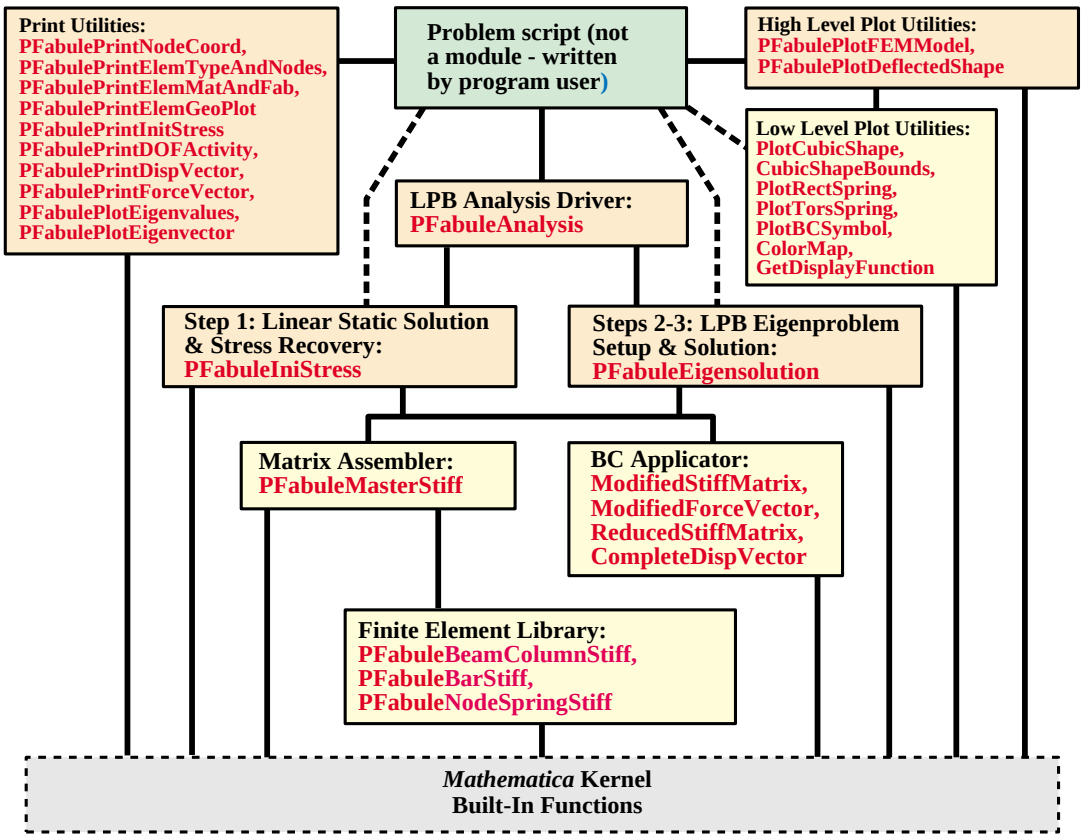
30

Linearized Prebuckling: Implementation

Configuration of PFabule Program

Plane Frame Analysis of Buckling Using Linearized Eigenproblem



Element Stiffness Library

```
PFabuleBarStiff[enodXY_,Em_,A0_,s0_,options_]:=Module[{X1,Y1,X2,Y2,
 X21,Y21,LL,L,B,BB,m=Length[options],numer=False,srule={},KeM,KeG},
  If [m>=1,numer=options[[1]]]; If [m>=2,srule=options[[2]]];
 {{X1,Y1},{X2,Y2}}=enodXY; {X21,Y21}={X2-X1,Y2-Y1};
  LL=X21^2+Y21^2; L=Sqrt[LL];
  If [!numer,{L,LL}=Simplify[{L,LL},srule]];
  B={-X21,-Y21,X21,Y21}; KeM=(Em*A0/L)*Outer[Times,B,B]/LL;
  KeG=(s0*A0/L)*{{1,0,-1,0},{0,1,0,-1},{-1,0,1,0},{0,-1,0,1}};
  If [numer,{KeM,KeG}=N[{KeM,KeG}],
           {KeM,KeG}=Simplify[{KeM,KeG},srule]];
  Return[{KeM,KeG}]];

PFabuleBeamColStiff[enodXY_,Em_,{A0_,Izz0_},s0_,options_]:=
   Module[{X1,X2,Y1,Y2,X21,Y21,XX,YY,XY,EA,EI,P=s0*A0,L,LL,LLL,
   LLLL,LLLLL,EALL,EILL,EIALL,EAXIY,EIXAY,
   m=Length[options],numer=False,srule={},KeM,KeG},
   If [m>=1,numer=options[[1]]]; If [m>=2,srule=options[[2]]];
  {{X1,Y1},{X2,Y2}}=enodXY; {X21,Y21}={X2-X1,Y2-Y1};
   EA=Em*A0; EI=Em*Izz0; LL=X21^2+Y21^2; L=Sqrt[LL];
   If [!numer,{L,LL}=Simplify[{L,LL},srule]];
   XX=X21^2; XY=X21*Y21; YY=Y21^2; LLL=LL*L; LLLL=LL*LL;
   EALL=EA*LL; EILL=EI*LL; EIALL=12*EI-EALL; LLLLL=LLL*LL;
   EAXIY=EALL*XX+12*EI*YY; EIXAY=12*EI*XX+EALL*YY;
   KeM={{EAXIY,-EIALL*XY,-6*EILL*Y21,-EAXIY,EIALL*XY,-6*EILL*Y21},
   {-EIALL*XY,EIXAY,6*EILL*X21,EIALL*XY,-EIXAY,6*EILL*X21},
   {-6*EILL*Y21,6*EILL*X21,4*EILL*LL,6*EILL*Y21,-6*EILL*X21,2*EILL*LL},
   {-EAXIY,EIALL*XY,6*EILL*Y21,EAXIY,-EIALL*XY,6*EILL*Y21},
   {EIALL*XY,-EIXAY,-6*EILL*X21,-EIALL*XY,EIXAY,-6*EILL*X21},
   {-6*EILL*Y21,6*EILL*X21,2*EILL*LL,6*EILL*Y21,-6*EILL*X21,
    4*EILL*LL}}/LLLLL;
   KeG={{36*YY,-36*XY,-3*LL*Y21,-36*YY,36*XY,-3*LL*Y21},
   {-36*XY,36*XX,3*LL*X21,36*XY,-36*XX,3*LL*X21},
   {-3*LL*Y21,3*LL*X21,4*LLLL,3*LL*Y21,-3*LL*X21,-LLLL},
   {-36*YY,36*XY,3*LL*Y21,36*YY,-36*XY,3*LL*Y21},
   {36*XY,-36*XX,-3*LL*X21,-36*XY,36*XX,-3*LL*X21},
   {-3*LL*Y21,3*LL*X21,-LLLL,3*LL*Y21,-3*LL*X21,4*LLLL}}*P/(30*LLL);
   Return[{KeM,KeG}]];

PFabuleNodeSpringStiff[type_,k_]:=Module[{KeM,KeG=Table[0,{3},{3}]},
   If [type=="SpringX", KeM=DiagonalMatrix[{k,0,0}]];
   If [type=="SpringY", KeM=DiagonalMatrix[{0,k,0}]];
   If [type=="SpringT", KeM=DiagonalMatrix[{0,0,k}]];
   Return[{KeM,KeG}]];
```

