```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Sep 26 10:59:53 2025

@author: frocha
"""

from oct2py import octave
import numpy as np
import scipy.sparse as sp
import scipy.sparse.linalg as spla
from contactFEA_python import *

# octave.addpath(octave.genpath("/home/felipe/UPEC/Bichon/codes/ContactFEA/"))  # doctest: +SKIP
octave.addpath(octave.genpath("/home/felipe/sources/pyola_contact2/src/"))  # doctest: +SKIP

# octave.ContactFEA_refac()

FEMod = octave.ModelInformation_Beam()
Dt=0.01; MinDt=1.0E-7; IterMax=16; GivenIter=8; MaxDt=0.1; Time = 0; # N-R parameters
TimeMax = 1.0

# Material
E=FEMod.Prop[0,0]; nu=FEMod.Prop[0,1];
Dtan= octave.getIsotropicCelas(E,nu);

contactPairs=octave.InitializeContactPairs(FEMod);

NodeNum, Dim = np.shape(FEMod.Nodes);
AllDOF = Dim*NodeNum;
Disp=np.zeros((AllDOF,1));
# Disp=np.zeros(AllDOF);

IterOld=GivenIter+1; NRConvergeNum=0; Istep = -1; Flag10 = 1;

op = "python"
# count = 0
# while count<1:  # Incremental loop
#     count += 1
```

```
while Flag10 == 1:  # Incremental loop
    Flag10 = 0
    Flag11 = 1
    Flag20 = 1
    ReductionNumber = 0

    DispSave = Disp.copy()
    tempContactPairs = contactPairs.copy()
    Time0 = Time

    Istep += 1
    Time += Dt

    while Flag11 == 1:  # Reduction loop
        NRConvergeNum += 1
        Flag11 = 0

        # Check whether the calculation is completed
        if (Time - TimeMax) > 1e-10:
            if (1 + Dt - Time) > 1e-10:
                Dt = 1 + Dt - Time
                Time = 1
            else:
                break

        Factor = Time
        SDisp = Dt * FEMod.Cons[:, 2]   # MATLAB 1-based -> Python 0-based
        Iter = 0
        PreDisp = Disp.copy()

        while Flag20 == 1:  # Newton-Raphson loop
            Flag20 = 0
            Iter += 1

            GKF = sp.lil_matrix((AllDOF, AllDOF))  # sparse stiffness
            Residual = np.zeros((AllDOF,1))
            ExtFVect = np.zeros((AllDOF,1))
            # Residual = np.zeros(AllDOF)
            # ExtFVect = np.zeros(AllDOF)
            NCon = FEMod.Cons.shape[0]
```

```
# Internal force and tangent stiffness

if(op == "python"):
    Residual, GKF = GetStiffnessAndForce(FEMod.Nodes, FEMod.Eles.astype('int'), Disp, Residual.flatten(), GKF, Dtan)
    Residual = Residual.reshape((len(Residual),1))
else:
    Residual, GKF = octave.GetStiffnessAndForce(FEMod, Disp, Residual, GKF, Dtan, nout = 2)

# print(np.allclose(GKF, GKF2))


# Contact state
contactPairs, GKF, Residual = octave.DetermineContactState(
    FEMod, contactPairs, Dt, PreDisp, GKF, Residual, Disp, nout = 3
)

# External load boundary
if FEMod.ExtF.shape[0] > 0:
    LOC = Dim * (FEMod.ExtF[:, 0].astype(int) - 1) + FEMod.ExtF[:, 1].astype(int) - 1  # convert to 0-based
    ExtFVect[LOC,0] += Factor * FEMod.ExtF[:, 2]
Residual += ExtFVect

# Displacement boundary conditions
if NCon != 0:
    FixDOF = Dim * (FEMod.Cons[:, 0].astype(int) - 1) + FEMod.Cons[:, 1].astype(int) - 1
    GKF[FixDOF, :] = 0
    for i, dof in enumerate(FixDOF):
        GKF[dof, dof] = 1.0
    Residual[FixDOF] = 0.0
    if Iter == 1:
        Residual[FixDOF,0] = SDisp



if Iter > 1:
    FixDOF = Dim * (FEMod.Cons[:, 0] - 1) + FEMod.Cons[:, 1] - 1
    FreeDOF = np.setdiff1d(np.arange(AllDOF), FixDOF)

    Resid = np.max(np.abs(Residual[FreeDOF]))

    if Iter > 2:
```

```
                    print(f"{Iter:27d} {Resid:14.5e}")
                else:
                    print("\n \t Time  Time step   Iter \t  Residual")
                    print(f"{Time:10.5f} {Dt:10.3e} {Iter:5d} {Resid:14.5e}")

                if Resid < 1e-7:  # Convergence
                    octave.updateContact(contactPairs)
                    if NRConvergeNum > 1 and Iter < GivenIter and IterOld < GivenIter:
                        Enlarge = 1.5
                        Dt = min(Enlarge * Dt, MaxDt)  # Increase step
                    IterOld = Iter
                    Flag10 = 1
                    break

                if Iter + 1 > IterMax:  # Too many NR iterations
                    Reduce = 0.25
                    Dt = Reduce * Dt
                    Time = Time0 + Dt
                    if Dt < MinDt:
                        raise RuntimeError("Incremental step too small")
                    Disp = DispSave.copy()
                    contactPairs = tempContactPairs.copy()
                    print(f"Not converged or reached MaxIteration. Reducing load increment {ReductionNumber:3d}")
                    NRConvergeNum = 0
                    Flag11 = 1
                    Flag20 = 1
                    break


            # Solve linear system: GKF \ Residual
            # print(spla.norm(GKF))
            # print(np.linalg.norm(Residual))
            IncreDisp = spla.spsolve(GKF.tocsr(), Residual)
            # print(np.linalg.norm(IncreDisp))
            Disp[:,0] += IncreDisp
            Flag20 = 1

    print(np.linalg.norm(Disp))

UM = np.linalg.norm(Disp.reshape((-1,3)), axis = 1)
octave.PlotStructuralContours(FEMod.Nodes,FEMod.Eles,Disp,UM.reshape((-1,1)))
```