電力系統の潮流多根に関する研究

—— 1．電力潮流計算の計算機プログラム ——

佐藤　弘之*

1．まえがき

電力系統は、エネルギーの供給システムとして社会的にきわめて重要なシステムである。また、電力系統は大規模かつ複雑で、絶えず成長を続けている。しかもこのような電力系統を運営している電気事業者は、需要家に常に良質で安定した電力をできるだけ安価に供給しなければならない義務がある。そのため発電所や送電線、変電所、配電線などの諸設備を有効に活用してゆくことが必要である。

電力系統は、数多くの発電所、送電線、変電所、配電線及び需要家から構成され、電気エネルギーの生産、輸送、分配を受け持つシステムである。そして、電力の潮流計算とは、電気エネルギーがどのような状態で電力系統の中を流れるかを調べるための計算である。

一般に、発電機や送電線、変圧器などの系統を構成する設備の電気的特性とその運用状態及び各設備の相互間の接続状態が与えられると、系統の電圧の大きさや電力の入出状態に関してエネルギー保存則を満たす条件式を記述することができる。この条件式は一般に多元連立非線形方程式の形で表される。したがって潮流計算とは、与えられた系統条件から数式モデルとなる基礎方程式を導き、それを電子計算機により数値的に解析して系統の電圧や電力の分布を求める問題である。

潮流多根は電力系統の電圧安定性と深く関係した状態量を与える。軽負荷時にはかなり多数の潮流の根が存在することが示されている。しかし、重負荷時には一対の2根のみが存在することが知られている。そして、電圧ベクトル空間内における多根間の距離が電力系統の電圧安定性の判定指標や、系統監視制御にとっても重要なパラメータを与える。潮流計算の基礎方程式系は電圧ベクトル空間内に張る二次超曲面群の相互関係、構造変化、限界潮流状態の推定、等数多くの問題解析においてもよく使用される。また、電力系統工学や数値計算法、等に関する教育用プログラムとしてもよく利用されている。

本論文は、電力多根、電力系統監視制御、安定度計算、系統計画、等の初期状態のモデル解析用として開発した計算機プログラムについて報告する。

2．電力潮流計算の数式モデル

母線数n+1の電力系統における電力潮流計算の基礎方程式は、電気回路網の節点解析法により、各母線入力電力の計算式をを導くことができる（付録 1.参照)。変形して電力潮流計算の数式モデルは次式のように、各母線の入出力電力と母線電圧分布に関する不整合関数の形に書き表すことができる。各母線における数式モデルは2個の数式と電圧の大き

*理工学部電気工学科　教授

さと位相角、または複素表示の実数部と虚数部、および有効電力と無効電力の4種類の変数により表現されている。したがって、4変数のうち2変数は電力系統の運用条件により指定して、残りの2個について解けばよい。

$$\phi_i=e_i\sum_{j=1}^{n}(G_{ij}e_j-B_{ij}f_j)+f_i\sum_{j=1}^{n}(B_{ij}e_j+G_{ij}f_j)+e_i\bar{a}_{i0}+f_i\bar{b}_{i0}-P_i \tag{1}$$

$$\psi_i=e_i\sum_{j=1}^{n}(B_{ij}e_i+G_{ij}f_j)-f_i\sum_{j=1}^{n}(G_{ij}e_j-B_{ij}f_j)+e_i\bar{b}_{i0}-f_i\bar{a}_{i0}+Q_i \tag{2}$$

または、電圧の大きさを一定に維持するために式（2）に変えて次式を導入する。

$$\psi_i=e_i^2+f_i^2-v_i^2 \tag{3}$$

ただし、

$$\bar{a}_{i0}=G_{i0}e_0-B_{i0}f_0$$

$$\bar{b}_{i0}=B_{i0}e_0+G_{i0}f_0$$

節点法による電力潮流計算では、少なくとも1個の母線を基準母線として電圧の大きさならびに位相角、または実数部と虚数部を指定しなければならない。この母線をスイング母線、またはスラック母線と呼ぶ。電力潮流の計算式の展開の際、スイング母線の母線番号を0とすると便利である。このとき、スイング母線の電圧 $\dot{E}_0=e_0+jf_0$ は定数とする。

一般に、発電機母線では有効電力と母線電圧の大きさを指定する。そして、無効電力と電圧の位相角を求める。負荷母線では有効電力と無効電力を指定し、母線電圧の大きさと位相角を求める。発電機も、負荷も接続されていない中間母線は入出力が零の負荷母線と考える。

スィング母線の式を考慮せずに、発電機母線では式（1）と（3）、そして負荷母線では式（1）と（2）を選択して、不整合関数を連立する。電力潮流計算の解は連立不整合関数系の零点に対応する複素電圧ベクトルを求めることである。したがって、電力潮流計算の数式モデルは2n元連立二次方程式により表される。

3．二次形式表現

計算機プログラムの開発には電力系統的表現より二次形式表現の方が便利である。まず、複素電圧ベクトル$\dot{\mathbf{E}}$を状態変数ベクトル$\mathbf{X}$に変換する。

$$\begin{aligned}\dot{\mathbf{E}}&=(e_1, f_1, e_2, f_2, \cdots, e_n, f_n)^T\\&=(X_1, X_2, X_3, X_4, \cdots, X_{2n-1}, X_{2n})^T=\mathbf{X}\end{aligned} \tag{4}$$

番号kの母線不整合関数は次の二次形式に書き変えることができる。

$$\phi_k=\sum_{i=1}^{n}\sum_{j=1}^{n}a_{i,j}^{(2k-1)}x_ix_j+2\sum_{i=1}^{n}b_i^{(2k-1)}x_i+c^{(2k-1)} \tag{5}$$

$$\psi_k=\sum_{i=1}^{n}\sum_{j=1}^{n}a_{i,j}^{(2k)}x_ix_j+2\sum_{i=1}^{n}b_i^{(2k)}x_i+c^{(2k)} \tag{6}$$

各母線における不整合関数の実数部ならびに虚数部に対応する式 (5) と (6) の係数$a_{i,j}$、b_iおよびc_kの母線アドミタンス行列要素からの誘導法を付録 2に示す。

実数部と虚数部の各式は全く同じ形である。したがって、m＝2nとすれば、mxm行列とm次元ベクトルにより、次式のように書くことができる。

$$F_k=\mathbf{x}^TA_k\mathbf{x}+2b_k^T\mathbf{x}+c_k \tag{7}$$

またはm＋1次元の行列とベクトルにより、次式のようにも書ける。

$$F_k=\tilde{\mathbf{x}}^T\tilde{A}_k\tilde{\mathbf{x}} \tag{8}$$

ただし、

$$\tilde{A}_k=\begin{bmatrix} A_k & b_k \\ b_k^T & c_k \end{bmatrix} \tag{9}$$

$$\tilde{\mathbf{x}}=\begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \tag{10}$$

付録 2に示した係数の誘導法から分かるように、各母線における二次形式の係数行列$\tilde{A}$の部分行列A_kとb_kにはきわめて特徴的な形をしていることが分かる。母線番号に関する 2行と 2列の中にのみ線路が接続に関係した要素が存在するのみである。

4.数値解析手法

電力潮流の計算モデルは多元連立二次方程式により表すことができた。多元連立非線形方程式の 1組の根$\mathbf{x}=(x_1,x_2,\cdots,x_m)^T$の解法にはニュートン・ラフソン法を適用する。多元連立非線形方程式としての不整合関数を$\mathbf{F}(\mathbf{x})$として、根の近傍に選ばれた近似値$\mathbf{x}^{(0)}$においてテーラー展開して、2次以上の高次項を打ち切り、零と置けば次式が得られる。

$$\mathbf{F}(\mathbf{x})=\mathbf{F}(\mathbf{x}^{(0)})+J(\mathbf{x}^{(0)})\Delta\mathbf{x}=0 \tag{11}$$

ただし、式 (11) においてベクトル$\mathbf{F}(\mathbf{x})$および$\Delta\mathbf{x}$は次式の形式である

$$\mathbf{F}(\mathbf{x})=(F_1(\mathbf{x}),F_2(\mathbf{x}),\cdots,F_n(\mathbf{x}))^T \tag{12}$$

$$\Delta\mathbf{x}=\mathbf{x}-\mathbf{x}^{(0)}=(\Delta x_1,\Delta x_2,\cdots,\Delta x_n)^T \tag{13}$$

式 (11) において、ヤコビアン行列J(x)は式 (14) のような形をしているが、不整合関数を 2次形式表現したことによりきわめて簡単に導くことができる。解析対象となる電力

系統の規模が大きくなるにつれてヤコビアン行列$\mathbf{J}(\mathbf{x})$に含まれる全要素数に対する非零要素数の比は小さくなる傾向がある。

$$J=\begin{pmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} & \cdots & \frac{\partial F_1}{\partial x_n} \\ \frac{\partial F_2}{\partial x_1} & \frac{\partial F_2}{\partial x_2} & \cdots & \frac{\partial F_2}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial F_n}{\partial x_1} & \frac{\partial F_n}{\partial x_2} & \cdots & \frac{\partial F_n}{\partial x_n} \end{pmatrix} \tag{14}$$

逆行列$\mathbf{J}^{-1}(\mathbf{x})$またはベクトル$\mathbf{J}^{-1}(\mathbf{x})\mathbf{F}(\mathbf{x})$を求めることができれば、式(11)から$\Delta\mathbf{x}$は式(15)のように計算することができる。

$$\Delta\mathbf{x}=-\mathbf{J}^{-1}(\mathbf{x}^{(0)})\mathbf{F}(\mathbf{x}^{(0)}) \tag{15}$$

