공개 전산열유체 시뮬레이션
OpenFOAM의 설치 및 사용 메뉴얼

손일엽 (KISTI)
김병윤, 노현석 (넥스트폼)

공개 전산열유체 시뮬레이션 OpenFOAM의 설치 및 사용 메뉴얼

손일엽 (KISTI)
김병윤, 노현석 (넥스트폼)

KiSTi 한국과학기술정보연구원
Korea Institute of Science and Technology Information

목 차

1. OpenFOAM을 시작하기 전에

1.1 OpenFOAM 소개

OpenFOAM은 Open Field Operation and Manipulation의 약자로, 1990년대 초 영국의 Imperial College에서 개발을 시작하였다. OpenFOAM은 연속체역학 전산해석 프로그램 작성을 위한 라이브러리로 2004년말 OpenCFD Ltd.에서 v1.0을 GNU-GPL 라이선스로 공개 되었다. 대표적인 연구자로는 Wikki Ltd.의 Hrvoje Jasak 박사와 OpenCFD Ltd.의 Henry Weller 박사가 있다.

OpenFOAM의 라이선스 형태인 GNU-GPL(General Public License, 공중사용허가서)은 Linux와 동일한 라이선스 형태로 source code가 공개되어 있으며, 누구나 수정 및 재배포가 가능하다. 2차적 저작물에도 동일한 GNU-GPL이 적용된다. 상업적, 비상업적 사용에 제약이 없으며, 개인이나 단체에 대한 차별도 없다. 상업적으로 사용하는 경우 구매자가 source code 요청 시 반드시 source code를 제공해야만 한다.

OpenFOAM의 가장 큰 장점은 라이선스 비용이 발생하지 않는 것이다. 대부분의 상용프로그램이 범용으로 개발되는 것과 달리 OpenFOAM 프로그램들은 전용으로 개발되기 때문에 문제에 최적화 되고 사용이 단순하다. 또한, 숙련된 사용자의 경우 비교적 빠른 시간에 새로운 형태의 전산열유체역학(CFD) 프로그램 작성이 가능하며, 필요에 따라 독자적인 물리모델을 적용할 수 있다. 다양한 상용 프로그램들의 입출력 형식으로 OpenFOAM의 데이터를 변환하거나 OpenFOAM의 데이터를 상용 프로그램 형식으로 변환시키는 프로그램들이 개발되어 있어 호환이 가능하다. 그러나 OpenFOAM은 리눅스에서만 작동하기 때문에 리눅스 사용 경험이 없는 사용자의 경우 거부감이 들 수 있다. OpenFOAM을 윈도우에서 사용할 수 있도록 porting(포팅)한 경우도 있지만 버전이 낮고 일부 기능이 제한된다. 또한, Graphic User Interface(GUI)가 아닌 Text User Interface(TUI)로 되어 있어 사용자의 실수가 많을 수 있다. OpenFOAM의 가장 큰 단점은 매뉴얼이나 기술지원 조직이 약하다는 것이다. 아직 국내에는 사용자 수도 적어 정보와 지식을 공유하기도 어려운 실정이다. 국내에서 OpenFOAM에 대한 기술지원을 받을 수 있는 사이트는 www.nextfoam.co.kr과 www.okuc.org가 대표적이다.

OpenFOAM은 CFD 프로그램을 개발하기 위한 C++ 라이브러리와 그 라이브러리로 개발된 Application(어플리케이션)으로 구성되어 있다. 어플리케이션은 solver와 utility로 구성되는데, solver는 유동해석을 위한 프로그램을 의미하며, utility는 전후처리 또는 형식변환 등에 필요한 프로그램을 의미한다.

OpenFOAM 라이브러리를 이용하면 아래와 같이 벡터형태의 미분방정식을 단순하게 표현 할 수 있다. CFD 이론과 OpenFOAM을 잘 이해하고 있는 사용자는 새로운 solver를 비교적 빠른 시간에 개발할 수 있지만, solver를 개발할 수 있을 만큼

OpenFOAM을 이해하는 데는 많은 시간과 노력이 필요하다.

$$\frac{\partial \rho \mathbf{U}}{\partial t} + \nabla \cdot \phi \mathbf{U} - \nabla \cdot \mu \nabla \mathbf{U} = -\nabla p$$

```
solve
(
    fvm::ddt(rho, U)
  + fvm::div(phi, U)
  - fvm::laplacian(mu, U)
    ==
  - fvc::grad(p)
);
```

본 매뉴얼은 상용유동해석 소프트웨어 사용경험이 있고 OpenFOAM을 처음 사용하는 사용자를 위한 것으로 유체역학과 CFD에 대한 기본이론은 설명하지 않으며 OpenFOAM의 설치와 몇 가지 표준 어플리케이션의 사용법에 대해서만 설명한다.

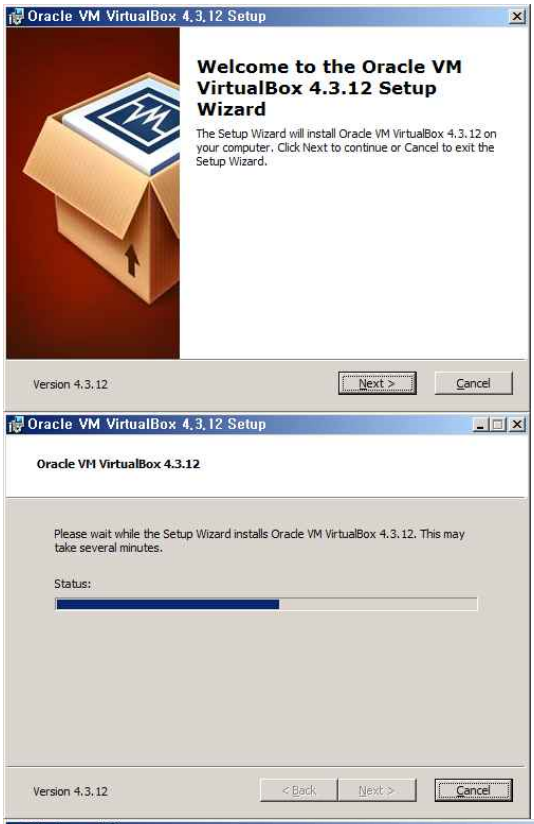
1.2 OpenFOAM을 위한 환경구축

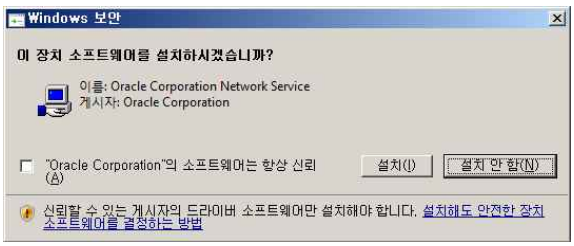
윈도우를 사용하고 있는 예비사용자라면 OpenFOAM을 학습하기 위하여 윈도우를 삭제하고 리눅스를 설치할 필요는 없다. 본 매뉴얼에서는 Oracle에서 무료로 제공하는 VirtualBox라는 가상화 소프트웨어를 이용하여 Ubuntu(우분투) 리눅스 환경을 구현할 것이다.

