certain portion of your code.

Useful steps: (All discussion here assume that you are following the bus ordering convention as in the Aliprantis notes.)

1. Write down the Y-bus matrix, and voltage unknowns x, as we did in class. Also convert all the non-p.u. quantities to p.u. All variables below are in per unit unless mentioned otherwise.

2. Implement an Matlab function that converts the voltage unknowns x to the vector of voltage r.m.s. $V \in \mathbb{R}^N$ and an Matlab function that converts the voltage unknowns x to the vector of voltage phase angles $\phi \in \mathbb{R}^N$. The functions might begin with

```
function V = x2V(x, Vg, Ng, N)
% Vg: Ng x 1, Vector of known voltage magnitudes for slack and gen buses
% Ng: scalar, number of generator buses
% N: scalar, total number of buses
% V: N x 1, Vector of voltage magnitudes for all buses
```

   and

```
function phi = x2phi(x, Ng, N)
% Ng: scalar, number of generator buses
% N: scalar, total number of buses
% phi: N x 1, Vector of voltage phase angles for all buses
```

   For example, for function x2V with inputs

```
 x = [0 0 0 1.0 1.0]  %[phi2 phi3 phi4 V3 V4]
Vg = Vk               % Vk =[1 1.02] (V1 V2)
Ng = 2
 N = 4
```

   you should get the following output

```
    V = [1 1.02 1.0 1.0]

    and
```

for function x2phi with inputs

```
     x = [0 0 0 1.0 1.0] %[phi2 phi3 phi4 V3 V4]
    Ng = 2
     N = 4
```

you should get the following output

```
    phi = [0 0 0 0]
```

3. Implement Matlab functions that compute the vector of $P_i(x)$ and $Q_i(x)$ values, respectively, given the voltage rms vector V and phase angle vector ϕ. Note that the voltage rms vector V and phase angle vector ϕ in turn depends on x as calculated in the previous step. The function might begin with:

```
function PixVec = Pifn(Ybus, V, phi, N)
```

and

```
function QixVec = Qifn(Ybus, V, phi, N)
```

For example, for function Pifn and Qifn with inputs

```
Ybus = [9-63i -3+19i -5+25i -1+19i;
       -3+19i  8-44i     0   -5+25i;
       -5+25i   0     8-40i -3+15i;
       -1+19i -5+25i -3+15i  9-59i];

  V = [1 1.02 1.0 1.0]                %[V1 V2 V3 V4]
phi = [0 0 0 0]                       %[phi1 phi2 phi3 phi4]
    N = 4
```

you should get the following output

```
    PixVec = [0.1632 0 -0.1]       %[P2x P3x P4x]
    QixVec = [0.8976 0 -0.5]      %[Q2x Q3x Q4x]
```

Note that this function does not output $P_1(x)$ and $Q_1(x)$ because bus 1 is the slack bus and we never have P_1 and Q_1 as a part of the known quantities.

4. Implement an Matlab function that calculate the mismatch for the relevant power flow equations. By relevant power flow equations, we mean the equations that are used to solve the voltage unknowns as discussed in class. The vector of mismatch values is simply the vector $f(x)$ in the Aliprantis notes. The function may begin with

```
 function f = pqmismatch(x, Pknown, Qknown, PixVec, QixVec)
```

For example, for function pqmismatch with inputs

```
     x = [0 0 0 1.0 1.0]        %[phi2 phi3 phi4 V3 V4]
Pknown = [2.2 -1.7 -2]          %[P2i P3i P4i] in p.u
Qknown = [-0.5 -1.05 -1.25]     %[Q2i Q3i Q4i] in p.u
  PixVec = [0.1632 0 -0.1]        %[P2x P3x P4x] in p.u
  QixVec = [0.8976 0 -0.5]        %[Q2x Q3x Q4x] in p.u
```

you should get the following output

```
    fx = [-2.0368 1.7 1.9 1.05 0.75]
```

5. Implement an Matlab function that evaluate the Jacobian matrix for any vector of voltage unknowns x. The function might begin with

   ```
   function J = pfJacobian(Ybus, V, phi, PixVec, QixVec)
   ```

   For example, for function pfJacobian with inputs

   ```
   Ybus = [9-63i -3+19i -5+25i -1+19i;
          -3+19i  8-44i     0   -5+25i;
          -5+25i   0     8-40i -3+15i;
          -1+19i -5+25i -3+15i  9-59i];

      V = [1 1.02 1.0 1.0]                %[V1 V2 V3 V4]
    phi = [0 0 0 0]                       %[phi1 phi2 phi3 phi4]
    PixVec = [0.1632 0 -0.1]                %[P2x P3x P4x] in p.u
    QixVec = [0.8976 0 -0.5]                %[Q2x Q3x Q4x] in p.u
   ```

   you should get the following output

   ```
   J =  [44.8800        0    -25.5000        0      -5.1000
               0      40.0000  -15.0000    8.0000   -3.0000
        -25.5000  -15.0000   59.5000   -3.0000    8.9000
               0      -8.0000    3.0000   40.0000  -15.0000
          5.1000    3.0000   -9.1000  -15.0000   58.5000]
   ```

   You may hard code each element of the Jacobian matrix for this particular network using formula from the Aliprantis notes, or you can write a routine to automatically generate the Jacobian matrix for any network.

6. Implement the Newton-Raphson iterations. The update equation may look like

```
Jx = pfJacobian(Ybus, V, phi, PixVec, QixVec);
fx = pqmismatch(x, Pknown, Qknown, PixVec, QixVec);
x = x - Jx \ fx;
```

For example, with inputs as listed in item 5, which is consistent with the following initial guess x

```
 x = [0 0 0 1.0 1.0]
```

you should get

```
Jx =  [44.8800        0    -25.5000       0      -5.1000
          0      40.0000  -15.0000    8.0000   -3.0000
     -25.5000  -15.0000   59.5000   -3.0000    8.9000
          0      -8.0000    3.0000   40.0000  -15.0000
        5.1000    3.0000   -9.1000  -15.0000   58.5000]

fx = [-2.0368 1.7 1.9 1.05 0.75]
```

The updated x after the first iteration is

```
    x = [0.0242   -0.0476   -0.0315    0.9558    0.9713]
```

To get the second iteration working, you should make sure to generate your new V, ϕ, $P_i(x)$, $Q_i(x)$ vectors using the updated x, and get a new Jacobian matrix and mismatch vector. You should get the following output for the 2nd iteration

```
    x = [0.0239   -0.0502   -0.0331    0.9524    0.9690]
```