```
% ****************************************************************
%*                                                              *
%*                          S T A P M A T                       *
%*                                                              *
%*      An In-CORE SOLUTION STATIC ANALYSIS PROGRAM IN MATLAB   *
%*      Adapted from STAP90 (FORTRAN 90) for teaching purpose   *
%*                                                              *
%*  Computational Dynamics Group, School of Aerospace Engineering  *
%*  Tsinghua University, 2019.02.20                             *
%*                                                              *
%* ****************************************************************

% Set paths of functions
AddPath();

% Define Global Variables
global cdata;
global sdata;
cdata = ControlData;
sdata = SolutionData;

% Read InPut file
fname = 'stap90.in';              % Specify the file name
ReadFile(fname);

% Write basic data of program
WriteParasOut();

% Form the stiffness matrix
GetStiff();

% Triangularize stiffness matrix
Solve();

% Finalize
Finalize();

% ----------------------- Functions ----------------------------------

% Functions
% Add paths of functions
function AddPath()
clear;
close all;
clc;

addpath .\SRC\Initiation
addpath .\SRC\BasicData
addpath .\SRC\Mechanics
addpath .\SRC\Mechanics\Truss
addpath .\SRC\Solver
end

function Finalize()
global cdata;
TIM = cdata.TIM;
time = zeros(5, 1, 'double');
time(1) = etime(TIM(2,:), TIM(1,:));
time(2) = etime(TIM(3,:), TIM(2,:));
time(3) = etime(TIM(4,:), TIM(3,:));
time(4) = etime(TIM(5,:), TIM(4,:));
time(5) = etime(TIM(5,:), TIM(1,:));

fprintf(cdata.IOUT, ['\n\n' ...
    ' S O L U T I O N   T I M E   L O G   I N   S E C\n\n' ...
    '     TIME FOR INPUT PHASE  . . . . . = %12.2f\n' ...
    '     TIME FOR CALCULATION OF STIFFNESS MATRIX  . . . . = %12.2f\n' ...
    '     TIME FOR FACTORIZATION OF STIFFNESS MATRIX  . . . = %12.2f\n' ...
    '     TIME FOR LOAD CASE SOLUTIONS  . . . . . = %12.2f\n\n' ...
    '      T O T A L   S O L U T I O N   T I M E  . . . . . = %12.2f\n'], ...
    time(1), time(2), time(3), time(4),time(5));
```

```
72.
73.  fprintf(['\n' ...
74.      ' S O L U T I O N   T I M E   L O G   I N   S E C\n\n' ...
75.      '     TIME FOR INPUT PHASE  . . . . . . . . . . . . . . . = %12.2f\n' ...
76.      '     TIME FOR CALCULATION OF STIFFNESS MATRIX  . . . . = %12.2f\n' ...
77.      '     TIME FOR FACTORIZATION OF STIFFNESS MATRIX  . . . = %12.2f\n' ...
78.      '     TIME FOR LOAD CASE SOLUTIONS  . . . . . . . . . . = %12.2f\n\n' ...
79.      '      T O T A L   S O L U T I O N   T I M E  . . . . . = %12.2f\n'], ...
80.      time(1), time(2), time(3), time(4),time(5));
81.
82.  fclose(cdata.IIN);
83.  fclose(cdata.IOUT);
84.  end
85.
86.  %* ****************************************************************
87.  %* - Basic data class of STAPMAT                                   *
88.  %*                                                                 *
89.  %* - Purpose:                                                      *
90.  %*     Storing variables which control the running of STAPMAT      *
91.  %*                                                                 *
92.  %* - Programmed by:                                                *
93.  %*     LeiYang Zhao, Yan Liu,                                      *
94.  %*     Computational Dynamics Group, School of Aerospace           *
95.  %*     Engineering, Tsinghua University, 2019.02.20                *
96.  %*                                                                 *
97.  %* ****************************************************************
98.  classdef ControlData
99.      properties
100.         NUMNP;          % Total number of nodal points
101.                         % = 0 : Program stop
102.
103.         NPAR;           % Element group control data
104.                         %   NPAR(1) - Element type
105.                         %             1 : Truss element
106.                         %   NPAR(2) - Number of elements
107.                         %   NPAR(3) - Number of different sets of material
108.                         %             and cross-sectional constants
109.         NUMEG;          % Total number of element groups, > 0
110.         NLCASE;         % Number of load case (>0)
111.         LL;             % Load case number
112.         NLOAD;          % The number of concentrated loads applied in this load case
113.
114.         MODEX;          % Solution mode: 0 - data check only; 1 - execution
115.
116.         TIM;            % Timing information
117.         HED;            % Master heading information for usr in labeling the output
118.
119.         IIN;            % file pointer used for input
120.         IOUT;           % file pointer used for output
121.     end
122. end
123.
124. %* ****************************************************************
125. %* - Basic data class of STAPMAT                                   *
126. %*                                                                 *
127. %* - Purpose:                                                      *
128. %*     Storing variables used in solving process                   *
129. %*                                                                 *
130. %* - Programmed by:                                                *
131. %*     LeiYang Zhao, Yan Liu,                                      *
132. %*     Computational Dynamics Group, School of Aerospace           *
133. %*     Engineering, Tsinghua University, 2019.02.20                *
134. %*                                                                 *
135. %* ****************************************************************
136.
