論 文
50A - 4 - 2

객체지향기법을 이용한 효율적인 조류계산

The Efficient Method of Power Flow Calculation using Object-Oriented Programming

金 在 顯*
(Jaehyeon Gim)

Abstract - Object-oriented programming is a solution for problems in the development, maintenance, and update of large software such as power system analysis software. However, many applications in the electrical industry critically depend on the computational efficiency of the implementation. In order to be flexible and reduce the computation time, this paper represents an efficient method for constructing a Jacobian matrix and for factorizing it, and designs the class hierarchy for power system. This method is applied to four different power systems for load flow calculation. The performance of the object-oriented program developed in C++ is assessed in computer simulation.

Key Words : Object-oriented Programming, Load Flow, *LU* Factorization

1. 서 론

객체지향 프로그래밍은 각 객체의 데이터와 이를 조작하는 함수를 가지고 있는 클래스들로 구성되어 있다. 각 객체는 독립성을 유지하며, 상위 클래스들의 기능을 계승하여 사용할 수 있는 확장성을 가지며 하위 클래스는 자기 자신만의 기능을 가질 수 있어 프로그램의 활용도를 높일 수 있은 프로그램 기법이다. 그리고 각 객체들은 모듈형식으로 되어 있어 재 사용성이 좋다. 이와 같은 장점으로 객체지향 기법은 개발시간의 단축과 신뢰성이 있는 소프트웨어 개발에 많이 적용되고 있다. 그러므로 전력계통과 같은 복잡한 시스템 해석용 프로그램 개발에 적합한 기법이다.

전력계통에서 객체지향기법을 이용하는 분야는 주로 데이터베이스와 그래픽 인터페이스 관련된 분야, 프로그램의 잦은 변경으로 유연성을 요구하는 EMS (Energy Management Systems)분야, 및 배전 계통 자동화에 많이 적용될 뿐만 아니라[1] 전력계통 해석용 프로그램인 조류계산, 안정도에도 객체지향기법이 적용되고 있다.[2,3,4,5] 또한 새로이 개발된 장치들이 전력계통에 적용됨에 따라 전력계통 해석 프로그램도 새로운 장치들의 모델링과 알고리즘의 개발을 통한 지속적인 유지보수가 필요하게 되었다. 그러므로 객체지향기법은 이러한 유연성에 대비할 수 있는 효과적인 기법이다.

그러나 객체지향 프로그램은 기존의 프로그램에 비하여 속도가 현저히 느려 아직은 많은 분야에는 채용되지 못하고 있다.[2,3,6] 본 논문에서는 실행 시간 단축을 위한 전력계통의 요소에 대한 객체 클래스를 개발하였고, 객체지향기법에 기반을 둔 자코비안 행렬 구성과 *LU*분할 기법을 개발하여 조류계산에 적용하여 비교 분석하였다.

* 正 會 員 : 順天大 電氣制御工學科 助教授, 工博
接受日字 : 2000年 12月 15日
最終完了 : 2001年 3月 28日

2. 조류계산의 정식화

조류계산은 한 모선의 주어진 조건에서 전력방정식을 이용하여 전압을 구하고, 이것으로 각 선로의 전력흐름을 계산하는 것이다. 따라서 조류계산을 위한 전력방정식은 식(1)과 같다.

$$W_{bus} = V^{*}_{bus}\, I_{bus} \tag{1}$$

모선의 전압과 전류의 관계식은 식(2)와 같다.

$$I_{bus} = Y_{bus}\, V_{bus} \tag{2}$$

본 논문에서 사용한 조류계산 방법은 Newton-Raphson법을 이용하였다. 이는 전력방정식을 선형화하여 근을 구하는 것으로 식(3)과 같다.

$$\Delta W = [J]\, \Delta V \tag{3}$$

여기서 $\Delta W = W^{spec}_{bus} - W_{bus}$ 전력 편차이다. W^{spec}_{bus}은 주어진 전력이고, $[J]$ 은 자코비안 행렬이다. 전압 편차는 자코비안 행렬의 역 행렬을 취함으로서 구할 수 있다

$$\Delta V = [J]^{-1} \Delta W \tag{4}$$

식(4)에서 전압편차를 구하여 식(5)과 같이 새로운 전압 값으로 개선하여 편차 전력이 허용오차 범위까지 반복 계산한다.

$$V^{new} = V^{old} + \Delta V \tag{5}$$

또한 조류계산에서 제어 변수로서는 발전기의 모선전압제어와 변압기의 모선전압제어가 있다. 이 제어변수들은 Newton-Raphson법에서 조류계산의 수렴속도에 많은 영향을 미친다. 본 논문의 조류계산에서 사용한 발전기의 전압과 변압기의 탭 제어방법은 식(6)와 (7)과 같다. [7]

$$V^{new} = V^{old} + a(V^{spec} - V) \tag{6}$$

$$T^{new} = T^{old} + b(V^{spec} - V) \tag{7}$$

여기서 V^{spec} 은 제어하려는 전압, V은 매 반복 시 계산된 전압이다.