Master Stiffness Assembler (either KM or KG, as per argument)

```
PFabuleMasterStiff[nodXYZ_,eletyp_,elenod_,elemat_,elefab_,elestr_,
  options_,KMorKG_]:=Module[{numele=Length[eletyp],numnod=Length[nodXYZ],
  neldof,eftab,enodXY,ni,nj,i,j,ii,jj,type,type3,Em,fab,s0,numer,
  modname="PFabuleMasterStiff:",KeM,KeG,Ke,K},
  If [KMorKG!="M"&&KMorKG!="G", Print [modname," KMorKG not M or G"];
      Return[Null]]; K=Table[0,{3*numnod},{3*numnod}];
  For [e=1, e<=numele, e++, type=eletyp[[e]]; type3=StringTake[type,3];
      If [!MemberQ[{"Beam","Bar","SpringX","SpringY","SpringT"},type],
         Print [modname, " Illegal element type: ",type]; Return[Null]];
      If [type3=="Spr",    {ni}=elenod[[e]];
          eftab={3*ni-2,3*ni-1,3*ni}; fab=elefab[[e]];
         {KeM,KeG}=PFabuleNodeSpringStiff[type,fab[[1]]]];
      If [type=="Bar",
         {ni,nj}=elenod[[e]]; enodXY={nodXYZ[[ni]],nodXYZ[[nj]]};
          Em=elemat[[e]]; fab=elefab[[e]]; s0=elestr[[e]];
          eftab={3*ni-2,3*ni-1,3*nj-2,3*nj-1};
         {KeM,KeG}=PFabuleBarStiff[enodXY,Em,fab,s0,options]];
      If [type=="Beam",
         {ni,nj}=elenod[[e]]; enodXY={nodXYZ[[ni]],nodXYZ[[nj]]};
          Em=elemat[[e]]; fab=elefab[[e]]; s0=elestr[[e]];
          eftab={3*ni-2,3*ni-1,3*ni,3*nj-2,3*nj-1,3*nj};
         {KeM,KeG}=PFabuleBeamColStiff[enodXY,Em,fab,s0,options]];
      If [KMorKG=="M", Ke=KeM, Ke=KeG];  neldof=Length[Ke];
      For [i=1, i<=neldof, i++, ii=eftab[[i]];
          For [j=i, j<=neldof, j++, jj=eftab[[j]];
               K[[jj,ii]]=K[[ii,jj]]+=Ke[[i,j]]]];
      ];  If [!numer, K=Simplify[K]];
  Return[K]];
```

Boundary Condition Application

(different in stress analysis and buckling analysis)

```
ModifiedStiffMatrix[nodtag_,K_]:=Module[{numdof,tags,
  fixdof,i,j,k,d,Kmod=K}, tags=Flatten[nodtag];
  numdof=Length[tags]; fixdof=Flatten[Position[tags,_?(#>0 &)]];
  For [k=1,k<=Length[fixdof],k++, i=fixdof[[k]];
      For [j=1,j<=numdof,j++, Kmod[[i,j]]=Kmod[[j,i]]=0];
      Kmod[[i,i]]=1;
      ];  ClearAll[tags,vals,fixdof];
  Return[Kmod]];

ModifiedForceVector[nodtag_,nodval_,K_,f_]:=Module[{numdof,tags,vals,
  fixdof,i,j,k,d,fmod=f}, tags=Flatten[nodtag]; vals=Flatten[nodval];
  numdof=Length[tags]; fixdof=Flatten[Position[tags,_?(#>0 &)]];
  For [k=1,k<=Length[fixdof],k++, i=fixdof[[k]]; d=vals[[i]];
      fmod[[i]]=d; If [d==0, Continue[]];
      For [j=1,j<=numdof,j++,
           If [tags[[j]]==0,fmod[[j]]-=K[[i,j]]*d]];
      ]; ClearAll[tags,vals,fixdof];
  Return[fmod]];

ReducedStiffMatrix[nodtag_,K_]:=Module[{numdof,tags,i,ii,j,jj,
  nred,retdof,Kred}, tags=Flatten[nodtag]; numdof=Length[tags];
  retdof=Flatten[Position[tags,_?(#==0 &)]]; nred=Length[retdof];
  If [nred<=0, Return[Null]]; Kred=Table[0,{nred},{nred}];
  For [i=1,i<=nred,i++,  ii=retdof[[i]];
       For [j=1,j<=nred,j++, jj=retdof[[j]];
            Kred[[i,j]]=K[[ii,jj]] ]];
  ClearAll[tags,retdof]; Return[Kred]];

CompleteDispVector[nodtag_,ured_]:=Module[{numdof,tags,i,ii,
  nred,retdof,u}, tags=Flatten[nodtag]; numdof=Length[tags];
  retdof=Flatten[Position[tags,_?(#==0 &)]]; nred=Length[retdof];
  u=Table[0,{numdof}]; If [nred<=0, Return[u]];
  For [i=1,i<=nred,i++,  ii=retdof[[i]]; u[[ii]]=ured[[i]] ];
  ClearAll[tags,retdof]; Return[u]];
```