137. classdef SolutionData
138.     properties (Constant)
139.         % Gauss coord, 1D to 3D
140.         GC1 = double(0.0);
141.         GC2 = double([1/3,-1/3]);
142.         GC3 = double([sqrt(0.6), 0.0, -sqrt(0.6)]);
143.         % Gauss weight, 1D to 3D
144.         GW1 = double(2.0);
```

```
        GW2 = double([1.0, 1.0]);
        GW3 = double([5.0/9.0, 8.0/9.0, 5.0/9.0]);
    end
    properties
        % Basic data
        ID;       % int, ID(3, NUMNP), Boundary condition codes (0=free, 1=deleted)
        IDOrigin; % int, backups of ID after computing of NEQ
        X;        % double, X(NUMNP), X coordinates
        Y;        % double, Y(NUMNP), Y coordinates
        Z;        % double, Z(NUMNP), Z coordinates
        R;        % double, R(NEQ), Load vector
        NOD;      % int, NOD(NLOAD), Node number to which this load is applied (1~NUMNP)
        IDIRN;    % int, IDIRN(NLOAD), Degree of freedom number for this load component
                  %                     1 : X-direction;
                  %                     2 : Y-direction;
                  %                     3 : Z-direction;
        FLOAD;    % double, FLOAD(NLOAD), Magnitude of load



        % Element data
        NUME;     % int, number of elements
        NNODE;    % int, number of nodes in an element
        NINIP;    % int, number of integration points in an element
        NDOF;     % int, the DOF of displacement
        NSTIFF;   % int, the number of number in element stiffness matrix
        XYZ;      % double, XYZ(3*NNODE, NUME), element position

        InitCoord;  % double array, integration coordinates
        InitWeight; % double array, integration weights

        % Material data
        NUMMAT;     % int, the number of types of material
        E;          % double array, Young's Modulus
        nu;         % double array, possion ratio
        AREA;       % double array, cross-sectional constants
        MATP;       % int, MATP(NUME), types of elements

        % Solve data
        NEQ;      % int, Number of equations
        NWK;      % Number of matrix elements
        MK;       % Maximum half bandwidth
        MHT;      % int, MHT(NEQ), Vector of column heights
        LM;       % int, LM(6, NUME), Connectivity matrix
        MAXA;     % int, MAXA(NEQ)
        STIFF;    % double ,STIFF(NWK), store the elements of stiffness matrix

        % Result data
        DIS;      % double, DIS(NEQ, NLCASE), Displacement of nodes
        STRAIN;   % double, STRAIN(NEQ, NLCASE), Strain
        STRESS;   % double, STRESS(NEQ, NLCASE), Stress

    end
end

%* ****************************************************************
%* - Function of STAPMAT in initialization phase                   *
%*                                                                 *
%* - Purpose:                                                      *
%*     Read input file of STAPMAT                                  *
%*                                                                 *
%* - Call procedures:                                              *
%*     SRC/Initiation/ReadFile.m - InitBasicData()                 *
%*                                                                 *
%* - Called by :                                                   *
%*     stapmat.m                                                   *
%*                                                                 *
%* - Programmed by:                                                *
%*     LeiYang Zhao, Yan Liu,                                      *
%*     Computational Dynamics Group, School of Aerospace           *
%*     Engineering, Tsinghua University, 2019.02.21                *
%*                                                                 *
%* ****************************************************************
```

```
218.
219.  function ReadFile(fname)
220.  fname = strcat('.\Data\', fname);           % Deal the filename
221.
222.  % Get global class
223.  global cdata;
224.  global sdata;
225.
226.  % Open files
227.  cdata.IIN = fopen(fname, 'r');
228.
229.  % Begin Read input file
230.  fprintf('Input phase ...\n\n');
231.
232.  % the first time stamp
233.  cdata.TIM = zeros(5, 6, 'double');
234.  cdata.TIM(1,:) = clock;
235.
236.  IIN = cdata.IIN;
237.  %% Read Control data
238.  cdata.HED = fgetl(IIN);
239.
240.  tmp = str2num(fgetl(IIN));
241.  cdata.NUMNP = int64(tmp(1));
242.  cdata.NUMEG = int64(tmp(2));
243.  cdata.NLCASE = int64(tmp(3));
244.  cdata.MODEX = int64(tmp(4));
245.
246.  if (cdata.NUMNP == 0) return; end
247.