3. 클래스 구축

객체지향기법는 객체들의 모델링과 객체간의 상호관계에 의하여 프로그램이 수행되므로 프로그램의 재사용성과 유연성의 확보를 위하여 객체들의 적절한 클래스 구축이 중요하다. 일반적으로 객체지향기법에서 클래스의 구조는 실제로 전력계통에 구성되는 요소들로 각각의 클래스를 구성한다.[2,5] 본 논문에서의 클래스는 전력계통 구성을 나타내는 상위 클래스와 계통 해석을 위한 하위 클래스의 2단계로 나눈다. 상위 클래스는 전력 계통 해석에 필요한 모든 데이터를 저장하는 클래스이며 하위 클래스는 상위 클래스를 상속받아 전력계통 해석을 실행하는 클래스이다. 상위 클래스는 발전기, 부하, 모선, 선로, 변압기 등의 클래스가 있으며 각각의 클래스는 동적 연결 리스트로 연결되어 계통의 크기에 상관없이 실행되는 유연성을 가지고 있으며 저장용량도 적게 한다. 하위 클래스는 전력계통 해석을 위한 클래스로서 조류계산, 고장계산, 안정도 등의 클래스가 있을 수 있다. 하위 클래스에서 멤버 변수는 계통 해석을 위한 각 각의 기능별로 입, 출력되는 데이터를 정의하고, 멤버 함수는 그 기능을 수행하게 된다. 멤버 함수 실행 중 필요한 전력계통의 데이터는 상위 클래스에서 상속받아 사용하게 된다. 그리고 상위 클래스의 멤버 변수는 데이터를 저장하는 변수들을 가지며, 멤버 함수로는 그 데이터를 입출력을 위한 함수와 데이터 계산에 관련된 함수가 있다. 한 클래스에서 다른 클래스의 데이터가 필요할 경우 그 클래스는 한 멤버 변수로 다른 클래스의 포인터를 지정하여 데이터를 찾기 위한 실행시간을 단축하도록 하였다. 또한 각 클래스가 독립적으로 실행할 수 있게 하기 위하여 클래스의 멤버 변수는 자기 자신의 클래스에서만 그 변수 값을 지정하도록 하였다. 그리고 프로그램의 수정이 필요할 때 그 객체만의 수정으로도 가능하므로 재사용과 유연성을 확보할 수 있다. 이와 같이 두 단계로 클래스를 구성함으로 객체지향 기법의 장점을 가지고 있으면서 실행속도를 단축할 수 있다. 그림 1은 전체 클래스 구조를 나타내며, 이중실선은 상위클래스를 하위클래스가 상속을 나타내고 실선은 클래스가 다른 클래스의 포인터를 멤버 변수로 가지고 있는 것을 말한다. 다음절들은 조류계산에서 필요한 클래스들의 기본 구성 요소만을 정의한다.

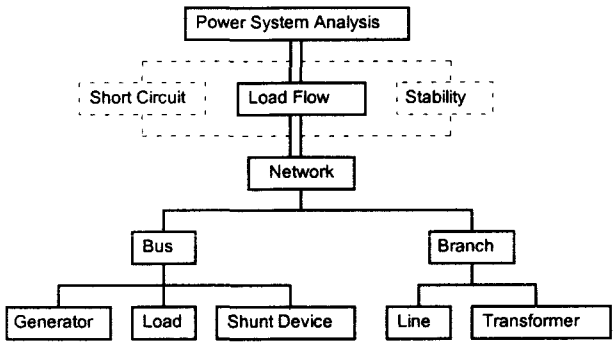

Figure 1. Class Structure
그림 1. 클래스 구조

3.1. 전력계통 클래스

전력계통 클래스는 전력계통해석을 위한 클래스로서 해석에 필요한 클래스를 정의하고 실행하는 클래스이다. 이 클래스는 조류계산 뿐만 아니라, 고장계산, 안정도 계산 등의 계통을 해석하는 멤버함수를 정의하고 있다. 그러나 여기서는 조류계산에 필요한 멤버함수만을 정의한다. 조류계산 클래스는 네트워크 클래스에서 상속받아 필요한 데이터를 사용하며 조류계산에 필요한 멤버함수들이 있다. 즉 조류계산에 사용되는 알고리즘이 이 멤버함수를 이용해서 구현된다. 예를 들면 뉴턴 랩슨법, 가우스 사이델법, 자코비안 행렬 계산, 및 조류계산결과의 입출력 등이 있다.