したがって、根の近似値は次式のようにして求められる。

$$\mathbf{x}=\mathbf{x}^{(0)}+\Delta\mathbf{x} \tag{16}$$

高次項の打ち切り誤差が無視できないときには、最新の状態ベクトルを推定値としてさらに同様の計算を行う。したがって、計算の出発点においてn＝0とし初期推定値を与え、次式の計算を繰り返す毎に関数値または変位量などから収束しているか否かを調べ、収束していなければn＋1→nに置き換え、次式の計算操作を反復すればよい。

$$\mathbf{x}^{(n+1)}=\mathbf{x}^{(n)}-\mathbf{J}^{-1}(\mathbf{x}^{(n)})\mathbf{F}(\mathbf{x}^{(n)}) \tag{17}$$

4 プログラム作成の基本方針

4.1 プログラミング言語

将来の教育研究用プログラムの開発動向、ならびに本文に提案したプログラムの応用を考えたときC言語を使用するのが最良の選択と考えれる。言語仕様としてANSI Cに準拠していること、旧型PCのMS-DOS上でも動作し、そしてWindows 3.1ならびにWindows 95上のC＋＋およびUNIX上のCならびにC＋＋にも容易に移植可能なことから、Turbo C ver 2を使用することにした。

4.2 行列要素のスパース記憶方式

スパース行列は零要素数に対し非零要素数の少ない行列である。非零要素のみを高速メモリー内に記憶し、メモリーを有効利用することにより解析モデル規模の拡大をはかることができる。一般に行列要素を識別するには行および列番号を指定したとき行列要素の数値を知ることができればよい。その一つの方法として行番号インデックス、列番号データ、行列要素の数値データを使用し、次のような方法がある。

潮流計算におけるスパース行列は母線アドミタンス行列およびヤコビの行列がある。前者は複素数対称行列そして後者は実数非対称行列である。

いずれの行列においても、行番号インデックスデータに対象としている行中に含まれる非零要素数、行番号インデックスで指定された範囲内にある列番号データ、その位置番号と同一番号に位置する行列要素データから構成する。

アドミタンス行列要素$y_{i,j}$=g+jbは次のようにして参照することができる。

第i番目の行に含まれる非零要素はindex[i-1] からindex[i] 未満の位置に存在する。

行番号インデックスデータindex [i] において、次式により第i行要素存在範囲を求める。

```
index[i-1]≦k<index[i]
```

この範囲の中から第j列要素の記憶位置を行要素データrow[k]の中から探し、列番号jと一致する位置kを求める。

```
j=row[k]
```

アドミタンス行列要素データ $y_{i,j}$=g+jb に対応して次のように得られる。

```
g=gb_sp[loc2(k, 0)]
b=gb_sp[loc2(k, 1)]
```

ヤコビ行列の場合もアドミタンス行列の場合と同様にして行列要素を得ることができる。しかし、ヤコビアン行列は実数非対称行列であるので次のように三角化されていない一要素を求める。

```
a=ana[k]
```

4.3 配列要素コーディング法

C言語において、多数の関数プログラムから構成されるプログラムシステムを作成するとき、二次元以上の配列を使用したプログラムをコード化しない方が便利なことが多い。全ての配列を一次元化する。しかし、いろいろな局面で二次元配列を必要とすることがあるが、その場合には一次元配列を利用し、その配列内の場所を計算する関数を利用し、間接的に二次元配列とすれば問題は解決される。

4.4 プログラムとモデル規模

対象モデル規模が大きくなって、プログラムのデータサイズが不足したとき、現状では次のように定義されているファイルNetSize.h中に定義されている母線数パラメータ NN およびブランチ数 LL の数値をメモリーサイズの許容する範囲で調整すればよい。

```
/*      Sparse Matrix Programming Technique/Network Analyzer    */
#define         NN      32
#define         LL      50
#define         L2      2*LL
#define         L4      4*LL
#define         N2      2*NN
#define         N3      3*NN
```

```
#define        N4      4*NN
#define        NGB     NN+LL
#define        NGW     NGB+NGB
#define        NSP     N2*N2/4
```

4.5 途中経過のモニタリング

各種の途中経過を観察できるように行列やベクトルのCRTディスプレイ上への出力が簡単にできるよう配慮した。パラメータの変更により簡単に切り替えることができる。

4.6 ファイル　PwrFlow.C

メインプログラムのファイルで、次のアルゴリズムを連続的に実行させる。

(1) データ入力
(2) 母線アドミタンス行列
(3) 母線電圧の初期推定値の設定
(4) 等価電力源
(5) 母線電圧ベクトルから電圧状態変数ベクトルへの変換
(6) ニュートンラフソン法
(7) 状態変数ベクトルから母線電圧ベクトルへの変換
(8) 発電機母線の無効電力の計算
(9) スイング母線の発電力の有効電力と無効電力の計算
(10) 計算結果の出力

4.7 ファイル　LdFlow.C

電力潮流計算に含まれる各種の計算並びにデータ処理関係の関数サブプログラムから構成されているファイルである。

関数　loc2
計算式 2*(i-1)+j-1 により、(nx2) 二次元配列の要素位置を求めるときに使用する。

関数　loc4
計算式 4*(i-1)+j-1 により、(nx4) 二次元配列の要素位置を求めるときに使用する。

関数　gb_mat
アドミタンス行列要素の発生関数。
対角線を含めた右上三角複素数行列をスパース形式でメモリー上に配置されているデータから指定された行番号と列番号から母線アドミタンス行列の要素を発生する関数。

関数　ysp_disp
スパース形式で記憶されている母線アドミタンス行列を通常形式に変換してCRTディスプレイへ出力する関数。

関数　asp

スパース形式表示された行列Ãから指定された行列要素を発生する関数。

関数　vec_disp

指定されたベクトルxをCRTディスプレイへに出力する関数。

関数　z_input

(a)　ベース容量　スィング母線番号

(b)　線路データ

π型等価回路により定式化するため、線路データの形式は枝要素の両端の母線番号iとj、直列インピーダンスの抵抗分rとリアクタンス分x、並列アドミタンスのコンダクタンス分は無視，サセプタンス分bの1/2をyとおいて、次のようにデータを並べる。

i　j　　r　　x　　y

線路データは母線番号iを0とした終了マークが現れるまで読み込みを続ける。終了時までに読み込まれたデータ数を線路数として確定する。そして、線路データ読み込み中に母線番号の最大値を見いだし母線数とする。

(c)　母線データ

全母線を負荷母線と見なし、母線種別コードを2、電圧の大きさを1.0、発電機および負荷の有効電力と無効電力を0にする。

上記の設定と異なる場合には下記のようなデータを用意する。

母線番号　i

母線種別　0:スィング母線　1:発電機母線　2:負荷母線

電圧指定値　V_i

発電機の有効電力と無効電力　P_G Q_G

負荷の有効電力と無効電力　　P_L Q_L

(d)　スィング母線の設定

(e)　変圧器の非公称電圧比データ

線路番号　非公称電圧比

(f)　チェック用データの出力

関数　y_sparse

線路データから母線アドミタンス行列を作りスパース行列形式でメモリーに保存する。母線アドミタンス行列は対称複素数行列であるので右上三角部分のみをスパース形式で保存する。

関数　v_guess

母線電圧の初期推定値の設定。

関数　pq_equ
　母線入力電力の設定
　　$P+jQ=(P_G+jQ_G)-(P_L+jQ_L)$

関数　pq gen
　発電機出力電力の分離
　　$P_G+jQ_G=(P+jQ)-(P_L+jQ_L)$

関数　quad_sp
　アドミタンス行列から指定された母線の実数部または虚数部のスパース形式表示された行列$\tilde{A}$の発生関数。

関数　psi_xy
　スパース形式表示された行列$\tilde{A}$から二次形式$\tilde{x}^T\tilde{A}\tilde{y}$の値を計算する関数。

関数　jacobi_sp
行列Jの発生関数

関数　jac_mat
行列J要素の抽出関数

関数　jac_disp
行列JのCRTディスプレイへの出力関数。

関数　gauss
スパース行列表示された連立一次方程式のガウス消去法による計算関数。
連立一次方程式 $\mathbf{Ax=b}$ において、係数行列 A を下三角行列 L 上三角行列 U の積で表せるものとすれば、$\mathbf{A=LU}$ を連立一次方程式に代入して次式を得る。

$$LUx=b$$

上式に対して消去計算を実行すれば次式のように変形される。

$$Ux=L^{-1}b=y$$

上式に対して逆進代入を行い連立一次方程式の解を得ることができる。連立一次方程式において定数項 **b** を変えて何回も計算をしないならば、L^{-1}を陽形式に求める必要はない。

関数　mismatch
不整合関数値を計算する関数

関数　newton
ニュートンラフソン法による多元連立非線形方程式 (11) ～(17) にしたがって状態変数ベクトル$\mathbf{x}^{(k+1)}$を計算する関数。

$$x^{(k+1)}=x^{(k)}-J^{-1}F(x^{(k)})$$

関数　i_branch
電圧変数ベクトルと線路データから枝路電流を計算する関数

関数　g_mvar
発電機無効電力の計算関数

関数　g_slack
スィング発電機の有効電力と無効電力の計算関数

関数　pwr_flow
電力潮流計算結果のCRTディスプレイへの出力関数

関数　eftox
電圧ベクトルから状態変数ベクトルへの変換関数

関数　xtoef
状態変数ベクトルから電圧ベクトルへの変換関数

5. 計算機プログラム

本論文において提案する電力潮流計算に必要な全ての計算機プログラムおよびサンプル系統の入力データを下記にしめす。計算機プログラムは二つのヘッダーファイル、二つのプログラムファイルから構成されている。これを Turbo C ver.2 のエディター画面からコンパイル、リンク、計算実行の連続操作ため一つのプロジェクトファイルを用意した。