248.  %% Read nodal point data
249.  InitBasicData();
250.  % Define local variables to speed
251.  ID = sdata.ID; X = sdata.X; Y = sdata.Y; Z = sdata.Z;
252.  for i = 1:cdata.NUMNP
253.      tmp = str2num(fgetl(IIN));
254.      ID(1, i) = int64(tmp(2));
255.      ID(2, i) = int64(tmp(3));
256.      ID(3, i) = int64(tmp(4));
257.      X(i) = double(tmp(5));
258.      Y(i) = double(tmp(6));
259.      Z(i) = double(tmp(7));
260.  end
261.  sdata.ID = ID; sdata.X = X; sdata.Y = Y; sdata.Z = Z;
262.  %% Compute the number of equations
263.  sdata.IDOrigin = ID;
264.  NEQ = 0;
265.  for N=1:cdata.NUMNP
266.      for I=1:3
267.          if (ID(I,N) == 0)
268.              NEQ = NEQ + 1;
269.              ID(I,N) = NEQ;
270.          else
271.              ID(I,N) = 0;
272.          end
273.      end
274.  end
275.  sdata.ID = ID;
276.  sdata.NEQ = NEQ;
277.  %% Read load data
278.  % Init control data
279.  NLCASE = cdata.NLCASE;
280.  sdata.R = zeros(NEQ, NLCASE, 'double');
281.  R = sdata.R;
282.  % Read data
283.  for N = 1:cdata.NLCASE
284.      tmp = str2num(fgetl(IIN));
285.      cdata.LL = int64(tmp(1)); cdata.NLOAD = int64(tmp(2));
286.      NLOAD = cdata.NLOAD;
287.  %   Init load data
288.      sdata.NOD = zeros(NLOAD, 1, 'int64');
289.      sdata.IDIRN = zeros(NLOAD, 1, 'int64');
290.      sdata.FLOAD = zeros(NLOAD, 1, 'double');
```

```
    NOD = sdata.NOD; IDIRN = sdata.IDIRN; FLOAD = sdata.FLOAD;

%   Read load data
    for I = 1:NLOAD
        tmp = str2num(fgetl(IIN));
        NOD(I) = int64(tmp(1));
        IDIRN(I) = int64(tmp(2));
        FLOAD(I) = double(tmp(3));
    end
    if (cdata.MODEX == 0) return; end

%   Compute load vector
    for L = 1:NLOAD
        II = ID(IDIRN(L), NOD(L));
        if (II > 0) R(II, N) = R(II, N) + FLOAD(L); end
    end
    sdata.NOD = NOD; sdata.IDIRN = IDIRN; sdata.FLOAD = FLOAD; sdata.R = R;
end

end

%% Functions
% InitBasicData
function InitBasicData()
global cdata;
global sdata;

cdata.NPAR = zeros(10, 1, 'int64');

sdata.ID = zeros(3,cdata.NUMNP, 'int64');
sdata.X = zeros(cdata.NUMNP, 1, 'double');
sdata.Y = zeros(cdata.NUMNP, 1, 'double');
sdata.Z = zeros(cdata.NUMNP, 1, 'double');
end

%* ******************************************************************
%* - Function of STAPMAT in initialization phase                     *
%*                                                                   *
%* - Purpose:                                                        *
%*     Write parameters to output file of STAPMAT                    *
%*                                                                   *
%* - Call procedures: None                                           *
%*                                                                   *
%* - Called by :                                                     *
%*     stapmat.m                                                     *
%*                                                                   *
%* - Programmed by:                                                  *
%*     LeiYang Zhao, Yan Liu,                                        *
%*     Computational Dynamics Group, School of Aerospace             *
%*     Engineering, Tsinghua University, 2019.02.21                  *
%*                                                                   *
%* ******************************************************************

function WriteParasOut()
global cdata;
global sdata;
% Open file
cdata.IOUT = fopen('.\Data\STAPMAT.OUT', 'w');
IOUT = cdata.IOUT;

fprintf(IOUT, ['\n %s \n\n'...
    ' C O N T R O L   I N F O R M A T I O N\n\n'...
    '      NUMBER OF NODAL POINTS . . . . . (NUMNP)  = %10d \n' ...
    '      NUMBER OF ELEMENT GROUPS . . . . . (NUMEG)  = %10d \n' ...
    '      NUMBER OF LOAD CASES . . . . . (NLCASE) = %10d \n' ...
    '      SOLUTION MODE  . . . . . (MODEX)  = %10d \n' ...
    '         EQ.0, DATA CHECK \n' ...
    '         EQ.1, EXECUTION'], ...
    cdata.HED, cdata.NUMNP, cdata.NUMEG, cdata.NLCASE, cdata.MODEX);

% Write complete nodal data
ID = sdata.IDOrigin; X = sdata.X; Y = sdata.Y; Z = sdata.Z;
fprintf(IOUT, '\n\n\n N O D A L   P O I N T   D A T A \n\n');
```

```
fprintf(IOUT, '       NODE          BOUNDARY                        NODAL POINT\n');
fprintf(IOUT, '      NUMBER     CONDITION  CODES                     COORDINATES\n');

fprintf(IOUT, '                    X    Y    Z              X           Y           Z\n'
for i = 1:cdata.NUMNP
    fprintf(IOUT, '%10d      %5d%5d%5d      %13.3f%13.3f%13.3f\n', ...
        i, ID(1,i), ID(2,i), ID(3,i), X(i), Y(i), Z(i));
end
sdata.IDOrigin = 0;                              % Delete old ID array

% Write equation numbers
ID = sdata.ID;
fprintf(IOUT, '\n\n EQUATION NUMBERS\n');
fprintf(IOUT, '\n        NODE         DEGREES OF FREEDOM\n');
fprintf(IOUT, '       NUMBER\n');
fprintf(IOUT, '          N                 X         Y         Z\n');
for N=1:cdata.NUMNP
    fprintf(IOUT, ' %10d         %10d%10d%10d\n', ...
        N, ID(1,N), ID(2,N), ID(3,N));
end

% Write the load vector
% Only the first load vector
fprintf(IOUT, '\n\n L O A D   C A S E   D A T A\n');
LL = cdata.LL; NLOAD = cdata.NLOAD;
NOD = sdata.NOD; IDIRN = sdata.IDIRN; FLOAD = sdata.FLOAD;
for I = 1:1 %cdata.NLCASE
    fprintf(IOUT, '\n     LOAD CASE NUMBER . . . . . . . = %10d\n', LL);
    fprintf(IOUT, '     NUMBER OF CONCENTRATED LOADS . = %10d\n', NLOAD);

    if (LL ~= I)
        error(' *** ERROR *** LOAD CASES ARE NOT IN ORDER');
    end
    fprintf(IOUT, '\n\n        NODE       DIRECTION      LOAD\n');
    fprintf(IOUT, '       NUMBER                    MAGNITUDE\n');

    for N = 1:NLOAD(I)
        fprintf(IOUT,'%10d         %4d       %12.5e',...