조류계산 클래스에서 자코비안 행렬의 요소 계산은 모선에 관련된 요소는 모선 클래스에서 계산하고, 가지 클래스에 관련된 요소는 가지 클래스에서 계산되고 그 결과로 자코비안 행렬이 구성된다. 그러므로 각 객체가 독립적으로 실행할 수 있게된다. 또한 조류계산에서 필요한 각종 조건 정수들에 대한 입력은 *ReadCtrlData*에서 수행한다.

```
template <class T>
class PowerSystem : public LoadFlow<T>
{
 public :
     void LoadFlowCalculation();
     . . . . . .
 private :
     LoadFlow<T> *LF;
};

template<class T>
class LoadFlow : public Network<T>
{
private :
 double AccFct;                  // Acceleration Factor
 int IterMax;                    // Criterion for Load Flow
 double VangCrit, VmagCrit;
 double PCrit, QCrit;
 Matrix<T> Jacobian;
                                 // Make Jacobian
 void CalJacobian(ofstream &, Matrix<double> &);
 void LFPrintOut(ofstream &); // Print out the result
public :
 LoadFlow();
 ~LoadFlow();
 void ReadCtrlData(char *);  // read the criterion for LoadFlow
 int GaussSeidel(ofstream &);
 int NewtonRaphson(ofstream &);
};
```

3.2. 네트워크 클래스

네트워크 클래스는 계통해석에 필요한 모든 데이터를 저장하고 있는 클래스로서 모선, 선로, 부하, 발전기 등의 클래스들이 연결 리스트에 의하여 연결되어 있는 연결리스트 포인터를 가지고 있다. 이들 클래스들은 전력계통이 서로 연결되어 있는 것과 같이 각 클래스를 포인터로 연결하고 있다. 예를 들면 선로의 조류계산에서 모선의 전압이 필요하면 모선을 찾기 위한 이동을 하지 않고 직접 모선 포인터에서 전압을 사용할 수 있다. 그리고 한 모선에 여러 회선의 선로가 연결되어 있을 경우는 선로가 연결 리스트로 연결된 포인터를 모선이 가지고 있으므로 여러 선로에 관련된 계산을 쉽게 할 수 있다. 네트워크 클래스의 멤버함수에는 모선, 선로, 부하 및 발전기 등의 추가 또는 제거에 따른 전력계통의 구조 변경을 위한 함수 및 조류계산을 효율적으로 하기 위한 최적 오더링 함수 등이 있다.[8] 최적 오더링 함수가 실행된 후는 모선의 연결 리스트가 최적 오더링 순서로 연결되어 계산을 위한 데이터 접근이 용이하다. 전력계통의 해석을 위한 하위 클래스는 네크워크 클래스를 상속하여 전력계통의 모든 데이터를 공유하여 실행하게된다. 특히 객체지향 기법을 이용하는 프로그램의 계산속도는 어떻게 데이터가 구성되어 있는가에 따라 좌우된다. 그러므로 각각의 객체는 알고리즘 적용이 용이하도록 구성되고, 또한 알고리

즘은 객체가 구성하고 있는 순서로 실행되도록 한다. 이 클래스에 포함하고 있는 각각의 변수는 다음과 같이 정의되며, 네크워크 클래스의 코드는 다음과 같다.

```
template <class T>
class Network
{
  Network();
  ~Network();
  LinkedList<Bus<T> > *pcBus;
  LinkedList<Branch<T> > *pcBranch;

  void ReadData(char *FileName);
  LinkedList<Bus<T> > *Ordering();
        . . . . . .
};
```

3.2.1. 모선 클래스

전력계통의 모선은 발전기, 부하, 및 조상설비 등의 설비가 연결되어 있을 뿐만 아니라 계통을 구성하는 선로 또는 변압기들과 연결되어 있다 그리하여 모선 클래스는 그림 2와 같이 이들을 포인터로 상호 연결관계를 구성하고 있다. 또한 모선 번호, 이름, 전압 등의 기본요소를 포함하고 있다. 특히 여러 선로나 발전기 및 부하가 한 모선에 연결되어 있는 경우는 각각의 클래스를 연결리스트 클래스[9]를 이용하여 여러 개의 클래스를 연결하고 있어 모선에 관련된 데이터에 쉽게 접근 할 수 있다. 모선 클래스의 멤버함수에는 모선 데이터 읽기와 결과를 프린트하는 함수 및 자코비안행렬을 계산하는 함수가 있다. 모선클래스의 코드는 다음과 같다.

