Implementation of Preprocessor for CSCM code by using Graphic User Interface

Evgeny G. Ivanov*1, Dong Joo Song*2

그래픽 환경을 이용한 CSCM 수치해석 코드에서의 전처리 과정 개발

예브게니 이바노프, 송 동 주

본 연구에서는 격자 생성, 초기유동조건 및 경계조건 설정 등 일련의 전처리 과정을 사용자에게 친숙한 그래픽 인터페이스 환경으로 개발하였다. MFC/Visual C++를 이용하여 개발된 전처리 프로그램은 Windows 계열의 OS와 호환이 가능하며, 기하학적 격자생성, 초기값 설정 및 수치해석 코드의 제어변수를 생성할 수 있다. 한편 사용자의 편의를 위해서 전처리 과정을 격자생성(단일격자생성, 다중격자생성), 유체 물성치정의, 경계조건 생성, 초기조건 생성 및 코드제어로 구분하였다.

개발된 전처리 프로그램의 특성으로서 다중 격자 생성 작업을 단일 격자계의 중첩으로 구성될 수 있도록 각 경계면을 "interface"형을 취하는 기능을 제공하도록 하였으며 개발된 전처리 과정을 16도의 경사면을 가지는 Compression ramp 문제 및 축대칭 Bump 문제에 적용하여 개발된 전처리 과정을 검증하였다.

Key Words : 전처리(Preprocessor), GUI(Graphic User Interface), CSCM code

1. Introduction

Preprocessor plays an important role in CFD field. The main function of preprocessor is to prepare initial starting data and generates control input parameters for CFD solver. It is responsible for correct input information and its successful operation.

For user, a preprocessor is a part of software package, which provides convenient working environment where one can generate geometry, computational mesh and required input data.

Graphic User Interface has significant importance for preprocessor since common users can use the preprocessor and, therefore, Navier-Stokes solver for their own problems.

Objective of this research is to develop the preprocessor for CSCM Upwind Flux Difference Splitting Navier-Stokes code [2] which generates geometry and mesh, input data for solver in convenient Graphic User Interface environment.

2. Analysis: strategy of Implementation

2.1 Examples of preprocess

There are a lot of preprocessors in many different famous commercial packages which can be good examples of preprocessor, but two of them were mainly considered as reference in this study; Gambit [3] and CFD-FASTRAN preprocessor [4].

*1 영남대학교 대학원, 영남대학교 대학원 기계공학과
E-mail : ievg@yumail.ac.kr
*2 종신회원, 영남대학교 기계공학부,
E-mail : djsong@yu.ac.kr

2.2 Visual C++ MFC and GUI environment

One of the main features of the developing preprocessor is a convenient Graphic User Interface. Visual C++ and MFC provide us programming tools and automatic techniques for rapid visual application development. An application created with Visual C++ and MFC will automatically generate most of its own windows, handle its own messages and do its own drawing [5].

The power of C language combined with visual application development determines the choice of Visual C++ 6.0 as the most suitable environment for preprocessor implementation.

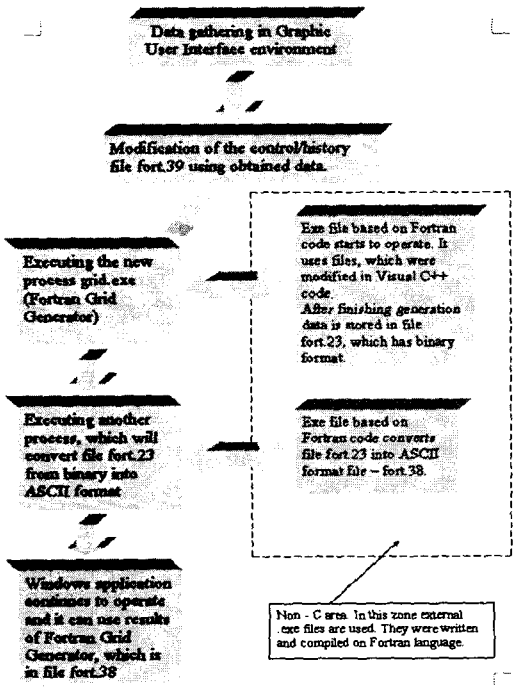


Fig.1 Schematic view of the processes interactions

2.3 FORTRAN 77 and Visual C++ 6.0 collaborative work

As mentioned above, some of preprocessor steps were usually done manually or with help of additional programs. The most important part of it was FORTRAN single block grid generator.

