伝達関数行列の誘導プログラム

佐藤弘之* 格爾麗** 坂島大輔***

1．まえがき

最近，状態空間法による現代制御理論が完全に実用の域に達したといえる．制御システムは実時間記述の微分方程式である状態方程式により記述される．古典制御理論における周波数応答法による制御系の設計には実用的価値が大きく見過ごすことはできない．このため，現代制御理論を学習中の学生にとって，状態方程式を伝達関数の形式に変形して考察するのが便利な場合が多い．

伝達関数行列の代表的誘導方法としてファディーヴの計算式[3][4][5][6]を使用することが多い．状態方程式からファディーヴの方法と代数方程式の性質を利用して既約伝達関数行列要素を導くアルゴリズムはかなり高度の数値解析技術を要する．状態方程式から伝達関数行列を導出する計算機プログラムを作製することは極めて有意義であるとともにそこに含まれる計算手法を各所に応用することができる．本文では一例として作製したC＋＋によりコーディングした教材用計算機プログラムを紹介する．

2．状態方程式と伝達関数行列の関係

多入出力線形時不変制御システムの状態方程式は一般に次のように表すことができる．

$$\dot{\boldsymbol{X}}(t) = \boldsymbol{A}\boldsymbol{X}(t)+\boldsymbol{B}\boldsymbol{U}(t) \quad (1)$$

$$\boldsymbol{Y}(t) = \boldsymbol{C}\boldsymbol{X}(t)+\boldsymbol{D}\boldsymbol{U}(t) \quad (2)$$

ここに

$\boldsymbol{X}(t)$；n 次元状態ベクトル
$\boldsymbol{Y}(t)$；l 次元出力ベクトル
$\boldsymbol{U}(t)$；m 次元制御ベクトル
$\boldsymbol{A}$；$(n\times n)$ 次元システム行列
$\boldsymbol{B}$；$(n\times m)$ 次元制御行列
$\boldsymbol{C}$；$(l\times n)$ 次元出力行列
$\boldsymbol{D}$；$(l\times m)$ 次元伝達行列

初期値ベクトル $\boldsymbol{x}_0$ とおき，式（1）をラプラス変換する，そして整理すれば次式が得ら

*明星大学理工学部電気工学科教授 工学博士
**明星大学理工学研究科電気工学専攻
***明星大学理工学研究科電気工学専攻

れる.

$$\boldsymbol{X}(s) = (s\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{x}_0+(s\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{B}\boldsymbol{U}(s) \tag{3}$$

ここに，$\boldsymbol{I}$ は n 次元単位行列

さらに，式（2）のラプラス変換に，式（3）を代入して，整理すれば次式を導くことができる.

$$\boldsymbol{Y}(s) = \boldsymbol{C}(s\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{x}_0+\boldsymbol{C}(s\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{B}\boldsymbol{U}(s)+\boldsymbol{D}\boldsymbol{U}(s) \tag{4}$$

式（4）において，初期値 $\boldsymbol{x}_0 = 0$ とすれば，次式のように書ける.

$$\boldsymbol{Y}(s) = \{\boldsymbol{C}(s\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{B}+\boldsymbol{D}\}\boldsymbol{U}(s) \tag{5}$$

これより次の伝達関数行列の計算式が得られる.

$$\boldsymbol{G}(s) = \boldsymbol{C}(s\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{B}+\boldsymbol{D} \tag{6}$$

ここに

$\boldsymbol{G}(s)$；$(l\times m)$ 次元伝達関数行列

現実の問題は $\boldsymbol{D} = 0$ の $\boldsymbol{G}(s)$ が厳密にプロパーな場合が多いので，計算機プログラムの簡単のため，次の伝達関数行列の数式表示形式要素の誘導方法を考える.

$$\boldsymbol{G}(s) = \boldsymbol{C}(s\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{B} \tag{7}$$

3．ファデーヴのアルゴリズム[4]

行列式の計算を避けて，伝達関数や $(s\boldsymbol{I}-\boldsymbol{A})^{-1}$ の計算が可能であるので便利である.