Initial Stress Analysis

(Only the Material Stiffness is Needed)

```
LPBPlaneFrameInitialStress[nodXYZ_,eletyp_,elenod_,elemat_,elefab_,
  nodtag0_,nodfor_,options_]:=Module[{numele=Length[eletyp],u,e,ni,nj,
  type,Em,X1,Y1,X2,Y2,X21,Y21,uX1,uY1,uX2,uY2,ebar,elestr,m=Length[options],
  numer=False,srule={},KM,Kmod,modname="TLPlaneFrameInitialStress:"},
  elestr=Table[0,{numele}];
  If [m>=1,numer=options[[1]]]; If [m>=2,srule=options[[2]]];
  KM=LPBPlaneFrameMasterStiff[nodXYZ,eletyp,elenod,elemat,
        elefab,elestr,options,"M"]; If [KM==Null, Return[Null]];
  Kmod=ModifiedStiffMatrix[nodtag0,KM];
  (*Print["KM=",KM//MatrixForm," Kmod=",Kmod//MatrixForm,
   " f=",Flatten[nodfor]//MatrixForm," u=",u//MatrixForm];*)
  u=LinearSolve[Kmod,Flatten[nodfor]]; elestr=Table[0,{numele}];
  For [e=1, e<=numele, e++,  type=eletyp[[e]];
      If [type!="Spring"&&type!="Bar"&&type!="Beam", Print [modname,
          " Illegal element type: ",type]; Return[Null]];
      If [type=="Spring", Continue[]];
      If [type=="Bar"||type=="Beam",
          {ni,nj}=elenod[[e]]; Em= elemat[[e]];
          {X1,Y1}=nodXYZ[[ni]]; {X2,Y2}=nodXYZ[[nj]];
          X21=X2-X1; Y21=Y2-Y1; LL=Simplify[X21^2+Y21^2];
          {uX1,uY1,uX2,uY2}={u[[3*ni-2]],u[[3*ni-1]],
                             u[[3*nj-2]],u[[3*nj-1]]};
          ebar=X21*(uX2-uX1)+Y21*(uY2-uY1)/LL;
          elestr[[e]]=Em*ebar];
      ]; If [!numer,elestr=Simplify[elestr,srule]];
  Return[elestr]];
```

Eigenvalue Analysis

```
PFabuleEigenMatrices[nodXYZ_,eletyp_,elenod_,elemat_,elefab_,
  elestr_,nodtag_,Tdof_,options_]:=Module[{KM,KG,numer=False,
  m=Length[options],srule={},TdofT,KMred,KGred},
  If [m>=1,numer=options[[1]]]; If [m>=2,srule=options[[2]]];
  KM=PFabuleMasterStiff[nodXYZ,eletyp,elenod,elemat,
        elefab,elestr,options,"M"];
  KG=PFabuleMasterStiff[nodXYZ,eletyp,elenod,elemat,
        elefab,elestr,options,"G"];
  If [KM==Null||KG==Null, Return[{Null,Null}]];
  KMred=ReducedStiffMatrix[nodtag,KM];
  KGred=ReducedStiffMatrix[nodtag,KG];
  If [Length[Tdof]>0, TdofT=Transpose[Tdof];
      KMred=Simplify[TdofT.KMred.Tdof];
      KGred=Simplify[TdofT.KGred.Tdof]];
  If [!numer, {KMred,KGred}=Simplify[{KMred,KGred},srule]];
  ClearAll[KM,KG]; Return[{KMred,KGred}]];

PFabuleEigenSolution[KMred_,KGred_,nodtag_,nodval_,Tdof_,invKM_,evonly_,
  options_]:=Module[{m=Length[options],numer=False,srule={},invKG=!invKM,
  nv=Length[KMred],Q,ev,evec={},λ,λv={},vred,V={}},
  If [m>=1,numer=options[[1]]]; If [m>=2,srule= options[[2]]];
  If [invKM, Q=-LinearSolve[KMred,KGred]];
  If [invKG, Q=-LinearSolve[KGred,KMred]];
  If [numer, Q=N[Q]]; If [!numer, Q=Simplify[Q,srule]];
  If [evonly, ev=Eigenvalues[Q],{ev,evec}=Eigensystem[Q]];
  If [!numer,{ev,evec}=Simplify[{ev,evec},srule]];
  If [numer, {ev,evec}=Chop[{ev,evec}]];
  For [i=1,i<=nv,i++, λ=ev[[i]];
       If [(invKM&&λ==0)||(invKG&&λ==Infinity), Continue[]];
       If [invKM, AppendTo[λv,1/λ], AppendTo[λv,λ]]];
  If [evonly, ClearAll[Q,ev,evec]; Return[{λv,{}}]];
  For [i=1,i<=nv,i++, λ=ev[[i]];
       If [(invKM&&λ==0)||(invKG&&λ==Infinity), Continue[]];
       vred=V[[i]]; If [Length[Tdof]>0, vred=Tdof.vred];
       AppendTo[V,CompleteDispVector[nodtag,nodval,vred]]];
  ClearAll[Q,ev,evec]; Return[{λv,V}]];
```

Analysis Driver


```
LPBPlaneFrameAnalysis[nodXYZ_,eletyp_,elenod_,elemat_,elefab_,
  nodtag0_,nodtag_,nodfor_,options_]:=Module[{elestr,λv,V},
  elestr=LPBPlaneFrameInitialStress[nodXYZ,eletyp,elenod,elemat,elefab,
      nodtag0,nodfor,options]; Print["elestr=",elestr];
  If [elestr==Null,Return[{Null,Null}]];
  {λv,V}=LPBPlaneFrameEigenSolution[nodXYZ,eletyp,elenod,elemat,
        elefab,elestr,nodtag,options];
  Return[{λv,V}]];
```