- 가상화 소프트웨어 VirtualBox 설치

VirtualBox는 아래 그림과 같이 https://www.virtualbox.org/wiki/Downloads에서 현재 사용 중인 컴퓨터의 OS에 맞는 버전을 선택하여 다운 받을 수 있다. 본 매뉴얼에서는 Windows용 VirtualBox를 기준으로 설명하겠다.

VirtualBox를 다운받고 실행하면 아래와 같이 설치가 시작된다. 가상장치 구동에 필요한 소프트웨어 설치를 동의하면 VirtualBox의 설치가 완료된다.





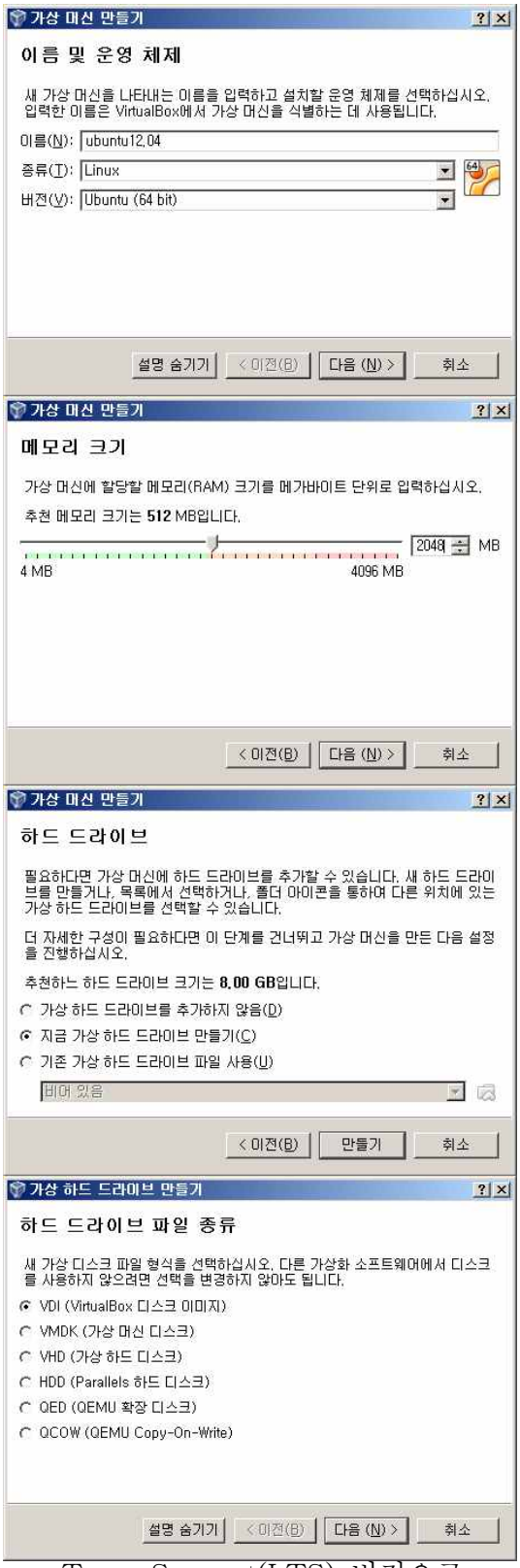
VirtualBox를 실행하면 아래와 같은 프로그램이 실행된다. '새로 만들기'버튼을 클릭하여 우분투 리눅스를 실행하기 위한 가상장치를 만들어야 한다.

설치할 OS를 구분하기 쉽게 이름을 정하고, 종류로 Linux를 선택하고 버전은 Ubuntu를 선택한다. 메모리의 크기는 2GB정도로 설정하면 OpenFOAM의 어플리케이션을 실행하는데 문제가 없다. '지금 가상 하드 드라이브 만들기'를 선택하여 진행하고, 하드 드라이브의 용량은 40GB 정도로 지정한다.

가상장치를 만들기를 완료하면 VirtualBox에 만들어진 가상장치를 볼 수 있다. 만들어진 가상장치를 실행하기 전에 우분투 리눅스의 설치파일을 다운받아야 한다.