行列の特性方程式を次式とする.

$$P(s) = |\,s\boldsymbol{I}-\boldsymbol{A}\,| = \boldsymbol{a}_0s^n+\boldsymbol{a}_1s^{n-1}+\boldsymbol{a}_2s^{n-2}+\cdots+\boldsymbol{a}_{n-1}s+\boldsymbol{a}_n \tag{8}$$

このとき

$$(s\boldsymbol{I}-\boldsymbol{A})^{-1} = \frac{adj(s\boldsymbol{I}-\boldsymbol{A})}{det(s\boldsymbol{I}-\boldsymbol{A})} = \frac{1}{P(s)}[\boldsymbol{\lambda}_{n-1}s^{n-1}+\boldsymbol{\lambda}_{n-2}s^{n-2}+\cdots\boldsymbol{\lambda}_1s+\boldsymbol{\lambda}_0] \tag{9}$$

ここに，特性方程式の係数 $\boldsymbol{a}_i$ および余因子行列 $adj(s\boldsymbol{I}-\boldsymbol{A})$ の係数行列 $\boldsymbol{\lambda}_i$ は次式で計算する.

まず最初に，次式のように置く

$$\boldsymbol{\lambda}_{n-1} = \boldsymbol{I} \tag{8}$$

そして，$i = n-1$ から $i = 0$ まで i を 1 ずつ減じながら式 (11) ～ (13) の計算を繰り返す．ただし，式 (13) は $i = 0$ のとき計算しない.

$$Z_i = A\lambda_i \tag{11}$$

$$a_{n-1} = -\frac{trace(Z_i)}{n-i} \tag{12}$$

$$\lambda_{i-1} = Z_i + a_{n-1}I \tag{13}$$

そして，最後に$a_0 = 1$とおけばよい．

4． 計算機プログラム

4．1 関数プログラムの構成

伝達関数行列の誘導プログラムはメイン・プログラムmain () の他に，次の関数プログラムにより構成されている．

メイン・プログラムが計算の流れをコントロールしている．

（1）宣言部で変数の記憶領域および入力データ・ファイル名．
（2）入力データ・ファイルのオープン
（3）システムのサイズ・パラメータの入力
（4）制御システムの係数行列の入力とそれらの確認用出力．
（5）そして，つぎの関数を連続的に呼び出すことにより一連の計算を実行する．

```
faddeev
sima
transf
```

（6）終了後，さらに入力データが存在すれば，ステップ（3）から繰り返す．
（7）入力データ・ファイルのクローズ，そしてプログラムの終了

```
void    mat_out(double [][N],int ,int ,char []);
void    mat_dsp(double [][M],int ,int ,char []);
void    matcpy(double [][N],double [][N],int );
double  trace(double [][N],int );
void    mat_mpy(double [][N],double [][N],double [][N],int );
void    save(double [][N][N],double [][N],int ,int );
void    faddeev(double [][N],double [],double [][N][N],int );
int     shift(double [],int );
double  sqr(double );
void    linear_eq(double [],double [],double []);
void    quadratic_eq(double [],double [],double []);
double  func(double [],double ,int );
void    synthetic_division(double [],double [],double ,int );
int     polynom(double ,double [],int );
void    simax(double [],double [][N][N],int ,int ,int ,char []);
void    sima(double [],double [][N][N],int );
void    teqcxp(double [][N],double [][N],double [][N],
                        int ,int );
void    geqtxb(double [][N],double [][M],double [][M],
```

```
                int ,int ,int );
void    stock(double [][M][N],double [][M],int ,int ,int );
void    transf(double [],double [][N][N],
                double [][M],double [][N],int ,int ,int );
```

4．2 多項式の表現法

一般に伝達関数行列の要素は次式に示すような有限次の多項式の比，すなわち有理関数，を要素とする行列で表される．

$$g_{i,j}(s)=\frac{q(s)}{p(s)}$$

ただし

$$p(s)=a_0s^n+a_1s^{n-1}+a_2s^{n-2}+\cdots+a_{n-1}s+a_n$$
$$q(s)=b_0s^m+b_1s^{m-1}+b_2s^{m-2}+\cdots+b_{m-1}s+b_m$$

一般に，多項式はデータとして，次数と係数の数値が与えられていればよい．たとえば，次式に示す伝達関数を考える．

$$g_{i,j}(s)=\frac{s+3}{s^3+4s^2+3s}$$

計算の途中ならびに最終結果をコンピュータ・ディスプレイ上に表示する方法として，本文では次に示すような多項式およびそれらの比の表現形式を使用する．すなわち，beta []. につずく数値は分子，alpha []. につずく数値は分母の係数を表し，次数は出力された係数の個数により容易に判断できる．

```
 beta[].
alpha[].
```