Print Utilities (1)

```
PFabulePrintNodeCoord[nodXYZ_,title_,digits_,form_]:= Module[
 {RV,ReV,numnod=Length[nodXYZ],n,Xn,Yn,d=6,f=6,label,
  pf=InputForm,tab},  tab=Table[" ",{numnod}];
  If [Length[digits]==2,{d,f}=digits];
  RV[expr_]:=VectorQ[expr,NumericQ[#]&&(Head[#]===Real)&];
  ReV=RV[Flatten[nodXYZ]];
  If [ReV,
     For [n=1,n<=numnod,n++, {Xn,Yn}=nodXYZ[[n]];
          tab[[n]]={ToString[n],PaddedForm[Xn,{d,f}],
                                PaddedForm[Yn,{d,f}]}]];
  If [!ReV, If [MemberQ[{InputForm,StandardForm,TextForm,
                TraditionalForm},form], pf=form];
     For [n=1,n<=numnod,n++, {Xn,Yn}=nodXYZ[[n]];
          tab[[n]]={ToString[n],pf[Xn],pf[Yn]}]];
  If [StringLength[title]>0, Print[title]];
  label={"node","X-coor","Y-coor"};
  Print[TableForm[tab, TableAlignments->{Right},
        TableDirections->{Column,Row},TableSpacing->{0,2},
        TableHeadings ->{None,label}]];
  ClearAll[tab]];

PFabulePrintElemTypeAndNodes[eletyp_,elenod_,title_,digits_,form_]:=
  Module[{ReV,RV,e,numele=Length[eletyp],type,enl,label,d=4,f=4,
          pf=InputForm,tab}, tab=Table[" ",{numele}];
  If [Length[digits]==2,{d,f}=digits];
  For [e=1,e<=numele,e++, enl=elenod[[e]]; type=eletyp[[e]];
       tab[[e]]={ToString[e],type,ToString[enl]}];
  If [StringLength[title]>0, Print[title]];
  label={"elem","type","nodelist"};
  Print[TableForm[tab, TableAlignments->{Right},
        TableDirections->{Column,Row},TableSpacing->{0,2},
        TableHeadings ->{None,label}]];
  ClearAll[tab]];

PFabulePrintElemMatAndFab[elemat_,elefab_,title_,digits_,form_]:=
  Module[{ReV,RV,e,numele=Length[elemat],mat,fab,
         label,d=4,f=2,pf=InputForm,tab},
   tab=Table[" ",{numele}]; If [Length[digits]==2,{d,f}=digits];
   RV[expr_]:=VectorQ[expr,NumericQ[#]&&(Head[#]===Real)&];
   ReV=RV[Flatten[{elemat,elefab}]];
   If [ReV,
       For [e=1,e<=numele,e++,  mat=elemat[[e]]; fab=elefab[[e]];
            tab[[e]]={ToString[e],ToString[PaddedForm[mat]],
                                  ToString[PaddedForm[fab]]}]];
   If [!ReV, If [MemberQ[{InputForm,StandardForm,TextForm,
                TraditionalForm},form], pf=form];
       For  [e=1,e<=numele,e++,  mat=elemat[[e]]; fab=elefab[[e]];
             tab[[e]]={ToString[e],pf[mat],pf[fab]}]];
   If [StringLength[title]>0, Print[title]];
   label={"elem","material","fabrication"};
   Print[TableForm[tab, TableAlignments->{Right},
       TableDirections->{Column,Row},TableSpacing->{0,2},
       TableHeadings ->{None,label}]];
   ClearAll[tab]];
```

Print Utilities (2)

```
PFabulePrintElemIniStress[eletyp_,elestr_,title_,digits_,form_]:=
  Module[{ReV,RV,e,numele=Length[eletyp],type,sig,label,
  d=4,f=4,pf=InputForm,tab}, tab=Table[" ",{numele}];
  If [Length[digits]==2,{d,f}=digits];
  RV[expr_]:=VectorQ[expr,NumericQ[#]&&(Head[#]===Real)&];
  ReV=RV[elestr];
  If [ReV,
     For [e=1,e<=numele,e++, type=eletyp[[e]]; sig=elestr[[e]];
          tab[[e]]={ToString[e],type,
                    ToString[PaddedForm[sig,{d,f}]]}];
  If [!ReV, If [MemberQ[{InputForm,StandardForm,TextForm,
               TraditionalForm},form], pf=form];
     For [e=1,e<=numele,e++, type=eletyp[[e]]; sig=elestr[[e]];
          tab[[e]]={ToString[e],type,pf[sig]}];
  If [StringLength[title]>0, Print[title]];
  label={"elem","type","axial stress"};
  Print[TableForm[tab, TableAlignments->{Right},
        TableDirections->{Column,Row},TableSpacing->{0,2},
        TableHeadings ->{None,label}]];
  ClearAll[tab]];

PFabulePrintDOFActivity[nodtag_,nodval_,title_,digits_,form_]:= Module[
  {numnod=Length[nodtag],n,t1,t2,t3,v1,v2,v3,d=6,f=4,
  pf=InputForm,tab},
  tab=Table[" ",{numnod}]; If [Length[digits]==2,{d,f}=digits];
  RV[expr_]:=VectorQ[expr,NumericQ[#]&&(Head[#]===Real)&];
  ReV=RV[Flatten[{nodtag,nodval}]];
  If [ReV,
      For [n=1,n<=numnod,n++,
           {t1,t2,t3}=nodtag[[n]]; {v1,v2,v3}=nodval[[n]];
           tab[[n]]={ToString[n],
           PaddedForm[t1,d],PaddedForm[t2,d],
           PaddedForm[t3,d],PaddedForm[v1,{d,f}],
           PaddedForm[v2,{d,f}],PaddedForm[v3,{d,f}] }];
  If [!ReV, If [MemberQ[{InputForm,StandardForm,TextForm,
               TraditionalForm},form], pf=form]; Print["pf=",pf];
      For [n=1,n<=numnod,n++,
           {t1,t2,t3}=nodtag[[n]]; {v1,v2,v3}=nodval[[n]];
           tab[[n]]={ToString[n],pf[t1],pf[t2],pf[t3],
                                 pf[v1],pf[v2],pf[v3] }];
   If [StringLength[title]>0, Print[title]];
   Print[TableForm[tab,TableAlignments->{Right},
         TableDirections->{Column,Row},TableSpacing->{0,1},
         TableHeadings->{None,{"node", "X-tag", "Y-tag","θ-tag",
                        "X-value", "Y-value", "θ-value"}} ]];
   ClearAll[tab]];

PFabulePrintDispVector[noddis_,title_,digits_,form_]:= Module[
 {ReV,RV,numnod,uv,n,uXn,uYn,θn,d=6,f=6,label,
  pf=InputForm,tab}, uv=Partition[noddis,3]; numnod=Length[uv];
  tab=Table[" ",{numnod}]; If [Length[digits]==2,{d,f}=digits];
  RV[expr_]:=VectorQ[expr,NumericQ[#]&&(Head[#]===Real)&];
  ReV=RV[noddis];
  If [ReV,
     For [n=1,n<=numnod,n++, {uXn,uYn,θn}=uv[[n]];
          tab[[n]]={ToString[n],PaddedForm[uXn,{d,f}],
          PaddedForm[uYn,{d,f}],PaddedForm[θn,{d,f}]}];
  If [!ReV, If [MemberQ[{InputForm,StandardForm,TextForm,
               TraditionalForm},form], pf=form];
     For [n=1,n<=numnod,n++, {uXn,uYn,θn}=uv[[n]];
          tab[[n]]={ToString[n],pf[uXn],pf[uYn],pf[θn]}];
  If [StringLength[title]>0, Print[title]];
  label={"node","X-dis","Y-dis","θ-rot"};
  Print[TableForm[tab, TableAlignments->{Right},
        TableDirections->{Column,Row},TableSpacing->{0,2},
        TableHeadings ->{None,label}]];
  ClearAll[u,tab]];
```

