

```

1  frame05. f90 のソースプログラムのソースコード(Ver 21MRZ2013)を掲げておく。
2  @プログラムの行を特定するには、ページ左端にある行番号を活用できる。
3
4  !23456##### <frame05> for advanced matrix method : 18 August 2008 #####
5      MODULE GLOBAL05 !既存の大域変数をパッケージ化したモジュール
6  !-----IMPLICIT NONE でも 必ず暗黙の型宣言にしたがうこと!
7      IMPLICIT INTEGER (I-N), REAL (8) (A-H, O-Z)
8      CHARACTER ELE_LIB*7 !使用する要素コードの文字列 (7 文字)
9      CHARACTER CODE*2    !入力する座標の座標系(XY, PL) (2 文字)
10 !----- (SYSTEM)  整数型 大域変数の初期設定
11     INTEGER :: MENTS ! 要素の総数
12     INTEGER :: NODES !  節点の総数
13     INTEGER :: NSECTS !   使用する断面の種類の数
14     INTEGER :: MNODES !   一要素あたりの節点の数
15     INTEGER :: MDOFS !    一要素あたりの節点自由度の数
16     INTEGER :: INPNEQ ! =0 (default) dof#をプログラムの中で自動計算する
17     !                      =1      dof#を入力データファイルで与える
18     INTEGER :: INWAIT ! =0(default)/1 入力データ読込中に WAIT をかける/かけない
19 !----- 隠しコマンド類 -----
20     INTEGER :: IPHIDN !=0/1 隠しコマンド類を出力しない/する
21     INTEGER :: IPELEK !=0/1 要素剛性行列を出力しない/する
22     INTEGER :: IPSYSK !=0/1 SYSK を出力しない/する
23     INTEGER :: IPPVEC !=0/1 PVEC(RHS) を出力しない/する
24     INTEGER :: IPSOLV !=0/1 TTLDOF(RHS) を出力しない/する
25     INTEGER :: ICGINI !=0/1 初期形状の Win3D ファイルを 作らない/作る
26     INTEGER :: ICGDEF !=0/1 変形形状の Win3D ファイルを 作らない/作る
27     INTEGER :: ICGXLS !=0/1 図形データである XLS ファイルを 作らない/作る 27/JAN/2005
28     INTEGER :: NCGDEF !=10/100 はり要素の変形曲線の描画区分数
29     REAL (8) :: SCGDEF !=1./10./100./1000. 並進変位の倍率
30 !-----MDOFS:要素あたりの自由度数-----
31     REAL (8), DIMENSION (:, :), ALLOCATABLE :: ELEK
32     !          ! ELEK (MDOFS, MDOFS) 要素剛性行列
33     REAL (8), DIMENSION (:, :), ALLOCATABLE :: PQELE    ! only for B2D_LIN
34     !          ! PQELE (MENTS, MDOFS) 等価節点断面力
35     REAL (8), DIMENSION (:, :), ALLOCATABLE :: BALPQ    ! only for B2D_LIN
36     !          ! BALPQ (MENTS, 4) 分布荷重強度 (Global) PA, QA, PB, QB
37     REAL (8), DIMENSION (:, :), ALLOCATABLE :: CALPQ    ! only for B2D_LIN
38     !          ! CALPQ (MENTS, 4) 分布荷重強度 (Local)  pa, qa, pb, qb
39 !----- (NODE) -----
40     REAL (8), DIMENSION (:, :), ALLOCATABLE :: UVWNOD ! UVWNOD (NODES, 2) for CG

```

```

41      REAL(8), DIMENSION (:,:), ALLOCATABLE :: XYZNOD ! XYZNOD(NODES, 2)
42      !----- (ELEM) ---各要素を定義する節点, 要素の断面種類番号
43      INTEGER, DIMENSION (:,:), ALLOCATABLE :: NODELE ! NODELE(MENTS, MNODES)
44      INTEGER, DIMENSION (:), ALLOCATABLE :: MATSEC ! MATSEC(MENTS)
45      !---- (DOF) ---節点変位の拘束条件: 総自由度数 NDOFS ---
46      INTEGER :: NDOFS ! 系全体の自由度の総数
47      INTEGER :: LSKYMAX ! for SYSK(LSKYMAX) when skyline used
48      INTEGER, DIMENSION (:), ALLOCATABLE :: LSKYPVT ! LSKYPVT(NDOFS)
49      !-----
50      INTEGER, DIMENSION (:,:), ALLOCATABLE :: NEQNOD ! NEQNOD(NODES, 3)      2D!
51      INTEGER, DIMENSION (:,:), ALLOCATABLE :: NEQELE ! NEQELE(MENTS, MDOFS)
52      !-----
53      REAL(8), DIMENSION (:), ALLOCATABLE, TARGET :: SYSK ! 剛性行列 SYSK(LSKYMAX)
54      REAL(8), DIMENSION (:), ALLOCATABLE, TARGET :: RHS ! 右辺項 RHS(NDOFS)
55      REAL(8), DIMENSION (:), ALLOCATABLE :: TTLDOF ! 変位全量 TTLDOF(NDOFS)
56      !----- (SECTYP) EA & EI only -----
57      REAL(8), DIMENSION (:,:), ALLOCATABLE :: SECTYP ! SECTYP(NSECTS, 2)
58      !----- (LOAD) -----
59      REAL(8), DIMENSION (:), ALLOCATABLE :: PVEC ! PVEC(NDOFS)
60      INTEGER :: IAFDFL ! frame99_out open/close control
61      !----- (QUIT) -----
62      END MODULE GLOBAL05
63      !*****
64      ! main control program <frame05> as of 18 August 2008
65      !*****
66      PROGRAM frame05
67      IMPLICIT INTEGER (I-N), REAL(8) (A-H, O-Z)
68      !-----
69      1000 WRITE( 6, *) '*****'
70      WRITE( 6, *) ' <frame05> Ver 18August2008 for 2D Frame Structures'
71      WRITE( 6, *) ' All Rights Reserved for DataScience '
72      WRITE( 6, *) '*****'
73      CALL WAIT
74      !-----
75      CALL LINEAR
76      !-----
77      WRITE( 6, *) '*****'
78      WRITE( 6, *) ' quit: <frame05> Ver 18August2008 executed'
79      WRITE( 6, *) '*****'
80      CALL WAIT

```

```

81  !-----
82      END PROGRAM frame05
83  !*****
84      SUBROUTINE LINEAR
85      USE GLOBAL05
86      IMPLICIT INTEGER (I-N), REAL(8) (A-H,O-Z)
87  !=====
88      WRITE( *,*) ' +----- opened C:\frame05\frame05_inp.txt -----+'
89      CALL READ_FD_INP
90      WRITE( *,*) ' +----- closed C:\frame05\frame05_inp.txt -----+'
91  !-----
92      CALL OPEN_FD_OUT
93      WRITE(11,' (" +---(SYSTEM) -----
94  1-----+')')
95      WRITE(11,' (5X,"Finite Element used      ELE_LIB = ",A7)') ELE_LIB
96      WRITE(11,' (5X,"Nodal Coordinate      CODE = ",A2)') CODE
97      WRITE(11,' (5X,"Numb. of Elements      MENTS = ",I6)') MENTS
98      WRITE(11,' (5X,"Numb. of Nodes      NODES = ",I6)') NODES
99      WRITE(11,' (5X,"Numb. of Cross-Sections NSECTS = ",I6)') NSECTS
100 !-----隠しコマンド類の印刷 -----
101     IF (IPHDN/=0) THEN
102     WRITE(11,' (5X,"** Classified Control Parameters **")')
103     WRITE(11,' (5X,"Print this List      IPHDN = ",I6)') IPHDN
104     WRITE(11,' (5X,"Wait Reading Input    INWAIT = ",I6)') INWAIT
105     WRITE(11,' (5X,"Print SYSK           IPSYSK = ",I6)') IPSYSK
106     WRITE(11,' (5X,"Print ELEK           IPELEK = ",I6)') IPELEK
107     WRITE(11,' (5X,"Print PVEC           IPPVEC = ",I6)') IPPVEC
108     WRITE(11,' (5X,"Print TTLDOF         IPSOLV = ",I6)') IPSOLV
109     WRITE(11,' (5X,"Make Initial W3D File ICGINI = ",I6)') ICGINI
110     WRITE(11,' (5X,"Make Deformed W3D File ICGDEF = ",I6)') ICGDEF
111     WRITE(11,' (5X,"Make Ini/Def XLS File ICGXLS = ",I6)') ICGXLS !27/JAN/2005
112     WRITE(11,' (5X,"Numb. of CG Segments  NCGDEF = ",I6)') NCGDEF
113     WRITE(11,' (5X,"Scale for Deformation SCGDEF = ",D13.5)') SCGDEF
114     ENDIF
115 !-----
116     WRITE(11,' (" +---(NODE)-----
117  1-----+')')
118     DO N=1,NODES
119     WRITE(11,' ("      node#:",I6," (X,Y):",2E13.5)')
120     1      N, XYZNOD(N,1), XYZNOD(N,2)

```

```

121      ENDDO
122      WRITE(11,' (" +---(ELE)-----
123      1-----+"') )
124      DO M=1, MENTS
125      WRITE(11,' ("      element <"," I6,"> ","      node A & B:", 2I6,
126      1"      sectyp:", I2') )
127      1 M, NODELE (M, 1), NODELE (M, 2), MATSEC (M)
128      ENDDO
129      WRITE(11,' (" +---(SECTYP)-----
130      1-----+"') )
131      DO NSECT=1, NSECTS
132      WRITE(11,' ("      sectyp #: ", I6, " EA=", E12.5, " EI=: ", E12.5') )
133      1      NSECT,      SECTYP (NSECT, 1), SECTYP (NSECT, 2)
134      ENDDO
135      WRITE(11,' (" +---(DOF)-----
136      1-----+"') )
137      NDF=MDOFS/MNODES ! dof#/node
138      DO MENT=1, MENTS
139      WRITE(11,' ("      element <"," I6,">","      node A & B dof#: ", 6I6') )
140      1 MENT, (NEQELE (MENT, N), N=1, NDF), (NEQELE (MENT, NDF+N), N=1, NDF)
141      ENDDO
142      CALL CLOSE_FD_OUT
143      !-----
144      1000 CONTINUE ! CG (初期形状図/変形図)
145      IF(ICGINI==0) GOTO 2000 ! frame05_ini.data NOT produced
146      WRITE( 6,*) ' +--- opened: Initial Configuration CG file -----+'
147      CALL CG_FD_INI
148      WRITE(6,*) ' +--- closed: Initial Configuration CG file -----+'
149      !-----
150      2000 CONTINUE !  BAU_SYSK
151      CALL BAU_SYSK
152      WRITE( *,*) ' +----- System Stiffness Assembled-----+'
153      CALL OPEN_FD_OUT
154      WRITE(11,*) ' +-----+'
155      WRITE(11,*) ' +----- System Stiffness Assembled-----+'
156      CALL CLOSE_FD_OUT
157      !-----
158      IF(IPSYSK/=0) THEN
159      CALL PRINT_SYSK
160      CALL OPEN_FD_OUT

```

```

161      WRITE(11,*) ' +----- System Stiffness Matrix (symmetric) -----+'
162      DO K=1,NDOFS
163      DO J=1,K
164      IF (0.1E-12 <= ABS(SYSK(LSKY(J,K))) ) THEN
165      WRITE(11,' (5X," SYSK( ",I6," ",",I6," )= ",E12.5)')
166      1 J, K, SYSK(LSKY(J,K))
167      ENDIF
168      END DO
169      END DO
170      CALL CLOSE_FD_OUT
171      ENDIF ! IPSYSK/=0
172  ! ----- 分解 剛性方程式 -----
173      CALL LDLT_SYSK_RHS(1,0,NEGA,0) ! singularity checked
174  !-----
175      WRITE(*,*) ' +----- System Stiffness LDLT-Decomposed -----+'
176      CALL OPEN_FD_OUT
177      WRITE(11,*) ' +----- System Stiffness LDLT-Decomposed -----+'
178      CALL CLOSE_FD_OUT
179  !-----
180      IF (IPSYSK/=0) THEN
181      CALL PRINT_SYSK
182      CALL OPEN_FD_OUT
183      WRITE(11,*) ' +----- LDLT-Decomposed System Stiffness -----+'
184      DO K=1,NDOFS
185      DO J=1,K
186      IF (0.1E-15 <= ABS(SYSK(LSKY(J,K))) ) THEN
187      IF (J==K) THEN
188      WRITE(11,' (5X," D ( ",I6," ",",I6," )= ",E12.5)')
189      1 J, K, SYSK(LSKY(J,K))
190      ENDIF
191      IF (J/=K) THEN
192      WRITE(11,' (5X," LT ( ",I6," ",",I6," )= ",E12.5)')
193      1 J, K, SYSK(LSKY(J,K))
194      ENDIF
195      ENDIF
196      END DO
197      END DO
198      CALL CLOSE_FD_OUT
199      ENDIF ! IPSYSK/=0
200  !-----

```

```

201      RHS=PVEC ! 右辺項
202      !-----
203      IF (IPPVEC/=0) THEN
204      CALL OPEN_FD_OUT
205      WRITE(11,*) ' +----- Load Vector in Stiffness Equations -----+'
206      DO J=1,NDOFS
207      IF (0.1E-15 <= ABS(PVEC(J)) ) THEN
208      WRITE(11,' (5X," load vector (" ,I6,")= ",E12.5)') J, PVEC(J)
209      ENDIF
210      END DO
211      CALL CLOSE_FD_OUT
212      ENDIF ! IPPVEC/=0
213      ! ----- 解ベクトルの計算 -----
214      CALL LDLT_SYSK_RHS(0,1,NEGA,0)
215      TTLDOF=RHS
216      !-----
217      CALL OPEN_FD_OUT
218      WRITE(11,*) ' +----- Stiffness Equations Solved -----+'
219      WRITE(11,*) ' +-----+
220      CALL CLOSE_FD_OUT
221      !-----
222      IF (IPSOLV/=0) THEN
223      CALL OPEN_FD_OUT
224      WRITE(11,*) ' +----- Computed Nodal Degrees-of-Freedom -----+'
225      DO J=1,NDOFS
226      IF (0.1E-7 <= ABS(TTLDOF(J)) ) THEN
227      WRITE(11,' (5X," dof #",I6," : ",E12.5)') J, TTLDOF(J)
228      ENDIF
229      END DO
230      CALL CLOSE_FD_OUT
231      !----- 計算された節点変位ベクトルの表示
232      WRITE(*,*) ' +----- Computed Nodal Degrees-of-Freedom -----+'
233      2120 DO J=1,NDOFS
234      WRITE(6,2110) J, TTLDOF(J)
235      2110 FORMAT( 5X, ' dof# ', I6, ' = ', E12.5)
236      END DO
237      ENDIF ! IPSOLV/=0
238      !-----各要素の節点断面力
239      CALL POST_SYSK
240      !-----

```

```

241      IF (ICGDEF==0.AND.ICGXLS==0) GOTO 1910 ! frame05_def & frame05_xls NOT produced
242      WRITE( 6,*) '    +---- opened: Deformed Configuration CG file ----+'
243      CALL CG_FD_DEF
244      WRITE( 6,*) '    +---- closed: Deformed Configuration CG file ----+'
245      1910 CONTINUE
246      !-----
247      9000 CALL OPEN_FD_OUT
248      WRITE(11,' (" *****
249      1*****")')
250      WRITE(11,' (15X,"          <frame05> : Ver 18AUG08 executed ")')
251      WRITE(11,' (" *****
252      1*****")')
253      CALL CLOSE_FD_OUT
254      !-----
255      9999 CONTINUE
256      END SUBROUTINE LINEAR
257      !*****
258      SUBROUTINE BAU_SYSK
259      USE GLOBAL05
260      IMPLICIT INTEGER (I-N), REAL (8) (A-H, O-Z)
261      !=====
262      !      要素剛性行列から構造系剛性行列を組み立てるサブルーチン
263      !=====
264      SYSK=0.0          ! 剛性行列の初期化
265      !-----
266      DO M=1,MENTS      ! 要素
267      MTYP=MATSEC(M)    ! 断面タイプ
268      NA=NODELE(M,1);NB=NODELE(M,2) ! 節点番号
269      !===== 要素剛性行列の計算―― 要素の種類による分類 =====
270      SELECT CASE (ELE_LTB) !
271      !-----線形要素-----
272      CASE (' T2D_LIN')      ! 2 元線形トラス要素
273      EA=SECTYP(MTYP,1)      ! 伸び剛性
274      CALL T2D_LIN(EA, NA, NB)
275      !-----
276      CASE (' B2D_LIN')      ! 2 次元線形はり要素
277      EA=SECTYP(MTYP,1)      ! 伸び剛性
278      EI=SECTYP(MTYP,2)      ! 曲げ剛性
279      CALL B2D_LIN(EA,EI, NA, NB)
280      !-----