4．3 Faddeev のアルゴリズムの計算機プログラム

関数faddeevがFaddeevのアルゴリズムを実行する計算機プログラムである．つぎに，この関数の引数を示す．

double a[][N]	状態方程式の係数行列
double alpha[]	特性方程式の係数
double store[][N][N]	行列λの集合配列
int n	システムの次数

プログラムの流れ

（1）単位行列を二次元配列lambdaに格納
（2）パラメータ$i=n-1$と設定
（3）二次元配列lambdaを三次元配列storeに格納
（4）行列Aと行列lambdaとの積Z

（5）係数alpha［$n-i$］の計算
（6）行列$\boldsymbol{Z}$の主対角線要素にalpha［$n-i$］を加算して二次元配列lambdaとする
（7）パラメータ$i>0$のとき，$i=i-1$として（3）に移動する．$i=0$のとき（8）に移動
（8）係数alpha［0］=1として，終了

4．4　特性方程式の逆行列

関数simaは行列$adj(s\boldsymbol{I}-\boldsymbol{A})$の要素生成制御プログラムである．その引数を示す．

double alpha[]	特性方程式の係数
double store[][N][N]	行列λの集合配列
int n	システムの次数

4．5　行列$adj(s\boldsymbol{I}-\boldsymbol{A})$の要素生成プログラム

関数simaxは行列$adj(s\boldsymbol{I}-\boldsymbol{A})$の要素を三次元配列storeから生成するプログラムである．引数を示す．

double alpha[]	特性方程式の係数
double store[][N][N]	行列λの集合配列
int i	行番号
int j	列番号
int n	システムの次数
char head[]	文字ストリング

4．6　特性方程式の逆行列と伝達関数行列要素の既約分数化法

分数の分母および分子に同一因子を含む場合それぞれを約して簡単な数式に置き変えることができる．手計算では，視察によっても可能なことがあり，それほど困難なことではない．しかし，コンピュータ上ではそれぞれの多項式を零とおいた代数方程式の根を計算する．そして，同一の根が含まれていれば，それを消して，低次の式に変形する計算操作を行わなければならない．

一般に，伝達関数は分子の方が分母に比較して低次の場合が多い．したがって，次のようなアルゴリズムを提案することができる．

（1）分子の多項式を零とおいた代数方程式の全ての根を求める．
（2）上で求めた根を分母の多項式に代入する．そして関数値が零になるか否かを調べる．
　零でない場合ステップ（4）に移動する．
　零の場合は次のステップ（3）に移動する．
（3）分母の多項式の関数値を零にする分子の根を除式として合成除法を使用して商式を求め，分母の多項式を分子との共通因子を取り除いた低次の多項式にする．
（4）分子の多項式の根全てについて調べたときステップ（5）に移動する．そうでないとき，次の根の代入操作のためステップ（2）に移動する．
（5）全ての根について調べた後，分子の根のうち約分に使用したものを除いて，根から多項式を再合成する．

4．7 伝達関数行列の生成

関数transfは行列 $(sI-A)^{-1}$ に行列 B と C を乗じ，伝達関数行列 $C(sI-A)^{-1}B$ の要素生成制御するプログラムである．引数

double alpha[]	特性方程式の係数
double store[][N][N]	行列 λ の集合配列
double b[][N]	制御行列
double c[][N]	観測行列
int n	システムの次数
int m	制御入力数
int l	観測出力数