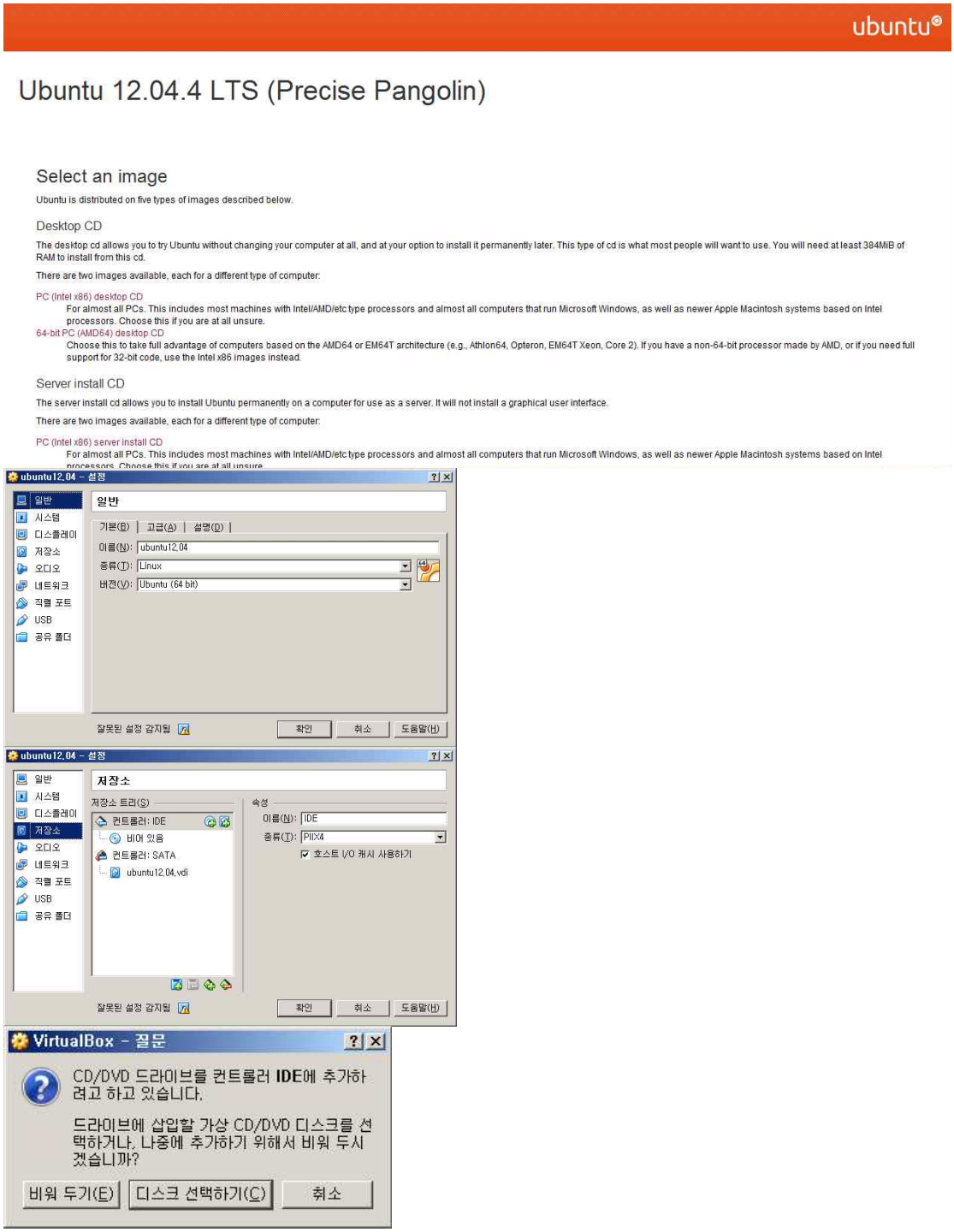
- 우분투 리눅스 설치

우분투의 버전은 발표 년도와 월을 의미한다. 짝수 년 4월에 발표되는 버전은



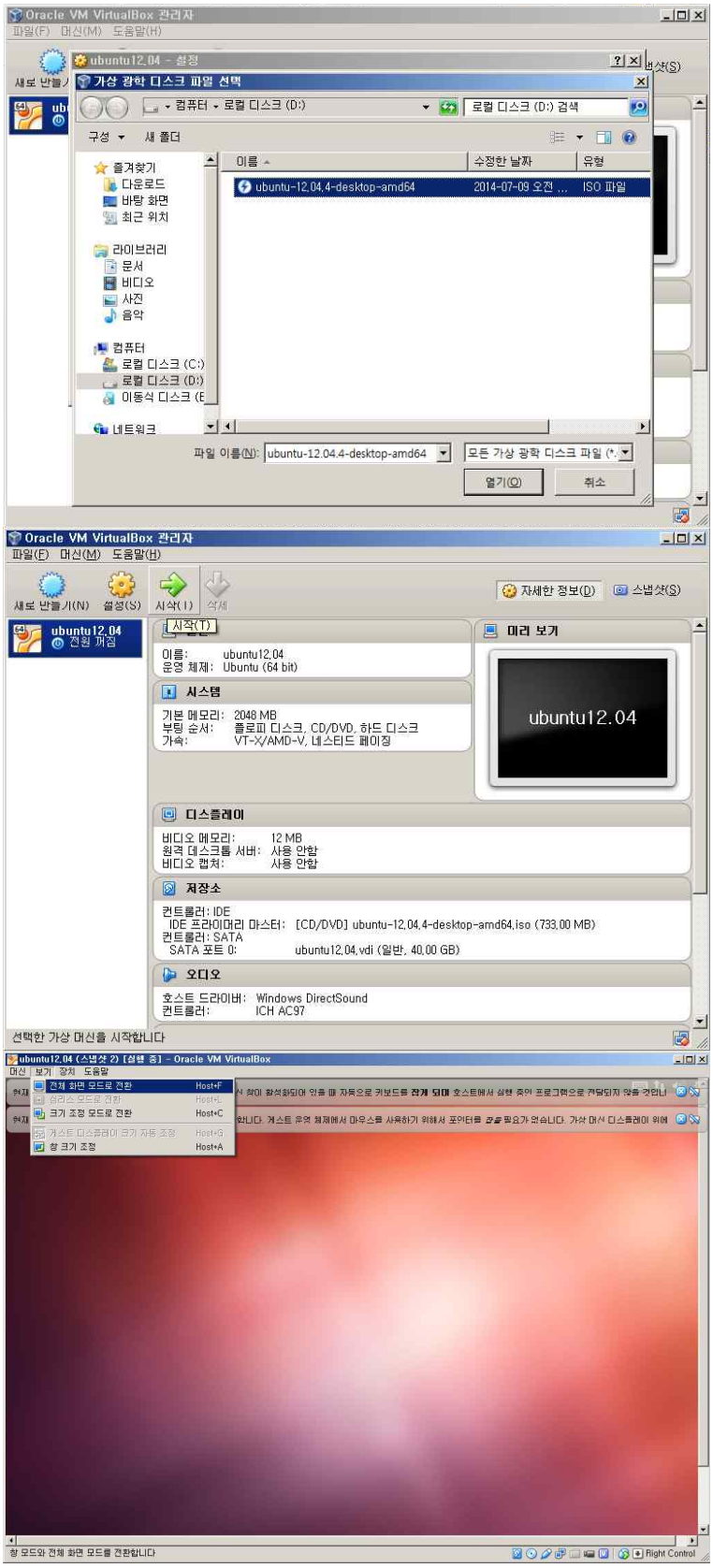
Long Term Support(LTS) 버전으로 update 기간이 길고 안정적이다. 최신 버전은 14.04이지만, 본 매뉴얼에서는 안정성이 검증된 12.04로 진행하겠다. 우분투 12.04의 설치파일은 http://releases.ubuntu.com/12.04/에서 다운 받을 수 있다.

이미지 파일을 다운 받은 후 VirtualBox의 '설정'버튼을 클릭하면 아래와 같은 창이 뜨게 된다.