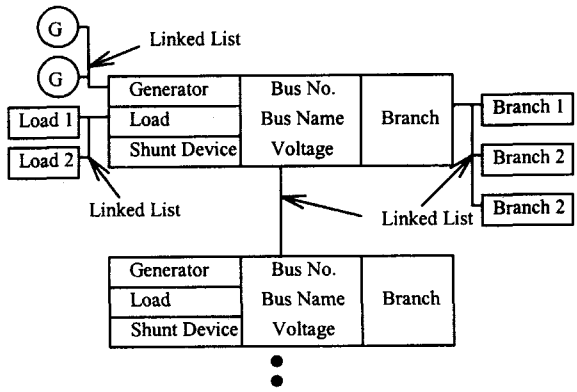

Figure 2. Connectivity of Bus Class

그림 2 모선 클래스의 연결관계

```
template <class T>
class Bus
{
  int No;                    // bus number
  char Name[12];             // bus name
  int ID;     // 1 : load bus,  2 : generator bus
              // 3 : swing bus, 4 : disconnected bus
  int Area;
  int Zone;
  double Vmag, Vang;         // bus voltage
  double basekV;
  LinkedList<Generator<T> > *pcGen;  // for generator
  LinkedList<Load<T> > *pcLoad;      // for load
  LinkedList<Shunt<T> > *pcShunt;    // for shunt
  LinkedList<Branch<T> > *pcBranch;  // for branch

  int ReadData(ifstream &);
  int CalFillinTerm();
  void CalJacobian(Matrix<T> &);
  void LFPrintOutOOP(ofstream &, double);
        . . . . . .
};
```

3.2.2. 가지 클래스

가지 클래스는 선로와 변압기를 나타내는 클래스이다. 선로의 모델은 일반적으로 많이 사용하고 있는 π형을 사용하였으며 R, X, 및 Y로 나타내었다. 이 클래스는 선로가 연결되어 있는 모선을 모선클래스 포인터로 지정하여 모선 클래스의 데이터에 쉽게 접근할 수 있으며, 변압기클래스도 포함되어 있다. 이 변압기클래스는 모선 전압조절기능(탭 조정)과 조류제어 기능(위상 변환)을 하는 멤버함수를 가지고 있다. 각 변압기의 탭조절은 변압기 클래스에 있는 데이터만을 가지고 계산되어지므로 그 알고리즘의 수정이 용이하다. 그리고 새로이 개발된 선로의 제어장치(예를 들면 HVDC, FACTS 등)들도 이 클래스에 쉽게 추가하여 이용 할 수 있다. 이 클래스는 선로와 변압기의 데이터를 읽는 함수, 선로 조류를 계산하는 함수, 자코비안행렬 계산에 필요한 함수를 멤버로 가지고 있다. 가지클래스와 변압기클래스의 코드는 다음과 같다.

```
template <class T>
class Branch
{
  int No;
  Bus<T> *pFrBus, *pToBus;
  double R, X, Y;          // line constant
  Transformer<T> *pTr;  // transformer data

  int ReadData(ifstream &);
  void CalJacobian(Matrix<T> &);
  void CalFlow();
        . . . . . .
};

template <class T>
class Transformer
{
  int No;
  double Tap, Tmin, Tmax, Tang;
  double CtrlV, CtrlVmax, CtrlVmin;
  Bus<T> *pFrBus, *pToBus;
  Bus<T> *pCtrlBus;

  int ReadData(ifstream &);
  int CtrlTap();
        . . . . . .
};
```

3.3. 발전기 클래스

발전기 클래스는 용량범위를 결정하는 최대, 최소의 유효, 무효전력과 발전기가 연결된 모선 및 전압을 제어할 모선이 포함되어 있다. 그리고 이 모선과 포인터로 연결되어 접근이 용이하게 한다. 이 클래스의 멤버함수는 발전기의 데이터를 읽는 함수, 무효전력 제한 함수 및 발전기 단자전압제어 함수들이 있다. 특히 전압제어 모선의 제어를 위한 어떤 알고리즘을 사용할 것인가에 따라 조류계산의 수렴속도에 많은 영향을 미친다. 새로운 제어 알고리즘이 개발되면 다른 클래스에 영향을 주지 않고 이 클래스의 수정만으로 가능하므로 프로그램의 재사용이 좋다. 그러나 제어 알고리즘은 본 논문의 범위를 벗어나므로 포함하지 않았다. 발전기 클래스의 코드는 다음과 같다.

```
template <class T>
class Generator
```

```
{
 int No;
 double baseMVA;
 double Pgen, Pmax, Pmin, Qgen, Qmin, Qmax;
 Bus<T> *pCtrlBus;  // controlled bus
 Bus<T> *pBus;      // connected bus
 int ReadData(ifstream &);
 int CheckQ();
 void AdjJacoGen(Matrix<T> &, int);
 void CtrlGenVolt();
    . . . . . .
};
```

