객체지향 기법을 적용한 삼상조류계산

이영민*, 김건중*, 김원겸*, 장정태**
*충남대학교, **전력연구원

Three Phase Loadflow Applied for Object-Oriented Programming

Young-Min Lee*, Kern-Joong Kim*, Won-Kyum Kim*, Jeong-Tae Jang**
*Chungnam National University, **KEPRI

Abstract - 조류계산은 전력계통해석에서 가장 기본적인 것이다. 일반적으로 조류계산은 선로의 3상을 평형으로 간주하여 한상에 대해서만 해석하였다. 삼상조류계산의 복잡함에 비해 그 필요성은 크지 않았기 때문이었다. 한편, 80년대에 소프트웨어 위기의 대안으로 제시되었던 객체지향기법(OOP)은 객체의 효율적인 모델링을 통해 복잡하고 거대한 프로그램의 작성을 보다 용이하게 할 수 있도록 하였다. 본 논문에서는 전력계통의 콤포넌트와 그 콤포넌트로 구성된 전력계통을 모델링하였고 계산에서 사용하는 수학적 모델을 모델링하였다. 또한 본 논문에서 사용한 객체지향 언어인 C++의 큰 특징인 template을 적용하였다. 결과적으로 기존의 단상조류계산과 삼상조류계산이 사용되는 콤포넌트의 모델이 다른 것을 제외하고는 전체적인 구조를 동일하게 할 수 있었다.

1. 서 론

Fortran이나 C와 같은 절차지향(procedural) 언어는 대형 소프트웨어 프로젝트에서 많은 문제점이 발생되었다. 특히 소프트웨어의 유지, 보수 측면에서 절차지향언어는 본래 노력의 배이상을 필요로 하게 되었다. 소프트웨어 위기라고 일컬어지는 이런 기존 언어의 문제점에 대한 대안으로 제시된 객체지향기법은 프로그램을 일의 진행 순서인 절차(procedure)로 보지 않고 데이터와 행위로 구성된 객체간의 상호작용이라고 보았다. 객체지향 기법의 적용으로 소프트웨어는 신뢰성, 안정성, 재사용성등이 제공될 수 있어 공학의 여러분야에 적용하려는 많은 연구결과가 발표되었다.

전력계통 분야에서도 객체지향기법을 적용하려는 움직임이 활발하였으며 특히 조류계산에 대한 연구는 국내에서도 여러 연구가 있었다. 본 논문은 94년 하계학술대회에서 발표된 '객체지향 프로그램의 조류계산 적용'이라는 논문의 내용에 기반한 것으로 객체지향 언어인 C++의 템플릿(template)과 연산자 중복(operator overloading) 기능을 이용하여 기본 골격을 유지한 상태로 쉽게 삼상조류계산으로 확장할 수 있음을 실현한 것이다.

2. 본 론

2.1 전력계통 콤포넌트 모델링

객체지향기법은 객체간의 상호 관계에 의해 프로그램이 수행되므로 객체 모델링이 선행된다. 전력계통에 사용되는 콤포넌트를 모델링하는 것은 시스템 해석 프로그램에서 뿐만 아니라, 객체지향 데이터베이스, 그래픽 사용자 인퍼테이스와 기본 클래스 모델 부분을 공유할 수 있다. 하지만 여기에서 다루고자 하는 것은 기본 조류계산에 국한된 것이므로 조류계산에 사용되는 기본 콤포넌트만을 간단히 정의하면 그림 1과 같이 각 클래스간 상속도를 그릴 수 있다. 여기서 cComponet 클래스는 베이스 클래스로 다른 클래스들이 모두 이 클래스로부터 파생되어진다.

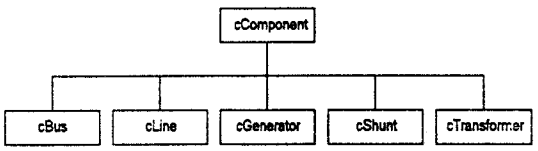

그림 1. 클래스 상속도

일반적으로 조류계산은 3상을 평형으로 간주하고 한상에 대해서만 해석 한다. 하지만 삼상조류계산은 3상을 모두 동시에 해석해야 하므로 데이터 모델링에서 부하와 전력 공급량, 선로 임피던스등 데이터 타입이 다양하다.

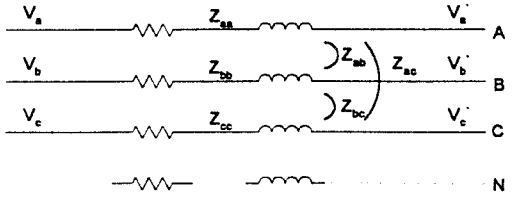

그림 2. 3상 선로 모델

부하와 전력 공급량은 3상의 값을 모두 한번에 처리할 수 있는 데이터 타입으로 사용하여야 하며, 선로 임피던스는 그림 2에 나타나 있는 각 상별 자체 임피던스와 상호 임피던스를 고려해 주어야 한다.