```

```

281      CASE DEFAULT
282      STOP ' error: ELE_LIB NOT identified/ BAU_SYSK'
283      !-----
284      END SELECT  ! ELEK computed
285      !-----
286      CALL OPEN_FD_OUT
287      WRITE(11,' (" +---Element Data for <" ,I6,">-----
288      1-----+")' ) M
289      !-----
290      XA=XYZNOD (NA, 1) ;YA=XYZNOD (NA, 2) !節点 A の座標
291      XB=XYZNOD (NB, 1) ;YB=XYZNOD (NB, 2) !節点 B の座標
292      ESPAN=SQRT ( (XB-XA)*(XB-XA)+(YB-YA)*(YB-YA) ) ! 要素長さ
293      CH=(XB-XA)/ESPAN;      SH=(YB-YA)/ESPAN
294      WRITE(11,' ("      Length L,   cosΦ &   sinΦ:" ,3E11.4)') ESPAN, CH, SH
295      WRITE(11,' ("      Axial Stiffness      EA :", E11.4)') EA
296      !-----
297      IF (ELE_LIB==' B2D_LIN' ) THEN
298      WRITE(11,' ("      Bending Stiffness      EI:" , E11.4)') EI
299      WRITE(11,' ("      Global Load (PA, QA, PB, QB):", 4E11.4)')
300      1      BALPQ (M, 1), BALPQ (M, 2), BALPQ (M, 3), BALPQ (M, 4)
301      WRITE(11,' ("      Local Load (pa, qa, pb, qb):", 4E11.4)')
302      1      CALPQ (M, 1), CALPQ (M, 2), CALPQ (M, 3), CALPQ (M, 4)
303      WRITE(11,' ("      Nodal Forces (PA, QA, MA):", 3E11.4)')
304      1      PQELE (M, 1), PQELE (M, 2), PQELE (M, 3)
305      WRITE(11,' ("      Nodal Forces (PB, QB, MB):", 3E11.4)')
306      1      PQELE (M, 4), PQELE (M, 5), PQELE (M, 6)
307      END IF
308      !-----
309      IF (IPELEK==0) GOTO 1015
310      WRITE(11,' (5X, "  ** element stiffness matrix **")')
311      DO J=1,MDOFS
312      WRITE(11,' (8X, 12E11.4)') (ELEK (J, K), K=1, MDOFS)
313      END DO
314      1015 CONTINUE
315      CALL CLOSE_FD_OUT
316      !-----その要素の節点変位番号-----
317      DO K=1,MDOFS  ! 列 要素剛性行列の対角要素を含む右上部分のみ
318      KSYS=NEQELE (M, K)
319      IF (KSYS==0) CYCLE
320      DO J=1,K ! 行 要素剛性行列の対角要素を含む右上部分のみ

```



```

321  !----- 系全体の剛性行列における行番号（行番地）と列番号（列番地） -----
322      JSYS=NEQELE (M, J)
323      IF (JSYS==0) CYCLE
324  !----- SYSK 重ね合わせ
325      LSYS=LSKY_SWAP (JSYS, KSYS) ! swapping, if necessary
326      SYSK (LSYS)=SYSK (LSYS)+ELEK (J, K)
327  !----- 節点自由度に拘束条件を課する場合の特別処置
328      IF (J/=K. AND. JSYS==KSYS) THEN ! ELEK の右上 offpvt が SYSK の pvt に来る場合
329      SYSK (LSYS)=SYSK (LSYS)+ELEK (J, K) ! ELEK の左下 offpvt も SYSK の pvt に 加算
330      ENDIF ! 13/Feb/98 世紀の大 debugging これで ICES98 shell buckling OK
331  !-----
332      END DO ! 行 J
333      END DO ! 列 K
334      END DO ! 要素 M
335  !-----
336      END SUBROUTINE BAU_SYSK
337  !*****
338      SUBROUTINE POST_SYSK
339  !=====
340  !      計算された節点変位をもとに、各要素についての節点断面力
341  !      を計算するサブルーチン
342  !=====
343      USE GLOBAL05
344      IMPLICIT INTEGER (I-N), REAL (8) (A-H, O-Z)
345  !-----
346      DO M=1, MENTS ! 要素
347      MTYP=MATSEC (M) ! 断面タイプ
348      NA=NODELE (M, 1); NB=NODELE (M, 2) ! 節点番号
349  !-----要素剛性行列の計算（当面は線要素のみ）-----
350  !===== 要素の種類による分類 =====
351      SELECT CASE (ELE_LTB) !
352  !-----線形要素-----
353      CASE (' T2D_LIN' ) ! 2元線形トラス要素
354      EA=SECTYP (MTYP, 1) ! 伸び剛性
355      CALL T2D_LIN (EA, NA, NB)
356  !-----
357      CASE (' B2D_LIN' ) ! 2次元線形はり要素
358      EA=SECTYP (MTYP, 1) ! 伸び剛性
359      EI=SECTYP (MTYP, 2)
360      CALL B2D_LIN (EA, EI, NA, NB)

```

```

361  !-----
362      CASE DEFAULT
363      STOP ' error ; ELE_LIB NOT identified /POST_SYSK '
364  !-----
365      END SELECT
366  !-----節点断面力の計算-----
367      WRITE(*, ' (" +----- Nodal Forces for < ", I6, " > -----
368      1-----+")' ) M
369  !-----
370      DO J=1, MDOFS !行 J
371      PQM=0.0
372      IF (ELE_LIB==' B2D_LIN' ) PQM=PQELE (M, J) ! 分布荷重による節点断面力 (拘束力)
373      DO K=1, MDOFS !列 K
374      IF (0<NEQELE (M, K) . AND. NEQELE (M, K)<=MDOFS) THEN
375      PQM=PQM+ELEK (J, K)*TTLD OF (NEQELE (M, K))
376      END IF
377      END DO !列 K
378      TTL=0.0
379      IF (0<NEQELE (M, J) . AND. NEQELE (M, J)<=MDOFS) THEN
380      TTL=TTLD OF (NEQELE (M, J))
381      END IF
382      WRITE(*, ' (" row#", I5, " dof#", I5, ":", E11.4,
383      1 " nodal force:", E11.4)' )
384      1 J, NEQELE (M, J), TTL, PQM
385      END DO ! 行 J
386  !-----
387      CALL OPEN_FD_OUT
388      WRITE(11, ' (" +----- Nodal Forces for < ", I6, " > -----
389      1-----+")' ) M
390      IF (IPELEK/=0) THEN
391      WRITE(11, ' (5X, "** element stiffness matrix **")' )
392      WRITE(11, ' (5X, " dof#", 12I11)' ) (NEQELE (M, K), K=1, MDOFS)
393      DO J=1, MDOFS
394      WRITE(11, ' (10X, 12E11.4)' ) (ELEK (J, K), K=1, MDOFS)
395      END DO
396      ENDIF ! ..... IF (IPELEK/=0)
397  !----- ZUG for T2D_LIN
398      XA=XYZNOD (NA, 1) ; YA=XYZNOD (NA, 2) ! 節点 A の座標
399      XB=XYZNOD (NB, 1) ; YB=XYZNOD (NB, 2) ! 節点 B の座標
400      ESPAN=SQRT ( (XB-XA)*(XB-XA)+(YB-YA)*(YB-YA) ) ! 要素長さ

```

```

401      CH=(XB-XA)/ESPAN;      SH=(YB-YA)/ESPAN
402      ZUG=0.0      ! for for T2D_LIN only
403      !-----
404      DO J=1,MDOFS      !行      J
405      PQM=0.0
406      IF (ELE_LIB==' B2D_LIN') PQM=PQELE (M, J)      ! 分布荷重による節点断面力 (拘束力)
407      DO K=1,MDOFS      !列      K
408      IF (0<NEQELE (M, K) . AND. NEQELE (M, K)<=NDOFS) THEN
409      PQM=PQM+ELEK (J, K)*TTLDOF (NEQELE (M, K))
410      END IF
411      END DO      !列 K
412      IF (J==1) ZUG=ZUG-PQM*CH      !- Pa*cos for T2D_LIN
413      IF (J==2) ZUG=ZUG-PQM*SH      !- Qa*sin for T2D_LIN
414      !-----
415      TTL=0.0
416      IF (0<NEQELE (M, J) . AND. NEQELE (M, J)<=NDOFS) THEN
417      TTL=TTLDOF (NEQELE (M, J))
418      END IF
419      WRITE (11, '(6X, "      row#:", I6, "      dof#:", I6, "      dof:", E11.4,
420      1      "      Nodal Force:", E11.4)')
421      1      J,      NEQELE (M, J), TTL,      PQM
422      END DO      !行 J
423      !-----
424      IF (ELE_LIB==' T2D_LIN') THEN
425      WRITE (11, '( "      Axial Force in Truss Element: (+) for Tension : "
426      1      , E11.4)') ZUG
427      ENDIF
428      CALL CLOSE_FD_OUT
429      !-----
430      END DO      ! 要素 M
431      !-----
432      END SUBROUTINE POST_SYSK
433      !*****
434      SUBROUTINE T2D_LIN (EA, NA, NB)
435      USE GLOBAL05
436      IMPLICIT INTEGER (I-N), REAL (8) (A-H, O-Z)
437      !-----
438      !      平面トラス要素の要素剛性行列 ELEK (4, 4) を
439      !      計算するサブルーチン      (ZA, ZB) は不要データ
440      !      コンパイラー通過 1996 年 10 月 14 日一応完成! 藤井

```

```

441  !-----
442      XA=XYZNOD (NA, 1) ; YA=XYZNOD (NA, 2) ! 節点 A の座標
443      XB=XYZNOD (NB, 1) ; YB=XYZNOD (NB, 2) ! 節点 B の座標
444  !-----要素長さ L -----
445      ESPAN=SQRT ( (XB-XA)*(XB-XA)+(YB-YA)*(YB-YA) )
446      IF (ESPAN==0.0) THEN
447          STOP ' error: ESPAN=0.0 /T2D_LIN'
448      END IF
449      CH=(XB-XA)/ESPAN;      SH=(YB-YA)/ESPAN
450      CC=CH*CH*EA/ESPAN
451      CS=CH*SH*EA/ESPAN
452      SS=SH*SH*EA/ESPAN
453  !----- 要素剛性行列 ELEK (4, 4) -----
454      ELEK (1, 1)=CC
455      ELEK (3, 3)=CC
456      ELEK (3, 1)=-CC
457      ELEK (1, 3)=-CC ! check ok
458  !-----
459      ELEK (2, 2)=SS
460      ELEK (4, 4)=SS
461      ELEK (4, 2)=-SS
462      ELEK (2, 4)=-SS ! check ok
463  !-----
464      ELEK (1, 2)=CS
465      ELEK (2, 1)=CS
466      ELEK (3, 4)=CS
467      ELEK (4, 3)=CS
468      ELEK (1, 4)=-CS
469      ELEK (2, 3)=-CS
470      ELEK (3, 2)=-CS
471      ELEK (4, 1)=-CS ! check ok
472  !-----
473      END SUBROUTINE T2D_LIN
474  !*****
475      SUBROUTINE B2D_LIN (EA, EI, NA, NB)
476      USE GLOBAL05
477      IMPLICIT INTEGER (I-N), REAL (8) (A-H, O-Z)
478  !-----
479  ! 平面はり要素の要素剛性行列 ELEK (6, 6) を
480  ! 計算するサブルーチン (ZA, ZB) は不要データ

```

481 ! コンパイラー通過 1996 年 10 月 14 日完成 ! 赤玉 智生

482 !-----

483 XA=XYZNOD (NA, 1) ;YA=XYZNOD (NA, 2) !節点 A の座標

484 XB=XYZNOD (NB, 1) ;YB=XYZNOD (NB, 2) !節点 B の座標

485 !-----要素長さ L

486 ESPAN=SQRT ( (XB-XA)\*(XB-XA)+(YB-YA)\*(YB-YA) )

487 IF (ESPAN==0.0) THEN

488 STOP ' error: ESPAN=0.0 /B2D\_LIN'

489 END IF

490 CH=(XB-XA)/ESPAN;SH=(YB-YA)/ESPAN

491 CC=CH\*CH; CS=CH\*SH; SS=SH\*SH

492 EACC=CC\*EA/ESPAN

493 EACS=CS\*EA/ESPAN

494 EASS=SS\*EA/ESPAN

495 !-----

496 EI02=2\*EI/ESPAN

497 EI04=4\*EI/ESPAN

498 EI06C=6\*CH\*EI/(ESPAN\*ESPAN)

499 EI06S=6\*SH\*EI/(ESPAN\*ESPAN)

500 EI12CC=12\*CC\*EI/(ESPAN\*ESPAN\*ESPAN)

501 EI12CS=12\*CS\*EI/(ESPAN\*ESPAN\*ESPAN)

502 EI12SS=12\*SS\*EI/(ESPAN\*ESPAN\*ESPAN)

503 ELEK=0.0

504 !----- 要素剛性行列 ELEK (6, 6)

505 ELEK (3, 3) = EI04

506 ELEK (6, 6) = EI04

507 ELEK (3, 6) = EI02

508 ELEK (6, 3) = EI02

509 ELEK (2, 3) = EI06C

510 ELEK (3, 2) = EI06C

511 ELEK (2, 6) = EI06C

512 ELEK (6, 2) = EI06C

513 ELEK (3, 5) = -EI06C

514 ELEK (5, 3) = -EI06C

515 ELEK (5, 6) = -EI06C

516 ELEK (6, 5) = -EI06C

517 ELEK (3, 4) = EI06S

518 ELEK (4, 3) = EI06S

519 ELEK (4, 6) = EI06S

520 ELEK (6, 4) = EI06S

```

521      ELEK(1,3)=-EI06S
522      ELEK(3,1)=-EI06S
523      ELEK(1,6)=-EI06S
524      ELEK(6,1)=-EI06S
525      ELEK(2,2)= EASS+EI12CG
526      ELEK(5,5)= EASS+EI12CG
527      ELEK(2,5)=-EASS-EI12CG
528      ELEK(5,2)=-EASS-EI12CG
529      ELEK(1,1)= EACC+EI12SS
530      ELEK(4,4)= EACC+EI12SS
531      ELEK(1,4)=-EACC-EI12SS
532      ELEK(4,1)=-EACC-EI12SS
533      ELEK(2,4)=-EACS+EI12CS
534      ELEK(4,2)=-EACS+EI12CS
535      ELEK(1,5)=-EACS+EI12CS
536      ELEK(5,1)=-EACS+EI12CS
537      ELEK(1,2)= EACS-EI12CS
538      ELEK(2,1)= EACS-EI12CS
539      ELEK(4,5)= EACS-EI12CS
540      ELEK(5,4)= EACS-EI12CS
541      END SUBROUTINE B2D_LIN
542      !*****
543      SUBROUTINE LDLT_SYSK_RHS(IDEK, ISLV, NEGA, IPRINT)
544      !=====
545      !          連立方程式の LDLT 分解による解法ルーチン
546      !      SYSK  (1次元配列)   : 対称な係数行列 (分解結果)
547      !      RHS   (1次元配列)   : 右辺項ベクトル (解ベクトル)
548      !      NDOFS (>=2)         : 元数 (未知量の個数)
549      !          分解結果は SYSK に格納されてサブルーチンを抜ける
550      !          解ベクトルは RHS に格納されてサブルーチンを抜ける
551      !          (2*2)元連立以上の連立方程式で応用可能!
552      !      IDEK    =0/1=分解しない/分解する
553      !      ISLV    =0/1=解ベクトルを求めない/求める
554      !      NEGA    =分解結果の中での負の対角要素の個数
555      !      DETLOG  = log (係数行列の行列式)
556      !      IPRINT  =0/1=分解結果と解ベクトルを画面表示しない/する
557      !
558      !=====
559      USE GLOBAL05
560      IMPLICIT INTEGER (I-N), REAL (8) (A-H, O-Z)

```

```

561  !----- IDEC=0 なら分解しない (直ちに解へ ｸﾄﾙを求める)
562      IF (IDEC==0) GOTO 2000
563      ZERO=0.1E-15
564  !=====
565      DO 340 J=1,NDOFS      !   すべての行について
566  !-----第 1 行目の対角要素-----
567      IF (J==1) THEN      !   第 1 行目について
568          PVT=SYSK( LSKY(J, J) )
569  !-----㊦対角要素の検出
570      IF (ABS(PVT)<=ZERO) THEN
571          WRITE(*,*) J, '...error! zero pivot in LDLT'
572          CALL OPEN_FD_OUT
573          WRITE(11,*) ' 剛性行列が特異行列です!入力データをチェックせよ.'
574          CALL CLOSE_FD_OUT
575          STOP
576      END IF
577  !-----第 1 行目の非対角要素
578      DO 100 K=2,NDOFS
579          SYSK( LSKY(J, K) )=SYSK( LSKY(J, K) )/PVT
580      100 CONTINUE
581  !-----以上 第 1 行目について
582      ELSE      !   その他の行の場合
583  !-----
584      DO 330 K=J,NDOFS      !   第 2 行目以降について
585  !-----対角要素
586      IF (J==K) THEN
587          PVT=SYSK( LSKY(J, K) )
588          DO 200 M=1, J-1
589              PVT=PVT-SYSK( LSKY(M, K) )*SYSK( LSKY(M, M) )*SYSK( LSKY(M, K) )
590          200 CONTINUE
591          SYSK( LSKY(J, K) )=PVT
592  !-----㊦対角要素の検出-----
593      IF (PVT==0.0) THEN
594          WRITE(6,*) J, 'error! zero pivot in LDLT'
595          CALL OPEN_FD_OUT
596          WRITE(11,*) ' 剛性行列が特異行列です!入力データをチェックせよ.'
597          CALL CLOSE_FD_OUT
598          STOP
599      END IF
600  !-----

```

```

601      ELSE
602      !-----非対角要素
603          SVAL=SYSK( LSKY(J, K) )
604          DO 300 M=1, J-1
605              SVAL=SVAL-SYSK( LSKY(M, K) ) * SYSK( LSKY(M, M) ) * SYSK( LSKY(M, J) )
606          300 CONTINUE
607      !-----
608          SYSK( LSKY(J, K) )=SVAL/PVT
609          END IF
610      !-----
611      330 CONTINUE ! 第2行目以降について
612      !-----
613          END IF
614      !-----
615      340 CONTINUE ! すべての行について
616      !=====
617      !          <LDLT 分解終了>
618      !=====
619      !----- マイナス 対角要素の個数のカウント
620          NEGA=0          ! マイナス対角要素の個数の初期値設定
621          DO 400 M=1, NDOFS
622              PVT=SYSK(LSKY(M, M))
623              IF (PVT<0.0) NEGA=NEGA+1
624          400 CONTINUE
625      !-----
626          IF (NEGA/=0) THEN
627              CALL OPEN_FD_OUT
628              WRITE(11,*) ' warning: negative diagonals in [D]!'
629              CALL CLOSE_FD_OUT
630              STOP          ' warning: negative pivot(s) in LDLT-decomposition'
631          END IF
632      !-----ISLV=0 なら (分解のみ) 終わり
633      2000 IF (ISLV==0) GOTO 5000
634      !----- 暫定解 {X} を求める
635          DO 700 L=2, NDOFS
636              SVAL=RHS(L)
637              DO 600 N=1, L-1
638                  SVAL=SVAL-SYSK(LSKY(N, L)) * RHS(N)
639          600 CONTINUE
640          RHS(L)=SVAL

