Busan science high school

2023 Ocean ICT Festival
2023 BOIF

D 09

QR 코드 영역
QR 삽입 후
테두리 삭제

Youtube 영상 QR

심해 잠수정 모양과 내파(implosion)의 연관성

Team Lee

2316 이승준
2317 이윤호

1. 탐구 동기:

타이타닉 잠수정 사고로 8명이상의 사람들이 심해속에서 죽음을 맞이하였다. 주요 요인 중 하나가 잠수정의 모양이 내파를 견디기에 좋지 않았다는 것이다 ,이를 해결하기위해 잠수정 모양에 따른 내파 현상을 시뮬레이션하여 사용자들에게 제시하는 코드를 만들기로 했다.

2. 코드 작동 설명

- main.py: 시뮬레이션하고자 하는 상황에 대한 데이터를 만들며, 입출력 GUI를 처리한다.
- assembly.py: 각각의 (유한)요소들을 조합, 전역강성행렬(global stiffness matrix)를 계산한다.
- fem.py: 유한요소법에 대한 코드로, 유한 요소들에 대한 함수들(여러 모양, 텐서 계산)을 포함한다.
- gauss.py: Gauss-Legendre quadrature에서 가우스 점의 좌표와 가중치를 계산한다.
- uel.py: UEL(User ELement)에 대한 함수들로, 각각의 요소의 강성행렬(stiffness matrix)을 계산한다.

*FEA(유한요소해석): 연속적인 물체에 대한 문제(연속체 문제)를 이산화하여, 유한한 요소로 표현하여 수치적인 방법으로 근사해를 구하는 것

3. 융합 분야

내파(implosion)는 물체가 자체적으로 붕괴, 또는 압착되어 파괴되는 과정이다. 내파는 폭발과 반대로 안에서 폭발한다. 외부 압력에 맞서 내부 압력이 증가하다가 결국 내부에서 폭파가 일어나 듯 터지게 된다. 이러한 내파현상을 잘 고려하지 못한 심해 잠수정은 탐사 시에 갑작스런 폭파로 소멸하게 된다. 만약 무인이 아닌 유인 잠수정이라면 소중한 인명피해가 발생하고, 심해의 여러 생명체들에게도 큰 피해를 주게 된다. 우리가 만든 내파 시뮬레이션으로 잠수정 폭파를 사전에 방지하여 위 문제들을 해결할 수 있다.

4 코드 및 실행 결과 (코드양이 굉장히 많은 관계로, 자세한 것은 영상 참고 부탁)

```python
#!/usr/bin/python3
"""
main.py

References
[1] O.C. Zienkiewicz and J.Z. Zhu, The Superconvergent patch recovery and
    a posteriori error estimators. Part 1: The recovery technique,
    Int. J. Numer. Methods Eng., 33, 1331-1364 (1992).
"""

from datetime import datetime
import sys

import matplotlib.pyplot as plt
from matplotlib.tri import Triangulation
import numpy as np
from scipy.sparse import csr_matrix
from scipy.sparse.linalg import spsolve

import assembly
import fem


def complete_disp(bc_array, nodes, sol, ndof_node=2):
    """Fills the displacement vectors with imposed and computed values.

    Args:
        bc_array (ndarray): Indicates if the nodes has any type of
            boundary conditions applied to it.
        nodes (ndarray): Array with number and nodes coordinates.
        sol (ndarray): An array with the computed displacements.
        ndof_node (int, optional): The number of DOF per node. Defaults to 2.

    Returns:
        sol_complete (ndarray): The array with the displacements.
    """
    nnodes = nodes.shape[0]
    sol_complete = np.zeros([nnodes, ndof_node], dtype=float)
    for row in range(nnodes):
        for col in range(ndof_node):
            cons = bc_array[row, col]
            if cons == -1:
                sol_complete[row, col] = 0.0
            else:
                sol_complete[row, col] = sol[cons]
    return sol_complete

def create_input(num_nodes, radius, thickness, young, poisson, pressure):
    thetas = np.linspace(0, 2 * np.pi, num_nodes, endpoint=False)
    units = np.stack([np.cos(thetas), np.sin(thetas)], axis=1)
    nodes = np.concatenate([(radius-thickness)*units, radius*units], axis=0)
```

Main.py의 초반부분, 변위벡터를 계산 후 반환하는 코드 및 GUI 환경 설정 등이 보인다.

```python
def sparse_assem(elements, mats, nodes, neq, assembly_operator, uel=None):

    rows = []
    cols = []
    stiff_vals = []
    mass_vals = []
    nels = elements.shape[0]
    for ele in range(nels):
        kloc, mloc = retriever(elements, mats, nodes, ele, uel=uel)
        ndof = kloc.shape[0]
        dme = assembly_operator[ele, :ndof]
        for row in range(ndof):
            glob_row = dme[row]
            if glob_row == -1:
                continue
            for col in range(ndof):
                glob_col = dme[col]
                if glob_col == -1:
                    continue
                rows.append(glob_row)
                cols.append(glob_col)
                stiff_vals.append(kloc[row, col])
                mass_vals.append(mloc[row, col])

    stiff = coo_matrix((stiff_vals, (rows, cols)), shape=(neq, neq)).tocsr()
    mass = coo_matrix((mass_vals, (rows, cols)), shape=(neq, neq)).tocsr()
    return stiff, mass
```

