ปีการศึกษา 2540

Power System Planning

Load Flow and Load Forecasting

โดย

นาย โยชัย ศศิวรรณ
นาย รุ่งโรจน์ วนพฤกษาศิลป์

อาจารย์ที่ปรึกษา

อาจารย์ สมโภชน์ ประไพ

ปริญญานิพนธ์ ปีการศึกษา 2540

ภาควิชาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การใช้คอมพิวเตอร์ช่วยในงานวางแผนระบบไฟฟ้ากำลัง
: การวิเคราะห์โหลดโฟล และการทำนายโหลด
Power System Planning
: Load Flow and Load Forecasting

ผู้จัดทำ

1. นาย โยชัย	ศศิวรรณ	37.014346
2. นาย รุ่งโรจน์	วนพฤกษาศิลป์	37.014363

อาจารย์ที่ปรึกษา
(สมโภชน์ ประไพ)

การใช้คอมพิวเตอร์ช่วยในงานวางแผนระบบไฟฟ้ากำลัง การวิเคราะห์โหลดโฟล และการทำนายโหลด

นาย โยชัย ศศิวรรณ
นาย รุ่งโรจน์ วนพฤกษาศิลป์
อาจารย์ที่ปรึกษา
อาจารย์ สมโภชน์ ประไพ
ปีการศึกษา ๒๕๔๐

บทคัดย่อ

วิทยานิพนธ์ฉบับนี้อธิบายถึง วิธีการวิเคราะห์โหลดโฟลในระบบไฟฟ้ากำลัง โดยใช้ไมโครคอมพิวเตอร์ในการคำนวณ จากการรับข้อมูลบัส ,ข้อมูลสายส่ง ,และข้อมูลของหม้อแปลงไฟฟ้ากำลังในระบบ แล้วนำข้อมูลดังกล่าว มาทำการวิเคราะห์โดยวิธีของนิวตัน-ราฟสัน จะได้ผลลัพธ์ คือค่ากำลังไฟฟ้าแอคทีฟ ,กำลังไฟฟ้ารีแอคทีฟ ,มุมเฟสของแรงดัน ,และขนาดของแรงดัน ณ บัสต่าง ๆ ในระบบ ซึ่งจะทำให้ทราบการถ่ายโอนกำลังไฟฟ้าระหว่างบัสต่าง ๆ หลังจากนั้นก็จะนำผลการวิเคราะห์โหลดโฟล มาใช้ในการทำนายโหลด ซึ่งเป็นไปในลักษณะของการทดสอบ ปรับปรุง เปลี่ยนแปลง หรือเพิ่มเติมคุณสมบัติของระบบ แล้วนำกลับไปหาผลลัพธ์ที่น่าพอใจด้วยวิธีวิเคราะห์โหลดโฟลอีกครั้งหนึ่ง เพื่อนำไปใช้กับการวางแผนงานในการรองรับการเปลี่ยนแปลงของโหลด ซึ่งมีแต่จะเพิ่มขึ้นในอนาคตได้อย่างถูกต้อง

Power System Planning
Load Flow and Load Forecasting

Mr. Yochai Sasiwan
Mr. Roongrot Wanaphurksasilp
Advisor Mr. Sompotsh Prapai
1997

Abstract

This thesis describes the method for determine loadflow solution of power system based on microcomputer. From inputting of buses data, transmission lines data and transformers data those are the components in the power system, and using Newton-Raphson's method in analysis procedures. The first solution is contain with Active power, Reactive power, Voltage phase angle and Votage magnitude at each bus, which show the power transfer between buses in entire system. In advantage, loadflow solution was used in Load Forecasting concentrate in testing, improving ,modifying or adding new system characteristics and analyze loadflow solution in the system again to find the appropriate solution. The lastest solution was used in power system planning to support the increasing load in the future.

สารบัญ

เรื่อง	*Page No.*
1. บทนำ	1
2. แบบจำลองของระบบ (System Modelling)	2
3. การคำนวณในระบบไฟฟ้ากำลัง (Network Calculations)	6
4. การศึกษาโหลดโฟล (Load Flow Studies)	11
5. การใช้วิธีของนิวตัน-ราฟสัน แก้ปัญหาโหลดโฟล (Newton-Raphson Method of Solving Load Flow)	16
6. การออกแบบโปรแกรมโหลดโฟล และกระบวนการทำนายโหลด (Design Load Flow Program and Load Forecasting Procedure)	21
7. ผลการทดสอบ	29
8. สรุปผลและวิจารณ์	40
9. ภาคผนวก	41
หนังสืออ้างอิง	92
กิตติกรรมประกาศ	93.

บทที่ 1

บทนำ

Power System Planning : Load Flow and Load Forecasting เป็นการนำความรู้ที่ได้จากการเรียนวิชา Power System มาประยุกต์ใช้งาน โดยมี Computer เป็นอุปกรณ์สำคัญในการศึกษา เราสามารถนำทฤษฎีของ Load Flow Study มาเขียนเป็น Program Computer ช่วยในการวิเคราะห์หาคำตอบที่ต้องการในระบบไฟฟ้ากำลังขนาดใหญ่ อันประกอบไปด้วยค่าของ Active Power ,Reactive Power ,Voltage Magnitude และ Voltage Phase Angle ได้อย่างรวดเร็ว และถูกต้องแม่นยำ สามารถนำผลที่ได้นี้ไปใช้งานต่อได้ทันที และในส่วนของ Load Forecasting ก็จะเป็นการวางแผนเกี่ยวกับระบบไฟฟ้ากำลังในอนาคต โดยอาจจะเป็นการเปลี่ยนแปลง ปรับปรุงหรือเพิ่มเติมคุณสมบัติ ในระบบที่ใช้กันอยู่ในปัจจุบัน เพื่อให้การผลิตกำลังไฟฟ้าสอดคล้องกับปริมาณของ Load ที่คาดว่าจะเพิ่มขึ้นในอนาคต และระบบมีความน่าเชื่อถือได้มากขึ้น โดยผลลัพธ์สุดท้ายที่จะบอกว่าการปรับปรุงระบบนั้นเป็นที่น่าพอใจหรือไม่ จะได้จากการคำนวณ Load Flow Study เป็นผลให้เราสามารถตัดสินใจได้ก่อนการลงมือปรับปรุงระบบกันจริง ๆ และยังจะช่วยลดความเสี่ยงในการลงทุนลงได้อีกด้วย

จากผลลัพธ์ที่ได้มานี้เอง เราสามารถนำไปใช้ประโยชน์ได้มากมาย มีทั้งการนำไปใช้ในการคำนวณออกแบบหาขนาดของ Equipment ต่าง ๆ ในระบบไฟฟ้าแรงสูง ไม่ว่าจะเป็นขนาด Busbar ,Transmission Line หรือ Transformer หรือนำไปใช้ในการคำนวณผลที่เกิดขึ้นจากความผิดพลาดในระบบไฟฟ้าแรงสูง (Fault Calculation) เพื่อการออกแบบระบบ Protection ที่น่าเชื่อถือภายในสถานีไฟฟ้าแรงสูง รวมทั้งการวางแผนการส่งจ่ายพลังงานไฟฟ้าอย่างคุ้มค่าที่สุด (Economic Load Dispatch) ซึ่งล้วนแล้วแต่มีพื้นฐานมาจาก Load Flow Study ทั้งสิ้น

บทที่ 2

System Modelling.

เป็นการแทนวงจรสมมูลย์ของอุปกรณ์ที่ใช้ในระบบส่งผ่านกำลังไฟฟ้าอย่างต่อเนื่อง ซึ่งอาจจะประกอบไปด้วยเทอมของ inductance ,capacitance และ resistacne แล้วแต่ว่าอุปกรณ์นั้นมีการเชื่อมต่อและคุณสมบัติในระบบเป็นเช่นไร ดังรายละเอียดต่อไปนี้

1. Transmission Line Modelling.

ในกรณีของ transmission line ค่าโดยรวมของ resistance และ inductive reactance ของสาย ถูกรวมไว้ในส่วนที่ต่ออนุกรมของวงจรสมมูลย์-π และค่าโดยรวมของ capacitance ต่อ neutral ถูกแบ่งอยู่ระหว่างส่วนที่ต่อขนาน

เช่น transmission line ต่ออยู่ระหว่าง 2 nodes i และ k จะมี series impedance เป็น

$$z_{ik} = |z_{ik}|\exp(j\zeta_{ik}) = r_{ik} + jx_{ik} \quad (2.1)$$

และ shunt admittance บนด้าน i เป็น

$$y'_{ik} = g_{ik} + jb_{ik} \quad (2.2)$$

transmission line modelling สามารถแสดงได้ดังรูป

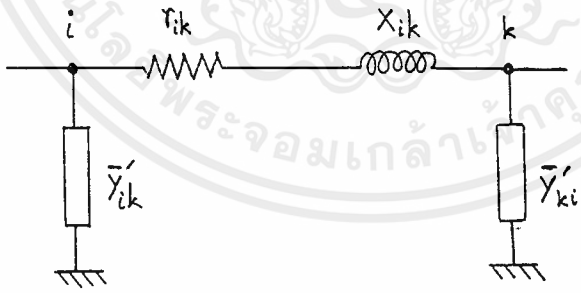


Fig 2.1 Transmission Line Modelling.

2. Transformer Modelling.

2.1 Transformer on Nominal Ratio.

รูปแบบวงจรสมมูลย์-π ของหม้อแปลงแสดงดังในรูปที่ 2.2 เมื่อ y_{oc} เป็นค่าอินเวอร์สของ Z_{oc} (magnetising impedance) และ y_{sc} เป็นค่าอินเวอร์สของ Z_{sc} (leakage impedance) Z_{sc} และ Z_{oc} ได้จากการทำ short-circuit test และ open-circuit test

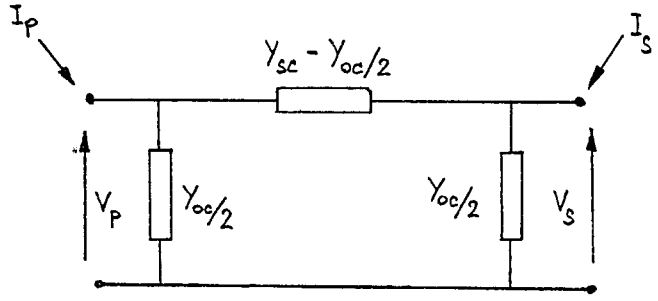


Fig 2.2 Transformer equivalent circuit.

และสามารถสร้างสมการความสัมพันธ์ในรูปเมตริกซ์ ได้เป็น

$$\begin{bmatrix} I_p \\ I_s \end{bmatrix} = \begin{bmatrix} y_{sc} & -y_{se} + \frac{y_{oc}}{2} \\ -y_{sc} + \frac{y_{oc}}{2} & y_{sc} \end{bmatrix} \begin{bmatrix} V_p \\ V_s \end{bmatrix} \qquad (2.3)$$

เมื่อ y_{sc} เป็น short-circuit หรือ leakage admittance และ y_{oc} เป็น open-circuit หรือ magnetising admittance

การใช้ three-terminal network ถูกจำกัดอยู่เฉพาะในการแสดงในรูปที่เป็น single phase และไม่สามารถใช้สร้างเป็น block สำหรับแบบจำลองของ 3-phase transformer banks ได้

โดยทั่วไปแล้วค่าของ magnetising admittance จะถูกตัดออกจากแบบจำลองของหม้อแปลง และจะเพิ่มเข้าไปในภายหลังด้วย shunt-connected admittance ค่าน้อย ๆ ที่ขั้วของหม้อแปลง ในระบบที่เป็น per-unit แบบจำลองของ single-phase transformer สามารถทำการลดรูปให้เหลือเพียง lumped leakage admittance เท่านั้น ระหว่างด้าน primary และ secondary ของบัสบาร์

2.2 Off-nominal Transformer Tap Setting.

หม้อแปลงซึ่งมี turn ratio a ต่อเข้ากับ node 2 nodes i และ k สามารถที่จะแสดงได้ด้วยหม้อแปลงอุดมคติ ต่ออนุกรมกับ nominal transformer leakage admittance ดังแสดงในรูปที่ 2.3(a)

ถ้าหม้อแปลงเป็นแบบ on-nominal tap (a=1) สมการ node สำหรับวงจรในระบบ per-unit จะเป็น

$$I_{ik} = y_{ik}V_i - y_{ik}V_k \qquad (2.4)$$

$$I_{ki} = y_{ik}V_k - y_{ik}V_i \qquad (2.5)$$

ในกรณีนี้ $I_{ik} = -I_{ki}$

สำหรับ off-nominal tap setting และให้แรงดันบนด้าน k ของหม้อแปลงอุดมคติเป็น V_t เราสามารถเขียนได้ว่า

$$V_t = V_i / a \qquad (2.6)$$

$$I_{ki} = y_{ik}(V_k - V_t) \qquad (2.7)$$

$$I_{ik} = -I_{ki} / a \qquad (2.8)$$

กำจัด V_t ระหว่าง สมการที่ (2.6) และ (2.7) เราจะได้

$$I_{ki} = y_{ik}V_k - y_{ik}V_t/a \qquad (2.9)$$

$$I_{ik} = -y_{ik}V_t/a + y_{ik}V_t/a^2 \qquad (2.10)$$

วงจรสมมูลย์-π แบบง่าย ๆ สามารถสรุปได้จากสมการที่ (2.9) และ (2.10) สมาชิกแต่ละตัวสามารถจัดอยู่ในรูปของ admittance matrix ได้ แสดงเป็นวงจรได้ดังรูปที่ 2.3(b)

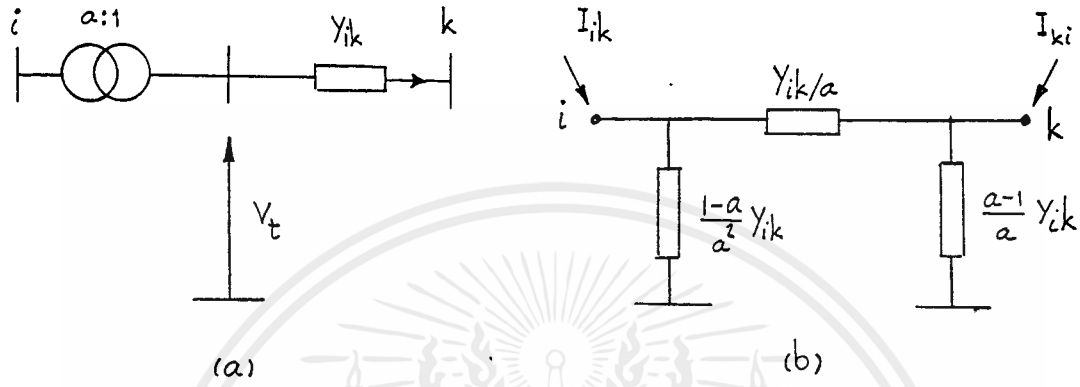


Fig 2.3 Transformer with off-nominal tap setting.

2.3 Phase-shifting Transformers.

ในกรณีของ phase-shifting หม้อแปลงในรูปที่ 2.4 จะมีลักษณะเป็น complex turn ratio นอกจากนี้ ความไม่เปลี่ยนแปลงของผลคูณ VI* ผ่านหม้อแปลงอุดมคติต้องแสดงความแตกต่างระหว่าง turn ratios ของกระแสและแรงดันด้วย

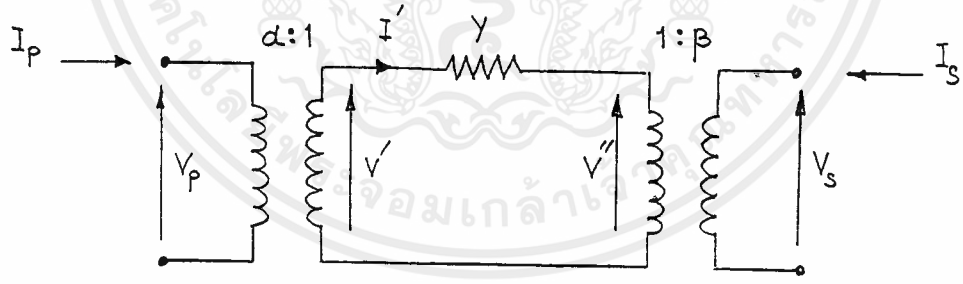


Fig 2.4 Basic equivalent circuit in p.u. for coupling between primary and secondary coils with both primary and secondary off-nominal tap ratios of α and β

$$V_p I_p^* = -V'I'^*$$

หรือ

$$V_p = (a+jb)V' = \alpha V'$$

$$I_p^* = -I'^*/(a+jb)$$

$$I_p = -I'/(a-jb) = -I'/\alpha^*$$

ดังนั้น วงจรในรูปที่ 2.4 จึงมี turn ratios 2 ค่า ที่แตกต่างกัน

$\alpha_v = a + jb$ สำหรับ voltages

และ $\alpha_i = a - jb$ สำหรับ currents

แก้สมการของ terminal current สำหรับวงจรที่ปรับปรุงแล้ว จะได้

$$I_p = I'/\alpha_i = (V' - V'')y/\alpha_i$$

$$= (V_p/\alpha_v - V_s/\beta)y/\alpha_i = yV_p/\alpha_v\alpha_i - yV_s/\alpha_i\beta \quad (2.11)$$

$$I_s = I'/\beta = yV_p/\alpha_v\beta - yV_s/\beta^2 \quad (2.12)$$

ดังนั้น รูปแบบทั่วไปของ single-phase admittance ของหม้อแปลง รวมทั้ง phase-setting เป็น

$$[y] = \begin{bmatrix} \frac{y}{\alpha_i\alpha_v} & \frac{-y}{\alpha_i\beta} \\ \frac{-y}{\alpha_v\beta} & \frac{y}{\beta^2} \end{bmatrix} \quad (2.13)$$

3. Generation and Consumption Modelling.

ลักษณะของการผลิตกำลังไฟฟ้า และการใช้กำลังไฟฟ้า จะแสดงในลักษณะกำลังไฟฟ้าที่ไหลเข้าไปยัง nodes ทางไฟฟ้า การไหลของกำลังไฟฟ้าจะเป็นบวก หรือลบ ขึ้นอยู่กับว่าเป็นการผลิตกำลังไฟฟ้า หรือการใช้กำลังไฟฟ้า ตามลำดับ ผลรวมทางพีชคณิตของ active หรือ reactive powers ที่ไหลเข้าไปยัง node i จะถูกแสดงด้วย Pi และ Qi ตามลำดับ

บทที่ 3

Network Calculations.

1. Basic Nodal Method.

ในการประยุกต์ใช้วิธี node กับระบบไฟฟ้ากำลัง ตัวแปรของแต่ละ node จะอยู่ในรูปเชิงซ้อน ไม่ว่าจะเป็นแรงดันหรือกระแส โดยมีการเทียบกับ node (busbar) อ้างอิง โดยตามจริงแล้ว ค่าอ้างอิงที่ใช้จะแตกต่างกันมี 2 ค่า คือ ขนาดของแรงดันอ้างอิงกับกราวด์ และสำหรับมุมของแรงดันอ้างอิงที่ได้เลือกไว้ที่บัสบาร์ใดบัสบาร์หนึ่งแล้ว โดยกำหนดค่าให้เป็นศูนย์ กระแสที่ node คือ กระแสสุทธิที่ไหลเข้ามาในระบบที่ node นั้น ๆ จากแหล่งจ่าย และ/หรือ โหลดภายนอกของระบบ จากที่ได้ให้ความหมายไปนี้ กระแสที่ไหลเข้าระบบ (จากแหล่งจ่าย) จะมีเครื่องหมายเป็นบวก ในขณะที่กระแสที่ไหลออกจากระบบ (ไปยังโหลด) มีค่าเป็นลบ และกระแสสุทธิที่พุ่งเข้ามายัง node คือผลรวมพีชคณิตของค่าเหล่านี้ หรืออาจจะกล่าวอีกลักษณะหนึ่งคือ กำลังไฟฟ้าที่ไหลเข้า node เป็น S = P + jQ นั่นเอง ดังในรูปที่ 3.1 เป็นการแสดงกระแส ,แรงดัน และกำลังไฟฟ้าที่ node ในระบบง่าย ๆ

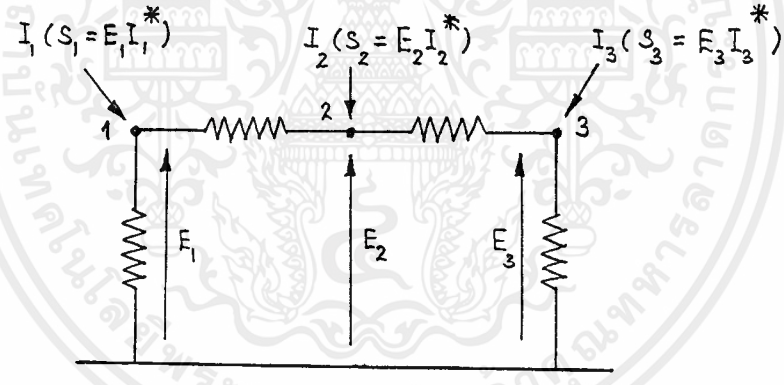


Fig 3.1 Simple network showing nodal quantities.

จากวิธี node จะมีความสะดวกในการใช้ branch admittances มากกว่าใช้ impedances โดยแรงดันที่ nodes k และ i เป็น E_k และ E_i ตามลำดับ และ admittance ของ branch ที่อยู่ระหว่าง nodes k และ i เป็น y_{ki} จะได้กระแสที่ไหลใน brance นี้ จาก node k ไปยัง node i เป็น

$$I_{ki} = y_{ki}(E_k - E_i) \tag{3.1}$$

ให้ nodes ในระบบมีหมายเลขเป็น 0,1,...,n เมื่อ 0 กำหนดให้เป็น node อ้างอิง (ground) โดยใช้ Kirchhoff's current law กระแสที่ไหลเข้า I_k ต้องมีค่าเท่ากับผลรวมของกระแสที่ไหลออกจาก node k นั่นคือ

$$I_k = \sum_{i=0}^{n} I_{ki} = \sum_{i=0}^{n} y_{ki}(E_k - E_i) \tag{3.2}$$

เมื่อ $E_0 = 0$ และระบบเป็นเชิงเส้น จะได้

$$I_k = \sum_{i=0=k}^{n} y_{ki}E_k - \sum_{i=1\neq k}^{n} y_{ki}E_i \qquad (3.3)$$

ถ้าสมการนี้ถูกเขียนสำหรับ node ทุก node ยกเว้น node อ้างอิง หรือสำหรับบัสบาร์ทั้งหมดในกรณีของระบบไฟฟ้ากำลัง เราสามารถแสดงสมการที่สมบูรณ์ได้ในรูปแบบของเมตริกซ์ ดังนี้

$$\begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_n \end{bmatrix} = \begin{bmatrix} Y_{11} & Y_{12} & \cdots & Y_{1n} \\ Y_{21} & Y_{22} & \cdots & Y_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ Y_{n1} & Y_{n2} & \cdots & Y_{nn} \end{bmatrix} \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_n \end{bmatrix} \qquad (3.4)$$

เมื่อ $Y_{kk} = \sum_{i=1\neq k}^{n} Y_{ki}$ = self-admittance ของ node k

$Y_{ki} = -y_{ki}$ = mutual-admittance ระหว่าง node k และ i

จากสมการที่ (3.4) เราสามารถเขียนในรูปง่าย ๆ ได้เป็น

$$I = YE \qquad (3.5)$$

หรือในรูปของผลรวม จะได้

$$I_k = \sum_{i=1}^{n} Y_{ki}E_i \qquad \text{สำหรับ } i = 1,2,\ldots,n \qquad (3.6)$$

เมตริกซ์ node admittance ในสมการที่ (3.5) หรือ (3.6) สามารถกำหนดโครงสร้างได้เพื่อความง่ายในการสร้าง มีคุณสมบัติตามนี้

-เป็นเมตริกซ์จัตุรัส มิติ $n \times n$

-มีความสมมาตร เนื่องจาก $y_{ki} = y_{ik}$

-เป็นเมตริกซ์เชิงซ้อน

-สมาชิกแต่ละตัวนอกแนวทแยงมุมหลัก y_{ki} เป็นค่าลบของ branch admittance ระหว่าง node k และ i และมักจะมีค่าเป็นศูนย์

-สมาชิกแต่ละตัวในแนวทแยงมุมหลัก y_{kk} เป็นผลรวมของ admittance ของ branch ซึ่งมีขั้วหนึ่งอยู่บน node k รวมทั้ง branch ที่ต่อกับกราวด์ด้วย

2. The Bus Admittance and Impedance Matrices.

จาก bus admittance matrix Y_{bus} เมื่อเราทำการ invert จะได้ผลเป็น bus impedance matrix Z_{bus} ดังสมการ

$$Z_{bus} = Y_{bus}^{-1} \qquad (3.7)$$

และสำหรับ network ที่ประกอบไปด้วย 3 nodes ที่แยกกัน จะได้

$$Z_{bus} = \begin{bmatrix} Z_{11} & Z_{12} & Z_{13} \\ Z_{21} & Z_{22} & Z_{23} \\ Z_{31} & Z_{32} & Z_{33} \end{bmatrix} \quad (3.8)$$

เมื่อ Y_{bus} มีความสมมาตรรอบแนวทแยงมุมหลัก ดังนั้น Z_{bus} จึงต้องมีความสมมาตรด้วยเช่นกัน

impedance แต่ละตัวที่เป็นสมาชิกของ Z_{bus} ในแนวทแยงมุมหลักจะเรียกเป็น drivingpoint impedance of the nodes และสมาชิกนอกแนวทแยงมุมหลักจะเรียกเป็น transfer impedances of the nodes

จากสมการ node equations ที่ว่า

$$I = Y_{bus}E \quad (3.9)$$

เราสามารถแก้สมการโดยการคูณทั้งสองข้างของสมการที่ (3.9) ด้วย $Y_{bus}^{-1} = Z_{bus}$ จะได้

$$E = Z_{bus}I \quad (3.10)$$

เราต้องพึงระลึกไว้เสมอว่าเมื่อพิจารณาเป็น Z_{bus} นั้น E และ I เป็น column matrices ของแรงดันที่ node และ กระแสที่ไหลเข้า node จากแหล่งจ่ายกระแส ตามลำดับ เมื่อทำการขยายสมการที่ (3.10) สำหรับ network ที่ประกอบไปด้วย nodes ที่แยกกัน n nodes จะได้

$$\begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_n \end{bmatrix} = \begin{bmatrix} Z_{11} & Z_{12} & \cdots & Z_{1r} \\ Z_{21} & Z_{22} & \cdots & Z_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ Z_{n1} & Z_{n2} & \cdots & Z_{nn} \end{bmatrix} \cdot \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_n \end{bmatrix} \quad (3.11)$$

3. Modification of an Existing Bus Impedance Matrix.

ในการสร้าง matrix Z_{bus} $[Z_{bus}]$ โดยวิธีการเพิ่ม impedance ทีละค่าในระบบ เราสามารถแยกพิจารณาออกได้เป็น 4 กรณี ดังนี้

<u>กรณีที่ 1</u> : การเพิ่ม Z_b ระหว่างบัสใหม่ p กับบัสอ้างอิง เราสามารถทำการปรับปรุง matrix เดิมได้ดังนี้

$$\begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_n \\ E_p \end{bmatrix} = \begin{bmatrix} & & & & 0 \\ & Z_{orig} & & & 0 \\ & & & & \vdots \\ & & & & 0 \\ 0 & 0 & \cdots & 0 & Z_b \end{bmatrix} \cdot \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_n \\ I_p \end{bmatrix} \quad (3.12)$$

$Z_{bus(new)}$

กรณีที่ 2 : เพิ่ม Z_b จากบัสใหม่ p ไปยังบัสที่มีอยู่แล้ว k ดังแสดงในรูปที่ 3.2

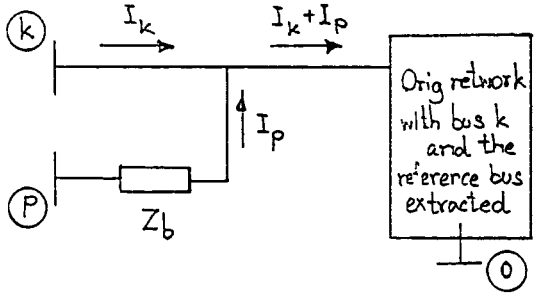


Fig 3.2 Addition of new bus p connected through impedance Z_b to existing bus k.

เราสามารถเพิ่มเติมสมการใน matrix เดิมได้ ดังนี้

$$\begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_n \\ \cdots \\ E_p \end{bmatrix} = \left[\begin{array}{ccc:c} & Z_{orig} & & \begin{matrix} Z_{1k} \\ Z_{2k} \\ \vdots \\ Z_{nk} \end{matrix} \\ \hdashline Z_{k1} & Z_{k2} \cdots & Z_{kn} & Z_{kk}+Z_b \end{array}\right] \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_n \\ \cdots \\ I_b \end{bmatrix} \qquad (3.13)$$

$Z_{bus(new)}$

กรณีที่ 3 : เพิ่ม Z_b จากบัสที่มีอยู่แล้ว k ไปยังบัสอ้างอิง เปรียบได้กับการเพิ่ม Z_b ของในกรณีที่ 2 แต่บัส p ทำการต่ออยู่กับบัสอ้างอิง จึงทำให้ E_p เป็นศูนย์ และเมื่อเราทำการเพิ่มสมการใน matrix เดิมแล้ว ดังกรณีที่ 2 เราต้องทำการกำจัดแถวและหลักที่ (n+1) ออกให้มีมิติเท่ากับ matrix เดิม โดยใช้สมการที่ (3.14)

$$Z_{hi(new)} = Z_{hi(orig)} - \frac{Z_{h(n+1)} Z_{(n+1)i}}{Z_{kk}+Z_b} \qquad (3.14)$$

กรณีที่ 4 : เพิ่ม Z_b ระหว่างสองบัส j และ k ที่มีอยู่แล้ว ดังแสดงในรูปที่ 3.3

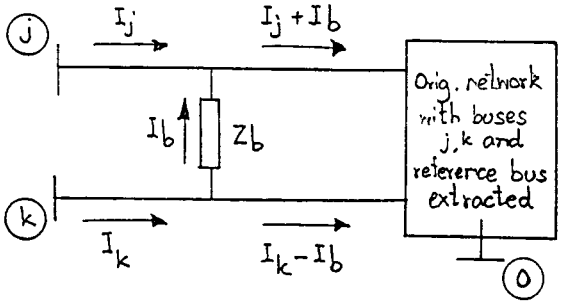


Fig 3.3 Addition of impedance Z_b between existing buses j and k.

เราสามารถเพิ่มเติมสมการใน matrix เดิมได้ดังนี้

$$\begin{bmatrix} E_1 \\ \vdots \\ E_j \\ E_k \\ \vdots \\ E_n \\ 0 \end{bmatrix} = \left[\begin{array}{ccc|c} & Z_{orig} & & \begin{matrix} Z_{1j}-Z_{1k} \\ \vdots \\ Z_{jj}-Z_{jk} \\ Z_{kj}-Z_{kk} \\ \vdots \\ Z_{nj}-Z_{nk} \end{matrix} \\ \hline (Z_{j1}-Z_{k1}) & \cdots\ (Z_{kj}-Z_{kk}) \ \cdots & & Z_{bb} \end{array}\right] \begin{bmatrix} I_1 \\ \vdots \\ I_j \\ I_k \\ \vdots \\ I_n \\ I_b \end{bmatrix} \qquad (3.15)$$

$Z_{bus(new)}$

โดย $Z_{bb} = Z_b + Z_{jj} + Z_{kk} - 2Z_{jk}$

จากนั้นต้องทำการกำจัดแถวและหลักที่ (n+1) ออกให้มีมิติเท่ากับ matrix เดิม โดยใช้สมการที่ (3.16)

$$Z_{hi(new)} = Z_{hi(orig)} - \frac{Z_{h(n+1)} Z_{(n+1)i}}{Z_b + Z_{jj} + Z_{kk} - 2Z_{jk}} \qquad (3.16)$$

บทที่ 4

Load Flow Studies.

