'95 대한전기학회 하계학술대회 논문집(1995. 7. 20~22)

객체지향기법을 이용한 전력조류계산 및 스파시티 연구

金 定 年° · 白 榮 植
慶北大學校 電氣工學科

Load Flow Analysis And Sparsity Study Using Object-Oriented Programming Technique

Jung-Nyun Kim · Young-Sik Baek
Dept of Electrical. Eng. Kyung Pook Univ.

ABSTRACT - Power system is becomming more and more complex and large. Exsiting procedual programming technique can't cope with software flexibility and maintance problems. So, Object-Oriented Programming is increasingly used to solve these problems. OOP in power system analysis field has been greatly developed. This paper applies OOP in power flow analysis, and presents new algorithm which uses only a Jacobian to solve mismatch equations, and introduces new method which is different from exsisting method to store elements.

1. 서 론

복잡한 대규모의 전력계통해석 문제를 해석하기 위해서 컴퓨터의 의존도가 더욱더 증가하고 있다. 계통이 복잡하고 대형화함에따라 해석하는 소프트웨어도 복잡해 가는 추세에 있다. 그러나 전력계통 뿐만 아니라 일반적으로 소프트웨어의 유지보수에 있어서 기존의 프로그램의 확장, 변경 및 재사용이 어렵다는 문제점이 대두되었다. 뿐만 아니라 기존의 소프트웨어의 재사용이 줄어듬에따라 하드웨어에 비해 소프트웨어의 발전은 기술축적 또한 어렵다. 소프트웨어의 유지 및 보수를 하는데 상당한 시간과 경비가 소요되며 특히 경비는 전체 소프트웨어 예산의 80%를 차지한다[11]. 이러한 문제점을 해결하기 위해서 기존의 함수중심의 프로그램에서 객체를 중심으로 프로그램하는 객체지향기법이 널리 사용되고 있다.

전력계통 해석에서도 객체지향기법은 지금까지 발전을 거듭해왔다. Foley와 Bose는 전력계통요소 각각을 객체로 모델링하고 그것을 토대로 그래픽 인터페이스의 기본이 되게 하였다. 계통요소를 객체화 하는데 객체 상호간의 기하학적인 연결상태에 중점을 두고 연구 하였다[3][4]. 그리고 실시간해석이 가능하도록 적용시켰다[6]. Liu와 Shahidehpour도 퍼스널 컴퓨터상에서 객체지향기법을 이용해 그래픽 패키지를 제시하였다[5]. 실제 계통 요소의 객체화가 중점적으로 연구되는 반면 추상적인 행렬을 객체로 표현하려는 노력도 있었다[10]. 본 논문에서는 전력계통의 기본 요소들의 일반적인 모델링을 중심으로 이를 각각의 독립된 객체로 표현하고 다른해석분야의 확장이 가능하도록 하였다. 그 일례로서 계통해석의 가장 기본인 전력조류계산에 적용시켰으며 이 계산에 필수적인 행렬식연산을 위한 스파스행렬에서는 기존의 저장방식과는 다른 저장방식을 택함으로써 객체로 표현하는데 용이하게 하였다. 또한 조류계산의 방식에 있어서는 가우스 사이델법과 고속분할법을 적용시켰다. 고속분할법에 있어서 자코비안 연산에서 하나의 자코비안으로써 연산이 가능하도록 알고리즘을 수정하였다.

2. 객체지향 프로그래밍 기법

객체지향 프로그래밍(Object - Oriented Programming)의 주요 목표는 소프트 웨어의 확장성과 재사용 가능성을 개선하여 프로그래밍 효율을 향상시키고, 소프트웨어의 유지보수를 용이하게 하며 동시에 개발비용을 줄이고자 하는 것이다.

객체는 어떤 사물의 상태를 나타내는 데이타와 그 데이타의 조작할수 있는 멤버함수로 나누어진 수 있다. 일반적으로 객체지향 프로그램은 캡슐화, 계승, 다형성의 세가지 특성을 들 수 있다.