4．8 計算機プログラム・コード

```
#include        <fstream.h>
#include        <iomanip.h>
#include        <conio.h>
#include        <stdlib.h>
#include        <ctype.h>
#include        <string.h>
#include        <math.h>

#define N       5
#define M       5
#define L       5

void    mat_out(double [][N],int ,int ,char []);
void    mat_dsp(double [][M],int ,int ,char []);
void    matcpy(double [][N],double [][N],int );
double  trace(double [][N],int );
void    mat_mpy(double [][N],double [][N],double [][N],int );
void    save(double [][N][N],double [][N],int ,int );
void    faddeev(double [][N],double [],double [][N][N],int );
int     shift(double [],int );
double  sqr(double );
void    linear_eq(double [],double [],double []);
void    quadratic_eq(double [],double [],double []);
double  func(double [],double ,int );
void    synthetic_division(double [],double [],double ,int );
int     polynom(double ,double [],int );
void    simax(double [],double [][N][N],int ,int ,int ,char []);
void    sima(double [],double [][N][N],int );
void    teqcxp(double [][N],double [][N],double [][N],
                    int ,int );
void    geqtxb(double [][N],double [][M],double [][M],
                int ,int ,int );
void    stock(double [][M][N],double [][M],int ,int ,int );
void    transf(double [],double [][N][N],
                double [][M],double [][N],int ,int ,int );

ifstream        infile;

void    main()
{       int     i,j,k,n,m,l;
        static double   a[N][N],b[N][M],c[L][N],
                alpha[N],store[N][N][N];
        char    fname[]="System.txt ;
```

```
        infile.open(fname,ios::in|ios::nocreate);
        if(!infile) {
                cerr<<"File "<<fname<<" not found.";
                exit(0); }

        k=0;infile>>n;
        do {
                infile>>m>>l;
                for(i=0;i<n;i++) for(j=0;j<n;j++) infile>>a[i][j];
                for(i=0;i<n;i++) for(j=0;j<m;j++) infile>>b[i][j];
                for(i=0;i<l;i++) for(j=0;j<n;j++) infile>>c[i][j];

                if(k>0) cout<<endl<<endl<<endl;k++;
                cout<<setw(14)<<"Example  "<<k<<'.';
                mat_out(a,n,n,"Matrix A[i,j].");
                mat_dsp(b,n,m,"Matrix B[i,j].");
                mat_out(c,l,n,"Matrix C[i,j].");

                faddeev(a,alpha,store,n);
                sima(alpha,store,n);
                transf(alpha,store,b,c,n,m,l);

                infile>>n; }
        while(!infile.eof());
        infile.close();
}

void    mat_out(double a[][N],int n,int m,char name[])
{       int     i,j;

                cout<<endl<<endl;for(i=0;i<5;i++) cout<<' ';
                cout<<name<<endl<<endl;
                for(j=0;j<m;j++) cout<<setw(12)<<(j+1);cout<<endl;
                for(i=0;i<n;i++) {
                        cout<<setw(3)<<(i+1);
                        for(j=0;j<m;j++) cout<<setw(12)<<a[i][j];
                        cout<<endl; }
                if(tolower(getch())=='q') exit(0);
}

void    mat_dsp(double b[][M],int n,int m,char name[])
{       int     i,j;

                cout<<endl<<endl;for(i=0;i<5;i++) cout<<' ';
                cout<<name<<endl<<endl;
                for(j=0;j<m;j++) cout<<setw(12)<<(j+1);cout<<endl;
                for(i=0;i<n;i++) {
                        cout<<setw(3)<<(i+1);
                        for(j=0;j<m;j++) cout<<setw(12)<<b[i][j];
                        cout<<endl; }
                if(tolower(getch())=='q') exit(0);
}

void    matcpy(double a[][N],double b[][N],int n)
{       int     i,j;

        for(i=0;i<n;i++) for(j=0;j<n;j++) a[i][j]=b[i][j];
}
```

```
double  trace(double a[][N],int n)
{       int     i;
        double  t;

        t=0.0;
        for(i=0;i<n;i++) t+=a[i][i];
        return t;
}

void    mat_mpy(double a[][N],double b[][N],double c[][N],int n)
{       int     i,j,k;
        double  s;

        for(i=0;i<n;i++) for(j=0;j<n;j++) {
                s=0.0;
                for(k=0;k<n;k++) s+=a[i][k]*b[k][j];
                c[i][j]=s; }
}

void    save(double store[][N][N],double p[][N],int n,int k)