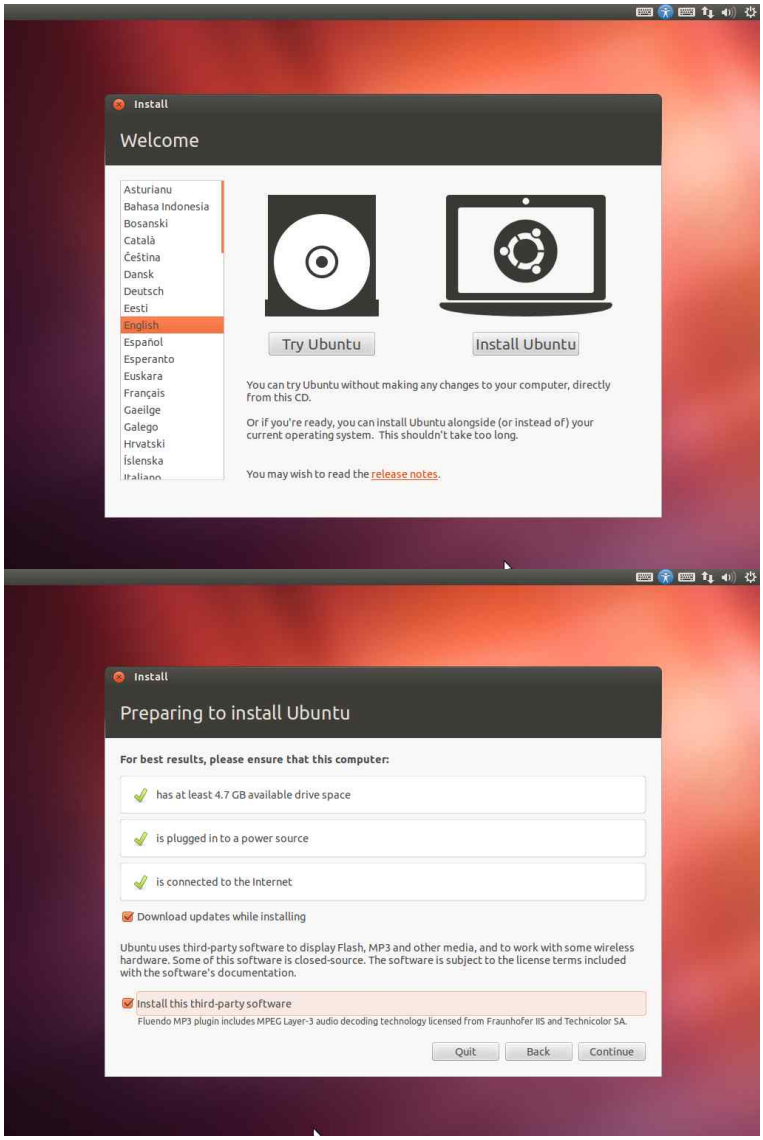
'저장소'의 'IDE'에서 다운 받은 우분투 이미지 파일을 선택하고 가상 장치를 실행하면 아래와 같이 우분투 화면이 뜨게 된다. 화면이 작아 불편하면 pull-down메뉴의 보기에서 전체화면모드로 전환할 수 있다. 전체화면모드에서 마우스를 화면의 중앙 아래로 내리면 전체화면모드에서 빠져나올 수 있는 아이콘이 나타난다.

리눅스 설치 시 '한국어'를 선택하는 것보다 아래와 같이 'English'로 선택하고 별

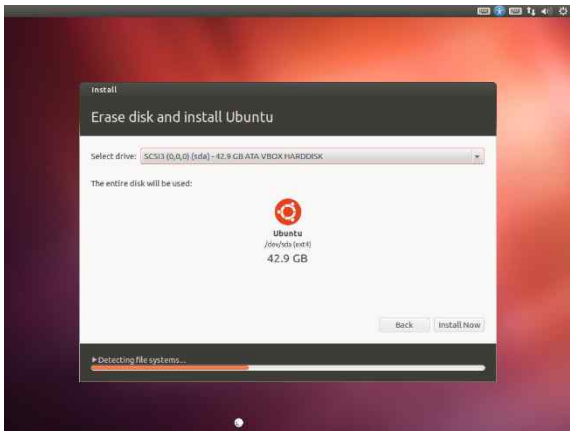


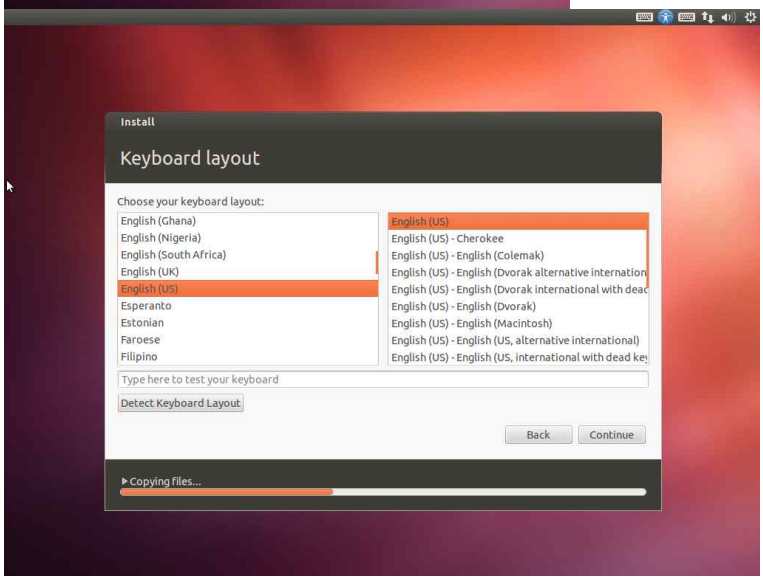
도로 한글입력기를 설치하는 것이 더 안정적이다. 아래와 같은 화면에서 'English'

와 ‘Install Ubuntu’를 선택한다.