캡슐화는 데이타와 멤버함수를 묶어주는 역할로서 특정한 객체만이 자신의 식별자로서 접근하며 데이타 조작이 가능하다. 이는 함수나 데이타를 사용자들이 직접 접근하는 기존의 방식에서 객체에게 메세지를 보내고, 메세지를 받은 객체의 반응을 중심으로 프로그램하며 직접적인 참조에 의한 오류를 방지하며 객체 데이타가 보호될 수 있다. 또한 이 특징은 프로그램의 가독성을 증가시킨다.

계승은 선조에게서 물려 받는다는 것을 의미한다. 이미 만들어진 클래스에서 수정을 가하지 않고 필요한 속성를 계승받고 추가되는 데이타를 정의할 수 있기 때문에 프로그램 소스코드의 길이가 상당히 짧아지며 효율적으로 코드를 관리할 수 있다.

다형성은 서로 다른 객체가 같은 이름의 함수를 사용하는 것으로, 서로 다른 객체가 같은 메세지를 받더라도 서로 다른 반응을 보인다. 그렇게 함으로써 프로그램의 가시성을 높일 수 있다.

3. 계통의 객체지향적 모델링

전력계통의 구성하는데 기본이되는 요소로는 모선, 전송선로, 발전기, 변압기, 부하, 커패터뱅크등이 있는데 이들의 조합으로 전체 계통시스템이 구성된다. 이들 소자를 객체로 표현하는데 있어서 각 요소들이 가져야 할 특성과 기능을 각각 객체의 데이타와 멤버함수로 표현함으로써 계통시스템을 구성할 수 있다.

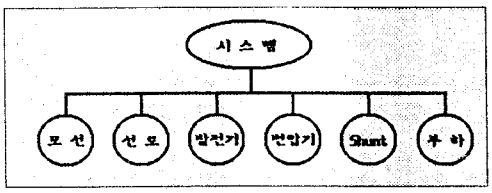

그림1. 전력계통요소의 계층구조

다음 그림은 각 객체의 심볼을 나타낸 것이다.

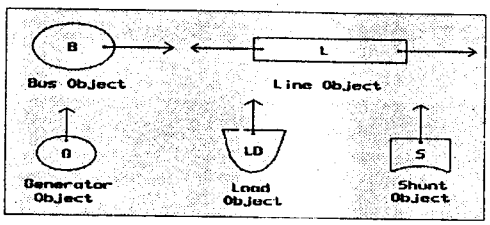

그림2. 객체의 심볼

3.1 각요소간의 메세지 전달

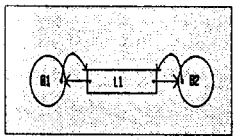

그림3. 모선-선로의 메세지 경로

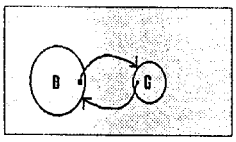

그림4. 모선-발전기의 메세지 경로

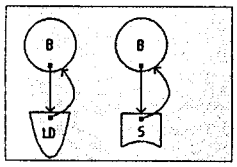

그림5. 모선-부하 및 Shunt의 메세지 경로

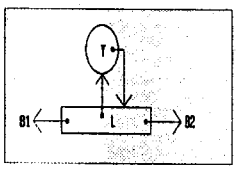

그림6. 변압기-선로의 메세지 경로

이상의 객체의 연결상태및 메세지 경로를 통하여 어떠한 한 객체에서 다른 모든 객체로의 메세지 전달이 가능하고 각 객체의 상호 연결시킴으로써 프로그램의 가독성을 증가 시킬 수 있다.

3.2 3모선 3선로 계통의 객체화

그림8는 그림7의 간단한 3모선 3선로 시스템을 객체의 심볼로서 나타낸것이다.