5.1 プロジェクトファイル PwrFlow.prj

```
PwrFlow
LdFlow_M
```

5.2 ヘッダーファイル NetSize.h

```
/*      Sparse Matrix Programming Technique/Network Analyzer      */
#define         NN      32
#define         LL      50
#define         L2      2*LL
#define         L4      4*LL
#define         N2      2*NN
#define         N3      3*NN
#define         N4      4*NN
#define         NGB     NN+LL
#define         NGW     NGB+NGB
#define         NSP     N2*N2/4
```

5.3 ヘッダーファイル LdFlow.h

```
/*      Sparse Matrix Programming Technique/Network Analyzer        */
int     loc2(int , int );
int     loc4(int , int );
double  sqr(double );
void    gb_mat(double [], int , int , int [], int [], double []);
void    ysp_disp(int [], int [], double [], int , char []);
double  asp(int , int , double [], int , int );
void    sp_disp(double [], int , int , char []);
void    vec_disp(double [], int , char []);
double  z_input(int [], int [], double [], int [], double [],
                double [], double [], double [], double [],
                char [], FILE *, int );
void    y_sparse(int [], double [], int [], int [],
                double [], int , int , int );
void    v_guess(double [], double [], double [], int , int , int );
void    pq_equ(double [], double [], double [], double [],
                double [], double [], int );
void    pq_gen(double [], double [], double [], double [],
                double [], double [], int );
void    quad_sp(double [], double [], double [], double [], double [],
                double [], int [], int [], double [],
                int , int , int , int , int );
double  psi_xy(double [], double [], double [], int , int );
void    jacobi_sp(int [], int [], double [], double [],
                        double [], int , int , int );
double  jac_mat(int , int , int [], int [], double []);
void    jac_disp(int [], int [], double [], int , char []);
void    rowxchg(int [], int [], int [], int , int );
double  lmat(int , int , int [], int [], double []);
double  umat(int , int , int [], int [], double []);
void    lmat_disp(int [], int [], double [], int , char []);
void    umat_disp(int [], int [], double [], int , int , char []);
void    mismatch(double [], double [], double [], double [], double [],
                double [], double [], int [], double *, int [], int [],
                double [], double [], int , int , int , int );
void    newton_sp(double [], double [], double [], double [], double [],
                double [], double [], int [], int [], int [], double [],
                double [], int , int , int , int *);
void    i_branch(double [], double [], double [],
                int [], double [], int , int );
void    g_mvar(double [], double [], double [], double [], int [],
                int [], int [], double [], int );
void    g_slack(double [], double [], double [], double [],
                int [], int [], double [], int , int );
void    pwr_flow(double [], double [], double [],
                double [], double [], double [], int [],
                double [], char [], int , int , int );
void    eftox(double [], double [], double [], int []);
void    xtoef(double [], double [], double [], int []);
```

5.4 プログラムファイル PwrFlow.c

```
/*      Sparse Matrix Programming Technique/Network Analyzer        */
#include <stdio. h>
#include <math. h>
#include <stdlib. h>
#include <string. h>
#include "netsize. h"
#include "ldflow. h"

void    main()
{
        int     i, nr, disp;
        double  base, alpha, beta, tmp;
        static int key[40], nodes[2*LL], bus[NN], index[NN], row[NSP];
        static double       senro[4*LL], v_dat[NN], p_gen[NN], q_gen[NN],
                p_load[NN], q_load[NN], gb_sp[NSP], h[NSP], ana[NSP],
                e[NN], f[NN], p[NN], q[NN], x[N2];
        char    ch, f_name[80];
        FILE    *fpi;
        disp=2;
        for(i=0;i<40;i++) key[i]=0;
        for(i=0;i<15;i++) printf("¥n");
        strcpy(f_name, "Sample. dat");
        printf("¥n¥n Input Data File is [ %s ]. ¥n¥n", f_name);
        printf("    Is this OK ?  <Yes/No/Quit> ");
        ch=tolower(getche());printf("¥n");if(ch=='q') exit(0);
        while(ch=='n') {
                printf("¥n    Tell me the Name of your Input File .....  ");
                scanf("%s", f_name);
                printf("¥n¥n Input Data File is [ %s ]. ¥n¥n", f_name);
                printf("    Is this OK ? <Yes/No/Quit> ");
                ch=tolower(getche());printf("¥n");if(ch=='q') exit(0); }

        if((fpi=fopen(f_name, "r"))==NULL) {
                printf("¥n        File [ %s ] : can't open. ¥n", f_name);
                getch();exit(0); }

        base=z_input(key, nodes, senro, bus, v_dat,
                p_gen, q_gen, p_load, q_load, f_name, fpi, disp);
        fclose(fpi);
        key[3]=key[1]-1;key[4]=2*key[3];key[5]=key[4]+1;

        if(disp>0) {
                printf("¥n¥n      [ %s ]              ", f_name);
                printf("      Base = %5. 1lf [MVA]¥n¥n", base);
                for(i=0;i<40;i++) {
                        printf("%7d", key[i]);
                        if((i+1)%10==0) printf("¥n"); }
                if(tolower(getch())=='q') exit(0); }
```

```
disp=2;
y_sparse(nodes, senro, index, row, gb_sp, key[1], key[2], disp);
if(disp> 1) ysp_disp(index, row, gb_sp, key[1], "Matrix Y[i, j]. ");
disp=0;

v_guess(e, f, v_dat, key[1], key[0], disp);

pq_equ(p, q, p_gen, q_gen, p_load, q_load, key[1]);
eftox(e, f, x, key);

disp=1;
newton_sp(ana, h, v_dat, p, q, e, f, bus, index, row, gb_sp,
                    x, key[1], key[0], disp, &nr);
xtoef(x, e, f, key);
g_mvar(q, q_load, e, f, bus, index, row, gb_sp, key[1]);
g_slack(p, q, e, f, index, row, gb_sp, key[1], key[0]);
pq_gen(p, q, p_gen, q_gen, p_load, q_load, key[1]);
pwr_flow(p_gen, q_gen, p_load, q_load, e, f, nodes, senro,
                    f_name, key[1], key[2], nr);
```

