REFERENCES

[1] W.N. Anderson , Jr. , T.D. Morley , and G.E.Trapp " Positive solutions to $X=A-BX^{-1}B^{*}$ " , Linear Algebra Appl., 134 (1990), (53-62) .

[2] P.J. Antsaklis and A.N.Michel "Linear systems ", McGraw-Hill ,1998 .

[3] D.S. Bernstein and Wasin So , " Some explicit formulas for the matrix exponential " , IEEE Transactions on Automatic Control , vol.38 (1993) no. 8 , pp (1228-1232).

[4] R. Bellman " Introduction to matrix analysis" , McGraw-Hill, 1960.

[5] R. Bronson , " Theory and problems of matrix operations " , McGraw-Hill ,1989 .

[6] R.Byers and S.Nash "On the singular vectors of the Lyapunov operator" , Siam J. Alg. Disc. Math . Vol.8 , No.1, pp(59-66), (1987).

[7] C.Moler and C.Van Loan "Ninteen dubious ways to compute the exponential of a matrix " Siam Review . Vol. 20 , No.4 ,(1978) , pp(801-836).

[8] J.C.Engwerda , " On the existence of a positive definite solution of the matrix equation $X+A^{T}X^{-1}A=I$ " Linear Algebra Appl. 194 (1993) , pp (91-108) .

[9] J.C.Engwerda ,A.C.M.Ran and A.L.Rijkeboer "Necessary and sufficient conditions for the existence of a positive definite solution of the matrix equation $X+A^{*}X^{-1}A=Q$ " Linear Algebra Appl .186 (1993) pp (255-275)

[10] W. Fair and Y.L. Luke. Num. Math. (1970) , vol. 14 , pp (379-382).

[11] F.W. Fairman , " Linear control theory : The state space approach" (1998) .

[12] M. Fiedler , " Special matrices and their applications in numerical mathematics " , Martinus Nijhoff Publishers, (1986).

[13] S. Ganapathy and A.S. Rao " Transient response evaluation from the state transition matrix " Proc. IEEE. (1969), pp (347-349) .

[14] Hassan N.A.Ismail and Elsayed M. El-sayed , " Restrictive Padè approximation", 1st national conference of mathematical and physical science and applications , pp (63-76) , (1995) Cairo, Egypt.

[15] Hassan N.A.Ismail and Elsayed M.El-Barbary "Restrictive Padè approximation and parabolic partial differential equations " Int. J. Computer Math. , vol. 66 (1998) , pp (343-351) .

[16] Hassan N.A.Ismail and Mohamed A.Ramadan "An efficient computation of the transient response evaluation from the state transition matrix" Proceedings of the 13th world congress on computation and Applid Math . July (22-26), (1991) . Trinity college, Dublin, Ireland .

[17] P. Lancaster , " Theory of matrices" , Academic press , N.4.(1969).

[18] M.L. Liou , " A novel method of evaluating transient response" Proc. IEEE. , vol .54 (1966), pp (20-23) .

[19] Salah M.El-sayed , " The study on special matrices and numerical methods for special matrix equations" ,Ph.D. Thesis , Sofia , (1996).

[20] R.S Varga. J. Math. Phys. (1961), vol. 40, pp (220-231) .

[21] So.Wasin " Equality cases in matrix exponential inequalities ", Siam.J. Matrix Analy. Vol.13, no.4 , pp (1154-1158).October (1992).

[22] Wei-Yong Yan, John.B. Moore and Uwe Helmke "Recursive algorithm for solving a class of nonlinear matrix equations with applications to certain sensitivity optimization problems" Saim J. Control and optimization, 32 (1994), pp (1559-1576) .

[23] A.Wragg , " Computation of the exponential of a matrix I : Theoretical considerations " , J. Inst. Math. Applic , II , pp (369-375) (1973) .

[24] X. Zhan and Jianjumxie , " On the matrix equation $X + A^T X^{-1} A = I$ Linear Algebra Appl. vol. 247, pp (337-345) , (1996) .