ในระบบไฟฟ้ากำลัง กำลังไฟฟ้าไหลจากศูนย์กลางการผลิตไฟฟ้าไปยังศูนย์กลางโหลด ในกระบวนการนี้ ต้องมีการศึกษาวิเคราะห์ในหลายสิ่งหลายอย่าง เช่น ส่วนของ bus voltage ,การไหลของ MW และ MVAR ในสายส่งแรงสูง ,ผลของการปรับเปลี่ยนวงจร และการติดตั้งอุปกรณ์รักษาระดับแรงดันเข้าไป เป็นต้น สำหรับในสภาวะโหลดที่แตกต่างกัน ในระบบไฟฟ้ากำลังสมัยใหม่จะมีขนาดใหญ่ และซับซ้อนมาก ทำให้การวิเคราะห์ระบบควรจะทำในลักษณะของระบบจำลอง ระบบจำลองและลำดับขั้นตอนที่เกี่ยวกับการไหลของกำลังไฟฟ้านี้เองที่เรียกกันว่าการวิเคราะห์ Load-Flow

การศึกษา Load-Flow จะช่วยให้ได้คำตอบในสภาวะ steady-state ของวงจรไฟฟ้ากำลังที่สมบูรณ์ ในปัจจุบันนี้ Digital Computer มีการพัฒนาไปอย่างมาก จึงใช้ Computer ในการทำการวิเคราะห์ โดยวิธีนี้จะมีจุดมุ่งหมายไปที่ค่าของ complex voltages ของบัสต่าง ๆ ในระบบ และค่าของการไหลของ active power และ reactive power ก็สามารถพิจารณาได้จาก complex voltages นั่นเอง

การศึกษา Load-Flow จะทำระหว่างการวางแผนกับระบบใหม่ หรือการขยายระบบที่มีอยู่เดิม ซึ่งสิ่งเหล่านี้ต้องมีการคำนวณผลของสภาวะของโหลดที่แตกต่างกันในระบบเดิม และผลของ Load-Flow เหล่านี้ยังสามารถนำไปใช้วิเคราะห์กับเรื่องอื่น ๆ ได้อีก เช่น การศึกษาการลัดวงจรในระบบ ,เสถียรภาพของระบบ ,การจ่ายโหลดอย่างประหยัด เป็นต้น

1. Chassification of System Buses.

ระบบไฟฟ้ากำลังเป็นวงจร ac ที่ใช้พลังงานไฟฟ้า เห็นได้ชัดว่า ทุก ๆ node หรือ บัส ของระบบจะมีคุณลักษณะเฉพาะของ active และ reactive powers P ,Q และ complex voltage V โดย complex voltage จริง ๆ แล้ว สามารถแยกได้เป็น 2 ตัวแปร คือ ขนาดแรงดัน |V| และมุมเฟส δ ดังนั้น ในทุก ๆ บัสของระบบจึงประกอบไปด้วย 4 ตัวแปร คือ P ,Q ,|V| และ δ บัสที่ไม่ใช่ generator bus จะมีเฉพาะความต้องการใช้กำลังไฟฟ้าเท่านั้น ซึ่งเรียกกันว่าเป็น load bus จะรู้ค่าของ P และ Q อย่างชัดเจน โดย |V| และ δ จะแปรเปลี่ยนตามความต้องการกำลังไฟฟ้า ดังนั้น ค่าของ |V| และ δ จึงเป็นค่าที่ต้องการหาในบัสชนิดนี้ ส่วนบัสอีกชนิดหนึ่ง คือ บัสที่มีแหล่งกำเนิดไฟฟ้าปรากฏอยู่ เรียกว่าเป็น generator bus โดยขนาดของ |V| จะมีค่าคงที่ โดยการใช้อุปกรณ์คงค่าแรงดัน นอกจากนี้ปริมาณของการผลิต active power เราสามารถทราบได้ เนื่องจากทราบขีดจำกัดในเรื่องของการไหลของ ไอน้ำ/น้ำ ที่ใช้ในการขับ turbine ดังนั้น สำหรับ generator bus จึงทราบค่าของ P และ |V| โดยเหลือ 2 ตัวแปรที่ไม่ทราบค่าและต้องการที่จะหา

ในระบบไฟฟ้ากำลัง การผลิตกำลังไฟฟ้าทั้งหมดต้องสอดคล้องกับปริมาณความต้องการทั้งหมดของกำลังไฟฟ้า รวมกับความสูญเสียที่เกิดในสายส่ง เนื่องจากความสูญเสียในสายส่งไม่สามารถที่จะหาก่อนล่วงหน้าได้ ดังนั้น การผลิตกำลังไฟฟ้าทั้งหมดที่ต้องการจ่าย จึงไม่สามารถที่จะเจาะจงลงไปได้ ภายใต้สภาวะนี้ วิธีที่สามารถนำมาใช้ คือ ให้บัสหนึ่งในบัสที่ผลิตกำลังไฟฟ้าทั้งหมด เป็น slack bus ซึ่งสามารถที่จะระบุขนาดของแรงดัน |V| ลงไปได้ แต่ไม่มีการเจาะจงในปริมาณของการผลิตกำลังไฟฟ้า

สุดท้าย มุมเฟส δ ของบัสแต่ละบัสควรวัดโดยเทียบกับบัสอ้างอิง ซึ่ง voltage phasor ของ slack bus จะถูกใช้เป็นบัสอ้างอิง และมีมุมเฟส δ เป็นศูนย์ ดังนั้น บัสที่ใช้ในการศึกษา Load-Flow จึงสามารถแยกได้เป็น 3 ประเภท ดังนี้

1. Slack Bus : $|V|$,δ รู้ค่า P ,Q ไม่ระบุ
2. Generator Bus : P ,$|V|$ รู้ค่า Q ,δ ไม่รู้ค่า
3. Load Bus : P ,Q รู้ค่า $|V|$,δ ไม่รู้ค่า

โดย generator bus อาจเรียกได้ว่าเป็น PV-bus และ load bus อาจเรียกได้ว่าเป็น PQ-bus

2. Load-Flow Equations.

เป็นที่ทราบแล้วว่า การศึกษา Load-Flow จริง ๆ แล้วต้องการที่จะหา complex voltages V ของบัสต่าง ๆ ในระบบ ที่สภาวะโหลดต่าง ๆ ซึ่งใช้ Digital Computer ช่วยในการหาคำตอบ สมการของกำลังไฟฟ้า active และ reactive ของบัสต่าง ๆ จะอยู่ในเทอมของ complex voltages V สำหรับระบบที่มี n nodes ซึ่งไม่รวมกราวด์ จะได้จำนวนของสมการเท่ากับจำนวน node สามารถเขียนแสดงได้ ดังนี้

$$I_1 = Y_{11}V_1 + Y_{12}V_2 + Y_{13}V_3 + \ldots + Y_{1n}V_n$$
$$I_2 = Y_{21}V_1 + Y_{22}V_2 + Y_{23}V_3 + \ldots + Y_{2n}V_n$$
$$\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$$
$$I_n = Y_{n1}V_1 + Y_{n2}V_2 + Y_{n3}V_3 + \ldots + Y_{nn}V_n$$

จากสมการข้างต้น สามารถเขียนในรูปทั่ว ๆ ไปได้เป็น

$$I_i = \sum_{m=1}^{n} Y_{im}V_m \qquad , i = 1,2,\ldots,n \qquad (4.1)$$

เมื่อ I_i = กระแสเชิงซ้อนที่ไหลเข้าที่บัส i

V_m = แรงดันเชิงซ้อนเทียบกราวด์ของบัส m

Y_{im} = แอดมิตแตนซ์เชิงซ้อนระหว่างบัส i และ m ; เมื่อ i = m จะเป็น driving-point admittance ในกรณีอื่นเป็น transfer admittance

เขียนสมการที่ (4.1) ในรูปเมตริกซ์ ได้เป็น

$$I = YV \qquad (4.1a)$$

เมื่อ I เป็น column matrix ที่ประกอบไปด้วยสมาชิก $I_1, I_2, \ldots, I_n$ เช่นเดียวกันกับ V ที่เป็น column matrix ที่มีสมาชิกเป็น $V_1, V_2, \ldots, V_n$ ส่วนแอ็ดมิตแต็นซ์ทั้ง driving-point และ transfer จะบรรจุอยู่ใน admittance matrix Y

ในระบบไฟฟ้ากำลัง ค่าของ complex power มีความสำคัญมากกว่ากระแส ดังนั้น complex power ที่เข้าสู่บัส จึงสามารถเขียนได้เป็น

$$P_i + jQ_i = V_i I_i^* \qquad (4.2)$$

subscript * ในสมการข้างบนแสดงว่าเป็นการ conjugate เมื่อแทนค่าสมการที่ (4.1) ลงในสมการที่ (4.2) จะได้

$$P_i + jQ_i = V_i \sum_{m=1}^{n} Y_{im}^* V_{im}^* \qquad ,i = 1,2,\ldots,n \qquad (4.3a)$$

โดยทราบว่า $V_i = e_i + jf_i = |V_i| \angle \delta_i$

$$V_m^* = e_m - jf_m = |V_m| \angle -\delta_m$$

$$Y_{im}^* = G_{im} - jB_{im} = |Y_{im}| \angle -\theta_{im}$$

สมการที่ (4.3a) สามารถแสดงในรูปของ polar และ rectangular ได้ดังสมการที่ (4.3b) และ (4.3c) ตามลำดับ

$$P_i + jQ_i = |V_i| \sum_{m=1}^{n} |Y_{im}||V_m| e^{j(\delta i - \delta m - \theta im)} \qquad ,i = 1,2,\ldots,n \qquad (4.3b)$$

$$P_i + jQ_i = (e_i + jf_i) \sum_{m=1}^{n} (G_{im} - jB_{im})(e_m - jf_m) \qquad ,i = 1,2,\ldots,n \qquad (4.3c)$$

สำหรับแต่ละบัส จะมี 2 ตัวแปรที่ไม่ทราบค่า และสำหรับระบบที่มี n บัส ก็จะมีตัวไม่ทราบค่า 2n ตัว และ 2n สมการค่าจริง อันเป็นผลมาจากสมการเชิงซ้อน สมการที่ (4.3) แต่ในกรณีที่พิจารณาถึงตัวไม่ทราบค่าของ slack bus ไม่จำเป็นต้องมีสมการเพิ่มเติม ค่าของตัวไม่ทราบค่าสำหรับ slack bus จะหาได้โดยอัตโนมัติ เมื่อทราบค่าของตัวไม่ทราบค่าในบัสอื่น ๆ แล้ว ดังนั้นจึงสามารถตัดสมการในส่วนของ slack bus ออกไปได้ เหลือเพียง (n-1) สมการ ในสมการที่ (4.3) สำหรับการศึกษา Load-Flow โดยใช้ Digital Computer ต่อไป

3. Power-Flow Through Lines.

ภายหลังจาก complex bus voltages หาได้จากผลเฉลยของการวิเคราะห์ Load-Flow แล้ว กำลังไฟฟ้าที่ไหลใน transmission lines ก็สามารถคำนวณได้เช่นกัน จากกระแสที่ไหลใน transmission lines ซึ่งต่ออยู่ระหว่างบัส i และบัส m โดย

$$I_{im} = y_{im}(V_i - V_m) + y_{io} V_i \qquad (4.4)$$

เมื่อ y_{im} เป็น complex series admittance และ y_{io} เป็น half ของ line charging susceptance ของ transmission lines กำลังไฟฟ้าเชิงซ้อนจะไหลจากบัส i ไปยังบัส m ดังนั้น จะได้ว่า

$$P_{im} + jQ_{im} = V_i I_{im}^*$$

$$= V_i[y_{im}(V_i - V_m)]^* + V_i y_{io}^* V_i^*$$

$$= |V_i|^2 y_{im}^* - V_i V_m^* y_{im}^* + |V_i|^2 y_{io}^* \quad (4.5a)$$

เช่นเดียวกันกับกำลังไฟฟ้าที่ไหลจากบัส m ไปยังบัส i จะได้ว่า

$$P_{mi} + jQ_{mi} = V_m[y_{im}(V_m - V_i)]^* + V_m y_{mo}^* V_m^*$$

$$= |V_m|^2 y_{im}^* - V_m V_i^* y_{im}^* + |V_m|^2 y_{mo}^* \quad (4.5b)$$

4. Power-Flow Through Transformers.

ในระบบที่มีระดับ voltage แตกต่างกัน บัส 2 บัส ถูกแยกจากกันโดย transformer ที่ทำงานไม่เป็นแบบ nominal ก็เป็นแบบ off-nominal turn-ratio ในกรณีของ nominal turn-ratio transformer จะถูกแสดงด้วย leakage admittance ของตัวมันเอง และกำลังไฟฟ้าที่ไหลผ่านตัวมันก็สามารถกล่าวได้โนทำนองเดียวกับ สมการที่ (4.5) แต่ตัดเทอมที่รวมค่าของ shunt admittance ออกไป แต่ถ้าเป็นลักษณะของ off-nominal turn-ratio tranformer สามารถแสดงด้วย leakage admittance ของตัวมันเอง ต่ออนุกรมกับ ideal autotranformer ดังแสดงในรูปที่ 4.1(a) โดย leakage admittance y ของ transformer ได้ถูกอ้างอิงกับค่า base ร่วมกับบัส m ในส่วนของด้าน non-tap ของ transformer และด้าน tap ของบัส จะถูกต่อโดยตรงกับ ideal autotransformer ซึ่งมี turn-ratio a = (1 ±t) โดย t เป็นค่า per-unit tap setting

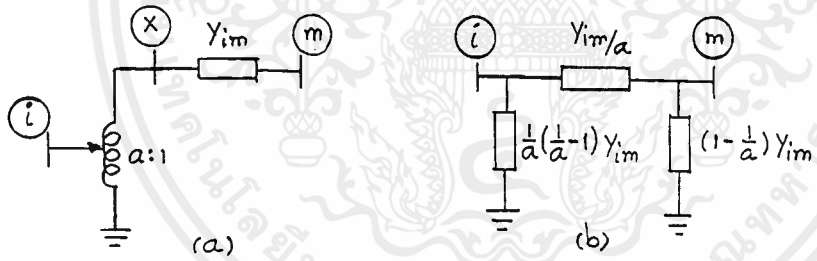


Fig 4.1 (a) Representation of Transformer Having Off-nominal Turns-ratio and (b) Its Equivalent Pi-circuit. The leakage admittance y is denoted by y_{im}.

กระแสที่ไหลใน transformer ในทิศทางจาก x ไปยัง m คือ

$$I = (V_x - V_m)y \quad (4.6)$$

และ $V_i/V_x = a$

เมื่อไม่มีกำลังไฟฟ้าสูญเสียใน autotransformer

$$V_i^* I_i = V_x^* I \quad (4.7)$$

จากที่ $$I_i = (V_x^*/V_i^*)I = (1/a)[V_i/a - V_m]y \quad (4.8a)$$

ในทำนองเดียวกัน กระแสที่ไหลจาก m ไปยัง x

$$I_m = (V_m - V_x)y$$

$$= (V_m - V_i/a)y \quad (4.8b)$$

จากสมการที่ (4.8) กระแสตามวงจรสมมูลย์ Pi ของรูปที่ 4.1(a) แสดงในรูปที่ 4.1(b) ดังนั้น driving-point admittance ของ บัส i จึงกลายเป็น y/a^2 แต่ของ บัส m ยังคงเป็น y ส่วน transfer admittances กลายเป็น -y/a ค่าของ admittances สามารถแสดงในรูปเมตริกซ์ ดังนี้

$$\begin{bmatrix} \frac{y}{a^2} & -\frac{y}{a} \\ -\frac{y}{a} & y \end{bmatrix} \tag{4.9}$$

กำลังไฟฟ้าที่ไหลจาก บัส i เป็นดังนี้

$$P_{im} + jQ_{im} = V_i[1/a(V_i/a - V_m)y]^*$$

$$= (1/a^2)|V_i|^2 y^* - (1/a)V_i V_m^* y^* \tag{4.10a}$$

และจากบัส m $\quad P_{mi} + jQ_{mi} = V_m[(V_m - V_i/a)y]^*$

$$= |V_m|^2 y^* - (1/a)V_m V_i^* y^* \tag{4.10b}$$

5. Power-Flow Through Phase Shifter.

Phase shifter เป็นตัวควบคุมการไหลของ active power โดยการปรับมุมเฟสของตัวมันเอง ในการศึกษา load-flow มันสามารถที่จะแสดงได้โดย admittance ต่ออนุกรมกับ ideal autotransformer ซึ่งมีค่าเป็น complex turn ratio ดังแสดงในรูปที่ 4.2

Fig 4.2 Representation of Phase Shifter.

จากสมการที่ (4.8a) $\quad I_i = \frac{V_x^*}{V_i^*} I = \frac{1}{a}\left(\frac{V_i}{a} - V_m\right) y$

$$= \frac{V_i}{a^2} y - \frac{V_m}{a^*} y \tag{4.11}$$

สมการที่ (4.8b) ยังไม่มีการเปลี่ยนแปลงแต่อย่างไร แต่ก็ต้องจำไว้เสมอว่า a เป็นจำนวนเชิงซ้อน ดังนั้นจากสมการที่ (4.11) และ (4.8b) ทำให้ได้ admittance matrix สำหรับ phase shifter เป็น

$$\begin{bmatrix} \frac{y}{a^2} & -\frac{y}{a^*} \\ -\frac{y}{a} & y \end{bmatrix} \tag{4.12}$$

เนื่องจาก Y เป็นลักษณะที่ไม่สมมาตร วงจรสมมูลย์จึงไม่สามารถแสดงได้ในกรณีนี้

กำลังไฟฟ้าไหลจาก บัส i เป็นดังนี้

$$P_{im} + jQ_{im} = V_i\left[\frac{V_i}{a^2} y^* - \frac{V_m}{a^*} y\right]^* = \frac{|V_i|^2}{a^2} y^* - \frac{V_i V_m^*}{a} y^* \tag{4.13a}$$

และจาก บัส m $\quad P_{mi} + jQ_{mi} = V_m\left[(V_m - \frac{V_i}{a^*}) y\right]^* = |V_m|^2 y^* - \frac{1}{a} V_m V_i^* y^* \tag{4.13b}$

บทที่ 5

Newton-Raphson Method of Solving Load-Flow.

1. Newton-Raphson Method.

วิธีของ Newton-Raphson เป็น iterative algorithm สำหรับแก้ปัญหาคับสมการที่ไม่เป็นเชิงเส้น ซึ่งมีจำนวนสมการเท่ากับจำนวนของตัวไม่ทราบค่า

$$f_k(x_m) = 0 \qquad \text{สำหรับ } k = 1 \rightarrow N \text{ และ } m = 1 \rightarrow N \tag{5.1}$$

ในแต่ละรอบ iteration ของวิธี Newton-Raphson ปัญหาที่ไม่เป็นเชิงเส้นจะถูกประมาณเทียบเคียงโดยสมการเมตริกซ์ที่เป็นเชิงเส้น การประมาณเชิงเส้น จะเป็นวิธีที่ดีที่สุดในกรณีของปัญหาตัวแปรเดียว

ในรูปที่ 5.1 x^p คือ คำตอบโดยประมาณ ซึ่งมีความผิดพลาด Δx^p ที่รอบ iteration p แล้ว

$$f(x^p + \Delta x^p) = 0 \tag{5.2}$$

สมการนี้สามารถกระจายได้โดยใช้ Taylor's theorem:

$$f(x^p + \Delta x^p) = 0$$

$$= f(x^p) + \Delta x^p f'(x^p) + (\Delta x^p)^2 f''(x^p)/2! + \ldots \tag{5.3}$$

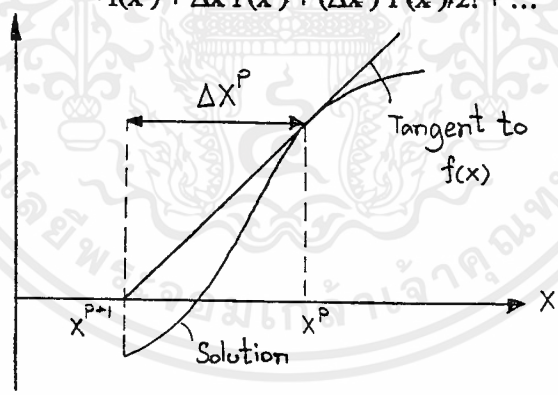


Fig 5.1 Single-variable linear approximation.

ถ้าการประเมินค่าเริ่มต้นของตัวแปร x^p ใกล้เคียงกับค่าของคำตอบ Δx^p จะมีค่าน้อย ๆ และทุกเทอมของเทอมที่มีกำลังสูงกว่า สามารถที่จะละทิ้งได้ นั่นคือ

$$f(x^p) + \Delta x^p f'(x^p) = 0 \tag{5.4}$$

หรือ

$$\Delta x^p = - f(x^p)/f'(x^p) \tag{5.5}$$

จะได้ค่าใหม่ของตัวแปรจาก

$$x^{p+1} = x^p + \Delta x^p \tag{5.6}$$

จากสมการที่ (5.4) อาจเขียนใหม่ได้เป็น

$$f(x^p) = - J\Delta x^p \tag{5.7}$$

วิธีง่าย ๆ คือ ทำการกระจาย N สมการใน N ตัวไม่ทราบค่า J เป็นเมตริกซ์จัตุรัส Jacobian matrix ของ first-order partial differentials ของฟังก์ชัน $f_k(x_m)$ สมาชิกของ [J] กำหนดโดย

$$J_{km} = \partial f_k / \partial x_m \tag{5.8}$$

และแสดงด้วยความชันของเส้นสัมผัส กับฟังก์ชันประมาณการ $f_k(x_m)$ ที่แต่ละจุดของ iteration

algorithm ของวิธี Newton-Raphson จะลู่เข้าหาแบบกำลังสอง ถ้าฟังก์ชันมีความต่อเนื่องที่อนุพันธ์อันดับที่ 1 ในตำแหน่งใกล้เคียงกับคำตอบ Jacobian matrix ไม่ได้มีเพียงหนึ่งค่า และการประมาณค่าเริ่มต้นของ x เข้าใกล้กับคำตอบที่แท้จริง แต่อย่างไรก็ตาม วิธีนี้จะไวต่อพฤติกรรมของฟังก์ชัน $f_k(x_m)$ และสูตรที่ใช้ ความเป็นเชิงเส้นมากขึ้น จะทำให้การลู่เข้าสู่คำตอบของวิธี Newton-Raphson มีความรวดเร็วและน่าเชื่อถือได้มากขึ้น ฟังก์ชันที่ไม่เรียบในช่วงที่สนใจช่วงหนึ่ง ๆ จะเป็นสาเหตุให้การลู่เข้าสู่คำตอบถูกหน่วงเวลาออกไป ความผิดพลาดรวม หรือ การลู่ผิดทิศทาง จะทำให้คำตอบที่ได้ไม่ถูกต้อง

2. Equations Relating to Power System Load-Flow.

จากสมการที่ว่า

$$I_k = \sum_{m \in k} y_{km} E_m \qquad \text{สำหรับทุก ๆ } k \tag{5.9}$$

เมื่อ I_k เป็นกระแสที่ไหลเข้าไปยังบัส k จะได้กำลังไฟฟ้าที่บัสเป็น

$$S_k = P_k + jQ_k = E_k I_k^*$$

$$= E_k \sum_{m \in k} y_{km}^* E_m^* \tag{5.10}$$

จะกล่าวกันในทางคณิตศาสตร์ สมการ Load-Flow ที่เป็นเชิงซ้อน จะไม่สามารถวิเคราะห์และทำการหาอนุพันธ์ในรูปของเชิงซ้อนได้ จากการประยุกต์ใช้วิธีของ Newton-Raphson ปัญหาจะถูกแยกเป็นสมการค่าจริง และตัวแปร polar หรือ rectangular coordinates อาจจะใช้สำหรับแก้ปัญหา bus voltages นั่นคือ เราจะได้ 2 สมการ คือ

$$P_k = P(V, \theta) \qquad \text{หรือ} \qquad P(e, f)$$

และ

$$Q_k = Q(V, \theta) \qquad \text{หรือ} \qquad Q(e, f)$$

ใน polar coordinates ส่วนจริง และส่วนจินตภาพของสมการที่ (5.10) เป็น

$$P_k = \sum_{m \in k} V_k V_m (G_{km} \cos \theta_{km} + B_{km} \sin \theta_{km}) \tag{5.11}$$

$$Q_k = \sum_{m \in k} V_k V_m (G_{km} \sin \theta_{km} - B_{km} \cos \theta_{km}) \tag{5.12}$$

เมื่อ $\theta_{km} = \theta_k - \theta_m$

ความสัมพันธ์เชิงเส้นที่ได้รับ สำหรับการเปลี่ยนแปลงน้อย ๆ ของตัวแปร θ และ V ได้จากการจัดรูปของ total differentials จะได้ผลของสมการ เป็นดังนี้

-สำหรับ PQ-busbar

$$\Delta P_k = \sum_{m \in k} (\partial P_k/\partial \theta_m)\Delta\theta_m + \sum_{m \in k} (\partial P_k/\partial V_m)\Delta V_m \quad (5.13)$$

และ

$$\Delta Q_k = \sum_{m \in k} (\partial Q_k/\partial \theta_m)\Delta\theta_m + \sum_{m \in k} (\partial Q_k/\partial V_m)\Delta V_m \quad (5.14)$$

-สำหรับ PV-busbar จะใช้เพียงสมการที่ (5.13) เท่านั้น เนื่องจากค่า Q_k ไม่ได้ถูกกำหนด

-สำหรับ slack busbar ไม่มีสมการ

ขนาดของแรงดันที่ปรากฏในสมการที่ (5.13) และ (5.14) สำหรับ PV และ slack busbars ไม่ใช่ตัวแปร แต่เป็นค่าที่ได้ระบุมา เช่นเดียวกับ θ ที่กำหนดใน slack busbar

สมการสมบูรณ์ที่ได้จะประกอบด้วย 2 ส่วนของแต่ละ PQ-busbar และแต่ละ PV-busbar ตัวแปรปัญหาคือ V และ θ สำหรับแต่ละ PQ-busbar และ θ สำหรับแต่ละ PV-busbar ดังนั้น จำนวนของตัวแปรจึงเท่ากับจำนวนของสมการ และจะได้ algorithm ของสมการที่ (5.7) เป็น

$$\begin{bmatrix} \Delta P^{p-1} \\ \Delta Q^{p-1} \end{bmatrix} = \begin{bmatrix} H^{p-1} & N^{p-1} \\ J^{p-1} & L^{p-1} \end{bmatrix} \begin{bmatrix} \Delta\theta^p \\ \Delta V^p / V^{p-1} \end{bmatrix} \quad (5.15)$$

Jacobian matrix

โดย $\Delta P_k = P_{k,spec} - P_{k,calc}$

และ $\Delta Q_k = Q_{k,spec} - Q_{k,calc}$

$P_{k,spec}$ และ $Q_{k,spec}$ ได้จากการคำนวณผลรวมของ Generation Power และ Load Power ที่ได้กำหนดมาใน ข้อมูลป้อนเข้าตั้งแต่เริ่มต้น

$P_{k,calc}$ และ $Q_{k,calc}$ ได้จากการคำนวณตามสมการที่ (5.11) และ (5.12) ตามลำดับ ในแต่ละรอบของ iterations

การหาค่าของ ΔV_i^p แต่ละตัวด้วย V_i^{p-1} จะไม่มีผลต่อการคำนวณของ algorithm แต่จะช่วยให้บางเทอมของ Jacobian matrix ง่ายขึ้น สำหรับ busbars k และ m (ไม่ใช่แถว k และหลัก m ในเมตริกซ์)

$$H_{km} = \partial P_k/\partial\theta_m = V_k V_m(G_{km}\sin\theta_{km} - B_{km}\cos\theta_{km})$$

$$N_{km} = V_m \partial P_k/\partial V_m = V_k V_m(G_{km}\cos\theta_{km} + B_{km}\sin\theta_{km})$$

$$J_{km} = \partial Q_k/\partial\theta_m = -V_k V_m(G_{km}\cos\theta_{km} + B_{km}\sin\theta_{km})$$

$$L_{km} = V_m \partial Q_m/\partial V_m = V_k V_m(G_{km}\sin\theta_{km} - B_{km}\cos\theta_{km})$$

และสำหรับ m = k

$$H_{kk} = \partial P_k/\partial\theta_k = -Q_k - B_{kk}V_k^2$$

$$N_{kk} = V_k \partial P_k/\partial V_k = P_k + G_{kk}V_k^2$$

$$J_{kk} = \partial Q_k/\partial\theta_k = P_k - G_{kk}V_k^2$$

$$L_{kk} = V_k \partial Q_k/\partial V_k = Q_k - B_{kk}V_k^2$$

จากสมการที่ (5.15) ทำการแก้สมการหาคำตอบของ $\Delta\theta$ และ ΔV โดยการ invert jacobian matrix และนำไปบวกกับค่า θ และ V ที่ได้จากการคำนวณในรอบ iteration ก่อนหน้านี้

$\theta^{(p)} = \theta^{(p-1)} + \Delta\theta$ และ $V^{(p)} = V^{(p-1)} + \Delta V$ สำหรับ PQ buses

$\theta^{(p)} = \theta^{(p-1)} + \Delta\theta$ สำหรับ PV buses

แล้วนำค่า θ และ V ที่คำนวณได้ใหม่นี้ ไปคำนวณค่า $P_{k,calc}$ และ $Q_{k,calc}$ สำหรับการเริ่มคำนวณในรอบ iteration ต่อไป กระบวนการเหล่านี้จะกระทำไปเรื่อย ๆ จนกระทั่งผลเฉลยมี tolerance อยู่ภายในเงื่อนไขของการลู่เข้าหาคำตอบของระบบที่ทำการวิเคราะห์

ในที่นี้ เราจะแสดงเฉพาะส่วนที่เป็นวิธี polar coordinates เนื่องจากวิธีนี้มีความสะดวกในการคำนวณมากกว่า rectangular coordinates และจะนำวิธี polar coordinates นี้ไปใช้ในการสร้าง algorithm และเขียนโปรแกรมวิเคราะห์ Load-Flow

สรุปวิธีของ Newton-Raphson เป็นขั้นตอน :

1. คำนวณค่าของ $P_{k,calc}$ และ $Q_{k,calc}$ ซึ่งไหลภายในระบบที่ทุก ๆ บัส สำหรับใช้คำนวณค่าขนาดของแรงดัน V_k พร้อมทั้งมุมเฟส θ_k ในรอบของการคำนวณแต่ละรอบ

2. คำนวณ ΔP ของทุก ๆ บัส

3. คำนวณค่าของ Jacobian โดยใช้ค่าประมาณ หรือค่าที่กำหนดของขนาดของแรงดัน V_k และมุมเฟส θ_k ในสมการสำหรับการหาค่า patial derivatives โดยการ differentiate ของสมการที่ (5.11) และ (5.12)

4. invert jacobian และคำนวณค่า $\Delta\theta_k$ และ ΔV_k ที่ทุกบัส

5. คำนวณค่าใหม่ของ θ_k และ V_k โดยการบวกด้วย $\Delta\theta_k$ และ ΔV_k กับค่าเดิม

6. กลับไปยังขั้นที่ 1 และทำกระบวนการเดิมซ้ำในการหาขนาดของแรงดัน และมุมเฟส จนกระทั่ง ทุกค่าของ ΔP และ ΔQ หรือ ทุกค่าของ $\Delta\theta$ และ ΔV น้อยกว่าค่า tolerance ที่ได้ทำการเลือกเอาไว้

ค่า P และ Q ที่ slack bus และ ค่า Q ที่ voltage controlled buses สามารถคำนวณได้จากสมการที่ (5.11) และ (5.12) นอกจากนี้ Line flow สามารถที่จะคำนวณได้จากความแตกต่างใน bus voltages

จากวิธีของ Newton-Raphson ที่กล่าวมา เราสามารถแสดงเป็น Flow Diagram แบบพื้นฐานแสดง algorithm ของ Load-Flow ได้ดังรูปที่ 5.2

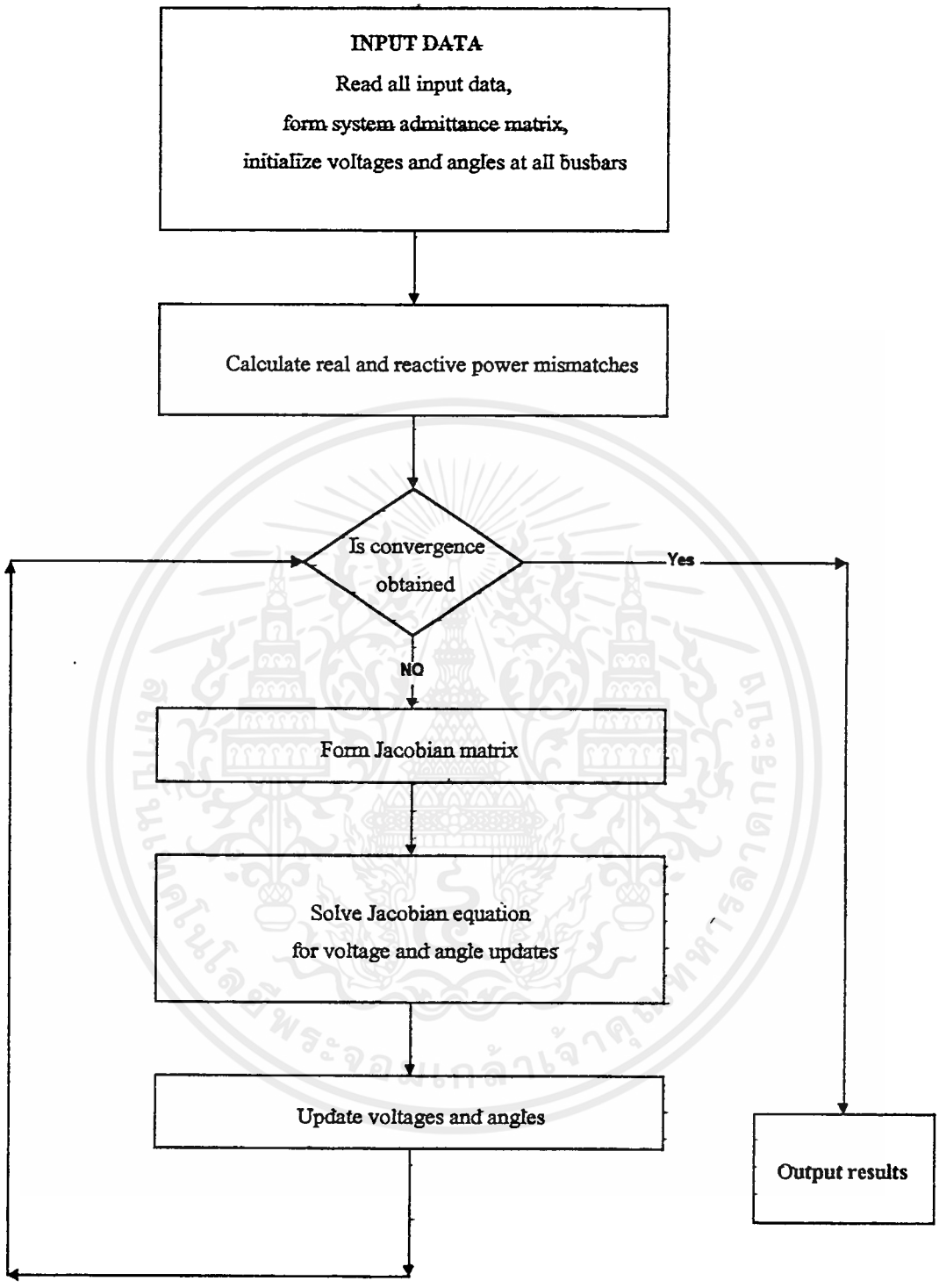


Fig 5.2 Flow diagram of the basic Newton-Raphson load-flow algorithm.

บทที่ 6

Design Load-Flow Program and Load-Forecasting Procedure.