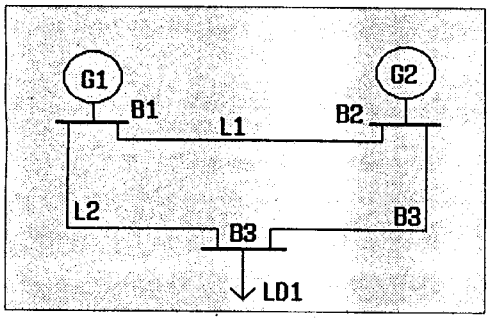

그림7. 3모선 3선로 시스템의 단선도

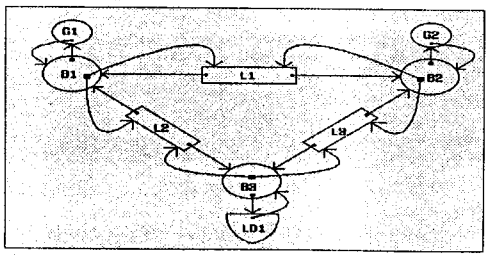

그림8. 시스템의 객체화

4. 객체지향 모델의 이용예

4.1 사고시의 모델링

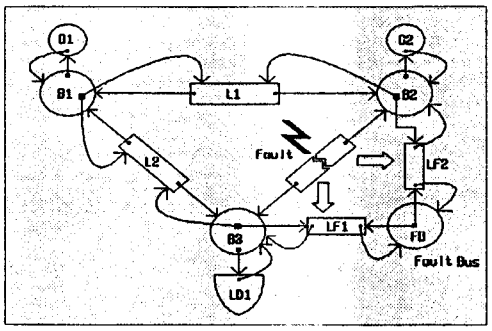

그림9. 사고시 객체의 변형(선로 3에서 사고)

그림9은 그림9에서 선로3번에서 지락사고가 발생했을때의 과정을 나타낸 것이다.

선로 3에서 사고시 사고가 발생한 지점을 사고모선으로 간주하고 새로운 모선 FB로 새로이 생성된다. 그리고 사고지점을 중심으로 선로객체 LF의 임피던스가 분리되어 선로 LF1과 LF2가 생성되고 사고모선과 연결된다. 그리고 사고가 발생한 선로3는 소멸되어서 새로운 계통이 형성된다.

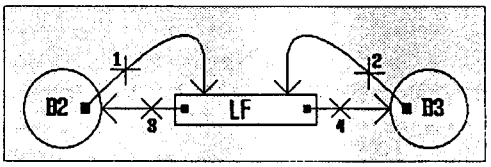

그림10. 선로의 소멸

그림10에서는 사고 선로가 소멸하는 모습을 선로와 모선을 중심으로 표현한 것이다.

소멸될 선로 객체 LF는 선로가 연결된 두 모선객체 B2, B3에게 메세지를 보내 메세지의 경로가 되는 1과 2를 제거하고, 자신이 모선에게 메세지를 보내는 경로 3, 4를 제거한다.

이렇게 함으로써 모선 객체 B2, B3는 LF에 메세지 전달을 하지 못하도록 한다. LF또한 B2, B3에게 메세지 전달을 하지 못한다. 마지막으로 선로객체 LF은 소멸한다.

이렇게 객체들이 독립적으로 존재함으로써 다른 객체들에게는 영향을 미치고 않고 계통 시스템을 유지할 수 있다. 선로가 생성될 경우에는 반대의 작업을 따르면 된다.

기존의 방식 절차위주(Procedual) 프로그램의 경우 선로가 제거되는 경우 전 시스템을 새로이 구성해야 하는데 비해 이 경우는 한 객체만을 제거함으로써 전 시스템에 영향을 미치지 않도록 구성할 수 있다. 이 경우를 IEEE14모선 시스템을 모의(simulation)하고 그 시간을 비교하여 보았다. 선로에서 사고가 발생하고 그때문에 계통을 새롭게 구성해서 조류계산을 행하는데 걸리는 시간을 비교해 보았다. 절차위주 프로그램의 경우 계산시간(T_p)은 0.299초 인데 비해 제안한 방법(T_o)에서는 0.217초의 결과가 나왔다.

기존의 방법과 제안한 방법의 비 T_o/T_p=0.726이다. 이는 계통이 변했을 경우 새로운 시스템을 구성하기 위하여 드는 시간의 차이가 나기 때문이다. IEEE30모선의 경우 T_o/T_p=0.85로 시간을 단축할 수 있다.

4.2 가우스 사이델법을 이용한 전력조류계산.