            NOD(N), IDIRN(N), FLOAD(N));
    end
end
end

%* *****************************************************************
%* - Function of STAPMAT in stiffness phase                        *
%*                                                                 *
%* - Purpose:                                                      *
%*     Forming the stiffness matrix                                *
%*                                                                 *
%* - Call procedures:                                              *
%*     SRC/Mechanics/Truss/TrussStiff.m - TrussStiff()             *
%*                                                                 *
%* - Called by :                                                   *
%*     stapmat.m                                                   *
%*                                                                 *
%* - Programmed by:                                                *
%*     LeiYang Zhao, Yan Liu,                                      *
%*     Computational Dynamics Group, School of Aerospace           *
%*     Engineering, Tsinghua University, 2019.02.21                *
%*                                                                 *
%* *****************************************************************

function GetStiff()
% Get global variables
global cdata;

% Read the type of element
IIN = cdata.IIN;
IOUT = cdata.IOUT;
fprintf(IOUT, '\n\n E L E M E N T   G R O U P   D A T A\n');

for N = 1:cdata.NUMEG
```

```
436.     tmp = str2num(fgetl(IIN));
437.     for I = 1:length(tmp) cdata.NPAR(I) = tmp(I); end
438.
439.     fprintf(IOUT, '\n\n E L E M E N T   D E F I N I T I O N\n');
440.     fprintf(IOUT, ['\n ELEMENT TYPE  . . . . . .
    ( NPAR(1) ) . . = %10d\n' ...
441.         '     EQ.1, TRUSS ELEMENTS\n' ...
442.         '     EQ.2, ELEMENTS CURRENTLY\n' ...
443.         '     EQ.3, NOT AVAILABLE\n' ...
444.         ' NUMBER OF ELEMENTS. . . . . .( NPAR(2) ) . . = %10d\n'], ...
445.         cdata.NPAR(1), cdata.NPAR(2));
446.
447. %   Different kinds of element
448.     NPAR1 = cdata.NPAR(1);
449.     if (NPAR1 == 1) TrussStiff()
450.     else error(' *** ERROR *** No Such Element'); end
451.
452. end
453.
454. end
455.
456. %* *****************************************************************
457. %* - Function of STAPMAT in stiffness phase                        *
458. %*                                                                 *
459. %* - Purpose:                                                      *
460. %*     Compute the stiffness matrix of truss                       *
461. %*                                                                 *
462. %* - Call procedures:                                              *
463. %*     TrussStiff.m - InitTruss()                                  *
464. %*     ./ReadTruss.m - ReadTruss()                                 *
465. %*     SRC/Mechanics/Addres.m - Addres()                           *
466. %*                                                                 *
467. %* - Called by :                                                  *
468. %*     SRC/Mechanics/GetStiff.m                                    *
469. %*                                                                 *
470. %* - Programmed by:                                               *
471. %*     LeiYang Zhao, Yan Liu,                                      *
472. %*     Computational Dynamics Group, School of Aerospace           *
473. %*     Engineering, Tsinghua University, 2019.02.21                *
474. %*                                                                 *
475. %* *****************************************************************
476.
477. function TrussStiff()
478.
479. % Init variables of the element
480. InitTruss();
481.
482. % Read Material and Elements
483. ReadTruss();
484.
485. fprintf('Solution phase ...\n\n');
486.
487. % calculate addresses of diagonal elements
488. Addres();
489.
490. % Data check Or Solve
491. global cdata;
492. if (cdata.MODEX == 0)
493.     cdata.TIM(3,:) = clock;
494.     cdata.TIM(4,:) = clock;
495.     cdata.TIM(5,:) = clock;
496.     return;
497. end
498.
499. % Assemble structure stiffness matrix
500. Assemble();
501.
502.
503.
504.
505. end
506.
507. % ---------------------- Functions ----------------------------------
```

```
508.
509.  % Init parameters of truss element
510.  function InitTruss()
511.  global sdata;
512.  sdata.NNODE = 2;
513.  sdata.NDOF = 3;
514.
515.  end
516.
517.  % Assemble structure stiffness matrix
518.  function Assemble()
519.  global sdata;
520.  global cdata;
521.  S = zeros(6, 6, 'double');
522.  ST = zeros(6, 1, 'double');
523.  sdata.STIFF = zeros(sdata.NWK, 1, 'double');
524.
525.  NUME = sdata.NUME; MATP = sdata.MATP; XYZ = sdata.XYZ;
526.  E = sdata.E; AREA = sdata.AREA; LM = sdata.LM;
527.  for N = 1:NUME
528.      MTYPE = MATP(N);
529.
530.  %   compute the length of truss element
531.      DX = XYZ(1, N) - XYZ(4, N);
532.      DY = XYZ(2, N) - XYZ(5, N);
533.      DZ = XYZ(3, N) - XYZ(6, N);
534.      XL2 = DX*DX + DY*DY + DZ*DZ;
535.      XL = sqrt(XL2);
536.
537.      XX = E(MTYPE) * AREA(MTYPE) * XL;
538.
539.      ST(1) = DX / XL2;
540.      ST(2) = DY / XL2;
541.      ST(3) = DZ / XL2;
542.      ST(4) = -ST(1); ST(5) = -ST(2); ST(6) = -ST(3);
543.
544.      for J = 1:6
545.          YY = ST(J) * XX;
546.          for I = 1:J
547.              S(I, J) = ST(I)*YY;
548.          end
549.      end
550.
551.  %   SRC/Mechanics/ADDBAN.m
552.      ADDBAN(S, LM(:, N));
553.
554.  end
555.
556.  % The third time stamp
557.  cdata.TIM(3, :) = clock;
558.
559.  end
560.