1. Design Load-Flow Program.

จากการศึกษาพบว่า คุณสมบัติสำคัญ 5 ประการ ของวิธีวิเคราะห์ปัญหาโหลดโฟล มีดังนี้

1. ความเร็วสูง (High Computation Speed) -โดยเฉพาะในระบบขนาดใหญ่จะต้องคำนึงถึงทั้งความเร็วในการคำนวณ และความเร็วในการส่งข้อมูล (On line)

2. มีการเก็บข้อมูลน้อย (Low Computer Storage) -ในระบบขนาดใหญ่จะมีปริมาณข้อมูลมาก วิธีการวิเคราะห์ที่ดี จะมีผลต่อการลดปริมาณข้อมูลที่จะต้องใช้ในการคำนวณ

3 ความเชื่อถือได้ของผลลัพธ์ (Reliability of Solution) -วิธีการนั้นควรจะสามารถแก้ปัญหาได้แม้ในเงื่อนไขปัญหาที่อยู่นอกเหนือการศึกษา

4 สามารถปรับตัวได้ (Versatility) -ความยืดหยุ่นของการแก้ปัญหา เมื่อมีการปรับค่าต่าง ๆ ให้เปลี่ยนไป เช่น การปรับ Tap Setting ของหม้อแปลง ,การเพิ่มเติมอุปกรณ์บางอย่างเข้าไปในระบบเดิม

5 ความไม่ซับซ้อน (Simplicity) -วิธีการแก้ปัญหานั้น ๆ ควรง่ายต่อการเขียนโปรแกรมคอมพิวเตอร์

- เงื่อนไขการพิจารณาอีกส่วนหนึ่งที่ต้องคำนึงถึง ได้แก่

ความถูกต้องแม่นยำ (Accurate)	การประมาณค่า (Approximate)
ไม่สามารถปรับค่าได้ (Unadjusted)	ปรับค่าได้ (Adjusted)
ระบบปิด (Off-line)	ระบบเปิด (On-line)
การศึกษาเฉพาะกรณี (Single case)	การศึกษาหลายกรณี (Multiple cases)

ในคอลัมน์ด้านซ้าย จะแสดงถึงการวิเคราะห์โหลดโฟลที่เหมาะสมคุ้มค่า (Optimal) และใช้ในการศึกษาเกี่ยวกับเสถียรภาพของระบบ

ในคอลัมน์ด้านขวา จะแสดงถึงการประเมินค่าความปลอดภัยของระบบ

วิธีการวิเคราะห์และแก้ปัญหาโหลดโฟลที่ดี นอกจากจะมีคุณสมบัติดัง 5 ข้อข้างต้นแล้ว ยังต้องมีการผสมผสานคุณสมบัติในคอลัมน์ทั้งซ้าย และขวา เข้ากันอย่างเหมาะสม ทั้งนี้ขึ้นอยู่กับตัวระบบที่ทำการศึกษาด้วย

จาก Flow chart ในรูปที่ 6.1 เราจะทำการแบ่งงานออกเป็น 3 ส่วนใหญ่ ๆ ดังนี้

1. ส่วนรับข้อมูล (Input Data)

2. ส่วนคำนวณ Bus Admittance Matrix

3. ส่วน Iteration ทำการวิเคราะห์โหลดโฟล

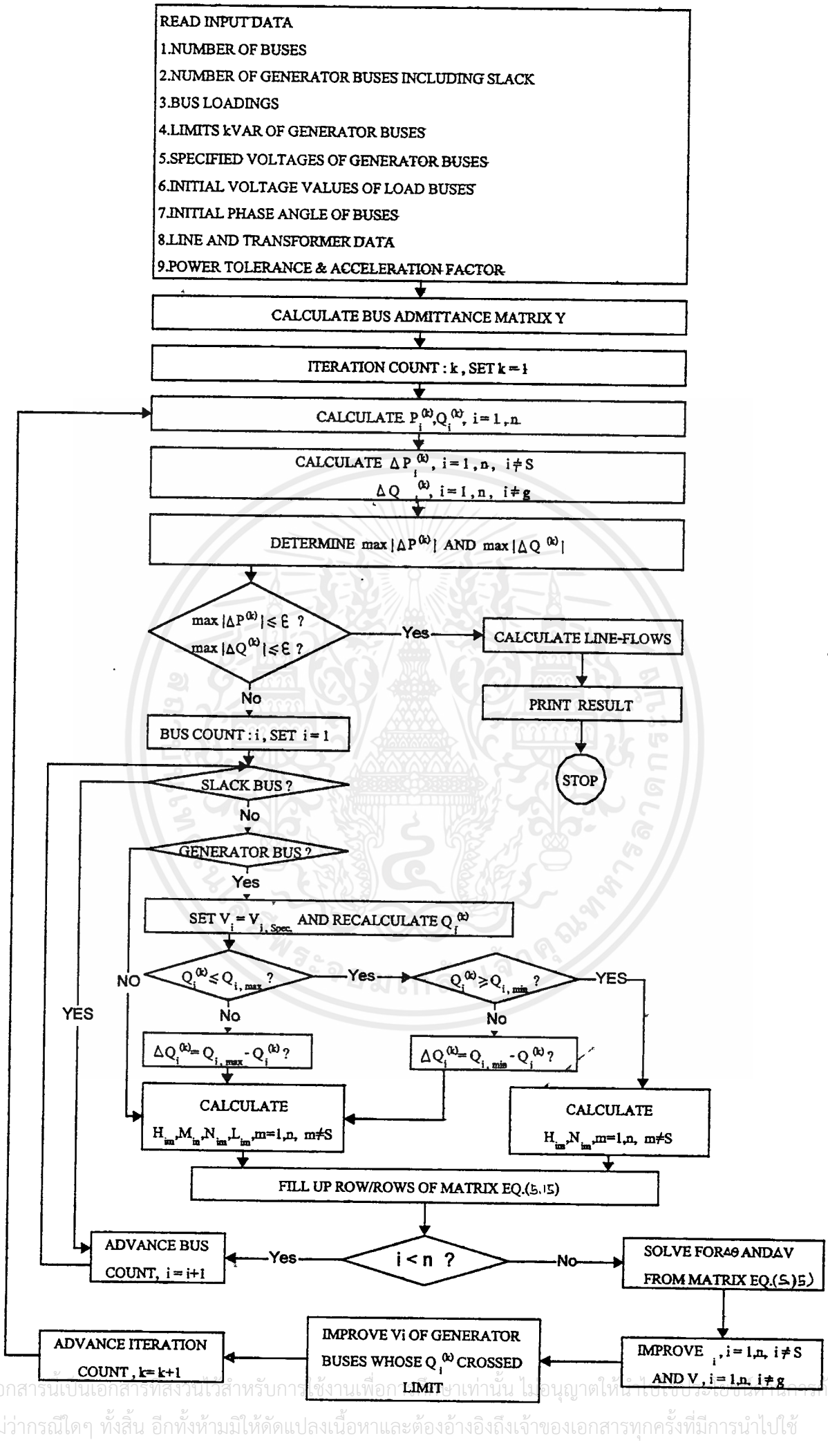


Fig 6.1 Flow Diagram Showing Load-Flow Analysis Using Newton-Raphson Method.

โปรแกรมทั้ง 3 ส่วน จะทำงานเป็นลำดับขั้นจาก 1 ถึง 3 ดังรายละเอียดต่อไปนี้

ส่วนที่ 1 : ส่วนรับข้อมูล (Input Data)

-เป็นส่วนแรกสุดของโปรแกรม ทำหน้าที่รับข้อมูลจากผู้ปฏิบัติงาน ซึ่งจะนำมาใช้ในการวิเคราะห์โหลดโฟล

ลำดับการรับข้อมูลเป็น ดังนี้

1. ข้อมูลทั่วไป (ข้อมูลพื้นฐาน)

1.1 หมายเลข Slack Bus

1.2 จำนวนการทำ Iteration สูงสุด (ใช้ป้องกันการทำงานไม่รู้จบในกรณีที่การวิเคราะห์โหลดโฟล ไม่เป็น Convergence)

1.3 ค่า Power Tolerance ใช้ในการกำหนดพิสัยของเงื่อนไขในการลู่เข้าหาผลเฉลย

1.4 จำนวนบัสในระบบ

1.5 จำนวนสายส่งในระบบ

1.6 จำนวนหม้อแปลงในระบบ

2. ข้อมูลบัส

2.1 หมายเลขบัส

2.2 ชื่อบัส

2.3 ชนิดบัส (1 = Generator Bus or 0 = Load Bus)

2.4 แรงดันที่บัส ซึ่งเป็นค่า per-unit

2.5 ค่า Load ที่บัสเป็น MW

2.6 ค่า Load ที่บัสเป็น MVAR

2.7 ค่า Power Generation ที่บัสเป็น MW

2.8 ค่า Power Generation ที่บัสเป็น MVAR

2.9 ค่า MVAR สูงสุดและต่ำสุดของบัส

2.10 ค่า Shunt Susceptance

3. ข้อมูลสายส่ง

3.1 หมายเลขสายส่ง

3.2 หมายเลขบัสต้นทาง และปลายทางที่สายส่งต่ออยู่

3.3 ค่า Resistance ,Reactance และค่า Susceptance ของสายส่งซึ่งเป็นค่า per-unit

4. ข้อมูลหม้อแปลง

4.1 หมายเลขหม้อแปลง

4.2 หมายเลขบัสต้นทาง และปลายทางที่หม้อแปลงต่ออยู่

4.3 ค่า Resistance และ Reactance ของหม้อแปลงซึ่งเป็นค่า per-unit

4.4 ค่า Tapsetting

4.5 เลขรหัสแสดงการ tap ด้านแรงสูง (1) หรือ tap ด้านแรงต่ำ (0)

ตัวโปรแกรมส่วนแรกจะทำการรับข้อมูลเข้ามา แล้วบันทึกข้อมูลเก็บไว้โดยแยกไฟล์ (file) ข้อมูลทั้งหมด 4 ส่วน ออกจากกัน เพื่อจะได้นำมาใช้งานในภายหลัง

<u>ส่วนที่ 2</u> : ส่วนคำนวณ Bus Admittance Matrix.

-ส่วนที่ 2 นี้ จะนำข้อมูลจากส่วนแรกมาใช้ในการสร้าง Bus Admittance Matrix หรือ Y-Matrix ค่าต่าง ๆ ใน Y-Matrix ได้มาจากค่า Admittance ภายในระบบ ซึ่งมี 2 ส่วน คือ ค่า Admittance ของสายส่ง และของหม้อแปลง โดยจะทำการพิจารณาสร้าง Y-Matrix จาก Admittance ของสายส่งก่อน แล้วจึงทำการพิจารณาผลของหม้อแปลง ขนาดของ Y-Matrix ที่ได้ คือ n×n (เมื่อ n คือ จำนวนบัสภายในระบบ) ซึ่งในการสร้าง Y-Matrix นี้จะยึดตามหลักทฤษฎี ดังนี้

$$[Y]=\begin{bmatrix} Y_{11} & Y_{12} & \cdots & Y_{1n} \\ Y_{21} & Y_{22} & \cdots & Y_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ Y_{n1} & Y_{n2} & \cdots & Y_{nn} \end{bmatrix}$$

เมื่อ $Y_{kk} = \sum_{i=0\neq k}^{n} y_{ki}$ = Self-admittance of node k

$Y_{ki} = -y_{ki}$ = Mutual-admittance between node k and I

<u>ส่วนที่ 3</u> : ส่วน Iteration ทำการวิเคราะห์โหลดโฟล

ในส่วนนี้ จะทำการอ้างอิง algorithm จาก Flow diagram รูปที่ 6.1 มาใช้แสดงเป็นขั้นตอนดังนี้ (ต่อจากส่วนของการสร้าง Y-Matrix)

-ทำการกำหนดตัวแปรที่ใช้ในการนับรอบ iteration (k) และ set ค่าเริ่มต้นเป็น 1

-เริ่มเข้าสู่ Loop of Iteration รอบ k ใด ๆ ให้ทำการคำนวณค่าของ $P_i^{(k)}$ และ $Q_i^{(k)}$ โดย i คือตัวแปรชี้หมายเลขบัส และ i = 1,2,...,n โดย

$$P_i^{(k)} = \sum_{m\in i} V_i V_m (G_{im}\cos\theta_{im} + B_{im}\sin\theta_{im})$$

$$Q_i^{(k)} = \sum_{m\in i} V_i V_m (G_{im}\sin\theta_{im} - B_{im}\cos\theta_{im})$$

เมื่อ $\theta_{im} = \theta_i - \theta_m$ m ชี้หมายเลขบัส และ m = 1,2,...,n

-ทำการคำนวณค่า Power Mismatch $\Delta P_i^{(k)}$ และ $\Delta Q_i^{(k)}$ โดย i = 1,2,..,n และ i ≠ slack bus

$$\Delta P_i^{(k)} = P_{i,spec} - P_{i,cal}$$

$$\Delta Q_i^{(k)} = Q_{i,spec} - Q_{i,cal}$$

ซึ่ง $P_{i,spec}$ และ $Q_{i,spec}$ จะเป็นค่าที่ได้จากผลรวมกำลังไฟฟ้า จากข้อมูลที่ป้อนเข้าไป

แต่ $P_{i,cal}$ และ $Q_{i,cal}$ จะได้จากการคำนวณในขั้นตอนก่อนหน้านี้

-ทำการพิจารณาหาค่า Power Mismatch ที่มีค่าสูงที่สุด เป็น $\Delta P^{(k)}_{max}$ และ $\Delta Q^{(k)}_{max}$

-ทำเงื่อนไขทดสอบการลู่เข้าหาคำตอบในพิสัยที่ต้องการ (ε = power tolerance) ทดสอบว่าทั้ง $\Delta P^{(k)}_{max}$ และ $\Delta Q^{(k)}_{max}$ มีค่าน้อยกว่าหรือเท่ากับ ε หรือไม่ ถ้าใช่ก็หมายถึงได้คำตอบที่ต้องการแล้ว ให้ทำการแสดงผลที่คำนวณได้ทั้งหมดออกมา แต่ถ้าไม่ใช่ จะดำเนินตาม Loop of Iteration ต่อไป

-ทำการกำหนดตัวแปรที่ใช้ในการนับจำนวนบัส (i) และ set ค่าเริ่มต้นที่บัส 1

-ทำการทดสอบชนิดของบัส i นั้น ๆ ถ้า i = slack bus ให้ทำการเพิ่มค่าของบัสขึ้น 1 แล้วทำการทดสอบชนิดของบัสใหม่ แต่ถ้า i ≠ slack bus ให้ไปทำในขั้นตอนต่อไป

-ทำการทดสอบชนิดของบัส i นั้น ๆ ว่าเป็น generator bus หรือไม่ ถ้าไม่ใช่ก็แสดงว่าเป็น PQ-bus ก็ให้ไปคำนวณสมาชิกของ Jacobian Matrix ในส่วนที่เกี่ยวข้องกับ PQ-bus คือ H_{im} ,J_{im} ,N_{im} และ L_{im} โดย m*= 1,2,...,n และ m ≠ slack bus แล้วนำสมาชิกเหล่านั้นไปใส่ใน Jacobian Matrix ต่อจากนั้นให้ทำการทดสอบว่า หมายเลขบัส (i) ยังน้อยกว่า จำนวนบัสทั้งหมด (n) หรือไม่ ถ้าใช่ก็ให้เพิ่มค่าของบัสขึ้น 1 และไปทำการทดสอบชนิดของบัสอีกครั้ง

ถ้าบัสที่ทดสอบเป็น generator bus ให้ไปทำการ set ค่าแรงดันที่บัส i (V_i) ให้เท่ากับแรงดันที่ระบุมาของบัส i ($V_{i,spec}$) และทำการคำนวณค่า reactive power $Q_i^{(k)}$ ใหม่อีกครั้ง เพื่อใช้ดำเนินการขั้นต่อไป

-เป็นขั้นตอนของการตรวจสอบเงื่อนไขขอบเขตของ reactive power $Q_i^{(k)}$ ของ generator bus เพื่อระบุว่าเป็น PQ-bus หรือ PV-bus ดังนี้

ทำการทดสอบ $Q_i^{(k)}$ ว่ามีค่าน้อยกว่าหรือเท่ากับ $Q_{i,max}$ ซึ่งข้อมูลระบุมาหรือไม่ ถ้าไม่แสดงว่าเป็น PQ-bus ให้กำหนดค่า $\Delta Q_i^{(k)}$ ใหม่

$$\Delta Q_i^{(k)} = Q_{i,max} - Q_i^{(k)}$$

แล้วทำการคำนวณสมาชิกของ Jacobian Matrix ในส่วนที่เกี่ยวข้องกับ PQ-bus ดังแสดงในส่วนของ PQ-bus ข้างต้น พร้อมทั้งเพิ่มค่าบัส และทำการทดสอบบัสใหม่

แต่ถ้าทดสอบแล้ว $Q_i^{(k)}$ น้อยกว่าหรือเท่ากับ $Q_{i,max}$ จริง ก็ให้ไปทดสอบอีกว่า $Q_i^{(k)}$ มีค่ามากกว่าหรือเท่ากับ $Q_{i,min}$ ที่ข้อมูลระบุมาหรือไม่ ถ้าไม่ก็แสดงว่าเป็น PQ-bus ให้กำหนดค่า $\Delta Q_i^{(k)}$ ใหม่เป็น

$$\Delta Q_i^{(k)} = Q_{i,min} - Q_i^{(k)}$$

แล้วทำการคำนวณสมาชิกของ Jacobian Matrix เหมือนกับ PQ-bus ที่ได้กล่าวไปแล้ว

แต่ถ้าทดสอบแล้ว $Q_i^{(k)}$ มากกว่าหรือเท่ากับ $Q_{i,min}$ จริง ก็แสดงว่าเป็น PV-bus ให้ทำการคำนวณสมาชิกของ Jacobian Matrix ในส่วนของ PV-bus ซึ่งประกอบไปด้วย H_{im} และ N_{im} เมื่อ m = 1,2,...,n และ m ≠ slack bus พร้อมทั้งนำค่าเหล่านี้ ไปใส่ใน Jacobian Matrix และทำการทดสอบจำนวนบัส เพื่อขึ้นรอบทดสอบชนิดบัสต่อไป

-เมื่อทำการทดสอบบัสทั้งหมด และสร้าง Jacobian Matrix ที่สมบูรณ์ได้แล้ว ก็ให้ทำการหาคำตอบของ $\Delta\theta$ และ ΔV จากสมการหลัก

$$\begin{bmatrix} \Delta\theta \\ \frac{\Delta|V|}{|V|} \end{bmatrix} = \begin{bmatrix} H & N \\ J & L \end{bmatrix}^{-1} \begin{bmatrix} \Delta P \\ \Delta Q \end{bmatrix}$$

-ทำการปรับปรุงค่ามุมเฟส θ และขนาดแรงดัน $|V|$ ของบัสแต่ละบัส ยกเว้น slack bus

-ทำการปรับปรุงค่าแรงดัน V_i ของ generator bus ที่มีค่า $Q_i^{(k)}$ crossed limit

-ทำการเพิ่มค่าตัวนับรอบ iteration (k) และขึ้นการคำนวณในรอบใหม่ จนกว่า Power Tolarance จะเป็นไปตามพิสัยที่ได้ระบุเอาไว้

-ทำการแสดงผลลัพธ์ที่ได้ทั้งหมดออกมา ซึ่งประกอบไปด้วย Bus Voltages ,Bus Voltage Angles ,Bus Active Powers และ Bus Reactive Powers

-ทำการคำนวณ Line Flows ของสายส่งแต่ละชุด เช่น สายส่งซึ่งต่ออยู่ระหว่างบัส i และบัส m โดยกำลังไฟฟ้าเชิงซ้อนจะไหลจากบัส i ไปยังบัส m ดังนั้น จะได้ว่า

$$\begin{aligned} P_{im} + jQ_{im} &= V_i I_{im}^* \\ &= V_i[y_{im}(V_i - V_m)]^* + V_i y_{io}^* V_i^* \\ &= |V_i|^2 y_{im}^* - V_i V_m^* y_{im}^* + |V_i|^2 y_{io}^* \end{aligned}$$

เช่นเดียวกันกับกำลังไฟฟ้าที่ไหลจากบัส m ไปยังบัส i จะได้ว่า

$$\begin{aligned} P_{mi} + jQ_{mi} &= V_m[y_{im}(V_m - V_i)]^* + V_m y_{mo}^* V_m^* \\ &= |V_m|^2 y_{im}^* - V_m V_i^* y_{im}^* + |V_m|^2 y_{mo}^* \end{aligned}$$

เมื่อ y_{im} เป็น complex series admittance และ y_{io} เป็น half ของ line charging susceptance ของ transmission lines

-ทำการคำนวณ Transformer Flows ของหม้อแปลงแต่ละตัว เช่น หม้อแปลงซึ่งต่ออยู่ระหว่างบัส i และบัส m โดยกำลังไฟฟ้าจะไหลจากบัส i ไปยังบัส m ดังนั้น จะได้ว่า

$$\begin{aligned} P_{im} + jQ_{im} &= V_i[1/a(V_i/a - V_m)y]^* \\ &= (1/a^2)|V_i|^2 y^* - (1/a)V_i V_m^* y^* \end{aligned}$$

และจากบัส m ไปยังบัส i

$$\begin{aligned} P_{mi} + jQ_{mi} &= V_m[(V_m - V_i/a)y]^* \\ &= |V_m|^2 y^* - (1/a)V_m V_i^* y^* \end{aligned}$$

เมื่อ a เป็น transformer tap setting และ y เป็น leakage admittance ของหม้อแปลง

-ทำการแสดงค่าของ Line Flows และ Transformer Flows ออกมา ก็เป็นอันเสร็จขั้นตอนของการแก้ปัญหาโหลดโฟล ของระบบไฟฟ้ากำลังที่นำมาวิเคราะห์

2. Load-Forecasting Procedure.

จากการศึกษาโหลดโฟล ทำให้เราสามารถวิเคราะห์หาคุณสมบัติที่สำคัญ ๆ ของระบบไฟฟ้ากำลังได้ ไม่ว่าจะเป็นค่าขนาดของแรงดันที่บัส มุมเฟสของแรงดันที่บัส กำลังไฟฟ้าที่บัส กำลังไฟฟ้าที่ไหลใน transmission lines หรือ กำลังไฟฟ้าที่ไหลผ่าน transformers ผลที่ได้เหล่านี้ล้วนแล้วแต่มีความสำคัญในการนำมาใช้ในการทำนายโหลดเป็นอย่างยิ่ง แต่ลักษณะของการทำนายโหลดจะเป็นการวิเคราะห์ระบบที่มีความเปลี่ยนแปลงไปตามช่วงเวลา เราสามารถแบ่งแยกการทำนายโหลดออกได้เป็น 2 ลักษณะด้วยกัน คือ

1. Short Term Forecasting.

เป็นการทำนายโหลดในช่วงเวลาสั้น ๆ เช่นในช่วงเวลา 24 ชั่วโมง แม้ว่าในช่วงเวลานี้ ระบบไฟฟ้าที่มีอยู่จะไม่มีการเพิ่มเติมหรือลดลงก็ตาม คือ จำนวนโรงจักรก็เหมือนเดิม สายส่งในระบบก็เหมือนเดิม แต่โหลด หรือผู้ใช้ไฟฟ้ามีความต้องการไฟฟ้าที่ขนาดต่าง ๆ กัน ตามเวลาที่เราเรียกว่า load curve ดังแสดงในรูปที่ 6.2 เป็นโหลดในลักษณะต่าง ๆ กันในช่วงเวลา 1 วัน ดังนั้นการวิเคราะห์ระบบในลักษณะการทำนายโหลดแบบนี้ จะต้องไม่มีการเปลี่ยนแปลงระบบ แต่เป็นการวิเคราะห์กับการเปลี่ยนแปลงของโหลดในระบบเพื่อพิจารณาควบคุมให้ระบบจ่ายกำลังไฟฟ้าอย่างเหมาะสมต่อไป

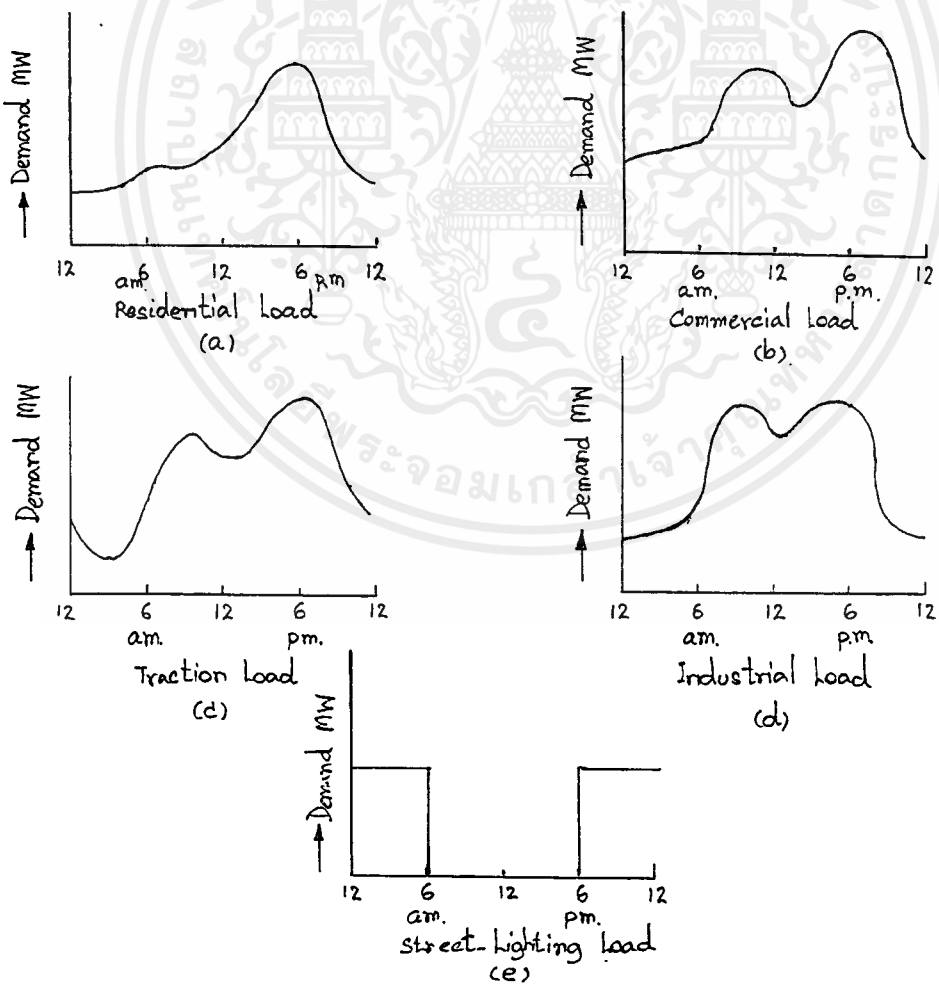


Fig 6.2 Specimen Load Curves; (a) Residential Load, (b) Commercial Load, (c) Traction Load, (d) Industrial Load, (e) Street-lighting Load.

2. Long Term Forecasting.

เป็นการทำนายโหลดในอนาคต เช่นในช่วงเวลา 5 ปีข้างหน้า ซึ่งอาจจะมีการเปลี่ยนแปลงของระบบไฟฟ้า การทำการซ่อมบำรุงหน่วยการผลิตกำลังไฟฟ้า แผนการขยายขนาดการผลิตกำลังไฟฟ้าในอนาคต หรือการตกลงซื้อขายพลังงานไฟฟ้าระหว่างองค์การผลิตไฟฟ้า ล้วนแล้วแต่เป็นแผนงานระยะยาวทั้งสิ้น ดังนั้นการวิเคราะห์ระบบในลักษณะการทำนายโหลดแบบนี้ จึงต้องมีการเปลี่ยนแปลงหน่วยในระบบ อันได้แก่ การสร้างโรงจักรไฟฟ้าใหม่หรือเป็นการเพิ่มบัสเข้าไปในระบบ การเพิ่มเติมระบบ transmission lines หรือ transformers รวมทั้งการที่มีโหลดเพิ่มขึ้น การวิเคราะห์หลังจากเปลี่ยนแปลงสมบัติของระบบไปแล้วนี้ จะทำให้ทราบว่ากำลังไฟฟ้า ศักดาไฟฟ้าและกระแสในจุดต่าง ๆ ของระบบเป็นอย่างไร ก่อนที่จะมีการตัดสินใจลงทุน และปฏิบัติงานจริง เป็นการช่วยลดความเสี่ยงลงไปได้ทางหนึ่งด้วย

จากลักษณะของการทำนายโหลดทั้ง 2 แบบ ทำให้เราได้แนวทางในการพัฒนาโปรแกรมต่อไป จากโปรแกรมวิเคราะห์โหลดโฟลปกติ เราก็จะเพิ่ม sub-routine ของการ Forecasting เข้าไป ซึ่งจะเป็นไปในลักษณะของการเข้าไปเปลี่ยนแปลงข้อมูลต่าง ๆ ในระบบเดิมที่มีอยู่แล้ว ไม่ว่าจะเป็นการปรับปรุงเปลี่ยนแปลง เพิ่มเติม หรือการตัดส่วนของระบบออก แล้วให้ย้อนกลับไปใช้การคำนวณโหลดโฟลอีกครั้ง จะทำให้ได้ผลลัพธ์จากการเปลี่ยนแปลงระบบออกมา เราจึงสามารถบอกได้ว่าผลเป็นที่น่าพอใจหรือไม่ ถ้าไม่ก็สามารถทำการออกแบบปรับปรุงได้ตลอด จนกว่าจะได้คุณสมบัติของระบบที่เราต้องการอย่างแท้จริง

สรุปแล้วส่วนของ การ Forecasting จะประกอบไปด้วย

1. การเปลี่ยนแปลงข้อมูลพื้นฐาน ข้อมูลบัส ข้อมูลสายส่ง และข้อมูลหม้อแปลง

2. การเพิ่มเติมข้อมูล อันได้แก่ การเพิ่มบัสใหม่เข้าไปในระบบ การเพิ่มสายส่ง และการเพิ่มหม้อแปลง

3. การตัดข้อมูลออก อันได้แก่ การตัดข้อมูลบัสออกจากระบบซึ่งจะมีผลกระทบกับสายส่งและหม้อแปลงที่ต่ออยู่กับบัสนั้นด้วย การตัดข้อมูลสายส่ง และการตัดข้อมูลของหม้อแปลงออกจากระบบ

จากลักษณะการทำงานทั้ง 3 ส่วนนี้ ทำให้เราสามารถออกแบบระบบได้ดังที่เราต้องการ เป็นคุณสมบัติของการ Forecasting ที่ใช้งานร่วมกับการวิเคราะห์โหลดโฟลอย่างสมบูรณ์

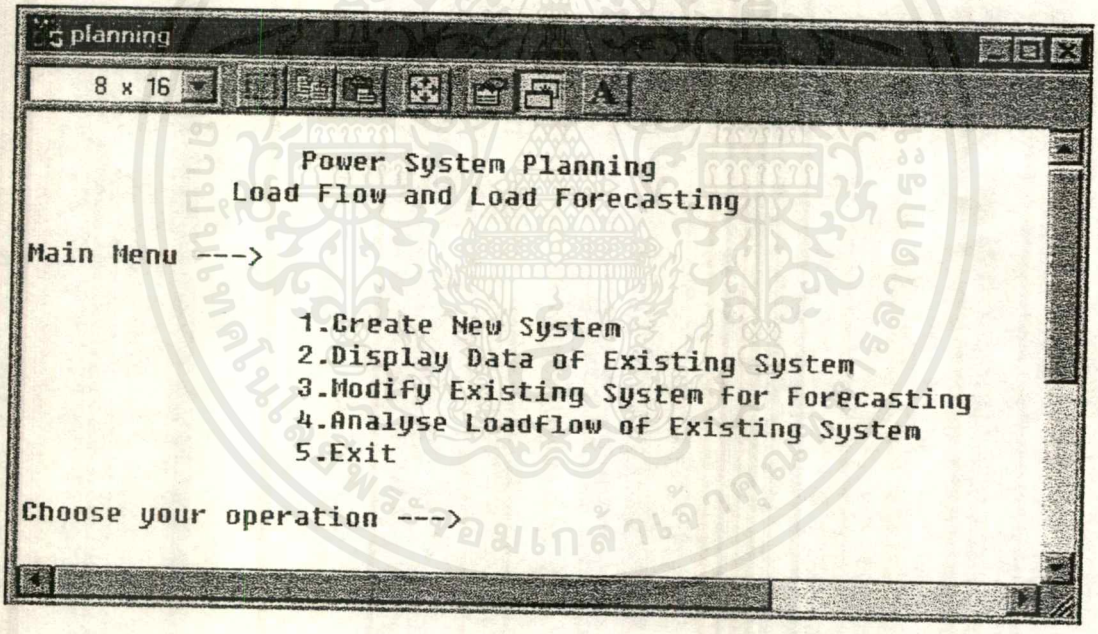
ผลการทดสอบ

ในการทดสอบการทำงานของโปรแกรม เราได้ทำการแบ่งลักษณะของการทดสอบออกเป็น 2 ลักษณะด้วยกัน คือ

1. การทดสอบความถูกต้องของโปรแกรมการวิเคราะห์โหลดโฟล (Load Flow)
2. การทดสอบการทำงานในลักษณะการปรับปรุง เปลี่ยนแปลง คุณสมบัติของระบบ (Load Forecasting)

ซึ่งข้อมูลที่นำมาใช้ในการทดสอบ แสดงในภาคผนวก โดยข้อมูลระบบ 4 บัส(beta) และระบบ 6 บัส(alpha) จะนำมาใช้ในการทดสอบในลักษณะที่ 1 ส่วนข้อมูลระบบ 16 บัส(gamma) จะนำมาใช้ในการทดสอบในลักษณะที่ 2

ลำดับแรกจะเป็นการแสดงส่วนต่าง ๆ ของ โปรแกรม ดังแสดงตามรูป ต่อไปนี้

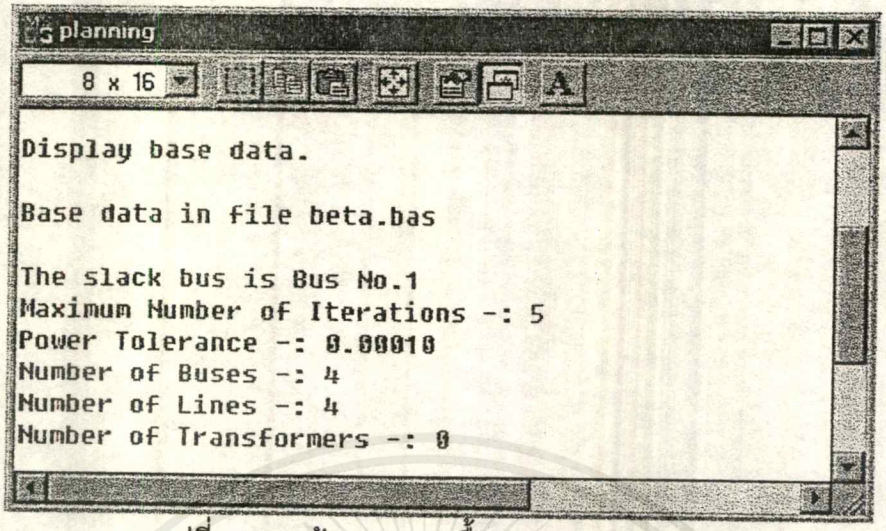


รูปที่ 1 แสดงให้เห็นเมนูหลักของโปรแกรมซึ่งประกอบไปด้วย

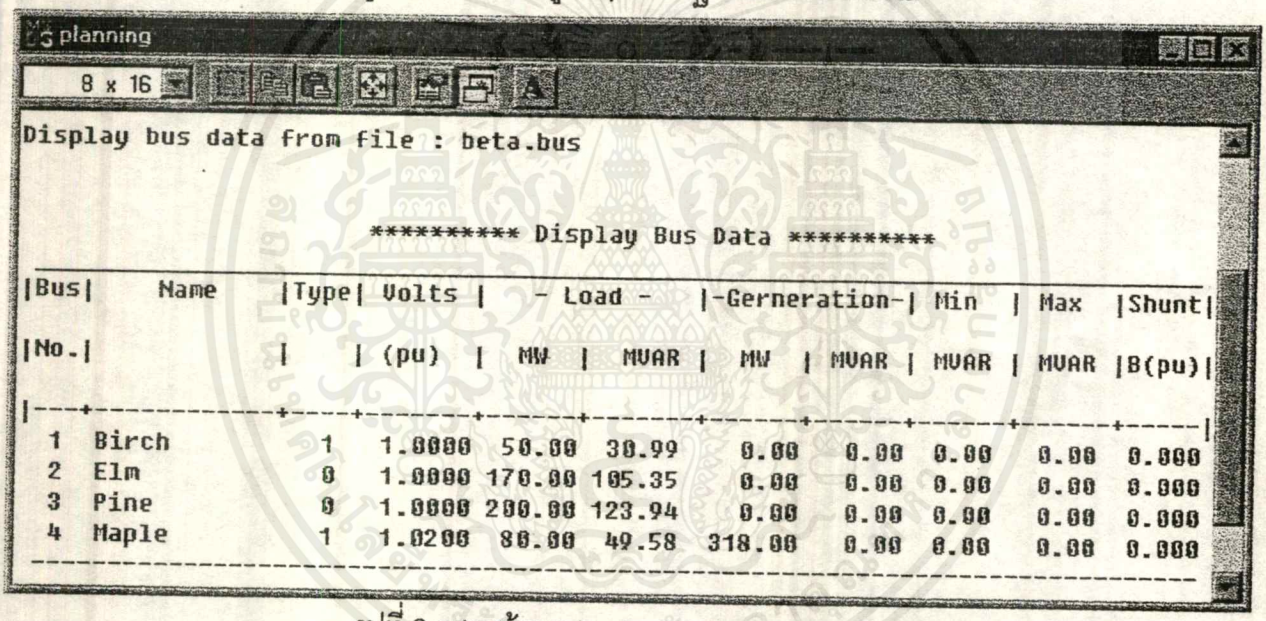
1. การป้อนข้อมูลของระบบไฟฟ้าขึ้นใหม่
2. การแสดงข้อมูลของระบบไฟฟ้าที่มีอยู่เดิม
3. การปรับปรุงระบบไฟฟ้าเดิม เพื่อใช้งานในด้าน Load Forecasting ซึ่งประกอบไปด้วย การเปลี่ยนข้อมูล การเพิ่มเติมข้อมูล และการตัดข้อมูลในระบบออก
4. การวิเคราะห์โหลดโฟล และการคำนวณ Ybus Matrix
5. ออกจากโปรแกรม