그림 3의 cBus 클래스는 모선의 이름과 각 상의 부하, 모선 종류가 멤버 데이터로 정의된다. 그림 4의 cLine 클래스는 연결 모선 번호와 선로의 임피던스가 3×3행렬로 구성되는 선로 임피던스, 그리고 각상별 충전용량값으로 정의된다. 또 cShunt, cGenerator, cTransformer도 삼상의 데이터를 각각 가지도록 벡터 또는 행렬을 이용하여 정의한다.

```
template <class T>
class cBus : public cComponent {
  private:
    cString name;
    Vector<T> power(3);
    Type bus_type;
  public:
        ⋮
};
```

그림 3. cBus 클래스

```
template <class T>
class cLine : public cComponent {
  private:
    int from_bus, to_bus;
    Matrix<T> impedance(3, 3);
    Vector<T> halflinecharge(3);
  public:
        ⋮
};
```

그림 4. cLine 클래스

전력계통 콤포넌트를 정의한 클래스에서 특별한 점은 다음에 설명하는 벡터와 행렬을 같은 타입을 적용시켰다는 것이다.

2.2 수학 라이브러리

C++는 다형성(polymorphism)의 특성인 연산자 중복 기능과 템플릿 기능을 이용하여 수치해석 분야에서 활발히 적용되고 있다. 대부분 Fortran으로 작성되었던 수학 라이브러리가 C++를 이용하여 재작성되고 있는데 이것은 템플릿이 가지는 데이터 추상화 기능으로, 검증된 알고리즘을 하나의 추상 타입에 대해서만 구현하면 어떤 데이터 타입에서도 적용이 가능한 것과, 연산자 중복기능을 사용하면 수식 표현과 같은 형태로 프로그래밍 이 가능한 이점이 높이 평가받기 때문이다.

2.2.1 템플릿

템플릿은 객체지향언어중 C++만이 가지는 기능으로 그림 5, 6를 통해 이해할 수 있다.

그림 5과 6에서는 벡터와 행렬을 정의하여 그림 5의 dim은 벡터 크기를 뜻하고 그림 6의 row_dim과 col_dim은 행과 열의 크기를 뜻한다. 하지만 기본 데이터 타입이 무엇인지는 정의하지 않고 T라는 이름을 대신 사용하였다. 기존 프로그램에서 int형의 벡터에 대한 라이브러리를 만들 경우 double의 벡터에 적용시키고자 할 때, 해당 소스를 모두 변경해야 하지만 템플릿을 사용하면 그럴 필요가 없다.

```
template <class T>
class Vector {
  private:
    int dim;
    T *Value;
  public:
        ⋮
};
```

그림 5. 벡터 클래스

```
template <class T>
class Matrix {
  private:
    int row_dim, col_dim;
    T **Value;
  public:
        ⋮
};
```

그림 6. 행렬 클래스

2.2.2 연산자 중복

연산자 중복기능은 클래스를 이용하여 작성한 임의 데이터 타입을 자주 사용하는 연산자에 대해 정의한 것이다. Double형의 실수부와 허수부 값을 가지는 복소수 클래스를 그림 7과 같이 정의한다.

```
class complex {
  private:
    double re, im;
  public:
        ⋮
};
```

그림 7. 복소수(complex) 클래스

그림 8에서 행렬과 벡터의 연산자를 정의하고 그림 9에서는 앞에 선언한 Vector와 Matrix를 이용하여 vec1, vec2, mat라는 변수를 정의하였다. Vector<complex>라는 의미는 T 자리에 complex를 대체하겠다는 의미로 사용한 것이며 그림 9에 있는 프로그램은 복소수 형의 벡터와 행렬간의 곱셈 연산을 수행하여 그 결과를 vec2에 저장한다.

```
template <class T>
Vector<T> operator*(const Matrix<T> &mat, const Vector<T>
&vec) {
  Vector<T> ret;
        ⋮
  return ret;
}
```

그림 8. 행렬과 벡터간의 * 연산자

```
void main()
{
        ⋮
  Vector<complex> vec1, vec2;
  Matrix<complex> mat;
        ⋮
  vec2 = mat * vec1;
        ⋮
}
```

그림 9. 행렬과 벡터간의 연산 예제

벡터와 행렬의 확장으로 대각 행렬과 희소행렬에 대한 클래스도 정의하였다. 대형 문제의 프로그래

밍에서 희소행렬은 자주 나타나는 형태이다. 이를 클래스 라이브러리로 정의해 놓음으로써 희소행렬 문제를 프로그래밍 할 경우 희소행렬 구조를 이해할 필요가 없다.

2.3 조류계산

조류계산의 대표적인 방법은 가우스-자이델(Gauss-Seidel)법과 뉴톤-랍손(Newton-Raphson)법이 있다. 본 논문에서는 두 방법을 모두 구현하였으며 다음의 결과 수식을 프로그래밍에 적용하였다.