Print Utilities (3)

```
PFabulePrintForceVector[nodfor_,title_,digits_,form_]:= Module[
 {ReV,RV,numnod,fv,n,fXn,fYn,mθn,d=6,f=6,label,
  pf=InputForm,tab}, fv=Partition[nodfor,3]; numnod=Length[fv];
  tab=Table[" ",{numnod}]; If [Length[digits]==2,{d,f}=digits];
  RV[expr_]:=VectorQ[expr,NumericQ[#]&&(Head[#]===Real)&];
  ReV=RV[nodfor];
  If [ReV,
     For [n=1,n<=numnod,n++, {fXn,fYn,mθn}=u[[n]];
          tab[[n]]={ToString[n],PaddedForm[fXn,{d,f}],
          PaddedForm[fYn,{d,f}],PaddedForm[mθn,{d,f}]}]];
  If [!ReV, If [MemberQ[{InputForm,StandardForm,TextForm,
               TraditionalForm},form], pf=form];
     For [n=1,n<=numnod,n++, {fXn,fYn,mθn}=fv[[n]];
          tab[[n]]={ToString[n],pf[fXn],pf[fYn],pf[mθn]}]];
  If [StringLength[title]>0, Print[title]];
  label={"node","X-for","Y-for","θ-mom"};
  Print[TableForm[tab, TableAlignments->{Right},
        TableDirections->{Column,Row},TableSpacing->{0,2},
        TableHeadings ->{None,label}]];
  ClearAll[fv,tab]];

PFabulePrintEigenVector[V_,ivec_,digits_,form_]:= Module[{i,evec},
  If [ivec<=0||ivec>Length[V], Print["Illegal ivec"]; Return[]];
  i=ToString[ivec]; evec=V[[ivec]];
  PFabulePrintDispVector[evec,"Eigenvector #"<>i,digits,form]];

PFabulePrintEigenValues[λv_,{imin_,imax_},title_,digits_,form_]:=
  Module[{ReV,RV,ibeg,iend,nev,i,j=1,λi,d=6,f=6,label,pf=InputForm,
  tab}, ibeg=Max[imin,1]; iend=Min[imax,Length[λv]]; nev=imax-imin+1;
  If [nev<=0, Print[" Empty prt eigval range"]; Return[]];
  tab=Table[" ",{nev}]; If [Length[digits]==2,{d,f}=digits];
  RV[expr_]:=VectorQ[expr,NumericQ[#]&&(Head[#]===Real)&]; ReV=RV[λv];
  If [ReV,
     For [i=ibeg,i<=iend,i++, λi=λv[[i]];
          tab[[j++]]={ToString[i],PaddedForm[λi,{d,f}]}]];
  If [!ReV,  If [MemberQ[{InputForm,StandardForm,TextForm,
                TraditionalForm},form], pf=form];
     For [i=ibeg,i<=iend,i++, λi=λv[[i]];
          tab[[j++]]={ToString[i],pf[λi]}]];
  If [StringLength[title]>0, Print[title]];
  label={"node","eigenvalue"};
  Print[TableForm[tab, TableAlignments->{Right},
        TableDirections->{Column,Row},TableSpacing->{0,2},
        TableHeadings ->{None,label}]];
  ClearAll[tab]];
```

Plot Utilities (1)


```
DisplayChannel[]:=Module[{},
     If [$VersionNumber>=6.0,Return[Print]];
     Return[$DisplayFunction]];

ColorMap[color_]:=Module[{colspec=color},
  If [color=="Black",     colspec=RGBColor[0,0,0]];
  If [color=="Blue",      colspec=RGBColor[0,0,1]];
  If [color=="Green",     colspec=RGBColor[0,1,0]];
  If [color=="Red",       colspec=RGBColor[1,0,0]];
  If [color=="White",     colspec=RGBColor[1,1,1]];
  If [color=="GrayBG",    colspec=GrayLevel[.8]];
  If [color=="BlueBG",    colspec=Hue[.50,.6,1]];
  If [color=="GreenBG",   colspec=Hue[.25,.6,1]];
  If [color=="YellowBG",  colspec=Hue[.18,.6,1]];
  Return[colspec]];
```