1. Load Flow.

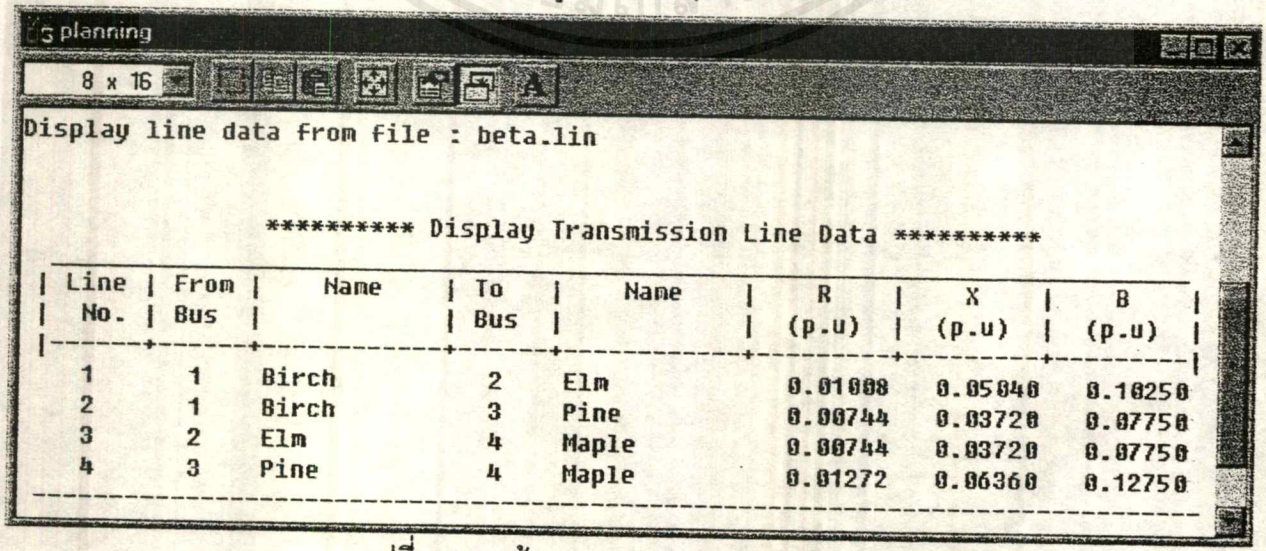
1.1 ทำการทดสอบกับระบบ beta (4 buses) ผลการทดสอบแสดงดังรูป



รูปที่ 2 แสดงข้อมูล input พื้นฐานของระบบ beta

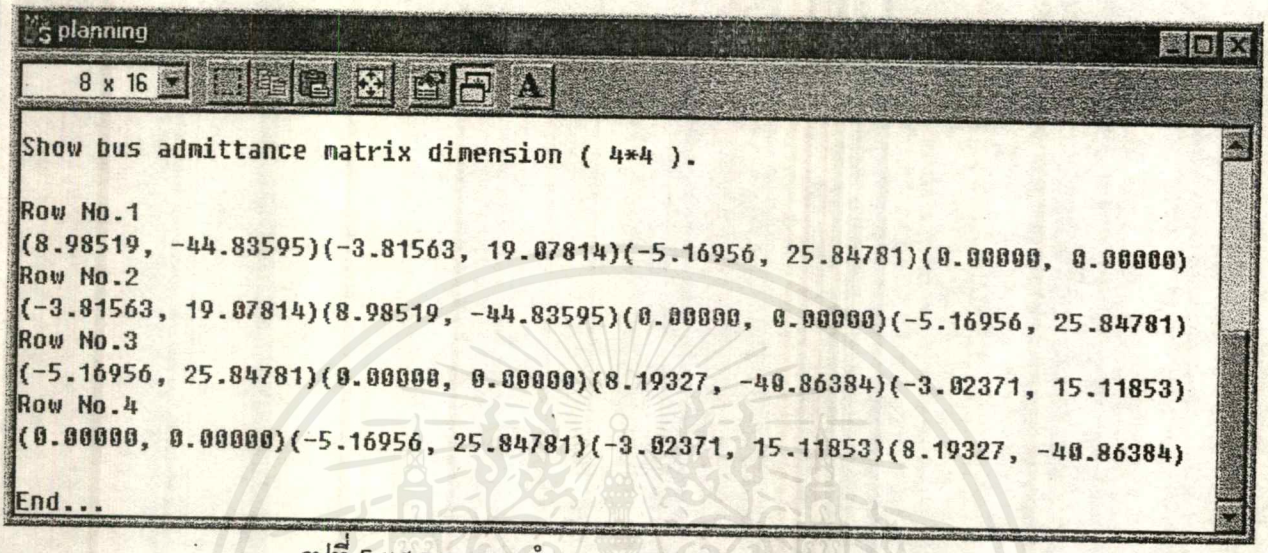


รูปที่ 3 แสดงข้อมูล buses input ของระบบ beta

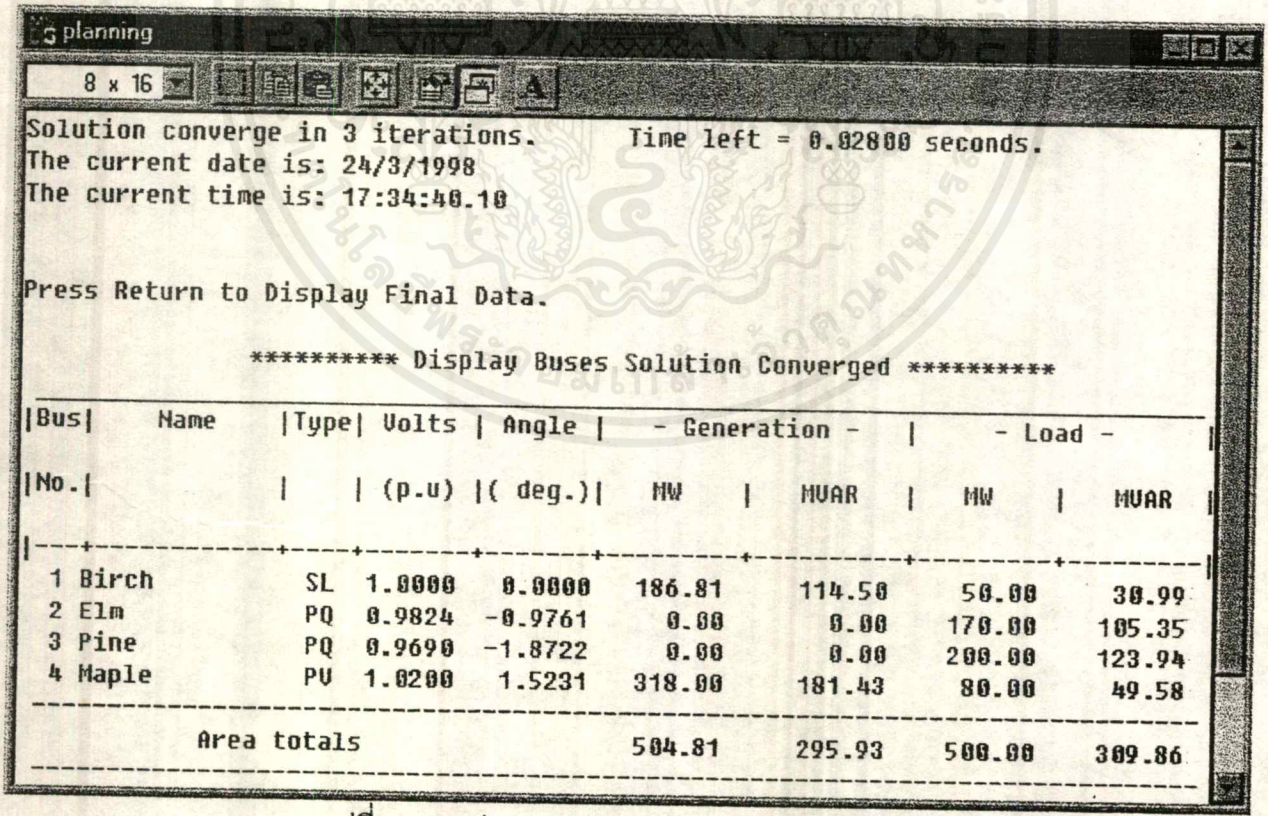


รูปที่ 4 แสดงข้อมูล lines input ของระบบ beta

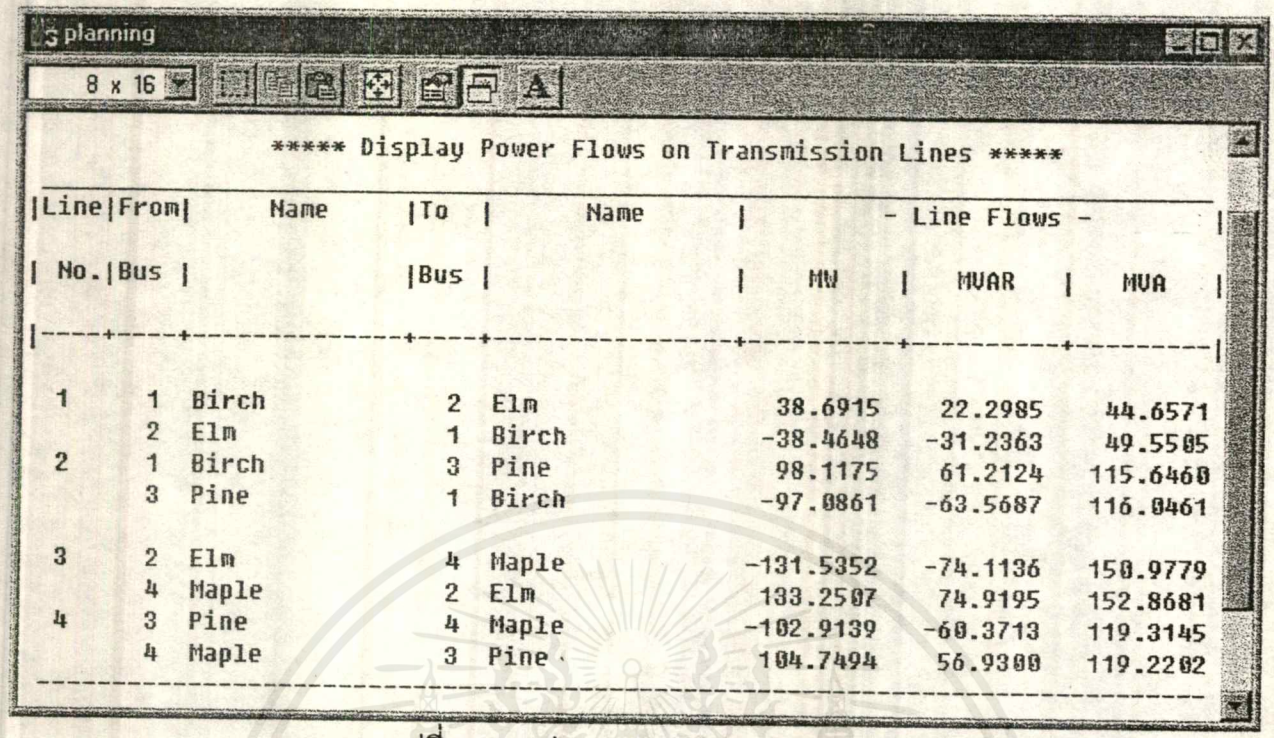
จากการวิเคราะห์โหลดโฟลของระบบ beta ทำให้เราได้ค่าของ Ybus Matrix ,final buses solution และกลศึกษาผลลัพธ์แสดงในหน้าต่างไปนี้



รูปที่ 5 แสดงผลการคำนวณ Ybus Matrix ของระบบ beta

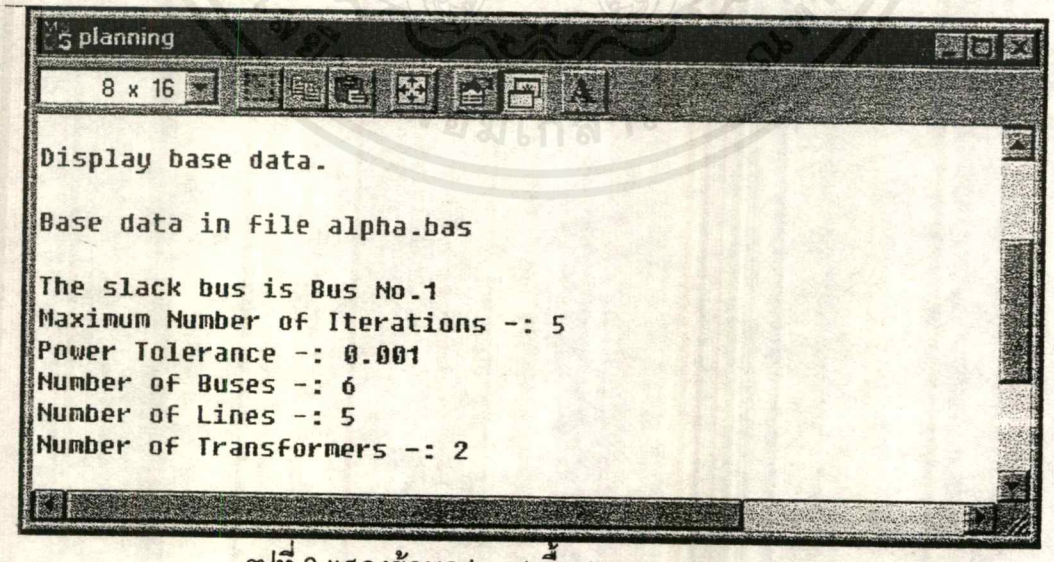


รูปที่ 6 แสดงค่า final buses solution ของระบบ beta

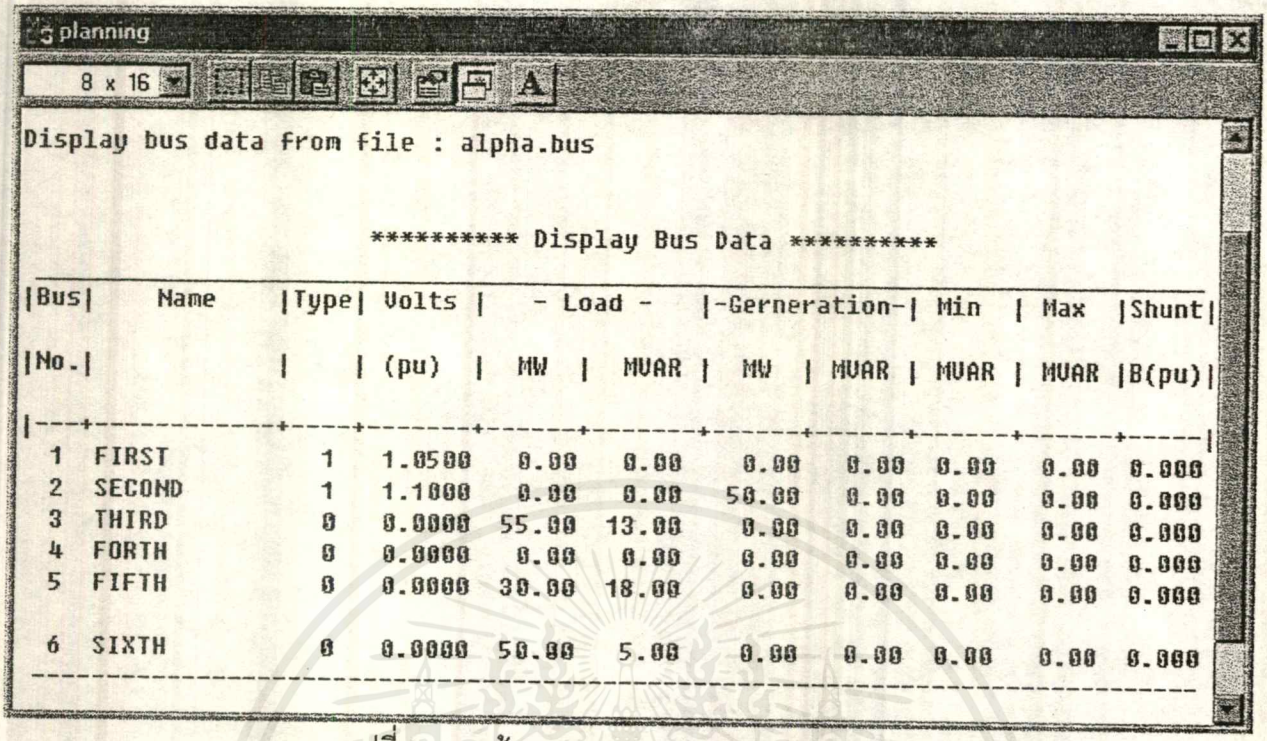


รูปที่ 7 แสดงค่า line flows ของระบบ beta

1.2 ทำการทดสอบกับระบบ alpha (6 buses) ผลการทดสอบแสดงดังรูป



รูปที่ 8 แสดงข้อมูล input พื้นฐานของระบบ alpha



s planning

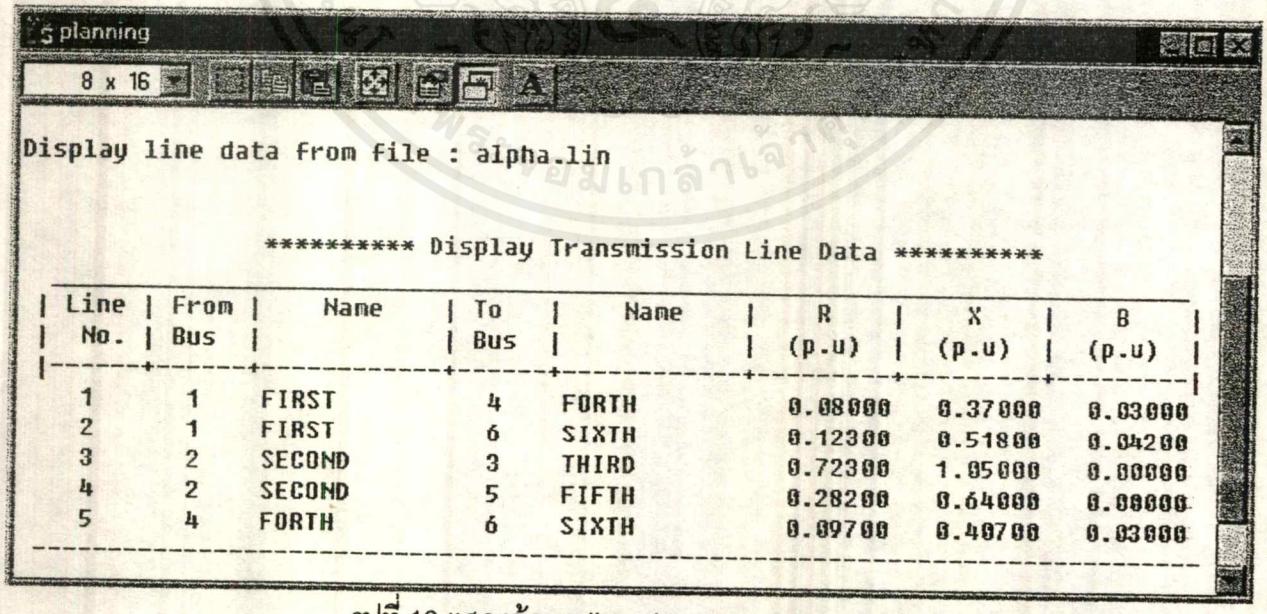
8 x 16

Display bus data from file : alpha.bus

********** Display Bus Data **********

Bus No.	Name	Type	Volts (pu)	- Load - MW	- Load - MVAR	-Gerneration- MW	-Gerneration- MVAR	Min MVAR	Max MVAR	Shunt B(pu)
1	FIRST	1	1.0500	0.00	0.00	0.00	0.00	0.00	0.00	0.000
2	SECOND	1	1.1000	0.00	0.00	50.00	0.00	0.00	0.00	0.000
3	THIRD	0	0.0000	55.00	13.00	0.00	0.00	0.00	0.00	0.000
4	FORTH	0	0.0000	0.00	0.00	0.00	0.00	0.00	0.00	0.000
5	FIFTH	0	0.0000	30.00	18.00	0.00	0.00	0.00	0.00	0.000
6	SIXTH	0	0.0000	50.00	5.00	0.00	0.00	0.00	0.00	0.000

รูปที่ 9 แสดงข้อมูล buses input ของระบบ alpha



s planning

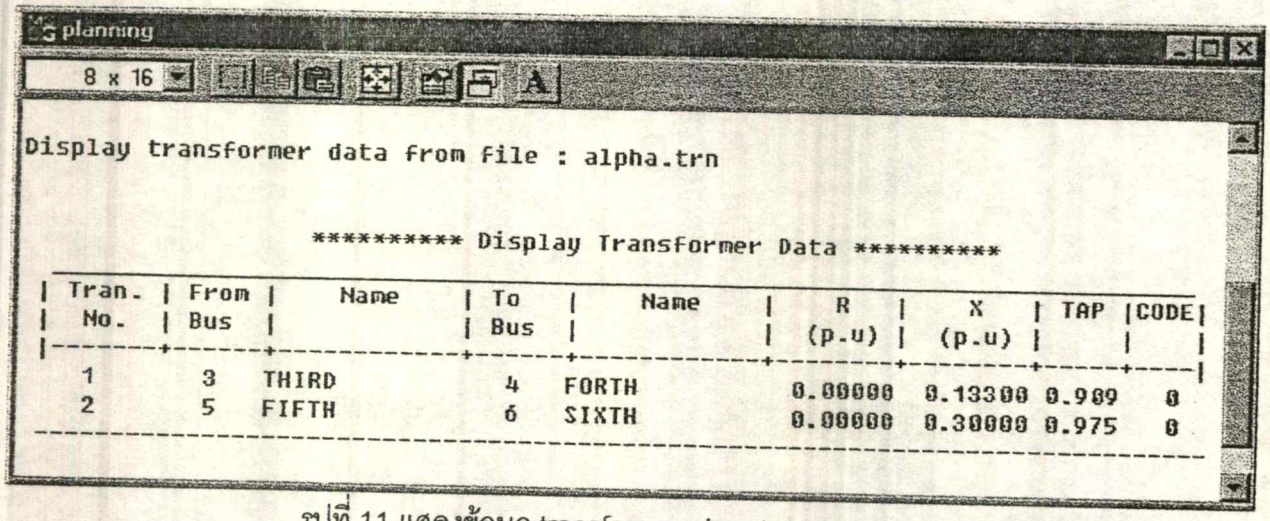
8 x 16

Display line data from file : alpha.lin

********** Display Transmission Line Data **********

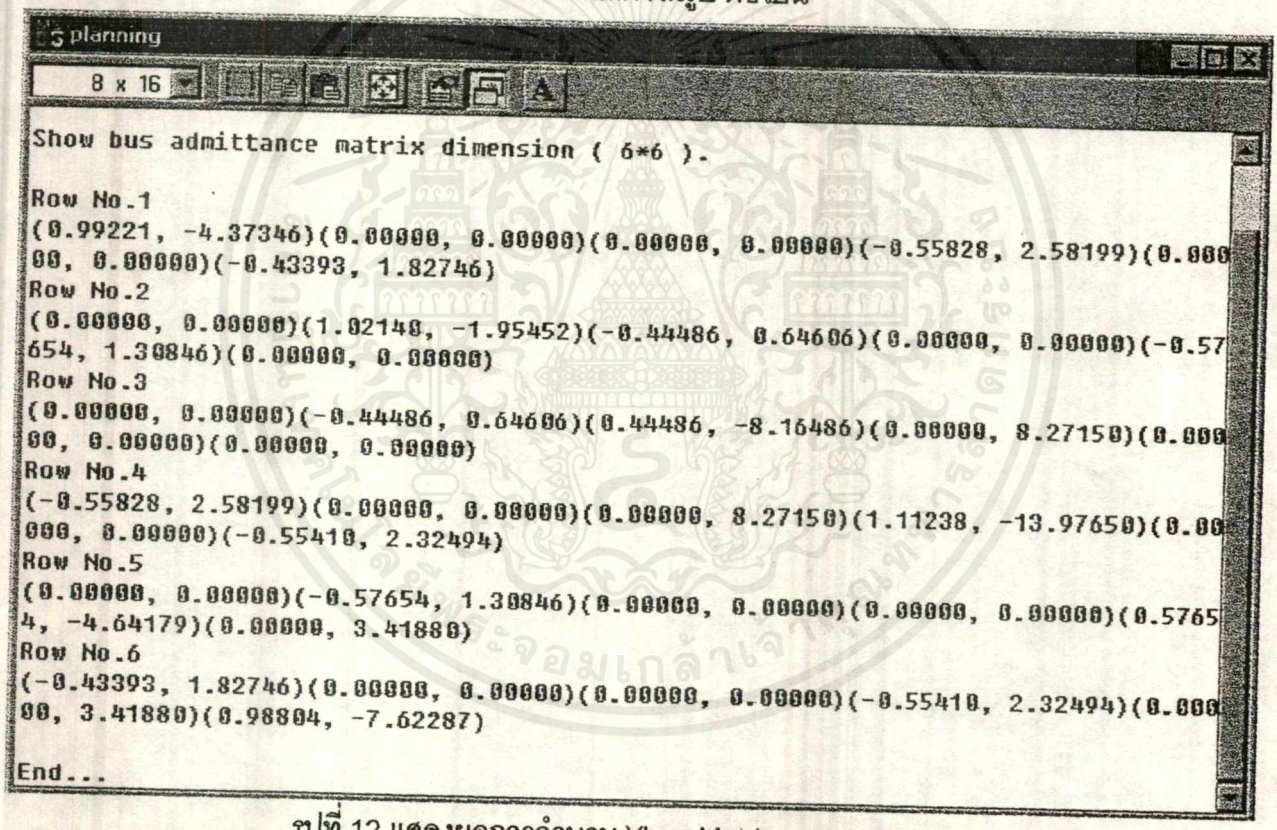
Line No.	From Bus	Name	To Bus	Name	R (p.u)	X (p.u)	B (p.u)
1	1	FIRST	4	FORTH	0.08000	0.37000	0.03000
2	1	FIRST	6	SIXTH	0.12300	0.51800	0.04200
3	2	SECOND	3	THIRD	0.72300	1.05000	0.00000
4	2	SECOND	5	FIFTH	0.28200	0.64000	0.00000
5	4	FORTH	6	SIXTH	0.09700	0.40700	0.03000

รูปที่ 10 แสดงข้อมูล lines input ของระบบ alpha

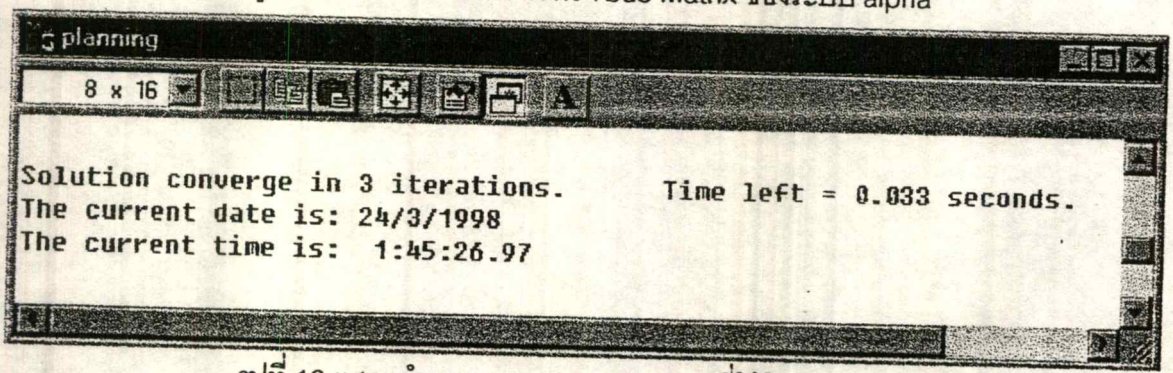


รูปที่ 11 แสดงข้อมูล transformers input ของระบบ alpha

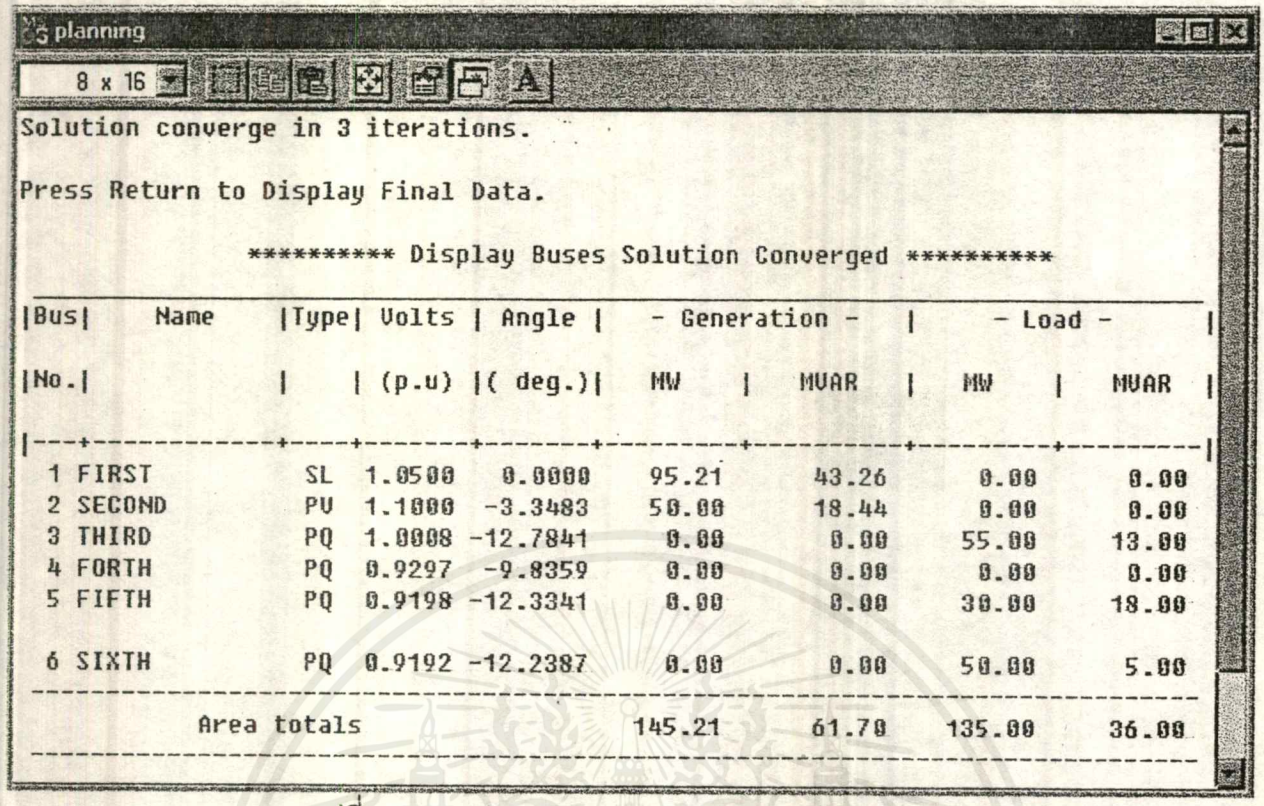
จากการวิเคราะห์โหลดโฟลของระบบ alpha ทำให้เราได้ค่าของ Ybus Matrix ,final buses solution ,line flows และ transformer flows ดังแสดงในรูป ต่อไปนี้



รูปที่ 12 แสดงผลการคำนวน Ybus Matrix ของระบบ alpha



รูปที่ 13 แสดงจำนวน iterations และเวลาที่ใช้ในการคำนวณ



planning

8 x 16

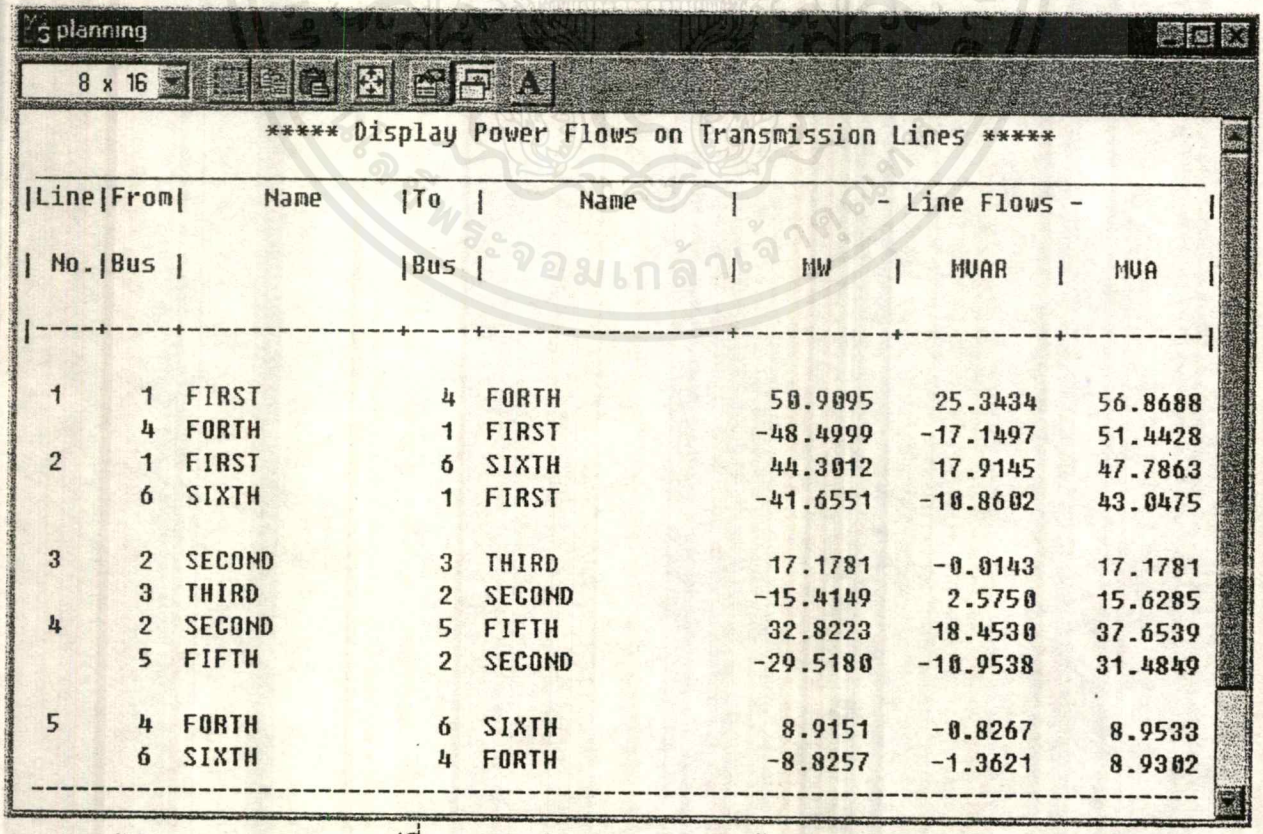
Solution converge in 3 iterations.

Press Return to Display Final Data.