APPENDIX :

```
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
#include<math.h>
#include<stdlib.h>
/*-------------------------------------------------------*/
// << Global D E C L A R A T I O N S >>
/*-------------------------------------------------------*/
double     SpaceMat [30][2*30], A [30][30],B [30][30];
double     C [30][30],Tempo [30][30],D [30][30],I [30][30];
double     X1 [30][30],X2 [30][30],N [30][30];
double     alpha,stoppingvalue,nm,pivot ;
int        n,no=0,i,j,k,row,clmn,index;
int        clmndiff=15,rowdiff=2,clmn0=2,row0=2;
/*-------------------------------------------------------*/
// << Function P R O T O T Y P E S >>
/*-------------------------------------------------------*/
void    Inits (void);
void    InputMatA (void);
void    DisplayMat(double [30][30]);
void    DisplayMatSpaceMat (void);
void    Inverse (double[30][30],double[30][30]);
void    Transpose (double[30][30],double[30][30]);
void    Assign (double[30][30],double[30][30]);
void    Add (double[30][30],double[30][30],double[30][30]);
void    Sub (double[30][30],double[30][30],double[30][30]);
void    Mult(double[30][30],double[30][30],double[30][30]);
void    Mult (double ,double [30][30],double [30][30]);
double  Norm (double [30][30]);
void    Root (double [30][30],double [30][30]);
int     Equal(double [30][30],double [30][30]);
/*-------------------------------------------------------*/
// << Function:  M A I N ( ) >>
/*-------------------------------------------------------*/
void    main ()
{
 cout<<setprecision(8)<<setw(10); clrscr();
 cout<<"Enter the size of the Matrix n= ";cin>>n;
 n>30?n=30:(n<1?n=1:n=n);          //Check bounds of Matrix
 cout<<"\nEnter the positive real number Alpha=";cin>>alpha;
 alpha<0?alpha=0:alpha=alpha;
 cout<<"\nEnter the stopping value= ";cin>>stoppingvalue;
 clrscr ();
 Inits  ();
 char     ch;
 do
 { Assign (X1,N);
   Transpose (A,D);
   Inverse (X1,C);
   Root (C,X1);
   Mult (D,X1,C);
   Mult (C,A,D);
   Add (I,D,X2);                  // X[k+1] has been calculated
```

```
 cout <<"\nA is";getch();
 DisplayMat (A);
 cout <<"\nX2 is";getch();
 DisplayMat (X2);
 Sub (X2,N,D);                          // X[k+1]-X[k]
 cout <<"\nSub=";getch();
 DisplayMat(D);
 nm=Norm (D);                           // || X[k+1]-X[k] ||
 no++;
 cout <<"\nIteration no."<<no<<", Norm="<<nm<<
        ",stop. value="<<stoppingvalue;getch();
 DisplayMat (X2);
 cout <<"\nPress s to stop processing...";ch=getch();
 Assign(X2,X1);
}
 while ((nm>stoppingvalue)&&(ch!='s'));
 DisplayMat(X2);
 }
/*-------------------------------------------------------*/
// << Function B O D I E S >>
/*-------------------------------------------------------*/
// Function :  Inits()
/*-------------------------------------------------------*/
void     Inits (void)
{
  for(i=0;i<n;i++)
  for(j=0;j<n;j++)
{
 I[i][j]=(i==j);
 if(i==j) A[i][j]=(double)(((double)(j+2+i))/((double)(3*n)));
 else
 A[i][j]=(double)(((double)(2+i+j))/((double)(3*n*n)));
 }
 InputMatA ();
 Mult (alpha,I,X1);
 Assign(X1,X2);
 }
/*-------------------------------------------------------*/
// Function :  Inverse(a,b)
/*-------------------------------------------------------*/
void    Inverse(double a[30][30],double b[30][30])
{
 for(i=0;i<n;i++)
 for(j=0;j<n;j++)
 { SpaceMat[i][j]=(i==j);
   SpaceMat[i][n+j]=a[i][j];
  }
// CALCULATIONS -->      // UPPER TRIANGULAR ...
int         startrow = 0;
for(clmn=n;clmn<(n+n);clmn++)
{
  pivot=SpaceMat[startrow][clmn];
  for(i=0;i<(n+n);i++)
  SpaceMat[startrow][i]/=pivot;
  startrow++;
  for(row=startrow;row<n;row++)
```

```
 {
  pivot=-1*SpaceMat[row][clmn];
  for(index=0;index<(n+n);index++)
  SpaceMat[row][index]+=SpaceMat[startrow-1][index]*pivot;
 } }
// LOWER TRIANGULAR ...
for(;clmn>=n;clmn--)
 {
 startrow--;
 for(row=startrow;row>=0;row--)
 {
  pivot=-1*SpaceMat[row][clmn];
  for(index=0;index<(n+n);index++)
  SpaceMat[row][index]+=SpaceMat[startrow+1][index]*pivot;
 } }
// THE INVERSE MATRIX B  ...
 for(i=0;i<n;i++)
 for(j=0;j<n;j++)
 b[i][j]=SpaceMat[i][j];
 }
/*-------------------------------------------------------*/
// Function :  Transpose(a,b) [i.e. b=aT]
/*-------------------------------------------------------*/
void     Transpose (double a[30][30],double b[30][30])
