รายการอ้างอิง

1. IEEE Committee Report. Voltage Stability of Power System : Analytical Tools and Industry Experience. IEEE PES publication, 90TH0358-2-PWR
2. CIGRE Task Force 38.02.10 Report. Modeling of Voltage Stability including Dynamic Phenomena. CIGRE 1993
3. P. Kundur. Power System Stability and Control. Mc-Graw Hill,New York (EPRI Power System Engineering. Series),1994
4. C.W. Taylor. Power System Voltage Stability. Mc-Graw Hill,New York (EPRI Power System Engineering. Series),1994
5. IEEE Task Force Report. Load Representation for Dynamic Performance Analysis. IEEE Trans. PWRS-8,No. 2,pp. 472 - 482, May 1993
6. IEEE Task Force Report. Bibliography on Load Models for Power Flow and Dynamic Performance Simulation. IEEE Trans. PWRS-10,No. 1,pp. 523 - 538, February 1995
7. IEEE Task Force Report. Standard Load Models for Power Flow and Dynamic Performance Simulation. IEEE Trans. PWRS-10,No. 3,pp. 1302 - 1313, August 1995
8. S.C. Srivastava. Power System Dynamics , Control and Operation. เอกสารประกอบการอบรมที่จัดโดยสถาบันเทคโนโลยีแห่งเอเชีย (AIT),1996
9. B. Gao,G.K. Morison,P. Kundur. Voltage Stability Evaluation using Modal Analysis. IEEE Trans. PWRS-7,No. 4,pp. 1529 - 1542, November 1992
10. B. Gao,G.K. Morison,P. Kundur. Voltage Stability Analysis using Static and Dynamic Approaches. IEEE Trans. PWRS-8,No. 3,pp. 1159 - 1171, August 1993

11. V. Ajjarapu,C. Christy. The Continuation Power Flow : A tool for Steady State Voltage Stability Analysis. IEEE Trans. PWRS-7,No. 1,pp. 416 - 423, February 1992

12. สมนึก ชยพรกุล. Development of Load Models in Electrical Power System. วิทยานิพนธ์ระดับมหาบัณฑิต ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย 2532

13. Mo-Shing Chen. Modeling and Analysis of Modern Power Systems. เอกสารประกอบการอบรมที่จัดโดย สถาบันจัดการด้านพลังงาน (PMI) ,1996

14. G. W. Stagg, Ahmed H. Ei-Abiad. Computer Methods in Power Systems Analysis. Mc-Graw Hill, International Editions, 1968

15. W. W. Price, K. A. Wirgau, A. Murdoch, J. V. Mitsche. Load Modeling for Power Flow and Transient Stability Computer Studies. IEEE Trans. on PWRS-3, No. 1, pp. 180-187, February 1988

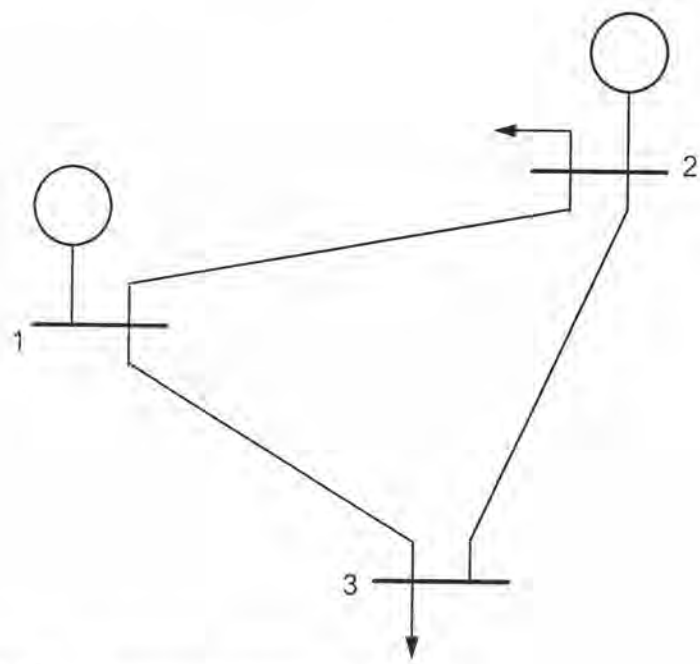
16 P. M. Anderson, A. A. Fouad. Power System Control and Stability. IEEE Press, Revised Printing, 1994

ภาคผนวก

ภาคผนวก ก

ผลของแบบจำลองโหลดที่มีต่อระบบทดสอบ 3 บัส

ระบบทดสอบ 3 บัสที่ใช้ทดสอบนี้ จะเป็นระบบเดียวกับในบทที่ 5 ข้อมูลต่างๆ ของระบบสามารถแสดงได้ดังนี้ โดยแรงดันไฟฟ้าของระบบขนาด 24 kV และกำลังไฟฟ้าฐาน (Base kVA) ของระบบขนาด 1,000 kVA โดยที่ระบบนี้มีความถี่อยู่ที่ 60 Hz



รูปที่ ก. 1 ระบบทดสอบขนาด 3 บัส

ตารางที่ ก.1 ตารางแสดงข้อมูลของสายส่งของระบบทดสอบ

จากบัส - ถึงบัส	ค่าอิมพิแดนซ์ z_{pq}	ค่าไลน์ชาร์จจิง $y'_{pq}/2$
1 - 2	0.080 + j0.240	0.0 + j0.000
1 - 3	0.020 + j0.060	0.0 + j0.000
2 - 3	0.060 + j0.180	0.0 + j0.000

ตารางที่ ก.2 ตารางแสดงแรงดันและกำลังไฟฟ้าของแต่ละบัส

บัส	แรงดันไฟฟ้า	กำลังไฟฟ้าที่ผลิต		กำลังไฟฟ้าโหลด	
		kW	kVAr	kW	kVAr
1	1.0500 + j0.0000	913.7	240.7	0.0	0.0
2	1.0287 - j0.0512	200.0	250.5	500.0	200.0
3	1.0241 - j0.0348	0.0	0.0	600.0	250.0

ตารางที่ ก.3 ตารางแสดงข้อมูลของเครื่องกำเนิดไฟฟ้า

เครื่องกำเนิดไฟฟ้า	ค่าคงที่ความเฉื่อย H	ความต้านทาน r_a	ค่ารีแอคแตนซ์ x'_d
1	30.0	0.04	0.24
2	12.0	0.02	0.52

ระบบ 3 บัสนี้จะมีโหลดอยู่ที่บัส 2 และ บัส 3 ในการทดสอบจะใช้แบบจำลองโหลดเช่นเดียวกับในบทที่ 5 คือ แบบจำลองโหลดที่ 1 , 2 , 4.2 , 4.4 , 5.1 แต่จะทดลองเปลี่ยนแบบจำลองโหลดสลับกันระหว่างบัส 2 และ บัส 3 โดยจะพิจารณาเสถียรภาพที่บัส 3 เป็นหลัก เหตุการณ์รบกวนจะใช้เหตุการณ์ที่ 1 ของระบบ 3 บัส โดยสามารถแสดงผลได้ดังนี้

เหตุการณ์ที่ 1 เวลา 0.0 วินาที ที่บัส 2 เกิดฟอลต์ลงดินผ่านอิมพิแดนซ์ 0.0 - j70.0

0.06 วินาที ที่บัส 2 กำจัดฟอลต์ออกไป

ผลการคำนวณสามารถแสดงได้ดังตาราง

แบบจำลองโหลด	1	2	4.2	4.4	5.1
1	O	O	O	X	X
2	O	O	O	X	X
4.2	O	O	O	X	X
4.4	X	X	X	X	X
5.1	X	X	X	X	X

จากผลที่ได้ทดลองตัดโหลดที่บัส 2 ออกไป 300 kW เช่นเดียวกับเหตุการณ์รบกวนที่ 2 ในระบบ 3 บัส โดยจะทดลองเฉพาะกรณีที่แบบจำลองโหลดที่ทำให้ระบบไม่มีเสถียรภาพเท่านั้น กรณีที่ระบบมีเสถียรภาพจะไม่ทำการคำนวณ

เหตุการณ์ที่ 2 เวลา 0.0 วินาที ที่บัส 2 เกิดฟอลต์ลงดินผ่านอิมพิแดนซ์ 0.0 - j70.0
0.06 วินาที ที่บัส 2 กำจัดฟอลต์ออกไป
0.06 วินาที ที่บัส 2 โหลดถูกตัดออกไป 300 kW

ผลการคำนวณสามารถแสดงได้ดังตาราง

แบบจำลองโหลด	1	2	4.2	4.4	5.1
1	-	-	-	O	X
2	-	-	-	O	X
4.2	-	-	-	O	X
4.4	X	X	X	X	X
5.1	X	X	X	X	X

ต่อมาทดลองเปลี่ยนจุดเกิดฟอลต์เป็นบัส 1 แทน โดยเหตุการณ์ที่เกิดขึ้นจะยังคงเหมือนเดิม สามารถแสดงรายละเอียดได้ดังนี้

เหตุการณ์ที่ 3 เวลา 0.0 วินาที ที่บัส 1 เกิดฟอลต์ลงดินผ่านอิมพิแดนซ์ 0.0 - j70.0
0.06 วินาที ที่บัส 1 กำจัดฟอลต์ออกไป

ผลการคำนวณสามารถแสดงได้ดังตาราง

แบบจำลองโหลด	1	2	4.2	4.4	5.1
1	O	O	X	X	X
2	O	O	X	X	X
4.2	O	O	X	X	X
4.4	X	X	X	X	X
5.1	X	X	X	X	X

หมายเหตุ " O " หมายถึง ระบบมีเสถียรภาพ
" X " หมายถึง ระบบไม่มีเสถียรภาพ
" - " หมายถึง ไม่ได้ทำการคำนวณ เนื่องจากระบบมีเสถียรภาพ

ภาคผนวก ข

คู่มือการใช้งานโปรแกรมคอมพิวเตอร์

โปรแกรมคอมพิวเตอร์ที่ใช้ในการคำนวณเสถียรภาพของแรงดันไฟฟ้าถูกพัฒนาขึ้นด้วยภาษาซี (C Language) โดยใช้คอมไพล์เลอร์ของ Turbo C ver. 2.0 โปรแกรมต้นฉบับ (Source Code) จะอยู่ในไฟล์ "VOLTAGE.C" หลังจากคอมไพล์แล้วจะได้โปรแกรมใช้งานที่มีชื่อว่า "VOLTAGE.EXE" ประมวลผลในระบบปฏิบัติการณ์ดอส โดยตัวโปรแกรมจะแบ่งการทำงานออกเป็น 2 ส่วน คือ

1. ส่วนคำนวณโหลดโฟลว์
2. ส่วนคำนวณเสถียรภาพของแรงดันไฟฟ้า

ตัวโปรแกรมจะสามารถใช้คำนวณโหลดโฟลว์แต่เพียงอย่างเดียว หรือใช้คำนวณเสถียรภาพแรงดันไฟฟ้าก็ได้ โดยในการทำงานจะแยกออกเป็น 2 ส่วน ซึ่งแต่ละส่วนสามารถอธิบายการใช้งานโปรแกรมได้ดังนี้

การใช้งานโปรแกรมในส่วนโหลดโฟลว์

1. หลังจากเรียกโปรแกรมแล้วบนหน้าจอ ภายในกรอบสีเหลืองจะปรากฏข้อความ

Please enter number (1 or 2) :- __

คำอธิบายในกรอบสี่เหลี่ยมด้านซ้ายจะให้ผู้ใช้เลือกว่า จะคำนวณโหดลโฟลว์หรือคำนวณเสถียรภาพของแรงดันไฟฟ้า ให้เลือกข้อ 1 เพื่อคำนวณเฉพาะโหลดโฟลว์

2. ภายหลังจากเลือกคำนวณโหลดโฟลว์แล้ว บรรทัดต่อมาภายในกรอบสีเหลืองจะถาม

File name :- _____

ให้ใส่ชื่อและไดเร็กทอรี่ของไฟล์ข้อมูลของระบบที่จะคำนวณ ถ้าไฟล์ข้อมูลนั้นอยู่ในไดเร็กทอรี่เดียวกันกับตัวโปรแกรมก็ให้ใส่แต่ชื่อไฟล์ข้อมูลเพียงอย่างเดียว โดยรายละเอียดของข้อมูลและวิธีการใส่ของไฟล์ข้อมูลจะอธิบายไว้ส่วนท้ายของภาคผนวกนี้

3. บรรทัดต่อมาจะปรากฏข้อความ

```
Tolerance Error :- _____
```

ให้ป้อนค่าความคลาดเคลื่อน (Tolerance) ในการคำนวณที่ต้องการ แต่ถ้าไม่ใส่ค่าความคลาดเคลื่อนโปรแกรมจะตั้งค่านี้ (Default value) ไว้ที่ 0.00001 ในการคำนวณ

4. หลังจากป้อนค่าความคลาดเคลื่อนแล้ว รอสักครู่โปรแกรมจะแสดงผลการคำนวณออกมาทั้งแรงดันไฟฟ้าที่บัสและกำลังไฟฟ้าที่ผลิตได้ของแต่ละบัส โดยแรงดันไฟฟ้าจะแสดงในรูปของขนาดของแรงดันไฟฟ้าและมุมของแรงดัน ส่วนกำลังไฟฟ้าที่ผลิตได้จะแสดงในรูปของกำลังไฟฟ้าจริงและกำลังไฟฟ้าเสมือนของแต่ละบัส