5.5 プログラムファイル LdFlow_M.c

```
/*      Sparse Matrix Programming Technique/Network Analyzer        */
#include <stdio. h>
#include <math. h>
#include <stdlib. h>
#include <string. h>
#include "netsize. h"
#include "ldflow. h"

int     loc2(int i, int j)
{       return 2*i+j-3;         }

int     loc4(int i, int j)
{       return 4*i+j-5;         }

double  sqr(double x)
{       return x*x;             }

void    gb_mat(double gb[], int ix, int jx, int index[],
                int row[], double gb_sp[])
{
        int     i, j, k, m, n, go;

        if(ix>jx) { i=jx;j=ix; }
        else { i=ix;j=jx; }
```

```
        go=1;m=0;k=index[i-1];n=index[i]-1;
        while(go>0) if(j==row[k]) { m=1;go=0; }
                else if(k<n) k++;
                else go=0;

        if(m==0) gb[0]=gb[1]=0. 0;
        else { gb[0]=gb_sp[loc2(k, 1)];gb[1]=gb_sp[loc2(k, 2)]; }
}

void    ysp_disp(int index[], int row[], double gb_sp[],
                int n, char head[])
{
        int     i, j, low, high;
        double  gb[2], b[NN];

        low=1;
        while(low<=n) {
                high=low+5;if(high>n) high=n;
                printf("¥n¥n  %s¥n¥n", head);
                for(j=low;j<=high;j++) printf("          %2d    ", j);
                printf("¥n");
                for(i=1;i<=n;i++) {
                        printf("¥n%3d", i);
                        for(j=low;j<=high;j++) {
                                gb_mat(gb, i, j, index, row, gb_sp);
                                printf("%12. 4le", gb[0]);b[j]=gb[1]; }
                        printf("¥n    ");
                        for(j=low;j<=high;j++) printf("%12. 4le", b[j]);

                        if(i%6==0&&tolower(getch())=='q') exit(0);
                        if(i<n) printf("¥n");           }
                low=high+1;
                if(n%6!=0&&tolower(getch())=='q') exit(0);
                printf("¥n¥n"); }
}

double  asp(int i, int j, double h[], int i2, int mp)
{
        double a;

        if(i==i2) a=h[loc2(j, 2)];
        else if(i==i2-1) a=h[loc2(j, 1)];
        else if(j==i2) a=h[loc2(i, 2)];
        else if(j==i2-1) a=h[loc2(i, 1)];
        else if((i==mp)&&(j==mp)) a=h[loc2(mp+1, 1)];
        else a=0. 0;

        return a;
}
```

```
void    sp_disp(double h[], int i2, int mp, char head[])
{
        int     i, j, low, high;

        low=1;
        while(low<=mp) {
                high=low+5;if(high>mp) high=mp;
                printf("¥n¥n %s¥n¥n", head);
                for(j=low;j<=high;j++) printf("          %2d    ", j);
                printf("¥n¥n");
                for(i=1;i<=mp;i++) {
                        printf("%3d", i);
                        for(j=low;j<=high;j++) printf("%12. 4le",
                                                asp(i, j, h, i2, mp));
                        printf("¥n");
                        if((i%18==0)&&(tolower(getch())=='q')) exit(0); }
                low=high+1;
                if(tolower(getch())=='q') exit(0); }
}

void    vec_disp(double x[], int m, char head[])
{
        int     i;

        printf("¥n¥n %s¥n¥n", head);
        for(i=1;i<=m;i++) {
                printf("%12. 4le", x[i]);
                if(i!=m&&i%6==0) printf("¥n"); }
        if(tolower(getch())=='q') exit(0);
        printf("¥n");
}

double  z_input(int key[], int nodes[], double senro[],
                int bus[], double v_dat[], double p_gen[], double q_gen[],
                double p_load[], double q_load[],
                char f_name[], FILE *fpi, int disp)
{
        int     i, k, n, ll, nn, swing;
        float   base, v, pg, qg, pl, ql;

        fscanf(fpi, "%e %d", &base, &swing);key[0]=swing;
        n=nn=0;
        fscanf(fpi, "%d", &k);
        while(k!=0) {
                n++;
                if(n>=LL) {
                        printf("¥n¥n  Memory Shortage for LINES. ¥n");
                        getch();exit(0); }
                nodes[loc2(n, 1)]=k;if(nn<k) nn=k;
                fscanf(fpi, "%d", &k);nodes[loc2(n, 2)]=k;if(nn<k) nn=k;
```

```
        for(i=1;i<3;i++) { fscanf(fpi, "%e", &v);senro[loc4(n, i)]=v; }
        fscanf(fpi, "%e", &v);senro[loc4(n, 3)]=v;senro[loc4(n, 4)]=1. 0;
        fscanf(fpi, "%d", &k); }
ll=n;
key[1]=nn;key[2]=ll;

for(n=1;n<=nn;n++) {
        bus[n]=2;v_dat[n]=1. 0;
        p_gen[n]=q_gen[n]=0. 0;
        p_load[n]=q_load[n]=0. 0; }

fscanf(fpi, "%d", &n);
while(n!=0) {
        fscanf(fpi, "%d%e%e%e%e%e", &k,
                &v, &pg, &qg, &pl, &ql);
        if(n!=0) {
                bus[n]=k;
                if(v==0. 0) v=1. 0;
                v_dat[n]=v;
                p_gen[n]=pg/base;q_gen[n]=qg/base;
                p_load[n]=pl/base;q_load[n]=ql/base;
                fscanf(fpi, "%d", &n);
                } }
if(bus[swing]!=0) bus[swing]=0;

fscanf(fpi, "%d", &i);
while(i!=0) {
        fscanf(fpi, "%e", &v);senro[loc4(i, 4)]=v;
        fscanf(fpi, "%d", &i);
        }

if(disp!=0) {
        printf("¥n¥n¥n    [ %s ]     Base Cap. =%7. 1f MVA¥n",
                        f_name, base);
        printf("¥n    N[swing] = %d", key[0]);
        printf("         NN      = %d", key[1]);
        printf("         LL      = %d", key[2]);
        if(tolower(getch())=='q') exit(0);

        printf("¥n¥n¥n    n    i    j        R                L");
        printf("                B/2                t¥n¥n");
        for(n=1;n<=ll;n++) {
                printf("%4d", n);
                for(i=1;i<3;i++) printf("%4d", nodes[loc2(n, i)]);
                for(i=1;i<5;i++) printf("%13. 4le", senro[loc4(n, i)]);
                if(n%20==0&&tolower(getch())=='q') exit(0);
                if(n<ll) printf("¥n"); }
        if(ll%20!=0&&tolower(getch())=='q') exit(0);
```

```
                printf("¥n¥n¥n");
                printf("  i  bus      V          Pg          Qg");
                printf("          PL          QL¥n¥n");
                for(n=1;n<=nn;n++) {
                        printf("%4d%4d%13.4le%13.4le%13.4le%13.4le%13.4le",
                                n, bus[n], v_dat[n], p_gen[n], q_gen[n],
                                p_load[n], q_load[n]);
                        if(n%20==0&&tolower(getch())=='q') exit(0);
                        if(n<nn) printf("¥n"); }
                if(nn%20!=0&&tolower(getch())=='q') exit(0);
                printf("¥n");
        }
        return base;
}

void    y_sparse(int ijbus[], double rxb[], int index[], int row[],
                double gb_sp[], int nn, int ll, int disp)
{
        int     i, j, k, m, n, nx;
        double  r, x, y, t, d, g, b, gr[NN], br[NN];

        nx=0;
        for(i=1;i<=nn;i++) {
                index[i-1]=nx;
                for(j=1;j<=nn;j++) gr[j]=br[j]=0.0;
                for(m=1;m<=ll;m++) for(n=1;n<3;n++)
                if(i==ijbus[loc2(m, n)]) {
                        r=rxb[loc4(m, 1)];
                        x=rxb[loc4(m, 2)];
                        y=rxb[loc4(m, 3)];
                        t=rxb[loc4(m, 4)];
                        k=n%2+1;
                        j=ijbus[loc2(m, k)];

                        if((r==0.0)&&(x==0.0)) g=b=0.0;
                        else { d=sqr(r)+sqr(x);g=r/d;b=-x/d; }

                        if(j==0) { d=sqr(t);gr[i]+=g/d;br[i]+=(b+y)/d; }
                        else if(k==2) {
                                gr[j]-=g/t;br[j]-=b/t;
                                d=sqr(t);gr[i]+=g/d;br[i]+=(b+y)/d; }
                        else if(k==1) {
                                gr[j]-=g/t;br[j]-=b/t;
                                gr[i]+=g;br[i]+=b+y; } }

                for(k=i;k<=nn;k++) if((gr[k]!=0.0)||(br[k]!=0.0)) {
                        gb_sp[loc2(nx, 1)]=gr[k];
                        gb_sp[loc2(nx, 2)]=br[k];
                        row[nx]=k;nx++; } }
        index[nn]=nx;
```

```
        if(disp>0) {
        printf("¥n¥n  Sparse form of Matrix Y[i, j]. ¥n");
        for(i=1;i<=nn;i++) { printf("¥n");
                for(k=index[i-1];k<index[i];k++)
                printf("%5d%5d%5d%14. 4le%14. 4le¥n",
                        k, i, row[k], gb_sp[loc2(k, 1)], gb_sp[loc2(k, 2)]);
                if(tolower(getch())=='q') exit(0); } }
}

void    v_guess(double e[], double f[], double v[], int nn, int swing, int disp)
{
        int     i;

        for(i=1;i<=nn;i++) {
                e[i]=v[i]*0. 995;f[i]=v[i]*0. 0995; }
        e[swing]=v[swing];f[swing]=0. 0;

        if(disp>0) {
                printf("¥n¥n");for(i=0;i<9;i++) printf(" ");
                printf("Initial Guess of Nodal Voltages. ¥n");
                for(i=1;i<=nn;i++) {
                        printf("¥n%10d%15. 4le%15. 4le", i, e[i], f[i]);
                        if(i%18==0) getch(); }
                if(tolower(getch())=='q') exit(0);printf("¥n"); }
}

void    pq_equ(double p[], double q[], double p_gen[], double q_gen[],
                double p_load[], double q_load[], int nn)
{
        int     i;

        for(i=1;i<=nn;i++) {
                p[i]=p_gen[i]-p_load[i];
                q[i]=q_gen[i]-q_load[i]; }
}

void    pq_gen(double p[], double q[], double p_gen[], double q_gen[],
                double p_load[], double q_load[], int nn)
{
        int     i;

        for(i=1;i<=nn;i++) {
                p_gen[i]=p[i]+p_load[i];
                q_gen[i]=q[i]+q_load[i]; }
}

void    quad_sp(double h[], double e[], double f[], double v[], double p[],
                double q[], int index[], int row[], double gb_sp[],
                int nn, int is, int ii, int ir, int nb)
```

```
{
        int     i, j, i1, i2, j1, j2, mp;
        double  a, b, gb[2];

        mp=2*nn-1;
        for(i=1;i<=loc2(mp, 2);i++) h[i]=0. 0;

        if(ir==0) h[loc2(mp+1, 1)]=-p[ii];
        else if(nb==2) h[loc2(mp+1, 1)]=q[ii];
        else if(nb==1) h[loc2(mp+1, 1)]=-sqr(v[ii]);

        if(ii>is) i2=2*(ii-1);else i2=2*ii;i1=i2-1;
        if((ir==1)&&(nb==1)) h[loc2(i1, 1)]=h[loc2(i2, 2)]=1. 0;
        else {
                gb_mat(gb, ii, is, index, row, gb_sp);
                a=(gb[0]*e[is]-gb[1]*f[is])/2. 0;
                b=(gb[1]*e[is]+gb[0]*f[is])/2. 0;

                if(ir==0) { h[loc2(mp, 1)]=a;h[loc2(mp, 2)]=b; }
                else { h[loc2(mp, 1)]=b;h[loc2(mp, 2)]=-a; }

                for(j=1;j<=nn;j++) if(j!=is) {
                        if(j>is) j2=2*(j-1);else j2=2*j;j1=j2-1;
                        gb_mat(gb, ii, j, index, row, gb_sp);

                        if(j==ii) {
                                if(ir==0) h[loc2(j1, 1)]=h[loc2(j2, 2)]=gb[0];
                                else h[loc2(j1, 1)]=h[loc2(j2, 2)]=gb[1]; }
                        else {
                                if(ir==0) {
                                        h[loc2(j1, 1)]=
                                        h[loc2(j2, 2)]=gb[0]/2. 0;
                                        h[loc2(j2, 1)]=-gb[1]/2. 0;
                                        h[loc2(j1, 2)]=gb[1]/2. 0; }
                                else if((ir==1)&&(nb==2)) {
                                        h[loc2(j1, 1)]=
                                        h[loc2(j2, 2)]=gb[1]/2. 0;
                                        h[loc2(j2, 1)]=gb[0]/2. 0;
                                        h[loc2(j1, 2)]=-gb[0]/2. 0; } } } }
}

double  psi_xy(double h[], double x[], double y[], int i2, int mp)
{
        int     i, j, k, i0, i1;
        double  p, s;

        i1=i2-1;i0=i1-1;p=0. 0;
        if(i1>1) for(i=1;i<i1;i++) p+=x[i]*(h[loc2(i, 1)]*y[i1]
                                        +h[loc2(i, 2)]*y[i2]);
        for(k=1;k<3;k++) {
```

```
                s=0. 0;
                for(j=1;j<=mp;j++) s+=h[loc2(j, k)]*y[j];
                p+=x[i0+k]*s; }
        for(i=mp;i>i2;i--) p+=x[i]*(h[loc2(i, 1)]*y[i1]
                                        +h[loc2(i, 2)]*y[i2]);
        p+=h[loc2(mp+1, 1)];
        return p;
}

void    jacobi_sp(int ind[], int jas[], double ana[], double x[],
                        double h[], int i2, int mp, int k)
{
        int     i, j, n, i0, i1, mm;
        double  s, y[N2];

        i1=i2-1;i0=i1-1;mm=mp-1;
        if(i1>1) for(i=1;i<i1;i++) y[i]=2. 0*(h[loc2(i, 1)]*x[i1]
                                        +h[loc2(i, 2)]*x[i2]);
        for(i=1;i<3;i++) {
                s=0. 0;
                for(j=1;j<=mp;j++) s+=h[loc2(j, i)]*x[j];

                y[i0+i]=2. 0*s; }
        for(i=mp;i>i2;i--) y[i]=2. 0*(h[loc2(i, 1)]*x[i1]
                                        +h[loc2(i, 2)]*x[i2]);

        n=ind[k+1];