```



```

641      700 CONTINUE
642      !-----暫定解 {Y} を求める (対角要素で割るだけ)
643          DO 800 L=1, NDOFS
644              RHS(L)=RHS(L)/SYSK( LSKY(L, L) )
645          800 CONTINUE
646      !-----最終的な解ベクトル RHS(NDOFS) を求める--
647          DO 1000 L=NDOFS-1, 1, -1
648              SVAL=RHS(L)
649              DO 900 M=L+1, NDOFS
650                  SVAL=SVAL-SYSK( LSKY(L, M) ) *RHS(M)
651          900 CONTINUE
652              RHS(L)=SVAL
653      1000 CONTINUE
654      !=====結果の画面表示 (IPRINT=0/1) =====
655      5000 CONTINUE
656          IF(IPRINT==0) GOTO 9000
657      !-----分解結果の画面表示-----
658          WRITE(*,*) '----- LDLT-Decomposed Stiffness -----'
659          DO K=1, NDOFS
660              DO J=1, K
661                  WRITE(*, ' (" LDLT (" , (I6), ", ", (I6), ") =", E12.5) ' )
662                      1 J, K, SYSK( LSKY(J, K) )
663              END DO
664          END DO
665          WRITE(*,*) '----- Diagonal Matrix [D]----- '
666          DO L=1, NDOFS
667              WRITE(*, ' (" D (" , I6, ") =", E12.5) ' ) L, SYSK( LSKY(L, L) )
668          END DO
669      !-----行列式と負の対角成分の個数の画面表示
670          WRITE(*, ' (" Numb. of Negative Diagonals in [D]= ", I6) ' ) NEGA
671      !-----
672          CALL WAIT
673      !-----解の画面表示
674          WRITE(*,*) '----- Solution Vector -----'
675          DO 7000 J=1, NDOFS
676              WRITE(*, ' (" RHS (" , I6, ") =", E12.5) ' ) J, RHS(J)
677          7000 CONTINUE
678      !-----
679      9000 CONTINUE
680          END SUBROUTINE LDLT_SYSK_RHS

```

```

681  !*****
682      SUBROUTINE WAIT
683  !---- 画面の連続表示を一時停止して キー 入力を待つ ---
684      WRITE(*, '(A)', ADVANCE='NO')
685      1 ' frame05 is waiting for your enter key strike....'; READ(*,*)
686      END SUBROUTINE WAIT
687  !*****
688      FUNCTION LSKY_SWAP(JSYS, KSYS)
689      USE GLOBAL05
690      IMPLICIT INTEGER (I-N), REAL (8) (A-H, O-Z)
691  !-----
692  !      skyline 対称剛性行列 SYSK のなかで
693  !      (JSYS, KSYS)要素の番地 LSKY_SWAP を与える関数プログラム
694  !-----
695  !      (JSYS, KSYS)要素は“左下に位置するかも知れない”. この場合は
696  !      対称性から, 右上に自動的に転換処理(swapping)した番地を与える
697  !      (JSYS, KSYS)は保存される
698  !----- 使用例 : -----
699  !      SYSK(LSKY_SWAP(J, K)) = -4343.88*EI
700  !      SYSK(LSKY_SWAP(J+1, K+1)) = SYSK(LSKY_SWAP(J, K))
701  !----- simple check the compatibility (jsys, ksys) -----
702      IF (JSYS<=0. OR. NDOFS<JSYS) THEN
703      WRITE(*, *) JSYS, ' error:row#<=0 or NDOFS<row#/LSKY_SWAP'
704      STOP
705      ENDIF
706      IF (KSYS<=0. OR. NDOFS<KSYS) THEN
707      WRITE(*, *) KSYS, ' error:clm#<=0 or NDOFS<clm#/LSKY_SWAP'
708      STOP
709      ENDIF
710  !-----
711      J=JSYS; K=KSYS
712  !----- swapping, if necessary -----
713      IF (JSYS>KSYS) THEN ! 左下にある場合
714      J=KSYS; K=JSYS ! 行番号と列番号を SWAPPING
715      END IF
716  !-----
717      IF (J>K) THEN
718      STOP ' error: j>k, not right_upper/LSKY_SWAP'
719      ENDIF
720  !----SYSK(LSKYMAX)のなかでの番地 LSKY_SWAP ----

```

```

721 ! 一次元配列 SYSK なかの要素は対角要素から上に向けて番号付けしてある.
722 ! 注目しているのは第 K 目の列.
723 ! この列の対角成分の番地は LSKYPVT(K) で、そして
724 ! この対角成分の上に K-JRWSKY(K) 個の非ゼロ成分がある.
725 ! このうち第 J 行にある成分の番地は
726 ! LSKYPVT(K) + K - J
727 ! である.
728 !-----
729 LSKY_SWAP = LSKYPVT(K) + K - J ! SYSK(LSKYMAX) のなかの番地
730 !-----
731 IF (LSKYMAX < LSKY_SWAP) THEN
732 WRITE(*,*) LSKY_SWAP,
733 1 ' error: larger than LSKYMAX/LSKY_SWAP'
734 STOP
735 ENDIF
736 !-----
737 END FUNCTION LSKY_SWAP
738 !*****
739 FUNCTION LSKY(JSYS, KSYS)
740 USE GLOBAL05
741 IMPLICIT INTEGER (I-N), REAL (8) (A-H, O-Z)
742 !-----
743 ! skyline 対称剛性行列 SYSK のなかで
744 ! 必ず「右上に位置する」 (JSYS, KSYS)要素の
745 ! 番地 LSKY を与える関数副プログラム
746 ! 注意) swapping しない!
747 ! 使い方の例: SYSK(LSKY(J, K)) = -4343.88*EI
748 ! 使い方の例: SYSK(LSKY(J+1, K+1)) = SYSK(LSKY(J, K)) など
749 !-----
750 IF (JSYS > KSYS) THEN
751 STOP ' error: j>k, not right_upper/LSKY'
752 ENDIF
753 !---SYSK(LSKYMAX)のなかでの番地 LSKY ----
754 ! 一次元配列 SYSK なかの要素は対角要素から上に向けて番号付けしてある.
755 ! 注目しているのは第 KSYS 目の列.
756 ! この列の対角成分の番地は LSKYPVT(KSYS) で、そして
757 ! この対角成分の上に KSYS-JRWSKY(KSYS) 個の非ゼロ成分がある.
758 ! このうち第 JSYS 行にある成分の番地は
759 ! LSKYPVT(KSYS) + KSYS - JSYS
760 ! である.

```

```

761  !-----
762      LSKY=LSKYPVT(KSYS)+KSYS-JSYS ! SYSK(LSKYMAX) のなかの番地
763  !-----
764      IF (LSKYMAX<LSKY) THEN
765          WRITE(*,*) LSKY, ' error: larger than LSKYMAX/LSKY'
766          STOP
767          ENDIF
768  !-----
769      END FUNCTION LSKY ! 18/Aug/97 coded by Fujii,F.
770  !*****
771      SUBROUTINE PRINT_SYSK
772      USE GLOBAL05
773      IMPLICIT INTEGER (I-N), REAL(8) (A-H,O-Z)
774  !-----SYSK の画面表示-----
775          DO K=1,NDOFS
776              DO J=1,K
777                  IF (0.1E-12 <= ABS(SYSK(LSKY(J,K))) ) THEN
778                      WRITE(*, ' (" SYSK(", I5, ", ", I5, ")=", E12.5) ' )
779                      1 J, K, SYSK(LSKY(J,K))
780                  ENDIF
781              END DO
782          END DO
783      END SUBROUTINE PRINT_SYSK
784  !*****
785      SUBROUTINE READ_FD_INP          ! 一応完成
786  !      inputdata file FILE_NAME='C:\¥frame05¥frame05_inp.txt'
787      CALL      OPEN_FD_INP
788      CALL      READ_DATA
789      CALL      CLOSE_FD_INP
790      END SUBROUTINE READ_FD_INP ! 一応完成
791  !*****
792      SUBROUTINE OPEN_FD_INP
793  !-----
794          OPEN(UNIT=97, IOSTAT=IOS, ERR=9999,
795              1FILE='C:\¥frame05¥frame05_inp.txt', ! 固定した ファイル 名前
796              1STATUS=' OLD',
797              1ACCESS=' SEQUENTIAL',
798              1FORM=' FORMATTED' )
799          GOTO 9000
800      9999 WRITE(*,*) IOS, ' error: OPEN_FILE error!'

```

```

801      STOP
802      9000 WRITE(6,*) ' .....unit is now connected.'
803      WRITE(*,*) ' opened : C:\¥frame05¥frame05_inp.txt'
804      END SUBROUTINE OPEN_FD_INP
805      !*****
806      SUBROUTINE CLOSE_FD_INP
807      CLOSE (UNIT=97, IOSTAT=IOS, ERR=9999, STATUS=' KEEP' )
808      GOTO 9000
809      9999 WRITE(6,*) ' error: CLOSE_FILE error!'
810      STOP
811      9000 WRITE(6,*) ' closed: C:\¥frame05¥frame05_inp.txt'
812      END SUBROUTINE      CLOSE_FD_INP
813      !*****
814      SUBROUTINE READ_DATA  !一応完成
815      USE GLOBAL05
816      CHARACTER TEXT80*80
817      !-----
818      !   READ ALL LINES IN THE INPUT DATA FILE
819      !   UNTIL (ALLEND) WILL BE READ.
820      !-----
821      WRITE(*,*) ' reading: C:\¥frame05¥frame05_inp.txt'
822      1000 CONTINUE
823      CALL READ_LINE(TEXT80)
824      !-----最初の一行目の有無をチェック-----
825      IF (INDEX(TEXT80, ' (MEMO)' )==0) THEN
826      WRITE(*,*) ' error: (MEMO) missed.'
827      STOP
828      ENDIF
829      2000 CONTINUE
830      !----- read data blocks in sequence
831      WRITE(*,*) ' reading.... (MEMO)'
832      3110 CALL DATA_MEMO
833      IF (INWAIT/=0) CALL WAIT
834      !-----system control data
835      WRITE(*,*) ' reading... (SYSTEM)'
836      3120 CALL DATA_SYSTEM
837      IF (INWAIT/=0) CALL WAIT
838      !-----structural data
839      WRITE(*,*) ' reading.... (NODE)'
840      3150 CALL DATA_NODE

```

```

841      IF (INWAIT/=0) CALL WAIT
842      WRITE(*,*) ' reading.... (ELE)'
843      3160 CALL DATA_ELE
844      IF (INWAIT/=0) CALL WAIT
845      WRITE(*,*) ' reading... (SECTYP)'
846      3170 CALL DATA_SECTYP
847      IF (INWAIT/=0) CALL WAIT
848      WRITE(*,*) ' reading..... (DOF)'
849      3180 CALL DATA_DOF
850      IF (INWAIT/=0) CALL WAIT
851      !----- loading
852      WRITE(*,*) ' reading.... (LOAD)'
853      3190 CALL DATA_LOAD
854      IF (INWAIT/=0) CALL WAIT
855      !----- check ' (QUIT)'
856      CALL READ_LINE (TEXT80)
857      IF (INDEX (TEXT80, ' (QUIT)') == 0) THEN
858      STOP ' error: (QUIT) missed.'
859      ENDIF
860      WRITE(*,*) ' (QUIT) detected and READ_FILE quit'
861      END SUBROUTINE READ_DATA  !一応完成
862      !*****
863      SUBROUTINE READ_LINE (TEXT80)          !一応完成
864      CHARACTER TEXT80*80
865      !-----
866      1000 CONTINUE
867      READ (97, 1100) TEXT80
868      1100 FORMAT (A80)
869      !-----左端に！ があるコメント文は無視（飛ぶ）-----
870      IF (TEXT80(1:1) == '!') GOTO 1000 ! 次の行を読み込む
871      CALL EXCLAM (TEXT80) ! “!” より右の文字を全てブランクにする
872      !-----
873      END SUBROUTINE READ_LINE  !一応完成
874      !*****
875      SUBROUTINE EXCLAM (TEXT80)  !一応完成
876      !-----“!” より右の文字を全てブランクにする
877      CHARACTER TEXT80*80
878      !---- “!” が 80 文字の文字列 TEXT80 に含まれているか? ----
879      KEX = INDEX (TEXT80, '!')
880      !-----含まれていない なら直ちに終了

```

```

881      IF (KEX==0) GOTO 9000
882  !-----KEXCLAM 列目以降をすべて blank out
883      TEXT80 (KEX:80)= ' '
884  !-----
885      9000 CONTINUE
886      END SUBROUTINE EXCLAM !一応完成
887  !*****
888      SUBROUTINE NUMB (N09, ZAHL)      !一応完成
889  !-----
890  !      convert the string  ZAHL  into integer  N09
891  !-----
892      IMPLICIT INTEGER (I-N), REAL (8) (A-H, O-Z)
893      CHARACTER ZAHL*1
894  !----暫定的な値 (-230241 ) を代入しておく
895      N09= -230241
896      IF (ZAHL==' 0' )  N09=0
897      IF (ZAHL==' 1' )  N09=1
898      IF (ZAHL==' 2' )  N09=2
899      IF (ZAHL==' 3' )  N09=3
900      IF (ZAHL==' 4' )  N09=4
901      IF (ZAHL==' 5' )  N09=5
902      IF (ZAHL==' 6' )  N09=6
903      IF (ZAHL==' 7' )  N09=7
904      IF (ZAHL==' 8' )  N09=8
905      IF (ZAHL==' 9' )  N09=9
906  !----- 文字 -> 整数 の変換が実行されなかった場合
907      IF (N09== -230241) THEN
908          WRITE(6,*) ZAHL, ' error: not integer/NUMB'
909          STOP
910      END IF
911      END SUBROUTINE NUMB      !一応完成
912  !*****
913      SUBROUTINE GET_NAME_EQ (TEXT80, KEQ, STRING)      !一応完成
914      IMPLICIT INTEGER (I-N), REAL (8) (A-H, O-Z)
915      CHARACTER TEXT80*80, STRING*80
916  !-----
917  !      TEXT80 のなかで 等号のある第 KEQ 列 より 左にあり
918  !      最初に出現する max 6 文字から成る文字列 (変数名) を採取し
919  !      STRING = ' SEVEN '      :   に左詰めで格納
920  !-----

```

```

921      NWRD= 0
922      STRING(1:80)=' ' !文字列 STRING の初期化
923      !----- TEXT80 の中の等号の左にある文字列（変数名）の採取
924      DO 3000 K=1, KEQ-1
925          STRING(K:K)=TEXT80(K:K) ! 空白部に挟まれている
926      3000 CONTINUE
927      !--- STRING= ' SEVEN ' : を左に詰める
928          STRING=ADJUSTL (STRING) ! STRING= ' SEVEN '
929      !----空白部を除いた文字長
930          NWRD=LEN_TRIM (STRING)
931      !-----変数名が何も検出されないとき-----
932          IF (NWRD==0) THEN
933              STOP ' error: no string detected/GET_NAME_EQ'
934          END IF
935      !-----
936          END SUBROUTINE GET_NAME_EQ !一応完成
937      !-----
938      SUBROUTINE GET_EQ_VAL (TEXT80, K1, K2, STRING) !一応完成
939      IMPLICIT INTEGER (I-N), REAL (8) (A-H, O-Z)
940      CHARACTER TEXT80*80, STRING*80
941      !-----
942      ! TEXT80 のなかで 第 K1 列から右にあり最初に出現する文字列を採取
943      ! その採取した文字列の最後の文字の位置を K2 に格納
944      ! STRING=' -9.007655E-32 ' : に左詰めで格納
945      ! K2: 重要な出力値
946      !-----
947          NWRD= 0 ;K2=K1
948          STRING(1:80)=' ' ! initialize STRING
949      !-----第 K1 列から右にあり最初に出現する文字列を採取--
950          DO 3000 K=K1, 80
951      !-----これまで何等かの文字を検出してきたが、空白が出た場合
952          IF (NWRD/=0. AND. TEXT80(K:K)==' ') EXIT !採取終了
953      !-----
954          IF (TEXT80(K:K)/=' ') THEN ! 何等かの文字を検出
955              NWRD=NWRD+1
956              STRING (NWRD:NWRD)=TEXT80 (K:K) ! 左詰め
957              K2=K
958          END IF
959      3000 CONTINUE
960      !-----STRING のなかの最初の空白部は、文字列の最後尾のつぎに出現するはず

```



```

961      IF (NWRD/=INDEX (STRING, ' ')-1) THEN
962      STOP ' error: NWRD incorrect /GET_EQ_VAL'
963      END IF
964      !-----数値の文字列が採取されず-----
965      IF (NWRD==0) THEN
966      STOP ' error:STRING not detected /GET_EQ_VAL'
967      END IF
968      !-----
969      END SUBROUTINE GET_EQ_VAL    !一応完成
970      !*****
971      SUBROUTINE GET_NAME_TYP (NAME, INTREL)      !一応完成
972      !   check the variable name type (real? or integer?)
973      !-----
974      IMPLICIT INTEGER (I-N), REAL (8) (A-H, O-Z)
975      CHARACTER  NAME*6
976      INTREL=0  !最初は実数を仮定
977      !----- (I-N)は整数である-----
978      IF (' I' <=NAME (1:1).AND. NAME (1:1)<=' N') INTREL=1    !整数である
979      9999 CONTINUE
980      END SUBROUTINE GET_NAME_TYP    !一応完成
981      !*****
982      SUBROUTINE GET_NODE_NDF6 (JEQS, NODE, NDF6)      !一応完成
983      USE GLOBAL05
984      IMPLICIT INTEGER (I-N), REAL (8) (A-H, O-Z)
985      !-----
986      !   find the node number (NODE) and the dof number (NDF6) for
987      !   the Eqs  number (JEQS)
988      !-----
989      NODE=0 ; NDF6=0
990      !-----
991      DO ND=1, NODES
992      DO N6=1, 6
993      NVAL=NEQNOD (ND, N6)
994      IF (NVAL==JEQS) THEN
995      NODE=ND      ; NDF6=N6
996      GOTO 3000
997      END IF
998      END DO
999      END DO
1000      WRITE (6,*) JEQS, ' error: not identified /NEQNOD/NEQS_NODE_DF6'