3.4. 부하 클래스

부하 클래스는 데이터 읽기와 일정 전력, 일정 전류, 일정 임피던스의 부하의 특성을 포함하고 있으며, 부하특성을 멤버함수로 추가할 수 있다. 부하클래스의 코드는 다음과 같다.

```
template <class T>
class Load
{
 int No;
 int Area;
 int Zone;
 double PL, IP, YP;
 double QL, IQ, YQ;
 Bus<T> *pBus;

 int ReadData(ifstream &);
    . . . . . .
};
```

3.5. 행렬 클래스

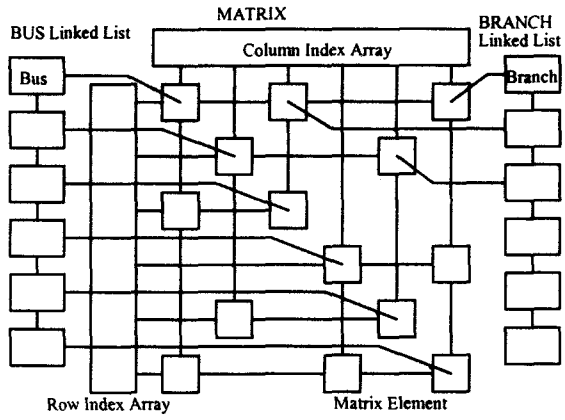

Figure 3. Matrix Structure
그림 3. 행렬 구조

행렬 클래스에서 대부분의 조류계산이 수행되므로 행렬 클래스의 성능이 전체 프로그램의 속도를 많이 좌우한다. 객체지향프로그램은 데이터에 접근하는 시간이 많이 소요되므로 배열을 이용하는 프로그램에 비하여 행렬 계산에 시간이 많이 소요된다. 그러므로 본 논문에서는 연결리스트보다 배열의 요소에 접근하는 시간이 빠르므로 행렬 클래스의 생성과 동시에 행렬의 인덱스를 동적 메모리 배열로 할당하여 행렬의 요소에 접근을 빠르게 하였다. 저장 용량을 적게 하기 위하여 행렬요소는 동적 연결 리스트를 이용하여 필요한 메모리만 사용하게 하였다. 그러므로 행렬은 시스템에 따라 크기가 결정되는 구조를 가지게 되어 프로그램의 유연성을 가지고 있다.

조류계산의 자코비안 행렬계산에 필요한 데이터가 모선 클래스와 가지 클래스에 있으므로 각 모선 클래스와 가지 클래스에서 자코비안 행렬요소를 계산한다. 그러므로 각 클래스에 자코비안 행렬요소를 지정하는 포인터 가지고 있어 행렬요소에 접근하는 시간이 단축된다. 그 구조는 그림3와 같다. 이와 같이 함으로서 자코비안 행렬을 구성할 때 행렬의 요소로 이동하지 않고 모선 연결리스트와 가지(선로, 변압기) 연결리스트에서 직접 자코비안 행렬요소가 계산된다. 또한 각 클래스의 실행 시 독립성을 가지게 프로그램을 개발할 수 있다.

행렬 클래스는 종복 연산자를 사용하여 프로그램을 쉽게 할 수 있게 하였다. 행렬 클래스의 코드는 다음과 같다.

```
template <class T>
class Matrix
{
 int Rows, Cols;
 MatrixIndex<T> *RowIndex, *ColIndex;
 Matrix(int, int);
 ~Matrix();
 void Insert(int, int, T);
 void Erase();
 Matrix &LU();
 Vector<T> *Solve(Vector<T> &b);

 T operator () (int, int);
 const Matrix<T> & operator = (const Matrix<T> &);
 void Output(ostream & out) const;
    . . . . . .
};
```