561.  %* *****************************************************************
562.  %* - Function of STAPMAT in stiffness phase                        *
563.  %*                                                                 *
564.  %* - Purpose:                                                      *
565.  %*     Read the element information of truss                       *
566.  %*                                                                 *
567.  %* - Call procedures:                                              *
568.  %*     ReadTruss.m - ReadMaterial()                                *
569.  %*     ReadTruss.m - ReadElements()                                *
570.  %*                                                                 *
571.  %* - Called by :                                                   *
572.  %*     ./TrussStiff.m                                              *
573.  %*                                                                 *
574.  %* - Programmed by:                                                *
575.  %*     LeiYang Zhao, Yan Liu,                                      *
576.  %*     Computational Dynamics Group, School of Aerospace           *
577.  %*     Engineering, Tsinghua University, 2019.02.21                *
578.  %*                                                                 *
579.  %* *****************************************************************
580.
```

```
function ReadTruss()

% Read Material information
ReadMaterial()

% Read Element information
ReadElements()

% the second time stamp
global cdata;
cdata.TIM(2,:) = clock;

end

% ---------------------- Functions ----------------------------------
% Read Material information
function ReadMaterial()

global cdata;
global sdata;
% Get file pointers
IIN = cdata.IIN;
IOUT = cdata.IOUT;

if (cdata.NPAR(3) == 0) cdata.NPAR(3) = 1; end
fprintf(IOUT, '\n M A T E R I A L   D E F I N I T I O N\n');
fprintf(IOUT, '\n NUMBER OF DIFFERENT SETS OF MATERIAL\n');
fprintf(IOUT, ' AND CROSS-SECTIONAL  CONSTANTS  . . . .( NPAR(3) ) . . = %10d\n', ...
    cdata.NPAR(3));
fprintf(IOUT, '  SET       YOUNG''S     CROSS-SECTIONAL\n');
fprintf(IOUT, ' NUMBER     MODULUS          AREA\n');
fprintf(IOUT, '               E              A\n');


% Read material datas
sdata.NUME = cdata.NPAR(2);
sdata.NUMMAT = cdata.NPAR(3);
NUMMAT = cdata.NPAR(3);
sdata.E = zeros(NUMMAT, 1, 'double');
sdata.AREA = zeros(NUMMAT, 1, 'double');
for I = 1:cdata.NPAR(3)
    tmp = str2num(fgetl(IIN));
    N = round(tmp(1));
    sdata.E(N) = tmp(2);
    sdata.AREA(N) = tmp(3);
    fprintf(IOUT, '%5d    %12.5e  %14.6e\n', N, tmp(2), tmp(3));
end

end

% Read elements information
function ReadElements()

global cdata;
global sdata;

% Get file pointer
IIN = cdata.IIN;
IOUT = cdata.IOUT;

fprintf(IOUT, '\n\n E L E M E N T   I N F O R M A T I O N\n');
fprintf(IOUT, '\n      ELEMENT          NODE          NODE       MATERIAL\n');
fprintf(IOUT, '      NUMBER-N           I             J       SET NUMBER\n');

% Get Position data
NUME = cdata.NPAR(2);
sdata.XYZ = zeros(6, NUME, 'double');
sdata.MATP = zeros(NUME, 1, 'int64');                % the type of material
sdata.LM = zeros(6, NUME, 'double');                 % connectivity matrix
sdata.MHT = zeros(sdata.NEQ, 1, 'int64');
X = sdata.X; Y = sdata.Y; Z = sdata.Z; ID = sdata.ID;
XYZ = sdata.XYZ; MATP = sdata.MATP; LM = sdata.LM;

```

```
for N = 1:NUME
    tmp = str2num(fgetl(IIN));
    I = round(tmp(2));
    J = round(tmp(3));
    MTYPE = round(tmp(4));

%   Save element information
    XYZ(1, N) = X(I);
    XYZ(2, N) = Y(I);
    XYZ(3, N) = Z(I);
    XYZ(4, N) = X(J);
    XYZ(5, N) = Y(J);
    XYZ(6, N) = Z(J);
    MATP(N) = MTYPE;

    fprintf(IOUT, '%10d      %10d    %10d       %5d\n', N, I, J, MTYPE);

%   Compute connectivity matrix
    LM(1, N) = ID(1, I);
    LM(4, N) = ID(1, J);
    LM(2, N) = ID(2, I);
    LM(5, N) = ID(2, J);
    LM(3, N) = ID(3, I);
    LM(6, N) = ID(3, J);

%   Updata column heights and bandwidth
    ColHt(LM(:, N))
end
sdata.XYZ = XYZ; sdata.MATP = MATP; sdata.LM = LM;

% Clear the memory of X, Y, Z
sdata.X = double(0);
sdata.Y = double(0);
sdata.Z = double(0);

end

%* ******************************************************************
%* - Function of STAPMAT in solver phase                            *
%*                                                                  *
%* - Purpose:                                                       *
%*     To calculate stresses                                        *
%*                                                                  *
%* - Call procedures: None                                          *
%*                                                                  *
%* - Called by :                                                    *
%*     SRC/Solver/GetStress.m                                       *
%*                                                                  *
%* - Programmed by:                                                 *
%*     LeiYang Zhao, Yan Liu,                                       *
%*     Computational Dynamics Group, School of Aerospace            *
%*     Engineering, Tsinghua University, 2019.02.22                 *
%*                                                                  *
%* ******************************************************************

function TrussStress(NUM, NG)

% Get global data
global cdata;
global sdata;

IOUT = cdata.IOUT;
NUME = sdata.NUME; MATP = sdata.MATP; XYZ = sdata.XYZ;
E = sdata.E; AREA = sdata.AREA; LM = sdata.LM;
U = sdata.DIS(:, NUM);

fprintf(IOUT, ['\n\n  S T R E S S  C A L C U L A T I O N S  F O R  ' ...
    'E L E M E N T  G R O U P %4d\n\n' ...