Plot Utilities (2)


```
PlotCubicShape[xye_,ue_,amp_,th_,color_,dash_,subs_]:=Module[
  {x1,y1,x2,y2,x21,y21,L,c,s,x0,y0,ux1,uy1,θ1,ux2,uy2,θ2,k,ξ,
   uxb1,uyb1,uxb2,uyb2,uxb,uyb,darg={},xyP,p},
  {{x1,y1},{x2,y2}}=xye; x21=x2-x1; y21=y2-y1;
  {ux1,uy1,θ1,ux2,uy2,θ2}=amp*ue; L=N[Sqrt[x21^2+y21^2]];
   If [L<=0||th<=0, Return[{}]]; If [dash, darg={4,3}*th];
   p={Graphics[AbsoluteThickness[th]],Graphics[color],
      Graphics[AbsoluteDashing[darg]]}; c=x21/L; s=y21/L;
   uxb1= c*ux1+s*uy1; uxb2= c*ux2+s*uy2;
   uyb1=-s*ux1+c*uy1; uyb2=-s*ux2+c*uy2;
   xyP=Table[{0,0},{subs+1}]; xyP[[1]]={x1+ux1,y1+uy1};
   For [k=1, k<=subs, k++, ξ=N[(2*k-subs)/subs];
       x0= 0.5*(x1+x2+x21*ξ); y0=0.5*(y1+y2+y21*ξ);
       uxb=0.5*(uxb1+uxb2+(uxb2-uxb1)*ξ);
       uyb=0.125*(4*(uyb1+uyb2)+2*(uyb1-uyb2)*(ξ^2-3)*ξ+
           L*(ξ^2-1)*(θ2-θ1+(θ1+θ2)*ξ));
       xyP[[k+1]]={x0+uxb*c-uyb*s,y0+uxb*s+uyb*c}];
   AppendTo[p,Graphics[Line[xyP]]]; ClearAll[xyP];
   Return[p]];

CubicShapeMBF[xye_,ue_,amp_,subs_]:=Module[
  {x1,y1,x2,y2,x21,y21,L,c,s,x0,y0,ux1,uy1,θ1,ux2,uy2,θ2,k,ξ,
   uxb1,uyb1,uxb2,uyb2,uxb,uyb,xnew,ynew,xmin,xmax,ymin,ymax},
  {{x1,y1},{x2,y2}}=xye; x21=x2-x1; y21=y2-y1;
  {ux1,uy1,θ1,ux2,uy2,θ2}=amp*ue; L=N[Sqrt[x21^2+y21^2]];
   xmin=xmax=x1+ux1; ymin=ymax=y1+uy1;
   If [L<=0, Return[{{xmin,xmax,ymin,ymax}}]]; c=x21/L; s=y21/L;
   uxb1= c*ux1+s*uy1; uxb2= c*ux2+s*uy2;
   uyb1=-s*ux1+c*uy1; uyb2=-s*ux2+c*uy2;
   For [k=1, k<=subs, k++, ξ=N[(2*k-subs)/subs];
       x0= 0.5*(x1+x2+x21*ξ); y0=0.5*(y1+y2+y21*ξ);
       uxb=0.5*(uxb1+uxb2+(uxb2-uxb1)*ξ);
       uyb=0.125*(4*(uyb1+uyb2)+2*(uyb1-uyb2)*(ξ^2-3)*ξ+
           L*(ξ^2-1)*(θ2-θ1+(θ1+θ2)*ξ));
       xnew=x0+uxb*c-uyb*s; ynew=y0+uxb*s+uyb*c;
       xmin=Min[xmin,xnew]; ymin=Min[ymin,ynew];
       xmax=Max[xmax,xnew]; ymax=Max[ymax,ynew];
       ]; Return[{xmin,xmax,ymin,ymax}]];
```

Plot Utilities (4)

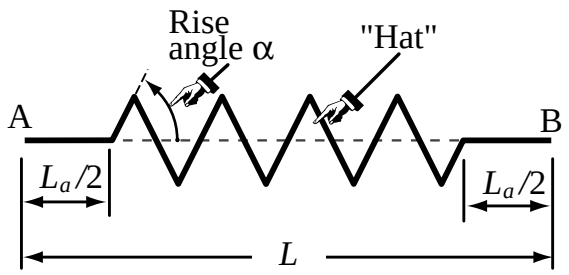
```
PlotRectSpring[xye_,rise_,m_,th_,color_,hats_]:=Module[
 {x1,y1,x2,y2,x0,y0,k,n,r,q,subs,lines,L,H,ξη,xL,yL,xyG,p},
   {{x1,y1},{x2,y2}}=xye; x21=x2-x1; y21=y2-y1; t=Min[th,1];
   L=Sqrt[x21^2+y21^2]; {x0,y0}={x1+x2,y1+y2}/2;
   If [L==0||th<=0||hats<=0||m<=3||m>12, Return[{}]];
   r={4,0,3,0,8/3,0,0,0,12/5}[[m-3]]; If [r==0, Return[{}]];
   subs=r*hats; n=subs/m; If [!IntegerQ[n], Return[{}]];
   lines=hats+3; H=rise*L/subs; q=Mod[hats,2];
   ξη=Table[{(2*k-2-subs)/subs,0},{k,subs+1}];
   For [k=n+2,k<=subs-n-2+2*q,k=k+4, ξη[[k,2]]= 1];
   For [k=n+4,k<=subs-n-2*q  ,k=k+4, ξη[[k,2]]=-1];
   For [k=2,k<=n,k++, ξη[[k,1]]= ξη[[k,2]]=Null];
   For [k=subs-n+2,k<=subs,k++, ξη[[k,1]]=ξη[[k,2]]=Null];
   For [k=n+3,k<=subs-n-1,k++, If [ξη[[k,2]]==0,
        ξη[[k,1]]=ξη[[k,2]]=Null]];
   ξη=Transpose[DeleteCases[Transpose[ξη],Null,2]];
   p={Graphics[AbsoluteThickness[th]],Graphics[ColorMap[color]]};
   xyG=Table[{0,0},{lines+1}]; c=x21/L; s=y21/L;
   For [k=1,k<=lines+1,k++, xL=L*ξη[[k,1]]/2; yL=H*ξη[[k,2]];
        xyG[[k]]={x0+c*xL-s*yL,y0+s*xL+c*yL}];
   AppendTo[p,Graphics[Line[xyG]]]; ClearAll[ξη,xyG];
   Return[p]];

PlotCircSpring[xye_,R_,rise_,m_,th_,color_,hats_]:=Module[{x1,y1,
  x2,y2,x0,y0,arg,α,θ,k,n,r,q,subs,lines,L,H,ξη,xL,yL,xyG,p},
   If [R==∞,Return[PlotRectSpring[xye,rise,m,th,color,hats]]];
   {{x1,y1},{x2,y2}}=xye; x21=x2-x1; y21=y2-y1;
   L=Sqrt[x21^2+y21^2]; {x0,y0}={x1+x2,y1+y2}/2; arg=L/(2*R);
   If [L==0||th<=0||hats<=0||m<=3||m>12||Abs[arg]>1, Return[{}]];
   α=ArcSin[N[L/(2*R)]]; c=x21/L; s=y21/L;
   r={4,0,3,0,8/3,0,0,0,12/5}[[m-3]]; If [r==0, Return[{}]];
   subs=r*hats; n=subs/m; If [!IntegerQ[n], Return[{}]];
   lines=hats+2*n+1; H=rise*L/subs; q=Mod[hats,2];
   ξη=Table[{(2*k-2-subs)/subs,0},{k,subs+1}];
   For [k=n+2,k<=subs-n-2+2*q,k=k+4, ξη[[k,2]]= 1];
   For [k=n+4,k<=subs-n-2*q  ,k=k+4, ξη[[k,2]]=-1];
   For [k=n+3,k<=subs-n-1,k++, If [ξη[[k,2]]==0,
        ξη[[k,1]]=ξη[[k,2]]=Null]];
   ξη=Transpose[DeleteCases[Transpose[ξη],Null,2]];
   p={Graphics[AbsoluteThickness[th]],Graphics[ColorMap[color]]};
   xyG=Table[{0,0},{lines+1}];
   For [k=1,k<=lines+1,k++, θ=α*ξη[[k,1]]; h=H*ξη[[k,2]];
        xL=(h-R)*Sin[θ]; yL=R*Cos[α]+(h-R)*Cos[θ];
        xyG[[k]]={x0+c*xL-s*yL,y0+s*xL+c*yL}];
   AppendTo[p,Graphics[Line[xyG]]]; ClearAll[ξη,xyG];
   Return[p]];
```