{       int     i,j;

        for(i=0;i<n;i++) for(j=0;j<n;j++)
                store[i][j][k]=p[i][j];
}

void    faddeev(double a[][N],double alpha[],
                        double store[][N][N],int n)
{       int     i,j;
        double  lambda[N][N],z[N][N];
        char    tag[10],name[20];

        for(i=0;i<n;i++) {
                for(j=0;j<n;j++) lambda[i][j]=0.0;
                lambda[i][i]=1.0; }

        for(i=n-1;i>=0;i--) {
                itoa(i,tag,10);
                strcpy(name,"Lambda[i,j].");
                strcat(name,tag);
                mat_out(lambda,n,n,name);
                save(store,lambda,n,i);

                mat_mpy(a,lambda,z,n);
                strcpy(name,"Matrix Z[i,j].");
                strcat(name,tag);
                mat_out(z,n,n,name);

                alpha[n-i]=-trace(z,n)/(n-i);
                cout<<endl<<setw(15)<<"alpha["<<i<<"] = "<<alpha[n-i]<<endl;

                if(i>0) {
                        matcpy(lambda,z,n);
                        for(j=0;j<n;j++) lambda[j][j]+=alpha[n-i]; }}

        alpha[0]=1.0;
        cout<<endl<<setw(10)<<"Alpha[i]"<<endl;
        for(i=0;i<=n;i++) cout<<setw(12)<<alpha[i];cout<<endl;
}
```

```
int     shift(double beta[], int n)
{       int     i,m;

        m=n;
        for(i=0;i<n;i++) if(beta[i]==0) m--;else break;
        if(m==0) m=1;
        if(m<n) for(i=0;i<m;i++) beta[i]=beta[i+n-m];
        return m-1;
}

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

void    linear_eq(double a[], double xr[], double xi[])
{       xr[0]=-a[1]/a[0];xi[0]=0.0;     }

void    quadratic_eq(double a[], double xr[], double xi[])
{       double  d,s;

        d=sqr(a[1])-4.0*a[0]*a[2];
        if(d<0.0) {
                xr[0]=xr[1]=-0.5*a[1]*a[0];
                xi[0]=0.5*sqrt(-d)/a[0];
                xi[1]=-xi[0]; }
        else if(d>0.0) {
                xi[0]=xi[1]=0.0;
                if(a[1]<0.0) s=0.5/a[0];else s=-0.5/a[0];
                xr[0]=s*(fabs(a[1])+sqrt(d));
                xr[1]=a[2]/(a[0]*xr[0]); }
        else {
                xr[0]=xr[1]=-0.5*a[1]*a[0];
                xi[0]=xi[1]=0.0; }
}

double  func(double a[], double x, int n)
{       int     i;
        double  f;

        f=a[0];
        for(i=1;i<=n;i++) f=f*x+a[i];
        return f;
}

void    synthetic_division(double a[], double b[], double p, int n)
{       int     i;

        b[0]=a[0];
        for(i=1;i<=n;i++) b[i]=b[i-1]*p+a[i];
}

int     polynom(double p, double a[], int n)
{       int     i,j,k,m;
        double b[2],c[N];

        b[0]=1.0;b[1]=-p;m=1;k=m+n;
        for(i=0;i<=k;i++) c[i]=0.0;
        for(i=0;i<=n;i++) for(j=0;j<=m;j++) c[i+j]+=a[i]*b[j];
        for(i=0;i<=k;i++) a[i]=c[i];
        return k;
}
```

```
void    simax(double alpha[],double store[][N][N],int i,int j,int n,char head[])
{       int     k,l,m,nn,nr,red[N],disp=0;
        double  beta[N],xr[N],xi[N],a[N],b[N];

                for(k=0;k<n;k++) beta[n-1-k]=store[i][j][k];
                cout<<endl<<head
                        <<"   row : "<<(i+1)<<"  column : "<<(j+1)<<endl;
                m=shift(beta,n);

                cout<<endl<<"     beta[].";
                for(k=0;k<=m;k++) cout<<setw(12)<<beta[k];cout<<endl;

                if(m>0||beta[0]!=0.0) {
                        cout<<"    alpha[].";
                        for(k=0;k<=n;k++) cout<<setw(12)<<alpha[k];
                        cout<<endl; }
                if(m==2) quadratic_eq(beta,xr,xi);
                else if(m==1) linear_eq(beta,xr,xi);
                if(m>0) {
                        for(k=0;k<m;k++) red[k]=0;
                        if(disp>0) {
                        cout<<endl<<"  Linear factors of the nominator."
                                <<endl<<endl;
                        cout<<setw(3)<<setw(12)<<"xr"<<setw(12)<<"xi"<<endl;
                        for(k=0;k<m;k++) cout<<setw(3)<<setw(12)<<xr[k]
                                <<setw(12)<<xi[k]<<endl; }

                        for(k=0;k<=n;k++) a[k]=alpha[k];
                        if(disp>0) {