********** Display Buses Solution Converged **********

Bus No.	Name	Type	Volts (p.u)	Angle (deg.)	Generation MW	Generation MVAR	Load MW	Load MVAR
1	FIRST	SL	1.0500	0.0000	95.21	43.26	0.00	0.00
2	SECOND	PV	1.1000	-3.3483	50.00	18.44	0.00	0.00
3	THIRD	PQ	1.0008	-12.7841	0.00	0.00	55.00	13.00
4	FORTH	PQ	0.9297	-9.8359	0.00	0.00	0.00	0.00
5	FIFTH	PQ	0.9198	-12.3341	0.00	0.00	30.00	18.00
6	SIXTH	PQ	0.9192	-12.2387	0.00	0.00	50.00	5.00
	Area totals				145.21	61.70	135.00	36.00

รูปที่ 14 แสดงค่า final buses solution ของระบบ alpha



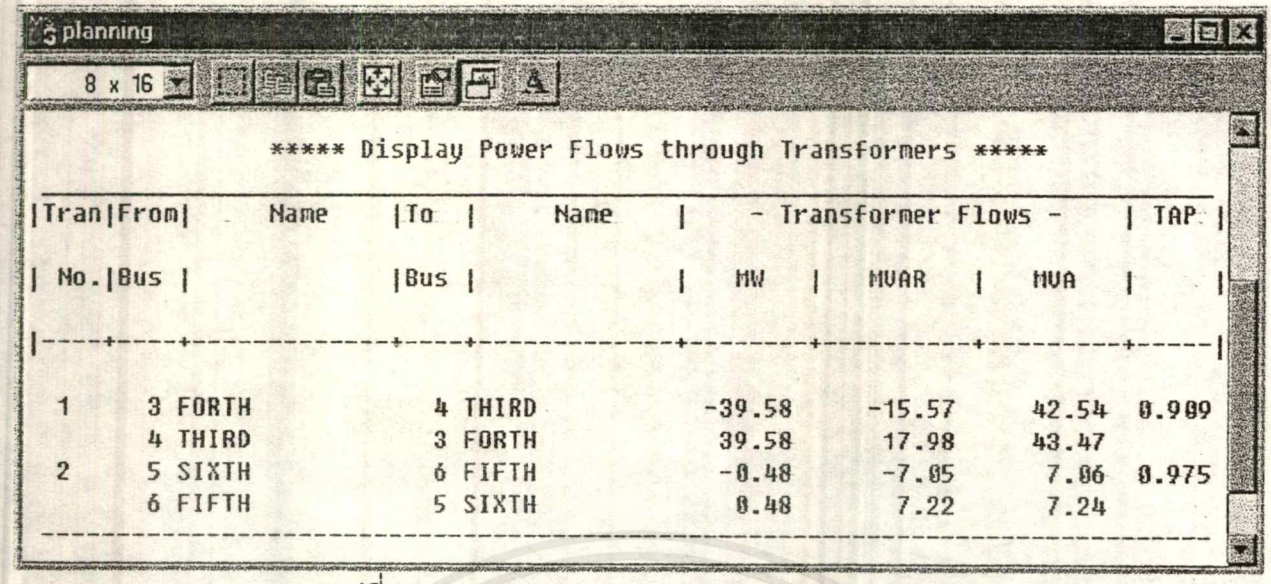
planning

8 x 16

***** Display Power Flows on Transmission Lines *****

Line No.	From Bus	Name	To Bus	Name	Line Flows MW	Line Flows MVAR	Line Flows MVA
1	1	FIRST	4	FORTH	50.9095	25.3434	56.8688
	4	FORTH	1	FIRST	-48.4999	-17.1497	51.4428
2	1	FIRST	6	SIXTH	44.3012	17.9145	47.7863
	6	SIXTH	1	FIRST	-41.6551	-16.8602	43.0475
3	2	SECOND	3	THIRD	17.1781	-0.0143	17.1781
	3	THIRD	2	SECOND	-15.4149	2.5750	15.6285
4	2	SECOND	5	FIFTH	32.8223	18.4530	37.6539
	5	FIFTH	2	SECOND	-29.5180	-16.9538	31.4849
5	4	FORTH	6	SIXTH	8.9151	-0.8267	8.9533
	6	SIXTH	4	FORTH	-8.8257	-1.3621	8.9302

รูปที่ 15 แสดงค่า line flows ของระบบ alpha



planning

8 x 16

***** Display Power Flows through Transformers *****

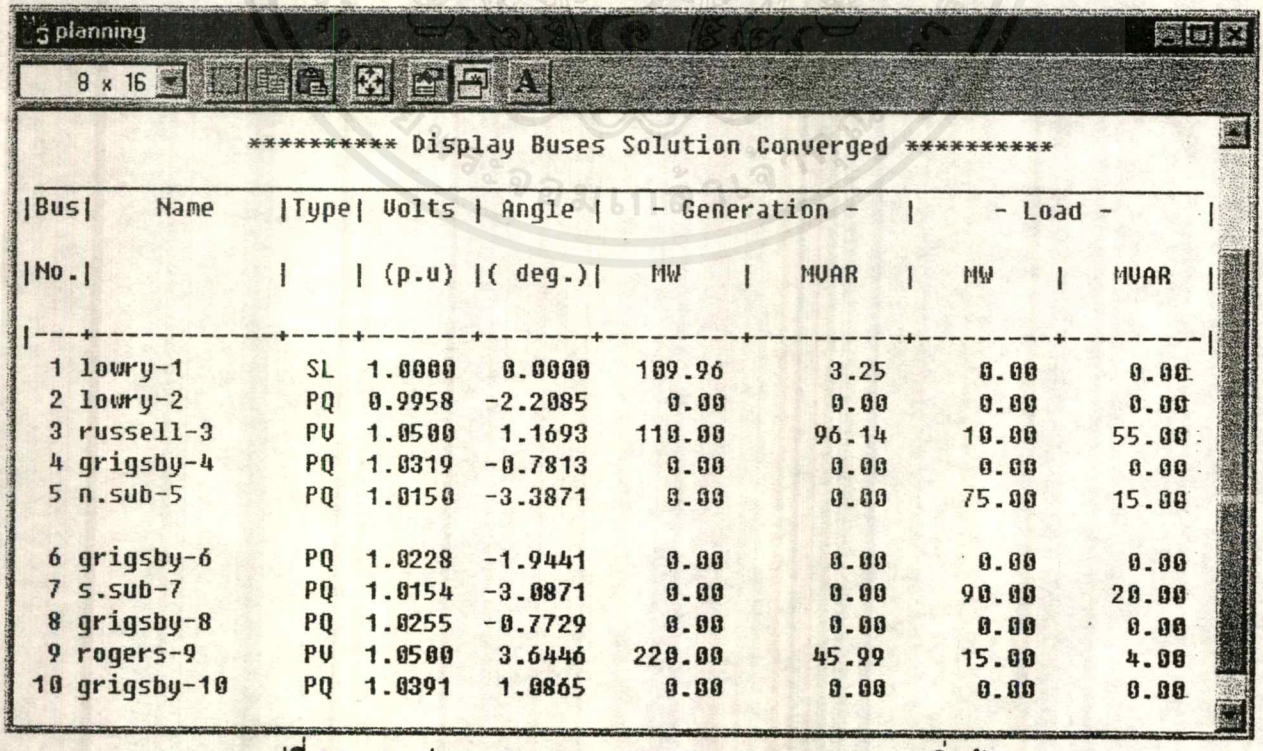
Tran No.	From Bus	Name	To Bus	Name	- Transformer Flows - MW	MVAR	MVA	TAP
1	3	FORTH	4	THIRD	-39.58	-15.57	42.54	0.989
	4	THIRD	3	FORTH	39.58	17.98	43.47	
2	5	SIXTH	6	FIFTH	-0.48	-7.05	7.06	0.975
	6	FIFTH	5	SIXTH	0.48	7.22	7.24	

รูปที่ 16 แสดงค่า transformer flows ของระบบ alpha

จากระบบที่นำมาวิเคราะห์โหลดโฟล ทั้ง 2 ระบบ สามารถทำให้เห็นถึงลักษณะการทำงานของโปรแกรมวิเคราะห์โหลดโฟลได้อย่างชัดเจนยิ่งขึ้น ต่อไปก็จะเป็นการนำวิธีการวิเคราะห์โหลดโฟลนี้ไปประยุกต์ใช้กับการทำงานที่เป็น ลักษณะของ Load Forecasting ดังผลการทดลอง ในลักษณะต่อไป

2. Load Forecasting.

ในส่วนนี้จะทำการทดลองกับระบบ 16 buses (gamma) ข้อมูลเริ่มต้นที่ใช้ดังแสดงในภาคผนวก ผลการวิเคราะห์ระบบเริ่มต้น พิจารณาที่ผลลัพธ์ final buses solution แสดงดังรูป



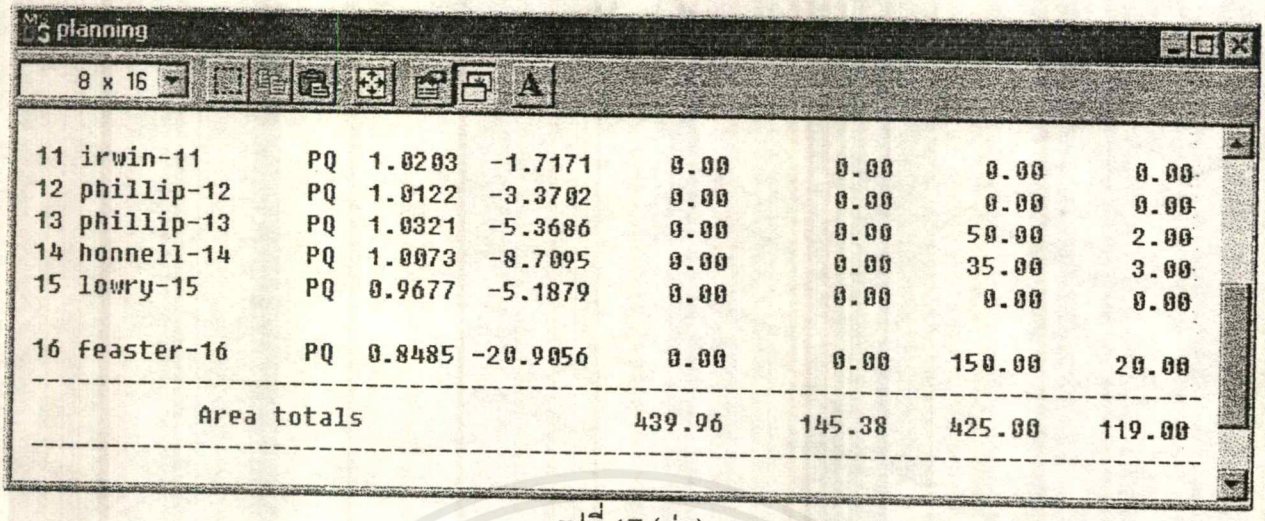
planning

8 x 16

********** Display Buses Solution Converged **********

Bus No.	Name	Type	Volts (p.u)	Angle (deg.)	- Generation - MW	MVAR	- Load - MW	MVAR
1	lowry-1	SL	1.0000	0.0000	109.96	3.25	0.00	0.00
2	lowry-2	PQ	0.9958	-2.2085	0.00	0.00	0.00	0.00
3	russell-3	PV	1.0500	1.1693	110.00	96.14	10.00	55.00
4	grigsby-4	PQ	1.0319	-0.7813	0.00	0.00	0.00	0.00
5	n.sub-5	PQ	1.0150	-3.3871	0.00	0.00	75.00	15.00
6	grigsby-6	PQ	1.0228	-1.9441	0.00	0.00	0.00	0.00
7	s.sub-7	PQ	1.0154	-3.0871	0.00	0.00	90.00	20.00
8	grigsby-8	PQ	1.0255	-0.7729	0.00	0.00	0.00	0.00
9	rogers-9	PV	1.0500	3.6446	220.00	45.99	15.00	4.00
10	grigsby-10	PQ	1.0391	1.0865	0.00	0.00	0.00	0.00

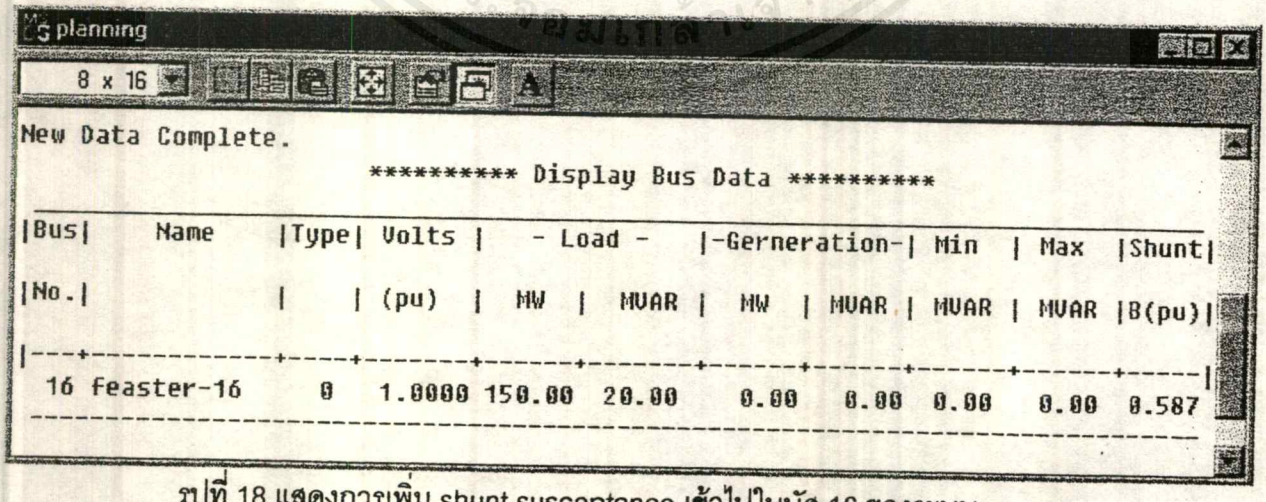
รูปที่ 17 แสดงค่า final buses solution ของระบบ gamma เริ่มต้น



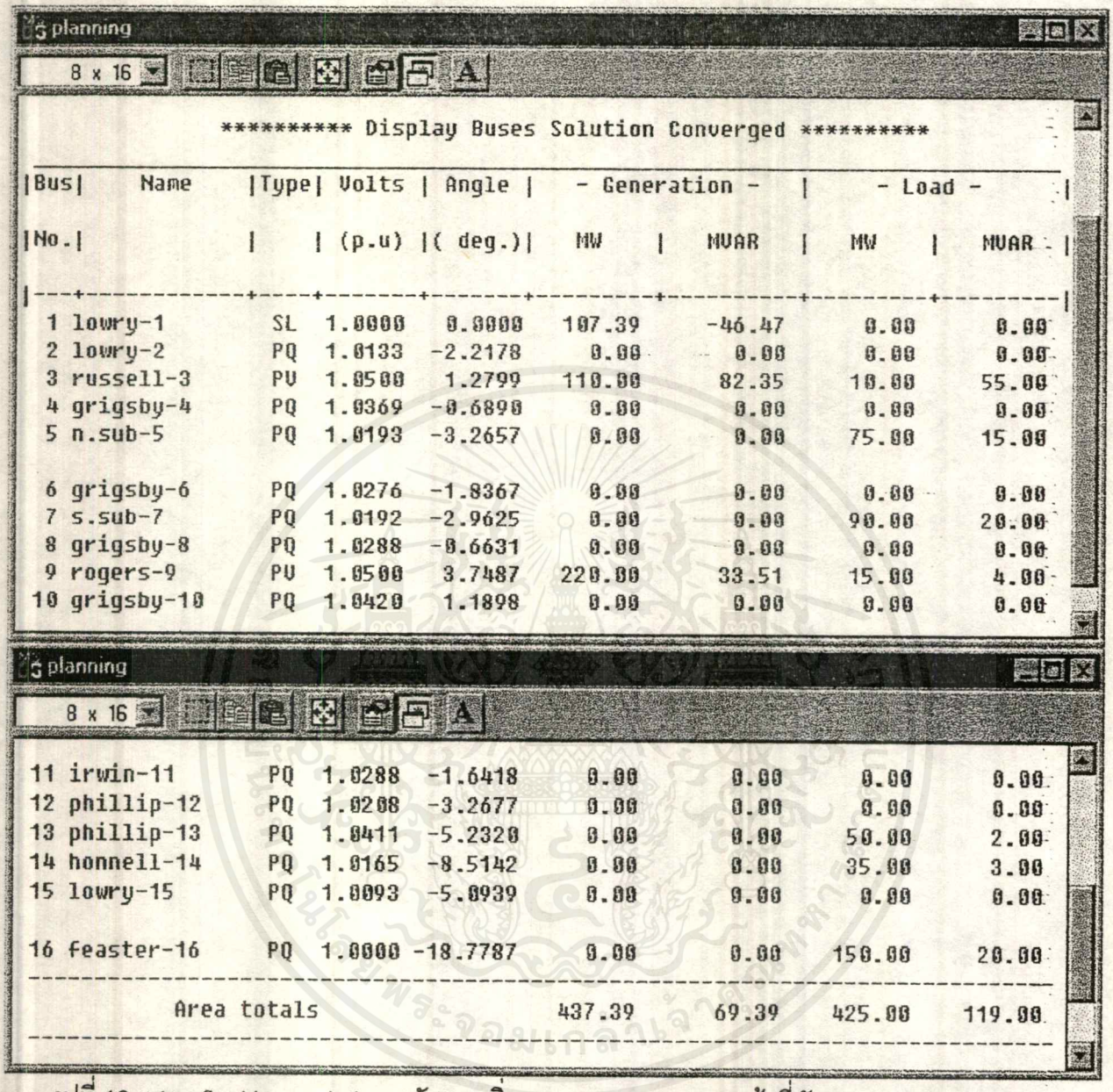
รูปที่ 17 (ต่อ)

จากผลของ final buses solution ของระบบ gamma เริ่มต้น เราจะพบว่า ค่าแรงดันที่บัส 16 มีขนาดแรงดันต่ำกว่าพิกัดมาก (1.05 - 0.95 per-unit) ดังนั้น เราจึงต้องมีการปรับปรุงคุณสมบัติของระบบให้เหมาะสม โดยเราจะใช้วิธีในการปรับปรุง 2 วิธี คือ

1. โดยการเพิ่มค่า shunt susceptance เข้าไปที่บัส 16 โดยมุ่งให้ แรงดันที่บัส 16 มีค่าเป็น 1.0000 per-unit เปรียบเสมือนเราจะทำการเพิ่ม ค่า Q_G ให้กับบัสที่ 16 เป็นค่า 58.70 MVAR หรือเทียบเป็นค่า shunt susceptance ขนาด 0.5870 per-unit แสดงผลต่าง ๆ ได้ดังรูป



รูปที่ 18 แสดงการเพิ่ม shunt susceptance เข้าไปในบัส 16 ของระบบ gamma



planning

8 x 16

********** Display Buses Solution Converged **********

Bus No.	Name	Type	Volts (p.u)	Angle (deg.)	- Generation - MW	MVAR	- Load - MW	MVAR
1	lowry-1	SL	1.0000	0.0000	107.39	-46.47	0.00	0.00
2	lowry-2	PQ	1.0133	-2.2178	0.00	0.00	0.00	0.00
3	russell-3	PV	1.0500	1.2799	110.00	82.35	10.00	55.00
4	grigsby-4	PQ	1.0369	-0.6890	0.00	0.00	0.00	0.00
5	n.sub-5	PQ	1.0193	-3.2657	0.00	0.00	75.00	15.00
6	grigsby-6	PQ	1.0276	-1.8367	0.00	0.00	0.00	0.00
7	s.sub-7	PQ	1.0192	-2.9625	0.00	0.00	90.00	20.00
8	grigsby-8	PQ	1.0288	-0.6631	0.00	0.00	0.00	0.00
9	rogers-9	PV	1.0500	3.7487	220.00	33.51	15.00	4.00
10	grigsby-10	PQ	1.0420	1.1898	0.00	0.00	0.00	0.00

planning

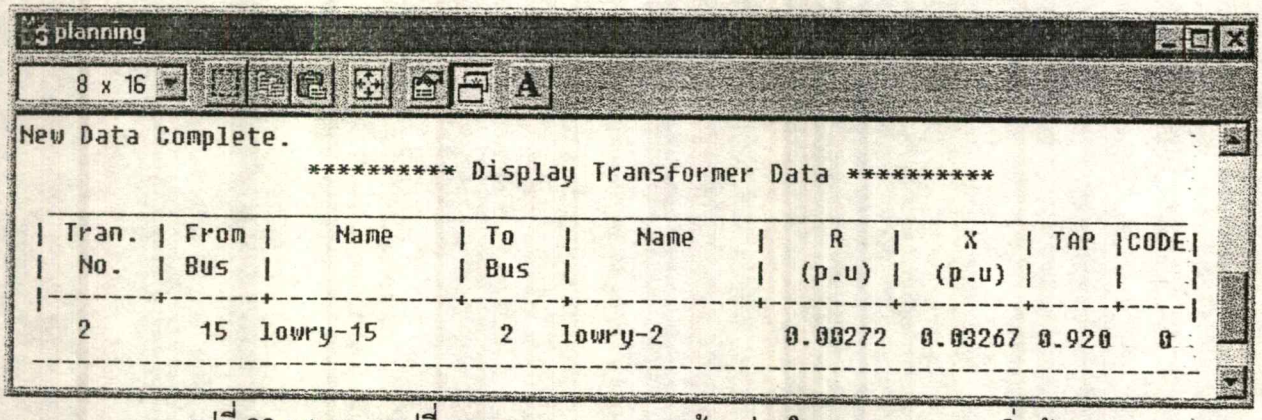
8 x 16

11	irwin-11	PQ	1.0288	-1.6418	0.00	0.00	0.00	0.00
12	phillip-12	PQ	1.0208	-3.2677	0.00	0.00	0.00	0.00
13	phillip-13	PQ	1.0411	-5.2320	0.00	0.00	50.00	2.00
14	honnell-14	PQ	1.0165	-8.5142	0.00	0.00	35.00	3.00
15	lowry-15	PQ	1.0093	-5.0939	0.00	0.00	0.00	0.00
16	feaster-16	PQ	1.0000	-18.7787	0.00	0.00	150.00	20.00
	Area totals				437.39	69.39	425.00	119.00

รูปที่ 19 แสดง final bus solution หลังจากเพิ่ม shunt susceptance เข้าที่บัส 16 ของระบบ gamma

จากผลของการเพิ่ม shunt susceptance เข้าไปในบัสที่ 16 พบว่า สามารถยกระดับแรงดันที่บัส 16 ให้มีค่าเพิ่มขึ้นเป็น 1.0000 ตามพิกัดได้ ทำให้ระบบเสถียรภาพมากขึ้น

2. โดยวิธีการเปลี่ยนค่า tap-setting ของ transformer ในกรณีนี้เราเลือกเปลี่ยน tap ทางด้าน Low ของ transformer ที่ต่ออยู่ระหว่างบัส 15 กับบัส 2 จาก 1.000 เป็น 0.920 เพื่อเป็นการยกระดับแรงดันของบัส 16 จากระบบ gamma เริ่มต้น ผลที่ได้แสดงดังรูป



planning

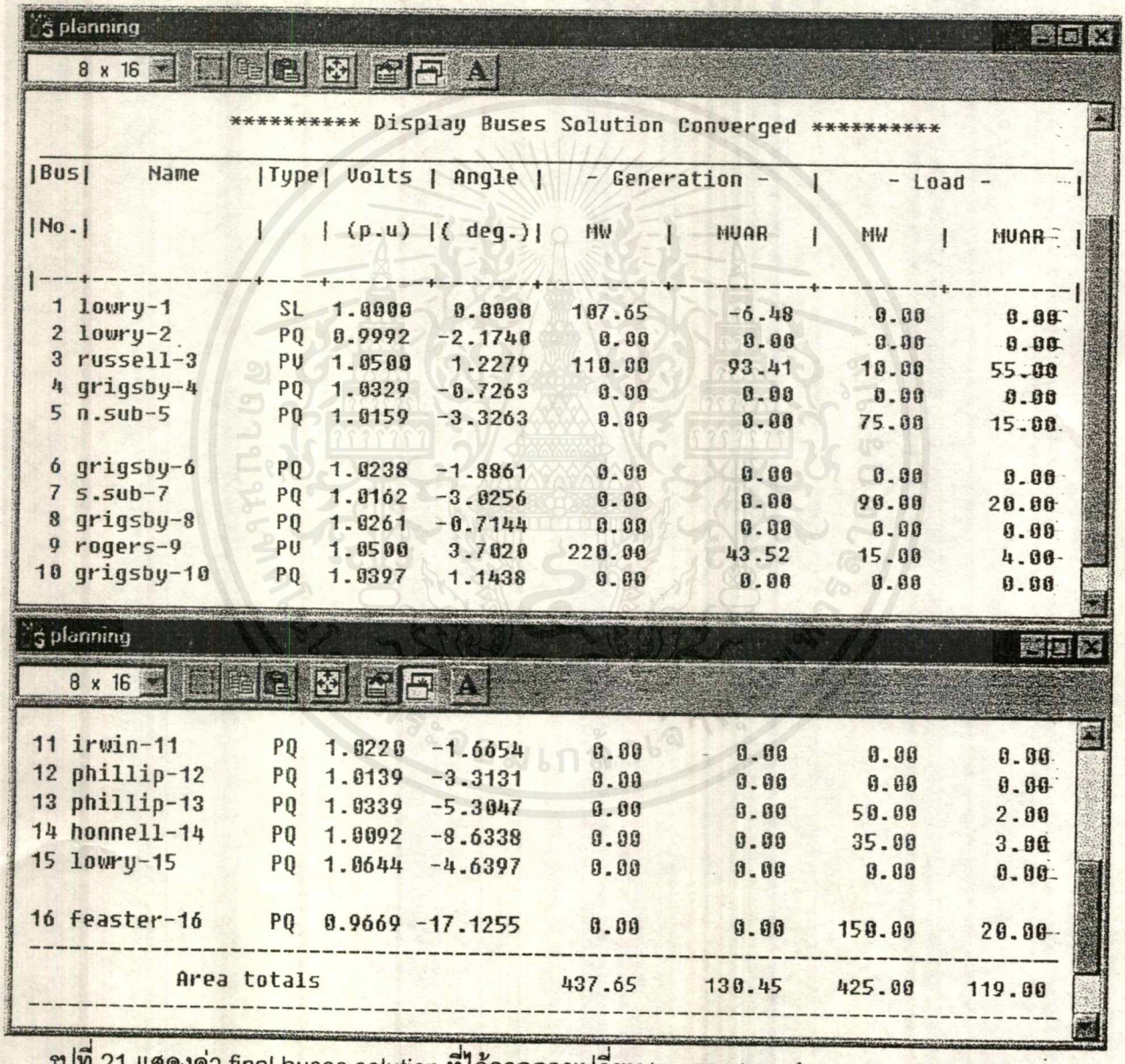
8 x 16

New Data Complete.

********** Display Transformer Data **********

Tran. No.	From Bus	Name	To Bus	Name	R (p.u)	X (p.u)	TAP	CODE
2	15	lowry-15	2	lowry-2	0.00272	0.03267	0.920	0

รูปที่ 20 แสดงการเปลี่ยน tap-setting ของหม้อแปลงในระบบ gamma เริ่มต้น



planning

8 x 16

********** Display Buses Solution Converged **********

Bus No.	Name	Type	Volts (p.u)	Angle (deg.)	- Generation - MW	MVAR	- Load - MW	MVAR
1	lowry-1	SL	1.0000	0.0000	107.65	-6.48	0.00	0.00
2	lowry-2	PQ	0.9992	-2.1740	0.00	0.00	0.00	0.00
3	russell-3	PV	1.0500	1.2279	110.00	93.41	10.00	55.00
4	grigsby-4	PQ	1.0329	-0.7263	0.00	0.00	0.00	0.00
5	n.sub-5	PQ	1.0159	-3.3263	0.00	0.00	75.00	15.00
6	grigsby-6	PQ	1.0238	-1.8861	0.00	0.00	0.00	0.00
7	s.sub-7	PQ	1.0162	-3.0256	0.00	0.00	90.00	20.00
8	grigsby-8	PQ	1.0261	-0.7144	0.00	0.00	0.00	0.00
9	rogers-9	PV	1.0500	3.7020	220.00	43.52	15.00	4.00
10	grigsby-10	PQ	1.0397	1.1438	0.00	0.00	0.00	0.00

planning

8 x 16

11	irwin-11	PQ	1.0220	-1.6654	0.00	0.00	0.00
12	phillip-12	PQ	1.0139	-3.3131	0.00	0.00	0.00
13	phillip-13	PQ	1.0339	-5.3047	0.00	0.00	50.00	2.00
14	honnell-14	PQ	1.0092	-8.6338	0.00	0.00	35.00	3.00
15	lowry-15	PQ	1.0644	-4.6397	0.00	0.00	0.00	0.00
16	feaster-16	PQ	0.9669	-17.1255	0.00	0.00	150.00	20.00
	Area totals				437.65	130.45	425.00	119.00

รูปที่ 21 แสดงค่า final buses solution ที่ได้จากการเปลี่ยน tap ของ transformer ของระบบ gamma

ผลจากการเปลี่ยน tap ทางด้าน Low ของ transformer ที่ต่ออยู่ระหว่าง บัส 15 และบัส 2 จาก 1.000 เป็น 0.920 ทำให้สามารถยกระดับแรงดันที่บัส 16 ขึ้นได้ เป็น 0.9669 (>0.95) จะส่งผลให้บัสที่ 15 อยู่ในสภาวะ high-voltage 1.0644 (>1.05) แต่เนื่องจาก เป็นการ no-load ที่บัส 15 ทำให้สภาวะนี้สามารถยอมรับได้

สรุปผลและวิจารณ์

จากผลการทดสอบที่ได้ทำการแบ่งออกเป็น 2 ลักษณะ คือ

1. ส่วน Load Flow พบว่าผลที่ได้จากการคำนวณ และผลจาก case study มีค่าตรงกันทั้งในระบบ beta และ alpha เป็นการยืนยันให้เห็นว่า โปรแกรมวิเคราะห์โหลดโฟลที่ได้ออกแบบ และเขียนขึ้นนี้สามารถที่จะนำไปประยุกต์ใช้กับระบบไฟฟ้าจริง ๆ ได้ โดยวิธีที่ใช้นี้ ซึ่งเป็น วิธีของนิวตัน-ราฟสัน จะมีการคำนวณในแต่ละรอบที่ยุ่งยาก แต่จำนวนรอบของการคำนวณที่ใช้น้อย และจำนวนรอบของการคำนวณไม่เปลี่ยนแปลงตามจำนวนบัสของระบบมากนัก

2.ส่วน Load Forecasting จากการทดลองกับระบบ gamma เป็นการแสดงให้เห็นถึงประโยชน์ ของการปรับปรุงระบบในลักษณะของการ Forecasting ได้เป็นอย่างดี โดยในการทดลองได้ใช้การเพิ่ม shunt susceptance เข้าไปที่บัสใด ๆ และการใช้การเปลี่ยน tap ของ transformer เป็นแนวทางให้เราสามารถที่จะทำการทดสอบระบบของเราเพื่อการวางแผนปฏิบัติงานจริงในอนาคต จะช่วยให้การตัดสินใจทำได้ง่ายขึ้น และลดความเสี่ยงในการลงทุน ลงไปได้อีกด้วย

ส่วนการปรับปรุงในส่วนโปรแกรมนั้นควรมีการปรับปรุง ดังต่อไปนี้

1. ปรับปรุงการคำนวณในส่วน Load Flow โดยใช้เทคนิคทางคอมพิวเตอร์ให้มากขึ้น เพื่อการคำนวณที่เร็ว และประหยัดหน่วยความจำมากขึ้น

2. ปรับปรุงในส่วน interactive ระหว่าง user และ computer ให้ดียิ่งขึ้น เพื่อให้สะดวก สวยงามเหมาะแก่การใช้งานมากยิ่งขึ้น

3. โปรแกรมที่ได้สร้างขึ้นมานี้ สามารถใช้เป็นพื้นฐานในการคำนวณระบบไฟฟ้าในด้านอื่น ๆ เช่น Economic Load Dispatch ,Fault Calculation หรือ Power System Stability จะเป็นการดีอย่างยิ่งถ้าจะได้มีการพัฒนาโปรแกรมนี้ ต่อ ๆ ไป