```

```

1001      STOP
1002      3000 CONTINUE
1003      END SUBROUTINE GET_NODE_NDF6    !一応完成
1004      !-*****
1005      SUBROUTINE GET_INT (STRING, NVAL) ! needs recheck 一応完成
1006      IMPLICIT INTEGER (I-N), REAL (8) (A-H, O-Z)
1007      CHARACTER STRING*80, ZAHL*1
1008      !-----
1009      !      convert STRING*80 into integer
1010      !      (整数値を表す文字列を整数値データに変換する)
1011      !-----
1012      !      impossible !      : -2333.442E-2
1013      !-----空白部を除いた文字長
1014      NWRD=LEN_TRIM (STRING)
1015      !-----小数点(は有り得ない)の有無をチェック-----
1016      DO K=1, NWRD
1017      IF (STRING (K:K)=='.') THEN
1018      WRITE (6,*) ' error:  "."      unacceptable/GET_INT'
1019      STOP
1020      END IF
1021      END DO
1022      !----- check "E" and "D"
1023      NEXP=0      ! 指数の大きさ
1024      KED=0      ! 指数記号のある位置
1025      !----('E', 'e', 'D', 'd') のうちどれか一つがあるかも (2つはない)--
1026      KED=INDEX (STRING, 'E')+INDEX (STRING, 'e')+
1027      1      INDEX (STRING, 'D')+INDEX (STRING, 'd')
1028      !----指数記号のない場合は、最初に出現する空白の位置を KED に代入
1029      IF (KED==0) THEN
1030      KED=INDEX (STRING, ' ')      ! STRING には文字列が左詰め
1031      GOTO 4000
1032      END IF
1033      !----- 指数表現のある場合 (指数記号の位置は KED)
1034      1500 CONTINUE
1035      !----- if E-03 : error 指数が負であることは許されない
1036      DO 2000 K=KED+1, NWRD
1037      IF (STRING (K:K)=='-') THEN
1038      WRITE (6,*) ' error: (E-03) unacceptable/GET_INT '
1039      STOP
1040      END IF

```

```

1041      2000 CONTINUE
1042      !-----
1043      !      "E+03" is ACCEPTABLE IN GET_INT
1044      !--- check +03 rightward from E 指数部のチェック-----
1045          DO 3000 KK=1, NWRD-KED          !      指数部の位番号(右から)
1046      !-----
1047          K=NWRD+1-KK          !文字の列番号(左から)
1048          IF (STRING(K:K)=='+') GOTO 3000 !指数部の採取終了
1049      !----(符号は出現したとしても + のみ)
1050          ZAHL=STRING(K:K)          ! 第 k 列の文字を採取
1051          CALL NUMB(N09, ZAHL)          ! 0-9 の整数に変換
1052          NEXP=NEXP+ N09*(10**(KK-1))          !      指数の値 (整数値で - 々)
1053      3000 CONTINUE
1054      !-----指数記号のない場合 (指数記号より左の部分) ----
1055      4000 CONTINUE
1056      !----- check : -2333 without "." 小数点は有り得ない-----
1057          NSCA=0          !      指数記号より左の部分の整数値 (数値で - 々)
1058          NVRZ=+1          !      符号
1059      !-----leftward from E (指数記号・空白桁の位置は KED)
1060          DO 5000 KK=1, KED-1          ! 位の番号(右から)
1061              K=KED-KK          ! 文字の列番号 (左から)
1062      !----- 符号 の出現 check
1063          IF (STRING(K:K)=='+') GOTO 5000          !正符号を採取したら終了
1064          IF (STRING(K:K)=='-') THEN !負符号を採取したら NVRZ= - 1 ! 符号を逆転させて
1065              GOTO 5000          !      終了
1066          END IF
1067      !-----右から (1 の位から)
1068          ZAHL=STRING(K:K)          ! 文字の列番号 (左から)
1069          CALL NUMB(N09, ZAHL)          ! 0-9 の整数に変換
1070          NSCA=NSCA+ N09*(10**(KK-1))          ! 位の番号(右から)
1071      5000 CONTINUE
1072      !-----符号を考慮
1073          NSCA=NVRZ*NSCA ! 指数記号より左の部分の整数値 (数値で - 々)
1074      !-----指数表現の文字列が意味する最終的な整数値 (数値で - 々)
1075          NVAL=NSCA*(10**NEXP)
1076          GOTO 9500
1077      !-----
1078      9500 CONTINUE
1079          END SUBROUTINE GET_INT          ! needs recheck 一応完成
1080      !-*****

```

```

1081      SUBROUTINE GET_REL (STRING, SVAL) ! needs recheck 一応完成
1082      IMPLICIT INTEGER (I-N), REAL (8) (A-H, O-Z)
1083      CHARACTER STRING*80, ZAHL*1
1084      !-----
1085      !      実数値を意味する文字列を実数データに変換する:  -2333.442E-02
1086      !-----空白部を除いた文字長
1087      NWRD=LEN_TRIM (STRING)
1088      !-----('E', 'e', 'D', 'd') のうちどれか一つがあるかも (2つはない)
1089      KED=INDEX (STRING, 'E') + INDEX (STRING, 'e') + INDEX (STRING, 'D')
1090      1 + INDEX (STRING, 'd') ! 指数記号のある位置
1091      !-----
1092      NEXP=0 ! 指数の大きさ
1093      !-----指数表現でない場合: (-2333.442)の類をチェック
1094      IF (KED==0) GOTO 4000
1095      !-----
1096      2000 CONTINUE
1097      !-----符号の初期設定(正とする)
1098      NVRZ=+1
1099      !-----指数部について-----
1100      DO 3000 KK=1, NWRD-KED ! 指数部の位番号(右から)
1101      !-----
1102      K=NWRD+1-KK ! 文字の桁番号(左から)
1103      ZAHL=STRING (K:K) ! 第 k 列の文字を採取
1104      IF (ZAHL=='+') GOTO 3100 ! 正符号を採取したら終了
1105      IF (ZAHL=='-') THEN ! 負符号を採取したら
1106          NVRZ=-1 ! 符号を逆転させて
1107          GOTO 3100 ! 終了
1108      END IF
1109      CALL NUMB (NO9, ZAHL) ! 0-9 の整数に変換
1110      NEXP=NEXP+NO9*(10**(KK-1)) ! 指数の値 (整数値データ)
1111      3000 CONTINUE
1112      3100 CONTINUE
1113      !-----指数の値 (負の整数値データも有り得る) E-03
1114      NEXP=NEXP*NVRZ
1115      !-----
1116      4000 CONTINUE
1117      !-----指数記号 (または空白部) から左の部分について: -232.675
1118      IF (KED==0) THEN ! 指数表現でない場合は
1119      KED=INDEX (STRING, ' ') ! 最初の空白の列番号を KED に格納
1120      END IF

```

```

1121  !-----
1122      NVRZ=+1          !符号の再初期化
1123      SUM=0.0          !実数値の初期化
1124  !-----少数点 "." の位置  KPNT
1125      KPNT=INDEX (STRING, '.')
1126  !----- 少数点なしの場合: "-907654E-02", "-6766" など
1127      IF (KPNT.EQ.0) THEN      !少数点なしの場合
1128          KPNT=KED              !指数記号（または空白部）の列番号を KPNT に格納
1129          GOTO 5000            !1の位以上の値(整数)の扱いへ
1130      END IF
1131  !-- "-32214." も 少数点なしの場合と同じく, 1の位以上の値(整数)の扱いへ
1132      IF (KPNT==NWRD) GOTO 5000
1133  !-- "-32214.E-09" も 少数点なしの場合と同じく, 1の位以上の値(整数)の扱いへ
1134      IF (KPNT+1==KED) GOTO 5000
1135  !----- "-45.0"  "-.3210"  "-56." のように-----
1136  !      指数表現なし, 少数点あり,
1137  !      空白部の位置が最後の文字の次に出現する場合: KED=NWRD+1=INDEX (STRING, ' ')
1138  !      小数点より右の部分(小数点以下), 列番号で KPNT と KED の間をチェック
1139  !-----
1140      IF (KPNT==KED) GOTO 5000  ! 小数点以下の文字列全くなし
1141      DO 6000 K=1, KED-KPNT-1    ! 小数点から 右に向けて
1142          KK=KPNT+K              ! 文字の列番号
1143          ZAHL=STRING (KK:KK)    ! 文字の採取
1144          CALL NUMB (NO9, ZAHL)  ! 0-9 の整数に変換
1145          SUM=SUM+DBLE (NO9)*(DBLE (10)**DBLE (-K)) ! 小数点以下の実数値
1146      6000 CONTINUE
1147  !-----これで "*****.9554523" の扱い終了 -----
1148  !      sum = 小数点以下の実数値 (数値データ)
1149  !-----小数点が第1列目に出現する場合: ".4523" -----
1150      IF (KPNT==1) GOTO 8000 ! 小数点より左部分の処理不要
1151  !----- "-.9554523"  "-325532.9554523"
1152      5000 CONTINUE
1153  !-----小数点 "." より左部分の処理: "-325532.*****"
1154      DO 7000 K=1, KPNT-1    ! 位数(小数点より左に向けて)
1155          KK=KPNT-K          ! 文字の列番号
1156          ZAHL=STRING (KK:KK) ! 文字の列番号
1157          IF (ZAHL=='+') GOTO 7000 !正符号を採取したら終了
1158          IF (ZAHL=='-') THEN      !負符号を採取したら
1159              NVRZ=-1              ! 符号を逆転させて
1160          GOTO 7000              ! 終了

```

```

1161      END IF
1162      CALL NUMB(N09, ZAHL)          ! 0-9 の整数に変換
1163      SUM=SUM+DBLE(N09*(10**(+K-1))) ! 小数点より左の実数値 (数値データ)
1164      7000 CONTINUE
1165      !-----
1166      8000 CONTINUE
1167      !-----指数表現の文字列が意味する最終的な実数値 (数値データ)
1168      SVAL = DBLE(NVRZ)*SUM*DBLE(10)**DBLE(NEXP)
1169      !-----
1170      CONTINUE
1171      END SUBROUTINE GET_REL ! needs recheck 一応完成
1172      !-----
1173      SUBROUTINE DATA_MEMO ! 入力データファイルの書き
1174      CHARACTER TEXT80*80
1175      !-----
1176      CALL OPEN_FD_OUT
1177      WRITE(11,' (" *****
1178      1*****"')
1179      WRITE(11,' (14X,"<frame05> Ver 18AUG08 for 2D Frame Structures"')
1180      WRITE(11,' (" *****
1181      1*****"')
1182      WRITE(11,' (" +---(MEMO)-----
1183      1-----+"')
1184      !-----最初の第一行目 (MEMO) はすでに読み込まれている---
1185      WRITE(*,*) ' (MEMO) being processed'
1186      1000 CALL READ_LINE(TEXT80) ! 読み込むだけ & nothing done
1187      !-----
1188      IF (INDEX(TEXT80,'END ')==0) THEN ! "END" でない場合
1189      WRITE(*, ' (2X,A75)') TEXT80
1190      WRITE(11,' (2X,A75)') TEXT80
1191      GOTO 1000 ! 'END ' で抜ける
1192      ENDIF
1193      !-----
1194      CALL CLOSE_FD_OUT
1195      !-----
1196      END SUBROUTINE DATA_MEMO ! 入力データファイルの書き
1197      !-----
1198      SUBROUTINE DATA_SYSTEM ! 計算ジョブ全体の制御用パラメータ
1199      USE GLOBAL05
1200      IMPLICIT INTEGER (I-N), REAL (8) (A-H, O-Z)

```

```

1201      CHARACTER TEXT80*80, STRING*80, NAME*6
1202      !-----
1203      CALL READ_LINE(TEXT80)
1204      IF (INDEX(TEXT80, ' (SYSTEM) ') == 0) THEN
1205          STOP ' error: (SYSTEM) missed!'
1206      ENDIF
1207      WRITE(6, *) ' (SYSTEM) being processed '
1208      !----- inital setting -----
1209      ELE_LIB = ' B2D_LIN' ! default
1210      CODE=' XY' ! default 何もないければ XYZ-座標系 (暗黙の了解) ' PL' =Polar
1211      INPNEQ=0
1212      ISKYLN=1
1213      INWAIT=0
1214      IPELEK=0 ! 15/Oct/98
1215      IPSYSK=0 ! 11/Oct/03
1216      IPPVEC=0 ! 11/Oct/03
1217      IPSOLV=0 ! 11/Oct/03
1218      ICGINI=0 ! for Win3D 20/Nov/98
1219      ICGDEF=0 ! for Win3D 26/Dez/98
1220      ICGXLS=0 ! 27/JAN/05
1221      NCGDEF=10 ! =10/100/1000 coded: 30/Jan/05
1222      SCGDEF=10.0 ! 26/Dez/98
1223      !-----
1224      1000 CALL READ_LINE(TEXT80)
1225      IF (INDEX(TEXT80, ' END' ) /= 0) GOTO 9000 ! END 行なら
1226      IF (INDEX(TEXT80, '=' ) == 0) THEN
1227          STOP ' error: "=" missed/(SYSTEM) '
1228      ENDIF
1229      !----- 等号記号の位置 ---
1230      KEQ=INDEX(TEXT80, '=') ! TEXT80 での "=" の位置を採取
1231      STRING(1:80)=' ' ! 初期化
1232      !===== [要素コード名 "ELE_LIB=" ]の採取
1233      IF (INDEX(TEXT80, ' ELE_LIB' ) /= 0) THEN ! ' ELE_LIB=' の場合
1234          DO K=KEQ+1, 80 ! get 等号の右にある要素コード
1235              STRING(K:K)=TEXT80(K:K) ! 一般には空白に挟まれている
1236          ENDDO
1237          STRING=ADJUSTL(STRING) ! 左に詰める
1238          NWRD=LEN_TRIM(STRING) ! 空白部を除いた文字長
1239      !----- 使用要素コード は必ず 7 文字で指定 -----
1240      IF (NWRD /= 7) THEN

```

```

1241      STOP ' error: ELE_LIB not 7 characters /(SYSTEM)'
1242      END IF
1243      DO K=1,7 ! 必ず 7 文字
1244      ELE_LIB(K:K)=STRING(K:K)
1245      END DO
1246      WRITE(*,*) ' ELE_LIB := ', ELE_LIB
1247      GOTO 1000 ! go to the next data line
1248      ENDIF ! ELE_LIB
1249      !----- [座標コード名 "CODE="]の採取
1250      IF (INDEX(TEXT80,'CODE')/=0) THEN ! 座標系の指定がある場合
1251      CODE='**' ! temporary code
1252      IF (INDEX(TEXT80,'XY')/=0) CODE='XY' ! XY 座標系
1253      IF (INDEX(TEXT80,'PL')/=0) CODE='PL' ! POLAR 座標系
1254      IF (CODE=='**') THEN
1255      STOP ' error: "CODE=" incorrect/(SYSTEM)'
1256      ENDIF
1257      WRITE(*,*) ' CODE := ', CODE
1258      GOTO 1000 ! read the next line
1259      ENDIF
1260      !-----
1261      CALL GET_NAME_EQ(TEXT80,KEQ, STRING) ! get 等号左側の変数名
1262      ! ! STRING 左づめ
1263      !-----STRING*80 のなかから先頭の 6 文字を採取して NAME*6 へ格納
1264      DO K=1,6 ! 最大 6 文字までの変数名
1265      NAME(K:K)=STRING(K:K)
1266      ENDDO
1267      !-----実数型の変数名か 整数型の変数名か
1268      CALL GET_NAME_TYP(NAME, INTREL)
1269      !-----等号の右側の文字列 (数値) を採取-----
1270      KSTART=KEQ+1
1271      CALL GET_EQ_VAL(TEXT80,KSTART, KEND, STRING) ! STRING 左づめ
1272      !-----変数名が実数型・整数型によりより分岐
1273      IF (INTREL==0) GOTO 3000
1274      IF (INTREL==1) GOTO 4000
1275      WRITE(6,*) ' ',NAME,' error:INTREL incorrect/(SYSTEM)'
1276      STOP
1277      !-----変数名が実数型の場合---
1278      3000 CONTINUE
1279      !-----文字列を実数型数値データに変換-----
1280      CALL GET_REL (STRING,SVAL)