Basically, there were two choices:

1) To write new grid generator or rewrite FORTRAN grid generator into C language.

2) Somehow, use FORTRAN single block grid generator in the new developed Windows application.

Obviously, second way is much easier than first one. But it requires sophisticated method of implementation.

Following programs or data files are involved in current preprocessor step:

Grid.exe FORTRAN grid generator .exe file.

Fort.39 control / history data file for FORTRAN grid generator.

Fort.23 FORTRAN grid generator output binary solution data file.

Plotg.exe FORTRAN .exe file which converts fort.23 data file from binary into ASCII format.

Fort.38 ASCII format data file which is result of converting fort.23.

Suppose, the way to execute alien process from the main Windows application is found.

Figure 1 shows scheme of processes interactions. Data exchange between processes will be carried out through files.

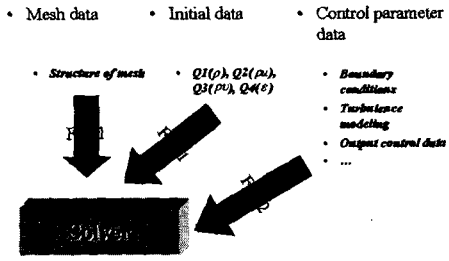


Fig.2 Structure of output data

2.4 Structure of data and their location

A preprocessor provides data to solver. This data can be classified into three groups: grid with geometry, initial data, and control parameters data. It is necessary to provide all information for successful solver operation.

All data are provided in the form of files: fort.1 includes mesh data and initial solution data, fort.5 control parameter data. Figure 2 shows structure of output preprocessor data.

2.5 Multi block grid generator

Usually, single block grid is not adequate for solving realistic problems. Existence of multi block option for grid generation crucially changes number of available types of grid which can be generated by preprocessor. FORTRAN grid generator is a single block grid generator. It means, that user have to generate several single blocks and join them together manually to solve complex flow problems. It is another objective of the work to make this process automatically.

3. Results and discussions

3.1 _spawn functions and new FORTRAN process

By calling spawn function [6] it is possible to execute a new process. In this case FORTRAN single block grid generator should be executed. By using the spawn function it is possible to implement the scheme of processes interactions as shown in Fig.1. Figure 3 shows how _spawn function is implemented in this MFC application.

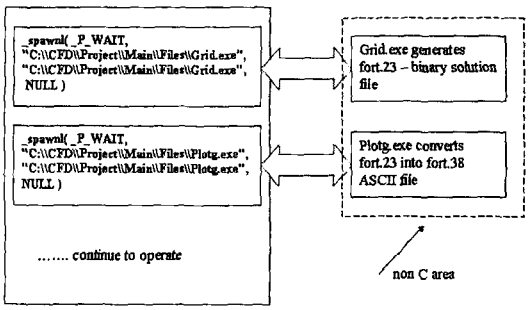


Fig.3_spawn function implementation

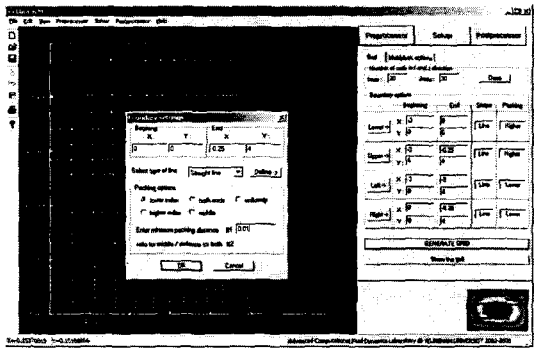


Fig.4 Grid generation part in the application

3.2 Contents of preprocessor

3.2.1 Grid generation

The grid generation part is the most important step in preprocessor work. If grid generation part has many options and convenient interface, user can create desired mesh where he is going to get solution.