ภาคผนวก

```
//=========================================================//
//  KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG      //
//  PROJECT : POWER SYSTEM PLANNING                        //
//            LOADFLOW AND LOAD FORECASTING                //
//  ADVISOR : MR. SOMPOTSH     PRAPAI                      //
//  ADVISEE : MR. YOCHAI       SASIWAN                     //
//            MR. ROONGROT     WANAPHURKSASILP             //
//=========================================================//

//FILE :: PLANNING.CPP

#include <iostream.h>
#include <fstream.h>
#include <iomanip.h>
#include <conio.h>
#include <string.h>
#include <complex.h>
#include "planning.h"

//Main Function of Program Load Flow and Load Forecasting...

main()
   {
    Base base_dat;
    Bus bus_dat[MAX_BUS];
    Line line_dat[MAX_LINE];
    Transformer tran_dat[MAX_TRAN];
    int num_bus,num_line,num_tran;
    char Filename[13],Filebase[13],Filebus[13],Fileline[13],Filetran[13];
    char operation,option,option1;

    do{
       clrscr();
       cout << "\n                Power System Planning";
       cout << "\n           Load Flow and Load Forecasting";
       cout << "\n\nMain Menu --->\n";
       cout << "\n     1.Create New System";
       cout << "\n     2.Display Data of Existing System";
       cout << "\n     3.Modify Existing System for Forecasting";
       cout << "\n     4.Analyse Loadflow of Existing System";
       cout << "\n     5.Exit";
       do{
       cout << "\n\nChoose your operation ---> ";
       cin  >> operation;
          }
       while ((operation!='1')&(operation!='2')&(operation!='3')&(operation!='4')&(o
peration!='5'));

    switch (operation) {
    case '1' : //Create New System
               {
    clrscr();
    do{
    cout << "Enter System Filename \n<Don't exceed 8 characters> : ";
    cin >> Filename;
       }
    while (strlen(Filename) > 8);
    strcpy(Filebase,Filename);
    strcpy(Filebus,Filename);
    strcpy(Fileline,Filename);
    strcpy(Filetran,Filename);
    strcat(Filebase,".bas");
```

```
    strcat(Filebus,".bus");
    strcat(Fileline,".lin");
    strcat(Filetran,".trn");
    cout << "\nFilename : " << Filebase << " , " << Filebus << " , "
         << Fileline << " and " << Filetran;
    cout << "\nPlease input completely data.\n";

    base_dat.in_data();
    base_dat.write_to_disk(Filebase);

    base_dat.read_from_disk(Filebase);
    num_bus = base_dat.num_bus;
    num_line = base_dat.num_line;
    num_tran = base_dat.num_tran;

    cout << "\nInput Data for Buses Data.\n";
    int count_bus = 0;
    for (int i=0;i<num_bus;i++)
       {
          bus_dat[i].in_bus_data((i+1));
          if ((i+1)%5 == 0)
             {
             count_bus += 1;
             for (int j=(i-4);j<i+1;j++)
                {
                   bus_dat[j].write_to_disk(Filebus,j,num_bus);
                }
             }
          else if ((i+1) == num_bus)
             for (int j=(count_bus*5);j<num_bus;j++)
                {
                   bus_dat[j].write_to_disk(Filebus,j,num_bus);
                }
       }

    if (num_line != 0) cout << "\nInput Data for Lines Data.\n";
    int count_line = 0;
    for (int i=0;i<num_line;i++)
       {
          line_dat[i].in_line_data(Filebus,num_bus,(i+1));
          if ((i+1)%5 == 0)
             {
             count_line += 1;
             for (int j=(i-4);j<i+1;j++)
                {
                   line_dat[j].write_to_disk(Fileline,j,num_line);
                }
             }
          else if ((i+1) == num_line)
             for (int j=(count_line*5);j<num_line;j++)
                {
                   line_dat[j].write_to_disk(Fileline,j,num_line);
                }
       }
    if (num_line == 0)
       {
          Line line_buff;
          line_buff.write_to_disk(Fileline,0,1);
       }

    if (num_tran != 0) cout << "\nInput Data for Transformers Data.\n";
    int count_tran = 0;
    for (int i=0;i<num_tran;i++)
       {
          tran_dat[i].in_tran_data(Filebus,num_bus,(i+1));
          if ((i+1)%5 == 0)
```

```
                {
                count_tran += 1;
                for (int j=(i-4);j<i+1;j++)
                   {
                      tran_dat[j].write_to_disk(Filetran,j,num_tran);
                   }
                }
             else if ((i+1) == num_tran)
                for (int j=(count_tran*5);j<num_tran;j++)
                   {
                      tran_dat[j].write_to_disk(Filetran,j,num_tran);
                   }
          }
       if (num_tran == 0)
          {
             Transformer tran_buff;
             tran_buff.write_to_disk(Filetran,0,1);
          }

       cout << "\nPress Return to Read and Display Data\n\n";
       getche();

       cout << "Display base data.\n";
       base_dat.read_from_disk(Filebase);
       base_dat.display_data(Filebase);

       cout << "\n\nDisplay bus data from file : " << Filebus << "\n\n";
       display_bus_list();
       for (int i=0;i<num_bus;i++)
          {
             bus_dat[i].read_from_disk(Filebus,i);
             bus_dat[i].display_data((i+1),num_bus);
          }
       display_end_line();
       cout << "\n\nPress Return to be continue.";
       getche();

       cout << "\n\nDisplay line data from file : " << Fileline << "\n\n";
       display_line_list();
       for (int i=0;i<num_line;i++)
          {
             line_dat[i].read_from_disk(Fileline,i);
             line_dat[i].display_data((i+1),num_line);
          }
       display_end_line();
       cout << "\n\nPress Return to be continue.";
       getche();

       cout << "\n\nDisplay transformer data from file : " << Filetran << "\n\n";
       display_transformer_list();
       for (int i=0;i<num_tran;i++)
          {
             tran_dat[i].read_from_disk(Filetran,i);
             tran_dat[i].display_data((i+1),num_tran);
          }
       display_end_line();
       cout << "\n\nPress Return to be continue.";
       getche();

       break;
             }

       case '2' : //Display Data of Existing System
             {
       clrscr();
```

```
      do{
      cout << "Enter System Filename \n<Don't exceed 8 characters> : ";
      cin >> Filename;
         }
      while (strlen(Filename) > 8);
      strcpy(Filebase,Filename);
      strcpy(Filebus,Filename);
      strcpy(Fileline,Filename);
      strcpy(Filetran,Filename);
      strcat(Filebase,".bas");
      strcat(Filebus,".bus");
      strcat(Fileline,".lin");
      strcat(Filetran,".trn");
      cout << "\nFilename : " << Filebase << " , " << Filebus << " , ";
      cout << Fileline << " and " << Filetran;

      ifstream file1,file2,file3,file4;

      file1.open(Filebase,ios::in);
      file2.open(Filebus,ios::in);
      file3.open(Fileline,ios::in);
      file4.open(Filetran,ios::in);
      if ((!file1)||(!file2)||(!file3)||(!file4))
         {
         cout << "\nCannot open file " << Filebase << " , " << Filebus << " , ";
         cout << Fileline << " and " << Filetran << "\n";
         cout << "\n\nPress Return to be continue.";
         getche();
         }
      else {

      cout << "\n\nPress Return to Read and Display Data\n\n";
      getche();

      cout << "Display base data.\n";
      base_dat.read_from_disk(Filebase);
      num_bus = base_dat.num_bus;
      num_line = base_dat.num_line;
      num_tran = base_dat.num_tran;
      base_dat.display_data(Filebase);

      cout << "\n\nDisplay bus data from file :" << Filebus << "\n\n";
      display_bus_list();
      for (int i=0;i<num_bus;i++)
         {
            bus_dat[i].read_from_disk(Filebus,i);
            bus_dat[i].display_data((i+1),num_bus);
         }
      display_end_line();
      cout << "\n\nPress Return to be continue.";
      getche();

      cout << "\n\nDisplay line data from file : " << Fileline << "\n\n";
      display_line_list();
      for (int i=0;i<num_line;i++)
         {
            line_dat[i].read_from_disk(Fileline,i);
            line_dat[i].display_data((i+1),num_line);
         }
      display_end_line();
      cout << "\n\nPress Return to be continue.";
      getche();

      cout << "\n\nDisplay transformer data from file : " << Filetran << "\n\n";
      display_transformer_list();
      for (int i=0;i<num_tran;i++)
```

```
    {
       tran_dat[i].read_from_disk(Filetran,i);
       tran_dat[i].display_data((i+1),num_tran);
    }
display_end_line();
cout << "\n\nPress Return to be continue.";
getche();
     }
break;
          }
case '3' : //Modify Existing System for Forecasting
    {
    do{
    clrscr();
    cout << "\n              Modify Existing System for Forecasting";
    cout << "\n            Change ,Add and Remove System data";
    cout << "\n\nMenu Options--->\n";
    cout << "\n     1.Change System Data";
    cout << "\n     2.Add System Data";
    cout << "\n     3.Remove System Data";
    cout << "\n     4.Back <---";
    do{
    cout << "\n\nChoose your option ---> ";
    cin  >> option;
       }
    while ((option!='1')&(option!='2')&(option!='3')&(option!='4'));

    if (option!='4')
    {
    do{
    cout << "Enter System Filename \n<Don't exceed 8 characters> : ";
    cin >> Filename;
      }
    while (strlen(Filename) > 8);
    strcpy(Filebase,Filename);
    strcpy(Filebus,Filename);
    strcpy(Fileline,Filename);
    strcpy(Filetran,Filename);
    strcat(Filebase,".bas");
    strcat(Filebus,".bus");
    strcat(Fileline,".lin");
    strcat(Filetran,".trn");
    cout << "\nFilename : " << Filebase << " , " << Filebus << " , ";
    cout << Fileline << " and " << Filetran;

    ifstream file1,file2,file3,file4;

    file1.open(Filebase,ios::in);
    file2.open(Filebus,ios::in);
    file3.open(Fileline,ios::in);
    file4.open(Filetran,ios::in);
    if ((!file1)|(!file2)|(!file3)|(!file4))
       {
       cout << "\nCannot open file " << Filebase << " , " << Filebus << " , ";
       cout << Fileline << " and " << Filetran << "\n";
       cout << "\n\nPress Return to be continue.";
       getche();
       }
    else{

    cout << "\n\nPress Return to Read and Display Data\n\n";
    getche();

    cout << "Display base data.\n";
    base_dat.read_from_disk(Filebase);
    num_bus = base_dat.num_bus;
```

```
      num_line = base_dat.num_line;
      num_tran = base_dat.num_tran;
      base_dat.display_data(Filebase);

      cout << "\n\nDisplay bus data from file : " << Filebus << "\n\n";
      display_bus_list();
      for (int i=0;i<num_bus;i++)
         {
            bus_dat[i].read_from_disk(Filebus,i);
            bus_dat[i].display_data((i+1),num_bus);
         }
      display_end_line();
      cout << "\n\nPress Return to be continue.";
      getche();

      cout << "\n\nDisplay line data from file : " << Fileline << "\n\n";
      display_line_list();
      for (int i=0;i<num_line;i++)
         {
            line_dat[i].read_from_disk(Fileline,i);
            line_dat[i].display_data((i+1),num_line);
         }
      display_end_line();
      cout << "\n\nPress Return to be continue.";
      getche();

      cout << "\n\nDisplay transformer data from file : " << Filetran << "\n\n";
      display_transformer_list();
      for (int i=0;i<num_tran;i++)
         {
            tran_dat[i].read_from_disk(Filetran,i);
            tran_dat[i].display_data((i+1),num_tran);
         }
      display_end_line();
      cout << "\n\nPress Return to be continue.";
      getche();

   switch (option) {
      case '1' : //Change System Data
      {
      do{
      clrscr();
      cout << "\n                Modify Existing System for Forecasting";
      cout << "\n                     Change System Data";
      cout << "\n\nChange Options--->\n";
      cout << "\n    1.Change Base Data";
      cout << "\n    2.Change Bus Data";
      cout << "\n    3.Change Line Data";
      cout << "\n    4.Change Transformer Data";
      cout << "\n    5.Back <---";
      do{
      cout << "\n\nChoose your option ---> ";
      cin  >> option1;
         }
      while ((option1!='1')&(option1!='2')&(option1!='3')&(option1!='4')&(option1!=
'5'));
   switch (option1) {
      case '1' : //Change Base Data
         {
            int slack;
            float max_ite,power_tole;

            cout << "\nChange Base Data.\n";
            cout << "\nNumber of Buses : " << base_dat.num_bus;
            cout << "\nNumber of Transmission Lines : " << base_dat.num_line;
            cout << "\nNumber of Transformers : " << base_dat.num_tran;
```

```
            cout << "\n\nYou can change SLACK BUS,MAXIMUM ITERATION,and POWER TOLER
ANCE\n";
            do{
            cout << "Slack Bus 1-" << base_dat.num_bus << " <" << base_dat.slack <<
 "> : ";
            cin  >> slack;
              }
            while ((slack < 1)|(slack > num_bus));
            base_dat.slack = slack;
            do{
            cout << "Maximum Iteration 1-~ <" << base_dat.max_ite << "> : ";
            cin  >> max_ite;
               }
            while (max_ite < 1);
            base_dat.max_ite = max_ite;
            do{
            cout << "Power Tolerance (>=0.00000) <" << setprecision(5) << base_dat.
power_tole << "> : ";
            cin  >> power_tole;
               }
            while (power_tole < 0.00000);
            base_dat.power_tole = power_tole;
            base_dat.write_to_disk(Filebase);
            cout << "\nChange Base Data Complete.";
            cout << "\n\nPress Return to be continue.";
            getche();
         }
            break;
      case '2' : //Change Bus Data
         {
            if (num_bus > 0)
            {
            int bus_select;

            do{
            cout << "\nSelect Bus to Change <1-" << num_bus << "> : ";
            cin  >> bus_select;
              }
            while ((bus_select < 1)|(bus_select > num_bus));
            display_bus_list();
            bus_dat[bus_select-1].display_data(bus_select,num_bus);
            display_end_line();
            cout << "\nEnter New Data.";
            bus_dat[bus_select-1].change_data(bus_select,bus_dat[bus_select-1].bus
name);
            int count_bus = 0;
            for (int i=0;i<num_bus;i++)
               {
                  if ((i+1)%5 == 0)
                     {
                     count_bus += 1;
                     for (int j=(i-4);j<i+1;j++)
                        {
                           bus_dat[j].write_to_disk(Filebus,j,num_bus);
                        }
                     }
                  else if ((i+1) == num_bus)
                     for (int j=(count_bus*5);j<num_bus;j++)
                        {
                           bus_dat[j].write_to_disk(Filebus,j,num_bus);
                        }
               }
            cout << "\nNew Data Complete.";
            display_bus_list();
            bus_dat[bus_select-1].read_from_disk(Filebus,bus_select-1);
            bus_dat[bus_select-1].display_data(bus_select,num_bus);
```

```
            display_end_line();
            cout << "\n\nPress Return to be continue.";
            getche();
            }
            else
               {
               cout << "\nNo Buses Data for Change!";
               cout << "\n\nPress Return to be continue.";
               getche();
               }
         }
            break;
      case '3' : //Change Line Data
         {
            if (num_line > 0)
            {
            int line_select;

            do{
            cout << "\nSelect Line to Change <1-" << num_line << "> : ";
            cin  >> line_select;
              }
            while ((line_select < 1)|(line_select > num_line));
            display_line_list();
            line_dat[line_select-1].display_data(line_select,num_line);
            display_end_line();
            cout << "\nEnter New Data.";
            line_dat[line_select-1].change_data(line_select,line_dat[line_select-1]
.from_bus,line_dat[line_select-1].name1,line_dat[line_select-1].to_bus,line_dat[line_s
elect-1].name2);
            int count_line = 0;
            for (int i=0;i<num_line;i++)
               {
                  if ((i+1)%5 == 0)
                     {
                     count_line += 1;
                     for (int j=(i-4);j<i+1;j++)
                        {
                           line_dat[j].write_to_disk(Fileline,j,num_line);
                        }
                     }
                  else if ((i+1) == num_line)
                     for (int j=(count_line*5);j<num_line;j++)
                        {
                           line_dat[j].write_to_disk(Fileline,j,num_line);
                        }
               }
            cout << "\nNew Data Complete.";
            display_line_list();
            line_dat[line_select-1].read_from_disk(Fileline,line_select-1);
            line_dat[line_select-1].display_data(line_select,num_line);
            display_end_line();
            cout << "\n\nPress Return to be continue.";
            getche();
            }
            else
               {
               cout << "\nNo Lines Data for Change!";
               cout << "\n\nPress Return to be continue.";
               getche();
               }
         }
            break;
      case '4' : //Change Transformer Data
         {
            if (num_tran > 0)
```

```
            {
            int tran_select;

            do{
            cout << "\nSelect Tranformer to Change <1-" << num_tran << "> : ";
            cin  >> tran_select;
              }
            while ((tran_select < 1)|(tran_select > num_tran));
            display_transformer_list();
            tran_dat[tran_select-1].display_data(tran_select,num_tran);
            display_end_line();
            cout << "\nEnter New Data.";
            tran_dat[tran_select-1].change_data(tran_select,tran_dat[tran_select-1]
.from_bus,tran_dat[tran_select-1].name1,tran_dat[tran_select-1].to_bus,tran_dat[tran_s
elect-1].name2);
            int count_tran = 0;
            for (int i=0;i<num_tran;i++)
               {
                  if ((i+1)%5 == 0)
                     {
                     count_tran += 1;
                     for (int j=(i-4);j<i+1;j++)
                        {
                           tran_dat[j].write_to_disk(Filetran,j,num_tran);
                        }
                     }
                  else if ((i+1) == num_tran)
                     for (int j=(count_tran*5);j<num_tran;j++)
                        {
                           tran_dat[j].write_to_disk(Filetran,j,num_tran);
                        }
               }
            cout << "\nNew Data Complete.";
            display_transformer_list();
            tran_dat[tran_select-1].read_from_disk(Filetran,tran_select-1);
            tran_dat[tran_select-1].display_data(tran_select,num_tran);
            display_end_line();
            cout << "\n\nPress Return to be continue.";
            getche();
            }
            else
               {
               cout << "\nNo Transformers Data for Change!";
               cout << "\n\nPress Return to be continue.";
               getche();
               }
         }
               break;
      case '5' : //Back
               break;
                   }
         }
      while (option1!='5');
      break;
      }
      case '2' : //Add System Data
      {
      do{
      clrscr();
      cout << "\n              Modify Existing System for Forecasting";
      cout << "\n                   Add System Data";
      cout << "\n\nAdd Options--->\n";
      cout << "\n     1.Add Bus Data";
      cout << "\n     2.Add Line Data";
      cout << "\n     3.Add Transformer Data";
      cout << "\n     4.Back <---";
```

```
    do{
    cout << "\n\nChoose your option ---> ";
    cin  >> option1;
       }
    while ((option1!='1')&(option1!='2')&(option1!='3')&(option1!='4'));
  switch (option1) {
    case '1' : //Add Bus Data
    {
          int max_add,add_bus,num_zero;

          max_add = MAX_BUS-num_bus;
          num_zero = num_bus;
          do{
          cout << "\nNumber of Addition Buses <Don't Exceed " << max_add << "> :
";
          cin  >> add_bus;
             }
          while ((add_bus < 1)|(add_bus > max_add));
          base_dat.num_bus += add_bus;
          base_dat.write_to_disk(Filebase);
          cout << "\nEnter Addition Buses Data.";
          for (int i=num_bus;i<(num_bus+add_bus);i++)
             {
                bus_dat[i].in_bus_data(i+1);
             }
          int count_bus = 0;
          for (int i=0;i<(num_bus+add_bus);i++)
             {
                if ((i+1)%5 == 0)
                   {
                   count_bus += 1;
                   for (int j=(i-4);j<i+1;j++)
                      {
                         bus_dat[j].write_to_disk(Filebus,j,(num_bus+add_bus));
                      }
                   }
                else if ((i+1) == (num_bus+add_bus))
                   for (int j=(count_bus*5);j<(num_bus+add_bus);j++)
                      {
                         bus_dat[j].write_to_disk(Filebus,j,(num_bus+add_bus));
                      }
             }
          cout << "\nAddition Buses Data Complete.";
          base_dat.read_from_disk(Filebase);
          cout << "\nTotal Number of Buses : " << base_dat.num_bus;
          display_bus_list();
          for (int i=num_bus;i<(num_bus+add_bus);i++)
             {
                bus_dat[i].read_from_disk(Filebus,i);
                bus_dat[i].display_data((i+1),num_bus);
             }
          display_end_line();
          num_bus += add_bus;
          if ((num_zero == 0)&(num_bus > 1))
             {
             int slack;

             cout << "\n\nSlack Bus must be changed by user!";
             getche();
             cout << "\n\nDisplay bus data from file : " << Filebus << "\n\n";
             display_bus_list();
             for (int i=0;i<num_bus;i++)
                {
                   bus_dat[i].read_from_disk(Filebus,i);
                   bus_dat[i].display_data((i+1),num_bus);
                }
```

```
                display_end_line();
                do{
                   cout << "\nYou must choose new Slack Bus! <1-" << num_bus << "> :
";
                   cin  >> slack;
                   }
                while ((slack < 1)|(slack > num_bus));
                base_dat.slack = slack;
                base_dat.write_to_disk(Filebase);
                }
             else if (num_bus == 1)
                {
                base_dat.slack = 1;
                base_dat.write_to_disk(Filebase);
                }
             cout << "\n\nPress Return to be continue.";
             getche();
          }
             break;
       case '2' : //Add Line Data
          {
          if (num_bus > 1)
          {
             int max_add,add_line;

             max_add = MAX_LINE-num_line;
             do{
             cout << "\nNumber of Addition Lines <Don't Exceed " << max_add << "> :
";
             cin  >> add_line;
                }
             while ((add_line < 1)|(add_line > max_add));
             base_dat.num_line += add_line;
             base_dat.write_to_disk(Filebase);
             cout << "\nEnter Addition Lines Data.";
             for (int i=num_line;i<(num_line+add_line);i++)
                {
                   line_dat[i].in_line_data(Filebus,num_bus,i+1);
                }
             int count_line = 0;
             for (int i=0;i<(num_line+add_line);i++)
                {
                   if ((i+1)%5 == 0)
                      {
                      count_line += 1;
                      for (int j=(i-4);j<i+1;j++)
                         {
                            line_dat[j].write_to_disk(Fileline,j,(num_line+add_line)
);
                         }
                      }
                   else if ((i+1) == (num_line+add_line))
                      for (int j=(count_line*5);j<(num_line+add_line);j++)
                         {
                            line_dat[j].write_to_disk(Fileline,j,(num_line+add_line)
);
                         }
                }
             cout << "\nAddition Lines Data Complete.";
             base_dat.read_from_disk(Filebase);
             cout << "\nTotal Number of Lines : " << base_dat.num_line;
             display_line_list();
             for (int i=num_line;i<(num_line+add_line);i++)
                {
                   line_dat[i].read_from_disk(Fileline,i);
                   line_dat[i].display_data((i+1),num_line);
```

```
            }
            display_end_line();
            num_line += add_line;
         }
         else cout << "\nCan't Add any Lines into The System!";
            cout << "\n\nPress Return to be continue.";
            getche();
         }
            break;
      case '3' : //Add Transformer Data
         {
         if (num_bus > 1)
         {
            int max_add,add_tran;

            max_add = MAX_TRAN-num_tran;
            do{
            cout << "\nNumber of Addition Transformers <Don't Exceed " << max_add <
< ">: ";
            cin  >> add_tran;
               }
            while ((add_tran < 1)|(add_tran > max_add));
            base_dat.num_tran += add_tran;
            base_dat.write_to_disk(Filebase);
            cout << "\nEnter Addition Transformers Data.";
            for (int i=num_tran;i<(num_tran+add_tran);i++)
               {
                  tran_dat[i].in_tran_data(Filebus,num_bus,i+1);
               }
            int count_tran = 0;
            for (int i=0;i<(num_tran+add_tran);i++)
               {
                  if ((i+1)%5 == 0)
                  {
                  count_tran += 1;
                  for (int j=(i-4);j<i+1;j++)
                     {
                        tran_dat[j].write_to_disk(Filetran,j,(num_tran+add_tran)
);
                     }
                  }
                  else if ((i+1) == (num_tran+add_tran))
                     for (int j=(count_tran*5);j<(num_tran+add_tran);j++)
                        {
                           tran_dat[j].write_to_disk(Filetran,j,(num_tran+add_tran)
);
                        }
               }
            cout << "\nAddition Transformers Data Complete.";
            base_dat.read_from_disk(Filebase);
            cout << "\nTotal Number of Transformers : " << base_dat.num_tran;
            display_transformer_list();
            for (int i=num_tran;i<(num_tran+add_tran);i++)
               {
                  tran_dat[i].read_from_disk(Filetran,i);
                  tran_dat[i].display_data((i+1),num_tran);
               }
            display_end_line();
            num_tran += add_tran;
         }
         else cout << "\nCan't Add any Transformers into The System!";
            cout << "\n\nPress Return to be continue.";
            getche();
         }
            break;
      case '4' : //Back
```

```
                break;
                    }
            }
        while (option1!='4');
        break;
        }
        case '3' : //Remove System Data
        {
        do{
        clrscr();
        cout << "\n              Modify Existing System for Forecasting";
        cout << "\n                   Remove System Data";
        cout << "\n\nRemove Options--->\n";
        cout << "\n     1.Remove Bus Data";
        cout << "\n     2.Remove Line Data";
        cout << "\n     3.Remove Transformer Data";
        cout << "\n     4.Back <---";
        do{
        cout << "\n\nChoose your option ---> ";
        cin  >> option1;
           }
        while ((option1!='1')&(option1!='2')&(option1!='3')&(option1!='4'));
     switch (option1) {
        case '1' : //Remove Bus Data
        {
        if (num_bus > 0)
           {
              int bus_select;

              do{
              cout << "\nSelect Bus to Remove <1-" << num_bus << "> : ";
              cin  >> bus_select;
                 }
              while ((bus_select < 1)|(bus_select > num_bus));
              base_dat.num_bus -= 1;
              base_dat.write_to_disk(Filebase);
              cout << "\nRemove Bus No." << bus_select;
              display_bus_list();
              bus_dat[bus_select-1].display_data(bus_select,num_bus);
              display_end_line();
              getche();

              if (num_line > 0)
              {
              cout << "\nRemoved Bus Data has effect to Line Data and Transformer Dat
a.\n";
              cout << "\nRemoved Line Data.";
              display_line_list();
              int counter=0;
              do{
                 if ((strcmp(line_dat[counter].name1,bus_dat[bus_select-1].bus_name)=
=0)|(strcmp(line_dat[counter].name2,bus_dat[bus_select-1].bus_name)==0))
                    {
                       line_dat[counter].display_data(counter+1,num_line);
                       for (int i=counter,j=counter+1;j<num_line;i++,j++)
                          {
                             line_dat[i] = line_dat[j];
                          }
                       num_line -= 1;
                       counter -= 1;
                    }
                 counter += 1;
                 }
              while (counter != num_line);
              display_end_line();
              for (int i=0;i<num_line;i++)
```

```
            {
               if (line_dat[i].from_bus > bus_select)
                  line_dat[i].from_bus -= 1;
               if (line_dat[i].to_bus > bus_select)
                  line_dat[i].to_bus -= 1;
            }
         base_dat.num_line = num_line;
         base_dat.write_to_disk(Filebase);
         getche();
         }

         if (num_tran > 0)
         {
         cout << "\nRemove Transformer Data.";
         display_transformer_list();
         int counter=0;
         do{
            if ((strcmp(tran_dat[counter].name1,bus_dat[bus_select-1].bus_name)==0)||(strcmp(tran_dat[counter].name2,bus_dat[bus_select-1].bus_name)==0))
               {
                  tran_dat[counter].display_data(counter+1,num_tran);
                  for (int i=counter,j=counter+1;j<num_tran;i++,j++)
                    {}
                       tran_dat[i] = tran_dat[j];
                    }
                  num_tran -= 1;
                  counter -= 1;
               }
            counter += 1;
            }
         while (counter != num_tran);
         display_end_line();
         for (int i=0;i<num_tran;i++)
            {
              if (tran_dat[i].from_bus > bus_select)
                 tran_dat[i].from_bus -= 1;
              if (tran_dat[i].to_bus > bus_select)
                 tran_dat[i].to_bus -= 1;
            }
         base_dat.num_tran = num_tran;
         base_dat.write_to_disk(Filebase);
         }

         for (int i=bus_select-1,j=bus_select;j<num_bus;i++,j++)
            {
               bus_dat[i] = bus_dat[j];
            }
         int count_bus = 0;
         for (int i=0;i<(num_bus-1);i++)
            {
               if ((i+1)%5 == 0)
                  {
                  count_bus += 1;
                  for (int j=(i-4);j<i+1;j++)
                     {
                        bus_dat[j].write_to_disk(Filebus,j,(num_bus-1));
                     {{
                     }
                  else if ((i+1) == (num_bus-1))
                     for (int j=(count_bus*5);j<(num_bus-1);j++)
                        {
                           bus_dat[j].write_to_disk(Filebus,j,(num_bus-1));
                        }
            }
         int count_line = 0;
         for (int i=0;i<num_line;i++)
```

```
        {
            if ((i+1)%5 == 0)
                {
                    count_line += 1;
                    for (int j=(i-4);j<i+1;j++)
                        {
                            line_dat[j].write_to_disk(Fileline,j,num_line);
                        }
                }
            else if ((i+1) == num_line)
                    for (int j=(count_line*5);j<num_line;j++)
                        {
                            line_dat[j].write_to_disk(Fileline,j,num_line);
                        }
        }
    int count_tran = 0;
    for (int i=0;i<num_tran;i++)
        {
            if ((i+1)%5 == 0)
                {
                count_tran += 1;
                for (int j=(i-4);j<i+1;j++)
                    {
                        tran_dat[j].write_to_disk(Filetran,j,num_tran);
                    }
                }
            else if ((i+1) == num_tran)
                for (int j=(count_tran*5);j<num_tran;j++)
                    {
                        tran_dat[j].write_to_disk(Filetran,j,num_tran);
                    }
        }
    cout << "\nRemove Bus Data Complete.";
    num_bus -= 1;
    if ((base_dat.slack == bus_select)&(num_bus > 1))
        {
        int slack;

        cout << "\n\nSlack Bus must be changed by user!";
        getche();
        cout << "\n\nDisplay bus data from file : " << Filebus << "\n\n";
        display_bus_list();
        for (int i=0;i<num_bus;i++)
            {
                bus_dat[i].read_from_disk(Filebus,i);
                bus_dat[i].display_data((i+1),num_bus);
            }
        display_end_line();
        do{
            cout << "\nYou must choose new Slack Bus! <1-" << num_bus << "> :
";
            cin  >> slack;
            }
        while ((slack < 1)||(slack > num_bus));
        base_dat.slack = slack;
        base_dat.write_to_disk(Filebase);
        }
    else if (base_dat.slack > bus_select)
        {
            base_dat.slack -= 1;
            base_dat.write_to_disk(Filebase);
        }
    else if (num_bus == 1)
        {
        base_dat.slack = 1;
        base_dat.write_to_disk(Filebase);
```

```
            }
            else if (num_bus == 0)
               {
               base_dat.slack = 0;
               base_dat.write_to_disk(Filebase);
               }
            cout << "\n\nPress Return to be continue.";
            getche();
         }
         else
               {
               cout << "\nNo Buses Data for Remove!";
               cout << "\n\nPress Return to be continue.";
               getche();
               }
      }
               break;
      case '2' : //Remove Line Data
      {
      if (num_line > 0)
         {
            int line_select;

            do{
            cout << "\nSelect Line to Remove <1-" << num_line << "> : ";
            cin  >> line_select;
               }
            while ((line_select < 1)|(line_select > num_line));
            base_dat.num_line -= 1;
            base_dat.write_to_disk(Filebase);
            cout << "\nRemove Line No." << line_select;
            display_line_list();
            line_dat[line_select-1].display_data(line_select,num_line);
            display_end_line();
            for (int i=line_select-1,j=line_select;j<num_line;i++,j++)
               {
                line_dat[i] = line_dat[j];
               }
            int count_line = 0;
            for (int i=0;i<(num_line-1);i++)
              {
                if ((i+1)%5 == 0)
                   {
                   count_line += 1;
                   for (int j=(i-4);j<i+1;j++)
                      {
                         line_dat[j].write_to_disk(Fileline,j,(num_line-1));
                      }
                   }
                else if ((i+1) == (num_line-1))
                   for (int j=(count_line*5);j<(num_line-1);j++)
                      {
                         line_dat[j].write_to_disk(Fileline,j,(num_line-1));
                      }
              }
            cout << "\nRemove Line Data Complete.";
            num_line -= 1;
            cout << "\n\nPress Return to be continue.";
            getche();
      }
      else
            {
            cout << "\nNo Lines Data for Remove!";
            cout << "\n\nPress Return to be continue.";
            getche();
            }
```

```
    }
            break;
    case '3' : //Remove Transformer Data
    {
    if (num_tran > 0)
       {
          int tran_select;

          do{
          cout << "\nSelect Transformer to Remove <1-" << num_tran << "> : ";
          cin  >> tran_select;
             }
          while ((tran_select < 1)||(tran_select > num_tran));
          base_dat.num_tran -= 1;
          base_dat.write_to_disk(Filebase);
          cout << "\nRemove Transformer No." << tran_select;
          display_transformer_list();
          tran_dat[tran_select-1].display_data(tran_select,num_tran);
          display_end_line();
          for (int i=tran_select-1,j=tran_select;j<num_tran;i++,j++)
             {
                tran_dat[i] = tran_dat[j];
             }
          int count_tran = 0;
          for (int i=0;i<(num_tran-1);i++)
             {
                if ((i+1)%5 == 0)
                {
                count_tran +='1;
                for (int j=(i-4);j<i+1;j++)
                   {
                      tran_dat[j].write_to_disk(Filetran,j,(num_tran-1));
                   {}
                }
                else if ((i+1) == (num_tran-1))
                   for (int j=(count_tran*5);j<(num_tran-1);j++)
                   {
                      tran_dat[j].write_to_disk(Filetran,j,(num_tran-1));
                   }
             }
          cout << "\nRemove Transformer Data Complete.";
          num_tran -= 1;
          cout << "\n\nPress Return to be continue.";
          getche();
       }
       else
             {
             cout << "\nNo Transformers Data for Remove!";
             cout << "\n\nPress Return to be continue.";
             getche();
             }
    }
             break;
    case '4' : //Back
             break;
                }
       }
    while (option1!='4');
    break;
    }
    case '4' : //Back
             break;
                }
             }
          }
    }
```

```
    while (option!='4');
    break;
    }

case '4' : //Analyse Loadflow of Existing System
    {
    clrscr();
    do{
    cout << "Enter System Filename \n<Don't exceed 8 characters> : ";
    cin >> Filename;
      }
    while (strlen(Filename) > 8);
    strcpy(Filebase,Filename);
    strcpy(Filebus,Filename);
    strcpy(Fileline,Filename);
    strcpy(Filetran,Filename);
    strcat(Filebase,".bas");
    strcat(Filebus,".bus");
    strcat(Fileline,".lin");
    strcat(Filetran,".trn");
    cout << "\nFilename : " << Filebase << " , " << Filebus << " , ";
    cout << Fileline << " and " << Filetran;

    ifstream file1,file2,file3,file4;

    file1.open(Filebase,ios::in);
    file2.open(Filebus,ios::in);
    file3.open(Fileline,ios::in);
    file4.open(Filetran,ios::in);
    if ((!file1)|(!file2)|(!file3)|(!file4))
       {
       cout << "\nCannot open file " << Filebase << " , " << Filebus << " , ";
       cout << Fileline << " and " << Filetran << "\n";
       cout << "\n\nPress Return to be continue.";
       getche();
       }
    else {

    cout << "\n\nPress Return to Read and Display Data\n\n";
    getche();

    cout << "Display base data.\n";
    base_dat.read_from_disk(Filebase);
    num_bus = base_dat.num_bus;
    num_line = base_dat.num_line;
    num_tran = base_dat.num_tran;

    base_dat.display_data(Filebase);

    cout << "\n\nDisplay bus data from file : " << Filebus << "\n\n";
    display_bus_list();
    for (int i=0;i<num_bus;i++)
       {
          bus_dat[i].read_from_disk(Filebus,i);
          bus_dat[i].display_data((i+1),num_bus);
       }
    display_end_line();
    cout << "\n\nPress Return to be continue.";
    getche();

    cout << "\n\nDisplay line data from file : " << Fileline << "\n\n";
    display_line_list();
    for (int i=0;i<num_line;i++)
       {
          line_dat[i].read_from_disk(Fileline,i);
          line_dat[i].display_data((i+1),num_line);
```

```
        }
    display_end_line();
    cout << "\n\nPress Return to be continue.";
    getche();

    cout << "\n\nDisplay transformer data from file : " << Filetran << "\n\n";
    display_transformer_list();
    for (int i=0;i<num_tran;i++)
       {
          tran_dat[i].read_from_disk(Filetran,i);
          tran_dat[i].display_data((i+1),num_tran);
       }
    display_end_line();
    cout << "\n\nPress Return to be continue.";
    getche();

    do{
    clrscr();
    cout << "\n            Analyse Loadflow of Existing System";
    cout << "\n       Y-Bus Matrix and Load Flow Solution";
    cout << "\n\nMenu Options--->\n";
    cout << "\n    1.Calculate and Show Y-Bus Matrix";
    cout << "\n    2.Analyse Load Flow Solution";
    cout << "\n    3.Back <---";
    do{
    cout << "\n\nChoose your option ---> ";
    cin  >> option;
       }
    while ((option!='1')&(option!='2')&(option!='3'));

  switch (option) {
    case '1' : //Calculate and Show Y-Bus Matrix
          {
       {
       Y_bus ymatrix;

       cout << "\n\nPress Return Calculate and Display Y-Bus Matrix.";
       getche();
       ymatrix.ycal(num_bus,num_line,num_tran,bus_dat,line_dat,tran_dat);
       ymatrix.display_ybus(num_bus);
       }
            break;
          }
    case '2' : //Analysis Load Flow Solution
       {
       Y_bus ymatrix;
       Load_Flow loadflow;

       ymatrix.ycal(num_bus,num_line,num_tran,bus_dat,line_dat,tran_dat);
       loadflow.load_flow_cal(base_dat,bus_dat,ymatrix.ybus);
       if ((loadflow.delta_Pmax <= base_dat.power_tole)&(loadflow.delta_Qmax <= b
ase_dat.power_tole))
          {
          display_bus_solution_list();
          loadflow.display_final_bus_solution(base_dat,bus_dat,ymatrix.ybus);
          display_end_line();
          cout << "\n\nPress Return to Display Line Flows.\n";
          getche();
          display_line_flow_list();
          loadflow.display_line_flows(base_dat,line_dat);
          display_end_line();
          cout << "\n\nPress Return to Display Transformer Flows.\n";
          getche();
          display_transformer_flow_list();
          loadflow.display_transformer_flows(base_dat,tran_dat);
          display_end_line();
```

```
                cout << "\n\nPress Return to be continue.";
                getche();
                }
            }
                break;
        case '3' : //Back
                break;
                    }
            }
        while (option!='3');
             }
        break;
        }
    case '5' : //Exit
            break;
                    }
        }
    while (operation!='5');

    return 0;
}
```

```
//=============================================================//
//  KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG          //
//  PROJECT : POWER SYSTEM PLANNING                            //
//            LOADFLOW AND LOAD FORECASTING                    //
//  ADVISOR : MR. SOMPOTSH     PRAPAI                          //
//  ADVISEE : MR. YOCHAI       SASIWAN                         //
//            MR. ROONGROT     WANAPHURKSASILP                 //
//=============================================================//

//FILE :: PLANNING.H

#include <stdio.h>
#include <iostream.h>
#include <fstream.h>
#include <iomanip.h>
#include <conio.h>
#include <string.h>
#include <complex.h>
#include <time.h>
#include <dos.h>

const int MAX_BUS  = 50;
const int MAX_LINE = 50;
const int MAX_TRAN = 50;
const int BASE_MVA = 100;
const float pi = 3.141592654;
clock_t start, end;

void display_bus_list(void);
void display_line_list(void);
void display_transformer_list(void);
void display_end_line(void);
void display_bus_solution_list(void);
void display_line_flow_list(void);
void display_transformer_flow_list(void);

//Class declaration...

class Base
  {
    public :
       char index[];
       float power_tole;
       int slack,max_ite,num_bus,num_line,num_tran;
       void in_data(void);
       void write_to_disk(char Filename[]);
       void read_from_disk(char Filename[]);
       void display_data(char Filename[]);
       Base(void);
  };

class Bus
  {
    public :
       char index[];
       char bus_name[15];
       int  bus_type;
       float bus_volt,load_MW,load_MVAR,gen_MW,gen_MVAR,min_MVAR,max_MVAR,shunt_b;
       void in_bus_data(int num);
       void change_data(int num,char bus_name[]);
       void write_to_disk(char Filename[],int num,int num_bus);
       void read_from_disk(char Filename[],int num);
       void display_data(int num,int num_bus);
```

```
        Bus(void);
    };

class Line
    {
      public :
        char index[];
        char name1[15],name2[15];
        int from_bus,to_bus;
        float line_r,line_x,line_b;
        void in_line_data(char Filename[],int num_bus,int num);
        void change_data(int line_select,int from_bus,char name1[],int to_bus,char na
me2[]);
        void write_to_disk(char Filename[],int num,int num_line);
        void read_from_disk(char Filename[],int num);
        void display_data(int num,int num_line);
        Line(void);
    };

class Transformer
    {
      public :
        char index[];
        char name1[15],name2[15];
        int from_bus,to_bus;
        float tran_r,tran_x,tap;
        int code;
        void in_tran_data(char Filename[],int num_bus,int num);
        void change_data(int tran_select,int from_bus,char name1[],int to_bus,char na
me2[]);
        void write_to_disk(char Filename[],int num,int num_tran);
        void read_from_disk(char Filename[],int num);
        void display_data(int num,int num_tran);
        Transformer(void);
    };

class Y_bus
    {
      public :
        complex ybus[MAX_BUS][MAX_BUS];
        void ycal(int num_bus,int num_line,int num_tran,Bus bus_dat[],Line line_dat[]
,Transformer tran_dat[]);
        void display_ybus(int num_bus);
        Y_bus(void);
    };

class Load_Flow
    {
      public :
            complex volt[MAX_BUS],zbus[MAX_BUS][MAX_BUS];
            float v_mag[MAX_BUS],v_ang[MAX_BUS];
            float P_spec[MAX_BUS],Q_spec[MAX_BUS],P_cal[MAX_BUS],Q_cal[MAX_BUS];
            float delta_P[MAX_BUS],delta_Q[MAX_BUS];
            float delta_Pmax,delta_Qmax;
            float jacobian[2*(MAX_BUS-1)][2*(MAX_BUS-1)],PQ_matrix[2*(MAX_BUS-1)],AV_m
atrix[2*(MAX_BUS-1)];
            int check[MAX_BUS],type_buff[MAX_BUS];
            void invertmatrix(float firstmatrix[2*(MAX_BUS-1)][2*(MAX_BUS-1)],int dime
nsion);
            void load_flow_cal(Base base_dat,Bus bus_dat[],complex ybus[MAX_BUS][MAX_B
US]);
            void display_final_bus_solution(Base base_dat,Bus bus_dat[],complex ybus[M
AX_BUS][MAX_BUS]);
            void display_line_flows(Base base_dat,Line line_dat[]);
            void display_transformer_flows(Base base_dat,Transformer tran_dat[]);