```



```

1281  !----- DATA_SYSTEM のなかの既存の実数型登録変数
1282      IF (NAME==' SCGDEF' )    SCGDEF=SVAL
1283      WRITE (6,*) ' ',NAME,' := ', SVAL
1284      GOTO 1000
1285  !-----変数名が整数型の場合--文字列を整数型数値データに変換
1286      4000 CALL GET_INT (STRING, NVAL)
1287  !-----DATA_SYSTEM のなかの既存の整数型登録変数
1288      IF (NAME==' MENTS ' ) MENTS=NVAL
1289      IF (NAME==' NODES ' ) NODES=NVAL
1290      IF (NAME==' NSECTS' ) NSECTS=NVAL
1291      IF (NAME==' INPNEQ' ) INPNEQ=NVAL
1292      IF (NAME==' INWAIT' ) INWAIT=NVAL
1293      IF (NAME==' IPELEK' ) IPELEK=NVAL
1294      IF (NAME==' IPSYSK' ) IPSYSK=NVAL
1295      IF (NAME==' IPPVEC' ) IPPVEC=NVAL
1296      IF (NAME==' IPSOLV' ) IPSOLV=NVAL
1297      IF (NAME==' IPHIDN' ) IPHIDN=NVAL
1298      IF (NAME==' ICGINI' ) ICGINI=NVAL
1299      IF (NAME==' ICGDEF' ) ICGDEF=NVAL
1300      IF (NAME==' ICGXLS' ) ICGXLS=NVAL ! 16/DEZ/2003
1301      IF (NAME==' NCGDEF' ) NCGDEF=NVAL
1302  !-----
1303      WRITE (6,*) ' ',NAME,' := ',NVAL
1304  !-----
1305      5000 GOTO 1000      ! END 行を読み込まない限り続ける
1306  !----- システムパラメータの読み込み終了
1307      9000 CONTINUE      ! END 行を読み込みました
1308  !-----
1309      SELECT CASE (ELE_LIB)
1310      CASE ("B2D_LIN")
1311          MNODES=2 ;   MDOFS =6
1312      CASE ("T2D_LIN")
1313          MNODES=2 ;   MDOFS =4
1314      CASE DEFAULT
1315          STOP ' error: ELE_LIB not identified in (SYSTEM)'
1316      END SELECT
1317  !-----動的配列 (必要な記憶領域を確保)
1318      ALLOCATE (NODELE (MENTS, MNODES)) ;NODELE=0 ! 各要素を形成する節点番号
1319      ALLOCATE (NEQNOD (NODES, 3) ) ;NEQNOD=0 ! 各節点の節点変位番号
1320      ALLOCATE (NEQELE (MENTS, MDOFS) ) ;NEQELE=0 ! 各要素の節点変位番号

```

```

1321      ALLOCATE (SECTYP (NSECTS, 2)      );SECTYP=0.0      ! 各断面タイプ°の属性
1322      ALLOCATE (MATSEC (MENTS)          );MATSEC=0      ! 各要素の断面タイプ°番号
1323      ALLOCATE (UVWNOD (NODES, 2)      );UVWNOD=0.0      ! 各節点の並進変位 (CG 用)
1324      ALLOCATE (XYZNOD (NODES, 2)      );XYZNOD=0.0 ! 各節点の座標値
1325      ALLOCATE (ELEK (MDOFS, MDOFS)    );ELEK=0.0      ! 要素剛性行列
1326      IF (ELE_LIB==' B2D_LIN' ) THEN
1327      ALLOCATE (PQELE (MENTS, 6)      );PQELE=0.0 ! 要素の等価節点外力
1328      ALLOCATE (BALPQ (MENTS, 4)      ); BALPQ=0.0 ! global PQ 分布荷重強度
1329      ALLOCATE (CALPQ (MENTS, 4)      ); CALPQ=0.0 ! local PQ 分布荷重強度
1330      END IF
1331      !-----
1332      END SUBROUTINE DATA_SYSTEM !計算シ°ョフ°全体の制御用パラメータ
1333      !-----*****
1334      SUBROUTINE DATA_NODE              !節点座標
1335      USE GLOBAL05
1336      IMPLICIT INTEGER (I-N), REAL (8) (A-H, O-Z)
1337      CHARACTER TEXT80*80, STRING*80
1338      !-----
1339      CALL READ_LINE (TEXT80)
1340      IF (INDEX (TEXT80, ' (NODE)' )==0) THEN
1341      STOP ' error: (NODE) missed!'
1342      ENDIF
1343      WRITE (6, *) ' (NODE) being processed'
1344      ND=0 ! 読み込んだ 節点数のカンター
1345      !-----
1346      1000 CONTINUE
1347      CALL READ_LINE (TEXT80)          ! (NODE)の次の行を読み込む
1348      !----- write(*,*) TEXT80
1349      IF (INDEX (TEXT80, 'END' )/=0) GOTO 2000 ! END 行 が出現した時
1350      !-----
1351      IF (INDEX (TEXT80, ' NODE' )==0. OR. INDEX (TEXT80, '=' )==0 ) THEN
1352      STOP ' error:"NODE=" missed/(NODE)'
1353      ENDIF
1354      !----- 節点座標 (X, Y) の読み込み
1355      ND=ND+1      ! 読み込んだ 節点数のカンター
1356      KEQ=INDEX (TEXT80, '=' ) ! 等号記号の位置を採取
1357      !-----
1358      !      NODE = 節点番号 XY : 節 点 座 標
1359      !      NODE = 322      XY : -9.08 78.0
1360      !-----節点番号を採取-----

```

```

1361      KSTART=KEQ+1
1362      CALL GET_EQ_VAL(TEXT80,KSTART, KEND , STRING)
1363      CALL GET_INT(STRING, NVAL) ! 節点番号を整数型数値データに変換
1364      IF (NVAL/=ND) THEN
1365          STOP ' error: "NODE=" node# ?/(NODE)'
1366      ENDIF
1367      !-----節点座標を採取-----
1368      KEQ=INDEX(TEXT80,':') ! XY ":" 記号の位置を採取
1369      !----- X 座標を採取
1370      KSTART=KEQ+1
1371      CALL GET_EQ_VAL(TEXT80,KSTART, KEND , STRING)
1372      CALL GET_REL(STRING,X) ! X 座標を実数型数値データに変換
1373      XYZNOD(NVAL,1)=X
1374      !----- Y 座標を採取-----
1375      KSTART=KEND+1
1376      CALL GET_EQ_VAL(TEXT80,KSTART, KEND , STRING)
1377      CALL GET_REL(STRING,Y) ! Y 座標を実数型数値データに変換
1378      XYZNOD(NVAL,2)=Y
1379      !=====
1380      GOTO 1000 ! すべての節点について繰り返す
1381      2000 CONTINUE
1382      !-----
1383      IF (ND/=NODES) THEN
1384          STOP ' error: "NODE=" not for all nodes/(NODE)'
1385      END IF
1386      !-----必要に応じた座標変換と格納されたデータの表示確認-----
1387      XY: DO N=1, NODES
1388      !-----
1389      IF (CODE=='PL') THEN ! 極 座標系
1390          TH=XYZNOD(N,1) ; R=XYZNOD(N,2)
1391          X= R * COS(TH*(3.141592654/180.))
1392          Y= R * SIN(TH*(3.141592654/180.))
1393          XYZNOD(N,1)=X ; XYZNOD(N,2)=Y
1394      END IF !IF (CODE=='PL')
1395      !----- XYZ 座標系ならそのまま
1396      X=XYZNOD(N,1) ; Y=XYZNOD(N,2)
1397      WRITE(*,3300) N, X, Y
1398      3300 FORMAT(2X,' node#:',I5,' (X,Y) :',2E13.5)
1399      IF (INWAIT/=0.AND.MOD(N,20)==0) CALL WAIT
1400      !-----

```

```

1401      ENDDO XY
1402  !-----
1403      END SUBROUTINE DATA_NODE      ! 節点座標
1404  !-*****
1405      SUBROUTINE DATA_ELE      ! 要素属性
1406      USE GLOBAL05
1407      IMPLICIT INTEGER (I-N), REAL (8) (A-H, O-Z)
1408  !-----
1409      CHARACTER TEXT80*80, STRING*80
1410      CALL READ_LINE (TEXT80)
1411      IF (INDEX (TEXT80, ' (ELE)' )==0) THEN
1412      STOP ' error: (ELE) missed      !'
1413      ENDIF
1414      WRITE (*, *) ' (ELE) being processed'
1415      M=0 ! 要素数のカウンタ-
1416      4000 CONTINUE
1417  !-----END 行を読み込んだら -----
1418      CALL READ_LINE (TEXT80)
1419      IF (INDEX (TEXT80, ' END' )/=0) GOTO 9000
1420      IF (INDEX (TEXT80, ' ELE' )==0. OR. INDEX (TEXT80, ' =' )==0 ) THEN
1421      STOP ' error: "ELE=" missed /(ELE)'
1422      ENDIF
1423  !-----
1424      M=M+1 ! 要素数のカウンタ-
1425      KEQ=INDEX (TEXT80, ' =' ) ! 等号の位置の確認
1426  !-----
1427  !      ELE = 要素番号  節点番号      断面タイプ番号
1428  !      ELE =   12      NODE:  8   9  SECTYP : 1
1429  !----- 等号のすぐ右にある要素番号の採取-
1430      KSTART=KEQ+1
1431      CALL GET_EQ_VAL (TEXT80, KSTART, KEND , STRING)
1432      CALL GET_INT (STRING, M)
1433  !-----そのまた右にある節点番号の採取-----
1434      KEND=INDEX (TEXT80, ' :' ) ! 一番左にある ":" の位置の確認
1435      DO 1300 ND=1, MNODES ! 各要素について節点は MNODES 個ある
1436      KSTART=KEND+1
1437      CALL GET_EQ_VAL (TEXT80, KSTART, KEND , STRING)
1438      CALL GET_INT (STRING, NVAL)
1439      NODELE (M, ND)=NVAL
1440      1300 CONTINUE

```

```

1441      TEXT80(1:KEND)=' ' ! 読込んだ部分を空白置換
1442 !-----断面タイプ番号 TYP: の採取-
1443      KEQ=INDEX(TEXT80,':') ! 2 番目の ":" の位置の確認
1444      KSTART=KEQ+1
1445      CALL GET_EQ_VAL(TEXT80,KSTART, KEND , STRING)
1446      CALL GET_INT (STRING, NVAL)
1447      MATSEC(M)=NVAL
1448 !-----
1449      GOTO 4000 !すべての要素について読み込む
1450 9000 CONTINUE
1451 !-----カウンタされた要素数のチェック
1452      IF (M/=MENTS) THEN
1453          STOP ' error:"ELE=" not for all elements/(ELE)'
1454      END IF
1455 !----- 画面出力
1456      DO M=1, MENTS
1457          WRITE(*,' (" element < ",I5,> " ", "      node A & B:",I5,
1458 1"      sectyp:",I3)')
1459          1 M, NODELE (M, 1), NODELE (M, 2), MATSEC (M)
1460 !-----
1461          IF (INWAIT/=0.AND.MOD (M, 5)==0) CALL WAIT
1462          END DO ! for all elements
1463 !-----
1464      END SUBROUTINE DATA_ELE      ! 要素属性
1465 !*****
1466      SUBROUTINE DATA_SECTYP      ! 断面タイプのデータ
1467      USE GLOBAL5
1468      IMPLICIT INTEGER (I-N), REAL (8) (A-H, O-Z)
1469      CHARACTER TEXT80*80, STRING*80
1470      CALL READ_LINE (TEXT80)
1471      IF (INDEX (TEXT80, ' (SECTYP)') ==0) THEN
1472          STOP ' error:(SECTYP) missed !'
1473      END IF
1474      WRITE(6,*) ' (SECTYP) being processed '
1475 !-----各断面タイプのデータ-----
1476 !      断面タイプ番号      EA      EI
1477 !      TYP=1      EAEI: 100.0 10.0
1478 !-----
1479 1000 CONTINUE
1480      NSECT=0 !断面タイプ番号のカウンタの初期化

```

```

1481      2000 CONTINUE
1482      CALL READ_LINE(TEXT80)
1483      IF (INDEX(TEXT80, 'END')/=0) GOTO 9000
1484      IF (INDEX(TEXT80, 'TYP')=0) THEN
1485          STOP ' error:"TYP=" missed/(SECTYP) !'
1486      END IF
1487      !-----
1488      NSECT=NSECT+1
1489      KEQ=INDEX(TEXT80, '=') !等号記号の位置
1490      !-----断面タイプ 番号のカンターの採取-----
1491      KSTART=KEQ+1
1492      CALL GET_EQ_VAL(TEXT80, KSTART, KEND, STRING)
1493      CALL GET_INT(STRING, NVAL) ! 断面タイプ 番号
1494      KEQ=INDEX(TEXT80, ':') ! 記号 ":" の位置
1495      !----- EA の採取-----
1496      KSTART=KEQ+1
1497      CALL GET_EQ_VAL(TEXT80, KSTART, KEND, STRING)
1498      CALL GET_REL(STRING, EA)
1499      SECTYP(NVAL, 1)=EA
1500      !-----EI の採取-----
1501      KSTART=KEND+1
1502      CALL GET_EQ_VAL(TEXT80, KSTART, KEND, STRING)
1503      CALL GET_REL(STRING, EI)
1504      SECTYP(NVAL, 2)=EI
1505      !-----
1506      GOTO 2000      !      すべての 断面タイプ 番号 について繰り返す
1507      9000 CONTINUE
1508      IF (NSECT/=NSECTS) THEN
1509          STOP ' error: more "TYP=" needed/(SECTYP)'
1510      END IF
1511      !-----画面表示-----
1512      DO NSECT=1, NSECTS
1513          WRITE(*, '( " sectyp #:", I3, "      EA=", E12.5, "      EI=: ", E12.5) ' )
1514          1      NSECT,      SECTYP(NSECT, 1), SECTYP(NSECT, 2)
1515          IF (INWAIT/=0) CALL WAIT
1516      END DO
1517      END SUBROUTINE DATA_SECTYP ! 断面タイプ のデータ
1518      !*****
1519      SUBROUTINE DATA_DOF      ! 節点変位の拘束条件
1520      USE GLOBAL05

```

```

1521      IMPLICIT INTEGER (I-N), REAL (8) (A-H, O-Z)
1522      CHARACTER TEXT80*80, STRING*80
1523      !----- ' (DOF)' の読み込み-----
1524      CALL READ_LINE (TEXT80)
1525      IF (INDEX (TEXT80, ' (DOF)' )==0) THEN
1526      STOP ' error: (DOF) missed    !'
1527      END IF
1528      WRITE (*, *) ' (DOF) being processed '
1529      !-----
1530      IF (INPNEQ==1) GOTO 100
1531      !-----
1532      ! INPNEQ=0 のとき, 各節点の節点自由度に課せられた拘束条件に関する情報
1533      !      NEQNOD (NODES, 3) を読み込む
1534      !      NODE=45   DOF: 101      ! 0=constrained 1=NOT constrained
1535      !-----
1536      !      節点を均一に変位させるため措置
1537      !      (A, B, C) と 3 種類の均一値にしか拘束できない
1538      !-----
1539      !      NODE=15   DOF: 10A      ! A: この記号の節点自由度はお互いに同じ値をとる
1540      !      NODE=16   DOF: BCA      ! B: この記号の節点自由度はお互いに同じ値をとる
1541      !      NODE=17   DOF: BC1      ! C: この記号の節点自由度はお互いに同じ値をとる
1542      !-----
1543      NDOFS=0      ; ND=0      ! NDOFS のカウンタのための初期化-
1544      NEQNOD=0
1545      NEQAAA=0;NEQBBB=0;NEQCCC=0 ! 均一変位の場合 (3 グループのみ)
1546      2000 CONTINUE
1547      !-----END を読み来んだら終了-----
1548      CALL READ_LINE (TEXT80)
1549      IF (INDEX (TEXT80, ' END' )/=0) GOTO 9000 ! 読込終了
1550      !-----
1551      IF (INDEX (TEXT80, ' NODE' )==0. OR. INDEX (TEXT80, '=' )==0 ) THEN
1552      STOP ' error:"NODE=" missed/(DOF) !'
1553      END IF
1554      KEQ=INDEX (TEXT80, '=' ) ! 等号の位置
1555      ND=ND+1
1556      !-----
1557      !      NODE=45 DOF: 001 ! 0=constrained 1=NOT constrained
1558      !-----等号のすぐ右にある節点番号 NODE の採取
1559      KSTART=KEQ+1
1560      CALL GET_EQ_VAL (TEXT80, KSTART, KEND , STRING)

```

```

1561      CALL GET_INT (STRING, NODE)
1562      !-----さらに右にある節点自由度の拘束情報コード (3 文字) の採取-----
1563      KEQ=INDEX(TEXT80,':') ! ":" の位置
1564      KSTART=KEQ+1
1565      CALL GET_EQ_VAL (TEXT80,KSTART, KEND , STRING)
1566      IF (LEN_TRIM (STRING)/=3) THEN
1567      STOP ' error: 非 3 文字コード/(DOF)'
1568      END IF
1569      !-----採取した 3 文字拘束情報コードの解析-----
1570      DO K=1, 3 ! 101, AB0, 11A, ....
1571      !----- 単一 非ゼロ 自由度
1572      IF (STRING(K:K)=='1') THEN
1573      NDOFS=NDOFS+1
1574      NEQNOD (NODE, K)=NDOFS
1575      END IF
1576      !----- ゼロ 拘束 自由度
1577      IF (STRING(K:K)=='0') THEN
1578      NEQNOD (NODE, K)=0
1579      END IF
1580      !----- グループ A 自由度
1581      IF (STRING(K:K)=='A') THEN
1582      IF (NEQAAA==0) THEN
1583      NDOFS=NDOFS+1
1584      NEQAAA=NDOFS
1585      NEQNOD (NODE, K)=NEQAAA
1586      END IF
1587      IF (NEQAAA/=0) THEN
1588      NEQNOD (NODE, K)=NEQAAA
1589      END IF
1590      END IF
1591      !----- グループ B 自由度
1592      IF (STRING(K:K)=='B') THEN
1593      IF (NEQBBB==0) THEN
1594      NDOFS=NDOFS+1
1595      NEQBBB=NDOFS
1596      NEQNOD (NODE, K)=NEQBBB
1597      END IF
1598      IF (NEQBBB/=0) THEN
1599      NEQNOD (NODE, K)=NEQBBB
1600      END IF