3.5.1. *LU*분할와 전후진 대입법

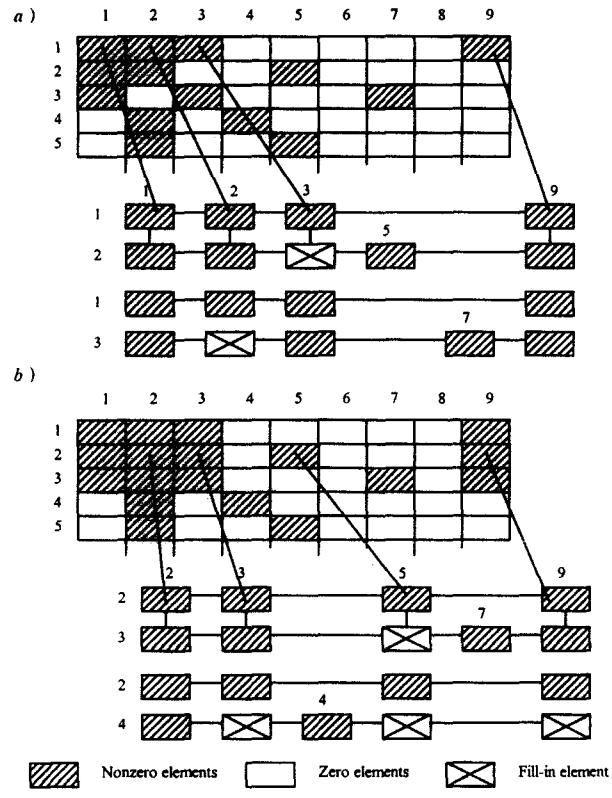

그림 4 *LU* factorization procedure
그림 4. *LU*분할 절차

고정행렬을 이용한 경우는 인덱스를 이용하여 행렬의 요소에 직접 이동하지만, 행렬이 동적 연결리스트로 연결되어

있을 때는 행렬요소의 위치로 찾아가는데 처음포인터로부터 시작해서 한 포인터씩 이동해서 위치를 찾아가게 되므로 시간이 많이 소모된다. 그러므로 본 논문에서는 LU분할을 수행하는 동안 필요하게 될 행렬의 요소들을 동적 연결리스트에서 연결되어 있는 순서대로 행렬 요소를 사용하게 하는 알고리즘을 이용함으로 이동하는 시간을 최소화하였다. LU분할을 위한 구체적인 설명은 다음과 같다.

그림 4(a)에서 피봇을 (1,1)을 선택하고 피봇 행은 식(8)과 같이 계산하여 상삼각행렬 U의 요소를 계산한다.

$$A_{1j}=A_{1j}/A_{11} \qquad j=2,\ldots,N \tag{8}$$

다음 행들에 대한 피봇된 열의 요소는 식(9)과 같이 계산된다.

$$A_{i1}=A_{i1} \qquad i=2,\ldots,N \tag{9}$$

피봇된 다음의 행과 열은 이미 계산된 피봇의 행과 열을 인자로 식(10)과 같이 계산한다.

$$A_{ij}=A_{ij}-A_{i1}*A_{1j} \quad i=2,\ldots N, \quad j=2,\ldots,N \tag{10}$$

다음행 계산시 인자로 사용된 피봇행은 피봇된 행의 포인터를 기억하고 있으면서 계산되어지므로 이동을 하지 않는다. 그림4(b)와 같이 다음 피봇을 선정되면 위와 같이 반복계산한다. 이렇게 같은 방법으로 행렬의 마지막 행까지 피봇을 반복하여 계산하면 LU분할을 실행한다.

피봇은 이미 최적 오더링이 되어 있으므로 행렬의 순서대로 피봇을 하면 동적 연결 리스트로 연결되어 있는 순서대로 진행할 수 있어 행렬요소에 접근하기 위한 불필요한 이동 없이 LU분할이 효율적으로 진행된다. 특히 계산 중에 새로이 추가되는 요소가 있으면 동적 연결 리스트에 삽입을 쉽게 할 수 있는 이점이 있다.

행렬을 LU분할된 식은 (11)과 같다.

$$LUx=b \tag{11}$$

식(11)은 식(12)와 (13)으로 나타낼 수 있다.

$$Ly=b \tag{12}$$

$$Ux=y \tag{13}$$

전진 대입법은 식(12)의 해를 구하는 것으로 식(14)와 같이 구한다.

$$y_i=b_i-\sum_{j=1}^{i-1}L_{ij} \quad i=1,\ldots,N \tag{14}$$

여기서 $L_{ij}=L_{ij}*y_j \quad j=1,\ldots,i-1$ (15)