5. หลังจากกดปุ่มใดๆ บนคีย์บอร์ดโปรแกรมจะจบการทำงาน

การใช้งานโปรแกรมในส่วนคำนวณเสถียรภาพของแรงดันไฟฟ้า

1. หลังจากเรียกโปรแกรมใช้งานแล้ววิธีการจะเหมือนในการคำนวณโหลดโฟลว์ โดยในคำถามแรกให้ใส่หมายเลข 2 แทน จากนั้นใส่ชื่อไฟล์ข้อมูลของระบบและใส่ค่าความคลาดเคลื่อนในการคำนวณซึ่งสามารถแสดงได้ดังนี้

```
Please enter number (1 or 2) :- _
File name :- _____
Tolerance Error :- _____
```

2. บรรทัดต่อมาโปรแกรมจะให้ใส่ค่าความถี่ที่ใช้ในการคำนวณสามารถแสดงได้ดังนี้

```
Frequency :- ___
```

ถ้าไม่ป้อนความถี่ไว้ โปรแกรมจะใช้ค่าความถี่ที่ 60 Hz

3. รอสักครู่โปรแกรมจะแสดงผลการคำนวณแรงดันไฟฟ้าและกำลังไฟฟ้าออกมา กดปุ่มใดๆ บนคีย์บอร์ดโปรแกรมจะถาม

```
Maximum Time (s) :- _____
```

ให้ใส่ช่วงเวลาที่จะคำนวณ โดยเวลาสูงสุดที่จะใส่มีหน่วยเป็นวินาที

4. บรรทัดต่อมาโปรแกรมจะถามว่า

```
Time step (s) :- _____
```

ให้ใส่ความละเอียดของช่วงเวลา (Step size) ที่ใช้คำนวณ ให้ใส่เป็นหน่วยวินาที เช่น 0.01 เป็นต้น

5. ต่อมาโปรแกรมจะถามว่า

Accurate number :- _____

ให้ใส่จำนวนรอบในการคำนวณ ถ้าใส่ค่ามากๆ จะทำให้ได้ความละเอียดมากยิ่งขึ้น

6. โปรแกรมจะถาม

Tolerance Error :- _____

ให้ผู้ใช้ใส่ค่าความคลาดเคลื่อนใหม่อีกครั้ง ถ้าไม่ป้อนค่าใหม่โปรแกรมจะใช้ค่าเดิมที่ป้อนไว้แล้ว

7. บรรทัดต่อมาจะแสดง

Acceleration Factor :- _____

ให้ป้อนค่าตัวเร่งที่ใช้คำนวณ ค่าที่ป้อนจะอยู่ในช่วงประมาณ 1.2 - 1.5 โดยโปรแกรมจะตั้งค่า (Default value) ไว้ที่ 1.4

8. บรรทัดสุดท้ายโปรแกรมจะถาม

Select method (1 or 2) :- _____

ให้ผู้ใช้เลือกระหว่างวิธีนิวตัน-ราฟสันหรือวิธีของเกาส์ไซเดิล ในการคำนวณส่วนโหลดโฟลว์ของการวิเคราะห์เสถียรภาพของแรงดันไฟฟ้า

9. ต่อมาโปรแกรมจะถาม

Number of Events :- _____

ให้ป้อนจำนวนครั้งของเหตุการณ์รบกวนที่เกิดขึ้น

10. โปรแกรมจะให้ป้อนเวลาที่เกิดเหตุการณ์รบกวนขึ้นดังนี้ (ป้อนเป็นหน่วยวินาที)

Time (s) :- _____

11. โปรแกรมจะถามชนิดของเหตุการณ์รบกวนที่เกิดขึ้น โดยแบ่งออกเป็น

1) เกิดฟอลต์ขึ้นโดยแรงดันที่บัสนั้นเป็น 0 2) กำจัดฟอลต์ออกไป

3) เกิดฟอลต์ลงดินผ่านค่าแอดมิตแตนซ์ 4) กำจัดฟอลต์ผ่านแอดมิตแตนซ์ออกไป

5) ตัดสายส่งในระบบออกไป 6) สายส่งถูกต่อเข้าระบบ

7) โหลดที่บัสถูกตัดออกไป

12. โปรแกรมจะถามบัสที่เกิดเหตุการณ์รบกวน ให้ป้อนบัสที่เกิดเหตุการณ์รบกวนเข้าไป

13. ถ้าเลือกเหตุการณ์รบกวนแบบที่ 3 และ แบบที่ 7 โปรแกรมจะให้ผู้ใช้ป้อนค่าดังนี้

เหตุการณ์รบกวนแบบที่ 3 ผู้ใช้จะต้องป้อนค่าแอดมิตแตนซ์ของฟอลต์ที่เกิดเข้าไป โดยอยู่ในรูปของ $Y = G + jB$

เหตุการณ์รบกวนแบบที่ 7 ผู้ใช้จะต้องป้อนกำลังไฟฟ้าของบัสนั้นๆ ที่เหลืออยู่ หลังจากถูกตัดโหลด (Load Shedding) ออกไปแล้ว

14. หลังป้อนค่าของเหตุการณ์รบกวนที่เกิดขึ้นครบหมดแล้ว ให้กดปุ่มใดๆ บนคีย์บอร์ด โปรแกรมจะเริ่มทำการคำนวณโดยจะแสดงเป็นเปอร์เซนต์ เมื่อคำนวณเสร็จแล้วก็จะครบ 100 % จากนั้นโปรแกรมจะให้ป้อนชื่อไฟล์เพื่อใช้เก็บผลการคำนวณทั้งหมดซึ่งสามารถใช้โปรแกรมแก้ไขเอกสาร (Text Editor) ทั่วไปดูผลการคำนวณได้ ในไฟล์นี้จะแสดงผลการคำนวณของ แรงดันไฟฟ้า , กำลังไฟฟ้าของโหลด , กำลังไฟฟ้าที่ผลิตได้ , ความเร็วในการหมุนของเครื่องกำเนิดไฟฟ้าและมุมโรเตอร์ของเครื่องกำเนิดไฟฟ้า เป็นต้น

15. หลังจากเก็บผลการคำนวณลงไฟล์แล้ว โปรแกรมจะจบการทำงาน

ตัวอย่างไฟล์ข้อมูลของระบบในการคำนวณ

ข้อมูลของระบบที่ใช้ในการคำนวณจะถูกเก็บไว้ในไฟล์ข้อมูล โดยผู้ใช้สามารถใช้โปรแกรมแก้ไขเอกสาร (Text Editor) ทั่วไป ในการแก้ไขข้อมูลเหล่านี้ได้ ในไฟล์ข้อมูลจะแบ่งออกเป็น 2 ส่วน คือ ส่วนแรกจะแสดงข้อมูลของสายส่ง , หม้อแปลง ส่วนที่สองจะเป็นข้อมูลของบัส สามารถแสดงตัวอย่างของไฟล์ข้อมูลได้ดังนี้

```
2     1     4     0.0000  0.0576                          1.000     0.0
1     4     5     0.0100  0.0850     0.0   0.0880
1     4     6     0.0170  0.0920     0.0   0.0790
1     5     7     0.0320  0.1610     0.0   0.1530
1     6     9     0.0390  0.1700     0.0   0.1790
2     2     7     0.0000  0.0625                          1.000     0.0
1     7     8     0.0085  0.0720     0.0   0.0745
2     3     9     0.0000  0.0586                          1.000     0.0
1     8     9     0.0119  0.1008     0.0   0.1045
*
1   1.04   0.0                 0.00  0.00                  10   23.64   0.0   0.0608
3   1.0    0.0    1.63  0.067  0.00  0.00                  11    6.40   0.0   0.1198
3   1.0    0.0    0.85 -0.109  0.00  0.00                  12    3.01   0.0   0.1813
3   1.0    0.0    0.00  0.000  0.00  0.00
3   1.0    0.0    0.00  0.000  1.25  0.50   1   2   1.0
3   1.0    0.0    0.00  0.000  0.90  0.30   1   2   1.0
3   1.0    0.0    0.00  0.000  0.00  0.00
3   1.0    0.0    0.00  0.000  1.00  0.35   1   2   1.0
3   1.0    0.0    0.00  0.000  0.00  0.00
```

รูป ข.1 ตัวอย่างไฟล์ข้อมูลของระบบ 9 บัส

1. ข้อมูลของสายส่ง , หม้อแปลง และชั้นท์อิลิเมนต์ ในส่วนแรกจะสามารถแสดงข้อมูลของแต่ละคอลัมภ์ได้ดังนี้

1.1 ชนิดของอุปกรณ์ (Element type) จะมีอยู่ 3 ประเภทดังนี้

- สายส่งจะใช้หมายเลข 1
- หม้อแปลงไฟฟ้าจะใช้หมายเลข 2
- ชั้นท์อิลิเมนต์จะใช้หมายเลข 3

1.2 ตำแหน่งของอุปกรณ์ จะเป็นการเชื่อมต่อจากบัสหนึ่งไปยังอีกบัสหนึ่ง ตัวอย่างเช่น อุปกรณ์ตัวแรกในรูป ข. 1 จะเป็น หม้อแปลงไฟฟ้าซึ่งต่อจากบัส 1 ไปยังบัส 4 เป็นต้น

1.3 ค่าอิมพิแดนซ์ของอุปกรณ์ ประกอบด้วยค่าความต้านทาน (R) และค่ารีแอกแตนซ์ (X) หน่วยเป็นเปอร์ยูนิต โดยอยู่ในรูปของ R + jX ถ้าอุปกรณ์เป็นชั้นท์อิลิเมนต์ค่านี้จะถูกเว้นว่างไว้

1.4 ค่าไลน์ชาร์จจิ้งของอุปกรณ์ หน่วยเป็นเปอร์ยูนิต (p.u.) โดยอยู่ในรูปของ G + jB สำหรับหม้อแปลงจะไม่มีค่าไลน์ชาร์จจิ้งนี้โดยจะเว้นช่องว่างไว้

1.5 อัตราส่วนแท็ปและการเลื่อนเฟส เป็นข้อมูลเฉพาะของหม้อแปลงเท่านั้น โดยอัตราส่วนจะมีหน่วยเป็นเปอร์ยูนิต (p.u.) ส่วนการเลื่อนเฟสจะมีหน่วยเป็นองศา

1.6 ตัวอักษร "*" เป็นตัวอักษรซึ่งบอกให้โปรแกรมรู้ว่าสิ้นสุดข้อมูลของอุปกรณ์แล้วบรรทัดต่อไปจะเป็นข้อมูลของบัส

2. ข้อมูลของบัสจะมีข้อมูลอยู่ในคอลัมภ์ต่างๆ โดยในแต่ละคอลัมภ์จะอธิบายได้ดังนี้

2.1 ชนิดของบัส (Bus Type) จะถูกกำหนดด้วยเลขรหัสดังนี้

- บัสอ้างอิง (Slack Bus) จะใช้หมายเลข 1
- บัสควบคุมแรงดัน (Voltage Controlled Bus) จะใช้หมายเลข 2
- บัสโหลด (Load Bus) จะใช้หมายเลข 3

2.2 ขนาดของแรงดันและมุมของแรงดันไฟฟ้า โดยขนาดของแรงดันไฟฟ้าจะมีหน่วยเปอร์ยูนิต ส่วนมุมจะมีหน่วยเป็นองศา ถ้าบัสนั้นเป็นโหลดบัสขนาดและมุมของแรงดันไฟฟ้าจะไม่ทราบค่าแต่จะสมมติให้มีค่า 1.0 และ 0.0 ตามลำดับ

2.3 กำลังไฟฟ้าที่ผลิตจะแบ่งออกเป็นกำลังจริงและกำลังเสมือน

- กำลังจริง ถ้าเป็นบัสอ้างอิงจะเว้นช่องว่างไว้ ส่วนบัสประเภทอื่นจะใส่ค่าตามที่กำหนดไว้ในหน่วยเปอร์ยูนิต
- กำลังเสมือน ถ้าเป็นบัสโหลดให้ใส่ค่าในหน่วยเปอร์ยูนิต (p.u.) แต่บัสประเภทอื่นให้เว้นช่องว่างไว้

2.4 โหลดให้ป้อนค่ากำลังไฟฟ้าของโหลดในหน่วยเปอร์ยูนิตตามที่กำหนด

2.5 ขีดจำกัดในการจ่ายกำลังไฟฟ้าเสมือน ให้ใส่ค่าของขีดจำกัดสูงสุดก่อนและขีดจำกัดต่ำสุดตาม โดยให้ใส่เป็นหน่วยเปอร์ยูนิต ซึ่งค่านี้จะใช้กับบัสควบคุมแรงดันเท่านั้น

* หมายเหตุ เนื่องจากในตัวอย่างนี้ไม่มีบัสควบคุมแรงดัน จึงให้เว้นช่องว่างไว้แทน

2.6 จำนวนชนิดของแบบจำลองโหลด ให้ป้อนจำนวนของแบบจำลองโหลดของบัสนั้น

* หมายเหตุ ถ้าบัสนั้นๆ ไม่มีโหลดให้เว้นช่องว่างไว้ไม่ต้องป้อนข้อมูล (รวมทั้งข้อ 2.7 , 2.8)