    '       ELEMENT             FORCE            STRESS\n' ...
    '       NUMBER\n'], NG);

for N = 1:NUME
    MTYPE = MATP(N);
```

```
727.
728.  %   compute the length of truss element
729.      DX = XYZ(1, N) - XYZ(4, N);
730.      DY = XYZ(2, N) - XYZ(5, N);
731.      DZ = XYZ(3, N) - XYZ(6, N);
732.      XL2 = DX*DX + DY*DY + DZ*DZ;
733.
734.      ST(1) = DX / XL2 * E(MTYPE);
735.      ST(2) = DY / XL2 * E(MTYPE);
736.      ST(3) = DZ / XL2 * E(MTYPE);
737.      ST(4) = -ST(1); ST(5) = -ST(2); ST(6) = -ST(3);
738.
739.      STR = 0.0;
740.
741.      if (LM(1, N) > 0) STR = STR + ST(1)*U(LM(1, N)); end
742.      if (LM(2, N) > 0) STR = STR + ST(2)*U(LM(2, N)); end
743.      if (LM(3, N) > 0) STR = STR + ST(3)*U(LM(3, N)); end
744.      if (LM(4, N) > 0) STR = STR + ST(4)*U(LM(4, N)); end
745.      if (LM(5, N) > 0) STR = STR + ST(5)*U(LM(5, N)); end
746.      if (LM(6, N) > 0) STR = STR + ST(6)*U(LM(6, N)); end
747.
748.      P = STR*AREA(MTYPE);
749.
750.      fprintf(IOUT, ' %10d           %13.6e    %13.6e\n', N, P, STR);
751.  end
752.
753.  end
754.
755.  %* *****************************************************************
756.  %* - Function of STAPMAT in stiffness phase                        *
757.  %*                                                                 *
758.  %* - Purpose:                                                      *
759.  %*     To calculate addresses of diagonal elements in banded       *
760.  %*     matrix whose column heights are known                        *
761.  %*                                                                 *
762.  %* - Call procedures: None                                         *
763.  %*                                                                 *
764.  %* - Called by :                                                   *
765.  %*     ./Truss/TrussStiff.m                                        *
766.  %*                                                                 *
767.  %* - Programmed in Fortran 90 by Xiong Zhang                       *
768.  %*                                                                 *
769.  %* - Adapted to Matlab by:                                         *
770.  %*     LeiYang Zhao, Yan Liu, Computational Dynamics Group,        *
771.  %*     School of Aerospace Engineering, Tsinghua University,       *
772.  %*     2019.02.22                                                  *
773.  %*                                                                 *
774.  %* *****************************************************************
775.
776.  function Addres()
777.
778.  % Get global data
779.  global sdata;
780.  global cdata;
781.
782.  NEQ = sdata.NEQ; MHT = sdata.MHT;
783.  sdata.MAXA = zeros(NEQ+1, 1, 'int64');
784.  MAXA = sdata.MAXA;
785.
786.  MAXA(1) = 1;
787.  MAXA(2) = 2;
788.  MK = 0;
789.
790.  if (NEQ > 1)
791.      for I = 2:NEQ
792.          if (MHT(I) > MK) MK = MHT(I); end
793.          MAXA(I+1) = MAXA(I) + MHT(I) + 1;
794.      end
795.  end
796.
797.  sdata.MK = MK + 1;
798.  sdata.NWK = MAXA(NEQ+1) - MAXA(1);
799.  sdata.MAXA = MAXA;
```

```
800.
801.  % Write total system data
802.  MM = round(sdata.NWK / NEQ);
803.  fprintf(cdata.IOUT, ['\n\n  TOTAL SYSTEM DATA\n\n' ...
804.      '     NUMBER OF EQUATIONS . . . . . . . . . . . . . .(NEQ) = %10d\n' ...
805.      '     NUMBER OF MATRIX ELEMENTS . . . . . . . . . . .(NWK) = %10d\n' ...
806.      '     MAXIMUM HALF BANDWIDTH  . . . . . . . . . . . .(MK ) = %10d\n' ...
807.      '     MEAN HALF BANDWIDTH . . . . . . . . . . . . . .(MM ) = %10d\n'], ...
808.      NEQ, sdata.NWK, sdata.MK, MM);
809.
810.  end
811.
812.  %* *****************************************************************
813.  %* - Function of STAPMAT in stiffness phase                        *
814.  %*                                                                 *
815.  %* - Purpose:                                                      *
816.  %*     To calculate column heights                                 *
817.  %*                                                                 *
818.  %* - Call procedures: None                                         *
819.  %*                                                                 *
820.  %* - Called by :                                                   *
821.  %*     ./Truss/ReadTruss.m                                         *
822.  %*                                                                 *
823.  %* - Programmed in Fortran 90 by Xiong Zhang                       *
824.  %*                                                                 *
825.  %* - Adapted to Matlab by:                                         *
826.  %*     LeiYang Zhao, Yan Liu, Computational Dynamics Group,        *
827.  %*     School of Aerospace Engineering, Tsinghua University,       *
828.  %*     2019.02.22                                                  *
829.  %*                                                                 *
830.  %* *****************************************************************
831.
832.  function ColHt(LM)
833.
834.  % Get global data
835.  global sdata;
836.  MHT = sdata.MHT;
837.  LS = min(LM(LM ~= 0));
838.  ND = sdata.NDOF * sdata.NNODE;
839.  for I = 1:ND
840.      II = LM(I);
841.      if (II ~= 0)
842.          ME = II - LS;
843.          if (ME > MHT(II)) MHT(II) = ME; end
844.      end
845.  end
846.
847.  sdata.MHT = MHT;
848.
849.  end
850.