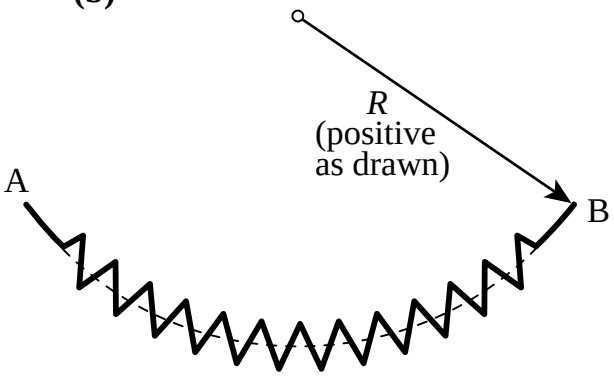
Spring Plots

(Coding analysis took 2 days, coding these plots 2 weeks)

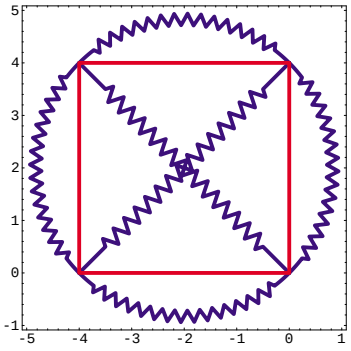
(a)



(b)



(c)



Plot Full Model
(Still Unfinished)

```
PFabulePlotFEMModel[nodXYZ_,eletyp_,elenod_,elegeo_,tcinfo_,ainfo_,
   einfo_,ninfo_,frame_,backgr_,imgsiz_,title_]:= Module[{
   numele=Length[elenod],numnod=Length[nodXYZ],ltc=Length[tcinfo],
   lai=Length[ainfo],lei=Length[einfo],lni=Length[ninfo],i,k,e,enl,
   n,ni,nj,x,y,asp=Automatic,xmin,xmax,ymin,ymax,dx,dy,xdim=0,ydim=0,
   dmin,tbeam=4,cbeam="Red",tbar=2,cbar="Blue",tspr=1,cspr="Black",
   ex,ey,cnx=2.,cny=1.5,xy0,xye,xyn,elabs=nlabs=False,frade,fradn,xylab,
   sink,rade=0,radn=0,elab,nlab,backg,dfun,arat,cmargin=0.08,mx,my,
   plow,phigh,pbeam=pbar=pspr=pnod=pnlab=pelab=pecir=pedisk=pbound={},
   Glin,Gtbeam,Gcbeam,Gtbar,Gcbar,Gtspr,Gcspr,Gtcir,Gwhite,Gblack},
   x=nodXYZ[[All,1]]; y=nodXYZ[[All,2]]; cnx=cny=0; cey=0.002;
  {xmin,xmax,ymin,ymax}=N[{Min[x],Max[x],Min[y],Max[y]}];
   If [ltc>=1,{tbeam,cbeam}=tcinfo[[1]]];
   If [ltc>=2,{tbar,cbar}=  tcinfo[[2]]];
   If [ltc>=3,{tspr,cspr}=  tcinfo[[3]]];
   If [lei==1, elabs=True; {frade}=einfo];
   If [lni==1, nlabs=True; {fradn}=ninfo];
   If [lni==3, nlabs=True; {fradn,cnx,cny}=ninfo];
   If [lai>=1, asp= ainfo[[1]]]; If [lai>=2, xdim=ainfo[[2]]];
   If [lai>=3, ydim=ainfo[[3]]]; If [lai>=4, cmargin=ainfo[[4]]];
   {dx,dy}={xmax-xmin,ymax-ymin};
   If [dx<xdim, xmin-=(xdim-dx)/2; xmax+=(xdim-dx)/2];
   If [dy<ydim, ymin-=(ydim-dy)/2; ymax+=(ydim-dy)/2];
   {dx,dy}={xmax-xmin,ymax-ymin}; dmin=Min[dx,dy];
   rade=frade*dmin; radn=fradn*dmin; sink={0,0.1*rade};
   For [e=1,e<=numele,e++, etype=eletyp[[e]]; enl=elenod[[e]];
        If [StringTake[type,3]=="Spr", Continue[]]; {ni,nj}=enl;
        xye={nodXYZ[[ni]],nodXYZ[[nj]]}; Glin=Graphics[Line[xye]];
        If [etype=="Beam", AppendTo[pbeam, Glin]];
        If [etype=="Bar",  AppendTo[pbar,  Glin]];
        If [elabs, xy0=(xye[[1]]+xye[[2]])/2; xylab=xy0-sink;
            AppendTo[pedisk,Graphics[Disk[xy0,rade]]];
            AppendTo[pecir,Graphics[Circle[xy0,rade]]];
            elab=StyleForm[e,FontFamily->"Times",FontSize->11];
            AppendTo[pelab,Graphics[Text[elab,xylab]]]]
       ]; {ex,ey}={cnx*radn,cny*radn}; {mx,my}=cmargin*{dx,dy};
   For [n=1,n<=numnod,n++, xyn=nodXYZ[[n]]; xylab=xyn+{ex,ey};
        If [nlabs,nlab=StyleForm[n,FontFamily->"Times",
                    FontSize->11, FontWeight->Bold];
            AppendTo[pnlab,Graphics[Text[nlab,xylab]]]];
         AppendTo[pnod, Graphics[Disk[xyn,radn]]]
       ];  plow={xmin-mx,ymin-my}; phigh={xmax+mx,ymax+my};
   pbound={Graphics[Point[plow]],Graphics[Point[phigh]]};
   If [asp>0, arat=asp, arat=dy/dx]; If [asp<0, arat=Automatic];
   dfun=DisplayChannel[]; backg=ColorMap[backgr];
   Gtbeam=Graphics[AbsoluteThickness[tbeam]];
   Gtbar =Graphics[AbsoluteThickness[tbar]];
   Gtspr =Graphics[AbsoluteThickness[tspr]];
   Gcbeam=Graphics[ColorMap[cbeam]]; Gcbar=Graphics[ColorMap[cbar]];
   Gcspr= Graphics[ColorMap[cspr]];  Gtcir=Graphics[AbsoluteThickness[1.25]];
   Gwhite=Graphics[RGBColor[1,1,1]]; Gblack=Graphics[RGBColor[0,0,0]];
   Show[Gtbeam,Gcbeam,pbeam,Gtbar,Gcbar,pbar,Gcspr,Gtspr,pspr,
        Gblack,pnod,Gtcir,Gwhite,pedisk,Gblack,pecir,
        pelab,pnlab,Gwhite,pbound, Background->backg,
        Frame->frame,PlotLabel->title,ImageSize->imgsiz,
        PlotRange->{{xmin-mx,xmax+mx},{ymin-my,ymax+my}},
        Axes->False,AspectRatio->arat,DisplayFunction->dfun];
    ClearAll[x,y,pnod,pnlab,pesid,pelab,pecir,pedisk,pbound];
];
```