                        cout<<endl<<"    a[].";
                        for(k=0;k<=n;k++) cout<<setw(12)<<a[k];cout<<endl; }

                        nr=n;
                        for(k=0;k<m;k++) if(xi[k]==0.0&&
                                func(a,xr[k],nr)==0.0) {
                                synthetic_division(a,b,xr[k],nr);nr--;
                                red[k]=1;
                                if(disp>0) {
                                cout<<"  Reduced a[] by x = "<<xr[k]<<endl;
                                for(l=0;l<=nr;l++) cout<<setw(12)<<b[l];
                                cout<<endl; }
                                for(l=0;l<=nr;l++) a[l]=b[l]; }
                        nn=0;
                        b[0]=beta[0];
                        for(k=0;k<m;k++) if(red[k]==0)
                                nn=polynom(xr[k],b,nn);

                        l=0;
                        for(k=0;k<m;k++) if(red[k]!=0) l++;

                        if(l!=0) {
                        cout<<endl<<"    Reduced element of Inv.[sI-A], "
                                <<"   row : "<<(i+1)<<"  column : "<<(j+1)
                                <<endl;
                        cout<<endl<<"     beta[].";
                        for(k=0;k<=nn;k++) cout<<setw(12)<<b[k];cout<<endl;

                        cout<<"    alpha[].";
                        for(k=0;k<=nr;k++) cout<<setw(12)<<a[k];cout<<endl; }}
}
```

```
void    sima(double alpha[],double store[][N][N],int n)
{       int     i,j,k,disp=0;
        static double   p[N][N];

        char    name[20],tag[10];

        for(k=n-1;k>=0;k--) {
                for(i=0;i<n;i++) for(j=0;j<n;j++) p[i][j]=store[i][j][k];
                if(disp>0) {
                        itoa(k,tag,10);
                        strcpy(name,"Lambda[i,j].");
                        strcat(name,tag);
                        mat_out(p,n,n,name);  }}

        for(i=0;i<n;i++) for(j=0;j<n;j++) {
        simax(alpha,store,i,j,n,"¥n¥n    Element of Inv.[sI-A], ");
        if(tolower(getch())=='q') exit(0); }
}

void    teqcxp(double c[][N],double p[][N],double t[][N],
                int l,int n)
{       int     i,j,k;
        double  s;

        for(i=0;i<l;i++) for(j=0;j<n;j++) {
                s=0.0;
                for(k=0;k<n;k++) s+=c[i][k]*p[k][j];
                t[i][j]=s; }
}

void    geqtxb(double t[][N],double b[][M],double g[][M],
                int l,int n,int m)
{       int     i,j,k;
        double  s;

        for(i=0;i<l;i++) for(j=0;j<m;j++) {
                s=0.0;
                for(k=0;k<n;k++) s+=t[i][k]*b[k][j];
                g[i][j]=s; }
}

void    stock(double h[][M][N],double g[][M],int l,int m,int k)
{       int     i,j;

        for(i=0;i<l;i++) for(j=0;j<m;j++)
                h[i][j][k]=g[i][j];
}

void    transf(double alpha[],double store[][N][N],
                double b[][M],double c[][N],int n,int m,int l)
{       int     i,j,k,disp=0;
        static double   p[N][N],t[L][N],g[L][M],h[L][M][N];
    char        name[20],tag[10];

        cout<<endl<<endl
                <<"     Transfer Function Matrix G(s)."<<endl;
        if(disp>0) {
                mat_dsp(b,n,m,"Matrix B[i,j].");
                mat_out(c,l,n,"Matrix C[i,j]."); }

        for(k=n-1;k>=0;k--) {
```

```
for(i=0;i<n;i++) for(j=0;j<n;j++) p[i][j]=store[i][j][k];
if(disp>0) {
        itoa(k, tag, 10);
        strcpy(name, "Lambda[i, j]. ");
        strcat(name, tag);
        mat_out(p, n, n, name); }

teqcxp(c, p, t, l, n);
if(disp>0) {
        itoa(k, tag, 10);
        strcpy(name, "Product C*inv. [sI-A]. ");
        strcat(name, tag);
        mat_out(t, l, n, name); }
        geqtxb(t, b, g, l, n, m);
        itoa(k, tag, 10);
        strcpy(name, "Matrix C*inv. [sI-A]*B. ");
        strcat(name, tag);
        mat_dsp(g, l, m, name);
        stock(h, g, l, m, k); }

        for(i=0;i<l;i++) for(j=0;j<m;j++) {
    simax(alpha, h, i, j, n, "¥n¥n    Element of G(s), ");
                if(tolower(getch())=='q') exit(0); }
```