851.  %* *****************************************************************
852.  %* - Function of STAPMAT in stiffness phase                        *
853.  %*                                                                 *
854.  %* - Purpose:                                                      *
855.  %*     To assemble element stiffness into global stiffness         *
856.  %*                                                                 *
857.  %* - Call procedures: None                                         *
858.  %*                                                                 *
859.  %* - Called by :                                                   *
860.  %*     ./Truss/ReadTruss.m - Assemble()                            *
861.  %*                                                                 *
862.  %* - Programmed in Fortran 90 by Xiong Zhang                       *
863.  %*                                                                 *
864.  %* - Adapted to Matlab by:                                         *
865.  %*     LeiYang Zhao, Yan Liu, Computational Dynamics Group,        *
866.  %*     School of Aerospace Engineering, Tsinghua University,       *
867.  %*     2019.02.22                                                  *
868.  %*                                                                 *
869.  %* *****************************************************************
870.
871.  function ADDBAN(S, LM)
872.
```

```
% Get global data
global sdata;
MAXA = sdata.MAXA;
STIFF = sdata.STIFF;
ND = sdata.NDOF * sdata.NNODE;
for J = 1:ND
    JJ = LM(J);
    if (JJ > 0)
        for I = 1:J
            II = LM(I);
            if (II > 0)
                if (JJ > II) KK = MAXA(JJ) + JJ - II;
                else KK = MAXA(II) + II - JJ; end
                STIFF(KK) = STIFF(KK) + S(I, J);
            end
        end
    end
end

sdata.STIFF = STIFF;

end

%* ******************************************************************
%* - Function of STAPMAT in Solver phase                            *
%*                                                                  *
%* - Purpose:                                                       *
%*     To solve finite element static equilibrium equations        *
%*                                                                  *
%* - Call procedures:                                               *
%*     ./LDLTFactor.m            - LDLTFactor()                     *
%*     Solve.m                   - Stiff2Sparse()                   *
%*     ./ColSol.m                - ColSol()                         *
%*     Solve.m                   - WriteDis()                       *
%*     SRC/Mechanics/GetStress.m - GetStress()                      *
%*                                                                  *
%* - Called by :                                                    *
%*     stapmat.m                                                    *
%*                                                                  *
%* - Programmed by:                                                 *
%*     LeiYang Zhao, Yan Liu,                                       *
%*     Computational Dynamics Group, School of Aerospace            *
%*     Engineering, Tsinghua University, 2019.02.22                 *
%*                                                                  *
%* ******************************************************************

function Solve()

global cdata;
global sdata;

NEQ = sdata.NEQ;
NLCASE = cdata.NLCASE;
MODEX = cdata.MODEX;
sdata.DIS = zeros(NEQ, NLCASE, 'double');
sdata.STRAIN = zeros(NEQ, NLCASE, 'double');
sdata.STRESS = zeros(NEQ, NLCASE, 'double');

% The pre-process of Solution
% MODEX = 1, LDLTFactor() - ColSol()
% MODEX = 2, Stiff2Sparse() - sdata.SPSTIFF \ Sdata.R(:, L)
if (MODEX == 1) LDLTFactor();
else SPSTIFF = Stiff2Sparse(); end

cdata.TIM(4,:) = clock;

% Solve
for L = 1:NLCASE

%   Solve the equilibrium equations to calculate the displacements
    if (MODEX == 1) ColSol(L);
    else sdata.DIS(:,L) = SPSTIFF \ sdata.R(:,L); end

```

```
946.  %   Print displacements
947.      WriteDis(L);
948.
949.  %   Calculation of stresses
950.      GetStress(L);
951.
952.  end
953.
954.  cdata.TIM(5, :) = clock;
955.
956.  end
957.
958.  % ---------------------- Functions ----------------------------------
959.
960.  % Convert the stiff vector to a sparse stiff matrix
961.  function SPSTIFF = Stiff2Sparse()
962.
963.  global sdata;
964.  A = sdata.STIFF; MAXA = sdata.MAXA; NEQ = sdata.NEQ; NWK = sdata.NWK;
965.  IIndex = zeros(NWK*2-NEQ, 1);
966.  JIndex = IIndex;
967.  STIFF = IIndex;
968.
969.  NUM = 1;
970.  NUMC = 0;
971.  for N = 1:NEQ
972.      KU = MAXA(N + 1) - MAXA(N);
973.      for L = 1:KU
974.          IIndex(NUM) = N;
975.          JIndex(NUM) = N - L + 1;
976.          STIFF(NUM) = A(NUM);
977.          NUM = NUM + 1;
978.          if (L == 1) NUMC = NUMC + 1;continue; end
979.          SYMN = NUM-1 - NUMC + NWK;
980.          IIndex(SYMN) = N - L + 1;
981.          JIndex(SYMN) = N;
982.          STIFF(SYMN) = A(NUM-1);
983.      end
984.  end
985.
986.  SPSTIFF = sparse(IIndex, JIndex, STIFF, NEQ, NEQ);
987.  end
988.
989.  % Print Displacements
990.  function WriteDis(NUM)
991.
992.  % Get global data
993.  global cdata;
994.  global sdata;
995.  IOUT = cdata.IOUT;
996.  NUMNP = cdata.NUMNP;
997.  DIS = sdata.DIS(:, NUM); ID = sdata.ID;
998.
999.  fprintf(IOUT, '\n\n LOAD CASE %3d', NUM);
1000. fprintf(IOUT, ['\n\n D I S P L A C E M E N T S\n' ...
1001.     '\n       NODE           X-DISPLACEMENT    Y-DISPLACEMENT    Z-DISPLACEMENT\n']);
1002.