Assembly.py의 전역강성행렬(global stiffness matrix)를 계산하는 핵심 부분

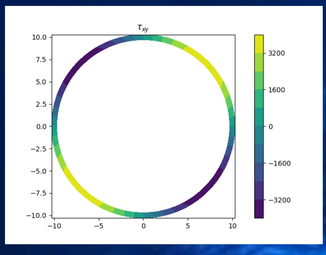
```python
def elastic_tri6(coord, params):
    stiff_matrix = np.zeros([12, 12])
    mass_matrix = np.zeros([12, 12])
    C = fem.umat(params[:2])
    if len(params) == 2:
        dens = 1
    else:
        dens = params[-1]
    gpts, gwts = gauss.gauss_tri(order=3)
    for cont in range(gpts.shape[0]):
        r, s = gpts[cont, :]
        H, B, det = fem.elastic_diff_2d(r, s, coord, fem.shape_tri6)
        factor = gwts[cont] * det
        stiff_matrix += 0.5 * factor * (B.T @ C @ B)
        mass_matrix += 0.5 * dens * factor * (H.T @ H)
    return stiff_matrix, mass_matrix


def spring(coord, stiff):
    vec = coord[1, :] - coord[0, :]
    nx = vec[0]/np.linalg.norm(vec)
    ny = vec[1]/np.linalg.norm(vec)
    Q = np.array([
        [nx, ny, 0, 0],
        [0, 0, nx, ny]])
    stiff_matrix = stiff * np.array([
        [1, -1],
        [-1, 1]])
    stiff_matrix = Q.T @ stiff_matrix @ Q
    return stiff_matrix, np.zeros(4)
```

Uel.py의 강성행렬(stiffness matrix)을 계산하는 여러 함수 중 일부분, 12개의 함수 중 2개가 위 사진

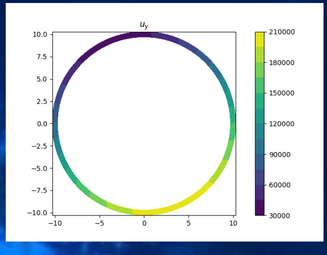
```python
# shape functions
def shape_tri6(r, s, t):
    N = np.array([
        (1 - r)*(1 - s)*(1 - t), (1 - s)*(1 - t)*(r + 1),
        (1 - t)*(r + 1)*(s + 1), (1 - r)*(1 - t)*(s + 1),
        (1 - r)*(1 - s)*(t + 1), (1 - s)*(r + 1)*(t + 1),
        (r + 1)*(s + 1)*(t + 1), (1 - r)*(s + 1)*(t + 1)])
    dNdr = np.array([
        [(1 - t)*(s - 1), (1 - s)*(1- t),
         (1 - t)*(s + 1), (1 - t)*(-s - 1),
         (1 - s)*(-t - 1), (1 - s)*(t + 1),
         (s + 1)*(t + 1), -(s + 1)*(t + 1)],
        [(1 - t)*(r - 1), (1 - t)*(-r - 1),
         (1 - t)*(r + 1), (1 - r)*(1 - t),
         -(1 - r)*(t + 1), -(r + 1)*(t + 1),
         (r + 1)*(t + 1), (1 - r)*(t + 1)],
        [-(1 - r)*(1 - s), -(1 - s)*(r + 1),
         -(r + 1)*(s + 1), -(1 - r)*(s + 1),
         (1 - r)*(1 - s), (1 - s)*(r + 1),
         (r + 1)*(s + 1), (1 - r)*(s + 1)]])
    return .125*N, .125*dNdr


def shape_tet4(r, s, t):
    N = np.array([1 - r - s - t, r, s, t])
    dNdr = np.array([
        [-1., 1., 0., 0.],
        [-1., 0., 1., 0.],
        [-1., 0., 0., 1.]])
    return N, dNdr


def shape_tri3(r, s):
    N = np.array([1 - r - s, r, s])
    dNdr = np.array([
        [-1., 1., 0.],
        [-1., 0., 1.]])
    return N, dNdr
```

Fem.py의 유한요소들에 대한 여러 함수 중 잠수정 모양과 관련된 shape function 부분 이 외에도 축대칭 탄성에 대한 보간행렬 등을 반환하는 함수가 있다.

결과



Strain(응력)을 나타낸 결과



변위벡터을 나타낸 결과

5.기대 효과

이러한 내파 시뮬레이션을 통해서 타이타닉 잠수정 사고와 같은 비극을 다시는 볼 수 없을 것이다. 또한 위 코드를 잘변환한다면, 잠수정외에 항공기와 같은 유체에 대한 압력을 받는 여러 운송수단에도 적용시킬 수 있다.