Program plax

1 FE program plax

The Matlab program plax allows to model and analyze two-dimensional planar and axisymmetric structures, where the deformation can be large and the material nonlinear..

Model geometry, topology (connectivity), geometrical and material parameters, boundary conditions (prescribed displacements and point loads) and link relations (dependecies between degrees of freedom) must be available as input data. Also the history of the prescribed boundary conditions must be specified. When the analysis is finished, output date are stored in the data base and various other data arrays.

2 Matlab program plax

The program plax.m calls a collection of command and function files, which can and should be inspected for through understanding of the procedure. The program structure is however clearly shown in the listing.

```
%*****************************************************************
%  plax : 2-dimensional planar/axisym.; nonlinear
%=================================================================
plaxchkinp;                                          % plaxchkinp.m
fbiblcase;                                            % fbiblcase.m
plaxinizer;                                          % plaxinizer.m
[eida0,eidaB,eidaC,eismB,eismC,elip,neip] = ...      % plaxinidat.m
                                  plaxinidat(ne,elgr,elda,neip,GDt,mm);

if res==0, save([matf num2str(0)]); end;
crdB = crd0; crd = crd0;
%=================================================================
%  Calculate shape functions and their derivatives
%   for a 4-node element              plaxq4.m
%   for a 8-node element              plaxq8.m
%=================================================================
if     nenod==4, [ksi,psi,psidksi,ipwf,lokvg] = plaxq4(lok,ne,nndof,neip);
elseif nenod==8, [ksi,psi,psidksi,ipwf,lokvg] = plaxq8(lok,ne,nndof,neip);
end;
%=================================================================
%  Incremental calculation
%=================================================================
ic = 1; ti = 0; it = 0; slow = 1;

if res>0                                         % Restart analysis
  ic = res; load([matf num2str(ic-1)]);
```

```
  crd = crdB; eidaC = eidaB; it = 1;
end;

while ic<=nic
%-----------------------------------------------------------------------
fbibcutback;                                             % fbibcutback.m
ti = ti + GDt; it = 0; loadincr;
pe = peC./slow; fe = feC;
rs = fe - fi;
Dp = zeros(ndof,1); Ip = zeros(ndof,1);

GDt = GDt0/slow;
%=======================================================================
%  System matrix is assembled from element matrices
%=======================================================================
if (ic==1 | nl==1 | ic==res)
%-----------------------------------------------------------------------
sm = zeros(ndof);                     % structural stiffness matrix
elmalivi = 0;                         % counter of lin.viscoel. elements
gip = 0;                              % global integration point number

for e=1:ne
  ety  = elda(elgr(e),1);               mat  = elda(elgr(e),2);
  em   = zeros(nedof);                  ef   = zeros(nedof,1);
  ec0  = crd0(lok(e,3:nenod+2),:);      ecB  = crdB(lok(e,3:nenod+2),:);
  ec   = crd(lok(e,3:nenod+2),:);       Tpe  = Tp(lokvg(e,:));
  vole0  = 0; voleC = 0;
  if mat==8, elmalivi = elmalivi + 1; end;

  plaxelem; % -> ef, em                                     % plaxelem.m

  sm(lokvg(e,:),lokvg(e,:)) = sm(lokvg(e,:),lokvg(e,:)) + em;
end;

if ic==1, sm00=sm; end; sm0=sm;
%-----------------------------------------------------------------------
end;
%=======================================================================
%  Iterative calculation
%=======================================================================
nrm = 1000; it = 1; sm0 = sm;

while (nrm>ccr) & (it<=mit)
%-----------------------------------------------------------------------
%=======================================================================
%  Links and boundary conditions are taken into account
%  Unknown nodal point values are solved
```

```
%  Prescribed nodal values are inserted in the solution vector
%=======================================================================
%sm = sm00;                  % only used to test modified Newton-Raphson

if npl>0, rs = rs - sm(:,plc)*lif'; end;
[smp,rsp] = fbibpartit(it,sm,rs,ndof,pa,ppc,plc,prc,pe,lim);% fbibpartit.m

sol = smp\rsp; soll=sol; % dsol = smp\(smp*sol-rsp); soll = sol-dsol;

p = zeros(ndof,1); p(pu) = soll;
if it==1, p(ppc) = pe(ppc); end;
if npl>0, p(plc) = lim*p(prc) + lif'; end;

Dp  = p;  Ip = Ip + Dp;  Tp = Tp + Dp;
crd = crd0 + reshape(Tp,nndof,nnod)';
%=======================================================================
%  Calculate stresses and strains.
%  Make system matrix and internal force vector for next step.
%=======================================================================
sm=zeros(ndof); fi=zeros(ndof,1); elmalivi=0; gip=0;

for e=1:ne
  ety  = elda(elgr(e),1);              mat  = elda(elgr(e),2);
  em   = zeros(nedof);                 ef   = zeros(nedof,1);
  ec0  = crd0(lok(e,3:nenod+2),:);     ecB  = crdB(lok(e,3:nenod+2),:);
  ec   = crd(lok(e,3:nenod+2),:);      Tpe  = Tp(lokvg(e,:));
  vole0  = 0; voleC = 0;
  if mat==8, elmalivi = elmalivi + 1; end;

  plaxelem; % -> ef, em                                     % plaxelem.m

  fi(lokvg(e,:))            = fi(lokvg(e,:)) + ef;
  sm(lokvg(e,:),lokvg(e,:)) = sm(lokvg(e,:),lokvg(e,:)) + em;
end;
%=======================================================================
%  Calculate residual force and convergence norm
%=======================================================================
rs = fe - fi;
nrm = fbibcnvnrm(cnm,pu,ppc,prs,Dp,Ip,rs,fi);           % fbibcnvnrm.m

it = it + 1;                       % increment the iteration step counter

loka=lok;plaxipc;                                            % plaxipc.m

fbibwr2scr;                                               % fbibwr2scr.m
%-----------------------------------------------------------------------
end; %it
```

```
%======================================================================
%  Update and store values
%======================================================================
fbibcol2mat1;                                         % fbibcol2mat1.m
crdB = crd; feB = fe; eidaB = eidaC; eismB = eismC;
savefile = [matf num2str(ic)]; savedata;                  % savedata.m

ic = ic + 1;                           % increment the increment counter
save([matf '00'],'ic');                           % save date to 'matf'

%----------------------------------------------------------------------
end; %ic

%**********************************************************************
```