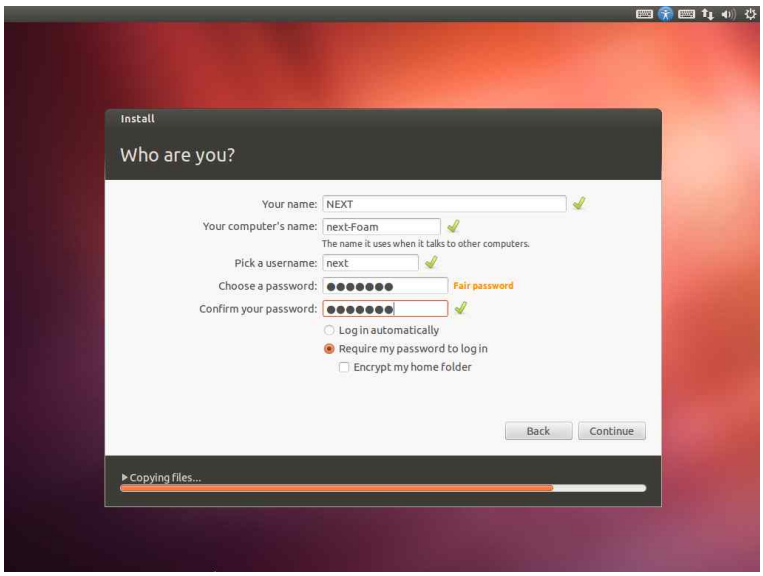


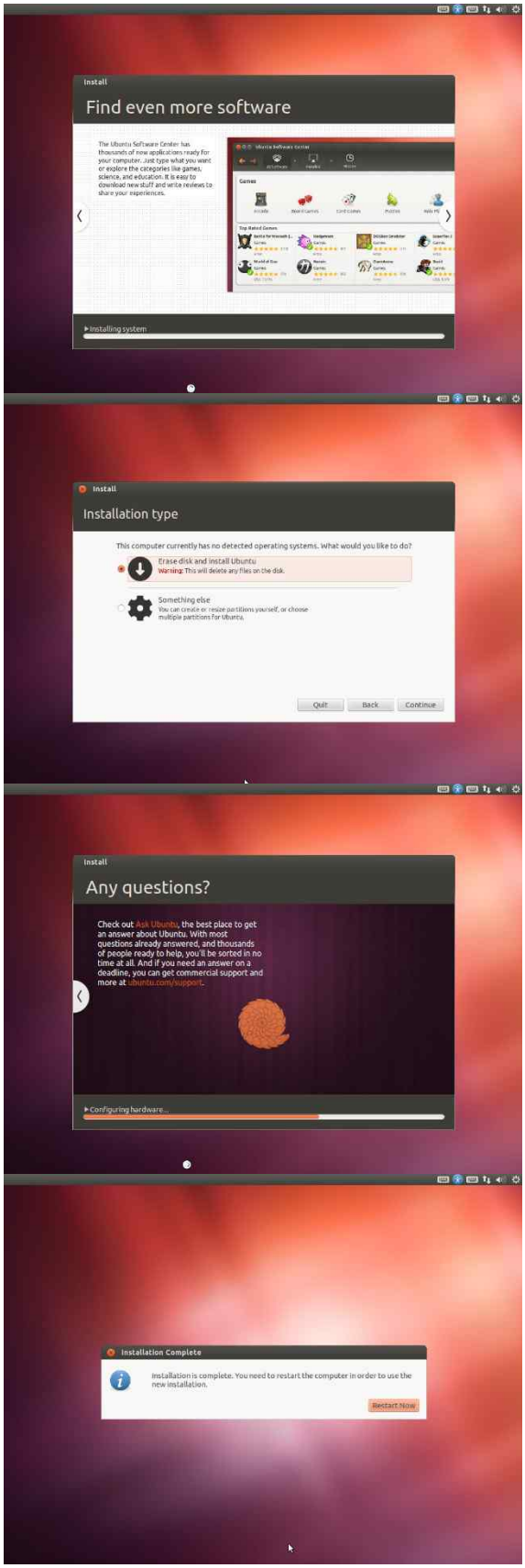
Update와 third-party 설치는 선택하지 않아도 된다.



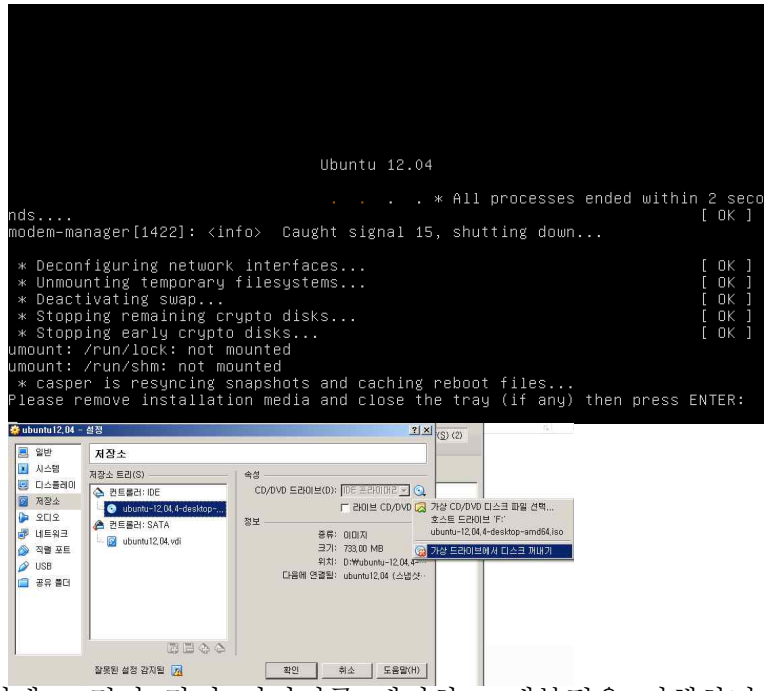


키보드 선택도 언어선택과 마찬가지로 영어로 선택한다. 사용자 이름과 회사명, 비밀번호들을 입력하고 설치를 계속 진행한다.

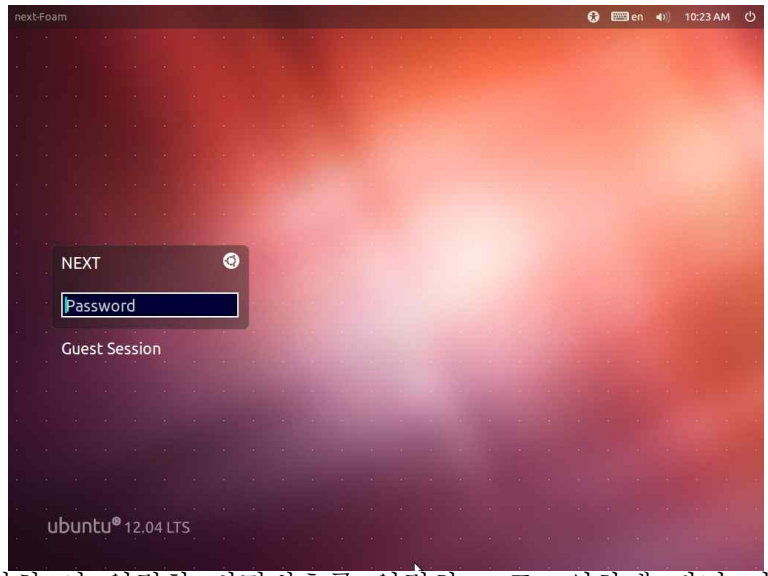



Install
Find even more software
The Ubuntu Software Center has thousands of new applications ready for your computer. Just type what you want or explore the categories like games, science, and education. It is easy to download new stuff and write reviews to share your experiences.
Games
Installing system
Install
Installation type
This computer currently has no detected operating systems. What would you like to do?
Erase disk and install Ubuntu
Warning: This will delete any files on the disk.
Something else
You can create or resize partitions yourself, or choose multiple partitions for Ubuntu.
Quit
Back
Continue
Install
Any questions?
Check out Ask Ubuntu, the best place to get an answer about Ubuntu. With most questions already answered, and thousands of people ready to help, you'll be sorted in no time at all. And if you need an answer on a deadline, you can get commercial support and more at ubuntu.com/support.
Configuring hardware...
Installation Complete
Installation is complete. You need to restart the computer in order to use the new installation.
Restart Now

설치가 완료되면 시스템 재부팅을 해야한다. 재부팅을 하게되면 아래 그림과 같이 설치 이미지파일을 제거하라는 메시지가 나오게 된다.