```



```

1601      END IF
1602  !----- グループ C 自由度
1603      IF (STRING(K:K)=='C') THEN
1604          IF (NEQCCC==0) THEN
1605              NDOFS=NDOFS+1
1606              NEQCCC=NDOFS
1607              NEQNOD (NODE, K)=NEQCCC
1608          END IF
1609          IF (NEQCCC/=0) THEN
1610              NEQNOD (NODE, K)=NEQCCC
1611          END IF
1612      END IF
1613  !-----
1614      END DO
1615      GOTO 2000      ! すべての節点について繰り返す
1616  !-----
1617  ! INPNEQ=1 のとき (2次元構造系で節点変位番号を直接入力する場合)
1618  !
1619  !      各要素の節点変位番号を直接入力
1620  !      NEQELE (MENTS, MDOFS) を読み込む
1621  !      ELE=45      NEQ: 0 0 23 0 11 3      ! 入力データとして準備した節点変位番号
1622  !      NDOFS = 読み取られたすべての節点番号のなかの最大値
1623  !===== debugged===== 12APR205 =====
1624      100 CONTINUE
1625      NDOFS=0      ;      MENT=0
1626      200 CALL READ_LINE (TEXT80)
1627      IF (INDEX (TEXT80, 'END')/=0) GOTO 9000
1628      IF (INDEX (TEXT80, 'ELE')==0. OR. INDEX (TEXT80, '=')==0 ) THEN !'ELE = ' の check
1629          STOP ' error: "ELE=" missed/(DOF)!'
1630      END IF
1631      KEQ=INDEX (TEXT80, '=') !等号記号の位置
1632      MENT=MENT+1
1633  !-----
1634  !      ELE=45      NEQ: 0 23      11 3      ! 2D TRUSS
1635  !-----等号記号のすぐ右にある要素番号 ME の採取
1636      KSTART=KEQ+1
1637      CALL GET_EQ_VAL (TEXT80, KSTART, KEND , STRING)
1638      CALL GET_INT (STRING, ME)
1639  !-----
1640      KEND=INDEX (TEXT80, ':') ! "NEQ : "の位置 12APR2005
1641      DO MDF=1, MDOFS      ! 要素自由度数について

```

```

1641      KSTART=KEND+1 ! 12APR2005
1642      CALL GET_EQ_VAL(TEXT80,KSTART, KEND , STRING) ! 節点変位番号の採取
1643      CALL GET_INT (STRING, NVAL)
1644      NEQELE (ME, MDF)=NVAL
1645      IF (NDOFS<NVAL) NDOFS=NVAL ! NDOFS:最大の節点変位番号
1646      END DO
1647      !-----
1648      GOTO 200 ! すべての要素について繰り返す
1649      !=====
1650      9000 CONTINUE
1651      IF (INPNEQ==0. AND. ND/=NODES) THEN !すべての節点についてデータあるか
1652      STOP ' error: "NODE=" not for all nodes /(DOF) '
1653      END IF
1654      !-----
1655      IF (INPNEQ==1. AND. MENT/=MENTS) THEN !すべての要素についてデータあるか
1656      STOP ' error:"ELE=" not for all elements/(DOF) '
1657      END IF
1658      !-----動的配列（必要な記憶領域を確保）-----
1659      !-----荷重ベクトル関係-----
1660      ALLOCATE ( PVEC (NDOFS) );PVEC=0.0 ! 荷重変数 p の荷重ベクトル e
1661      ALLOCATE ( RHS (NDOFS) );RHS=0.0 ! 連立方程式の右辺項（一般）
1662      ALLOCATE ( TTLDOF (NDOFS) );TTLDOF=0.0
1663      ALLOCATE ( LSKYPVT (NDOFS) );LSKYPVT=0
1664      !-----
1665      LSKYPVT (1)=1
1666      DO L=2, ndofs
1667      LSKYPVT (L)=LSKYPVT (L-1)+(L-1)
1668      ENDDO
1669      !----- 画面出力 NEQNOD (nodes, 3)
1670      IF (INPNEQ==0) THEN
1671      3400 DO ND=1, NODES
1672      WRITE (6, 3434) ND, (NEQNOD (ND, N6), N6=1, 3)
1673      3434 FORMAT (2X, ' node #:', I5, ' eqs # for 3 dofs:', 3I6)
1674      IF (INWAIT/=0. AND. MOD (ND, 20)==0) CALL WAIT
1675      END DO
1676      CALL PUT_NEQELE !要素ごとの節点変位番号の生成
1677      END IF ! end if INPNEQ=0
1678      !----- 出力 NEQELE (ments, mdofs)
1679      3600 CONTINUE
1680      DO M=1, MENTS

```

```

1681      WRITE(*, ' (" element <:", I5, " >", "   dof# :", I5)')
1682      1 M, (NEQELE (M, N), N=1, MDOFS)
1683      IF (INWAIT/=0. AND. MOD (MENT, 4)==0) CALL WAIT
1684      END DO      ! MENT=1, MENTS
1685      !----- 大次元の 1 次元配列 -----
1686      LSKYMAX=(NDOFS*NDOFS-NDOFS)/2+NDOFS
1687      ALLOCATE ( SYSK (LSKYMAX) );SYSK=0.0      ! 剛性行列
1688      !----- 画面出力 -----
1689      WRITE (6, *)      ' -----'
1690      WRITE (6, ' ("   系自由度の数 (NDOFS) =", I3)') NDOFS
1691      WRITE (6, *)      ' -----'
1692      END SUBROUTINE DATA_DOF      ! 節点の支持条件
1693      !-*****
1694      SUBROUTINE PUT_NEQELE !
1695      USE GLOBAL05
1696      IMPLICIT INTEGER (I-N), REAL (8) (A-H, O-Z)
1697      !-----
1698      ! only for INPNEQ=0 の時 (all pinned/ all rigid node を仮定) ,
1699      ! NODELE (MENTS, MNODES) :各要素を定義する節点の節点番号
1700      ! NEQNOD (NODES, 3)      :各節点の節点自由度の節点自由度番号 (方程式番号)
1701      !                          から
1702      ! NEQELE (MENTS, MDOFS) :各要素の節点自由度番号 (方程式番号)
1703      !                          を生成する.
1704      !-----
1705      NDFS=MDOFS/MNODES ! 1 節点あたりの自由度 = 要素自由度/要素節点数
1706      MDFS=MDOFS      ! 1 要素あたりの自由度
1707      NEQELE=0 ! initialized
1708      !-----節点の 並進・回転 自由度の数-----
1709      NUVW=2      ! 1 節点の並進自由度の数 (2D)
1710      NHHH=NDFS - NUVW ! 1 節点の回転自由度の数
1711      !-----
1712      DO M=1, MENTS ! まず要素番号 M を特定
1713      MDF=0
1714      DO N=1, MNODES ! 要素節点番号 N を特定
1715      ND=NODELE (M, N) ! 系の節点番号 ND を特定:
1716      !                  ! この節点 ND の節点自由度の方程式番号を求めたい
1717      !-----
1718      NDF=0
1719      DO N6=1, NUVW ! まず節点の並進自由度について
1720      NDF=NDF+1

```

```

1721      MDF=MDF+1
1722      NEQELE (M, MDF)=NEQNOD (ND, N6) ! 節点自由度番号を特定
1723      END DO
1724      !-----
1725      IF (NHHH==0) CYCLE !回転自由度がなければ 次の節点へ
1726      !-----2D---
1727      MDF=MDF+1
1728      NDF=NDF+1
1729      NEQELE (M, MDF)=NEQNOD (ND, 3) ! 節点自由度番号を特定
1730      !-----
1731      IF (NDF/=NDFS) THEN
1732      STOP ' error: NDF/=NDFS /PUT_NEQELE '
1733      END IF
1734      !-----
1735      END DO ! 要素節点番号(1-MNODES)を特定
1736      IF (MDF/=MDFS) THEN
1737      STOP ' error: MDF/=MDFS /PUT_NEQELE '
1738      END IF
1739      END DO ! 要素番号(1-MENTS)を特定
1740      !-----
1741      END SUBROUTINE PUT_NEQELE
1742      !*****
1743      SUBROUTINE DATA_LOAD ! 荷重条件の設定 作業中 一応完成
1744      USE GLOBAL05
1745      IMPLICIT INTEGER (I-N), REAL (8) (A-H, O-Z)
1746      CHARACTER TEXT80*80, STRING*80
1747      !----- (LOAD) 行の読み込み-----
1748      CALL READ_LINE (TEXT80)
1749      IF (INDEX (TEXT80, ' (LOAD) ')==0) THEN
1750      STOP ' error: (LOAD) missed !'
1751      END IF
1752      WRITE (*, *) ' (LOAD) being processed'
1753      !-----
1754      !          分布荷重
1755      !   ELE=13  GBL: PA QA PB QB
1756      !   ELE=56  LCL: Pa Qb Pb Qb
1757      !-----
1758      ! INPNEQ=0  節点番号   自由度番号(1-6)   荷重の大きさ(+, -)
1759      !          NODE= 11   DOF6 :    1         LOAD: -4.5
1760      !-----

```

```

1761 ! INPNEQ=1      節点変位番号(方程式番号)   荷重の大きさ(+,-)
1762 !              NEQ = 322                      LOAD: -4.5
1763 !-----
1764 1000 CONTINUE
1765      CALL READ_LINE(TEXT80)
1766 1100 IF (INDEX(TEXT80,'END')/=0) GOTO 9000 ! END 行を読込んだら
1767 !-----分布荷重の場合 -----
1768      IF (INDEX(TEXT80,'ELE')/=0.AND.ELE_LIB/='B2D_LIN') THEN
1769          STOP ' error: "ELE=" only for B2D_LIN/(LOAD)'
1770      END IF
1771      IF (INDEX(TEXT80,'ELE')/=0.AND.INDEX(TEXT80,'GBL')/=0) GOTO 5000
1772      IF (INDEX(TEXT80,'ELE')/=0.AND.INDEX(TEXT80,'LCL')/=0) GOTO 7000
1773 !----- check ----
1774      IF (INDEX(TEXT80,'NODE')/=0.AND.INPNEQ==1) THEN
1775          STOP ' error: "NODE=" not for INPNEQ=1/(LOAD)'
1776      END IF
1777 !----- check -----
1778      IF (INDEX(TEXT80,'NEQ')/=0.AND.INPNEQ==0) THEN
1779          STOP ' error: "NEQ=" not for INPNEQ=0/(LOAD)'
1780      END IF
1781 !-----
1782      IF (INDEX(TEXT80,'NODE')==0.AND.INDEX(TEXT80,'NEQ')==0.AND.
1783 1 INDEX(TEXT80,'ELE')==0) THEN
1784          STOP ' error: "ELE="or"NODE="or"NEQ="missed/(LOAD)'
1785      END IF
1786 !-----
1787      IF (INDEX(TEXT80,'NEQ')/=0.AND.INPNEQ==1) GOTO 3000
1788      IF (INDEX(TEXT80,'NODE')/=0.AND.INPNEQ==0) GOTO 3000
1789      STOP ' error: line not acceptable /(LOAD)'
1790 !-----[節点荷重の場合]
1791 3000 KEQ=INDEX(TEXT80,'=') !等号の位置      NEQ= or NODE=
1792 !-----節点番号/方程式番号の採取-----
1793      KSTART=KEQ+1
1794      CALL GET_EQ_VAL(TEXT80,KSTART, KEND , STRING)
1795      CALL GET_INT (STRING, NVAL) ! NVAL=節点番号 or 方程式番号
1796 !----- NODE= 21 DOF6: 2   LOAD:-4.5 -----
1797      IF (INPNEQ==0) THEN
1798          KEQ=INDEX(TEXT80,':') ! ":" の位置
1799          KSTART=KEQ+1
1800          CALL GET_EQ_VAL (TEXT80,KSTART, KEND , STRING)

```

```

1801      CALL GET_INT (STRING, N6)          !自由度番号の採取
1802      TEXT80(1:KEND)=' ' ! 込完了部の空白置換
1803      END IF
1804      !-----対応する節点変位番号（方程式番号）の採取
1805      IF (INPNEQ==0) NEQ=NEQNOD (NVAL, N6)
1806      IF (INPNEQ==1) NEQ=NVAL          ! 直接入力
1807      !----- 荷重の大きさ(+, -) の採取-----
1808      KEQ=INDEX(TEXT80,':') ! ":" の位置
1809      KSTART=KEQ+1
1810      CALL GET_EQ_VAL (TEXT80,KSTART, KEND , STRING)
1811      CALL GET_REL (STRING, SVAL)
1812      IF (0<NEQ.AND.NEQ<=NDOFS) PVEC (NEQ)=PVEC (NEQ)+SVAL ! reference loda vector e
1813      !-----
1814      GOTO 1000 ! すべての設定された非ゼロの荷重成分を読み込む
1815      !----- [分布荷重の場合 (GBL:GLOBAL)]
1816      5000 KEQ=INDEX(TEXT80,'=') !等号の位置
1817      !-----要素番号の採取-----
1818      KSTART=KEQ+1
1819      CALL GET_EQ_VAL (TEXT80,KSTART, KEND , STRING)
1820      CALL GET_INT (STRING, NVAL) ! NVAL=要素番号
1821      MENT=NVAL !
1822      !----- GBL: 荷重強度 PA QA PB QB(+, -) の採取-----
1823      KEQ=INDEX(TEXT80,':') ! ":" の位置
1824      !----- 荷重強度 PA を採取
1825      KSTART=KEQ+1
1826      CALL GET_EQ_VAL (TEXT80,KSTART, KEND , STRING)
1827      CALL GET_REL (STRING, PA)
1828      !-----荷重強度 QA を採取
1829      KSTART=KEND+1
1830      CALL GET_EQ_VAL (TEXT80,KSTART, KEND , STRING)
1831      CALL GET_REL (STRING, QA) !
1832      !----- 荷重強度 PB を採取
1833      KSTART=KEND+1
1834      CALL GET_EQ_VAL (TEXT80,KSTART, KEND , STRING)
1835      CALL GET_REL (STRING, PB) !
1836      !----- 荷重強度 QB を採取
1837      KSTART=KEND+1
1838      CALL GET_EQ_VAL (TEXT80,KSTART, KEND , STRING)
1839      CALL GET_REL (STRING, QB) !
1840      !-----

```

```

1841      BALPQ(MENT,1)=BALPQ(MENT,1)+PA !   Global PA
1842      BALPQ(MENT,2)=BALPQ(MENT,2)+QA !           Global QA
1843      BALPQ(MENT,3)=BALPQ(MENT,3)+PB !   Global PB
1844      BALPQ(MENT,4)=BALPQ(MENT,4)+QB !   Global QB
1845      !-----
1846      GOTO 1000
1847      !----- [分布荷重の場合(LCL:LOCAL)]
1848      7000 KEQ=INDEX(TEXT80,'=') !等号の位置
1849      !-----要素番号の採取-----
1850      KSTART=KEQ+1
1851      CALL GET_EQ_VAL(TEXT80,KSTART, KEND , STRING)
1852      CALL GET_INT(STRING, NVAL) ! NVAL=要素番号
1853      MENT=NVAL    !
1854      !----- LCL: 荷重強度 PA QA PB  QB(+,-) の採取-----
1855      KEQ=INDEX(TEXT80,':') ! ":" の位置
1856      !----- 荷重強度 PA を採取
1857      KSTART=KEQ+1
1858      CALL GET_EQ_VAL(TEXT80,KSTART, KEND , STRING)
1859      CALL GET_REL(STRING, PA)
1860      !-----荷重強度 QA を採取
1861      KSTART=KEND+1
1862      CALL GET_EQ_VAL(TEXT80,KSTART, KEND , STRING)
1863      CALL GET_REL(STRING, QA) !
1864      !----- 荷重強度 PB を採取
1865      KSTART=KEND+1
1866      CALL GET_EQ_VAL(TEXT80,KSTART, KEND , STRING)
1867      CALL GET_REL(STRING, PB) !
1868      !----- 荷重強度 QB を採取
1869      KSTART=KEND+1
1870      CALL GET_EQ_VAL(TEXT80,KSTART, KEND , STRING)
1871      CALL GET_REL(STRING, QB) !
1872      !-----
1873      CALPQ(MENT,1)=CALPQ(MENT,1)+PA ! Local PA
1874      CALPQ(MENT,2)=CALPQ(MENT,2)+QA ! Local QA
1875      CALPQ(MENT,3)=CALPQ(MENT,3)+PB ! Local PB
1876      CALPQ(MENT,4)=CALPQ(MENT,4)+QB ! Local QB
1877      !-----
1878      GOTO 1000
1879      !-----
1880      9000 CONTINUE

```

```

1881      IF (ELE_LIB==' B2D_LIN' ) THEN
1882      !----- assemble PQELE (ments, 6) to  PVEC (ndofs)
1883      DO M=1, MENTS
1884      !-----等価節点外力 ----- debugged and updated  18August2008
1885      PA=BALPQ (M, 1) ;QA=BALPQ (M, 2) ! Global PA QA
1886      PB=BALPQ (M, 3) ;QB=BALPQ (M, 4) ! Global PB QB
1887      !---- debugged 18 August 2008: thanks to CAE SS2008 機械院生 6 名
1888      CALL GLOBAL_PQ (M, PA, QA, PB, QB) ! PQELE (ments, mdofs) debugged 18Aug 2008
1889      !-----
1890      PA=CALPQ (M, 1) ;QA=CALPQ (M, 2) ! Local PA QA
1891      PB=CALPQ (M, 3) ;QB=CALPQ (M, 4) ! Local PB QB
1892      CALL LOCAL_PQ (M, PA, QA, PB, QB) ! PQELE (ments, mdofs)
1893      !-----
1894      DO ND=1, MDOFS
1895      NEQ=NEQELE (M, ND) ;PVAL=PQELE (M, ND)
1896      IF (0<NEQ. AND. NEQ<=MDOFS)  PVEC (NEQ)=PVEC (NEQ) - PVAL ! - (拘束力)
1897      END DO ! all element dofs
1898      END DO ! all elements
1899      END IF      !      IF (ELE_LIB==' B2D_LIN' )
1900      !-----非ゼロ 荷重成分の画面表示
1901      JJ=1
1902      DO J=1,  MDOFS
1903      IF (0.1E-15<ABS (PVEC (J))) THEN
1904      JJ=JJ+1
1905      WRITE (*, 9100) J,  PVEC (J)
1906      9100 FORMAT (' nonzero PVEC (' , 15, ') = ', E14.5)
1907      END IF
1908      IF (INWAIT/=0. AND. MOD (JJ, 21)==0) CALL WAIT
1909      END DO
1910      !-----
1911      END SUBROUTINE  DATA_LOAD ! 荷重条件の設定
1912      !*****
1913      SUBROUTINE GET_NAME_EQ_VAL (NAME, SVAL, NVAL)
1914      !-----
1915      !      書式      NABC=-879      &
1916      !      ABCD=-19.77E-3
1917      !      の読込専用サブルーチン
1918      !      reads only one line with NABC=-98707 or BCD=-0.980E-03
1919      !-----
1920      !      NAME: 変数名 (最大 6 文字からなる文字列)