모선에서의 전압수정식은 다음과 같다.

$$V_i^{(k+1)} = \frac{1}{Y_{ii}}\left(\frac{S_i^*}{V_i^{(k)*}} - \sum_{j=1}^{n} Y_{ij}V_j^{(k)}\right) \quad [\, i \neq j \,] \qquad (식.1)$$

$$S_i = P_i + jQ_i \qquad (식.2)$$

$$Q_i = -Im[\ (Y_{ii}V_i + \sum_{j=1}^{N} Y_{ij})V_i^*] \quad [\, i \neq j \,] \qquad (식.3)$$

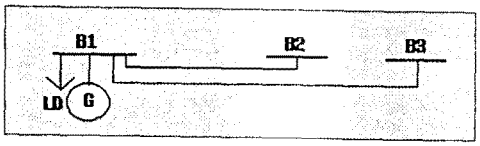

그림11. 모선에서의 전압수정

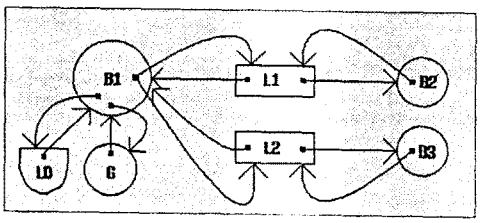

그림12. 전압수정시 메세지 경로

"전압수정하라"는 메세지를 받은 B1모선객체는 다음과 같이 반응한다.

전압 수정식 (식.1)에서 S_i를 구하기위해서는 모선에 연결된 발전기 G객체, 부하 LD객체에 메세지를 보내서 모선 B1으로의 유입전력을 구한다. 만약 모선이 PV모선인 경우 (식.3)의 Q_i를 계산하기 위해 선로객체L1, L2에 메세지를 보내 계산한다.

(식.1)에서 $\sum_{j=1}^{n} Y_{ij}V_j$항을 구하기 위해 모선에 연결된 모든 선로객체 L1, L2에 메세지를 보내 선로에서 유입전류를 구하라는 메세지를 보낸다. 메세지를 받은 선로객체L1, L2는 전류를 계산하기 위해 메세지가 온 모선객체 B1 뿐만 아니라 반대편 모선객체 (B2,B3)의 전압을 알아야 하기 때문에 두 모선객체(B2,B3)로 메세지를 보내고 그 값을 받은후 선로의 전류를 계산하고, 계산된 값을 모선객체에 준다. 이런 모선에 연결된 여러종류의 객체들에게 메세지를 보내는 형식으로 해당모선에서 전압을 수정할 수 있다.

가우스 사이델법은 객체지향 프로그램으로 구성할 경우 이해하기가 쉽고 객체의 개념을 적용시키기는 쉽지만 모선수가 많으면 반복수가 급격히 증가할 뿐만 아니라 수렴시간 또한 증가하기때문에 단일 프로세서에서는 효과적이지 못하다. 이러한 가우스 사이델법의 수렴시간 문제를 해결하기 위해서는 다중 프로세서로써 극복 할 수 있다. 객체들 상호간에 독립성이 유지되고 한 프로세서에 한 객체를 할당하고 각 프로세서가 병열처리된다면 수렴시간은 모선수에 의존하지 않을것이다[6].

5. 객체지향 기법을 이용한 뉴튼랍슨 전력조류계산

가우스 사이델 방법은 객체들 상호간의 독립성을 유지하면서 조류계산이 행해질 수가 있는 반면 뉴튼랍슨법은 자코비안 연산이 필수적이다. 이는 객체들간의 독립성은 미흡하지만 수렴속도면에서 우수한 특성을 나타낸다.

5.1 스파스 행렬의 객체화

모선 어드미턴스 행렬 구조를 살펴보면 대각요소는 값을 가지고 있으며, 비대각 요소의 경우 두모선간 연결을 가지고 있는 경우에만 값을 가지고 나머지는 모두 0이다. 일반적으로 전력계통에서 하나의 모선은 인접 2~3개 모선과 연결을 가지므로 Y_{Bus}는 '0'이 아닌 요소보다 '0'인 요소가 훨씬 많다. 이 경우 행렬은 소(sparse)하다고 하고, 그행렬을 소행렬이라고 한다. 실제 전력계통의 경우 '0'이 아닌 요소는 전체 요소의 2~3%에 불과하다. 이와 같이 스파시티(sparsity)가 큰경우 '0'의 요소까지 모두 기억하는 것은 기억용량뿐만 아니라 계산 효율성면에서도 바람직하지 못하다. 따라서 아래와 같은 sparcity기법을 이용하여 행렬요소를 저장하였다.