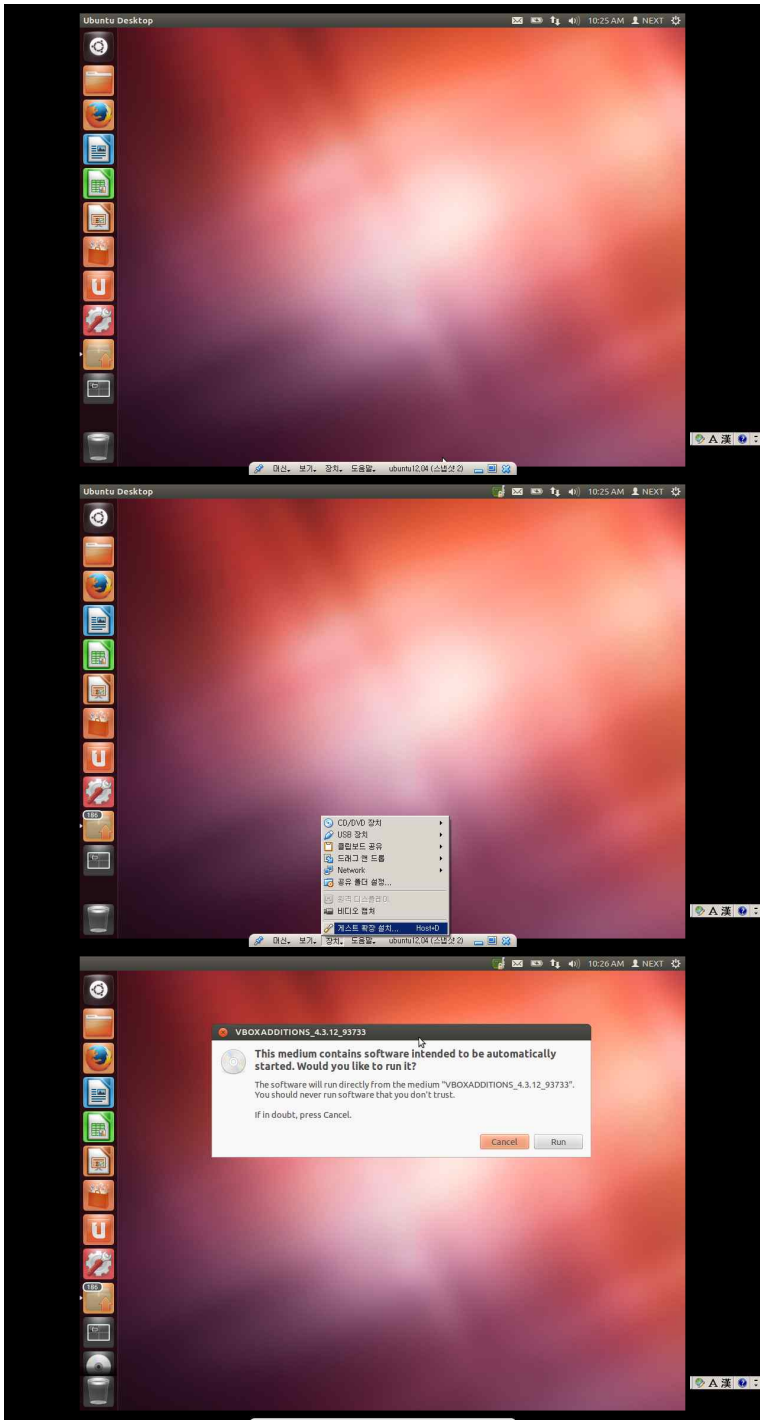


위에 그림과 같이 이미지를 제거하고 재부팅을 진행하면 아래 그림과 같이 우분투가 실행된다.

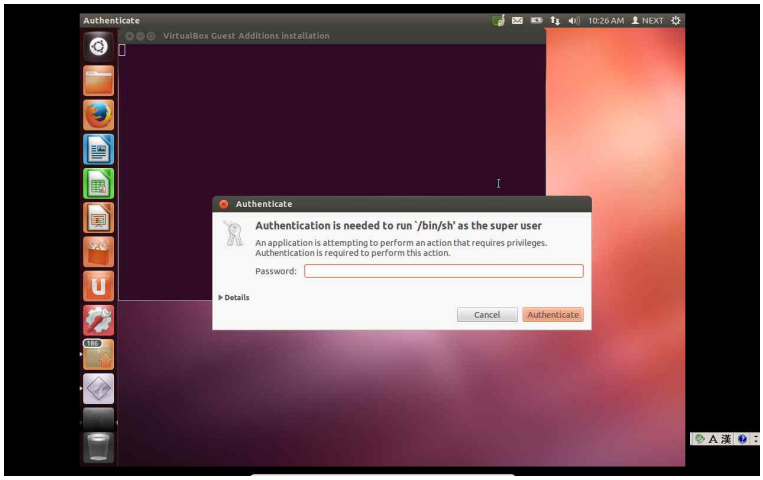


설치 시 입력한 비밀번호를 입력하고 로그인하게 되면 아래와 같은 화면이 나타난다. 화면해상도가 모니터의 실제 해상도보다 낮아 불편할 것이다.