        for(i=mm;i>0;i--) if(y[i]!=0. 0) {
                n--;jas[n]=i;ana[n]=y[i];

        if(n<0) {
                printf("¥n MEMORY shortage occurred at jacobi sp");
                printf(",  adjust NSP.     k=%d    n=%d¥n", k, n);
                getch();exit(0); } }
        ind[k]=n;
}

double  jac_mat(int i, int j, int ind[], int jas[], double ana[])
{
        int     k, m, n, go;
        double  s;

        go=1;m=0;k=ind[i];n=ind[i+1]-1;
        while(go>0) if(j==jas[k]) { m=1;go=0; }
                else if(k<n) k++;else go=0;

        if(m==0) s=0. 0;else s=ana[k];

        return s;
}
```

```
void    jac_disp(int ind[], int jas[], double ana[], int n, char head[])
{
        int     i, j, low, high;

        low=1;
        while(low<=n) {
                high=low+5;if(high>n) high=n;
                printf("¥n¥n %s¥n¥n", head);
                for(j=low;j<=high;j++) printf("          %2d    ", j);
                printf("¥n¥n");
                for(i=1;i<=n;i++) {
                        printf("%3d", i);
                        for(j=low;j<=high;j++) printf("%12.4le",
                        jac_mat(i, j, ind, jas, ana));
                        printf("¥n");
                        if((i%18==0)&&(tolower(getch())=='q')) exit(0); }
                low=high+1;
                if(tolower(getch())=='q') exit(0); }
}

void    gauss(int jal[], int jas[], int flag[], double ana[], double x[])

{
        int     i, j, k, l, m, n;
        double  a[N2], p;

        n=flag[0]=jal[0];m=flag[1]=0;
        for(k=1;k<=n;k++) {
                for(j=1;j<=n;j++) a[j]=0.0;
                for(i=jal[k];i<jal[k+1];i++) {
                        j=jas[i];a[j]=ana[i]; }
                if(k>1) for(i=1;i<k;i++) if(a[i]!=0.0) {
                        p=a[i];
                        for(l=flag[i];l<flag[i+1];l++) {
                                j=jas[l];a[j]-=p*ana[l]; }
                        x[k]-=p*x[i]; }
                p=a[k];
                if(p==0.0) {
                        printf("¥n¥n");for(i=0;i<17;i++) printf(" ");
                        printf("  ILL Condition detected. ");
                        getch();exit(0); }
                for(j=k;j<=n;j++) a[j]=a[j]/p;
                x[k]=x[k]/p;

                for(j=k;j<=n;j++) if(a[j]!=0.0) {
                        jas[m]=j;ana[m]=a[j];m++; }
                flag[k+1]=m;

                if(flag[k+1]>jal[k+1]) {
                        printf("¥n¥n Confliction occurred ");
```

```
                printf("in Gauss Elimination. ¥n");
                printf(" k=%d  flag=%d  jal=%d",
                        k, flag[k+1], jal[k+1]);
                getch();exit(0); }
        }

    for(i=n-1;i>0;i--) {
        p=x[i];
        for(k=flag[i];k<flag[i+1];k++) {
            j=jas[k];
            if(j>i) p-=ana[k]*x[j]; }
        x[i]=p; }
}

void    mismatch(double e[], double f[], double v[], double p[], double q[],
        double x[], double dx[], int bus[], double *err,
        int index[], int gyo[], double gb_sp[], double h[],
        int mp, int nn, int swing, int disp)
{
    int     i, j, k, m, i2;
    double  tmp;

    for(i=1;i<=nn;i++) if(i!=swing) {
        if(i>swing) i2=2*(i-1);else i2=2*i;k=i2-1;
        for(j=0;j<2;j++) {
            quad_sp(h, e, f, v, p, q, index, gyo, gb_sp, nn, swing,
                    i, j, bus[i]);

            if(disp>5) {
            printf("¥n Sparse Matrix A:%d %d¥n¥n", i, j);
            for(m=1;m<=mp;m++) printf("%5d%15.4le%15.4le¥n",
                        m, h[loc2(m, 1)], h[loc2(m, 2)]);
            printf("      %15.4le¥n", h[loc2(mp+1, 1)]);
            if(tolower(getch())=='q') exit(0); }

            if(disp>5) sp_disp(h, i2, mp, " Matrix A[i, j]. ");

            tmp=psi_xy(h, x, x, i2, mp);
            dx[j+k]=tmp;
            if(*err<fabs(tmp)) *err=fabs(tmp); } }
}

void    newton_sp(double ana[], double h[], double v[], double p[], double q[],
        double e[], double f[], int bus[], int index[], int gyo[],
        double gb_sp[], double x[], int nn, int swing, int disp, int *no)
{
    int     ind[N2], jet[N2], jas[NSP], i, j, k, m, n, i2, mm, mp, nr, n_max=200;
    double  dx[N2], tmp, err, eps=1.0e-10;
    char    ch;
```

```
mm=2*(nn-1);mp=mm+1;ind[0]=mm;nr=0;

if(disp>1)
vec_disp(x, mp, " Initial State of Vector x[i] from E=e+jf");

err=0. 0;
mismatch(e, f, v, p, q, x, dx, bus, &err, index, gyo, gb_sp,
                h, mp, nn, swing, disp);
if(disp>1) vec_disp(dx, mm, " Bus Mismatch phi[i]. ");

if((disp<=1)&&(nr==0)||(disp>0)) printf("¥n");
printf(" Maximum mismatch (%2d) ->%18. 10le", nr, err);
if(disp>0&&tolower(getch())=='q') exit(0);
printf("¥n");

while((err>eps)&&(nr<n_max)) { ind[mp]=NSP;
for(i=nn;i>0;i--) if(i!=swing) {
        if(i>swing) i2=2*(i-1);else i2=2*i;k=i2-1;
        for(j=1;j>=0;j--) {
                quad_sp(h, e, f, v, p, q, index, gyo, gb_sp, nn, swing,
                        i, j, bus[i]);

                if(disp>5) {
                printf("¥n Sparse Matrix A:%d %d¥n¥n", i, j);
                for(m=1;m<=mp;m++) printf("%5d%15. 4le%15. 4le¥n",
                                m, h[loc2(m, 1)], h[loc2(m, 2)]);
                printf("      %15. 4le¥n", h[loc2(mp+1, 1)]);
                if(tolower(getch())=='q') exit(0); }

                if(disp>5) sp_disp(h, i2, mp, " Matrix A[i, j]. ");

                jacobi_sp(ind, jas, ana, x, h, i2, mp, j+k); } }

if(disp>4) {
printf("¥n¥n Sparse form of Jacobi's Matrix J[i, j]. ¥n");
for(i=1;i<mp;i++) {
        printf("¥n");m=ind[i];n=ind[i+1];
        for(j=m;j<n;j++) printf("%5d%5d%5d%12. 4le¥n",
                                j, i, jas[j], ana[j]);
        if(tolower(getch())=='q') exit(0); } }

if(disp>4)
jac_disp(ind, jas, ana, mm, " Jacobi's Matrix J[i, j]. ");
gauss(ind, jas, jet, ana, dx);nr++;
for(i=1;i<mp;i++) x[i]-=dx[i];

if(disp>1) {
        vec_disp(dx, mm, " Displacement dx[i]. ");
        vec_disp(x, mp, "Corrected x[i]. "); }
```

```
        err=0. 0;
        mismatch(e, f, v, p, q, x, dx, bus, &err, index, gyo, gb_sp,
                        h, mp, nn, swing, disp);
        if(disp>1) vec_disp(dx, mm, " Bus Mismatch phi[i]. ");

        if(disp>1) printf("¥n");
        printf(" Maximum mismatch (%2d) ->%18. 10le", nr, err);
        if(disp>0&&tolower(getch())=='q') exit(0);
        printf("¥n"); }
        *no=nr;
}

void    i_branch(double ab[], double e[], double f[],
                int ijbus[], double rxb[], int k, int m)
{
        int     i, j;
        double  r, x, d, y, t, u, v;

        i=ijbus[loc2(k, m)];
        r=rxb[loc4(k, 1)];x=rxb[loc4(k, 2)];
        y=rxb[loc4(k, 3)];t=rxb[loc4(k, 4)];
        d=sqr(r)+sqr(x);

        if(m==1) {
                j=ijbus[loc2(k, 2)];
                if(j!=0) {
                        u=e[i]/t-e[j];
                        v=f[i]/t-f[j];d=t*d;
                        ab[0]+=(u*r+v*x)/d;
                        ab[1]+=(v*r-u*x)/d; }
                ab[0]-=y*f[i]/sqr(t);
                ab[1]+=y*e[i]/sqr(t); }
        else {
                j=ijbus[loc2(k, 1)];
                u=e[i]-e[j]/t;v=f[i]-f[j]/t;
                ab[0]+=(u*r+v*x)/d-y*f[i];
                ab[1]+=(v*r-u*x)/d+y*e[i]; }
}

void    g_mvar(double q[], double q_load[], double e[], double f[],
                int bus[], int index[], int row[], double gb_sp[], int nn)
{
        int     i, j;
        double  ab[2], gb[2];

        for(i=1;i<=nn;i++) if(bus[i]==1) {
                ab[0]=ab[1]=0. 0;
                for(j=1;j<=nn;j++) {
                        gb_mat(gb, i, j, index, row, gb_sp);
                        ab[0]+=gb[0]*e[j]-gb[1]*f[j];
```

```
                        ab[1]+=gb[0]*f[j]+gb[1]*e[j]; }
                        q[i]=q_load[i]+f[i]*ab[0]-e[i]*ab[1]; }
}

void    g_slack(double p[], double q[], double e[], double f[],
                int index[], int row[], double gb_sp[], int nn, int is)
{
        int     j;
        double ab[2], gb[2];

        ab[0]=ab[1]=0. 0;
        for(j=1;j<=nn;j++) {
                gb_mat(gb, is, j, index, row, gb_sp);
                ab[0]+=gb[0]*e[j]-gb[1]*f[j];
                ab[1]+=gb[0]*f[j]+gb[1]*e[j]; }
        p[is]=e[is]*ab[0]+f[is]*ab[1];
        q[is]=f[is]*ab[0]-e[is]*ab[1];
}

void    pwr_flow(double p_gen[], double q_gen[], double p_load[],
                double q_load[], double e[], double f[], int ijbus[],
                double rxb[], char f_name[], int nn, int ll, int nr)
{

        int     i, j, k, m, n;
        double  ab[2], c, d, s, t, v, w, z, theta, rad=57. 29577951;

        printf("¥n¥n LOAD FLOW STUDY of [ %s ]. ¥n", f_name);
        printf("¥n¥n Nodal Voltages,  Power/Generations and Loads. ");
        if(nr==0) printf("¥n");
        else {  for(i=0;i<20;i++) printf(" ");
                printf(" nr = %d¥n", nr); }

        printf("¥n  n");for(i=0;i<7;i++) printf(" ");
        printf("E = e+jf");for(i=0;i<9;i++) printf(" ");
        printf("V");for(i=0;i<6;i++) printf(" ");
        printf("Theta");for(i=0;i<5;i++) printf(" ");
        printf("Pg");for(i=0;i<7;i++) printf(" ");
        printf("Qg");for(i=0;i<7;i++) printf(" ");
        printf("Pl");for(i=0;i<7;i++) printf(" ");
        printf("Ql¥n");

        j=5;
        for(n=1;n<=nn;n++) {
                v=sqrt(sqr(e[n])+sqr(f[n]));
                theta=atan2(f[n], e[n])*rad;j++;
                printf("¥n%3d%9. 3lf%9. 3lf%9. 3lf%9. 3lf", n, e[n], f[n], v, theta);
                printf("%9. 3lf%9. 3lf%9. 3lf%9. 3lf",
                        p_gen[n], q_gen[n], p_load[n], q_load[n]);
                if(j>18) {j=0;getch();} }
        if(tolower(getch())=='q') exit(0);
```

```
        printf("¥n¥n¥n Line Flow of Currents,  Powers");
        printf(" and Distribution of Line Losses. ¥n¥n¥n");
        printf("Bus to bus");for(i=0;i<7;i++) printf(" ");
        printf("I=a+jb");for(i=0;i<10;i++) printf(" ");
        printf("I");for(i=0;i<7;i++) printf(" ");
        printf("phi");for(i=0;i<4;i++) printf(" ");
        printf("P flow");for(i=0;i<4;i++) printf(" ");
        printf("Q flow");for(i=0;i<3;i++) printf(" ");
        printf("Loss¥n");

        z=0. 0;j=7;
        for(n=1;n<=nn;n++) {
                for(k=1;k<=ll;k++) for(m=1;m<3;m++)
                if(ijbus[loc2(k, m)]==n) {
                        ab[0]=ab[1]=0. 0;
                        i_branch(ab, e, f, ijbus, rxb, k, m);
                        v=sqr(ab[0])+sqr(ab[1]);
                        theta=atan2(ab[1], ab[0])*rad;
                        if(ab[0]<0. 0) s=-sqrt(v);else s=sqrt(v);
                        c=e[n]*ab[0]+f[n]*ab[1];
                        d=f[n]*ab[0]-e[n]*ab[1];
                        t=rxb[loc4(k, 4)];

                        if(m==1) {
                                i=ijbus[loc2(k, 2)];
                                if(i==0) v=0. 0;
                                else v=sqr(e[n]/t-e[i])+sqr(f[n]/t-f[i]); }
                        else {  i=ijbus[loc2(k, 1)];
                                v=sqr(e[n]-e[i]/t)+sqr(f[n]-f[i]/t); }
                        if(i==0) w=0. 0;
                        else w=rxb[loc4(k, 1)]*v/
                                (sqr(rxb[loc4(k, 1)])+sqr(rxb[loc4(k, 2)]));
                        z+=w;j++;
                        printf("¥n%3d ->%3d%9. 3lf%9. 3lf", n, i, ab[0], ab[1]);
                        printf("%9. 3lf%9. 3lf%9. 3lf%9. 3lf%9. 3lf",
                                                s, theta, c, d, w);
                        if(j>18) {j=0;getch();}
                        }
                j++;printf("¥n"); }
        w=z/2. 0;printf("¥n");
        for(i=0;i<30;i++) printf(" ");
        printf("Total of Line Losses =%10. 4lf¥n", w);
        z=0. 0;
        for(n=1;n<=nn;n++) z+=p_gen[n]-p_load[n];
        for(i=0;i<30;i++) printf(" ");
        printf("Total of Nodal Power =%10. 4lf", z);
        if(tolower(getch())=='q') exit(0);printf("¥n");
}

void    eftox(double e[], double f[], double x[], int key[])
{
```

```
        int     i, i2, mp, nn, swing;

        swing=key[0];nn=key[1];mp=key[5];
        for(i=1;i<=nn;i++) if(i!=swing) {
                if(i>swing) i2=2*(i-1);else i2=2*i;
                x[i2-1]=e[i];x[i2]=f[i]; }
        x[mp]=1.0;
}

void    xtoef(double x[], double e[], double f[], int key[])
{
        int     i, i2, nn, swing;

        swing=key[0];nn=key[1];
        for(i=1;i<=nn;i++) if(i!=swing) {
                if(i>swing) i2=2*(i-1);else i2=2*i;
                e[i]=x[i2-1];f[i]=x[i2]; }
}
```