가우스-자이델

$$\dot{S}^{*} = [\dot{V}^{*}]\dot{I}$$

$$\dot{S}^{*}\,[\dot{V}^{*}]^{-1} = [\dot{Y}]\dot{V}$$

$$= ([\dot{Y}] - [\dot{D}])\dot{V} + [\dot{D}]\dot{V}$$

(단, $[\dot{D}]$는 $[\dot{Y}]$의 대각요소로 구성된 대각행렬)

$$\therefore \dot{V}_{new} = [\dot{D}]^{-1}\{\dot{S}^{*}[\dot{V}^{*}_{old}]^{-1} - ([\dot{Y}] - [\dot{D}])\dot{V}_{old}\}$$

뉴톤-랍손

$$f(x) = S^{calc} - S^{spec} = 0$$

$$T(x) = f_{x0} + [J]\Delta x \fallingdotseq 0$$

$$\therefore \Delta x_{new} = -[J]^{-1} f_{x_{old}}$$

그림 10은 조류계산 계산과정을 나타낸 것이다.

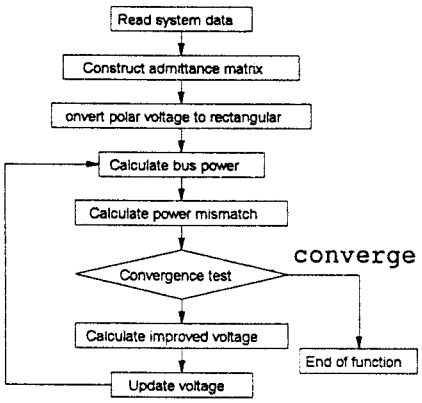

그림 10. 조류계산 흐름도

2.4 사례연구

사례연구에서 사용한 모델 계통은 5개의 노드로 구성되었으며 각 선로 임피던스와 노드선별 부하가 표1, 표2에 나타나 있다.

표 1. 모델 계통의 구성 자료

상 \ 노드	1	2	3	4	5
A	1.2+j0.35	1.7+j1.1	0.65+j0.1	1.6+j0.5	1.5+j0.3
B	0.8+j0.3	2.2+j0.55	2.35-j0.6		
C	0.9-j0.45	2.1+j0.55			

표 2. 모델 계통의 선로 임피던스

선로	임피던스		
999 - 1	0.0020+j0.0044	0.0006+j0.0092	0.0006+j0.0092
	0.0006+j0.0092	0.0020+j0.0044	0.0006+j0.0092
	0.0006+j0.0092	0.0006+j0.0092	0.0020+j0.0044
1-2	0.0040+j0.0088	0.0012+j0.0184	0.0012+j0.0184
	0.0012+j0.0184	0.0040+j0.0088	0.0012+j0.0184
	0.0012+j0.0184	0.0012+j0.0184	0.0040+j0.0088
2-3	0.0030+j0.0066	0.0009+j0.0138	
	0.0009+j0.0138	0.0030+j0.0066	
3-4	0.0034+j0.0048		
1-5	0.0017+j0.0024		

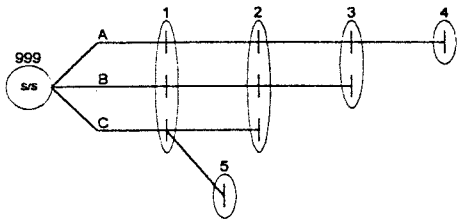

그림 11. 배전선로 모델 구성 계통도

표 3. 조류계산 결과

모선	전압		
999	1.0000+j0.0000	1.0000+j2.0944	1.0000+j4.1888
1	0.9788-j0.0189	0.9810+j2.0733	0.9846+j4.1713
2	0.9451-j0.0492	0.9499+j2.0353	0.9708+j4.1543
3	0.9334-j0.0640	0.9381+j2.0199	
4	0.9249-j0.0709		
5			0.9813+j4.1681

3. 결 론

이상과 같이 삼상조류계산에 대해 고찰하였는데 객체지향기법을 적용했을 경우 단상 조류계산과 삼상조류계산의 알고리즘상의 차이는 전혀 없다. 즉 삼상조류계산을 모두 단상처리로 병렬계산하는 것이므로 단상조류계산과 같다. 그러나 기본 데이터 타입이 변경되어야 하는 점이 요구된다. 한편 객체지향기법을 적용하면 그 재사용성이 크게 뛰어남을 알 수 있고 이를 적용함으로써 보다 간편하고 손쉽게 조류계산 프로그램이 완성됨을 알 수 있다.

[참 고 문 헌]

[1] 김건중외, "객체지향 프로그램의 조류계산 적용", '94 대한전기학회 하계학술대회, B권, p915-p917, 1994. 7.

[2] Richard Barrett, et al, "Templates", SIAM

[3] Stroustrup, "The C++ Programming Language : Second Edition", Addison-Wesley