            Load_Flow(void);
```

```
};

//Class function...

void Base::in_data(void)
  {
     cout << "\nInput data for Base data.\n\n";
     do{
     cout << "Number of Buses <2-" << MAX_BUS << "> : ";
     cin >> num_bus;
        }
     while ((num_bus < 2) | (num_bus > MAX_BUS));
     do{
     cout << "Number of lines <0-" << MAX_LINE << "> : ";
     cin >> num_line;
        }
     while ((num_line < 0) | (num_line > MAX_LINE));
     do{
     cout << "Number of transformers <0-" << MAX_TRAN << "> : ";
     cin >> num_tran;
        }
     while ((num_tran < 0) | (num_tran > MAX_TRAN));
     do{
     cout << "The slack bus <1-" << num_bus << "> : ";
     cin >> slack;
        }
     while ((slack < 1) | (slack > num_bus));
     do{
     cout << "Maximum Number of Iterations <1-~> : ";
     cin >> max_ite;
        }
     while (max_ite < 1);
     do{
     cout << "Power Tolerance <Greater than 0.00000) : ";
     cin >> power_tole;
        }
     while (power_tole < 0.00000);
     cout << "\nInput for Base data complete.\n";
  }

void Base::write_to_disk(char Filename[])
  {
     ofstream output_file;

     output_file.open(Filename,ios::binary);
     output_file.write((char*)index,sizeof (Base));
     output_file.close();
  }

void Base::read_from_disk(char Filename[])
  {
     ifstream input_file;

     input_file.open(Filename,ios::binary);
     input_file.read((char*)index,sizeof (Base));
     input_file.close();
  }

void Base::display_data(char Filename[])
  {
     cout << "\nBase data in file " << Filename;
     cout << "\n\nThe slack bus is Bus No." << slack;
     cout << "\nMaximum Number of Iterations -: " << max_ite;
     cout << "\nPower Tolerance -: " << setprecision(5) << power_tole;
     cout << "\nNumber of Buses -: " << num_bus;
     cout << "\nNumber of Lines -: " << num_line;
```

```
        cout << "\nNumber of Transformers -: " << num_tran;
        cout << "\n\nPress Return to be continue.";
        getche();
    }

Base::Base(void)
    {
        strcpy(index,"1");
    }

void Bus::in_bus_data(int num)
    {
        cout << "\nBus No." << num;
        do {
           cout << "\nBus Name <Don't exceed 14 characters.> : ";
           cin  >> bus_name;
           }
        while (strlen(bus_name) > 14 );
        do {
           cout << "Bus Type <0 or 1> : ";
           cin  >> bus_type;
           }
        while ((bus_type != 0) & (bus_type != 1));
        do{
        cout << "Bus Voltage <Not less than 0.00 in per-unit> : ";
        cin  >> bus_volt;
           }
        while (bus_volt < 0);
        do{
        cout << "Load <Not less than 0.00 in MW> : ";
        cin  >> load_MW;
           }
        while (load_MW < 0);
        do{
        cout << "Load <Not less than 0.00 in MVAR> : ";
        cin  >> load_MVAR;
           }
        while (load_MVAR < 0);
        do{
        cout << "Generation <Not less than 0.00 in MW> : ";
        cin  >> gen_MW;
           }
        while (gen_MW < 0);
        do{
        cout << "Generation <Not less than 0.00 in MVAR> : ";
        cin  >> gen_MVAR;
           }
        while (gen_MVAR < 0);
        cout << "Minimum <MVAR> : ";
        cin  >> min_MVAR;
        cout << "Maximum <MVAR> : ";
        cin  >> max_MVAR;
        cout << "Shunt Susceptance <per-unit> : ";
        cin  >> shunt_b;
    }

void Bus::change_data(int num,char bus_name[])
    {
        cout << "\nBus No." << num;
        cout << "\nBus Name : " << bus_name << "\n";
        do {
           cout << "Bus Type <0 or 1> : ";
           cin  >> bus_type;
           }
        while ((bus_type != 0) & (bus_type != 1));
        do{
```

```
    cout << "Bus Voltage <Not less than 0.00 in per-unit> : ";
    cin  >> bus_volt;
       }
    while (bus_volt < 0);
    do{
    cout << "Load <Not less than 0.00 in MW> : ";
    cin  >> load_MW;
       }
    while (load_MW < 0);
    do{
    cout << "Load <Not less than 0.00 in MVAR> : ";
    cin  >> load_MVAR;
       }
    while (load_MVAR < 0);
    do{
    cout << "Generation <Not less than 0.00 in MW> : ";
    cin  >> gen_MW;
       }
    while (gen_MW < 0);
    do{
    cout << "Generation <Not less than 0.00 in MVAR> : ";
    cin  >> gen_MVAR;
       }
    while (gen_MVAR < 0);
    cout << "Minimum <MVAR> : ";
    cin  >> min_MVAR;
    cout << "Maximum <MVAR> : ";
    cin  >> max_MVAR;
    cout << "Shunt Susceptance <per-unit> : ";
    cin  >> shunt_b;
}

void Bus::write_to_disk(char Filename[],int num,int num_bus)
{
    ofstream output_file;
    if (num == 0)
    output_file.open(Filename);
    else
    output_file.open(Filename,ios::app|ios::binary);
    output_file.seekp(num*sizeof (Bus),ios::beg);
    output_file.write((char*)index,sizeof (Bus));
    if (num == (num_bus-1)) output_file.close();
}

void Bus::read_from_disk(char Filename[],int num)
{
    ifstream input_file;

    input_file.open(Filename,ios::binary);
    input_file.seekg(num*sizeof (Bus),ios::beg);
    input_file.read((char*)index,sizeof (Bus));
    input_file.close();
}

void Bus::display_data(int num,int num_bus)
{
    cout << "  ";
    cout << setw(3) << setiosflags(ios::left) << num;
    cout << setiosflags(ios::showpoint);
    cout << setiosflags(ios::fixed);
    cout << setw(15) << setiosflags(ios::left) << bus_name;
    cout << setw(3) << setiosflags(ios::left) << bus_type;
    cout << setw(7) << setiosflags(ios::right) << setprecision(4) << bus_volt;
    cout << setw(7) << setprecision(2) << load_MW;
    cout << setw(7) << setprecision(2) << load_MVAR;
    cout << setw(8) << setprecision(2) << gen_MW;
```

```
    cout << setw(7) << setprecision(2) << gen_MVAR;
    cout << setw(6) << setprecision(2) << min_MVAR;
    cout << setw(7) << setprecision(2) << max_MVAR;
    cout << setw(7) << setprecision(3) << shunt_b;
    cout << "\n";
    if (((num%5) == 0)&(num != num_bus)) cout << "\n";
    if (((num%10) == 0)&(num != num_bus)) getche();
  }

Bus::Bus(void)
  {
    strcpy(index,"1");
    bus_name[0] = NULL;
    bus_type = 0;
    bus_volt = 0;
    load_MW = 0;
    load_MVAR = 0;
    gen_MW = 0;
    gen_MVAR = 0;
    min_MVAR = 0;
    max_MVAR = 0;
    shunt_b = 0;
  }

void Line::in_line_data(char Filename[],int num_bus,int num)
  {
    Bus bus_dat;
    int f_bus,t_bus;

    cout << "\nLine Number : " << num;
    do{
    cout << "\nFrom Bus Number <1-" << num_bus << "> : ";
    cin  >> f_bus;
      }
    while ((f_bus < 1)|(f_bus > num_bus));
    bus_dat.read_from_disk(Filename,f_bus-1);
    cout << "\n                    Name : ";
    cout << bus_dat.bus_name;
    strcpy(name1,bus_dat.bus_name);
    from_bus = f_bus;
    do{
    cout << "\nTo Bus Number <1-" << num_bus << " except " << f_bus << "> : ";
    cin  >> t_bus;
      }
    while ((t_bus < 1)|(t_bus > num_bus)|(t_bus == f_bus));
    bus_dat.read_from_disk(Filename,t_bus-1);
    cout << "\n                    Name : ";
    cout << bus_dat.bus_name;
    strcpy(name2,bus_dat.bus_name);
    to_bus = t_bus;
    do{
    cout << "\nResistance of Line <Not less than 0.00 in per-unit> : ";
    cin  >> line_r;
      }
    while (line_r < 0);
    do{
    cout << "Reactance of Line <Not less than 0.00 in per-unit> : ";
    cin  >> line_x;
      }
    while (line_x < 0);
    do{
    cout << "Susceptance of Line <Not less than 0.00 in per-unit> :.";
    cin  >> line_b;
      }
    while (line_b < 0);
  }
```

```
void Line::change_data(int line_select,int from_bus,char name1[],int to_bus,char name2
[])
    {
     cout << "\nLine Number : " << line_select;
     cout << "\nFrom Bus Number : " << from_bus;
     cout << "\n                        Name : ";
     cout << name1;
     cout << "\nTo Bus Number : " << to_bus;
     cout << "\n                        Name : ";
     cout << name2;
     do{
     cout << "\nResistance of Line <Not less than 0.00 in per-unit> : ";
     cin  >> line_r;
        }
     while (line_r < 0);
     do{
     cout << "Reactance of Line <Not less than 0.00 in per-unit> : ";
     cin  >> line_x;
        }
     while (line_x < 0);
     do{
     cout << "Susceptance of Line <Not less than 0.00 in per-unit> : ";
     cin  >> line_b;
        }
     while (line_b < 0);
    }

void Line::write_to_disk(char Filename[],int num,int num_line)
    {
     ofstream output_file;
     if (num == 0)
     output_file.open(Filename);
     else
     output_file.open(Filename,ios::app|ios::binary);
     output_file.seekp(num*sizeof (Line),ios::beg);
     output_file.write((char*)index,sizeof (Line));
     if (num == (num_line-1)) output_file.close();
    }

void Line::read_from_disk(char Filename[],int num)
    {
     ifstream input_file;

     input_file.open(Filename,ios::binary);
     input_file.seekg(num*sizeof (Line),ios::beg);
     input_file.read((char*)index,sizeof (Line));
     input_file.close();
    }

void Line::display_data(int num,int num_line)
    {
     cout << "    ";
     cout << setw(7) << setiosflags(ios::left) << num;
     cout << setiosflags(ios::showpoint);
     cout << setiosflags(ios::fixed);
     cout << setw(5) <<  from_bus;
     cout << setw(15) << setiosflags(ios::left) << name1;
     cout << setw(5) <<  to_bus;
     cout << setw(15) << setiosflags(ios::left) << name2;
     cout << setw(7) << setiosflags(ios::right) << setprecision(5) << line_r;
     cout << setw(10) << setprecision(5) << line_x;
     cout << setw(10) << setprecision(5) << line_b;
     cout << "\n";
     if (((num%5) == 0)&(num != num_line)) cout << "\n";
     if (((num%10) == 0)&(num != num_line)) getche();
    }
```

```
}

Line::Line(void)
  {
    strcpy(index,"1");
    name1[0] = '1';
    name2[0] = NULL;
    from_bus = 0;
    to_bus = 0;
    line_r = 0;
    line_x = 0;
    line_b = 0;
  }

void Transformer::in_tran_data(char Filename[],int num_bus,int num)
  {
    Bus bus_dat;
    int f_bus,t_bus;

    cout << "\nTransformer Number : " << num;
    do{
    cout << "\nFrom Bus Number <1-" << num_bus << "> : ";
    cin  >> f_bus;
      }
    while ((f_bus < 1)|(f_bus > num_bus));
    bus_dat.read_from_disk(Filename,f_bus-1);
    cout << "\n                    Name : ";
    cout << bus_dat.bus_name;
    strcpy(name1,bus_dat.bus_name);
    from_bus = f_bus;
    do{
    cout << "\nTo Bus Number <1-" << num_bus << " except " << f_bus << "> : ";
    cin  >> t_bus;
      }
    while ((t_bus < 1)|(t_bus > num_bus)|(t_bus == f_bus));
    bus_dat.read_from_disk(Filename,t_bus-1);
    cout << "\n                    Name : ";
    cout << bus_dat.bus_name;
    strcpy(name2,bus_dat.bus_name);
    to_bus = t_bus;
    do{
    cout << "\nResistance of Transformer <Not less than 0.00 in per-unit> : ";
    cin  >> tran_r;
      }
    while (tran_r < 0);
    do{
    cout << "Reactance of Transformer <Not less than 0.00 in per-unit> : ";
    cin  >> tran_x;
      }
    while (tran_x < 0);
    do{
    cout << "Tap Setting of Transformer <Greater than 0.00> : ";
    cin  >> tap;
      }
    while (tap <= 0);
    do {
       cout << "Code of Trnsformer <Tension 0=Low or 1=High> : ";
       cin  >> code;
       }
    while ((code != 0) & (code != 1));
  }

void Transformer::change_data(int tran_select,int from_bus,char name1[],int to_bus,cha
r name2[])
  {
    cout << "\nTransformer Number : " << tran_select;
```

```
    cout << "\n                              Name : ";
    cout << name1;
    cout << "\nTo Bus Number : " << to_bus;
    cout << "\n                              Name : ";
    cout << name2;
    do{
    cout << "\nResistance of Transformer <Not less than 0.00 in per-unit> : ";
    cin  >> tran_r;
       }
    while (tran_r < 0);
    do{
    cout << "Reactance of Transformer <Not less than 0.00 in per-unit> : ";
    cin  >> tran_x;
       }
    while (tran_x < 0);
    do{
    cout << "Tap Setting of Transformer <Greater than 0.00> : ";
    cin  >> tap;
       }
    while (tap <= 0);
    do {
       cout << "Code of Trnsformer <Tension 0=Low or 1=High> : ";
       cin  >> code;
       }
    while ((code != 0) & (code != 1));
  }

void Transformer::write_to_disk(char Filename[],int num,int num_tran)
  {
    ofstream output_file;
    if (num == 0)
    output_file.open(Filename);
    else
    output_file.open(Filename,ios::app|ios::binary);
    output_file.seekp(num*sizeof (Transformer),ios::beg);
    output_file.write((char*)index,sizeof (Transformer));
    if (num == (num_tran-1)) output_file.close();
  }

void Transformer::read_from_disk(char Filename[],int num)
  {
    ifstream input_file;

    input_file.open(Filename,ios::binary);
    input_file.seekg(num*sizeof (Transformer),ios::beg);
    input_file.read((char*)index,sizeof (Transformer));
    input_file.close();
  }

void Transformer::display_data(int num,int num_tran)
  {
    cout << "    ";
    cout << setw(8) << setiosflags(ios::left) << num;
    cout << setiosflags(ios::showpoint);
    cout << setiosflags(ios::fixed);
    cout << setw(4) <<  from_bus;
    cout << setw(16) << name1;
    cout << setw(4) <<  to_bus;
    cout << setw(15) << name2;
    cout << setiosflags(ios::right);
    cout << setw(7) << setprecision(5) << tran_r;
    cout << setw(9) << setprecision(5) << tran_x;
    cout << setw(6) << setprecision(3) << tap;
    cout << setw(4) << code;
    cout << "\n";
```

```
      cout << "\nFrom Bus Number : " << from_bus;
      cout << "\n                        Name : ";
      cout << name1;
      cout << "\nTo Bus Number : " << to_bus;
      cout << "\n                        Name : ";
      cout << name2;
      do{
      cout << "\nResistance of Transformer <Not less than 0.00 in per-unit> : ";
      cin  >> tran_r;
         }
      while (tran_r < 0);
      do{
      cout << "Reactance of Transformer <Not less than 0.00 in per-unit> : ";
      cin  >> tran_x;
         }
      while (tran_x < 0);
      do{
      cout << "Tap Setting of Transformer <Greater than 0.00> : ";
      cin  >> tap;
         }
      while (tap <= 0);
      do {
         cout << "Code of Trnsformer <Tension 0=Low or 1=High> : ";
         cin  >> code;
         }
      while ((code != 0) & (code != 1));
   }

void Transformer::write_to_disk(char Filename[],int num,int num_tran)
   {
      ofstream output_file;
      if (num == 0)
      output_file.open(Filename);
      else
      output_file.open(Filename,ios::app|ios::binary);
      output_file.seekp(num*sizeof (Transformer),ios::beg);
      output_file.write((char*)index,sizeof (Transformer));
      if (num == (num_tran-1)) output_file.close();
   }

void Transformer::read_from_disk(char Filename[],int num)
   {
      ifstream input_file;

      input_file.open(Filename,ios::binary);
      input_file.seekg(num*sizeof (Transformer),ios::beg);
      input_file.read((char*)index,sizeof (Transformer));
      input_file.close();
   }

void Transformer::display_data(int num,int num_tran)
   {
      cout << "    ";
      cout << setw(8) << setiosflags(ios::left) << num;
      cout << setiosflags(ios::showpoint);
      cout << setiosflags(ios::fixed);
      cout << setw(4) <<  from_bus;
      cout << setw(16) << name1;
      cout << setw(4) <<  to_bus;
      cout << setw(15) << name2;
      cout << setiosflags(ios::right);
      cout << setw(7) << setprecision(5) << tran_r;
      cout << setw(9) << setprecision(5) << tran_x;
      cout << setw(6) << setprecision(3) << tap;
      cout << setw(4) << code;
      cout << "\n";
```

```
    if (((num%5) == 0)&(num != num_tran)) cout << "\n";
    if (((num%10) == 0)&(num != num_tran)) getche();
}

Transformer::Transformer(void)
{
    strcpy(index,"1");
    name1[0] = NULL;
    name2[0] = NULL;
    from_bus = 0;
    to_bus = 0;
    tran_r = 0;
    tran_x = 0;
    tap = 0;
    code = 0;
}

void Y_bus::ycal(int num_bus,int num_line,int num_tran,Bus bus_dat[],Line line_dat[],T
ransformer tran_dat[])
{
    for (int i=0;i<num_bus;i++)
    {
        int row = i;
        int column = i;
        complex admittance(0,bus_dat[i].shunt_b);
        ybus[row][column] = admittance;
    }
    for (int i=0;i<num_line;i++)
    {
        int row = line_dat[i].from_bus-1;
        int column = line_dat[i].to_bus-1;
        complex impedance(line_dat[i].line_r,line_dat[i].line_x);
        complex admittance = (1/impedance);
        ybus[row][column] = (-admittance);
        ybus[column][row] = (-admittance);
    }
    for (int i=1;i<=num_bus;i++)
    {
        for (int j=0;j<num_line;j++)
        {
            complex impedance(line_dat[j].line_r,line_dat[j].line_x);
            complex admittance = (1/impedance);
            complex linecharge(0,line_dat[j].line_b/2);
            if (i==line_dat[j].from_bus)
            {
                ybus[i-1][i-1] += (admittance+linecharge);
            }
        }
        for (int j=0;j<num_line;j++)
        {
            complex impedance(line_dat[j].line_r,line_dat[j].line_x);
            complex admittance = (1/impedance);
            complex linecharge(0,line_dat[j].line_b/2);
            if (i==line_dat[j].to_bus)
            {
                ybus[i-1][i-1] += (admittance+linecharge);
            }
        }
    }
    for (int i=0;i<num_tran;i++)
    {
        int row = tran_dat[i].from_bus-1;
        int column = tran_dat[i].to_bus-1;
        complex impedance(tran_dat[i].tran_r,tran_dat[i].tran_x);
        complex admittance = (1/impedance);
        float a = tran_dat[i].tap;
```

```
            ybus[row][column] += (-admittance/a);
            ybus[column][row] += (-admittance/a);
        }
    for (int i=1;i<=num_bus;i++)
        {
            for (int j=0;j<num_tran;j++)
                {
                    if (tran_dat[j].code == 0)
                    {
                        if (i == tran_dat[j].to_bus)
                        {
                            complex impedance(tran_dat[j].tran_r,tran_dat[j].tran_x);
                            complex admittance = (1/impedance);
                            float a = tran_dat[j].tap;
                            ybus[i-1][i-1] += (admittance/(a*a));
                        }
                        else if (i == tran_dat[j].from_bus)
                        {
                            complex impedance(tran_dat[j].tran_r,tran_dat[j].tran_x);
                            complex admittance = (1/impedance);
                            ybus[i-1][i-1] += admittance;
                        }
                    }
                    else if (tran_dat[j].code == 1)
                    {
                        if (i == tran_dat[j].from_bus)
                        {
                            complex impedance(tran_dat[j].tran_r,tran_dat[j].tran_x);
                            complex admittance = (1/impedance);
                            float a = tran_dat[j].tap;
                            ybus[i-1][i-1] += (admittance/(a*a));
                        }
                        else if (i == tran_dat[j].to_bus)
                        {
                            complex impedance(tran_dat[j].tran_r,tran_dat[j].tran_x);
                            complex admittance = (1/impedance);
                            ybus[i-1][i-1] += admittance;
                        }
                    }
                }
        }
    }

void Y_bus::display_ybus(int num_bus)
    {
        cout << "\n\nShow bus admittance matrix dimension ( " << num_bus << "*" << num_b
us << " ).\n";
        getche();
        for (int row=0;row<num_bus;row++)
            {
                cout << "\nRow No." << (row+1) << "\n";
                for (int column=0;column<num_bus;column++)
                    {
                        cout << setiosflags(ios::left);
                        cout << setprecision(5) << ybus[row][column];
                    }
                if (((row+1)%2==0)&(row!=0)) getche();
            }
        cout << "\n\nEnd...";
        getche();
    }

Y_bus::Y_bus(void)
    {
        for (int i=0;i<MAX_BUS;i++)
            {
```

```
            for (int j=0;j<MAX_BUS;j++)
               {
                  ybus[i][j]=(0,0);
               }
         }
   }

void Load_Flow::invertmatrix(float firstmatrix[2*(MAX_BUS-1)][2*(MAX_BUS-1)],int dimen
sion)
   {
      float secondmatrix[2*(MAX_BUS-1)][2*(MAX_BUS-1)];

      for(int i=0;i<dimension;i++)
         {
         for(int j=0;j<dimension;j++)
            {
             float off_diagonal=0;
             float diagonal=1;
             if (i==j){secondmatrix[i][j]=diagonal;}
             if (i!=j){secondmatrix[i][j]=off_diagonal;}
            }
         }
      for(int i=0;i<dimension;i++)
         {
         float divider;
         divider = firstmatrix[i][i];
         for(int j=0;j<dimension;j++)
            {
            firstmatrix[i][j] = firstmatrix[i][j]/divider;
            secondmatrix[i][j] = secondmatrix[i][j]/divider;
            }
         for(int rowcounter=0;rowcounter<dimension;rowcounter++)
            {
            float multiplier=firstmatrix[rowcounter][i];
            if(rowcounter!=i)
               {
               if(i<rowcounter)
                  {
                  for(int columncounter=0;columncounter<dimension;columncounter++)
                     {
                     firstmatrix[rowcounter][columncounter]=(firstmatrix[i][columncoun
ter]*multiplier)-firstmatrix[rowcounter][columncounter];
                     secondmatrix[rowcounter][columncounter]=(secondmatrix[i][columnco
unter]*multiplier)-secondmatrix[rowcounter][columncounter];
                     }
                  }
               if(i>rowcounter)
                  {
                  for(int columncounter=0;columncounter<dimension;columncounter++)
                     {
                     firstmatrix[rowcounter][columncounter]=firstmatrix[rowcounter][co
lumncounter]-(firstmatrix[i][columncounter]*multiplier);
                     secondmatrix[rowcounter][columncounter]=secondmatrix[rowcounter][
columncounter]-(secondmatrix[i][columncounter]*multiplier);
                     }
                  }
               }
            }
         }
      for (int i=0;i<dimension;i++)
         {
         for (int j=0;j<dimension;j++)
            {
               firstmatrix[i][j] = secondmatrix[i][j];
            }
         }
```

```
}

void Load_Flow::load_flow_cal(Base base_dat,Bus bus_dat[],complex ybus[MAX_BUS][MAX_BUS])
{
start = clock();
   for (int i=0;i<base_dat.num_bus;i++)
      {
         P_spec[i] = (bus_dat[i].gen_MW - bus_dat[i].load_MW)/BASE_MVA;
         Q_spec[i] = (bus_dat[i].gen_MVAR - bus_dat[i].load_MVAR)/BASE_MVA;

         if (bus_dat[i].bus_volt == 0)
            volt[i] = complex(1,0);
         else volt[i] = complex(bus_dat[i].bus_volt,0);

         v_mag[i] = sqrt(norm(volt[i]));
         v_ang[i] = arg(volt[i]);
      }

   int k = 1; //iteration count...

   for (int i=0;i<base_dat.num_bus;i++)
      {
         type_buff[i] = bus_dat[i].bus_type;
      }

   do{         //Start iteration...
      delta_Pmax = 0;
      delta_Qmax = 0;

      for (int i=0;i<base_dat.num_bus;i++)
         {
            P_cal[i] = 0;
            Q_cal[i] = 0;
            delta_P[i] = 0;
            delta_Q[i] = 0;
            if (i != (base_dat.slack-1))
               {
               for (int j=0;j<base_dat.num_bus;j++)
                  {
                     P_cal[i] += (v_mag[i]*v_mag[j]*(real(ybus[i][j])*cos(v_ang[i]-
v_ang[j]) + imag(ybus[i][j])*sin(v_ang[i]-v_ang[j])));
                     Q_cal[i] += (v_mag[i]*v_mag[j]*(real(ybus[i][j])*sin(v_ang[i]-
v_ang[j]) - imag(ybus[i][j])*cos(v_ang[i]-v_ang[j])));
                  }

               if (bus_dat[i].bus_type == 1)
                  {
                  delta_P[i] = P_spec[i] - P_cal[i];
                  }
               else if (bus_dat[i].bus_type == 0)
                  {
                  delta_P[i] = P_spec[i] - P_cal[i];
                  delta_Q[i] = Q_spec[i] - Q_cal[i];
                  }

               if (bus_dat[i].bus_type == 1)
                  {
                  if ((delta_P[i] < 0)&((-delta_P[i]) > delta_Pmax)) delta_Pmax = (
-delta_P[i]);
                  else if ((delta_P[i] > 0)&(delta_P[i] > delta_Pmax)) delta_Pmax =
 delta_P[i];
                  }
               else if (bus_dat[i].bus_type == 0)
                  {
                  if ((delta_P[i] < 0)&((-delta_P[i]) > delta_Pmax)) delta_Pmax = (
```

```
-delta_P[i]);
                    else if ((delta_P[i] > 0)&(delta_P[i] > delta_Pmax)) delta_Pmax =
 delta_P[i];

                    if ((delta_Q[i] < 0)&((-delta_Q[i]) > delta_Qmax)) delta_Qmax = (
-delta_Q[i]);
                    else if ((delta_Q[i] > 0)&(delta_Q[i] > delta_Qmax)) delta_Qmax =
 delta_Q[i];
                    }
                }
            }

        // Check for out of loop...
        if ((delta_Pmax <= base_dat.power_tole)&(delta_Qmax <= base_dat.power_tole))

            break;

        //Check PV-bus...
        for (int i=0;i<base_dat.num_bus;i++)
            {
            if (i != (base_dat.slack-1))
                {
                if (type_buff[i] == 1)
                    {
                    if (bus_dat[i].max_MVAR != bus_dat[i].min_MVAR)
                        {
                        if (Q_cal[i] > (bus_dat[i].max_MVAR/BASE_MVA))
                            {
                                delta_Q[i] = (bus_dat[i].max_MVAR/BASE_MVA) - Q_cal[i];
                                type_buff[i] = 0;
                                check[i] = 1;
                            }
                        else if (Q_cal[i] < (bus_dat[i].min_MVAR/BASE_MVA))
                            {
                                delta_Q[i] = (bus_dat[i].min_MVAR/BASE_MVA) - Q_cal[i];
                                type_buff[i] = 0;
                                check[i] = 2;
                            }
                        }
                    }
                }
            }

        //Jacobian start here...
        int n=0; //Jacobian index...

        for (int i=0;i<base_dat.num_bus;i++)
            {
            if (i != (base_dat.slack-1))
                {
                int m=0; //Jacobian index...

                for (int j=0;j<base_dat.num_bus;j++)
                    {
                    if (j != (base_dat.slack-1))
                        {
                            switch (type_buff[i])
                                {
                                case 0 :
                                    {
                                    switch (type_buff[j])
                                        {
                                        case 0 :
                                            {
                                                if (i != j)
                                                {
```

```
                                    jacobian[n][m]=(v_mag[i]*v_mag[j]*(real(ybus[i]
[j])*sin(v_ang[i]-v_ang[j]) - imag(ybus[i][j])*cos(v_ang[i]-v_ang[j])));
                                    jacobian[n][m+1]=(v_mag[i]*v_mag[j]*(real(ybus[
i][j])*cos(v_ang[i]-v_ang[j]) + imag(ybus[i][j])*sin(v_ang[i]-v_ang[j])));
                                    jacobian[n+1][m]=-jacobian[n][m+1];
                                    jacobian[n+1][m+1]=jacobian[n][m];
                                    }
                                    else if (i == j)
                                    {
                                    jacobian[n][m]=-Q_cal[i]-(imag(ybus[i][i])*(v_m
ag[i]*v_mag[i]));
                                    jacobian[n][m+1]=P_cal[i]+(real(ybus[i][i])*(v_
mag[i]*v_mag[i]));
                                    jacobian[n+1][m]=P_cal[i]-(real(ybus[i][i])*(v_
mag[i]*v_mag[i]));
                                    jacobian[n+1][m+1]=Q_cal[i]-(imag(ybus[i][i])*(
v_mag[i]*v_mag[i]));
                                    }
                                }
                                break;
                            case 1 :
                                {
                                    jacobian[n][m]=(v_mag[i]*v_mag[j]*(real(ybus[i]
[j])*sin(v_ang[i]-v_ang[j]) - imag(ybus[i][j])*cos(v_ang[i]-v_ang[j])));
                                    jacobian[n+1][m]=-(v_mag[i]*v_mag[j]*(real(ybus
[i][j])*cos(v_ang[i]-v_ang[j]) + imag(ybus[i][j])*sin(v_ang[i]-v_ang[j])));
                                }
                                break;
                            }
                        }
                        break;
                    case 1 :
                        {
                        switch (type_buff[j])
                            {
                            case 0 :
                                {
                                jacobian[n][m]=(v_mag[i]*v_mag[j]*(real(ybus[i][j]
)*sin(v_ang[i]-v_ang[j]) - imag(ybus[i][j])*cos(v_ang[i]-v_ang[j])));
                                jacobian[n][m+1]=(v_mag[i]*v_mag[j]*(real(ybus[i][
j])*cos(v_ang[i]-v_ang[j]) + imag(ybus[i][j])*sin(v_ang[i]-v_ang[j])));
                                }
                                break;
                            case 1 :
                                {
                                if (i != j)
                                    {
                                    jacobian[n][m]=(v_mag[i]*v_mag[j]*(real(ybus[i]
[j])*sin(v_ang[i]-v_ang[j]) - imag(ybus[i][j])*cos(v_ang[i]-v_ang[j])));
                                    }
                                else if (i == j)
                                    {
                                    jacobian[n][m]=-Q_cal[i]-(imag(ybus[i][i])*(v_m
ag[i]*v_mag[i]));
                                    }
                                }
                                break;
                            }
                        }
                        break;
                    }
                switch (type_buff[j])
                    {
                    case 0 : m = m+2;
                             break;
                    case 1 : m = m+1;
```

```
                                    break;
                                }
                            }
                        }
                    switch (type_buff[i])
                        {
                        case 0 :
                            {
                                PQ_matrix[n] = delta_P[i];
                                PQ_matrix[n+1] = delta_Q[i];
                                n = n+2;
                            }
                            break;
                        case 1 :
                            {
                                PQ_matrix[n] = delta_P[i];
                                n = n+1;
                            }
                            break;
                        }
                    }
                }

            invertmatrix(jacobian,n);

            for (int i=0;i<n;i++)
                {
                    AV_matrix[i] = 0;
                    for (int j=0;j<n;j++)
                        {
                            AV_matrix[i] += jacobian[i][j]*PQ_matrix[j];
                        }
                }

            int x=0;
            for (int i=0;i<base_dat.num_bus;i++)
                {
                    if (i != (base_dat.slack-1))
                    {
                        switch (type_buff[i])
                            {
                            case 0 :
                                {
                                    v_ang[i] = v_ang[i]+AV_matrix[x];
                                    v_mag[i] *= (1+AV_matrix[x+1]);
                                    x = x+2;
                                }
                                break;
                            case 1 :
                                {
                                    v_ang[i] = v_ang[i]+AV_matrix[x];
                                    x = x+1;
                                }
                                break;
                            }
                    }
                    volt[i] = polar(v_mag[i],v_ang[i]);
                }

            k += 1;
                }
            while (k <= base_dat.max_ite);


        end = clock();
        struct date d;
```

```
    struct  time t;
    getdate(&d);
    gettime(&t);

    if (k > base_dat.max_ite)
        {
           cout << "\n\nSolution don't converge in " << (k-1) << " iterations.\n";
           printf("The current date is: %d/%d/%d\n",d.da_day,d.da_mon,d.da_year);
           printf("The current time is: %2d:%02d:%02d.%02d\n",t.ti_hour, t.ti_min, t.
ti_sec, t.ti_hund);
           cout << "\nPress Return to be continue.\n";
           getche();
        }
     else if ((delta_Pmax <= base_dat.power_tole)&(delta_Qmax <= base_dat.power_tole)
)
        {
           cout << "\n\nSolution converge in " << (k-1) << " iterations.";
           cout << "      Time left = " << ((end-start)/CLK_TCK) << " seconds.\n";
           printf("The current date is: %d/%d/%d\n",d.da_day,d.da_mon,d.da_year);
           printf("The current time is: %2d:%02d:%02d.%02d\n\n",t.ti_hour, t.ti_min,
t.ti_sec, t.ti_hund);
           cout << "\nPress Return to Display Final Data.\n";
           getche();

        }
    }

void Load_Flow::display_final_bus_solution(Base base_dat,Bus bus_dat[],complex ybus[MA
X_BUS][MAX_BUS])
    {
    float total_gen_MW=0,total_gen_MVAR=0;
    float total_load_MW=0,total_load_MVAR=0;

    for (int i=0;i<base_dat.num_bus;i++)
       {
        cout << setw(3) << setiosflags(ios::right|ios::fixed) << (i+1) << " ";
        cout << setw(15) << setiosflags(ios::left) << bus_dat[i].bus_name;
        if ((i+1) == base_dat.slack)
           cout << "SL";
        else if ((i+1) != base_dat.slack)
           {
           if (bus_dat[i].bus_type == 1)
              cout << "PV";
           else if (bus_dat[i].bus_type == 0)
              cout << "PQ";
           }
        cout << setw(8) << setprecision(4) << setiosflags(ios::right) << v_mag[i];
        cout << setw(9) << setprecision(4) << (v_ang[i]*180/pi);
        if ((i+1) == base_dat.slack)
           {
           bus_dat[i].gen_MW  = 0;
           bus_dat[i].gen_MVAR = 0;

           for (int j=0;j<base_dat.num_bus;j++)
              {
                 bus_dat[i].gen_MW += (v_mag[i]*v_mag[j]*(real(ybus[i][j])*cos(v_ang[
i]-v_ang[j]) + imag(ybus[i][j])*sin(v_ang[i]-v_ang[j])));
                 bus_dat[i].gen_MVAR += (v_mag[i]*v_mag[j]*(real(ybus[i][j])*sin(v_an
g[i]-v_ang[j]) - imag(ybus[i][j])*cos(v_ang[i]-v_ang[j])));
              }
           bus_dat[i].gen_MW = (bus_dat[i].gen_MW*BASE_MVA)+bus_dat[i].load_MW;
           bus_dat[i].gen_MVAR = (bus_dat[i].gen_MVAR*BASE_MVA)+bus_dat[i].load_MVAR;
           }

        if (((i+1) != base_dat.slack)&(bus_dat[i].bus_type == 1))
           {
```

```
        bus_dat[i].gen_MVAR = 0;
        for (int j=0;j<base_dat.num_bus;j++)
           {
              bus_dat[i].gen_MVAR += (v_mag[i]*v_mag[j]*(real(ybus[i][j])*sin(v_an
g[i]-v_ang[j]) - imag(ybus[i][j])*cos(v_ang[i]-v_ang[j])));
           }

        if ((bus_dat[i].max_MVAR != bus_dat[i].min_MVAR)&(bus_dat[i].bus_type != t
ype_buff[i]))
           {
           if (check[i] == 1)
              {
              bus_dat[i].gen_MVAR = bus_dat[i].max_MVAR;
              }
           else if (check[i] == 2)
              {
              bus_dat[i].gen_MVAR = bus_dat[i].min_MVAR;
              }
           }
        else bus_dat[i].gen_MVAR = (bus_dat[i].gen_MVAR*BASE_MVA)+bus_dat[i].load_
MVAR;

        }

     cout << setw(9) << setprecision(2) << bus_dat[i].gen_MW;
     total_gen_MW += bus_dat[i].gen_MW;
     cout << setw(11) << setprecision(2) << bus_dat[i].gen_MVAR;
     total_gen_MVAR += bus_dat[i].gen_MVAR;
     cout << setw(10) << setprecision(2) << bus_dat[i].load_MW;
     total_load_MW += bus_dat[i].load_MW;
     cout << setw(10) << setprecision(2) << bus_dat[i].load_MVAR;
     total_load_MVAR += bus_dat[i].load_MVAR;
     cout << "\n";
     if ((((i+1)%5) == 0)&((i+1) != base_dat.num_bus)) cout << "\n";
     if ((((i+1)%10) == 0)&((i+1) != base_dat.num_bus)) getche();
  }
  display_end_line();
  cout << "          Area totals          ";
  cout << setw(9) << setprecision(2) << total_gen_MW;
  cout << setw(11)  << setprecision(2) << total_gen_MVAR;
  cout << setw(10)  << setprecision(2) << total_load_MW;
  cout << setw(10)  << setprecision(2) << total_load_MVAR;
  cout << "\n";
}

void Load_Flow::display_line_flows(Base base_dat,Line line_dat[])
{
  complex line_MVA1[MAX_LINE],line_MVA2[MAX_LINE],serie_y[MAX_LINE][MAX_LINE],half
_y[MAX_LINE];
  int j,k;

  for (int i=0;i<base_dat.num_line;i++)
     {
        complex serie_z_buff(line_dat[i].line_r,line_dat[i].line_x);
        complex serie_y_buff = (1/serie_z_buff);
        serie_y[line_dat[i].from_bus-1][line_dat[i].to_bus-1] = serie_y_buff;
        serie_y[line_dat[i].to_bus-1][line_dat[i].from_bus-1] = serie_y_buff;

        complex half_y_buff(0,line_dat[i].line_b/2);
        half_y[i] = half_y_buff;
     }

  for (int i=0;i<base_dat.num_line;i++)
     {
        cout << setw(3) << setiosflags(ios::right|ios::fixed) << (i+1);
        j = line_dat[i].from_bus;
        cout << setw(6) << j << "  ";
```

```
            cout << setw(15) << setiosflags(ios::left) << line_dat[i].name1;
            k = line_dat[i].to_bus;
            cout << setw(3) << setiosflags(ios::right|ios::fixed) << k << "  ";
            cout << setw(15) << setiosflags(ios::left) << line_dat[i].name2;

            line_MVA1[i] = (v_mag[(j-1)]*v_mag[(j-1)]*conj(serie_y[(j-1)][(k-1)]))-(vo
lt[(j-1)]*conj(volt[(k-1)])*conj(serie_y[(j-1)][(k-1)]))+(v_mag[(j-1)]*v_mag[(j-1)]*co
nj(half_y[i]));

            cout << setprecision(4) << setiosflags(ios::right|ios::fixed);
            cout << setw(11) << (real(line_MVA1[i])*BASE_MVA);
            cout << setw(11) << (imag(line_MVA1[i])*BASE_MVA);
            cout << setw(11) << (sqrt(norm(line_MVA1[i]))*BASE_MVA) << "\n";

            cout << setw(9) << k << "  ";
            cout << setw(15) << setiosflags(ios::left) << line_dat[i].name2;
            cout << setw(3) << setiosflags(ios::right|ios::fixed) << j << "  ";
            cout << setw(15) << setiosflags(ios::left) << line_dat[i].name1;

            line_MVA2[i] = (v_mag[(k-1)]*v_mag[(k-1)]*conj(serie_y[(j-1)][(k-1)]))-(vo
lt[(k-1)]*conj(volt[(j-1)])*conj(serie_y[(j-1)][(k-1)]))+(v_mag[(k-1)]*v_mag[(k-1)]*co
nj(half_y[i]));

            cout << setprecision(4) << setiosflags(ios::right|ios::fixed);
            cout << setw(11) << (real(line_MVA2[i])*BASE_MVA);
            cout << setw(11) << (imag(line_MVA2[i])*BASE_MVA);
            cout << setw(11) << (sqrt(norm(line_MVA2[i]))*BASE_MVA) << "\n";

            if ((((i+1)%2) == 0)&((i+1) != base_dat.num_line)) cout << "\n";
            if ((((i+1)%4) == 0)&((i+1) != base_dat.num_line)) getche();
        }
    }

void Load_Flow::display_transformer_flows(Base base_dat,Transformer tran_dat[])
    {
      complex tran_MVA1[MAX_TRAN],tran_MVA2[MAX_TRAN],y_leak[MAX_TRAN];
      int j,k;

      for (int i=0;i<base_dat.num_tran;i++)
          {
            complex z_leak_buff(tran_dat[i].tran_r,tran_dat[i].tran_x);
            y_leak[i] = 1/z_leak_buff;
          }

      for (int i=0;i<base_dat.num_tran;i++)
          {
            cout << setw(3) << setiosflags(ios::right|ios::fixed) << (i+1);

         if (tran_dat[i].code == 0)
            {
            j = tran_dat[i].to_bus;
            k = tran_dat[i].from_bus;

            cout << setw(6) << k << " ";
            cout << setw(15) << setiosflags(ios::left) << tran_dat[i].name2;
            cout << setw(3) << setiosflags(ios::right|ios::fixed) << j << " ";
            cout << setw(15) << setiosflags(ios::left) << tran_dat[i].name1;

            tran_MVA2[i] = (v_mag[(k-1)]*v_mag[(k-1)]*conj(y_leak[i]))-((1/tran_dat[i]
.tap)*volt[(k-1)]*conj(volt[(j-1)])*conj(y_leak[i]));

            cout << setprecision(2) << setiosflags(ios::right|ios::fixed);
            cout << setw(7) << (real(tran_MVA2[i])*BASE_MVA);
            cout << setw(11) << (imag(tran_MVA2[i])*BASE_MVA);
            cout << setw(10) << (sqrt(norm(tran_MVA2[i]))*BASE_MVA);
            cout << setw(7) << setprecision(3) << tran_dat[i].tap << "\n";
```

```
            cout << setw(9) << j << " ";
            cout << setw(15) << setiosflags(ios::left) << tran_dat[i].name1;
            cout << setw(3) << setiosflags(ios::right|ios::fixed) << k << " ";
            cout << setw(15) << setiosflags(ios::left) << tran_dat[i].name2;

            tran_MVA1[i] = ((1/(tran_dat[i].tap*tran_dat[i].tap))*v_mag[(j-1)]*v_mag[(
j-1)]*conj(y_leak[i]))-((1/tran_dat[i].tap)*volt[(j-1)]*conj(volt[(k-1)])*conj(y_leak[
i]));

            cout << setprecision(2) << setiosflags(ios::right|ios::fixed);
            cout << setw(7) << (real(tran_MVA1[i])*BASE_MVA);
            cout << setw(11) << (imag(tran_MVA1[i])*BASE_MVA);
            cout << setw(10) << (sqrt(norm(tran_MVA1[i]))*BASE_MVA) << "\n";
            }
         else if (tran_dat[i].code == 1)
            {
            j = tran_dat[i].from_bus;
            k = tran_dat[i].to_bus;

            cout << setw(6) << j << " ";
            cout << setw(15) << setiosflags(ios::left) << tran_dat[i].name1;
            cout << setw(3) << setiosflags(ios::right|ios::fixed) << k << " ";
            cout << setw(15) << setiosflags(ios::left) << tran_dat[i].name2;

            tran_MVA1[i] = ((1/(tran_dat[i].tap*tran_dat[i].tap))*v_mag[(j-1)]*v_mag[(
j-1)]*conj(y_leak[i]))-((1/tran_dat[i].tap)*volt[(j-1)]*conj(volt[(k-1)])*conj(y_leak[
i]));

            cout << setprecision(2) << setiosflags(ios::right|ios::fixed);
            cout << setw(7) << (real(tran_MVA1[i])*BASE_MVA);
            cout << setw(11) << (imag(tran_MVA1[i])*BASE_MVA);
            cout << setw(10) << (sqrt(norm(tran_MVA1[i]))*BASE_MVA);
            cout << setw(7) << setprecision(3) << tran_dat[i].tap << "\n";

            cout << setw(9) << k << " ";
            cout << setw(15) << setiosflags(ios::left) << tran_dat[i].name2;
            cout << setw(3) << setiosflags(ios::right|ios::fixed) << j << " ";
            cout << setw(15) << setiosflags(ios::left) << tran_dat[i].name1;

            tran_MVA2[i] = (v_mag[(k-1)]*v_mag[(k-1)]*conj(y_leak[i]))-((1/tran_dat[i]
.tap)*volt[(k-1)]*conj(volt[(j-1)])*conj(y_leak[i]));

            cout << setprecision(2) << setiosflags(ios::right|ios::fixed);
            cout << setw(7) << (real(tran_MVA2[i])*BASE_MVA);
            cout << setw(11) << (imag(tran_MVA2[i])*BASE_MVA);
            cout << setw(10) << (sqrt(norm(tran_MVA2[i]))*BASE_MVA) << "\n";
            }

            if ((((i+1)%2) == 0)&((i+1) != base_dat.num_tran)) cout << "\n";
            if ((((i+1)%4) == 0)&((i+1) != base_dat.num_tran)) getche();
         }
      }

Load_Flow::Load_Flow(void)
   {
      for (int i=0;i<(2*(MAX_BUS-1));i++)
         {
            PQ_matrix[i] = 0;
            for (int j=0;j<(2*(MAX_BUS-1));j++)
               {
                  jacobian[i][j] = 0;
               }
         }
      for (int i=0;i<MAX_BUS;i++)
         {
```

```
    }
}

//Main Function of Program Load Flow and Load Forecasting...

void display_bus_list(void)
{
    cout << "\n                          ********** Display Bus Data **********\n";
    cout << " ------------------------------------------------------------------------------------------
_____\n";
    cout << "|Bus|     Name     |Type| Volts |    - Load -    |-Gerneration-| Min   | Max
 |Shunt|\n";
    cout << "|No.|              |    | (pu)  |  MW   |  MVAR |  MW   | MVAR | MVAR | MVA
R |B(pu)|\n";
    cout << "|---+--------------+----+-------+-------+-------+-------+------+------+----
--+-----|";
}

void display_line_list(void)
{
    cout << "\n                 ********** Display Transmission Line Data **********\
n";
    cout << " ------------------------------------------------------------------------------------------
_____\n";
    cout << " | Line | From |     Name     | To   |     Name     |     R     |     X     |
  B    |\n";
    cout << " |  No. | Bus  |              | Bus  |              |  (p.u)    |  (p.u)    |
(p.u)  |\n";
    cout << " |------+------+--------------+------+--------------+-----------+-----------+--
-------|\n";
}

void display_transformer_list(void)
{
    cout << "\n                 ********** Display Transformer Data **********\n";
    cout << " ------------------------------------------------------------------------------------------
_____\n";
    cout << " | Tran. | From |     Name     | To   |     Name     |     R     |     X     | TA
P |CODE|\n";
    cout << " |  No.  | Bus  |              | Bus  |              |  (p.u)    |  (p.u)    |
   |\n";
    cout << " |-------+------+--------------+------+--------------+-----------+-----------+---
--+----|\n";
}
void display_end_line(void)
{
    cout << " ------------------------------------------------------------------------------------------
--------\n";
}
void display_bus_solution_list(void)
{
    cout << "\n                 ********** Display Buses Solution Converged **********
\n";
    cout << " ------------------------------------------------------------------------------------------
_____\n";
    cout << "|Bus|     Name     |Type| Volts | Angle |    - Generation -    |      - Load
-       |\n";
    cout << "|No.|              |    |  (p.u) |( deg.)|   MW     |   MVAR    |   MW     |
 MVAR  |\n";
    cout << "|---+--------------+----+-------+-------+----------+-----------+----------+-
--+-----|";
}
void display_line_flow_list(void)
{
```

```
      cout << "\n                ***** Display Power Flows on Transmission Lines ***
\n";
      cout << "  ____________________________________________________________________
_____\n";
      cout << "|Line|From|     Name      |To  |      Name      |          - Line Flows
  |\n";
      cout << "| No.|Bus |               |Bus |                |    MW    |   MVAR
 MVA   |\n";
      cout << "|----+----+---------------+----+----------------+----------+----------
-------|\n";
  }
void display_transformer_flow_list(void)
  {
      cout << "\n                ***** Display Power Flows through Transformers ****
n";
      cout << "  ____________________________________________________________________
_____\n";
      cout << "|Tran|From|     Name     |To  |     Name     |    - Transformer Flows -
 | TAP |\n";
      cout << "| No.|Bus |              |Bus |              |   MW   |   MVAR   |   MV
  |     |\n";
      cout << "|----+----+--------------+----+--------------+--------+----------+-----
-+-----|\n";
  }
```