5.6 データファイル Sample.dat

```
100.0           1
1    4   0.0       0.0576      0.0
2    7   0.0       0.0625      0.0
3    9   0.0       0.0586      0.0
4    5   0.010     0.085       0.088
4    6   0.017     0.092       0.079
5    7   0.032     0.161       0.153
6    9   0.039     0.17        0.179
7    8   0.0085    0.072       0.0745
8    9   0.0119    0.1008      0.1045
0

1    0   1.04      0.0    0.0    0.0    0.0
2    1   1.025   163.0    0.0    0.0    0.0
3    1   1.025    85.0    0.0    0.0    0.0
5    2   1.0       0.0    0.0  125.0   50.0
6    2   1.0       0.0    0.0   90.0   30.0
8    2   1.0       0.0    0.0  100.0   35.0
0

0
```

6.サンプル系統の数値計算例

入力データにも示すように9母線数、9枝路数のサンプル系統によりここに紹介する計算機プログラムによる計算例を示す。

9母線系であるので、スィング母線をのぞいた各母線における不整合関数の実数部と虚数部から16元連立非線形方程式が導かれる。

入力データファイル Sample.dat を読み込み、次に示すような確認のデータを必要に応じて CRT ディスプレイに出力する。

```
[ Sample.dat ]     Base Cap. =   100.0 MVA

N[swing] = 1       NN    = 9       LL    = 9


n   i   j       R             L             B/2           t

1   1   4   0.0000e+00   5.7600e-02   0.0000e+00   1.0000e+00
2   2   7   0.0000e+00   6.2500e-02   0.0000e+00   1.0000e+00
3   3   9   0.0000e+00   5.8600e-02   0.0000e+00   1.0000e+00
4   4   5   1.0000e-02   8.5000e-02   8.8000e-02   1.0000e+00
5   4   6   1.7000e-02   9.2000e-02   7.9000e-02   1.0000e+00
6   5   7   3.2000e-02   1.6100e-01   1.5300e-01   1.0000e+00
7   6   9   3.9000e-02   1.7000e-01   1.7900e-01   1.0000e+00
8   7   8   8.5000e-03   7.2000e-02   7.4500e-02   1.0000e+00
9   8   9   1.1900e-02   1.0080e-01   1.0450e-01   1.0000e+00

i   bus       V            Pg            Qg            PL            QL

1   0   1.0400e+00   0.0000e+00   0.0000e+00   0.0000e+00   0.0000e+00
2   1   1.0250e+00   1.6300e+00   0.0000e+00   0.0000e+00   0.0000e+00
3   1   1.0250e+00   8.5000e-01   0.0000e+00   0.0000e+00   0.0000e+00
4   2   1.0000e+00   0.0000e+00   0.0000e+00   0.0000e+00   0.0000e+00
5   2   1.0000e+00   0.0000e+00   0.0000e+00   1.2500e+00   5.0000e-01
6   2   1.0000e+00   0.0000e+00   0.0000e+00   9.0000e-01   3.0000e-01
7   2   1.0000e+00   0.0000e+00   0.0000e+00   0.0000e+00   0.0000e+00
8   2   1.0000e+00   0.0000e+00   0.0000e+00   1.0000e+00   3.5000e-01
9   2   1.0000e+00   0.0000e+00   0.0000e+00   0.0000e+00   0.0000e+00
```