식(14)으로 y_i을 구한다. y_{i+1} 계산 시 이전에 구한 y_i을 다시 사용하게 되는데 그때마다 계산하게 되면 행렬요소로의 이동이 많아지므로 y_i을 계산할 때 식(15)를 이용하여 앞으로 사용하게 될 모든 L_{ij} 계산함으로 이전 L_{ij}을 계산하기 위한 이동을 줄일 수이다. 후진 대입법은 식(16)과 같다.

$$x_i=y_i-\sum_{j=i+1}^{N}U_{ij} \quad i=N,\ldots,1 \tag{16}$$

여기서 $U_{ij}=U_{ij}*x_j \quad j=i+1,\ldots,N$ (17)

LU행렬의 대각요소가 그림2와 같이 전력계통의 모선클래스와 행렬의 대각 요소가 서로 포인터로 연결되어 있으므로 행렬의 대각 요소의 위치로 직접 이동할 수 있어 후진 대입법의 계산 시간을 단축한다. 이와 같이 계산함으로 자코비안 행렬의 해가 효율적으로 계산된다.

4. 사례연구

본 논문에서 개발한 클래스 구조를 이용하여 조류계산 프로그램을 개발하였다. 클래스 구조와 행렬 계산 알고리즘의 성능을 비교하기 위하여 연결리스트를 이용한 프로그램과 순환호출을 이용한 프로그램을 추가하여 분석하였다. 이 프로그램이 사용한 조류계산 방법은 Newton-Raphson법을 사용했으며 희소행렬기법을 이용하였다. 부하는 일정 전력 모델을 사용하였고 두 종류의 컴퓨터를 이용하여 실행속도를 비교 분석하였다.

4.1. 시험 계통

본 논문에서 개발한 객체지향 프로그램의 성능을 비교하기 위하여 4개의 계통에 적용하였다. Bus57, Bus118은 IEEE 기본 모형 시스템으로서 참고문헌[10]의 데이터를 이용하였고 시스템 A, B는 실제 운영되고 있는 실 계통을 이용하였다. 시스템의 특성은 표1과 같다.

Table 1. Test Systems
표 1. 모의 계통

System	Buses	Lines	Generators
Bus57	57	79	7
Bus118	118	179	53
A	306	521	85
B	5066	6200	511

4.2. 성능 비교

성능 비교에 사용한 컴퓨터는 Intel Pentium II 350MHz로서 메모리는 64MB가 있는 컴퓨터와 Pentium III 700MHz, 256MB의 컴퓨터를 사용하였고 속도를 비교하기 위하여 기본적인 조류계산은 같은 알고리즘을 이용하여 가능한 반복계산 횟수를 같게 하였다. 그러나 컴퓨터에 따라 표2와 같이 B시스템의 반복회수가 다르게 나타났다. 이는 계통의 많은 제어변수(발전기의 무효전력 또는 변압기기의 탭 등)와 소수 연산 능력이 컴퓨터에 따라 다르기 때문이다.

Table 2. Iteration Number
표2. 반복횟수

System	Petium II	Petium III
Bus57	6	6
Bus118	5	5
A	5	5
B	38	35

표3, 4는 4개의 시스템의 실행 시간을 각 프로그램별로 비교한 것이다. 표3, 4의 결과에 표시한 실행 시간은 다음으로 나타내었다.

T1 :자코비안 행렬 계산과 모선전압 갱신을 위한 반복시간,
T2 :행렬요소에 각 모선, 선로에 연결 시간을 포함한 시간
T3 : 전체 프로그램 실행시간

프로그램은 다음과 같다.

1 : 본 논문에서 개발한 기법을 이용한 프로그램
2 : 1에서 전, 후진 대입법을 순환호출을 이용한 프로그램
3 : 행렬의 인덱스를 연결리스트로 사용하고, 순환호출을 이용한 프로그램

이 결과에서 시스템이 커질수록 연결 리스트를 이용한 프로그램이 실행시간의 증가폭이 큰 것으로 나타났다. 이는 고정으로 정렬된 배열에서 데이터의 삽입이나 제거에 소요되는 시간보다 연결리스트에서 데이터의 삽입이나 제거가 더 많은 시간을 소모하기 때문이다. 그러나 고정배열은 메모리의 낭비나 부족으로 프로그램의 유연성을 저해하게 된다.