User has to follow certain scheme to create grid. First of all, define number of cells for both directions. Then, choose boundary and set for it: coordinates of beginning and end, shape of the line, type of packing. After program has all information about each boundary user can generate grid. Figure 4 shows how it is implemented in developed application.

The Main Frame window consists of two parts: first is part for drawing and second is control part for data input and information representation.

By clicking on button with boundary caption user can input necessary data. The edit boxes from the right side of the button display information such as coordinates of the beginning and end of boundary, shape and type of packing. When user clicks on the button, program calls dialog window where one can input all data for boundary setting.

If user chooses, for example, spline as a type of boundary, then, after confirmation by "Done" button, program will call object of class [7] for spline data manipulations (Fig.5).

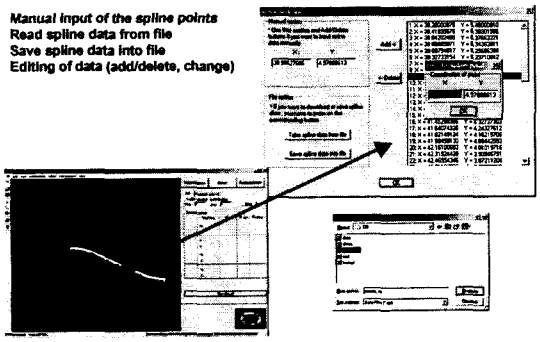


Fig.5 Geometry. Tools for spline data

3.2.2 Multi-block option

Multi-block option should be enabled for multi-block grid. It means that after generating several blocks they are joined together into one whole grid. For that purpose in Boundary Conditions part "interface" type of boundary is added. It indicates boundary interfacing with another block.

3.2.3 Flow definition

After grid is generated user can start to define his problem. Flow definition part in the developed preprocessor is for flow properties settings.

Here user can input problem name, reference values, Prandtl number, choose whether flow is viscous or not, steady or unsteady and etc. Figure 6shows how it is implemented in application and tree of options and models are available in this part.

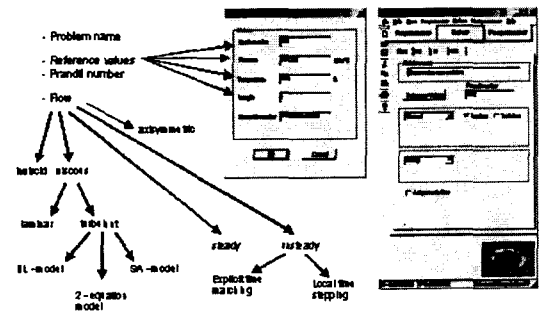


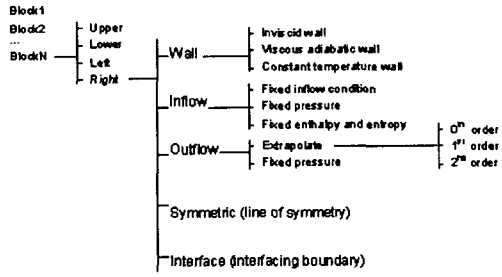
Fig.6 Flow definition part in the application

Reference values dialog is used for reference values input. User can input Mach number, pressure, temperature and length. Reynolds number will be calculated automatically based on input data.

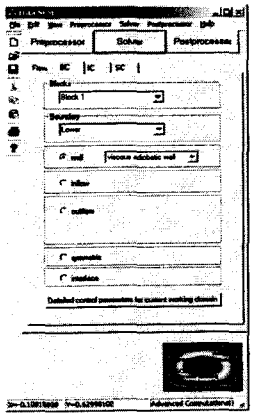
3.2.4 Boundary conditions

Boundary conditions part consists of three main sections. First section is where user has to choose block. Second is where user should choose boundary. And third is for type of boundary condition itself.

Figure 7(a) shows list of available options for Boundary Conditions part and how it is realized in the application (Fig. 7(b)).



(a) List of available options



(b) View in the application

Fig.7 Boundary Conditions part in the application

If user wants to control process of solution in

detail there is "Detailed control parameters for current working domain"button. By clicking on that button you call dialog box where it is possible to input detailed information for both directions of integration.