データの確認後、直ちに母線アドミッタンス行列をスパース行列記憶方式に従って作る。対称行列であるので対角要素を含む右上三角部分行列だけ保存する。次に示すのは確認のために出力したスパース行列記憶方式の内容である。

```
Sparse form of Matrix Y[i,j].

  0    1    1    0.0000e+00   -1.7361e+01
  1    1    4    0.0000e+00    1.7361e+01

  2    2    2    0.0000e+00   -1.6000e+01
  3    2    7    0.0000e+00    1.6000e+01

  4    3    3    0.0000e+00   -1.7065e+01
  5    3    9    0.0000e+00    1.7065e+01

  6    4    4    3.3074e+00   -3.9309e+01
  7    4    5   -1.3652e+00    1.1604e+01
  8    4    6   -1.9422e+00    1.0511e+01
```

```
 9    5    5    2.5528e+00   -1.7338e+01
10    5    7   -1.1876e+00    5.9751e+00

11    6    6    3.2242e+00   -1.5841e+01
12    6    9   -1.2820e+00    5.5882e+00

13    7    7    2.8047e+00   -3.5446e+01
14    7    8   -1.6171e+00    1.3698e+01

15    8    8    2.7722e+00   -2.3303e+01
16    8    9   -1.1551e+00    9.7843e+00

17    9    9    2.4371e+00   -3.2154e+01
```

スパース行列記憶方式の母線アドミタンス行列では、数式的なイメージが湧きにくいので、通常の行列方式に変換して CRT ディスプレイに出力することもできる。

```
Matrix Y[i,j].

        1            2            3            4            5            6

1   0.0000e+00   0.0000e+00   0.0000e+00   0.0000e+00   0.0000e+00   0.0000e+00
   -1.7361e+01   0.0000e+00   0.0000e+00   1.7361e+01   0.0000e+00   0.0000e+00

2   0.0000e+00   0.0000e+00   0.0000e+00   0.0000e+00   0.0000e+00   0.0000e+00
    0.0000e+00  -1.6000e+01   0.0000e+00   0.0000e+00   0.0000e+00   0.0000e+00

3   0.0000e+00   0.0000e+00   0.0000e+00   0.0000e+00   0.0000e+00   0.0000e+00
    0.0000e+00   0.0000e+00  -1.7065e+01   0.0000e+00   0.0000e+00   0.0000e+00

4   0.0000e+00   0.0000e+00   0.0000e+00   3.3074e+00  -1.3652e+00  -1.9422e+00
    1.7361e+01   0.0000e+00   0.0000e+00  -3.9309e+01   1.1604e+01   1.0511e+01

5   0.0000e+00   0.0000e+00   0.0000e+00  -1.3652e+00   2.5528e+00   0.0000e+00
    0.0000e+00   0.0000e+00   0.0000e+00   1.1604e+01  -1.7338e+01   0.0000e+00

6   0.0000e+00   0.0000e+00   0.0000e+00  -1.9422e+00   0.0000e+00   3.2242e+00
    0.0000e+00   0.0000e+00   0.0000e+00   1.0511e+01   0.0000e+00  -1.5841e+01

7   0.0000e+00   0.0000e+00   0.0000e+00   0.0000e+00  -1.1876e+00   0.0000e+00
    0.0000e+00   1.6000e+01   0.0000e+00   0.0000e+00   5.9751e+00   0.0000e+00

8   0.0000e+00   0.0000e+00   0.0000e+00   0.0000e+00   0.0000e+00   0.0000e+00
    0.0000e+00   0.0000e+00   0.0000e+00   0.0000e+00   0.0000e+00   0.0000e+00

9   0.0000e+00   0.0000e+00   0.0000e+00   0.0000e+00   0.0000e+00  -1.2820e+00
    0.0000e+00   0.0000e+00   1.7065e+01   0.0000e+00   0.0000e+00   5.5882e+00
```

```
Matrix Y[i,j].

       7            8            9

1   0.0000e+00   0.0000e+00   0.0000e+00
    0.0000e+00   0.0000e+00   0.0000e+00

2   0.0000e+00   0.0000e+00   0.0000e+00
    1.6000e+01   0.0000e+00   0.0000e+00

3   0.0000e+00   0.0000e+00   0.0000e+00
    0.0000e+00   0.0000e+00   1.7065e+01

4   0.0000e+00   0.0000e+00   0.0000e+00
    0.0000e+00   0.0000e+00   0.0000e+00

5  -1.1876e+00   0.0000e+00   0.0000e+00
    5.9751e+00   0.0000e+00   0.0000e+00

6   0.0000e+00   0.0000e+00  -1.2820e+00
    0.0000e+00   0.0000e+00   5.5882e+00

7   2.8047e+00  -1.6171e+00   0.0000e+00
   -3.5446e+01   1.3698e+01   0.0000e+00

8  -1.6171e+00   2.7722e+00  -1.1551e+00
    1.3698e+01  -2.3303e+01   9.7843e+00

9   0.0000e+00  -1.1551e+00   2.4371e+00
    0.0000e+00   9.7843e+00  -3.2154e+01
```

ニュートンラフソン法による繰り返し計算中の各反復回数における状態変数ベクトルを使用して不整合関数値ベクトルを計算した。そして、その中の絶対最大値を出力した。そのの変化の様子は次のようになった。電圧ベクトルの初期推定値は関数 vguess により設定されたほぼフラット電圧出発値である。反復回数の進行に伴い不整合関数値は加速度的に低下し、急速に収束することを示している。

```
Maximum mismatch ( 0) ->  1.7965277427e+00
Maximum mismatch ( 1) ->  2.7138839152e-01
Maximum mismatch ( 2) ->  6.6863502622e-03
Maximum mismatch ( 3) ->  6.6863502622e-06
Maximum mismatch ( 4) ->  6.1215477132e-12
```

ニュートンラフソン法による反復計算の終了後系統内の各母線および線路の諸データを計算し、希望があれば次のように出力する。これで、電力系統内における、ある運転条件に対応した電圧分布が得られた。

LOAD FLOW STUDY of [Sample.dat].

Nodal Voltages, Power/Generations and Loads. nr = 4

n	E = e+jf		V	Theta	Pg	Qg	Pl	Ql
1	1.040	0.000	1.040	0.000	0.716	0.270	0.000	0.000
2	1.012	0.165	1.025	9.280	1.630	0.067	0.000	0.000
3	1.022	0.083	1.025	4.665	0.850	-0.109	0.000	0.000
4	1.025	-0.040	1.026	-2.217	0.000	0.000	0.000	0.000
5	0.993	-0.069	0.996	-3.989	0.000	0.000	1.250	0.500
6	1.011	-0.065	1.013	-3.687	0.000	0.000	0.900	0.300
7	1.024	0.067	1.026	3.720	0.000	0.000	0.000	0.000
8	1.016	0.013	1.016	0.728	0.000	0.000	1.000	0.350
9	1.032	0.035	1.032	1.967	0.000	0.000	0.000	0.000

Line Flow of Currents, Powers and Distribution of Line Losses.

Bus to bus	I=a+jb		I	phi	P flow	Q flow	Loss
1 -> 4	0.689	-0.260	0.736	-20.683	0.716	0.270	0.000
2 -> 7	1.580	0.192	1.592	6.942	1.630	0.067	0.000
3 -> 9	0.818	0.173	0.836	11.945	0.850	-0.109	0.000
4 -> 1	-0.689	0.260	-0.736	159.317	-0.716	-0.239	0.000
4 -> 5	0.390	-0.238	0.457	-31.432	0.409	0.229	0.003
4 -> 6	0.299	-0.022	0.299	-4.138	0.307	0.010	0.002
5 -> 4	-0.381	0.416	-0.564	132.449	-0.407	-0.387	0.003
5 -> 7	-0.837	0.172	-0.854	168.370	-0.843	-0.113	0.023
6 -> 4	-0.290	0.182	-0.343	147.866	-0.305	-0.165	0.002
6 -> 9	-0.577	0.170	-0.602	163.561	-0.595	-0.135	0.014
7 -> 2	-1.580	-0.192	-1.592	-173.058	-1.630	0.092	0.000
7 -> 5	0.837	0.136	0.848	9.246	0.866	-0.084	0.023
7 -> 8	0.743	0.056	0.745	4.318	0.764	-0.008	0.005
8 -> 7	-0.748	0.096	-0.755	172.701	-0.759	-0.107	0.005
8 -> 9	-0.240	0.236	-0.337	135.490	-0.241	-0.243	0.001
9 -> 3	-0.818	-0.173	-0.836	-168.055	-0.850	0.150	0.000
9 -> 6	0.583	0.195	0.615	18.519	0.608	-0.181	0.014
9 -> 8	0.235	-0.022	0.236	-5.383	0.242	0.031	0.001