2.7 ชนิดของแบบจำลองโหลด แบบจำลองโหลดที่ใช้สามารถแสดงได้ดังตาราง ข.1 ดังนี้

ตาราง ข. 1 แสดงตัวอย่างของแบบจำลองโหลด

อุปกรณ์ไฟฟ้า	P_v	P_f	Q_v	Q_f
0. แบบกำลังไฟฟ้าคงที่	0.0	0.0	0.0	0.0
1. แบบกระแสคงที่	1.0	0.0	1.0	0.0
2. แบบอิมพิแดนซ์คงที่	2.0	0.0	2.0	0.0
3. โหลดที่กำหนดค่าเอง	-	-	-	-
4. เครื่องทำน้ำอุ่น	2.0	0.0	0.0	0.0
5. เครื่องล้างจาน	1.8	0.0	3.5	-1.4
6. เครื่องซักผ้า	0.08	2.9	1.6	1.8
7. หลอดไฟแบบไส้	1.54	0.0	0.0	0.0
8. หลอดฟลูออเรสเซนต์	1.0	1.0	3.0	-2.8
9. ตู้เย็น	0.8	0.5	2.5	-1.4
10. มอเตอร์ขนาดเล็ก	0.1	2.9	0.6	-1.8
11. มอเตอร์ขนาดใหญ่	0.05	1.9	0.5	1.2
12. เครื่องปรับอากาศแบบรวม	0.2	0.9	2.2	-2.7
13. เครื่องปรับอากาศ	0.5	0.6	2.5	-2.8
14. เตาหลอม	2.3	-1.0	1.61	-1.0
15. ปั๊มการเกษตร	1.4	5.6	1.4	4.2
16. เครื่องทำความร้อน	2.0	0.0	0.0	0.0
17. ปั๊มเครื่องทำความร้อน	0.2	0.9	2.5	-1.3
18. เครื่องอบผ้า	2.0	0.0	3.3	-2.6
19. โทรทัศน์สี	2.0	0.0	5.2	-4.6
20. พัดลมของเตาหลอม	0.08	2.9	1.6	1.8
21. ปั๊มเครื่องทำความร้อนอุตสาหกรรม	0.1	1.0	2.5	-1.3
22. แอร์อุตสาหกรรมแบบรวม	0.1	1.0	2.5	-1.3
23. แอร์คอนดิชั่นอุตสาหกรรม	0.5	0.6	2.5	-2.8

2.8 เปอร์เซ็นต์ของแบบจำลองโหลด ให้ผู้ใช้ใส่ค่าของเปอร์เซ็นต์ของแบบจำลองโหลดนั้นๆ จาก 100 % โดยจะต้องป้อนเป็นเลขทศนิยม เช่น 20 % ให้ป้อนเป็น 0.2 เป็นต้น

* หมายเหตุ แบบจำลองโหลดที่ 3 จะให้ผู้ใช้กำหนดค่าของ P_v , P_f , Q_v , Q_f เอง โดยผู้ใช้จะต้องใส่ค่าเหล่านี้ต่อท้ายเปอร์เซ็นต์ของแบบจำลองโหลด เช่น โหลดชนิดที่ 3 100 % จะต้องป้อน 3 1.0 1.25 1.83 1.25 1.83 เป็นต้น

* หมายเหตุ ถ้าแบบจำลองโหลดมีหลายชนิดรวมกันให้ป้อนดังนี้ เช่น แบบจำลองโหลดที่ 0 65% , แบบจำลองโหลดที่ 1 25% , แบบจำลองโหลดที่ 2 10% ให้ป้อน ดังนี้ 3 0 0.65 1 0.25 2 0.10 เป็นต้น (3 ตัวแรกหมายถึง แบบจำลองโหลด 3 ชนิด

2.9 หมายเลขบัสของเครื่องกำเนิดไฟฟ้า เป็นตัวเลขที่บอกให้โปรแกรมรู้ว่า ตัวเลขนั้นใช้แทนบัสของเครื่องกำเนิดไฟฟ้าในการคำนวณ เช่น เครื่องกำเนิดไฟฟ้าที่ 1 ของระบบทดสอบ 9 บัส ในรูป ข. 1 จะมีค่านี้เป็น 10 เนื่องจากเป็นบัสที่นับต่อจากบัสเดิมของระบบ

2.10 ค่าคงที่ความเฉื่อยของเครื่องกำเนิดไฟฟ้า (H)

2.11 ค่าความต้านทานของเครื่องกำเนิดไฟฟ้า (r_a)

2.12 ค่าทรานเซียนส์รีแอกแตนซ์ (x_d')

* หมายเหตุ ข้อ 2.9 ถึง 2.12 ถ้าบัสนั้นไม่มีเครื่องกำเนิดไฟฟ้าต่ออยู่ให้เว้นว่างไว้