3.2.5 Initial conditions

Initial conditions can be generated by three manners. It can be taken from file, generated by using reference values or can be entered manually by user.

As shown in Fig.8, in the lower part of the panel there are edit boxes which display density, U-velocity, V-velocity and pressure.

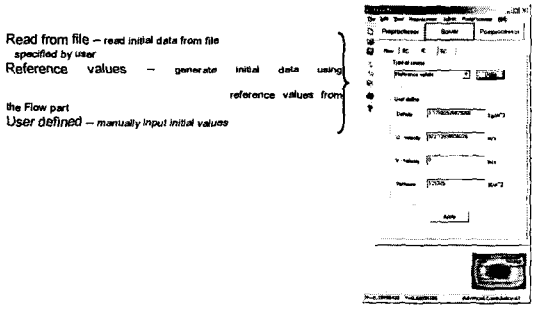


Fig.8 Initial conditions part in the application

3.2.6 Solver controls

Solver control part includes parameters to control solver work such as iterations number, order of accuracy, initial CFL number and final CFL number.

If user wants to control solution procedure more detailed, there is "Detailed options" button which calls dialog where you can input more detailed information and can see summary of data which was already entered (Fig.9).

"Run"button in the lower part of the panel executes solver with intermediate solution and "New Run" button executes solver with Initial Solution data.

3.3 Test problems

3.3.1 16 ◦ compression ramp

For demonstration purpose a 16 ◦ compression ramp problem was examined [8] with inflow conditions shown in Fig.10. All procedure of grid generation and input data preparation will be shown in presentation.

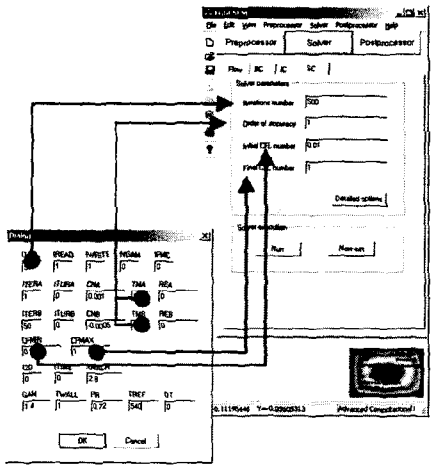


Fig.9 Solver controls in the application

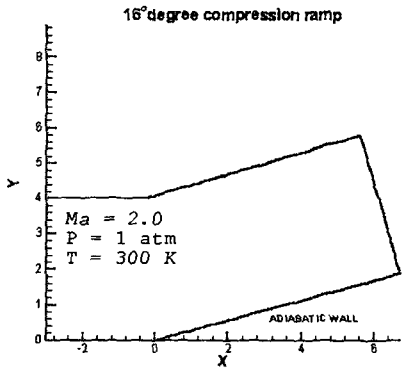


Fig.10 16 degree compression ramp problem

This grid consists of two blocks. First, should be generated left part and after that right part. Then, using multi-block option it will be connected into one grid (Fig.11).

Notice, that right boundary of the left block and

left boundary of the right block have special "interface" type, which indicates the interfacing boundary".

Next steps are Flow definition part, Initial conditions and Solver controls. After defining all parameters user can execute solver.

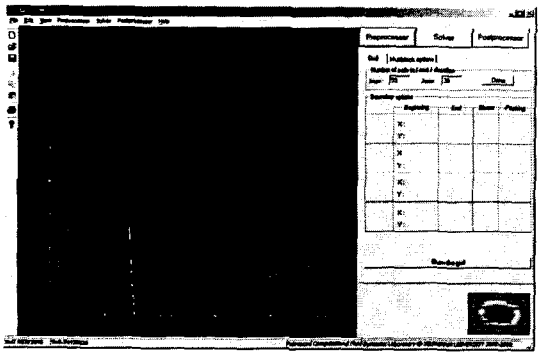


Fig.11 Compression ramp: summation of two blocks

3.3.2 Axisymmetric bump problem

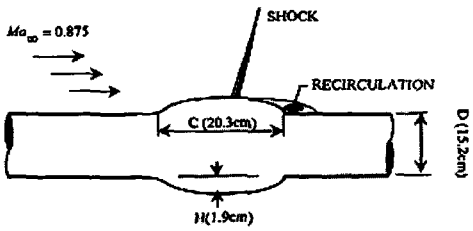


Fig.12 Axisymmetric bump problem

The grid for that problem(Fig. 12) consists of three blocks. Each of them will be generated. First is the left block, second - middle one and last will be right block. After that by using multi-block option it is possible to join these three blocks into one desired grid.