기존의 스파스행렬의 기억방식은 객체로 구성할때 복잡해지고, 대각요소및 비대각요소를 따로 기억시켜므로 행렬전체가 하나의 객체라는 의미가 없다. 또한 값의 삽입 및 삭제의 경우 번거로움이 발생한다. 그런 단점을 보완하기위해 새로운 기억방식을 택하므로서 그 문제점을 해결하려고 한다.

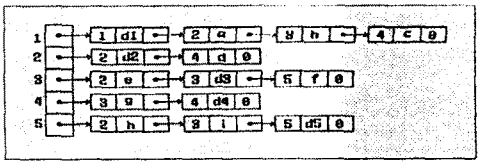

그림11. 스파스 행렬의 저장예

위의 행렬을 하나의 객체로 표현함으로서 독립성을 증가시킬 수 있다.

위의 행렬은 행렬요소의 값의 변경, 삽입, 삭제가 간단하며 행렬의 크기 또한 변경이 용이 하도록 구성하였다.

연산자 중복정의에 의한 ( )연산자로써 행렬요소값의 할당, 대입 및 변경, 삽입을 할 수 있다.

표1은 괄호( )연산자의 사용예이다.

표1. 연산자 괄호( ) (그림12 참조)

사용예	역 할
A(5,4) = 0.03	3행 3열에 0.03을 할당
A(5,4) = 15.4	3행 3열의 값이 0.03에서 15.4로 변경
x = A(3,3)	3행 3열의 값(15.4)가 x에 할당
x = A(2,5)	5행 5열의 0이 x에 할당

위와같은 저장방식을 택하고 좀 더 발전된 소행렬 모델을 제시한다. 이 모델은 행렬의 사고해석에서 행렬의 크기변환과 안정도 해석문제에서 행렬의 축약을 쉽게 구현함으로써 그 유용성을 입증하였다.

5.2 자코비안 연산

고속 분할법의 경우 다음과 같은 Jacobian연산이 필요하다.

$$\left[\ \frac{\Delta P}{|V|}\ \right] = [\ B_P\ ]\ [\ \Delta\theta\ ] \qquad (식.4)$$

$$\left[\ \frac{\Delta Q}{|V|}\ \right] = [\ B_Q\ ]\ [\ \Delta V\ ] \qquad (식.5)$$

B_P : $Y_{Bus} = G_{Bus} + jB_{Bus}$의 [$-B$] 행렬
(NPV+NPQ)×(NPV+NPQ)

B_Q : $Y_{Bus} = G_{Bus} + jB_{Bus}$의 [$-B$] 행렬
(NPQ)×(NPQ)

NPV : PV모선수

NPQ : PQ모선수

위의 B_P자코비안요소는 조류계산과정에서 변하지 않으므로 처음에 한번만 구해 놓으면 다시 구할 필요가 없다. 그러나 B_Q 자코비안의 경우 무효전력의 상·하한치가 벗어난 경우 모선의 무효전력을 상·하한치에 고정시킨후 그모선은 부하모선에 첨가

시켜야 하므로 B_Q의 크기에 변화가 생긴다. B_Q의 자코비안의 역행렬을 다시 구해야 한다.
제안한 방법에서는 B_P 자코비안을 이용하여 B_Q자코비안을 풀 수 있도록 함으로써 발전기 모선에서 무효전력의 상·하한치를 벗어난 경우의 행렬크기의 변화에 의한 B_Q^{-1}을 구하지 않고 변하지 않은 B_P 를 이용해 무효전력의 편차방정식을 구하였다. 그렇게 함으로써 B_Q자코비안 기억에 필요한 메모리를 절약하고 발전기 상·하한치를 벗어났을 경우 B_Q자코비안의 형성에드는 시간과 역행렬을 구하는데 드는 시간을 줄일수 있다.