ภาคผนวก ค

โปรแกรมคอมพิวเตอร์

```
/*************************************************************
**                                                          **
** Thesis           : Impact of Dynamic load Models         **
**                    on Voltage Stability                  **
** Description      : Program to calculate Voltage Stability**
**                    in the Power System                   **
** Programmer       : Prawit Paravanich                     **
** Compiler         : Turbo C  ver. 2.0                     **
** Revistion        : 1.0                                   **
** Date             : 31 May  1997                          **
**                                                          **
*************************************************************/

#include <conio.h>
#include <alloc.h>
#include <math.h>
#include "\source\screen1.c"

enum  ELEMENT   { lines = 1 , trafo , capa , ultc , reac };
enum  BUSTYPE   { SL_bus = 1 , PV_bus , PQ_bus };
enum  EVTYPE    { F_occur = 1 , F_clear , Li_chrg_in , Li_chrg_out , Ele_open , Ele_close , Load_chng };

typedef struct {
          double   re;
          double   im;
         } RECTAN;
typedef struct {
          int     row;
          int     col;
          RECTAN  *m;
         } MATRIX;
typedef struct {
          int     row;
          int     col;
          double  *m;
         } MATREAL;
struct el_data {
          enum ELEMENT   el_type;
          int            operate;
          int            from;
          int            to;
          RECTAN         impd;
          RECTAN         li_chrg;
          double         ratio;
          double         shift;
          struct el_data *next;
         };
struct iteration {
          RECTAN           *b;
          struct iteration *next;
         };
struct load_data { int      load_type;
          double     quantity;
          double     Pv;
          double     Pf;
          double     Qv;
```

```
                double     Qf;
               };
struct induc_data { double     Rs;
                double     Xs;
                double     Rr;
                double     Xr;
                double     Xm;
                double     slip;
                double     X;
                double     Xdat;
                double     To;
               };
struct bus_data  {
                enum BUSTYPE  bus_type;
                RECTAN        volt;
                RECTAN        gen;
                RECTAN        load;
                RECTAN        qlimit;
               };
struct mac_data  {
                enum BUSTYPE      bus_type;
                RECTAN            volt;
                RECTAN            gen;
                RECTAN            load;
                RECTAN            qlimit;
                int               numload;
                struct induc_data *induc;
                struct load_data  *lod;
                int               mac_bus;
                double            inertia;
                RECTAN            mac_impd;
               };
struct ev_data  {
                enum EVTYPE   ev_type;
                double        time;
                int           operate;
                int           from;
                int           to;
                RECTAN        li_chrg;
               };
struct lo_table  {
                char       *Name;
                int        PV;
                int        PF;
                int        QV;
                int        QF;
               };

float          Success;                 /* NR_GS = Newton Raphson or Gauss Seidel */
int            pivot,evt_do,NR_GS,LF_TS;    /* LF_TS = Load Flow  or  Transient */
FILE           *f_output;
int            num_bus,num_ele,iterations,num_SL,num_MAC,num_LOAD,num_INDUC;
int            SIZE_REC,SIZE_MAT,SIZE_ITE,SIZE_REAL;
int            num_STEP,num_EVENT,frequency,*machine,loop_acc,*induction;
int            t_run,acc_run,Mac,trend;
double         tolerance,max_mismatch,accelerate,Step_t,T_max;
MATREAL        *Pm,*W_real,*d_real,*Freq;
MATREAL        *Pm_temp,*Pe_temp,*W_temp,*d_temp,*dW_temp,*dd_temp;
MATRIX         *Voltbus,*PQload,*PQgen,*Load_temp,*Edat,*E_temp,*dE_temp;
MATRIX         *Y_bus,*init_PQ,*init_V;
MATREAL        *Jacob,*Inv_Jacob,*Ja_p,*Inv_p;
struct el_data   *el_ph,*el_pe,*el_load;
struct bus_data  *bus_ph;
struct mac_data  *mac_ph;
struct ev_data   *event;
struct iteration *V_ph,*V_pe,*I_ph,*I_pe,*P_ph,*P_pe;
struct iteration *DPQ_ph,*DPQ_pe,*DE_ph,*DE_pe;
struct lo_table  Table[] = { {"Constant Power",0,0,0,0 },          {"Constant Current",100,0,100,0 },
                  {"Constant Impedance",200,0,200,0 },     {"Customize Load",0,0,0,0 },
```

```
{"Water Heater",200,0,0,0 },              {"Dish Washer",180,0,350,-140 },
{"Clothes Washer",8,290,160,180 },        {"Incandescent Lighting",154,0,0,0 },
{"Fluorescent Lighting",100,100,300,-280 }, {"Refrigerator",80,50,250,-140 },
{"Small Industrial Motor",10,290,60,-180 }, {"Large Industrial Motor",5,190,50,120 },
{"Central Air Conditioner",20,90,220,-270 }, {"Room Air Conditioner",50,60,250,-280 },
{"Arc Furnace",230,-100,161,-100 },       {"Argriculture Pumps",140,560,140,420 },
{"Resistance Space Heater",200,0,0,0 },   {"Heat Pump Space Heating",20,90,250,-130 },
{"Clothes Dryer",200,0,330,-260 },        {"Color Television",200,0,520,-460 },
{"Furnace Fan",8,290,160,180 },           {"Commercial Heat Pump",10,100,250,-130 },
{"Commercial Central A/C",10,100,250,-130 }, {"Commercial Room A/C",50,60,250,-280 },
};

void   Clear(RECTAN *p,int num);
struct iteration *Make_ITE(int bu);
MATRIX  *Make_MAT(int r,int c);
MATREAL *Make_REAL(int r,int c);
void   Free_ITE(struct iteration *aa);
void   Free_MAT(MATRIX *abc);
void   Free_REAL(MATREAL *abc);
void   Clear_ITE(struct iteration *aa,int bu);
void   Transpose(MATRIX *p1,MATRIX *p2);
RECTAN  Conjugate(RECTAN a);
void   Multiply(MATREAL *p1,struct iteration *p2,struct iteration *p3,int n);
void   Rect_operation(char oper,RECTAN a,RECTAN b,RECTAN *c);
void   Convert_REC_POL(RECTAN *a,RECTAN *b,int method);
void   Inverse_Real(MATREAL *v,MATREAL *z);
void   Save_MAT(MATRIX *sha,int method);
void   Save_REAL(MATREAL *sha);
void   Show_MAT(MATRIX *sha,int posx,int posy,int method);
void   Show_REAL(MATREAL *sha,int posx,int posy);
void   Show_ITE(struct iteration *shb,int posx,int posy,int method);
void   Initial_Program(void);
void   Input_data(char *ff);
void   Form_Y_bus(void);
void   Calculate_Jacobian(void);
void   Prepare_for_GS(void);
void   Calculate_GS(void);
void   Prepare_for_NR(void);
void   Calculate_NR(void);
int    Check_event(double time_run);
void   Represent_LOAD(void);
void   Prepare_for_TR(void);
void   Calculate_Euler(void);

/*****************************************************************************/
void Clear(p,num)
 RECTAN *p;
 int num;
{  auto int i;

  for(i = 0 ; i != num ; i++)
   { p[i].re = (double)0.0;
     p[i].im = (double)0.0;
   }
}
/*****************************************************************************/
struct iteration *Make_ITE(bu)
 int bu;
{  auto struct iteration *ab;

  ab   = (struct iteration *)malloc(SIZE_ITE);
  ab->b = (RECTAN *)malloc(bu * SIZE_REC);
  ab->next = NULL;
  Clear(ab->b,bu);
  return ab;
}
/*****************************************************************************/
MATRIX *Make_MAT(r,c)
 int r,c;
```

```
{  auto MATRIX *abc;
   abc = (MATRIX *)malloc(SIZE_MAT);
   abc->row = r;
   abc->col = c;
   abc->m   = (RECTAN *)malloc(r * c * SIZE_REC);
   Clear(abc->m,r * c);
   return abc;
}
/*****************************************************************************/
MATREAL *Make_REAL(r,c)
 int r,c;
{  auto MATREAL *abc;
   auto int     i;
   abc = (MATREAL *)malloc(SIZE_REAL);
   abc->row = r;
   abc->col = c;
   abc->m   = (double *)malloc(r * c * sizeof(double));
   for(i = 0 ; i != r * c ; i++)
    abc->m[i] = (double)0.0;
   return abc;
}
/*****************************************************************************/
void Free_ITE(aa)
 struct iteration *aa;
{ auto struct iteration *ab;

  ab = aa->next;
  while(aa != NULL)
   { free(aa->b);
     free(aa);
     aa = ab;
     if(ab != NULL)
       ab = ab->next;
   }
}
/*****************************************************************************/
void Free_MAT(abc)
 MATRIX *abc;
{
 free(abc->m);
 free(abc);
}
/*****************************************************************************/
void Free_REAL(abc)
 MATREAL *abc;
{
 free(abc->m);
 free(abc);
}
/*****************************************************************************/
void Clear_ITE(aa,bu)
 struct iteration *aa;
 int              bu;
{
 while(aa != NULL)
  { Clear(aa->b,bu);
    aa = aa->next;
  }
}
/*****************************************************************************/
void Transpose(p1,p2)
 MATRIX *p1,*p2;
{  auto int x,y;

   for(x = 0 ; x != p1->row ; x++)
    for(y = 0 ; y != p1->col ; y++)
     { p2->m[y*p2->col + x].re = p1->m[x*p1->col + y].re;
       p2->m[y*p2->col + x].im = p1->m[x*p1->col + y].im; }
}
```

```
/*****************************************************************************/
RECTAN Conjugate(a)
 RECTAN a;
{  if(a.im != (double)0.0)    a.im *= -1;
   return (a);
}
/*****************************************************************************/
void Multiply(p1,p2,p3,n)
 MATREAL *p1;
 struct iteration *p2,*p3;
 int n;
{  auto int x,z;

   for(x = 0 ; x != n ; x++)
    if(bus_ph[x].bus_type != SL_bus)
     { for(z = 0 ; z != n ; z++)
        if(bus_ph[z].bus_type != SL_bus)
         { p3->b[x].re += p1->m[x*p1->col + z] * p2->b[z].re
                  +  p1->m[x*p1->col + z + n] * p2->b[z].im;
           p3->b[x].im += p1->m[(x+n)*p1->col + z] * p2->b[z].re
                  +  p1->m[(x+n)*p1->col + z +n] * p2->b[z].im;
         }
     }
}
/*****************************************************************************/
void Rect_operation(oper,a,b,c)
 char oper;
 RECTAN a,b,*c;
{  switch (oper)
     { case '+' : c->re = a.re + b.re;
                  c->im = a.im + b.im;
                  break;
       case '-' : c->re = a.re - b.re;
                  c->im = a.im - b.im;
                  break;
       case '*' : c->re = a.re*b.re - a.im*b.im;
                  c->im = a.re*b.im + a.im*b.re;
                  break;
       case '/' : c->re = (a.re*b.re + a.im*b.im)/(b.re*b.re + b.im*b.im);
                  c->im = (a.im*b.re - a.re*b.im)/(b.re*b.re + b.im*b.im);
                  break;
     }
}
/*****************************************************************************/
void    Convert_REC_POL(a,b,method)
 RECTAN  *a,*b;                          /* method = 1 REC => POL */
 int     method;                        /* method = 0 POL => REC */
{
 if(method)
   { b->re = (double)sqrt(a->re * a->re + a->im * a->im);
     if(a->re != 0.0)
      { b->im = 180/M_PI * (double)atan(a->im / a->re);
        if(a->re < 0.0)
         { if(a->im > 0.0)   b->im = b->im + 180.0;
           else          b->im = b->im - 180.0;
         }
      }
     else
      { if(a->im == 0.0)
          b->im = (double)0.0;
        else
          if(a->im > 0.0)    b->im = (double)90.0;
            else           b->im = (double)-90.0;
      }
   }
 else
  { a->re = b->re * (double)cos(b->im * M_PI/180);
    a->im = b->re * (double)sin(b->im * M_PI/180);
  }
```

```
}
/*****************************************************************************/
void Inverse_Real(v,z)
MATREAL *v,*z;
{ MATREAL *t;
  double   r,s;
  int      i,j,k;

  t = Make_REAL(v->row,2*v->row);

  /************** Set Initial Value ***************/
  for( i = 0 ; i != t->row ; i++)
   { for( j = 0 ; j != t->row ; j++)
      t->m[i*t->col + j] = v->m[i*v->col + j];
     t->m[i*t->col + i + t->row] = (double)1.0;
   }
  /************** Calculate Inverse ***************/
  for(i = 0 ; i != t->row ; i++)
   { if(t->m[i*t->col + i] == (double)0.0)
      { pivot++;
       j = i + 1;
       while((t->m[j*t->col+i] == (double)0.0)&&(j != t->row))
        j++;
       if(j == t->row)
        printf(" -: ERROR! CAN'T FIND MATRIX :- "); /********* error *********/
       for(k = 0 ; k != t->col ; k ++)
       { r = t->m[i*t->col + k];
         t->m[i*t->col + k] = t->m[j*t->col + k];
         t->m[j*t->col + k] = r;
       }
      }
     for(j = t->col - 1 ; j != i - 1 ; j--)
      t->m[i*t->col + j] /= t->m[i*t->col + i];
     for(k = 0 ; k != t->row ; k++)
      if(k != i)
       { r = t->m[k*t->col + i] / t->m[i*t->col + i];
         for(j = 0 ; j != t->col ; j++)
          { s = r * t->m[i*t->col + j];
            t->m[k*t->col + j] -= s;
          }
       }
   }
  for(i = 0 ; i != t->row ; i++)
   for(j = 0 ; j != t->row ; j++)
    z->m[i*t->row +j] = t->m[i*t->col + j + t->row];