5. 数値計算例

5. 1 モデルシステム

前述した計算機プログラム・コードの動作と計算性能を確認するために，次のモデル制御システムを対象に計算を試みる．

$$\frac{d}{dt}\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix}=\begin{bmatrix}1&1&0\\0&1&1\\0&0&1\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix}+\begin{bmatrix}0&0\\0&0\\1&0\end{bmatrix}\begin{bmatrix}u_1\\u_2\end{bmatrix}$$

$$\begin{bmatrix}y_1\\y_2\\y_3\end{bmatrix}=\begin{bmatrix}1&0&0\\0&0&0\\0&0&0\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix}$$

5. 2 理論的方法

計算機プログラムによらないで，行列の計算理論にもとずいて伝達関数行列を誘導する．モデルシステムの特性方程式は次式のように書ける．

$$det(s\boldsymbol{I}-\boldsymbol{A})=\begin{vmatrix}s-1&-1&0\\0&s-1&-1\\0&0&s-1\end{vmatrix}=(s-1)^3=s^3-3s^2+3s-1$$

また，余因子行列は簡単な計算により次式のようになる．

$$adj(s\boldsymbol{I}-\boldsymbol{A})=\begin{bmatrix}(s-1)^2&s-1&1\\0&(s-1)^2&s-1\\0&0&(s-1)^2\end{bmatrix}$$

したがって，伝達関数行列は，式（9）と式（6）により，次式のように誘導される．

$$G(s) = C(sI-A)^{-1}B = \frac{1}{(s-1)^3}\begin{bmatrix}1 & 0\\ 0 & 0\\ 0 & 0\end{bmatrix}$$

5．3　入力データ

付録に示す入力データファイルのうち，最初のものが上記のモデル制御システムに対応している．このデータファイルの2番目以降のモデルについては，紙面制約の都合で計算例の紹介を省略する．しかし，このようなデータファイルを作成すれば，各モデルは連続的に計算され，CRTディスプレイ画面上に表示される．CRT画面が静止したときにはスペースバーを押せば次の表示に移行する．

5．4　実行結果

前述した計算機プログラムと入力データによる，計算機のCRT画面上に表示されたものを示す．なお，紙面の制約の都合でスペース行など圧縮された部分があるが，計算結果の内容は変形されていない．

```
   Example  1.

   Matrix A[i,j].
         1           2           3
1           1           1           0
2           0           1           1
3           0           0           1

   Matrix B[i,j].
         1           2
1           0           0
2           0           0
3           1           0

   Matrix C[i,j].
         1           2           3
1           1           0           0
2           0           0           0
3           0           0           0

   Lambda[i,j].2
         1           2           3
1           1           0           0
2           0           1           0
3           0           0           1

   Matrix Z[i,j].2
         1           2           3
1           1           1           0
2           0           1           1
3           0           0           1

       alpha[2] = -3

   Lambda[i,j].1
         1           2           3
1          -2           1           0
```

```
2          0          -2          1
3          0           0         -2

   Matrix Z[i,j].1
          1          2          3
1        -2         -1          1
2         0         -2         -1
3         0          0         -2

       alpha[1] = 3

   Lambda[i,j].0
          1          2          3
1         1         -1          1
2         0          1         -1
3         0          0          1

   Matrix Z[i,j].0
          1          2          3
1         1          0          0
2         0          1          0
3         0          0          1

       alpha[0] = -1

Alpha[i]
         1         -3          3         -1

  Element of Inv.[sI-A],    row : 1  column : 1
   beta[].          1         -2          1
  alpha[].          1         -3          3         -1

  Reduced element of Inv.[sI-A],    row : 1  column : 1
   beta[].          1
  alpha[].          1         -1

  Element of Inv.[sI-A],    row : 1  column : 2
   beta[].          1         -1
  alpha[].          1         -3          3         -1

  Reduced element of Inv.[sI-A],    row : 1  column : 2
   beta[].          1
  alpha[].          1         -2          1

  Element of Inv.[sI-A],    row : 1  column : 3
   beta[].          1
  alpha[].          1         -3          3         -1

  Element of Inv.[sI-A],    row : 2  column : 1
   beta[].          0

  Element of Inv.[sI-A],    row : 2  column : 2
   beta[].          1         -2          1
  alpha[].          1         -3          3         -1

  Reduced element of Inv.[sI-A],    row : 2  column : 2
   beta[].          1
  alpha[].          1         -1

  Element of Inv.[sI-A],    row : 2  column : 3
   beta[].          1         -1
  alpha[].          1         -3          3         -1
```

```
  Reduced element of Inv.[sI-A],     row : 2  column : 3
   beta[].           1
  alpha[].           1          -2          1

  Element of Inv.[sI-A],     row : 3  column : 1
   beta[].           0

  Element of Inv.[sI-A],     row : 3  column : 2
   beta[].           0

  Element of Inv.[sI-A],     row : 3  column : 3
   beta[].           1          -2          1
  alpha[].           1          -3          3          -1

  Reduced element of Inv.[sI-A],     row : 3  column : 3
   beta[].           1
  alpha[].           1          -1

   Transfer Function Matrix G(s).

   Matrix C*inv.[sI-A]*B.2
         1           2
1           0           0
2           0           0
3           0           0

   Matrix C*inv.[sI-A]*B.1
         1           2
1           0           0
2           0           0
3           0           0

   Matrix C*inv.[sI-A]*B.0
         1           2
1           1           0
2           0           0
3           0           0

row : 1  column : 1
   beta[].           1
  alpha[].           1          -3          3          -1

row : 1  column : 2
   beta[].           0

row : 2  column : 1
   beta[].           0

row : 2  column : 2
   beta[].           0

row : 3  column : 1
   beta[].           0

row : 3  column : 2
   beta[].           0
```