```



```

1921  !          SVAL : 実数型変数 NAME の 実数型数値データ
1922  !                                     (NAME が整数型の場合は無効)
1923  !          NVAL : 整数型変数 NAME の 整数型数値データ
1924  !                                     (NAME が実数型の場合は無効)
1925  !-----
1926      IMPLICIT INTEGER (I-N), REAL (8) (A-H, O-Z)
1927      CHARACTER TEXT80*80, STRING*80, NAME*6
1928  !-----
1929      1000 CALL READ_LINE (TEXT80)
1930  !----- END 行を読み込んだら NAME='END' and 脱出-----
1931      IF (INDEX (TEXT80, 'END')/=0) THEN
1932          NAME='END'    ! 便宜的措置
1933          GOTO 9000    ! 脱出
1934      END IF
1935      IF (INDEX (TEXT80, '=')==0) THEN
1936          STOP          ' error: "=" missed/GET_NAME_EQ_VAL '
1937      ENDIF
1938      KEQ=INDEX (TEXT80, '=') ! 等号の位置
1939  !-----等号の左側の文字列 (変数名) を採取
1940      KSTART=KEQ-1
1941      CALL GET_NAME_EQ (TEXT80, KEQ, STRING) ! STRING 左づめ
1942  !-----STRING*80 のなかから先頭の6文字を採取して NAME*6 へ格納
1943      DO 2000 K=1, 6
1944          NAME (K:K)=STRING (K:K)
1945      2000 CONTINUE
1946  !-----実数型の変数名か 整数型の変数名か
1947      CALL GET_NAME_TYP (NAME, INTREL)
1948  !-----等号の右側の文字列 (数値) を採取-----
1949      KSTART=KEQ+1
1950      CALL GET_EQ_VAL (TEXT80, KSTART, KEND, STRING) ! STRING 左づめ
1951  !-----変数名が実数型・整数型により分岐
1952      IF (INTREL==0) GOTO 3000
1953      IF (INTREL==1) GOTO 4000
1954      WRITE (6, *) NAME, ' error: INTREL incorrect/GET_NAME_EQ_VAL '
1955      STOP
1956  !-----以上が単なる変数名 (文字列) の処理
1957      3000 CONTINUE
1958  !=====
1959  !          実数型変数名の場合
1960  !=====文字列を実数型数値データに変換=====

```

```

1961      CALL GET_REL (STRING, SVAL)
1962      WRITE (6, *) NAME, ' := ', SVAL
1963      GOTO 9000
1964      !-----
1965      4000 CONTINUE
1966      !=====
1967      !           整数型変数名の場合
1968      !=====文字列を整数型数値データに変換=====
1969      CALL GET_INT (STRING, NVAL)
1970      WRITE (6, *) NAME, ' := ', NVAL
1971      !-----読み込み終了
1972      9000 CONTINUE
1973      END SUBROUTINE GET_NAME_EQ_VAL !書式 NABC=-879 & ABCD=-19.77E-3 の読込
1974      !*****
1975      SUBROUTINE OPEN_FD_OUT
1976      USE GLOBAL05
1977      IMPLICIT INTEGER (I-N), REAL (8) (A-H, O-Z)
1978      !-----
1979      IF (IAFDL==0) THEN      ! first opening
1980      OPEN (UNIT=11, IOSTAT=IOS, ERR=9998,
1981      1FILE='C:\¥frame05¥frame05_out.txt', ! 固定した ファイル 名前
1982      1STATUS='new',
1983      1ACCESS='SEQUENTIAL',
1984      1FORM='FORMATTED' )
1985      GOTO 1000
1986      9998 WRITE (*, *) IOS, ' error: cannot open C:\¥frame05¥frame05_out.txt'
1987      STOP
1988      1000 WRITE (*, *) ' first opened: C:\¥frame05¥frame05_out.txt'
1989      END IF      ! first opening
1990      !-----
1991      IF (1<=IAFDL) THEN      ! re-open the existing file
1992      OPEN (UNIT=11, IOSTAT=IOS, ERR=9888,
1993      1FILE='C:\¥frame05¥frame05_out.txt', ! 固定した ファイル 名前
1994      1STATUS='old',           ! すでに存在している必要あり
1995      1ACCESS='SEQUENTIAL',
1996      1FORM='FORMATTED',
1997      1POSITION='APPEND' )
1998      GOTO 1010
1999      9888 WRITE (*, *) IOS, ' error: can not open C:\¥frame05¥frame05_out.txt'
2000      STOP

```

```

2001      1010 WRITE(*,*) ' re-opened: C:\frame05\frame05_out.txt'
2002          END IF          ! re-open the existing file
2003      !-----
2004          IAFDFL=IAFDFL+1      ! これで奇数になったはず
2005      !-----
2006          END SUBROUTINE OPEN_FD_OUT
2007      !*****
2008          SUBROUTINE CLOSE_FD_OUT
2009              USE GLOBAL05
2010              IMPLICIT INTEGER (I-N), REAL (8) (A-H, O-Z)
2011      !-----
2012              CLOSE (UNIT=11, IOSTAT=IOS, ERR=9999, STATUS=' KEEP' )
2013              GOTO 9000
2014      9999 WRITE(6,*) '      error: cannot close C:\frame05\frame05_out.txt'
2015              STOP
2016      9000 WRITE(6,*) '      closed: C:\frame05\frame05_out.txt'
2017      !-----
2018          IFDFL=IAFDFL+1 ! これで偶数になったはず
2019      !-----
2020          END SUBROUTINE CLOSE_FD_OUT
2021      !*****
2022          SUBROUTINE CG_FD_INI
2023              USE GLOBAL05
2024              IMPLICIT INTEGER (I-N), REAL (8) (A-H, O-Z)
2025      !-----
2026      !              初期形状の図化
2027      !      T2D_LIN & B2D_LIN の初期形状の図化は、線分の描画だけなので
2028      !      基本的に同じ subroutine T2D_B2D_INI が使える
2029      !-----
2030      8000 IF (ICGINI/=0) CALL      T2D_B2D_INI ! frame05_ini for Win3D
2031      !-----
2032      9000 CONTINUE
2033          END SUBROUTINE CG_FD_INI
2034      !*****
2035          SUBROUTINE T2D_B2D_INI
2036              USE GLOBAL05
2037              IMPLICIT INTEGER (I-N), REAL (8) (A-H, O-Z)
2038      !-----
2039              OPEN (UNIT=33, IOSTAT=IOS, ERR=3333,
2040              1FILE=' C:\frame05\frame05_ini.dat', STATUS=' new',

```

```

2041      1ACCESS=' SEQUENTIAL', FORM=' FORMATTED')
2042      GOTO 3000
2043      3333 WRITE(*,*) IOS,' error: can not open C:¥frame05¥frame05_ini.dat'
2044      STOP
2045      3000 WRITE(*,*) ' opened: C:¥frame05¥frame05_ini.dat'
2046      !----- 節点の XYZ 座標
2047      DO N=1,NODES
2048      XU=XYZNOD(N,1);YV=XYZNOD(N,2);ZW=0.0
2049      WRITE(33,' (17,2(X,E15.5))') N, XU, YV, ZW ! 節点番号 X Y Z
2050      END DO
2051      !-----要素を定義する節点番号 (要素番号は書かない!)
2052      WRITE(33,' (A)') ' Faces:'
2053      DO M=1, MENTS
2054      WRITE(33,' (217,A)') (NODELE(M,N),N=1,MNODES), ' ' !節点番号
2055      END DO
2056      !-----
2057      CLOSE(UNIT=33, IOSTAT=IOS, ERR=3344, STATUS=' KEEP')
2058      GOTO 4000
2059      3344 WRITE(6,*) ' error: cannot close C:¥frame05¥frame05_ini.dat'
2060      STOP
2061      4000 WRITE(6,*) ' closed: C:¥frame05¥frame05_ini.dat'
2062      !-----
2063      END SUBROUTINE T2D_B2D_INI
2064      !*****
2065      SUBROUTINE LOCAL_PQ(MENT, PA, QA, PB, QB) ! 25/Nov/98
2066      USE GLOBAL05
2067      IMPLICIT INTEGER (I-N), REAL (8) (A-H, O-Z)
2068      !-----
2069      ! Local 系で作用する分布荷重の荷重強度 (PA, QA, PB, QB) から
2070      ! その要素の等価節点外力 PQELE(ment, 6) を計算する
2071      !-----
2072      NA=NODELE(MENT, 1);NB=NODELE(MENT, 2)
2073      XA=XYZNOD(NA, 1);YA=XYZNOD(NA, 2)
2074      XB=XYZNOD(NB, 1);YB=XYZNOD(NB, 2)
2075      SP=SQRT((XB-XA)*(XB-XA)+(YB-YA)*(YB-YA))
2076      SS=(YB-YA)/SP;CC=(XB-XA)/SP
2077      !----- (PA, PB) -----
2078      PQELE(MENT, 1)=PQELE(MENT, 1)-(2.*PA+PB)*CC*SP/6.0
2079      PQELE(MENT, 2)=PQELE(MENT, 2)-(2.*PA+PB)*SS*SP/6.0
2080      PQELE(MENT, 4)=PQELE(MENT, 4)-(PA+2.*PB)*CC*SP/6.0

```

```

2081      PQELE (MENT, 5)=PQELE (MENT, 5) - (PA+2. *PB) *SS*SP/6. 0
2082      !----- (QA, QB) -----
2083      PQELE (MENT, 1)=PQELE (MENT, 1) + (7. *QA+3. *QB) *SS*SP/20.
2084      PQELE (MENT, 2)=PQELE (MENT, 2) - (7. *QA+3. *QB) *CC*SP/20.
2085      PQELE (MENT, 3)=PQELE (MENT, 3) - (3. *QA+2. *QB) *SP*SP/60.
2086      PQELE (MENT, 4)=PQELE (MENT, 4) + (3. *QA+7. *QB) *SS*SP/20.
2087      PQELE (MENT, 5)=PQELE (MENT, 5) - (3. *QA+7. *QB) *CC*SP/20.
2088      PQELE (MENT, 6)=PQELE (MENT, 6) + (2. *QA+3. *QB) *SP*SP/60.
2089      !-----
2090      END SUBROUTINE LOCAL_PQ
2091      !*****
2092      SUBROUTINE GLOBAL_PQ (MENT, PA, QA, PB, QB) ! 25/Nov/98
2093      USE GLOBAL05
2094      IMPLICIT INTEGER (I-N), REAL (8) (A-H, O-Z)
2095      !-----
2096      ! Global 系で作用する分布荷重の荷重強度 (PA, QA, PB, QB) から
2097      ! その要素の等価節点外力 PQELE (ment, 6) を計算する
2098      !-----
2099      NA=NODELE (MENT, 1) ; NB=NODELE (MENT, 2)
2100      XA=XYZNOD (NA, 1) ; YA=XYZNOD (NA, 2)
2101      XB=XYZNOD (NB, 1) ; YB=XYZNOD (NB, 2)
2102      SP=SQRT ( (XB-XA) * (XB-XA) + (YB-YA) * (YB-YA) )
2103      SS= (YB-YA) /SP ; CC= (XB-XA) /SP
2104      SS2=SS*SS ; CC2=CC*CC ; SP2=SP*SP
2105      !----- (PA, PB) -----
2106      PQELE (MENT, 1)=PQELE (MENT, 1) - (7. *PA+3. *PB) *ABS (SS) *SP/20.
2107      PQELE (MENT, 3)=PQELE (MENT, 3) + (3. *PA+2. *PB) *SIGN (1. , SS) *SS2*SP2/60.
2108      PQELE (MENT, 4)=PQELE (MENT, 4) - (3. *PA+7. *PB) *ABS (SS) *SP/20.
2109      PQELE (MENT, 6)=PQELE (MENT, 6) - (2. *PA+3. *PB) *SIGN (1. , SS) *SS2*SP2/60.
2110      !----- (QA, QB) -----
2111      PQELE (MENT, 2)=PQELE (MENT, 2) - (7. *QA+3. *QB) *ABS (CC) *SP/20.
2112      PQELE (MENT, 3)=PQELE (MENT, 3) - (3. *QA+2. *QB) *SIGN (1. , CC) *CC2*SP2/60.
2113      PQELE (MENT, 5)=PQELE (MENT, 5) - (3. *QA+7. *QB) *ABS (CC) *SP/20.
2114      PQELE (MENT, 6)=PQELE (MENT, 6) + (2. *QA+3. *QB) *SIGN (1. , CC) *CC2*SP2/60.
2115      !-----
2116      END SUBROUTINE GLOBAL_PQ
2117      !*****
2118      SUBROUTINE CG_FD_DEF
2119      USE GLOBAL05
2120      IMPLICIT INTEGER (I-N), REAL (8) (A-H, O-Z)

```

```

2121  !-----
2122  !   まず変形の図化に必要なデータ UVWNOD (NODES, 2) を
2123  !           TTLDOF (NDOFS) から 生成する.
2124  !           NEQNOD (NODES, 3) または NEQELE (MENTS, MDOFS) を参考にして,
2125  !           TTLDOF (NDOFS) から 並進変位 (U, V, W) のみを取り出す.
2126  !   実際の作図作業は,
2127  !   XYZNOD (NODES, 2), UVWNOD (NODES, 2), NODELE (MENTS, MNODES)
2128  !           の情報を用いて行う
2129  !-----
2130      UVWNOD=0.0
2131      IF (INPNEQ==1) GOTO 1000
2132  !----- INPNEQ=1 の場合: NEQNOD (ND, N6) は信頼できない
2133      DO N=1, NODES ! すべての節点について
2134          DO N3=1, 2 ! その節点の 2 個の並進変位について
2135              NEQ=NEQNOD (N, N3) !
2136              IF (NEQ<0. OR. NDOFS<NEQ) THEN !
2137                  STOP ' error: NEQ ? /CG_FD_DEF' !
2138              ENDIF
2139              IF (NEQ/=0) UVWNOD (N, N3)=TTLDOF (NEQ) !
2140              IF (NEQ==0) UVWNOD (N, N3)=0.0 !
2141          END DO ! その節点の 3 個の並進変位について
2142      END DO ! すべての節点について
2143      GOTO 8000
2144  !-----
2145      1000 CONTINUE
2146  !----- INPNEQ=1 の場合
2147      NDOFN=MDOFS/MNODES ! 1 節点あたりの自由度数=要素自由度数/要素節点数
2148  !----- 節点の並進自由度の数-----
2149      NUVW=2 ! 1 節点の並進自由度の数 (2D)
2150  !-----
2151      DO M=1, MENTS ! まず要素番号 M を特定
2152          DO N=1, MNODES ! その要素節点番号 N (=1, MNODES) を特定
2153              ND=NODELE (M, N) ! 系 節点番号 ND を特定:
2154  !----- この節点 ND の並進変位の方程式番号を求めたい
2155              MDF=(N-1)*NDOFN ! (1, MDOFS) のうちの何番目か
2156              DO N3=1, NUVW ! 節点の並進自由度について
2157                  NEQ=NEQELE (M, MDF+N3) ! 並進変位の方程式番号を特定
2158                  IF (NEQ<0. OR. NDOFS<NEQ) THEN !
2159                      STOP ' error: NEQ ? /CG_FD_def' !
2160                  ENDIF

```

```

2161      IF (NEQ/=0) UVWNOD (ND, N3)=TTLDOF (NEQ)      !
2162      IF (NEQ==0) UVWNOD (ND, N3)=0. 0      !
2163      END DO ! 節点の並進自由度について
2164  !-----
2165      END DO ! 要素節点番号 (1-MNODES) を特定
2166      END DO ! 要素番号 (1-MENTS) を特定
2167  !-----
2168      8000 IF (ICGDEF/=0) THEN
2169          IF (ELE_LIB==' T2D_LIN') CALL      T2D_DEF ! frame05_def for Win3D
2170          IF (ELE_LIB==' B2D_LIN') CALL      B2D_DEF ! frame05_def for Win3D
2171      ENDIF
2172      IF (ICGMLS/=0) THEN
2173          CALL XLS_T2D_B2D_DEF ! frame05_xls for T2D_LIN & B2D_LIN
2174      ENDIF
2175  !-----
2176      9000 CONTINUE
2177      END SUBROUTINE CG_FD_DEF
2178  !*****
2179      SUBROUTINE T2D_DEF
2180      USE GLOBAL05
2181      IMPLICIT INTEGER (I-N), REAL (8) (A-H, O-Z)
2182  !-----
2183  !  T2D_LIN の変形は単なる線分の描画にすぎない
2184  !-----
2185      OPEN (UNIT=55, IOSTAT=IOS, ERR=5555,
2186      1 FILE='C:\¥frame05¥frame05_def.dat',
2187      1 STATUS='new',
2188      1 ACCESS=' SEQUENTIAL',
2189      1 FORM=' FORMATTED')
2190      GOTO 5000
2191      5555 WRITE(*,*) IOS, ' error: can not open C:\¥frame05¥frame05_def.dat '
2192      STOP
2193      5000 WRITE(*,*) ' opened: C:\¥frame05¥frame05_def.dat'
2194  !----- 節点の XYZ 座標
2195      DO N=1, NODES
2196          XU=XYZNOD (N, 1)+ SCGDEF * UVWNOD (N, 1)
2197          YV=XYZNOD (N, 2)+ SCGDEF * UVWNOD (N, 2)
2198          ZW=0. 0 !
2199          WRITE(55, '(I7, 3(X, E15. 5))') N, XU, YV, ZW ! 節点番号 X Y Z
2200      END DO