  free(t->m);
}
/*****************************************************************************/
void  Save_MAT(sha,method)
MATRIX *sha;
int method;
{ auto int i,j;
  for( i = 0 ; i != sha->row ; i++)
  { for( j = 0 ; j != sha->col ; j++)
    { if(!(method % 2))
       fprintf(f_output," % 11.7lf ",sha->m[i*sha->col + j].re);
      if(method)
       fprintf(f_output,"%+ 11.7lf  ",sha->m[i*sha->col + j].im);
    }
   fprintf(f_output,"\n");
  }
  fprintf(f_output,"\n\n");
}
/*****************************************************************************/
void  Show_MAT(sha,posx,posy,method)
MATRIX *sha;
int   posx,posy,method;
{ auto int i,j;
  for( i = 0 ; i != sha->row ; i++)          /* method 0 :- Real Only  */
```

```
    { goto_xy(posx,posy + i);                /* method 1 :- Imag Only   */
      for( j = 0 ; j != sha->col ; j++)      /* method 2 :- Real + Imag */
      { if(!(method % 2))
          printf("% 10.6lf",sha->m[i*sha->col + j].re);
        if(method)
          printf("%+ 10.6lfi ",sha->m[i*sha->col + j].im);
      }
      write_screen(1,posy + i,3,78,"□");
    }
}
/*****************************************************************************/
void  Save_REAL(sha)
 MATREAL *sha;
{  auto int i,j;
   for( i = 0 ; i != sha->row ; i++)
    { for( j = 0 ; j != sha->col ; j++)
        fprintf(f_output," % 11.7lf ",sha->m[i*sha->col + j]);
      fprintf(f_output,"\n");
    }
   fprintf(f_output,"\n\n");
}
/*****************************************************************************/
void  Show_REAL(sha,posx,posy)
 MATREAL *sha;
 int   posx,posy;
{  auto int i,j;
   for( i = 0 ; i != sha->row ; i++)
    { goto_xy(posx,posy + i);
      for( j = 0 ; j != sha->col ; j++)
        printf("% 7.3lf",sha->m[i*sha->col + j]);
      write_screen(posx,posy + i,3,51,"□");
    }
}
/*****************************************************************************/
void  Show_ITE(shb,posx,posy,method)
 struct iteration *shb;
 int   posx,posy,method;
{  auto int i,j;
   for(i = 0 ; shb != NULL ; shb = shb->next,i++ )  /* method 0 :- Real Only   */
    { goto_xy(posx,posy + i);                       /* method 1 :- Imag Only   */
      for( j = 0 ; j != num_bus ; j++)              /* method 2 :- Real + Imag */
      { if(!(method % 2))
          printf("% 10.6lf",shb->b[j].re);
        if(method)
          printf("%+ 10.6lfi ",shb->b[j].im);
      }
      write_screen(posx,posy + i,3,51,"□");
    }
}
/*****************************************************************************/
void   Initial_Program(void)
{ auto char  *fl;
  auto int   *g;
  auto double *d;

  num_bus = num_ele = num_SL = num_MAC = num_LOAD = num_INDUC = (int)0;
  tolerance  = (double)0.00001;  accelerate = (double)1.4;
  LF_TS = NR_GS = (int)1;   frequency = (int)60;   evt_do = pivot = (int)0;
  Success = (float)0.0;
  loop_acc = (int)2;
  T_max = Step_t = (double)0.0;
  SIZE_REC  = sizeof(RECTAN);
  SIZE_MAT  = sizeof(MATRIX);
  SIZE_REAL = sizeof(MATREAL);
  SIZE_ITE  = sizeof(struct iteration);

  cls(0,0,79,24,BLUE * 16 + LIGHTCYAN);
  draw_a_frame(0,0,79,16,2,BLUE * 16 + MAGENTA);
  draw_a_frame(0,17,42,24,2,BLUE * 16 + MAGENTA);
```

```
 draw_a_frame(43,17,79,24,2,BLUE * 16 + YELLOW);
 cls(1,18,41,23,BLUE * 16 + LIGHTGRAY);
 write_scr_box(2,18,35,20,2,"Please select program to calculate"
                    "  1. Load Flow Program          "
                    "  2. Transient Stability Program  ");
 write_screen(45,18,2,31,"Please enter number (1 or 2) :-");
 write_screen(77,18,3,1,"Þ");                    /* YELLOW On L_CYAN */
 g = (int *)getnumber(77,18);  if(g != NULL) LF_TS = *g;
 write_screen(77,18,3,1,"-");                    /* YELLOW On BLUE */
 write_scr_box(2,18,37,20,2,"Please enter Data File of the system"
                    "                              "
                    "                              ");
 write_screen(45,19,2,11,"Filename :-");
 write_screen(57,19,3,20,"Þ");                    /* YELLOW On L_CYAN */
 fl = getstring(30,57,19);
 write_screen(57,19,3,20,"-");                    /* YELLOW On BLUE */
 write_scr_box(2,18,37,19,2,"Specific Tolerance Error          "
                    "  (Default value = 0.00001)      ");
 write_screen(45,20,2,24,"Tolerance Error     :-");
 write_screen(70,20,3,7,"Þ");                    /* YELLOW On L_CYAN */
 d = (double *)getnumber(70,20);  if(d != NULL) tolerance = *d;  else write_screen(70,20,2,7,"0.00001");
 write_screen(70,20,3,7,"-");                    /* YELLOW On BLUE */
 if(LF_TS == 2)
  { write_scr_box(2,18,35,20,2,"Please frequency of the system   "
                    "  (Default value = 60)          "
                    "                                ");
    write_screen(45,21,2,24,"Frequency           :-");
    write_screen(70,21,3,3,"Þ");                    /* YELLOW On L_CYAN */
    g = (int *)getnumber(70,21);    if(g != NULL) frequency = *g; else write_screen(70,21,2,2,"60");
    write_screen(70,21,3,3,"-");                    /* YELLOW On BLUE */
  }
 Input_data(fl);
}
/*****************************************************************************/
void Input_data(ff)
char *ff;
{ auto int  yn,i,j;
 auto char  junk;
 FILE     *fpp;
 auto int  ti;
 auto double tf;

 el_ph = (struct el_data *)malloc(sizeof(struct el_data));
 el_pe = el_ph;
 el_pe->operate   = (int)1;
 el_pe->impd.re   = (double)0.0;     el_pe->impd.im    = (double)0.0;
 el_pe->li_chrg.re = (double)0.0;    el_pe->li_chrg.im = (double)0.0;
 el_pe->ratio     = (double)0.0;     el_pe->shift      = (double)0.0;

 fpp = fopen(ff,"r");
 junk = fgetc(fpp);
 do
 {
  switch (junk)
   { case ';' : while((junk = fgetc(fpp)) != '\n');
             break;
     case ' ' : fscanf(fpp,"%d",&ti);  el_pe->el_type = ti;
             fscanf(fpp,"%d",&ti);  if(ti > num_bus) num_bus = ti;
              el_pe->from = --ti;
             fscanf(fpp,"%d",&ti);  if(ti > num_bus) num_bus = ti;
              el_pe->to = --ti;
             switch(el_pe->el_type)
              { case lines :
                case trafo :
                case ultc  : fscanf(fpp,"%lf",&tf); el_pe->impd.re = tf;
                          fscanf(fpp,"%lf",&tf); el_pe->impd.im = tf;
                          if(el_pe->el_type != lines) break;
                case capa  :
                case reac  : fscanf(fpp,"%lf",&tf); el_pe->li_chrg.re = tf;
```

```
                    fscanf(fpp,"%lf",&tf); el_pe->li_chrg.im = tf;
                    break;
                }
              if(el_pe->el_type == trafo)
                { fscanf(fpp,"%lf",&tf); el_pe->ratio = tf;
                  fscanf(fpp,"%lf",&tf); el_pe->shift = tf;
                }
              num_ele++;
              while((junk = fgetc(fpp)) != '\n');
              break;
    }
   junk = fgetc(fpp);
   if(junk == '.')
    { el_pe->next = (struct el_data *)malloc(sizeof(struct el_data));
     el_pe = el_pe->next;
     el_pe->operate   = (int)1;
     el_pe->impd.re   = (double)0.0;    el_pe->impd.im   = (double)0.0;
     el_pe->li_chrg.re = (double)0.0;   el_pe->li_chrg.im = (double)0.0;
     el_pe->ratio     = (double)0.0;    el_pe->shift     = (double)0.0;
    }
  }
 while(junk != '*');
 while((junk = fgetc(fpp)) != '\n');
 el_pe->next = NULL;

 mac_ph = (struct mac_data *)malloc(num_bus * sizeof(struct mac_data));
 for(i = 0 ; i != num_bus ; i++)
  { mac_ph[i].gen.re     = (double)0.0;   mac_ph[i].gen.im = (double)0.0;
   mac_ph[i].load.re     = (double)0.0;   mac_ph[i].load.im = (double)0.0;
   mac_ph[i].qlimit.re   = (double)0.0;   mac_ph[i].qlimit.im = (double)0.0;
   mac_ph[i].numload     = (int)0;
   mac_ph[i].induc       = NULL;          mac_ph[i].lod   = NULL;
   mac_ph[i].mac_bus     = (int)-1;       mac_ph[i].inertia    = (double)0.0;
   mac_ph[i].mac_impd.re = (double)0.0;   mac_ph[i].mac_impd.im = (double)0.0;
   fscanf(fpp,"%d",&ti); mac_ph[i].bus_type = ti;
   switch (mac_ph[i].bus_type)
   { case SL_bus : num_SL++;
              fscanf(fpp,"%lf",&tf); mac_ph[i].volt.re = tf;
              fscanf(fpp,"%lf",&tf); mac_ph[i].volt.im = tf;
              fscanf(fpp,"%lf",&tf); mac_ph[i].load.re = tf;
              fscanf(fpp,"%lf",&tf); mac_ph[i].load.im = tf;
              break;
     case PV_bus : fscanf(fpp,"%lf",&tf); mac_ph[i].volt.re = tf;
              fscanf(fpp,"%lf",&tf); mac_ph[i].volt.im = tf;
              fscanf(fpp,"%lf",&tf); mac_ph[i].gen.re = tf;
              /*                mac_ph[i].gen.im = 0.0;*/
              fscanf(fpp,"%lf",&tf); mac_ph[i].load.re = tf;
              fscanf(fpp,"%lf",&tf); mac_ph[i].load.im = tf;
              fscanf(fpp,"%lf",&tf); mac_ph[i].qlimit.re = tf;
              fscanf(fpp,"%lf",&tf); mac_ph[i].qlimit.im = tf;
              break;
     case PQ_bus : fscanf(fpp,"%lf",&tf); mac_ph[i].volt.re = tf;
              fscanf(fpp,"%lf",&tf); mac_ph[i].volt.im = tf;
              fscanf(fpp,"%lf",&tf); mac_ph[i].gen.re = tf;
              fscanf(fpp,"%lf",&tf); mac_ph[i].gen.im = tf;
              fscanf(fpp,"%lf",&tf); mac_ph[i].load.re = tf;
              fscanf(fpp,"%lf",&tf); mac_ph[i].load.im = tf;
              break;
   }
   if(LF_TS == 1)    goto LoadFlow;
   if((mac_ph[i].load.re != (double)0.0)||(mac_ph[i].load.im != (double)0.0))
    { auto double a,b,c,d,e,r,s;
     num_LOAD++;
     fscanf(fpp,"%d",&ti);    mac_ph[i].numload = ti;
     if(mac_ph[i].numload == -1)
      { mac_ph[i].induc = (struct induc_data *)malloc(sizeof(struct induc_data));
       mac_ph[i].lod   = NULL;
       num_INDUC++;
       fscanf(fpp,"%lf",&tf);   mac_ph[i].induc->Rs = tf;
```

```
            fscanf(fpp,"%lf",&tf);   mac_ph[i].induc->Xs = tf;
            fscanf(fpp,"%lf",&tf);   mac_ph[i].induc->Rr = tf;
            fscanf(fpp,"%lf",&tf);   mac_ph[i].induc->Xr = tf;
            fscanf(fpp,"%lf",&tf);   mac_ph[i].induc->Xm = tf;
            fscanf(fpp,"%lf",&tf);   mac_ph[i].induc->slip = tf;

            mac_ph[i].induc->X = mac_ph[i].induc->Xs + mac_ph[i].induc->Xm;
            r = mac_ph[i].induc->Xm * mac_ph[i].induc->Xr;
            s = mac_ph[i].induc->Xm + mac_ph[i].induc->Xr;
            mac_ph[i].induc->Xdat = mac_ph[i].induc->Xs + (r/s);
            r = 2 * M_PI * frequency * mac_ph[i].induc->Rr;
            mac_ph[i].induc->To = s/r;
           }
          else
           { mac_ph[i].induc = NULL;
            mac_ph[i].lod = (struct load_data *)malloc((ti+1) * sizeof(struct load_data));
            for(j = 1 ; j <= mac_ph[i].numload ; j++)
             { fscanf(fpp,"%d",&ti);  mac_ph[i].lod[j].load_type = ti;
              fscanf(fpp,"%lf",&tf);  mac_ph[i].lod[j].quantity  = tf;
              if(mac_ph[i].lod[j].load_type == 3)
               { fscanf(fpp,"%lf",&tf);   mac_ph[i].lod[j].Pv = tf;
                fscanf(fpp,"%lf",&tf);   mac_ph[i].lod[j].Pf = tf;
                fscanf(fpp,"%lf",&tf);   mac_ph[i].lod[j].Qv = tf;
                fscanf(fpp,"%lf",&tf);   mac_ph[i].lod[j].Qf = tf;
               }
              else
               { mac_ph[i].lod[j].Pv = (double)Table[mac_ph[i].lod[j].load_type].PV / 100.0;
                mac_ph[i].lod[j].Pf = (double)Table[mac_ph[i].lod[j].load_type].PF / 100.0;
                mac_ph[i].lod[j].Qv = (double)Table[mac_ph[i].lod[j].load_type].QV / 100.0;
                mac_ph[i].lod[j].Qf = (double)Table[mac_ph[i].lod[j].load_type].QF / 100.0;
               }
             }
            mac_ph[i].lod[0].load_type = 3;  /* Composite Load */
            if((mac_ph[i].numload == 1) && (mac_ph[i].lod[1].load_type < 3))
             mac_ph[i].lod[0].load_type = mac_ph[i].lod[1].load_type;
            a = b = c = d = e = (double)0.0;
            for(j = 1 ; j <= mac_ph[i].numload ; j++)
             { a += mac_ph[i].lod[j].quantity * mac_ph[i].lod[j].Pv;
              b += mac_ph[i].lod[j].quantity * mac_ph[i].lod[j].Pf;
              c += mac_ph[i].lod[j].quantity * mac_ph[i].lod[j].Qv;
              d += mac_ph[i].lod[j].quantity * mac_ph[i].lod[j].Qf;
              e += mac_ph[i].lod[j].quantity;
             }
            mac_ph[i].lod[0].Pv = a;     mac_ph[i].lod[0].Pf = b;
            mac_ph[i].lod[0].Qv = c;     mac_ph[i].lod[0].Qf = d;
            mac_ph[i].lod[0].quantity = e;
           }
      }
    if((mac_ph[i].bus_type != PQ_bus)||(mac_ph[i].gen.re)||(mac_ph[i].gen.im))
     { num_MAC++;
      fscanf(fpp,"%d",&ti);   mac_ph[i].mac_bus    = --ti;
      fscanf(fpp,"%lf",&tf);   mac_ph[i].inertia    = tf;
      fscanf(fpp,"%lf",&tf);   mac_ph[i].mac_impd.re = tf;
      fscanf(fpp,"%lf",&tf);   mac_ph[i].mac_impd.im = tf;
     }
    LoadFlow:                    /*  Calculate only Load Flow  */
    while((junk = fgetc(fpp)) != '\n');
  }
fclose(fpp);
bus_ph = (struct bus_data *)malloc(num_bus * sizeof(struct bus_data));