Table 3. CPU Time of Petium II
표 3. 펜티움 II의 CPU 시간

System		1 [sec]	2 [sec]	3 [sec]
Bus57	T1	0.040	0.040	0.040
	T2	0.110	0.110	0.110
	T3	0.160	0.165	0.160
Bus118	T1	0.055	0.055	0.055
	T2	0.165	0.165	0.215
	T3	0.350	0.380	0.380
A	T1	0.050	0.110	0.110
	T2	0.440	0.465	0.685
	T3	0.880	0.965	1.260
B	T1	20.845	22.490	25.705
	T2	24.995	26.605	110.485
	T3	59.155	60.770	199.895

Table 4. CPU Time of Petium III
표 4. 펜티움 III의 CPU 시간

System		1 [sec]	2 [sec]	3 [sec]
A	T1	0.043	0.045	0.05
	T2	0.196	0.196	0.28
	T3	0.390	0.390	0.52
B	T1	12.208	13.255	13.370
	T2	14.561	15.908	76.064
	T3	35.78	37.268	136.791

본 논문에서 개발한 알고리즘을 이용하여 행렬의 연산을 수행하면 연결리스트를 이용한 경우와 고정 인덱스를 사용한 경우의 연산시간이 많은 차이가 나지 않았다. 순환 호출을 이용한 전,후진 대입법이 순환 호출을 사용하지 않은 경우보다 조금 느렸다. 그러나 시스템의 크기가 적을수록 그 차이는 없었다.

5. 결 론

본 논문은 객체지향 프로그램을 위한 전력계통의 클래스를 실제계통 모형을 유지하고 해석에 필요한 멤버함수를 보유하고 있으므로 보다 쉽게 다른 프로그램으로 확장이 가능한 클래스를 개발했다. 또한 전력계통 해석용 프로그램을 객체지향 기법의 장점인 개발의 편리성과 유지보수 용이성을 유지하면서 결점인 계산 속도의 문제점을 개선하는 알고리즘을 개발하여 조류계산에 적용하였다. 특히 본 논문에서 제시한 방법은 LU분할과 전,후진 대입법시 행렬의 인덱스가 동적 연결리스트가 연결된 순서대로 이동할 수 있게 알고리즘이 되어 있어 동적 연결리스트를 이용할 때 문제시되는 행렬 요소를 찾아가는 시간을 최소화하였다. 객체지향 기법을 효율적으로 적용하기 위해서는 객체가 해석을 편리하도록 구성하는 것뿐만 아니라 객체가 구성되어 있는 순서로 알고리즘이 실행되게 하는 것도 중요하다.

감사의 글

본 논문은 순천대학교 공과대학 학술재단의 지원에 의하여 수행되었음

참 고 문 헌

[1] Mike Foley, Anjan Bose, William Mitchell, Antony Faustini, "An Objected Based Graphical User Interface for Power Systems," IEEE Trans. on Power Systems, vol.8, No.1, Feb. 1993 pp97-104
[2] Andreas F. Neyer, Felix F. Wu, Karl Imhof, "Object-Oriented Programming for Flexible Software Example of a Load Flow," IEEE Trans. on Power System, Vol.5, No.3, aug. 1990 pp689-696
[3] Ji-Ho Park, Young-Sik Baek, "The Power System Stability Analysis Method using Object-Oriented Programming", Trans. KIEE Vol.47, No.8, Aug. 1998
[4] Ji-Ho Park, Young-Sik Baek, "A Study of the Power System Stabilizer Design using Object-Oriented Method" Trans. KIEE Vol. 48A, No. 6, June 1999
[5] A Manzoni, A. S. e Silva, I. C. Decker, "Power Systems Dynamics Simulation Using Object-Oriented Programming", IEEE Trans. on Power Systems Vol.14, No. 1, Feb. 1999
[6] B. Hakavik, A. T. Holen, "Power System Modelling and Sparse Matrix Operations Using Object-Oriented Programming," IEEE Trans. on Power Systems, Vol. 9, No. 2, May 1994
[7] J. Mescua, "A Decoupled Method for Systematic Adjustments of Phase-Shifting and Tap-Changing Transformers", IEEE Trans. on Power Apparatus and Systems, Vol. 104 No. 9, Sept. 1985
[8] Jaehyeon Gim, Soyoung Lee, " A Study of the Effective method of LU factorization for Newton-Raphson Load Flow" KIEE Summer Meeting 2000
[9] Sartaj Sahni, "Data Structures, Algorithms, and Applications in C++," McGraw-Hill 1998
[10] Y. Wallach, "Calculations & Programs for Power System Networks," Prentice Hall, 1986

저 자 소 개

김 재 현 (金 在 顯)

1956년 1월 25일 생. 1977년 홍익대 전기공학과 졸업. 1989년 University of Texas at Arlington 졸업(석사). 1993년 동대 졸업(박사). 현재 국립 순천대 전기제어공학과 조교수

Tel : 061-750-3545,
E-mail : jhg@sunchon.ac.kr