6．あとがき

本文に紹介した伝達関数行列の誘導のための計算機プログラムは例題の範囲内では正確

な解を求めることができた．提案したプログラムにおいて，分母と分子の有理関数の約分には，コーディングの簡単のため一次因子によるものに限定した．二次因子による約分は，Bairstow法などで二次因子を求め，対応する合成除法により，一次因子の場合と同様な考えにより実現できる．しかし，計算過程において高次の多項式に変形しているため，使用している数値解析手法またはその類似の手法を利用する限りにおいて，制御システムの高次元化に対応できなくなる場合がないとは言えない．そのような場合には，とりあえず，それらの問題毎に対応せざるを得ない．しかし，10次程度までの低次元の制御システムしか処理できないとは言え，実際に計算機プログラムをコーディングし，そして計算してみることにより，現代制御理論の学習の途中において状態方程式と伝達関数の変形方法を実際に体験し，状態方程式と伝達関数行列の相互関係を理解する一助になるだろう．

参考文献

1）伊藤正美：自動制御，丸善，1981
2）近藤文治，藤井克彦：制御工学，オーム社，1972
3）小郷寛：システム制御理論入門，実教出版
4）白石昌武：入門現代制御理論，日刊工業新聞社，1995
5）古田勝久，佐野昭：基礎システム理論，コロナ社，1978
6）児玉慎三，須田信英：システム制御のためのマトリクス理論，計測自動制御学会，1978
7）須田信英：線形システム理論，朝倉書店，1993
8）佐藤弘之：数値計算法，森北出版，1993

付録

モデル制御システムの入力データ例6ケースを示す．ファイル名を"System.txt"とすれば，前述したプログラムでそのまま入力される．

```
3    2   3
1.0   1.0   0.0
0.0   1.0   1.0
0.0   0.0   1.0

0.0   0.0
0.0   0.0
1.0   0.0

1.0   0.0   0.0
0.0   0.0   0.0
0.0   0.0   0.0

3     1     1
0.0   1.0   0.0
0.0  -1.0  -1.0
0.0   0.0  -3.0

0.0   1.0   1.0

1.0   0.0   0.0

3     1     1
-2.0  0.0   1.0
0.0  -2.0   1.0
0.0  -3.0   2.0

1.0   2.0   2.0
```

```
-1.0    1.0    0.0

 3      2      1
 0.0    1.0    2.0
 1.0    3.0    4.0
 0.0    5.0    6.0

 1.0    0.0
 0.0    0.0
 0.0    1.0

 0.0    1.0    0.0

 2      1      1
-1.0    0.0
 0.0   -2.0

 0.0    1.0

 1.0    0.0

 3      2      1
 0.0    1.0    0.0
 0.0    0.0    1.0
 1.0    0.0    0.0

 1.0    0.0
 0.0    0.0
 0.0    1.0

 1.0    0.0    0.0
```