for(i = 0 ; i != num_bus ; i++)
 { bus_ph[i].bus_type  =  mac_ph[i].bus_type;
  bus_ph[i].volt.re  =  mac_ph[i].volt.re;
  bus_ph[i].volt.im  =  mac_ph[i].volt.im;
  bus_ph[i].gen.re   =  mac_ph[i].gen.re;
  bus_ph[i].gen.im   =  mac_ph[i].gen.im;
  bus_ph[i].load.re  =  mac_ph[i].load.re;
  bus_ph[i].load.im  =  mac_ph[i].load.im;
  bus_ph[i].qlimit.re =  mac_ph[i].qlimit.re;
```

```
    bus_ph[i].qlimit.im =  mac_ph[i].qlimit.im;
   }
  cls(1,1,78,15,BLUE * 16 + LIGHTCYAN);
}
/*****************************************************************************/
void Form_Y_bus(void)
{  auto RECTAN r,s,t,u;

  s.re = 1.0;  s.im = 0.0;
  for(el_pe = el_ph ; el_pe != NULL ; el_pe = el_pe->next)
  { if(el_pe->operate == 1)
     switch (el_pe->el_type)
      { case lines : Rect_operation('/',s,el_pe->impd,&r);
                 Y_bus->m[el_pe->from*Y_bus->col + el_pe->from].re
                   += r.re + el_pe->li_chrg.re;
                 Y_bus->m[el_pe->from*Y_bus->col + el_pe->from].im
                   += r.im + el_pe->li_chrg.im;
                 Y_bus->m[el_pe->to*Y_bus->col + el_pe->to].re
                   += r.re + el_pe->li_chrg.re;
                 Y_bus->m[el_pe->to*Y_bus->col + el_pe->to].im
                   += r.im + el_pe->li_chrg.im;
                 Y_bus->m[el_pe->from*Y_bus->col + el_pe->to].re -= r.re;
                 Y_bus->m[el_pe->from*Y_bus->col + el_pe->to].im -= r.im;
                 Y_bus->m[el_pe->to*Y_bus->col + el_pe->from].re -= r.re;
                 Y_bus->m[el_pe->to*Y_bus->col + el_pe->from].im -= r.im;
                 break;
        case trafo : Rect_operation('/',s,el_pe->impd,&r);
                 Y_bus->m[el_pe->from*Y_bus->col + el_pe->from].re
                   += (r.re/(el_pe->ratio * el_pe->ratio));
                 Y_bus->m[el_pe->from*Y_bus->col + el_pe->from].im
                   += (r.im/(el_pe->ratio * el_pe->ratio));
                 Y_bus->m[el_pe->to*Y_bus->col + el_pe->to].re += r.re;
                 Y_bus->m[el_pe->to*Y_bus->col + el_pe->to].im += r.im;
                 t.re = el_pe->ratio * (double)cos(el_pe->shift*M_PI/180);
                 t.im = el_pe->ratio * (double)sin(el_pe->shift*M_PI/180);
                 Rect_operation('/',r,Conjugate(t),&u);
                 Y_bus->m[el_pe->from*Y_bus->col + el_pe->to].re -= u.re;
                 Y_bus->m[el_pe->from*Y_bus->col + el_pe->to].im -= u.im;
                 Rect_operation('/',r,t,&u);
                 Y_bus->m[el_pe->to*Y_bus->col + el_pe->from].re -= u.re;
                 Y_bus->m[el_pe->to*Y_bus->col + el_pe->from].im -= u.im;
                 break;
        case ultc  : break;
        case capa  : Y_bus->m[el_pe->from*Y_bus->col + el_pe->from].re
                   += el_pe->li_chrg.re;
                 Y_bus->m[el_pe->from*Y_bus->col + el_pe->from].im
                   += el_pe->li_chrg.im;
                 break;
      }
  }
}
/*****************************************************************************/
void Calculate_Jacobian(void)
{  auto RECTAN r;
  auto int   i,j;

  for(i = 0 ; i != num_bus ; i++)
   switch (bus_ph[i].bus_type)
    { case SL_bus : break;
      case PQ_bus   : for(j = 0 ; j != num_bus ; j ++)
                 if(bus_ph[j].bus_type != SL_bus)
                 { Rect_operation('*',V_pe->b[i],Conjugate(Y_bus->m[i*Y_bus->col + j]),&r);
  /* Off_dia J1 */      Jacob->m[i*Jacob->col + j]                 = r.re;
  /* Off_dia J2 */      Jacob->m[i*Jacob->col + j + num_bus]       = r.im;
  /* Off_dia J3 */      Jacob->m[(i+num_bus)*Jacob->col + j]       = r.im;
  /* Off_dia J4 */      Jacob->m[(i+num_bus)*Jacob->col + j + num_bus]   = r.re * -1;
                  if(i == j)
  /* Diagonal J1 */     { Jacob->m[i*Jacob->col + j]               += I_pe->b[i].re;
  /* Diagonal J2 */       Jacob->m[i*Jacob->col + j + num_bus]     += I_pe->b[i].im;
```

```
   /* Diagonal J3 */       Jacob->m[(i+num_bus)*Jacob->col + j]         -= I_pe->b[i].im;
   /* Diagonal J4 */       Jacob->m[(i+num_bus)*Jacob->col + j + num_bus] += I_pe->b[i].re;
                         }
                       }
                     break;
      case PV_bus    : for(j = 0 ; j != num_bus ; j ++)
                       if(bus_ph[j].bus_type != SL_bus)
                       { Rect_operation('*',V_pe->b[i],Conjugate(Y_bus->m[i*Y_bus->col + j]),&r);
   /* Off_dia J1 */      Jacob->m[i*Jacob->col + j]                 = r.re;
   /* Off_dia J2 */      Jacob->m[i*Jacob->col + j + num_bus]       = r.im;
   /* Off_dia J5 */      Jacob->m[(i+num_bus)*Jacob->col + j]       = 0.0;
   /* Off_dia J6 */      Jacob->m[(i+num_bus)*Jacob->col + j + num_bus] = 0.0;
                         if(i == j)
   /* Diagonal J1 */     { Jacob->m[i*Jacob->col + j]                 += I_pe->b[i].re;
   /* Diagonal J2 */       Jacob->m[i*Jacob->col + j + num_bus]       += I_pe->b[i].im;
   /* Diagonal J5 */       Jacob->m[(i+num_bus)*Jacob->col + j]       = 2 * V_pe->b[i].re;
   /* Diagonal J6 */       Jacob->m[(i+num_bus)*Jacob->col + j + num_bus] = 2 * V_pe->b[i].im;
                         }
                       }
                     break;
    }
}
/*****************************************************************************/
void Prepare_for_GS(void)
{
 Y_bus    = Make_MAT(num_bus,num_bus);
 init_PQ  = Make_MAT(num_bus,1);
 init_V   = Make_MAT(num_bus,1);
 V_ph     = Make_ITE(num_bus);

 Form_Y_bus();
}
/*****************************************************************************/
void Calculate_GS(void)
{ auto RECTAN  r,s,t;
 auto double  z;
 auto int     i,j,loop_end;

 loop_end  = 1;
 iterations = 0;
 V_pe = V_ph;
 for(i = 0 ; i != num_bus ; i++)
  Convert_REC_POL(&(V_pe->b[i]),&(bus_ph[i].volt),0);
 while(loop_end)
 { max_mismatch = 0.0;
  DPQ_pe = V_pe;
  if(V_pe->next == NULL)   V_pe->next = Make_ITE(num_bus);
  V_pe = V_pe->next;
  for(i = 0 ; i != num_bus ; i++)
   { V_pe->b[i].re = DPQ_pe->b[i].re;
    V_pe->b[i].im = DPQ_pe->b[i].im;
   }
  for(i = 0 ; i != num_bus ; i++)
  if(bus_ph[i].bus_type != SL_bus)
   { s.re = s.im = 0.0;
    for(j = 0 ; j != num_bus ; j++)
     if(j != i)
      { Rect_operation('*',Y_bus->m[i*Y_bus->col + j],V_pe->b[j],&r);
       Rect_operation('+',s,r,&s);
      }
    Rect_operation('-',bus_ph[i].gen,bus_ph[i].load,&r);
    Rect_operation('/',r,V_pe->b[i],&t);
    r.re = t.re - s.re;   r.im = -t.im - s.im;
    Rect_operation('/',r,Y_bus->m[i*Y_bus->col + i],&s);
    Rect_operation('-',s,DPQ_pe->b[i],&t);
    z = (t.re >= 0) ? t.re : t.re * -1;
    if(z >= max_mismatch) max_mismatch = z;
    z = (t.im >= 0) ? t.im : t.im * -1;
    if(z >= max_mismatch) max_mismatch = z;
```

```
        V_pe->b[i].re = DPQ_pe->b[i].re + (accelerate * t.re);
        V_pe->b[i].im = DPQ_pe->b[i].im + (accelerate * t.im);
      }
    iterations++;
    if(tolerance >= max_mismatch)
     { loop_end = 0;   /************ Terminal Program ***********/
      for(i = 0 ; i != num_bus ; i++)
       { s.re = s.im = (double)0.0;
        for(j = 0 ; j != num_bus ; j++)
         { Rect_operation('*',Y_bus->m[i*Y_bus->col + j],V_pe->b[j],&r);
          Rect_operation('+',s,r,&s);
         }
        Rect_operation('*',V_pe->b[i],Conjugate(s),&r);
        init_PQ->m[i].re = r.re + bus_ph[i].load.re;
        init_PQ->m[i].im = r.im + bus_ph[i].load.im;
        Convert_REC_POL(&(V_pe->b[i]),&(init_V->m[i]),1);
       }
     }
   }
}
/*****************************************************************************/
void Prepare_for_NR(void)
{
 Y_bus    = Make_MAT(num_bus,num_bus);
 init_PQ  = Make_MAT(num_bus,1);
 init_V   = Make_MAT(num_bus,1);
 Jacob    = Make_REAL(2 * num_bus,2 * num_bus);
 Inv_Jacob = Make_REAL(2 * num_bus,2 * num_bus);
 Ja_p     = Make_REAL(2 * (num_bus - num_SL),2 * (num_bus - num_SL));
 Inv_p    = Make_REAL(2 * (num_bus - num_SL),2 * (num_bus - num_SL));
 V_ph     = Make_ITE(num_bus);
 I_ph     = Make_ITE(num_bus);
 P_ph     = Make_ITE(num_bus);
 DPQ_ph   = Make_ITE(num_bus);
 DE_ph    = Make_ITE(num_bus);

 Form_Y_bus();
}
/*****************************************************************************/
void Calculate_NR(void)
{ auto double  z;
  auto RECTAN  r;
  auto int     i,j,k,loop_end;
  auto struct iteration *pa,*pb;

  loop_end = 1;
  iterations = 0;
  V_pe  = V_ph;    I_pe  = I_ph;    P_pe  = P_ph;
  DPQ_pe = DPQ_ph;  DE_pe  = DE_ph;

  for(i = 0 ; i != num_bus ; i++)
   Convert_REC_POL(&(V_pe->b[i]),&(bus_ph[i].volt),0);

  while(loop_end)
   { max_mismatch = 0.0;
    for(i = 0 ; i != num_bus ; i++)
     if(bus_ph[i].bus_type != SL_bus)
      { for(j = 0 ; j != num_bus ; j++)
        { Rect_operation('*',Y_bus->m[i*Y_bus->col + j],V_pe->b[j],&r);
         Rect_operation('+',I_pe->b[i],r,&(I_pe->b[i]));
        }
       Rect_operation('*',V_pe->b[i],Conjugate(I_pe->b[i]),&(P_pe->b[i]));
       if(bus_ph[i].bus_type == PV_bus)
        { r.im = bus_ph[i].load.im + P_pe->b[i].im;
         if((r.im > bus_ph[i].qlimit.re) || (r.im < bus_ph[i].qlimit.im))
          { bus_ph[i].gen.im = (r.im > bus_ph[i].qlimit.re)
                    ? bus_ph[i].qlimit.re : bus_ph[i].qlimit.im;
           bus_ph[i].qlimit.re = bus_ph[i].qlimit.im = 0.0;
           bus_ph[i].bus_type  = PQ_bus;
```

```
            Rect_operation('-',bus_ph[i].gen,bus_ph[i].load,&r);
           }
           else
           { P_pe->b[i].im = V_pe->b[i].re * V_pe->b[i].re
                    + V_pe->b[i].im * V_pe->b[i].im;
            r.re = bus_ph[i].gen.re - bus_ph[i].load.re;
            r.im = bus_ph[i].volt.re * bus_ph[i].volt.re;
           }
         }
         else
          Rect_operation('-',bus_ph[i].gen,bus_ph[i].load,&r);
         Rect_operation('-',r,P_pe->b[i],&(DPQ_pe->b[i]));
         z = (DPQ_pe->b[i].re >= 0) ? DPQ_pe->b[i].re : DPQ_pe->b[i].re * -1;
         if(z >= max_mismatch) max_mismatch = z;
         z = (DPQ_pe->b[i].im >= 0) ? DPQ_pe->b[i].im : DPQ_pe->b[i].im * -1;
         if(z >= max_mismatch) max_mismatch = z;
        }
       if(tolerance >= max_mismatch)
        { auto RECTAN s;
         loop_end = 0; /********** Terminal Program **********/
         for(i = 0 ; i != num_bus ; i++)
         { s.re = s.im = (double)0.0;
          for(j = 0 ; j != num_bus ; j++)
          { Rect_operation('*',Y_bus->m[i*Y_bus->col + j],V_pe->b[j],&r);
           Rect_operation('+',s,r,&s);
          }
          Rect_operation('*',V_pe->b[i],Conjugate(s),&(P_pe->b[i]));
         }
         for(i = 0 ; i != num_bus ; i++)
         { Convert_REC_POL(&(V_pe->b[i]),&(init_V->m[i]),1);
          init_PQ->m[i].re = P_pe->b[i].re + bus_ph[i].load.re;
          init_PQ->m[i].im = P_pe->b[i].im + bus_ph[i].load.im;
         }
        }
       else
        { Calculate_Jacobian();
         for(i = 0 , k = 0 ; i != 2*num_bus ; i++)
         if(bus_ph[i % num_bus].bus_type != SL_bus)
          for(j = 0 ; j != 2*num_bus ; j++)
           if(bus_ph[j % num_bus].bus_type != SL_bus)
            { Ja_p->m[k] = Jacob->m[i*Jacob->col + j];
             k++;
            }
         Inverse_Real(Ja_p,Inv_p);
         for(i = 0 , k = 0 ; i != 2*num_bus ; i++)
          if(bus_ph[i % num_bus].bus_type != SL_bus)
           for(j = 0 ; j != 2*num_bus ; j++)
            if(bus_ph[j % num_bus].bus_type != SL_bus)
             { Inv_Jacob->m[i*Inv_Jacob->col + j] = Inv_p->m[k];
              k++;
             }
         Multiply(Inv_Jacob,DPQ_pe,DE_pe,num_bus);
         pa = V_pe; pb = DE_pe;
         if(V_pe->next == NULL)   V_pe->next  = Make_ITE(num_bus);
          V_pe  = V_pe->next;
         if(I_pe->next == NULL)   I_pe->next  = Make_ITE(num_bus);
          I_pe  = I_pe->next;
         if(P_pe->next == NULL)   P_pe->next  = Make_ITE(num_bus);
          P_pe  = P_pe->next;
         if(DPQ_pe->next == NULL)  DPQ_pe->next = Make_ITE(num_bus);
          DPQ_pe = DPQ_pe->next;
         if(DE_pe->next == NULL)  DE_pe->next  = Make_ITE(num_bus);
          DE_pe  = DE_pe->next;
         for(i = 0 ; i != num_bus ; i++)
          Rect_operation('+',pa->b[i],pb->b[i],&(V_pe->b[i]));
         iterations++;
        }
      }
    }
```

```
/*****************************************************************************/
int  Check_event(double time_run)
{  auto int i,j;
  auto struct el_data *pp;

  j = 0;
  for(i = 0 ; i != num_EVENT ; i++)
  if(((time_run - event[i].time) >= 0.0) && (event[i].operate))
   { j = (int)1;
    evt_do++;
    event[i].operate = (int)0;
    switch(event[i].ev_type)
     { case F_occur :
       case F_clear : if(event[i].ev_type == F_occur)
                      { num_SL++;
                        bus_ph[event[i].from].bus_type = SL_bus;
                        bus_ph[event[i].from].volt.re  = (double)0.0;
                        bus_ph[event[i].from].volt.im  = (double)0.0;