Total of Line Losses = 0.0464
Total of Nodal Power = 0.0464

7. あとがき

電力潮流計算の計算機プログラムは電力系統の解析に電子計算機が利用されるようになって以来種々の方法が提案されてきた。問題の非線形性の克服法として、線形方程式の反復的解法であるガウスザイデル法の利用可能性、逐次線形化による、いわゆるダイレクト解法などが報告された。また基礎方程式の複素数表示に極座標形式を使用するもの、直角座標形式を使用するものなどの長短などが議論された。そして、電子計算機の実用当時のいわゆる大型計算機で大規模な電力系統の数値解析を実用化するためにスパース行列処理技術が提案された。

本文の著者も電子計算機の実用化の初期の時代から電力潮流計算の計算機プログラム開発を試み、FORTRAN II, FORTRAN IV, ALGOL, FORTRAN 77などのコンパイラー言語を使用して改良を重ねてきたが、最近ではさらに汎用性の高いPASCALやCなどの言語により書き換えられ言語使用の高級化に伴う計算機プログラム性能の向上をはかってきた。

本文に紹介した計算機プログラムも研究用ならびに教育用として、小規模な部分プログラムの追加や削除を行い使用し、効果的な成果を収めてきた。

文献

1) 佐藤：空間曲線追跡による潮流多根,気学会論文誌B,115巻9号,1995
2) J.B.Ward, H.W.Hale:Digital Computer Solution of Power Flow Problems, AIEE Trans. of PAS, vol.75, 1956
3) A.F.Glimn, G.W.Stagg:Automatic Calculaion of Load Flows, AIEE Trans. of PAS, vol.76, 1957
4) J.E.VanNess:Iteration Methods for Digital Load Flow Studies, AIEE Trans. of PAS, vol. 78,1959
5) J.E.VanNess and J.H.Griffin:Elimination Methods for Load Flow Studies, AIEE Trans. of PAS, vol. 80,1961
6) W.F.Tinney, C.E.Hart:Power Flow Solution by Nwton's Method, IEEE Trans. of PAS, vol. 86, 1967
7) W.F.Tinney, J.W.Walker:Direct Solutions of Network Equations by Optimally Ordered Triangular Factorization, Proc. IEEE,vol. 55, 1967

付録 1. 母線入力電力の式の誘導

母線数n+1の電力系統において、それぞれの母線に0,1,2,…,nの番号を付ける。母線iとjの間は直列インピーダンス$\dot{z}_{i,j}=r_{i,j}+jx_{i,j}$、並列アドミタンスの半分による$\pi$型等価回路で近似した送電線線路要素により接続されているものとする。母線iから母線jに向けて流れる電流は次式のように書くことができる。

$$\dot{I}_{i,j}=\frac{1}{\dot{z}_{i,j}}\left(\dot{E}_i-\dot{E}_j\right)+\dot{y}_{i,j}\dot{E}_i \qquad (A.1\text{-}1)$$

母線iに接続される全ての送電線について和をとれば、次式のように書くことができる。

$$\dot{I}_i=\sum_{j=0,j\neq i}^{n}\left(\frac{1}{\dot{z}_{i,j}}(\dot{E}_i-\dot{E}_j)+\dot{y}_{i,j}\dot{E}_j\right) \tag{A.1-2}$$

上式を整理して、次のように書き変える。

$$\dot{I}_i=\dot{E}_i\sum_{j=[,j\neq i}^{n}\left(\frac{1}{\dot{z}_{i,j}}+\dot{y}_{i,j}\right)-\sum_{j=[,j\neq i}^{n}\frac{1}{\dot{z}_{i,j}}\dot{E}_j \tag{A.1-3}$$

ここで、母線アドミタンス行列 $[\dot{Y}]$ の各要素を次のように表す。

$$\dot{Y}_{i,i}=\sum_{j=0,j\neq i}^{n}\left(\frac{1}{\dot{z}_{i,j}}+\dot{y}_{i,j}\right) \tag{A.1-4}$$

$$\dot{Y}_{i,j}=-\frac{1}{\dot{z}_{i,j}} \tag{A.1-5}$$

母線iから送電線路網に流入する電流 $[\dot{I}]$ は母線アドミッタンス行列 $[\dot{Y}]$ と電圧ベクトル $[\dot{E}]$ により次式のように表される。

$$[\dot{i}]=[\dot{Y}][\dot{E}] \tag{A.1-6}$$

または、要素ごとに次式のように表される。

$$\dot{I}_i=\sum_{j=0}^{n}\dot{Y}_{i,j}\dot{E}_j \tag{A.1-7}$$

上式において、電流、電圧ならびにアドミタンスについて次のように複素表現する。

$$\dot{I}_i=\bar{a}_i+j\bar{b}_i \tag{A.1-8}$$

$$\dot{E}_i=e_i+jf_i \tag{A.1-9}$$

$$\dot{Y}_{i,j}=G_{i,j}+jB_{i,j} \tag{A.1-10}$$

式 (A.1-8)、(A.1-9) および (A.1-10) を (A.1-7) に代入して整理すれば、電流は次式のように書くことができる。

$$\bar{a}_i+j\bar{b}_j=\sum_{j=0}^{n}(G_{i,j}+jB_{i,j})(e_j+jf_j) \tag{A.1-11}$$

または、変形して次のように書ける。

$$\bar{a}_i+j\bar{b}_i=\sum_{j=0}^{n}(G_{i,j}e_j-B_{i,j}f_j)+j\sum_{j=0}^{n}(B_{i,j}e_j+G_{i,j}f_j) \tag{A.1-12}$$

母線iの入力電力と送電線路網に流入する線路潮流との平衡条件は次式のように書くことができる。

$$P_i+jQ_i=\dot{E}_i\dot{I}_i^*=(e_i+jf_i)(\bar{a}_i+j\bar{b}_i)^*=(e_i+jf_i)(\bar{a}_i-j\bar{b}_i)$$
$$=(e_i+jf_i)\left\{\sum_{j=0}^{n}(G_{i,j}e_j-B_{i,j}f_j)-j\sum_{j=0}^{n}(B_{i,j}e_j+G_{i,j}f_j)\right\} \quad (A.1\text{-}13)$$

上式を整理して、実数部と虚数部を分離すれば、次に示す有効電力ならびに無効電力の式が得られる。

$$P_i=e_i\sum_{j=0}^{n}(G_{i,j}e_j-B_{i,j}f_j)+f_i\sum_{j=0}^{n}(B_{i,j}e_j+G_{i,j}f_j) \quad (A.1\text{-}14)$$

$$Q_i=-e_i\sum_{j=0}^{n}(B_{i,j}e_j+G_{i,j}f_j)+f_i\sum_{j=0}^{n}(G_{i,j}e_j-B_{i,j}f_j) \quad (A.1\text{-}15)$$

付録 2　二次形式の係数

上に示した二次形式 (5) および (6) の係数は、式 (1)、(2) および (3) と式 (5) および (6) を比較して次のように書くことができる。

1. 実数部 ϕ_k の係数

$$a_{2i-1,2i-1}^{(2k-1)}=a_{2i,2i}^{(2k-1)}=G_{i,i} \quad (A.2\text{-}1)$$

$$a_{2i-1,2j-1}^{(2k-1)}=a_{2j-1,2i-1}^{(2k-1)}=G_{i,j}/2 \quad (A.2\text{-}2)$$

$$a_{2i-1,2j}^{(2k-1)}=a_{2j,2i-1}^{(2k-1)}=-B_{i,j}/2 \quad (A.2\text{-}3)$$

$$a_{2i,2j-1}^{(2k-1)}=a_{2j-1,2i}^{(2k-1)}=B_{i,j}/2 \quad (A.2\text{-}4)$$

$$a_{2i,2j}^{(2k-1)}=a_{2j,2i}^{(2k-1)}=G_{i,j}/2 \quad (A.2\text{-}5)$$

$$b_{2i-1}^{(2k-1)}=\bar{a}_{i0}/2 \quad (A.2\text{-}6)$$

$$b_{2i}^{(2k-1)}=\bar{b}_{i0}/2 \quad (A.2\text{-}7)$$

$$c^{(2k-1)}=-P_i \quad (A.2\text{-}8)$$

2. 虚数部 ψ_k の係数

(a) 負荷母線

$$a^{(2k)}_{2i-1,2i-1}=a^{(2k)}_{2i,2i}=B_{i,i} \tag{A. 2-9}$$

$$a^{(2k)}_{2i-1,2j-1}=a^{(2k)}_{2j-1,2i-1}=B_{i,j}/2 \tag{A. 2-10}$$

$$a^{(2k)}_{2i-1,2j}=a^{(2k)}_{2j,2i-1}=G_{i,j}/2 \tag{A. 2-11}$$

$$a^{(2k)}_{2i,2j-1}=a^{(2k)}_{2j-1,2i}=-G_{i,j}/2 \tag{A. 2-12}$$

$$a^{(2k)}_{2i,2j}=a^{(2k)}_{2j,2i}=B_{i,j}/2 \tag{A. 2-13}$$

$$b^{(2k)}_{2i-1}=\bar{b}_{i0}/2 \tag{A. 2-14}$$

$$b^{(2k)}_{2i}=-\bar{a}_{i0}/2 \tag{A. 2-15}$$

$$c^{(2k)}=Q_i \tag{A. 2-17}$$

(b) 発電機母線

$$a^{(2k)}_{2i-1,2i-1}=a^{(2k)}_{2i,2i}=1 \tag{A. 2-18}$$

$$c^{(2k)}=-v_i^2 \tag{A. 2-19}$$