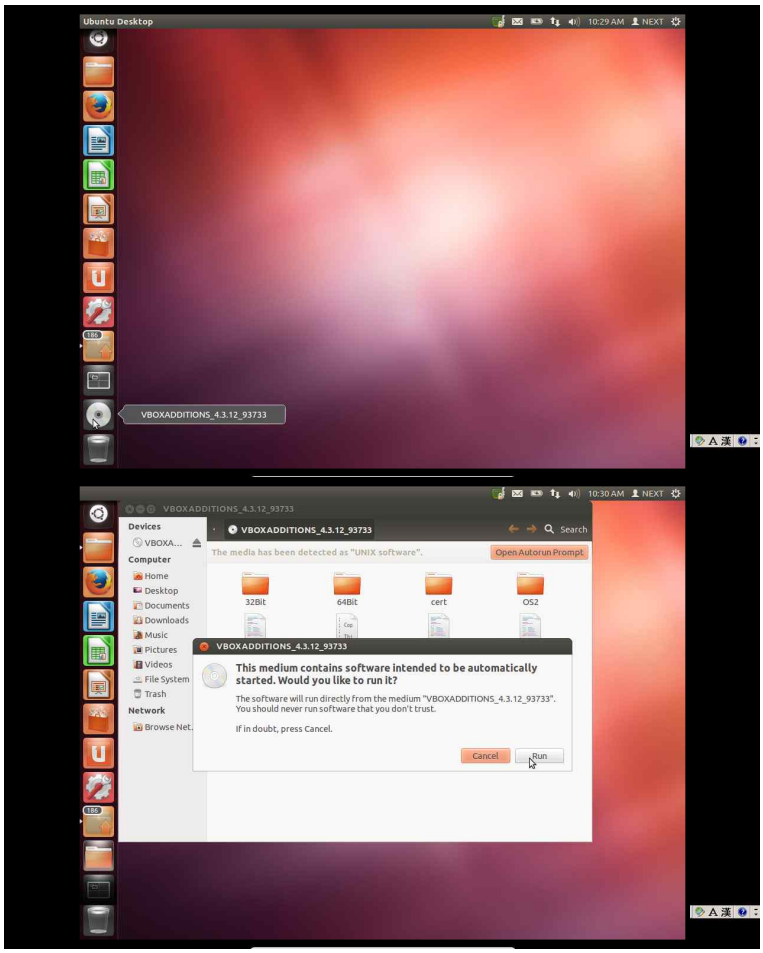
가상장치의 화면 해상도는 VirtualBox의 게스트확장을 설치하면 개선된다. 전체화면모드에서 화면의 중앙 하부에 마우스를 놓고 장치 탭에서 '게스트확장설치'를 클릭하면 우분투 화면에 설치를 물어보는 창이 뜨게 된다. 'Run'을 클릭하면 검정색 terminal(터미널)과 비밀번호를 물어보는 창이 뜨게 된다.

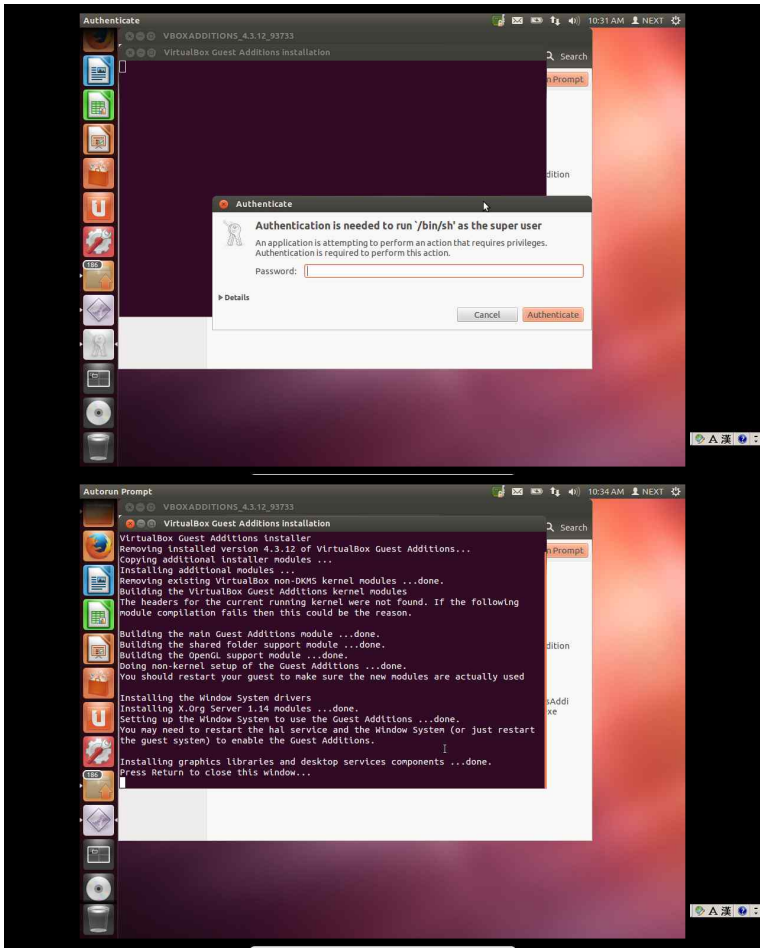


리눅스에서 사용자는 소프트웨어를 설치하는 권한이 없으므로 소프트웨어를 설치하기 위해서는 root(관리자)의 비밀번호를 소프트웨어 설치 시에 입력해 주어야 한다.

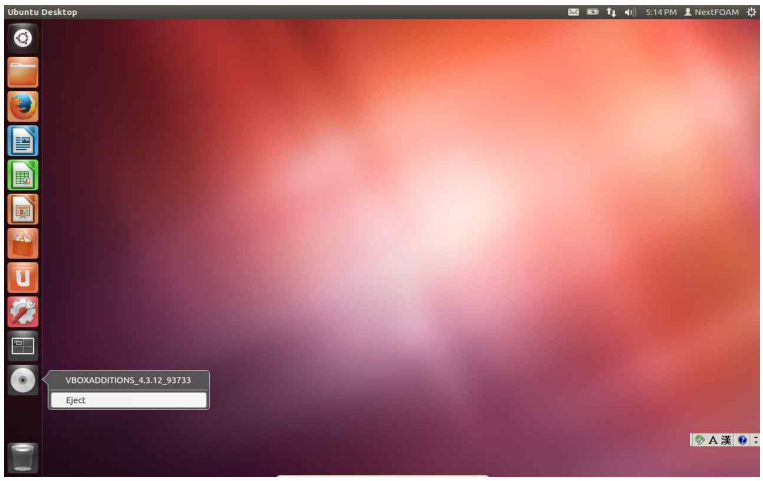


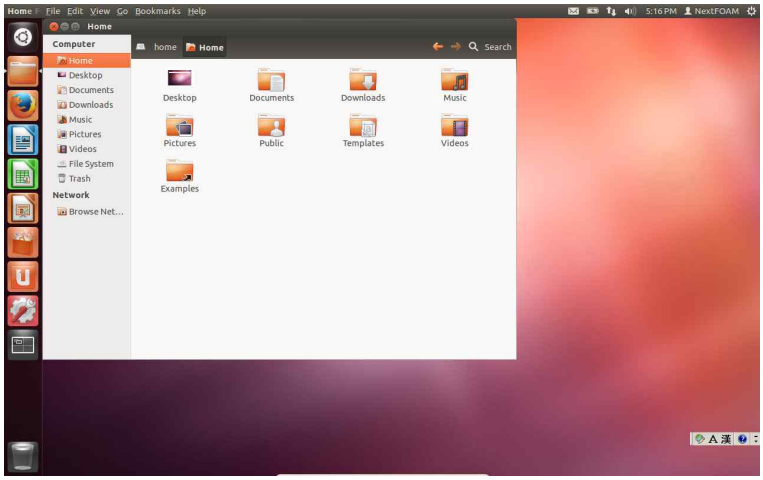
비밀번호를 입력하면 '게스트확장설치'를 위한 프로그램들이 이미지형태로 다운받아진다. 다운로드가 완료되면 아래 그림과 같이 왼쪽 luncher(런처)에 CD모양 아이콘을 볼 수 있다. 클릭하여 실행시킨 후 우측 상단의 'Open Autorun Prompt'를 클릭하면 다운로드 받은 프로그램이 설치된다.