                      }
                      else
                      { num_SL--;
                        bus_ph[event[i].from].bus_type = PQ_bus;
                        bus_ph[event[i].from].volt.re  = mac_ph[event[i].from].volt.re;
                        bus_ph[event[i].from].volt.im  = mac_ph[event[i].from].volt.im;
                      }
                      if(NR_GS == 1)
                      { auto int h;
                        Free_REAL(Ja_p);  Free_REAL(Inv_p);
                        Ja_p    = Make_REAL(2 * (num_bus - num_SL),2 * (num_bus - num_SL));
                        Inv_p   = Make_REAL(2 * (num_bus - num_SL),2 * (num_bus - num_SL));
                        for(h = 0 ; h != 4 * num_bus * num_bus ; h++)  /* Clear Jacob & Inv_Jacob */
                        { Jacob->m[h]    = (double)0.0;
                          Inv_Jacob->m[h] = (double)0.0;
                        }
                      }
                     break;
    case Li_chrg_in  :
    case Li_chrg_out : pp = el_ph;
                     while(pp->next != NULL)
                      pp = pp->next;
                     if(event[i].ev_type == Li_chrg_in)
                      { pp->next = (struct el_data *)malloc(sizeof(struct el_data));
                        pp = pp->next;
                        pp->el_type = capa;                pp->operate  = (int)1;
                        pp->from    = event[i].from;       pp->to       = event[i].to;
                        pp->impd.re  = (double)0.0;        pp->impd.im  = (double)0.0;
                        pp->li_chrg.re = event[i].li_chrg.re;  pp->li_chrg.im = event[i].li_chrg.im;
                        pp->ratio    = (double)0.0;        pp->shift    = (double)0.0;
                        pp->next = NULL;
                      }
                      else
                      { auto struct el_data *ee;
                        ee = el_ph;
                        while(ee->next != pp)
                          ee = ee->next;
                        ee->next = NULL;
                        free(pp);
                      }
                     break;
    case Ele_open  :
    case Ele_close : pp = el_ph;
                     while(((event[i].from != pp->from)||(event[i].to != pp->to))
                       &&((event[i].from != pp->to)||(event[i].to != pp->from)))
                      pp = pp->next;
                     if(pp != NULL)
                      pp->operate = (event[i].ev_type == Ele_open) ? (int)0 : (int)1;
                     break;
    case Load_chng : mac_ph[event[i].from].load.re = event[i].li_chrg.re;
                     mac_ph[event[i].from].load.im = event[i].li_chrg.im;
```

```
                    break;
            }
        }
      return (j);
    }
    /*****************************************************************************/
    void Represent_LOAD(void)
    { auto int load_run,i,j;
      auto struct el_data *load_pe;
      auto RECTAN r,s,t,It;

      load_run = j = 0;
      load_pe = el_load;
      for(i = 0 ; i != num_bus - num_MAC ; i++)
       { if((mac_ph[i].load.re != (double)0.0)||(mac_ph[i].load.im != (double)0.0))
         { if((mac_ph[i].numload == -1)&&(mac_ph[i].induc != NULL))
           { Convert_REC_POL(&r,&(init_V->m[i]),0);
             Rect_operation('-',r,Edat->m[t_run*Edat->col + j],&s);
             t.re = mac_ph[i].induc->Rs;
             t.im = mac_ph[i].induc->Xdat;
             Rect_operation('/',s,t,&It);
             Rect_operation('*',r,Conjugate(It),&(Load_temp->m[load_run]));
             j++;
           }
          else
           { switch(mac_ph[i].lod[0].load_type)
             { case 0 : Load_temp->m[load_run].re = mac_ph[i].load.re;    /* Constant Power */
                        Load_temp->m[load_run].im = mac_ph[i].load.im;
                        break;
               case 1 : r.re = init_V->m[i].re / mac_ph[i].volt.re;    /* Constant Current */
                        r.im = init_V->m[i].im - mac_ph[i].volt.im;
                        Convert_REC_POL(&(mac_ph[i].load),&s,1);
                        t.re = r.re * s.re;
                        t.im = r.im + s.im;
                        Convert_REC_POL(&(Load_temp->m[load_run]),&t,0);
                        break;
               case 2 : r.re = init_V->m[i].re / mac_ph[i].volt.re;    /* Constant Impedance */
                        r.im = r.re * r.re;
                        Load_temp->m[load_run].re = mac_ph[i].load.re * r.im;
                        Load_temp->m[load_run].im = mac_ph[i].load.im * r.im;
                        break;
               case 3 : r.re = init_V->m[i].re / mac_ph[i].volt.re;    /* Composite Load */
                        s.re = Freq->m[t_run] / frequency;
                        r.im = (double)pow(r.re,mac_ph[i].lod[0].Pv);
                        s.im = (double)pow(s.re,mac_ph[i].lod[0].Pf);
                        Load_temp->m[load_run].re = mac_ph[i].load.re * r.im * s.im;
                        r.im = (double)pow(r.re,mac_ph[i].lod[0].Qv);
                        s.im = (double)pow(s.re,mac_ph[i].lod[0].Qf);
                        Load_temp->m[load_run].im = mac_ph[i].load.im * r.im * s.im;
                        break;
             }
           }
          if(init_V->m[i].re != (double)0.0)
           { load_pe->li_chrg.re = Load_temp->m[load_run].re / (init_V->m[i].re * init_V->m[i].re);
             load_pe->li_chrg.im = -1 * Load_temp->m[load_run].im / (init_V->m[i].re * init_V->m[i].re);
           }
          load_pe = load_pe->next;
          load_run++;
         }
       }
    }
    /*****************************************************************************/
    void Prepare_for_TR(void)
    { auto int i,j;

      for(i = 0 ; i != num_bus ; i++)          /* Copy Volt,Power to mac_ph */
       { mac_ph[i].volt.re = init_V->m[i].re;
         mac_ph[i].volt.im = init_V->m[i].im;
         if(mac_ph[i].bus_type != PQ_bus)
```

```
    { mac_ph[i].gen.re = init_PQ->m[i].re;
      mac_ph[i].gen.im = init_PQ->m[i].im;
    }
  }

  el_pe = el_ph;
  while(el_pe->next != NULL)
   el_pe = el_pe->next;
  for(i = 0 ; i != num_bus ; i++)
   if(mac_ph[i].mac_bus != -1)
    { el_pe->next = (struct el_data *)malloc(sizeof(struct el_data));
      el_pe = el_pe->next;
      el_pe->el_type = lines;
      el_pe->operate  = (int)1;
      el_pe->from     = i;          el_pe->to       = mac_ph[i].mac_bus;
      el_pe->impd.re  = mac_ph[i].mac_impd.re;
      el_pe->impd.im  = mac_ph[i].mac_impd.im;
      el_pe->li_chrg.re = (double)0.0;  el_pe->li_chrg.im = (double)0.0;
      el_pe->ratio    = (double)0.0;  el_pe->shift    = (double)0.0;
    }
  el_pe->next = NULL;
  el_load   = el_pe;
  for(i = 0 ; i != num_bus ; i++)
   if((mac_ph[i].load.re != (double)0.0)||(mac_ph[i].load.im != (double)0.0))
    { el_pe->next = (struct el_data *)malloc(sizeof(struct el_data));
      el_pe = el_pe->next;
      el_pe->el_type = capa;
      el_pe->operate  = (int)1;
      el_pe->from     = i;          el_pe->to       = (int)-1;
      el_pe->impd.re  = (double)0.0;  el_pe->impd.im  = (double)0.0;
      el_pe->li_chrg.re = mac_ph[i].load.re / (mac_ph[i].volt.re * mac_ph[i].volt.re);
      el_pe->li_chrg.im = -1 * mac_ph[i].load.im / (mac_ph[i].volt.re * mac_ph[i].volt.re);
      el_pe->ratio    = (double)0.0;  el_pe->shift    = (double)0.0;
    }
  el_pe->next = NULL;
  el_load   = el_load->next;

  if(num_INDUC)
   { induction = (int *)malloc(num_INDUC * sizeof(int));
    i = 0;
    for(j = 0 ; j != num_bus ; j++)
     if((mac_ph[j].numload == -1)&&(mac_ph[j].induc != NULL))
      { induction[i] = j;
        i++;
      }
   }

  free(bus_ph);
  bus_ph = (struct bus_data *)malloc((num_bus+num_MAC) * sizeof(struct bus_data));
  machine = (int *)malloc(num_MAC * sizeof(int));
  for(i = 0 ; i != num_bus ; i++)
   { bus_ph[i].bus_type = PQ_bus;
    bus_ph[i].volt.re  = mac_ph[i].volt.re;
    bus_ph[i].volt.im  = mac_ph[i].volt.im;
    bus_ph[i].gen.re   = (double)0.0;  bus_ph[i].gen.im   = (double)0.0;
    bus_ph[i].load.re  = (double)0.0;  bus_ph[i].load.im  = (double)0.0;
    bus_ph[i].qlimit.re = (double)0.0;  bus_ph[i].qlimit.im = (double)0.0;
  }
  for(j = 0 ; j != num_bus ; j++)
   if(mac_ph[j].mac_bus != -1)
    { auto RECTAN r,s,t,It;
     Convert_REC_POL(&r,&(mac_ph[j].volt),0);
     Rect_operation('/',mac_ph[j].gen,r,&It);
     Rect_operation('*',mac_ph[j].mac_impd,Conjugate(It),&s);
     Rect_operation('+',r,s,&t);

     bus_ph[i].bus_type = SL_bus;
     Convert_REC_POL(&t,&(bus_ph[i].volt),1);
     bus_ph[i].gen.re   = (double)0.0;  bus_ph[i].gen.im   = (double)0.0;
```

```
    bus_ph[i].load.re  = (double)0.0;   bus_ph[i].load.im  = (double)0.0;
    bus_ph[i].qlimit.re = (double)0.0;   bus_ph[i].qlimit.im = (double)0.0;
    machine[i - num_bus] = j;
    i++;
   }
  num_bus += num_MAC;
  num_SL  = num_MAC;

  Free_MAT(Y_bus);   Free_MAT(init_PQ);     Free_MAT(init_V);
  Free_ITE(V_ph);

  if(NR_GS == 1)
   { Free_REAL(Jacob);   Free_REAL(Inv_Jacob);   Free_REAL(Ja_p);
     Free_REAL(Inv_p);
     Free_ITE(I_ph);     Free_ITE(P_ph);
     Free_ITE(DPQ_ph);   Free_ITE(DE_ph);
   }
}
/*****************************************************************************/
void Calculate_Euler(void)
{ auto double *d;
  auto int *g,i;
  auto RECTAN r,s,t,It;


  write_scr_box(2,18,33,18,2,"Please enter Maximum Time (sec.)");
  write_screen(45,18,2,25,"Maximum time (s)     :-");
  write_screen(71,18,3,5,"Þ");                    /* YELLOW On L_CYAN */
  d = (double *)getnumber(71,18);  if(d != NULL) T_max = *d;
  write_screen(71,18,3,5,"-");                    /* YELLOW On BLUE */

  write_scr_box(2,18,33,18,2,"Please enter Time Step (sec.)  ");
  write_screen(45,19,2,25,"Time step (s)        :-");
  write_screen(71,19,3,5,"Þ");                    /* YELLOW On L_CYAN */
  d = (double *)getnumber(71,19);  if(d != NULL) Step_t = *d;
  write_screen(71,19,3,5,"-");                    /* YELLOW On BLUE */
  num_STEP = (int)(T_max / Step_t);
  num_STEP++;                  /* Because! error of Double Variable */

  write_scr_box(2,18,30,19,2,"Please enter Acculate number  ( Range :- 2 - 10 )   ");
  write_screen(45,20,2,25,"Acculate number      :-");
  write_screen(71,20,3,2,"Þ");                    /* YELLOW On L_CYAN */
  g = (int *)getnumber(71,20);  if(g != NULL) loop_acc = *g;
  write_screen(71,20,3,2,"-");                    /* YELLOW On BLUE */

  write_scr_box(2,18,30,19,2,"Specific Tolerance Error    "
                "                          ");
  write_screen(45,21,2,25,"Tolerance Error      :-");
  write_screen(71,21,3,7,"Þ");    d = (double *)getnumber(71,21);
  if(d != NULL) tolerance = *d;
   else { goto_xy(71,21);  printf("%7.5lf",tolerance); }
  write_screen(71,21,3,7,"-");

  write_scr_box(2,18,30,18,2,"Specific Accelerate Factor  ");
  write_screen(45,22,2,25,"Accelerate Factor    :-");
  write_screen(71,22,3,4,"Þ");    d = (double *)getnumber(71,22);
  if(d != NULL) accelerate = *d;
   else { goto_xy(71,22);  printf("%4.2lf",accelerate); }
  write_screen(71,22,3,4,"-");

  write_scr_box(2,18,35,20,2,"Please select method to calculate "
                "  1. Newton Raphson Method      "
                "  2. Gauss-Seidel Method        ");
  write_screen(45,23,2,25,"Select method (1 or 2) :-");
  write_screen(71,23,3,1,"Þ");                    /* YELLOW On L_CYAN */
  g = (int *)getnumber(71,23);  if(g != NULL) NR_GS = *g;
  write_screen(71,23,3,1,"-");                    /* YELLOW On BLUE */
  if(NR_GS == 1)
    Prepare_for_NR();
```

```
 else
   Prepare_for_GS();
cls(1,18,41,23,BLUE * 16 + LIGHTGRAY);
cls(44,18,78,23,BLUE * 16 + LIGHTCYAN);
write_screen(2,18,2,34,"Please enter Number of Disturbance");
write_screen(45,18,2,19,"Number of Events :-");
write_screen(65,18,3,1,"Þ");
g = (int *)getnumber(65,18);    if(g != NULL) num_EVENT = *g;
write_screen(65,18,3,1,"-");
if(num_EVENT)
 { event = (struct ev_data *)malloc(num_EVENT * sizeof(struct ev_data));
  write_screen(11,4,2,53,"Time  Type  from  to  Admittance    Load-Power");
 }
for(i = 0 ; i != num_EVENT ; i++)
 { event[i].operate = (int)1;
  event[i].from = event[i].to = (int)-1;
  event[i].li_chrg.re = event[i].li_chrg.im = (double)0.0;

  write_screen(2,18,2,34,"Please enter Occured Time (sec)  ");
  write_screen(45,20,2,11,"Time (s) :-");
  write_screen(57,20,3,4,"Þ");
  d = (double *)getnumber(57,20);  if(d != NULL)  event[i].time = *d;
  write_screen(57,20,3,4,"-");

  write_scr_box(2,18,40,22,2,"Please enter Type of Disturbance      "
                    "1. Fault occur    2. Fault clear     "
                    "3. Fault by Admt.  4. Fault(Admt) clear"
                    "5. Line trip      6. Line reclosed    "
                    "7. Load Sheding                       ");