 {
   for(i=0;i<n;i++)
       for(j=0;j<n;j++)
           b[i][j]=a[j][i];
 }
/*-------------------------------------------------------*/
// Function :  Assign(a,b) [i.e. b:=a]
/*-------------------------------------------------------*/
void       Assign (double a[30][30],double b[30][30])
  {
   for(i=0;i<n;i++)
      for(j=0;j<n;j++)
          b[i][j]=a[i][j];
  }
/*-------------------------------------------------------*/
// Function :  DisplayMat(a)
/*-------------------------------------------------------*/
void      DisplayMat (double a[30][30])
 {
   clmndiff=11;rowdiff=2;clmn0=2;row0=2;
   clrscr();
   clmn=clmn0;row=row0;
   cout<<"The Matrix is : ";
  {
   for(i=0;i<n;i++)
    {
     for(j=0;j<n;j++)
      {
// clmn=wherex();row=wherey();
       gotoxy(clmn,row);
```

```
        cout<<a[i][j];
        clmn+=clmndiff;
       }
        row+=rowdiff;
        clmn=clmn0;
        cout <<'\n';
       }  }
   cout <<"\n\nPress any key to CONTINUE ...";getch();
     }
/*-------------------------------------------------------*/
// Function :  DisplayMatSpaceMat()
/*-------------------------------------------------------*/
void      DisplayMatSpaceMat (void)
  {                        // DISPLAY MATRIX SpaceMat [n][2n]
   clrscr();
   clmn=clmn0;row=row0;
   cout <<"The Matrix Space is:";
    for(i=0;i<n;i++)
    { for(j=0;j<2*n;j++)
     {
      gotoxy(clmn,row);
      cout <<SpaceMat[i][j];  //setprecision(2)<<setw(6)<<
      clmn+=clmndiff;
     }
     row+=rowdiff;
     clmn=clmn0;
     cout<<'\n';
    }
  cout<<"\n\nPress any key to EXIT...";getch();
   }
/*-------------------------------------------------------*/
// Function : Add (a,b,c)
/*-------------------------------------------------------*/
void Add(double a[30][30],double b[30][30],double c[30][30])
   {
    for(i=0;i<n;i++)
      for(j=0;j<n;j++)
         c[i][j]=a[i][j]+b[i][j];
   }
/*-------------------------------------------------------*/
// Function :  Sub(a,b,c)
/*-------------------------------------------------------*/
void Sub(double a[30][30],double b[30][30],double c[30][30])
   {
    for(i=0;i<n;i++)
      for(j=0;j<n;j++)
         c[i][j]=a[i][j]-b[i][j];
   }
/*-------------------------------------------------------*/
// Function :  Mult (a,b,c)
/*-------------------------------------------------------*/
void Mult(double a[30][30],double b[30][30],double c[30][30])
   { double   p ;
     for(i=0;i<n;i++)
       for(j=0;j<n;j++)
```

```
        { p=0;
          for(k=0;k<n;k++) p+=a[i][k]*b[k][j];
          c[i][j]=p;
        } }
/*-------------------------------------------------------*/
// Function :  double Norm(a)
/*-------------------------------------------------------*/
double      Norm(double a[30][30])
    {
     double sum,value=fabs(a[0][0]);
     for(i=0;i<n;i++)
      {
        sum=0;
        for(j=0;j<n;j++) sum+=fabs(a[i][j]);
        value<sum?value=sum:value=value;
      }
     return value;
    }
/*-------------------------------------------------------*/
// Function :  Mult (scalar,matrix)
/*-------------------------------------------------------*/
void       Mult (double x,double a[30][30],double b[30][30])
  {
    for(i=0;i<n;i++)
      for(j=0;j<n;j++)
      b[i][j]=x*a[i][j];
  }
/*-------------------------------------------------------*/
// Function :  Root (a,b)
/*-------------------------------------------------------*/
void      Root (double a[30][30],double b[30][30])
  { int FLAG,COUNT=0;
    double   Tn[30][30],An[30][30],Bn[30][30],Cn[30][30],
             Dn[30][30],Gn[30][30] ;
    double p=sqrt(Norm(a)+1);
    Mult(p,I,Cn);                    //this is M0 in Cn
    do {
        Inverse(Cn,Tn);
        Mult(a,Tn,Dn);
        Add(Cn,Dn,Tn);
        Mult(0.5,Tn,Cn);              //this is M1 in Cn
        Assign(Cn,Bn);
        Mult(Bn,Bn,An);
        FLAG=Equal(a, An);
        COUNT++;
       }
    while ((FLAG==0) && (COUNT<=10));
    Assign(Cn,b);
    }
/*-------------------------------------------------------*/
int Equal(double a[30][30],double b[30][30])
    { int  FLAG=1;
      for(i=0;i<n;i++)
       for(j=0;j<n;j++)
        if (a[30][30]!=b[30][30]) FLAG=0;
        return(FLAG);
    }
/*-------------------------------------------------------*/
```