1003. D = zeros(3, 1, 'double');
1004. for II = 1:NUMNP
1005.     D(:) = 0;
1006.     if (ID(1, II) ~= 0) D(1) = DIS(ID(1, II)); end
1007.     if (ID(2, II) ~= 0) D(2) = DIS(ID(2, II)); end
1008.     if (ID(3, II) ~= 0) D(3) = DIS(ID(3, II)); end
1009.
1010.     fprintf(IOUT, ' %10d        %18.6e%18.6e%18.6e\n', II, D(1), D(2), D(3));
1011. end
1012.
1013. end
1014.
1015. %* *************************************************************
1016. %* - Function of STAPMAT in Solver phase                        *
1017. %*                                                              *
1018. %* - Purpose:                                                   *
```

```
%*     Perform L*D*L(T) factorization of stiffness matrix          *
%*                                                                 *
%* - Call procedures: None                                         *
%*                                                                 *
%* - Called by :                                                   *
%*     ./Solve.m                                                   *
%*                                                                 *
%* - Programmed in Fortran 90 by Xiong Zhang                       *
%*                                                                 *
%* - Adapted to Matlab by:                                         *
%*     Yan Liu, Computational Dynamics Group, School of Aerospace  *
%*     Engineering, Tsinghua University, 2019.02.22                *
%*                                                                 *
%* *****************************************************************

function LDLTFactor()

% Get global data
global sdata;

A = sdata.STIFF;
MAXA = sdata.MAXA; NEQ = sdata.NEQ;

for N = 1:NEQ
    KN = MAXA(N);
    KL = KN + 1;
    KU = MAXA(N + 1) - 1;
    KH = KU - KL;

    if (KH > 0)
        K = N - KH;
        IC = 0;
        KLT = KU;
        for J = 1:KH
            IC = IC + 1;
            KLT = KLT - 1;
            KI = MAXA(K);
            ND = MAXA(K+1) - KI - 1;
            if (ND > 0)
                KK = min(IC, ND);
                C = 0.0;
                for L = 1:KK C = C+A(KI+L)*A(KLT+L); end
                A(KLT) = A(KLT) - C;
            end
            K = K + 1;
        end
    end

    if (KH >= 0)
        K = N;
        B = 0.0;
        for KK = KL:KU
            K = K - 1;
            KI = MAXA(K);
            C = A(KK) / A(KI);
            B = B + C*A(KK);
            A(KK) = C;
        end
        A(KN) = A(KN) - B;
    end

    if (A(KN) <= 0)
        error(['STOP - Stiffness matrix is not positive definite\n' ...
            'Nonpositive number for equation %8d is %20.12e\n'], N, A(KN));
    end
end

sdata.STIFF = A;

end

%* *****************************************************************
%* - Function of STAPMAT in Solver phase                           *
```

```
%*                                                                     *
%* - Purpose:                                                          *
%*     To solve finite element static equilibrium equations           *
%*                                                                     *
%* - Call procedures: None                                             *
%*                                                                     *
%* - Called by :                                                       *
%*     ./Solve.m                                                       *
%*                                                                     *
%* - Programmed in Fortran 90 by Xiong Zhang                           *
%*                                                                     *
%* - Adapted to Matlab by:                                             *
%*     Yan Liu, Computational Dynamics Group, School of Aerospace      *
%*     Engineering, Tsinghua University, 2019.02.22                    *
%*                                                                     *
%* ********************************************************************

function ColSol(NUM)

% Get global data
global sdata;
A = sdata.STIFF; MAXA = sdata.MAXA; R = sdata.R(:,NUM);
NEQ = sdata.NEQ; NWK = sdata.NWK;
NNM = NEQ + 1;

% Reduce right-hand-side load vector
for N = 1:NEQ
    KL = MAXA(N) + 1;
    KU = MAXA(N+1) - 1;
    if (KU-KL >= 0)
        K = N;
        C = 0.0;
        for KK = KL:KU
            K = K - 1;
            C = C + A(KK) * R(K);
        end
        R(N) = R(N) - C;
    end
end

% Back-Substitute
for N = 1:NEQ
    K = MAXA(N);
    R(N) = R(N) / A(K);
end

if (NEQ == 1) return; end;

N = NEQ;
for L = 2:NEQ
    KL = MAXA(N) + 1;
    KU = MAXA(N+1) - 1;
    if (KU-KL >= 0)
        K = N;
        for KK = KL:KU
            K = K - 1;
            R(K) = R(K) - A(KK)*R(N);
        end
    end
    N = N - 1;
end

sdata.DIS(:, NUM) = R(:);

end

%* ********************************************************************
%* - Function of STAPMAT in Solver phase                               *
%*                                                                     *
%* - Purpose:                                                          *
%*     Calculation of strain and stress                                *
%*                                                                     *
%* - Call procedures:                                                  *
```

```
%*     SRC/Mechanics/Truss/TrussStress.m   - TrussStress()         *
%*                                                                 *
%* - Called by :                                                   *
%*     ./Solver.m                                                  *
%*                                                                 *
%* - Programmed by:                                                *
%*     LeiYang Zhao, Yan Liu,                                      *
%*     Computational Dynamics Group, School of Aerospace           *
%*     Engineering, Tsinghua University, 2019.02.22                *
%*                                                                 *
%* *****************************************************************

function GetStress(NUM)

% Different type of element
global cdata;
NUMEG = cdata.NUMEG;
IOUT = cdata.IOUT;

for N = 1:NUMEG
    NPAR1 = cdata.NPAR(1);
    if (NPAR1 == 1) TrussStress(NUM, N)
    else error(' *** ERROR *** No Such Element'); end
end

end
```