<방법 1>

$$[\ \Delta V] = [\ B_P]^{-1}\left[\frac{\Delta Q}{|V|}\right] \qquad (식.6)$$

$$[\ A] = [\ B_P]^{-1} \qquad (식.7)$$

행렬A (NPV+NPQ)×(NPV+NPQ)

PV모선인 한행을 살펴보면 다음과 같다.

$$\Delta V_i = \sum_{j=1}^{NPQ} A_{ij}\frac{\Delta Q_j}{|V_j|} + \sum_{k=1}^{NPV} A_{ik}\frac{\Delta Q_k}{|V_k|} = 0 \qquad (식.8)$$

$$\sum_{j=1}^{NPQ} A_{ij}\frac{\Delta Q_j}{|V_j|} = -\sum_{k=1}^{NPV} A_{ik}\frac{\Delta Q_k}{|V_k|} \qquad (식.9)$$

위식에서 좌변은 반복과정에서 알수 있는 기지항들이고 우변은 미지항들이다.

$T_i = \sum_{j=1}^{NPQ} A_{ij}\frac{\Delta Q_j}{|V_j|}$ 라 하면

$$T_i = -\sum_{k=1}^{NPV} A_{ik}\frac{\Delta Q_k}{|V_k|} \qquad (식.10)$$

$$T = [\ A_0]\ [\ \frac{\Delta Q}{|V|}] \qquad (식.11)$$

행렬A_0 (NPV)×(NPV)
에서 PV모선에서의 무효전력의 변동분을 계산해 낼수 있다.
행렬 B_Q[NPQ×NPQ]대신 A_0[NPV×NPV]를 풀어야 한다.
일반적으로 조류계산에서 PV모선이 차지하는 비율이 대략 20%정도이다[7]. 이 방법은 PV모선수가 적은 계통에 적합 할 것이다.

<방법 2>
다음은 발전기 모선을 포함한 무효전력의 편차방정식이다.

$$\left[\frac{\Delta Q}{|V|}\right] = [\ B_P]\ [\ \Delta V] \qquad (식.12)$$

위 방정식에서 한 행(row)만을 살펴보면 다음과 같다.

$$\frac{\Delta Q_i}{|V_i|} = \sum_{j=1}^{NPQ} B_{ij}\Delta|V_j| + \sum_{k=1}^{NPV} B_{ik}\Delta|V_k| \qquad (식.13)$$

위식에서 PV모선의 무효전력의 변화분[ΔQ]을 계산해낸다.
ΔQ_0, ΔQ_1의 전력편차가 생겼을경우 모선의 전압의 크기의 변화를 ΔV_0, ΔV_1라 할때 다음과 같은 선형관계가 성립한다고 가정하면 다음과 같다.

$$\Delta V = [\ \frac{\Delta V_1 - \Delta V_0}{\Delta Q_1 - \Delta Q_0}]\ (\Delta Q - \Delta Q_0) + \Delta V_0 \qquad (식.14)$$

$\Delta Q_0 = 0$, $\Delta Q_1 = 1$라 하면

$$\Delta V = [\ \Delta V_1 - \Delta V_0]\ \Delta Q + \Delta V_0 \qquad (식.15)$$

여기서 PV모선에서의 전압의 변화(ΔV)는 0가 되어야 하므로

$$\Delta Q = -\frac{\Delta V_0}{\Delta V_1 - \Delta V_0} \qquad (식.16)$$

가 된다.

5.3 계산결과의 비교

표2는 기존의 두개의 자코비안으로 푸는 방법과 제안한 두가지 방법을 비교한 것이다.
세 방법에 있어서 반복횟수의 차이는 거의 없었다.
표2. 수렴시간 비교

방법 \ 모선		IEEE14 모선		IEEE30 모선	
		수렴시간	sec/iter	수렴시간	sec/iter
기존의 방법		0.1667	0.01515	0.9833	0.08939
제안한 방법	1	0.0833	0.00926	0.1667	0.01515
	2	0.0333	0.00370	0.1333	0.01212