  write_screen(45,21,2,11,"Type    :-");
  write_screen(57,21,3,1,"Þ");
  g = (int *)getnumber(57,21);  if(g != NULL)  event[i].ev_type = *g;
  write_screen(57,21,3,1,"-");

  cls(1,18,41,23,BLUE * 16 + LIGHTGRAY);
  write_screen(2,18,2,23,"Please enter Bus number");
  write_screen(45,22,2,11,"From Bus :-");
  write_screen(57,22,3,1,"Þ");
  g = (int *)getnumber(57,22);  if(g != NULL)  event[i].from = (*g) - 1;
  write_screen(57,22,3,1,"-");

  if((event[i].ev_type == 5)||(event[i].ev_type == 6))
   { write_screen(64,22,2,10,"To Bus  :-");
     write_screen(74,22,3,1,"Þ");
     g = (int *)getnumber(74,22);  if(g != NULL)  event[i].to = (*g) - 1;
     write_screen(74,22,3,1,"-");
   }
  if((event[i].ev_type == 3)||(event[i].ev_type == 7))
   { if(event[i].ev_type == 3)
       write_screen(2,18,2,23,"Please enter Admittance");
     else
       write_screen(2,18,2,23,"Please enter Load Power");
     write_screen(45,23,2,11,"Real    :-");
     write_screen(57,23,3,6,"Þ");
     d = (double *)getnumber(57,23); if(d != NULL)  event[i].li_chrg.re = *d;
     write_screen(57,23,3,6,"-");
     write_screen(64,23,2,8,"Imag  :-");
     write_screen(73,23,3,6,"Þ");
     d = (double *)getnumber(73,23); if(d != NULL)  event[i].li_chrg.im = *d;
     write_screen(73,23,3,6,"-");
   }
  cls(44,20,78,23,BLUE * 16 + LIGHTCYAN);
  goto_xy(7,5+i);
  printf("%d  % 5.2lf   %d     %d",i+1,event[i].time,event[i].ev_type,event[i].from+1);
  if((event[i].ev_type == 5)||(event[i].ev_type == 6))
    printf("     %d",event[i].to+1);
  else
    printf("      ");
  if((event[i].ev_type == 3)||(event[i].ev_type == 7))
```

```
    { if(event[i].ev_type == 7)
        printf("          ");
      printf("%7.3lf%+ 7.3lfi",event[i].li_chrg.re,event[i].li_chrg.im);
     }
    write_screen(7,5+i,3,68,"-");
  }
 cls(1,18,41,23,BLUE * 16 + LIGHTGRAY);
 cls(44,18,78,23,BLUE * 16 + LIGHTCYAN); getch();
 cls(1,1,78,15,BLUE * 16 + LIGHTCYAN);

 Voltbus  = Make_MAT(num_STEP + num_EVENT,num_bus);
 PQload   = Make_MAT(num_STEP + num_EVENT,num_LOAD);
 Load_temp= Make_MAT(num_LOAD,1);
 PQgen    = Make_MAT(num_STEP + num_EVENT,num_bus);
 Freq     = Make_REAL(num_STEP + 1,1);
 Pm       = Make_REAL(num_STEP + 1,num_MAC);
 W_real   = Make_REAL(num_STEP + 1,num_MAC);
 d_real   = Make_REAL(num_STEP + 1,num_MAC);
 Pm_temp  = Make_REAL(loop_acc,num_MAC);
 Pe_temp  = Make_REAL(loop_acc,num_MAC);
 W_temp   = Make_REAL(loop_acc + 1,num_MAC);
 d_temp   = Make_REAL(loop_acc + 1,num_MAC);
 dW_temp  = Make_REAL(loop_acc,num_MAC);
 dd_temp  = Make_REAL(loop_acc,num_MAC);
 if(num_INDUC)
 { Edat   = Make_MAT(num_STEP + 1,num_INDUC);
   E_temp  = Make_MAT(loop_acc + 1,num_INDUC);
   dE_temp = Make_MAT(loop_acc,num_INDUC);

   for(Mac = 0 ; Mac != num_INDUC ; Mac++)
    { Convert_REC_POL(&r,&(mac_ph[induction[Mac]].volt),0);
      Rect_operation('/',mac_ph[induction[Mac]].load,r,&It);
      s.re = mac_ph[induction[Mac]].induc->Rs;
      s.im = mac_ph[induction[Mac]].induc->Xdat;
      Rect_operation('*',s,Conjugate(It),&t);
      Rect_operation('-',r,t,&(Edat->m[Mac]));
     }
 }


 write_screen(2,18,3,40,"^");          /* YELLOW on MAGENTA */
 write_screen(2,19,2,12,"Complete (%)");
 trend = (int)0;
 Freq->m[0] = frequency;
 for(Mac = 0 ; Mac != num_MAC ; Mac++)
  { W_real->m[Mac]  = 2 * M_PI * frequency;
    d_real->m[Mac]  = bus_ph[num_bus - num_MAC + Mac].volt.im * M_PI / 180;

    Convert_REC_POL(&r,&(mac_ph[machine[Mac]].volt),0);
    Rect_operation('/',mac_ph[machine[Mac]].gen,r,&It);
    Rect_operation('*',It,Conjugate(It),&s);
    s.im = s.re * mac_ph[machine[Mac]].mac_impd.re;
    Pm->m[Mac] = mac_ph[machine[Mac]].gen.re + s.im;
  }
 for(t_run = 0 ; t_run != num_STEP ; t_run++)
  { if(t_run)
     { if(num_INDUC)
        { for(Mac = 0 ; Mac != num_INDUC ; Mac++)
           { Edat->m[t_run*Edat->col + Mac].re = E_temp->m[loop_acc*E_temp->col + Mac].re;
             Edat->m[t_run*Edat->col + Mac].im = E_temp->m[loop_acc*E_temp->col + Mac].im;
           }
        }
       r.re = r.im = 0.0;
       for(Mac = 0 ; Mac != num_MAC ; Mac++)
        { W_real->m[t_run*W_real->col + Mac] = W_temp->m[loop_acc*W_temp->col + Mac];
          d_real->m[t_run*d_real->col + Mac] = d_temp->m[loop_acc*d_temp->col + Mac];
          Pm->m[t_run*Pm->col + Mac] = Pm->m[(t_run-1)*Pm->col + Mac];
          r.re += W_real->m[t_run*W_real->col + Mac] * mac_ph[machine[Mac]].inertia;
          r.im += mac_ph[machine[Mac]].inertia;
```

```
    }
    Freq->m[t_run] = r.re / (r.im * 2 * M_PI);
    Represent_LOAD();
    for(Mac = 0 ; Mac != num_bus ; Mac++)
    { Voltbus->m[trend*Voltbus->col + Mac].re = init_V->m[Mac].re;
      Voltbus->m[trend*Voltbus->col + Mac].im = init_V->m[Mac].im;
      PQgen->m[trend*PQgen->col + Mac].re = init_PQ->m[Mac].re;
      PQgen->m[trend*PQgen->col + Mac].im = init_PQ->m[Mac].im;
    }
    for(Mac = 0 ; Mac != num_LOAD ; Mac++)
    { PQload->m[trend*PQload->col + Mac].re = Load_temp->m[Mac].re;
      PQload->m[trend*PQload->col + Mac].im = Load_temp->m[Mac].im;
    }
    trend++;
  }
  if(Check_event(t_run*Step_t))
  { Clear_ITE(V_ph,num_bus);
    Clear(Y_bus->m,num_bus * num_bus);
    Form_Y_bus();
    if(NR_GS == 1)
     { Clear_ITE(I_ph,num_bus);  Clear_ITE(P_ph,num_bus);
       Clear_ITE(DPQ_ph,num_bus);  Clear_ITE(DE_ph,num_bus);
       Calculate_NR();
     }
     else
       Calculate_GS();
    Represent_LOAD();
    for(Mac = 0 ; Mac != num_bus ; Mac++)
    { Voltbus->m[trend*Voltbus->col + Mac].re = init_V->m[Mac].re;
      Voltbus->m[trend*Voltbus->col + Mac].im = init_V->m[Mac].im;
      PQgen->m[trend*PQgen->col + Mac].re = init_PQ->m[Mac].re;
      PQgen->m[trend*PQgen->col + Mac].im = init_PQ->m[Mac].im;
    }
    for(Mac = 0 ; Mac != num_LOAD ; Mac++)
    { PQload->m[trend*PQload->col + Mac].re = Load_temp->m[Mac].re;
      PQload->m[trend*PQload->col + Mac].im = Load_temp->m[Mac].im;
    }
   trend++;
  }
  if(num_INDUC)
  { for(Mac = 0 ; Mac != num_INDUC ; Mac++)
    { E_temp->m[Mac].re = Edat->m[t_run*Edat->col + Mac].re;
      E_temp->m[Mac].im = Edat->m[t_run*Edat->col + Mac].im;
    }
  }
  for(Mac = 0 ; Mac != num_MAC ; Mac++)
  { W_temp->m[Mac] = W_real->m[t_run*W_real->col + Mac];
    d_temp->m[Mac] = d_real->m[t_run*d_real->col + Mac];
  }
  acc_run = 0;
  do
  { if(num_INDUC)
    { for(Mac = 0 ; Mac != num_INDUC ; Mac++)
      { Convert_REC_POL(&r,&(init_V->m[induction[Mac]]),0);
        Rect_operation('-',r,E_temp->m[acc_run*E_temp->col + Mac],&s);
        t.re = mac_ph[induction[Mac]].induc->Rs;
        t.im = mac_ph[induction[Mac]].induc->Xdat;
        Rect_operation('/',s,t,&It);
        s.re = (double)0.0;
        s.im = mac_ph[induction[Mac]].induc->X - mac_ph[induction[Mac]].induc->Xdat;
        Rect_operation('*',s,It,&t);
        Rect_operation('-',E_temp->m[acc_run*E_temp->col + Mac],t,&s);
        s.re = s.re / mac_ph[induction[Mac]].induc->To;
        s.im = s.im / mac_ph[induction[Mac]].induc->To;
        t.re = (double)0.0;
        t.im = -2 * M_PI * frequency * mac_ph[induction[Mac]].induc->slip;
        Rect_operation('*',t,E_temp->m[acc_run*E_temp->col + Mac],&r);
        Rect_operation('-',r,s,&(dE_temp->m[acc_run*dE_temp->col + Mac]));
        Rect_operation('+',dE_temp->m[Mac],dE_temp->m[acc_run*dE_temp->col + Mac],&r);
```

```
            r.re = (r.re / 2) * Step_t;
            r.im = (r.im / 2) * Step_t;
            Rect_operation('+',E_temp->m[Mac],r,&(E_temp->m[(acc_run+1)*E_temp->col + Mac]));
           }
         }
         for(Mac = 0 ; Mac != num_MAC ; Mac++)
         { Pm_temp->m[acc_run*Pm_temp->col + Mac] = Pm->m[t_run*Pm->col + Mac];
           Pe_temp->m[acc_run*Pe_temp->col + Mac] = init_PQ->m[num_bus - num_MAC + Mac].re;
           dW_temp->m[acc_run*dW_temp->col + Mac] = M_PI * frequency / mac_ph[machine[Mac]].inertia
                                    * (Pm_temp->m[acc_run*Pm_temp->col + Mac] - Pe_temp->m[acc_run*Pe_temp->col +
Mac]);
           W_temp->m[(acc_run+1)*W_temp->col+Mac] = W_temp->m[Mac]
                                    + ((dW_temp->m[Mac] + dW_temp->m[acc_run*dW_temp->col + Mac]) / 2 * Step_t);
           dd_temp->m[acc_run*dd_temp->col + Mac] = W_temp->m[acc_run*W_temp->col + Mac] - (2 * M_PI * frequency);
           d_temp->m[(acc_run+1)*d_temp->col+Mac] = d_temp->m[Mac]
                                    + ((dd_temp->m[Mac] + dd_temp->m[acc_run*dd_temp->col + Mac]) / 2 * Step_t);
           bus_ph[num_bus - num_MAC + Mac].volt.im = d_temp->m[(acc_run+1)*d_temp->col+Mac] * 180 / M_PI;
         }
         Clear_ITE(V_ph,num_bus);
         Clear(Y_bus->m,num_bus * num_bus);
         Form_Y_bus();
         if(NR_GS == 1)
          { Clear_ITE(I_ph,num_bus);  Clear_ITE(P_ph,num_bus);
            Clear_ITE(DPQ_ph,num_bus); Clear_ITE(DE_ph,num_bus);
            Calculate_NR();
          }
         else
           Calculate_GS();
         acc_run++;
         Success = (float)((t_run*loop_acc + acc_run + evt_do) * 100.0) / (float)(num_STEP*loop_acc + num_EVENT);
         write_screen(2,18,4,(int)floor(Success/2.5),"□");
         goto_xy(33,19); printf("% 7.2f%",Success);
         write_screen(33,19,3,8,"-");
       }
      while(acc_run != loop_acc);
    }
    for(Mac = 0 ; Mac != num_bus ; Mac++)
    { Voltbus->m[trend*Voltbus->col + Mac].re = init_V->m[Mac].re;
      Voltbus->m[trend*Voltbus->col + Mac].im = init_V->m[Mac].im;
      PQgen->m[trend*PQgen->col + Mac].re = init_PQ->m[Mac].re;
      PQgen->m[trend*PQgen->col + Mac].im = init_PQ->m[Mac].im;
    }
    for(Mac = 0 ; Mac != num_LOAD ; Mac++)
    { PQload->m[trend*PQload->col + Mac].re = Load_temp->m[Mac].re;
      PQload->m[trend*PQload->col + Mac].im = Load_temp->m[Mac].im;
    }
    r.re = r.im = 0.0;
    for(Mac = 0 ; Mac != num_MAC ; Mac++)
    { W_real->m[t_run*W_real->col + Mac] = W_temp->m[loop_acc*W_temp->col + Mac];
      d_real->m[t_run*d_real->col + Mac] = d_temp->m[loop_acc*d_temp->col + Mac];
      Pm->m[t_run*Pm->col + Mac] = Pm->m[(t_run-1)*Pm->col + Mac];
      r.re += W_real->m[t_run*W_real->col + Mac] * mac_ph[machine[Mac]].inertia;
      r.im += mac_ph[machine[Mac]].inertia;
    }
    Freq->m[t_run] = r.re / (r.im * 2 * M_PI);
    if(num_INDUC)
    { for(Mac = 0 ; Mac != num_INDUC ; Mac++)
      { Edat->m[t_run*Edat->col + Mac].re = E_temp->m[loop_acc*E_temp->col + Mac].re;
        Edat->m[t_run*Edat->col + Mac].im = E_temp->m[loop_acc*E_temp->col + Mac].im;
      }
    }
}
/*****************************************************************************/
void main(void)
{ auto char *fl;

  Find_addr_of_vmode();
  setcursor(7,0);
  Initial_Program();
```

```
if(NR_GS == 1)
 { Prepare_for_NR();
   Calculate_NR(); }
 else
 { Prepare_for_GS();
   Calculate_GS(); }
write_screen(15,4,2,13,"Bus Voltage ");
Show_MAT(init_V,10,5,2);
write_screen(50,4,2,14,"Generate Power");
Show_MAT(init_PQ,45,5,2);
cls(1,18,41,23,BLUE * 16 + LIGHTGRAY);      getch();
cls(1,1,78,15,BLUE * 16 + LIGHTCYAN);
cls(44,18,78,23,BLUE * 16 + LIGHTCYAN);
if(LF_TS == 2)
 { Prepare_for_TR();
   Calculate_Euler();
   write_screen(45,18,2,31,"Please enter Output Filename :-");
   write_screen(47,19,3,30,"Þ");     fl = getstring(30,47,19);
   if(fl != NULL)
    { write_screen(47,19,3,30,"-");
      write_screen(51,23,0,40,"P□I□e□a□s□e□ □W□a□i□t□!□ □.□.□.□.□.□.□.□.□");
      f_output = fopen(fl,"wt");
      fprintf(f_output,"   Voltage of Bus + Machine \n");  Save_MAT(Voltbus,2);
      fprintf(f_output,"   Load Power of Bus LOAD  \n");  Save_MAT(PQload,2);
      fprintf(f_output,"   Gen Power of Bus + Mac  \n");  Save_MAT(PQgen,2);
      fprintf(f_output,"   Speed of each Machine   \n");  Save_REAL(W_real);
      fprintf(f_output,"   Angle of each Machine   \n");  Save_REAL(d_real);
      fprintf(f_output,"   Frequency of the System \n");  Save_REAL(Freq);
      if(num_INDUC)
       { fprintf(f_output,"   Edat of Induction Motor \n");
         Save_MAT(Edat,2);
       }
      fprintf(f_output,"   Mechanical Power of Mac \n");  Save_REAL(Pm);
      fclose(f_output);
    }
 }
 setcursor(6,7);  cls(0,0,79,24,LIGHTGRAY);  goto_xy(0,0);
}
/*****************************************************************************/
```

ประวัติผู้เขียน

นาย ประวิทย์ ปาระวนิชย์ เกิดเมื่อวันที่ 13 มิถุนายน 2517 ที่กรุงเทพมหานคร เมื่อปี 2538 สำเร็จการศึกษาปริญญาวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมไฟฟ้ากำลัง ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง จากนั้นได้เข้าศึกษาต่อในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต สาขาพลังงานไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย โดยในระหว่างการศึกษาระดับปริญญาโทที่จุฬาลงกรณ์มหาวิทยาลัยนี้ ได้รับทุนการศึกษาจากโครงการทุนบัณฑิตศึกษาภายในประเทศ สำนักงานพัฒนาวิทยาศาสตร์และเทคโนโลยีแห่งชาติ เป็นระยะเวลา 2 ปี