```

```

2201  !-----要素を定義する節点番号 (要素番号は書かない!)
2202      WRITE(55,'(A)') 'Faces:'
2203      DO M=1, MENTS
2204          WRITE(55,'(2I7,A)') (NODELE(M,N),N=1,MNODES), '. ' !節点番号
2205      END DO
2206  !-----
2207      CLOSE(UNIT=55, IOSTAT=IOS, ERR=5566, STATUS=' KEEP')
2208      GOTO 6000
2209  5566 WRITE(6,*) ' error: can not close C:\frame05\frame05_def.dat'
2210      STOP
2211  6000 WRITE(6,*) ' closed: C:\frame05\frame05_def.dat'
2212  !-----
2213      END SUBROUTINE T2D_DEF
2214  !*****
2215      SUBROUTINE B2D_DEF
2216          USE GLOBAL05
2217          IMPLICIT INTEGER (I-N), REAL(8) (A-H, O-Z)
2218  !-----
2219  !      B2D_LIN の変形図は曲線図形のためちょっと厄介
2220  !      並進変位 UVWNOD(NODES, 2)の他に回転角 (HA, HB) も必要
2221  !      one 要素を NCGDEF 区間に均等区分
2222  !-----
2223      OPEN(UNIT=55, IOSTAT=IOS, ERR=5555,
2224      1 FILE='C:\frame05\frame05_def.dat',
2225      1 STATUS=' new',
2226      1 ACCESS=' SEQUENTIAL',
2227      1 FORM=' FORMATTED')
2228      GOTO 5000
2229  5555 WRITE(*,*) IOS, ' error: can not open C:\frame05\frame05_def.dat '
2230      STOP
2231  5000 WRITE(*,*) ' opened: C:\frame05\frame05_def.dat'
2232  !----- nodes 個 ある 節点の 変位後の 座標
2233      DO N=1, NODES
2234          XU=XYZNOD(N, 1)+ SCGDEF * UVWNOD(N, 1)
2235          YV=XYZNOD(N, 2)+ SCGDEF * UVWNOD(N, 2)
2236          ZW=0.0 ! 2D
2237          WRITE(55,'(I7,3(X,E15.5))') N, XU, YV, ZW ! 節点番号 X Y Z
2238      END DO
2239      IF(NCGDEF==1) GOTO 8000 ! 要素を内部分割しないとき
2240  !----- さらに 各要素を NCGDEF 分割 する (NCGDEF-1)個の内部節点

```



```

2241      NC=NODES      ! nodes+1, nodes+ments*(ncgdef-1)
2242      DO M=1,MENTS ! for all elements
2243      NA=NODELE(M,1);NB=NODELE(M,2) ! node A & B
2244      XA=XYZNOD(NA,1);YA=XYZNOD(NA,2)
2245      XB=XYZNOD(NB,1);YB=XYZNOD(NB,2)
2246      EL=SQRT((XB-XA)*(XB-XA)+(YB-YA)*(YB-YA))
2247      S=(YB-YA)/EL;C=(XB-XA)/EL
2248      SS=S*S;CC=C*C;EL2=EL*EL
2249      MTYP=MATSEC(M)
2250      EA=SECTYP(MTYP,1);EI=SECTYP(MTYP,2)
2251      !----- 6 dofs in GBL system -----
2252      UA=0.0; UB=0.0;VA=0.0;VB=0.0;HA=0.0;HB=0.0
2253      IF (0<NEQELE(M,1).AND. NEQELE(M,1)<=NDOFS) THEN
2254      UA=TTLDOF(NEQELE(M,1)) ; END IF
2255      IF (0<NEQELE(M,2).AND. NEQELE(M,2)<=NDOFS) THEN
2256      VA=TTLDOF(NEQELE(M,2)) ; END IF
2257      IF (0<NEQELE(M,3).AND. NEQELE(M,3)<=NDOFS) THEN
2258      HA=TTLDOF(NEQELE(M,3)) ; END IF
2259      !-----
2260      IF (0<NEQELE(M,4).AND. NEQELE(M,4)<=NDOFS) THEN
2261      UB=TTLDOF(NEQELE(M,4)) ; END IF
2262      IF (0<NEQELE(M,5).AND. NEQELE(M,5)<=NDOFS) THEN
2263      VB=TTLDOF(NEQELE(M,5)) ; END IF
2264      IF (0<NEQELE(M,6).AND. NEQELE(M,6)<=NDOFS) THEN
2265      HB=TTLDOF(NEQELE(M,6)) ; END IF
2266      !----- translations in LCL system -----
2267      CUA= UA*C+VA*S;CUB= UB*C+VB*S
2268      CVA=-UA*S+VA*C;CVB=-UB*S+VB*C
2269      !----- LCL: PA,QA,PB,QB converted from GBL: PA,QA,PB,QB --
2270      PA=CALPQ(M,1)+BALPQ(M,1)*ABS(S)*C+ BALPQ(M,2)*ABS(C)*S
2271      QA=CALPQ(M,2)-BALPQ(M,1)*ABS(S)*S+ BALPQ(M,2)*ABS(C)*C
2272      PB=CALPQ(M,3)+BALPQ(M,3)*ABS(S)*C+ BALPQ(M,4)*ABS(C)*S
2273      QB=CALPQ(M,4)-BALPQ(M,3)*ABS(S)*S+ BALPQ(M,4)*ABS(C)*C
2274      !-----
2275      DX=EL/FLOAT(NCGDEF)
2276      DO N=1,NCGDEF-1 ! for ncgdef segments
2277      NC=NC+1
2278      CX=DX*FLOAT(N);CY=0.0
2279      CALL UO_LCL(CUO,EL, CUA, CUB, CX)
2280      CALL VO_LCL(CVO,EL, CVA,HA, CVB,HB,CX)

```

```

2281      CALL UP_LCL (CUP, EA, EL, PA, PB, CX) ! EA=0.0 for bending_only 01/Feb/99
2282      CALL VQ_LCL (CVQ, EI, EL, QA, QB, CX)
2283      CU=CU0+CUP; CV=CV0+CVQ
2284      CXU=CX+SCGDEF*CU; CYV=CY+SCGDEF*CV
2285      !----- transformed to GBL system -----
2286      XU=XA+ CXU*C-CYV*S; YV=YA+ CXU*S+CYV*C
2287      ZW=0.0
2288      WRITE(55, '(17, 3(X, E15.5))') NC, XU, YV, ZW ! 区間番号 X Y Z 18Aug08
2289      END DO      ! ncgdef segments
2290      END DO      ! for all elements
2291      8000 CONTINUE
2292      !-----要素を定義する節点番号（要素番号は書かない！）
2293      WRITE(55, '(A)') 'Faces:'
2294      !-----
2295      IF (NCGDEF==1) THEN
2296      DO M=1, MENTS
2297      WRITE(55, '(217, A)') (NODELE (M, N), N=1, MNODES), ' ' !節点番号
2298      END DO
2299      END IF
2300      IF (NCGDEF==1) GOTO 7000
2301      !----- 各要素の NCGDEF 本の線分の定義： (NCGDEF-1)個の内部節点
2302      NC=NODES ! nodes+1, nodes+ments*(ncgdef-1)
2303      DO M=1, MENTS ! for all elements
2304      NA=NODELE (M, 1); NB=NODELE (M, 2) ! node A & B
2305      NC=NC+1
2306      WRITE(55, '(217, A)') NA, NC, ' ' !最左区間 節点番号
2307      !-----
2308      IF (3<=NCGDEF) THEN
2309      DO N=1, NCGDEF-2 ! for ncgdef segments
2310      WRITE(55, '(217, A)') NC, NC+1, ' ' !中間区間 節点番号
2311      NC=NC+1
2312      END DO      ! ncgdef segments
2313      END IF
2314      !-----
2315      WRITE(55, '(217, A)') NC, NB, ' ' ! 最右区間 節点番号
2316      END DO ! for all elements
2317      7000 CONTINUE
2318      !-----
2319      CLOSE (UNIT=55, IOSTAT=IOS, ERR=5566, STATUS=' KEEP' )
2320      GOTO 6000

```

```

2321      5566 WRITE(6,*) ' error: can not close C:\¥frame05¥frame05_def.dat'
2322          STOP
2323      6000 WRITE(6,*) ' closed: C:\¥frame05¥frame05_def.dat'
2324      !-----
2325          END SUBROUTINE B2D_DEF
2326      !*****
2327          SUBROUTINE UO_LCL(UO, EL, UA, UB, X)
2328              IMPLICIT INTEGER (I-N), REAL (8) (A-H, O-Z)
2329      !-----
2330      !          dof effects u0(x) in LCL system 12APR2005
2331      !-----
2332          UO=(1.-X/EL)*UA+(X/EL)*UB
2333      !-----
2334          END SUBROUTINE UO_LCL
2335      !*****
2336          SUBROUTINE VO_LCL(VO, EL, VA, HA, VB, HB, X)
2337              IMPLICIT INTEGER (I-N), REAL (8) (A-H, O-Z)
2338      !-----
2339      !          dof effects v0(x) in LCL system 12APR2005
2340      !-----
2341          S=X/EL
2342          VO= VA*( 2.*S*S*S-3.*S*S +1.)
2343          1 + EL*HA*( S*S*S-2.*S*S+S )
2344          2 + VB*(-2.*S*S*S+3.*S*S )
2345          3 + EL*HB*( S*S*S -S*S )
2346      !-----
2347          END SUBROUTINE VO_LCL
2348      !*****
2349          SUBROUTINE UP_LCL(UP, EA, EL, PA, PB, X)
2350              IMPLICIT INTEGER (I-N), REAL (8) (A-H, O-Z)
2351      !-----
2352      !          load effects up(x) in LCL system 12APR2005
2353      !-----
2354          IF (EA==0.0) THEN ! for bending only analysis 02/Feb/99
2355              UP=0.0
2356              GOTO 9000
2357          END IF ! 01/Feb/99
2358      !-----
2359          S=X/EL;S6=1./6.
2360          UP=+(EL*EL/EA)*( -S6*S*S*S*(PB-PA) ! 02 APRIL 2004

```

```

2361      1          -0.5*PA*S*S          ! 02 APRIL 2004
2362      2          +S6*(2.*PA+PB)*S      ) ! 02 APRIL 2004
2363      !-----
2364      9000 CONTINUE
2365      !-----
2366      END SUBROUTINE UP_LCL
2367      !*****
2368      SUBROUTINE VQ_LCL (VQ, EI, EL, QA, QB, X)
2369      IMPLICIT INTEGER (I-N), REAL (8) (A-H, O-Z)
2370      !-----
2371      !      load effects vq(x) in LCL system 12APR2005
2372      !-----
2373      S=X/EL; SVAL=EL*EL*X*X/(120.*EI)
2374      VQ=SVAL*(QA*(-S*S*S+5.*S*S-7.*S+3.))
2375      1          +QB*(S*S*S-3.*S+2.)) ! updated 18July2006
2376      !-----
2377      !      thanks to 数デ3年生 SS06 ; corrected from +QB*(S*S*S-3.*S*S+2.))
2378      !-----
2379      END SUBROUTINE VQ_LCL
2380      !*****
2381      SUBROUTINE XLS_T2D_B2D_DEF ! 27/JAN/2005
2382      USE GLOBAL05
2383      IMPLICIT INTEGER (I-N), REAL (8) (A-H, O-Z)
2384      !-----
2385      !      B2D_LIN の変形図は曲線図形のためちょっと厄介
2386      !      並進変位 UVWNOD (NODES, 2) の他に回転角 (HA, HB) も必要
2387      !      one 要素を NCGDEF 区間に均等区分
2388      !-----
2389      OPEN (UNIT=55, IOSTAT=IOS, ERR=5555,
2390      1 FILE='C:\¥frame05¥frame05_xls.xls',
2391      1 STATUS='new',
2392      1 ACCESS='SEQUENTIAL',
2393      1 FORM='FORMATTED')
2394      GOTO 5000
2395      5555 WRITE(*,*) IOS, ' error: can not open C:\¥frame05¥frame05_xls.xls'
2396      STOP
2397      5000 WRITE(*,*) ' opened: C:\¥frame05¥frame05_xls.xls'
2398      !----- 要素ごとに両端の節点の XYZ 座標を書き出す (要素と要素の間に空白行) -----
2399      DO M=1, MENTS ! for all elements
2400      WRITE(55, '(A)') ' ' ! 空白行 (要素の切れ目)

```

```

2401  !-----
2402      NA=NODELE (M, 1) ; NB=NODELE (M, 2)  ! node A & B
2403      XA=XYZNOD (NA, 1) ; YA=XYZNOD (NA, 2)
2404      XB=XYZNOD (NB, 1) ; YB=XYZNOD (NB, 2)
2405      EL=DSQRT ( (XB-XA)*(XB-XA)+(YB-YA)*(YB-YA) )
2406      S=(YB-YA)/EL ; C=(XB-XA)/EL
2407      SS=S*S ; CC=C*C ; EL2=EL*EL
2408      MTYP=MATSEC (M)
2409      EA=SECTYP (MTYP, 1) ; EI=SECTYP (MTYP, 2)
2410  !----- node A -----
2411      XUA=XA+ SCGDEF * UVWNOD (NA, 1)
2412      YVA=YA+ SCGDEF * UVWNOD (NA, 2)
2413      WRITE (55, ' (15, ">nodeA", 4E15.5)') M, XA, YA, XUA, YVA ! XUA YVA
2414  !-----
2415      IF (NCGDEF==1) GO TO 8000 ! 要素を内部分割しないとき go to node B
2416      IF (ELE_LIB=='T2D_LIN') GO TO 8000 ! for T2D_LIN go to node B
2417  !----- さらに 各要素を NCGDEF 分割 する (NCGDEF-1) 個の内部節点
2418  !----- 6 dofs in GBL system -----
2419      UA=0.0 ; UB=0.0 ; VA=0.0 ; VB=0.0 ; HA=0.0 ; HB=0.0
2420      IF (0<NEQELE (M, 1) . AND. NEQELE (M, 1)<=NDOFS) THEN
2421          UA=TTLD OF (NEQELE (M, 1)) ; END IF
2422      IF (0<NEQELE (M, 2) . AND. NEQELE (M, 2)<=NDOFS) THEN
2423          VA=TTLD OF (NEQELE (M, 2)) ; END IF
2424      IF (0<NEQELE (M, 3) . AND. NEQELE (M, 3)<=NDOFS) THEN
2425          HA=TTLD OF (NEQELE (M, 3)) ; END IF
2426  !-----
2427      IF (0<NEQELE (M, 4) . AND. NEQELE (M, 4)<=NDOFS) THEN
2428          UB=TTLD OF (NEQELE (M, 4)) ; END IF
2429      IF (0<NEQELE (M, 5) . AND. NEQELE (M, 5)<=NDOFS) THEN
2430          VB=TTLD OF (NEQELE (M, 5)) ; END IF
2431      IF (0<NEQELE (M, 6) . AND. NEQELE (M, 6)<=NDOFS) THEN
2432          HB=TTLD OF (NEQELE (M, 6)) ; END IF
2433  !----- translations in LCL system -----
2434      CUA= UA*C+VA*S ; CUB= UB*C+VB*S
2435      CVA=-UA*S+VA*C ; CVB=-UB*S+VB*C
2436  !----- LCL: PA, QA, PB, QB converted from GBL: PA, QA, PB, QB ---
2437      PA=CALPQ (M, 1)+BALPQ (M, 1)*ABS (S)*C+ BALPQ (M, 2)*ABS (C)*S
2438      QA=CALPQ (M, 2)-BALPQ (M, 1)*ABS (S)*S+ BALPQ (M, 2)*ABS (C)*C
2439      PB=CALPQ (M, 3)+BALPQ (M, 3)*ABS (S)*C+ BALPQ (M, 4)*ABS (C)*S
2440      QB=CALPQ (M, 4)-BALPQ (M, 3)*ABS (S)*S+ BALPQ (M, 4)*ABS (C)*C

```

```

2441  !-----
2442      DX=EL/FLOAT (NCGDEF)
2443      DO N=1,NCGDEF-1 !   for ncgdef segments
2444      CX=DX*FLOAT (N);CY=0.0 ! element coordinates  (x,y)
2445      CALL UO_LCL (CU0,EL, CUA,      CUB,      CX)
2446      CALL VO_LCL (CV0,EL,      CVA,HA,      CVB,HB,CX)
2447      CALL UP_LCL (CUP,EA,EL,PA,PB,CX) ! EA=0.0 for bending_only 01/Feb/99
2448      CALL VQ_LCL (CVQ,EI,EL,QA,QB,CX)
2449      CU=CU0+CUP; CV=CV0+CVQ
2450      CXU=CX+SCGDEF*CU;CYV=CY+SCGDEF*CV
2451  !----- transformed to GBL system -----
2452      XU=XA+ CXU*C-CYV*S ; X=XA+ CX*C-CY*S
2453      YV=YA+ CXU*S+CYV*C ; Y=YA+ CX*S+CY*C
2454      WRITE(55,' (I11,4E15.5)') N, X, Y, XU, YV !  XU  YV
2455      END DO ! for all (ncgdef) segments
2456  !-----
2457      8000 CONTINUE ! for node B
2458  !----- node B -----
2459      XUB=XB + SCGDEF * UVWNOD (NB,1)
2460      YVB=YB + SCGDEF * UVWNOD (NB,2)
2461      WRITE(55,' (I5,">nodeB",4E15.5)') M, XB,YB, XUB, YVB ! XUB  YVB
2462  !-----
2463      END DO ! M for all elements
2464  !-----
2465      CLOSE (UNIT=55, IOSTAT=IOS, ERR=5566, STATUS=' KEEP' )
2466      GOTO 6000
2467      5566 WRITE(6,*) ' error: can not close C:¥frame05¥frame05_xls.xls'
2468      STOP
2469      6000 WRITE(6,*) ' closed: C:¥frame05¥frame05_xls.xls'
2470  !-----
2471      END SUBROUTINE XLS_T2D_B2D_DEF ! 12APR2005
2472  !*****
2473
2474
2475
2476
2477

```

☐ frame05.exe の解析結果から、著者は免責される。  
☐ frame05.f90 の修正も読者自身の責任に依る。