Case study I (Network name : Beta)

THE SLACK BUS IS NUMBER 1

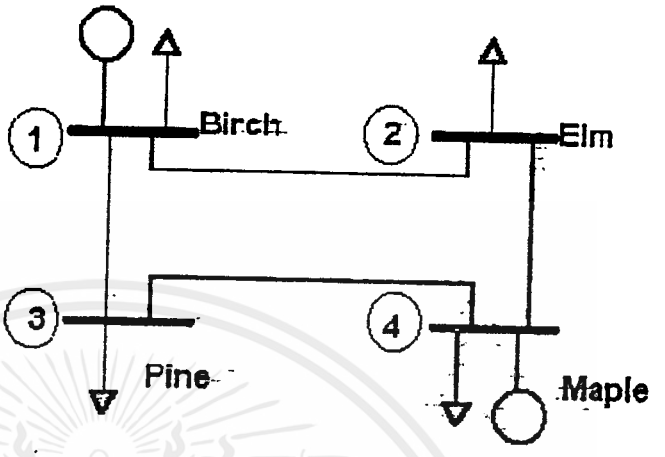
MAXIMUM NUMBER OF ITERATION = 5

POWER TOLERANCE = 0.0001

NUMBER OF BUSES = 4

NUMBER OF LINES = 4

NUMBER OF TRANSFORMER = 0



BUS	NAME	TYPE	VOLTAGE	LOAD		GENERATION		MAXIMUM	MINIMUM	SHUNT
				MW	MVAR	MW	MVAR	MVAR	MVAR	SUSCEPTANCE
1	Birch	1	1	50	30.99	0	0	0	0	0
2	Elm	0	1	170	105.35	0	0	0	0	0
3	Pine	0	1	200	123.94	0	0	0	0	0
4	Maple	1	1.02	80	49.58	318	0	0	0	0

LINES DATA

BUS	NAME	BUS	NAME	R	X	B
1	Birch	2	Elm	0.01008	0.0504	0.1025
1	Birch	3	Pine	0.00744	0.0372	0.0775
2	Elm	4	Maple	0.00744	0.0372	0.0775
3	Elm	4	Maple	0.01272	0.0636	0.1275

8.985190 -j44.835953	-3.815629 +j19.078144	-5.169561 +j25.847809	0
-3.815629 +j19.078144	8.985190 -j44.835953	0	-5.169561 +j25.847809
-5.169561 +j25.847809	0	8.193267 -j40.863838	-3.023705 +j15.118528
0	-5169561 +j25.847809	-3.023705 +j15.118528	8.193267 -j40.863838

Admittance bus matrix of "Beta" system.

Bus no.	Name	Volts (p.u.)	Angle (deg.)	Generation (MW)	Generation (Mvar)	Load (MW)	Load (Mvar)	Bus Type	To Bus	To Bus Name	Line Flow (MW)	Line Flow (Mvar)
1	Birch	1.000	0	186.81	114.50	50.00	30.99	SL	2	Elm	38.69	22.30
									3	Pine	98.12	61.21
2	Elm	0.982	-0.976	0	0	170.00	105.35	PQ	1	Birch	-38.46	-31.24
									4	Maple	-131.54	-74.11
3	Pine	0.969	-1.872	0	0	200.00	123.94	PQ	1	Birch	-97.09	-63.57
									4	Maple	-102.91	-60.37
4	Maple	1.020	1.523	318.00	181.43	80.00	49.58	PV	2	Elm	133.25	74.92
									3	Pine	104.75	56.93
Area totals				504.81	295.93	500.00	309.86					

Final buses solution of "Beta" system.

Case study II (Network name : alpha)

THE SLACK BUS IS NUMBER 1

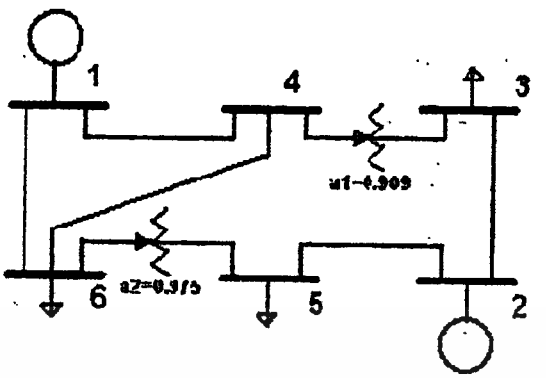
MAXIMUM NUMBER OF ITERATION = 5

POWER TOLERANCE = 0.001

NUMBER OF BUSES = 6

NUMBER OF LINES = 5

NUMBER OF TRANSFORMER = 2



BUSES DATA

BUS	NAME	TYPE	VOLTAGE	LOAD		GENERATION		MAXIMUM	MINIMUM	SHUNT
				MW	MVAR	MW	MVAR	MVAR	MVAR	SUSCEPTANCE
1	FIRST	1	1.05	0	0	0	0	0	0	0
2	SECOND	1	1.1	0	0	50	0	0	0	0
3	THIRD	0	0	55	13	0	0	0	0	0
4	FORTH	0	0	0	0	0	0	0	0	0
5	FIFTH	0	0	30	18	0	0	0	0	0
6	SIXTH	0	0	50	5	0	0	0	0	0

LINES DATA

BUS	NAME	BUS	NAME	R	X	B
1	FIRST	4	FORTH	0.08	0.37	0.03
1	FIRST	6	SIXTH	0.123	0.518	0.042
2	SECOND	3	THIRD	0.723	1.05	0
2	SECOND	5	FIFTH	0.282	0.64	0
4	FORTH	6	SIXTH	0.097	0.407	0.03

TRANSFORMERS DATA

BUS	NAME	BUS	NAME	R	X	TAP	CODE
3	THIRD	4	FORTH	0	0.133	0.909	0
5	FIFTH	6	SIXTH	0	0.3	0.975	0

0.9922 -j4.37346			-0.5583 +j2.58200		-0.43393 +j1.82746
	1.02140 -j1.95452	-0.44486 +j0.64606		-0.57654 +j1.30846	
	-0.44486 +j0.64606	0.44486 -j8.16486	0.000 +j8.27150		
-0.55827 +j2.58200		0.000 +j8.27150	1.11237 -j13.97650		-0.55410 +j2.32494
	-0.57654 +j1.30846			0.57654 -j4.64179	0.000 +j3.41880
-0.43393 +j1.82746			-0.55410 +j2.32494	0.000 +j3.41880	0.98804 -j7.62287

Admittance bus matrix of "Alpha" system.

Bus No.	Voltage Magnitude	Phase angles	Active power	Reactive power
1	1.05000	0.00000	0.95206	0.43252
2	1.10000	-3.34294	0.50016	0.18426
3	1.00080	-12.78397	-0.54999	-0.12987
4	0.92976	-9.83605	-0.00002	-0.00001
5	0.91981	-12.33389	-0.30005	-0.17984
6	0.91920	-12.23905	-0.50002	-0.05008

Voltage magnitude and angle solution of "Alpha" system.

Line	Buses		Power flow		Buses		Power flow	
No.	connected		Active	Reactive	Connected		Active	Reactive
1	1	4	0.50907	0.25339	4	1	-0.48497	-0.17147
2	1	6	0.44300	0.17913	6	1	-0.41654	-0.10860
3	2	3	0.17183	-0.00019	3	2	-0.15419	0.02582
4	2	5	0.32832	0.18446	5	2	-0.29527	-0.10945
5	3	4	-0.39580	-0.15569	4	3	0.39580	0.17971
6	4	6	0.08916	-0.00824	6	4	-0.08827	-0.01364
7	5	6	-0.00478	-0.07040	6	5	0.00478	0.07216

Power Flow solution of "Alpha" system.

Case study III (Network name : gamma)

THE SLACK BUS IS NUMBER 1

MAXIMUM NUMBER OF ITERATION = 10

POWER TOLERANCE = 0.001

NUMBER OF BUSES = 16

NUMBER OF LINES = 9

NUMBER OF TRANSFORMER = 7

BUSES DATA

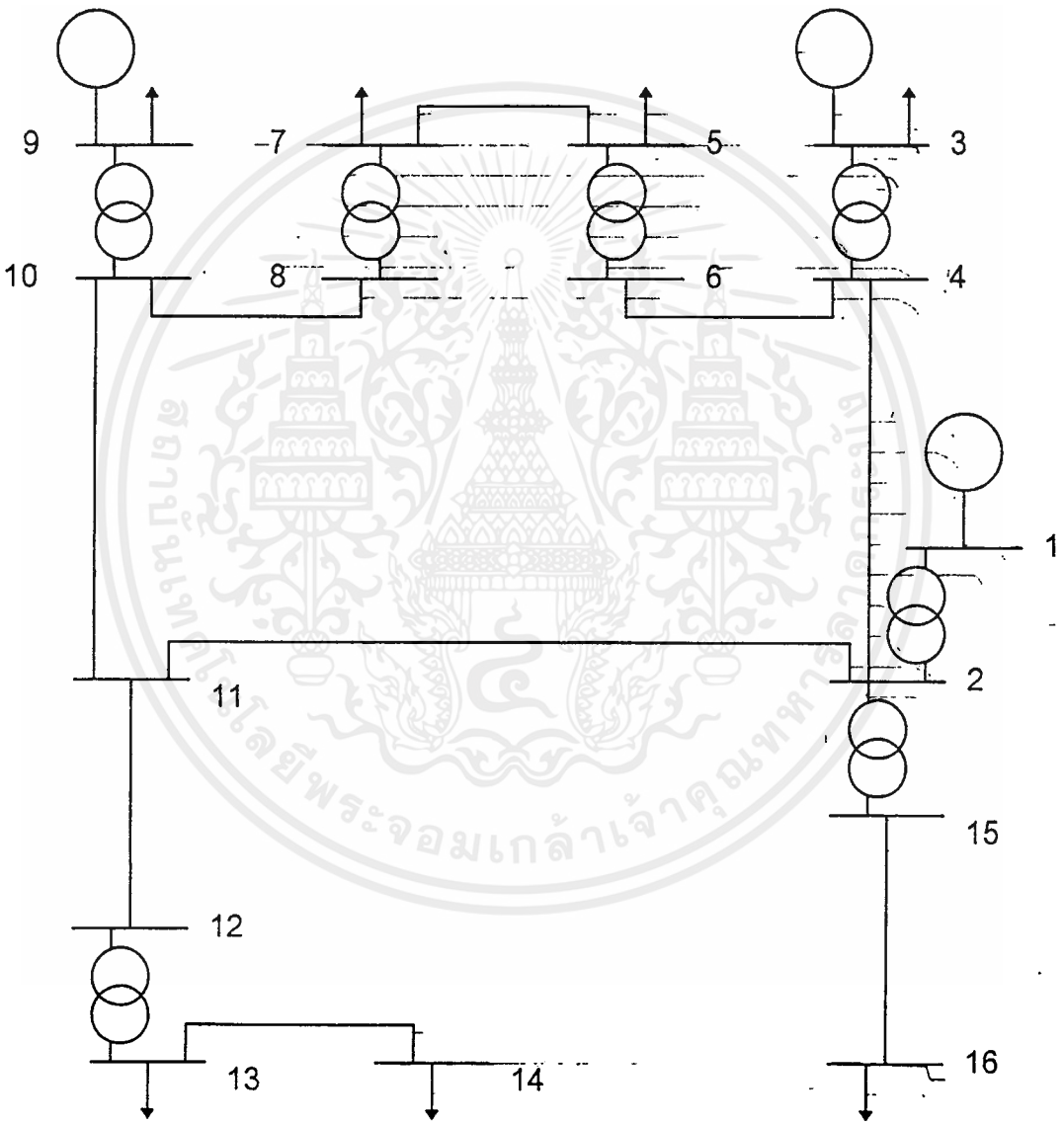
BUS	NAME	TYPE	VOLTAGE	LOAD		GENERATION		MAXIMUM	MINIMUM	SHUNT
				MW	MVAR	MW	MVAR	MVAR	MVAR	SUSCEPTANCE
1	Lowry_1	1	1	0	0	0	0	0	0	0
2	Lowry_2	0	1	0	0	0	0	0	0	0
3	Russell_3	1	1.05	10	55	110	0	0	0	0
4	Grigsby_4	0	1	0	0	0	0	0	0	0
5	N.sub_5	0	1	75	15	0	0	0	0	0
6	Grigsby_6	0	1	0	0	0	0	0	0	0
7	S.sub_7	0	1	90	20	0	0	0	0	0
8	Grigsby_8	0	1	0	0	0	0	0	0	0
9	Rogers_9	1	1.05	15	4	220	0	0	0	0
10	Grigsby_10	0	1	0	0	0	0	0	0	0
11	Irwin_11	0	1	0	0	0	0	0	0	0
12	Phillip_12	0	1	0	0	0	0	0	0	0
13	Phillip_13	0	1	50	2	0	0	0	0	0
14	Hornell_14	0	1	35	3	0	0	0	0	0
15	Lowry_15	0	1	0	0	0	0	0	0	0
16	Feaster_16	0	1	15	20	0	0	0	0	0

LINES DATA

BUS	NAME	BUS	NAME	R	X	B
4	Grigsby_4	6	Grigsby_6	0.00665	0.03519	0.07458
8	Grigsby_8	10	Grigsby_10	0.00665	0.03519	0.07458
10	Grigsby_10	11	Irwin_11	0.00998	0.05279	0.1119
2	Lowry_2	4	Grigsby_4	0.01664	0.08798	0.18644
2	Lowry_2	11	Irwin_11	0.01664	0.08798	0.18644
5	N.sub_5	7	S.sub_7	0.008302	0.04555	0.008129
15	Lowry_15	16	Feaster_16	0.02768	0.1518	0.0271
11	Irwin_11	12	Phillip_12	0.006656	0.035192	0.074576
13	Phillip_13	14	Honnell_14	0.0521	0.1773	0.003707

TRANSFORMERS DATA

BUS	NAME	BUS	NAME	R	X	TAP	CODE
1	Lowry_1	2	Lowry_2	0.0035	0.0035	1	0
15	Lowry_15	2	Lowry_2	0.002722	0.03267	1	0
12	Phillip_12	13	Phillip_13	0.002083	0.04167	1.025	0
4	Grigsby_4	3	Russell_3	0.003846	0.03846	1	0
6	Grigsby_6	5	N.sub_5	0.001667	0.04167	1	0
8	Grigsby_8	7	S.sub_7	0.001667	0.04167	1	0
10	Grigsby_10	9	Rogers_9	0.0012	0.024	1	0



Single line diagram of "gamma" system.

Reference.

1. B.R Gupta ,"Power System Analysis and Design" , Wheeler Publishing , New Delhi , India , 1993.
2. Charles A. Gross ,"Power System Analysis" , John Wiley & Sons , Singapore , 1986.
3. J. ARRILLAGA and C.P. ARNOLD ,"Computer Modelling of Electrical Power Systems" , John Wiley & Sons , UK , 1984.
4. J.-P. Barret ,P.Bornard and B.Meyer ,"Power System Simulation" , Chapman & Hall , London , UK , 1997.
5. John. J. Grainger and William D. Stevenson, Jr.,"Power System Analysis", McGraw-Hill, Singapore, 1994.
6. R N DHAR ,"Computer Aided Power System Operation and Analysis" ,TATA McGraw-Hill , New Delhi , India ,1984.
7. Stagg ,G.W and El-Abiad , A.H. ,"Computer Methods in Power System Analysis" , McGraw-Hill , Tokyo , Japan , 1968.
8. Toran Gone ,"Modern Power System Analysis" , John Wiley & Sons , New York , 1987.
9. William D. Stevenson, Jr. ,"Elements of Power System Analysis" , McGraw-Hill , Singapore ,1982.

กิตติกรรมประกาศ

กลุ่มผู้จัดทำ

ขอขอบพระคุณ ท่านอาจารย์ สมโภชน์ ประไพ ที่ให้คำปรึกษา แนะนำ แก้ปัญหา จนกระทั่งปริญญานิพนธ์นี้สำเร็จลุล่วงด้วยดี

ขอขอบพระคุณ คณะอาจารย์ภาควิชาไฟฟ้ากำลัง ที่ให้คำแนะนำติชม เพื่อการปรับปรุงแก้ไขในจุดบกพร่องต่าง ๆ จนกระทั่งปริญญานิพนธ์นี้สำเร็จลุล่วงด้วยดี

ขอขอบพระคุณ คุณพ่อ คุณแม่ ที่ เป็นกำลังใจ ให้อดทนในการแก้ปัญหาต่างๆ จนผ่านพ้นไปได้ด้วยดีมาตลอด

ขอขอบคุณ เพื่อนๆ ภาควิชาไฟฟ้ากำลัง ทุกคน ที่ร่วมเรียน ร่วมเล่น และช่วยผ่อนคลายความเครียดจากการทำปริญญานิพนธ์นี้

ท้ายนี้ ขอขอบคุณ "เธอ"(หลายๆคน) ผู้เป็นแรงบันดาลใจ