위와 같은 결과는 기존의 방법은 발전기 무효전력이 상하한치를 벗어난 경우 자코비안의 구성과 역행렬을 구하는데 시간이 걸리기 때문인 것으로 생각된다. 참고문헌[9]에서 처럼 객체지향 프로그램에서 메모리를 할당하고 해제하는데 대략 20%의 CPU시간이 필요하다. 반면 순수한 수학 연산을 하는데는 상대적으로 대략 5%라는 적은 시간을 요구하므로 두번째 방법이 수렴특성이 우수하다.
표3은 참고문헌[9]의 객체지향 조류계산에서의 CPU시간을 도표화 한것이다.
표3. 객체지향 조류계산에서 CPU시간[9]

구 분	CPU time
메세지 전달	40.6%
객체의 생성및소멸	19.3%
순수한 수학연산	4.9%
기 타	35.2%

6. 결 론

이상과 같이 객체 지향적인 프로그래밍 기법을 사용함으로써 기존의 방법에 비교해 볼때 다음과 같은점이 개선되었다.

- 각각의 전력계통 요소를 독립된 객체로서 표현함으로써 계통 구성요소의 파악이 용이하게 하였으며 다른 전력계통 해석분야의 이용가능성이 클것이다.
- 객체지향 프로그래밍 기법을 사용하여 전력조류계산에 행하고 가우스 사이델법 및 뉴튼라슨법을 적용시켰다.
- 프로그램의 확장성을 높이기 위해 전력조류계산 및 고장해석에 이르는 분야의 해석을 하였다.
- 새로운 행렬의 저장방식으로 행렬객체를 구성하였으며 다른해석에도 유용함을 보이기 위해 전력조류계산뿐만 아니라 사고해석, 안정도 해석문제에도 적용하여 그유용성을 입증하였다.
- 고속 분할법에 있어서 자코비안을 하나를 사용함으로써 Memory면에서 낭비를 줄이고 수렴속도가 개선되었다.

7 참 고 문 헌

[1] 이재용, "컴퓨터 그래픽기능을 이용한 전력조류패키지의 개발", 경북대학교 대학원, 1992.
[2] J.Arrillaga, C.P.Arnold, B.J.Harker, "Computer Modelling of Electriccal Power Systems," John Wiley & Sons Ltd, 1983. pp. 4~42.

[3] M. Foley, A. Bose, W. Mitchell and A. Faustini, "An Object Based Graphical user Interface for Power Systems," IEEE Transactions on Power Systems, Vol. 8, No. 1, Feb. 1993, pp.97-104

[4] M. Prais and A. Bose, "A Toplology Processor that Tracks Network Modification Over Time," IEEE Transactions on Power Systems, Vol. 3, No. 3 Aug 1988, pp 992-998

[5] S. Liu, S.M. Shahidehpour, "An Object-Oriented Power System Graphics Package for Personal Computer Environment," IEEE Summer Power meeeting, Seattle, WA, July, 1992.

[6] M. Foley, A. Bose, "Object-Oriented On-Line Network Analysis", IEEE Transactions on Power Systems, Vol. 10, No. 1 Feb 1995, pp 125-131

[7] Charles A. Gross, "Power System Analysis" John Wiley & Sons Ltd, 1986. pp,255~363.

[8] Curtis F. Gerald, Patrik O. Wheatly, "Appiled Numeracal Analysis", Addison-Wesley Publishing Company, 1989. pp, 153~156,158~163.

[9] A. F. Neyer, F. F. Wu, K. Imhof, "Object-Oriented Programming For Flexible Software:Example of A Loadflow," IEEE Transactions on Power Systems, Vol. 5, No. 3 Aug 1990, pp 689-696

[10] B. Hakavik, A. T. Holen, "Power Sysyem Modeling And Sparse Matrix Operations Using Object-Oriented Programming," IEEE Transactions on Power Systems, Vol. 9, No. 2 May 1994, pp 1045-1051

[11] Guido Buzzi-Ferraris, "Scientific C++ Building Numerical Libraries the Object - Oriented Way", Addison-Wesley Publishing Company, 1993. pp, 364~381.

[12] 이재규, "이재규의 C++와의 만남", 도서출판 志源社, 1987.

[13] 양해술, 최형진, "C++와 객체지향 프로그래밍" 도서출판 정일, 1989