

FSI_T1

Fluid-Structure Interaction Practice Homework

by harryzhou2000 @

School of Aerospace Engineering, Tsinghua University

- Introduction
 - Origin
 - Basic Purpose
 - About Project Source Code and Dependencies
 - What did I code?
 - Dependencies
- Numerical Methods and Algorithms
 - Fluid Methods
 - Structural Methods
 - FSI Methods
- A Brief Guide To Using This Code
 - Implementation Framework
 - Classes
 - Fluid
 - Structural
 - FSI
 - Input File Convention
- Case Building
- Compiling
 - Make
- Notes
- References

Introduction

Note: if you speak Chinese, my [report for coursework](#) is recommended if you desire to go through some analysis of specific cases.

Origin

This project was a coursework for *the Fundamental of Computational Mechanics* of THU in spring 2021. The course contains both CFD and CSD, and lecturers said I could complete a FSI program in place of both final assignments, as a result this program was born. (It was said FSI projects had seldom been seen in the course apart from one, but I didn't actually get an A for this second record

)


Basic Purpose

FSI_T1 is basically a c++ (with some template) library providing multiple classes, which enables you to assemble from them a custom

,



or

 computation case. Fluid part currently supports only Euler equation, which represents adiabatic and inviscid compressible ideal gas dynamics. Solid part currently supports linear elastic mechanics, including statics and modal-truncation method dynamics. FSI part, correspondingly, is basically meant for moderate structural displacement, typically aeroelastic cases.

The library currently does not have a interface supporting script input or interactive case setting, which means one must construct each single case inside a c++ calling of the relevant classes.

About Project Source Code and Dependencies

What did I code?

All the actual source code files are put in **root** of project directory, all as .hpp or .h files. The cpp files are meant for case building, containing various case sets, and they can be viewed as some kind of

'static scripts' for the program.

The draw-back of not having a script interface and using .cpp as 'script' is that each time you have a new case to compute a re-compiling is needed, which could be rather time consuming.

Dependencies

This project depends on Eigen, which completely is a c++ template library, with no binary file linking. For convenience, I just threw the entire Eigen source code in ./include/, so you won't have to install it. This project also needs c++ standard library and STL to work, which can be handled automatically by compilers mostly.

Numerical Methods and Algorithms

Fluid Methods

To comply with significant movement of boundaries caused by structural displacement, the method of arbitrary Euler-Lagrangian (ALE) description of fluid is adopted. The major differences between ALE and Euler descriptions is that ALE adds some items to Euler relating to the mesh speed[\[1\]](#).

Consider Euler equation for gas dynamics in conservative form:

$$\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} + \frac{\partial H}{\partial z} = 0$$
$$U = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ E \end{bmatrix}, F = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ u(E + p) \end{bmatrix}, G = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ v(E + p) \end{bmatrix}, H = \begin{bmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ w(E + p) \end{bmatrix}$$

Thus in a ALE control volume:

$$\frac{\partial}{\partial t} \int_{\Omega(t)} U dV + \int_{\partial\Omega(t)} U (u_{\Omega x} n_x + u_{\Omega y} n_y + u_{\Omega z} n_z) d\Gamma + \int_{\partial\Omega(t)} (F n_x + G n_y + H n_z) d\Gamma = 0$$

Where $u_{\Omega i}$ are the speed components of the C.V.'s boundaries. The corresponding bold symbols are for vectors in xyz space.

Annotating:

$$\mathbf{u}^* = \mathbf{u} - \mathbf{u}_{\Omega}$$

Thus:

$$\begin{aligned} \frac{\partial}{\partial t} \int_{\Omega(t)} \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ E \end{bmatrix} dV + \int_{\partial\Omega(t)} \left(\begin{bmatrix} \rho u^* \\ \rho u^* u^* + p \\ \rho u^* v^* \\ \rho u^* w^* \\ u^* (E + p) \end{bmatrix} + \begin{bmatrix} \rho v^* \\ \rho u^* v^* \\ \rho v^* v^* + p \\ \rho v^* w^* \\ v^* (E + p) \end{bmatrix} + \begin{bmatrix} \rho w^* \\ \rho u^* w^* \\ \rho v^* w^* \\ \rho w^* w^* + p \\ w^* (E + p) \end{bmatrix} \right) d\Gamma \\ + \int_{\partial\Omega(t)} \rho (u_x^* n_x + u_y^* n_y + u_z^* n_z) \begin{bmatrix} 0 \\ u_{\Omega x} \\ u_{\Omega y} \\ u_{\Omega z} \\ 0 \end{bmatrix} d\Omega = 0 \end{aligned}$$

Thus if using \mathbf{u}^* as speed, the flux could be approximated with a common approximate Riemann solver when the approximate field is not C^0 . Other items could be handled rather simply.

Finite volume spacial discretion is adopted(FVM). This program uses Roe's approximate Riemann solver[2], which is modified with entropy fix of Harten-Yee[3] for numerical flux. This program conducts 2nd-order reconstruction with Barth-Jespersion limiter[4].

The boundary conditions are built basically in the form of virtual cells. Slip-wall boundary and non-reflecting boundary are implemented. A pressure inlet/outlet boundary is also implemented , but its robustness is yet to be improved.

Time discretion in fluid part is a implicit Euler scheme.

The mesh considering FVM is completely composed of tetrahedra.

Structural Methods

FSI Methods

A Brief Guide To Using This Code

Implementation Framework

Classes

Fluid

Structural

FSI

Input File Convention

Case Building

Compiling

Make

```
make main.exe #for debug
make mainR.exe #for release
make mainSG.exe #for mainSG debug
make mainSGR.exe #for mainSG release
```

Notes

It was only after I STARTED this document when I find my English being rather amateur, even in my own domains... I even searched for a proper antonym for 'proficient' for use in the previous sentence... So please forgive me if any expression in my text seems strange or erroneous.

References

- [1] P, Le, Tallec, et al. Fluid structure interaction with large structural displacements[J]. *Computer Methods in Applied Mechanics and Engineering*, 2001, 190(24-25):3039-3067.
- [2] Roe P L . Approximate Riemann solvers, parameter vectors, and difference schemes[J]. *Journal of Computational Physics*, 1981, 43(2):357-372.
- [3] Yee H C . Upwind and symmetric shock-capturing schemes. 1987.
- [4] Barth T J , Jespersen D C . The design and application of upwind schemes on unstructured meshes[J]. AIAA Aerospace Sciences Meeting, 1989, 0366(13).