Figure 13 shows result of summation.

After grid is generated we can define flow properties, Initial conditions and Solver control parameters.

The "Run"button executes CSCM solver and postprocessor takes control of solution procedure [9].

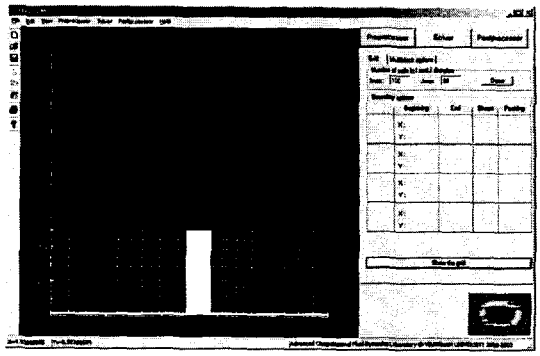


Fig.13. Axisymmetric bump: summation of three blocks

4. Summary and conclusions

4.1 Conclusions

As a result of the work the preprocessor for 2D/Axisymmetric CSCM Upwind Flux Difference Splitting Navier-Stokes solver was developed with MFC/ Visual C++. The application is compatible with Windows 95/98/Millenium/NT/2000/XP and capable to generate geometry and grid, initial solution data and required solver control parameters. Developed MFC application has a lot of visual tools such as grid system visualization, control tools for data manipulation, interactive data display, saving options, etc. Special feature of the created preprocessor is that work of grid generation part is based on cooperation of the MFC/Visual C++ application and FORTRAN single block grid generator process. Multi-block option allows user to construct block grid using special "interface" type of boundary in Boundary Conditions (BC) part. Simple geometries for boundary shape are available: straight line, spline, circle and ellipse. For user convenience preprocessor steps are classified as following: Grid Generation (Grid), Multi-block option, Flow Definition (FD), Boundary Conditions (BC), Initial Conditions (IC) and Solver Control (SC).

4.2 Further works

The following issues could be further works.

- Connection of Preprocessor, solver and postprocessor into one jointly working package.
- Expand multi-block option capability.
- Grid interface modification by using elliptic solver [10].
- Development of IC generation techniques for various turbulent models.
- Addition of visual control effects for grid development.
- Making of the application more flexible for the purpose of expansion of applicability.

References

[1] "Tecplot. Interactive Data Visualization for Scientists & Engineers. User's manual.", Ventura Software, Inc.

[2] Lombard, C. K., Bardina, J., Venkatapathy, E. and Oliger, J., 1983, "Multi-Dimensional Formulation of CSCM An Upwind Flux Difference Eigenvector Split Method for the Compressible Navier-Stokes equations, " AIAA-83-1895.

[3] FLUENT, 1998, Fluent Inc.

[4] "CFD-FASTRAN. User's Manual. Version 2002.", CFD Research Corporation.

[5] John E. Swanke, 1999, "Visual C++ MFC Programming by Example". CMP Books, CMP Media, Inc.

[6] MSDN Library, July 2001.

[7] Stepherd, George. 2003, "Programming with Visual C++ .NET, Sixth Edition (Core Reference)", Microsoft Press.

[8] Sang Dug Kim, 1999, "Performance Analysis of Three-Dimensional Transonic Centrifugal Compressor Diffuser.",1999. Ph.D. Thesis.

[9] Juraeva Makhsuda, 2003, "Inplementation of Postprocessor for CSCM code by using Graphic User Interface", M.S. Thesis.

[10] Klauss A. Hoffman, Steve T. Chiang, 1993, "Computational fluid dynamics for engineers", A Publication of Engineering Education System