Plot Deformed Shape
(Still Unfinished)

```
PFabulePlotDeformedShape[nodXYZ_,eletyp_,elenod_,elegeo_,u_,
 amp_,subs_,tcinfo_,ainfo_,frame_,backgr_,imgsiz_,title_]:=Module[{
 numele=Length[eletyp],numnod=Length[nodXYZ],na=Length[ainfo],
 e,etype,enl,i,ni,nj,ii,jj,ue,uez,xye,x,y,asp=Automatic,xdim=0,ydim=0,
 xmin,xmax,ymin,ymax,sframe,tbeam,cbeam,tbar,cbar,tspr,cspr,tref,cref,
 x1,x2,y1,y2,n,t,col,dash,arat=Automatic,dfun,pbound,pdef,pref},
 uez=Table[0,{6}]; x=nodXYZ[[All,1]]; y=nodXYZ[[All,2]];
 {xmin,xmax,ymin,ymax}=N[{Min[x],Max[x],Min[y],Max[y]}]; pdef=pref={};
{{tbeam,cbeam},{tbar,cbar},{tspr,cspr},{tref,cref}}=tcinfo;
 For [e=1,e<=numele,e++, etype=eletyp[[e]]; enl=elenod[[e]];
     If [etype=="Spring", Continue[]]; {ni,nj}=enl;
     xye={nodXYZ[[ni]],nodXYZ[[nj]]}; ii=3*ni; jj=3*nj;
     ue={u[[ii-2]],u[[ii-1]],u[[ii]],u[[jj-2]],u[[jj-1]],u[[jj]]};
     If [etype=="Beam", n=subs; t=tbeam; col=ColorMap[cbeam]];
     If [etype=="Bar",  n=1;    t=tbar;  col=ColorMap[cbar]];
     AppendTo[pdef,PlotCubicShape[xye,ue,amp,t,col,False,n,{}]];
     If [tref>0, col=ColorMap[cref];
         AppendTo[pref,PlotCubicShape[xye,ue,0,tref,col,True,1,{}]]];
     {x1,x2,y1,y2}=CubicShapeMBF[xye,ue,amp,subs];
     xmin=Min[xmin,x1]; ymin=Min[ymin,y1];
     xmax=Max[xmax,x2]; ymax=Max[ymax,y2];
     ];
 If [na>=1, asp= ainfo[[1]]]; If [na>=2, xdim=ainfo[[2]]];
 If [na>=3, ydim=ainfo[[3]]]; {dx,dy}={xmax-xmin,ymax-ymin};
 If [dx<xdim, xmin=xmin-(xdim-dx)/2; xmax=xmax+(xdim-dx)/2];
 If [dy<ydim, ymin=ymin-(ydim-dy)/2; ymax=ymax+(ydim-dy)/2];
 {dx,dy}={xmax-xmin,ymax-ymin};  If [asp==0, arat=dy/dx];
 If [asp>0, arat=asp]; dfun=DisplayChannel[]; backg=ColorMap[backgr];
 pbound={Graphics[RGBColor[1,1,1]],Graphics[AbsolutePointSize[1]],
         Graphics[Point[{xmin,ymin}]],Graphics[Point[{xmax,ymax}]]};
 Show[pref,pdef,pbound, Background->backg, AspectRatio->arat,
      ImageSize->imgsiz,Frame->frame,
      PlotLabel->title,DisplayFunction->dfun];
 ClearAll[x,y,pdef,pref,pbound];
 ];
```