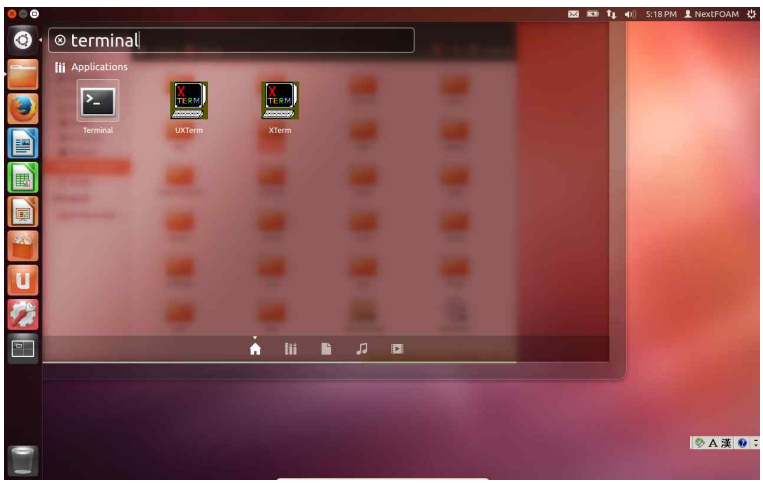


설치 시에 관리자 비밀번호를 한번 더 입력하면 프로그램이 설치된다. 설치 후 리눅스를 재부팅하면 아래와 같이 화면 해상도가 개선된다.



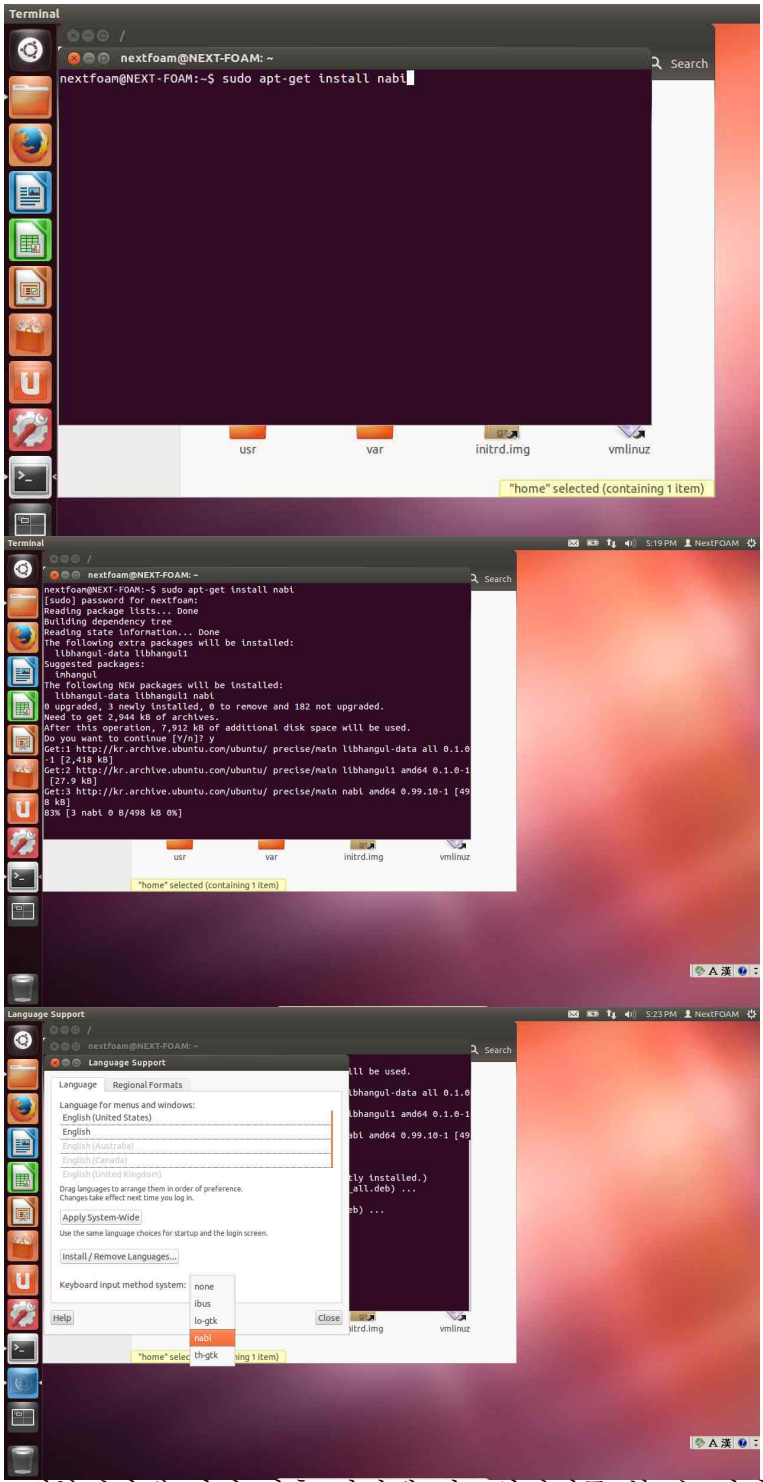


좌측 런처의 상단에 있는 폴더모양을 클릭하면 윈도우의 '내 컴퓨터'를 실행한 것과 같은 창이 열리게 된다. 리눅스에서 사용자는 'Home' 폴더 안에서만 파일 쓰기 권한이 있다. 우분투 설치 시 언어와 키보드를 모두 영어로 선택하였기 때문에 한글을 입력하기 위해서는 한글입력기를 설치해야만 한다. 가장 많이 사용되는 한글 입력기는 'nabi'(나비)이다. 나비를 설치하기 위해서는 윈도우의 명령프롬프트와 같은 terminal(터미널)을 실행시켜야 한다. 런처의 최상단에 있는 'dash home'을 클릭하고 'terminal'을 입력하면 터미널 아이콘이 나타나고, 클릭하여 터미널을 실행할 수 있다.



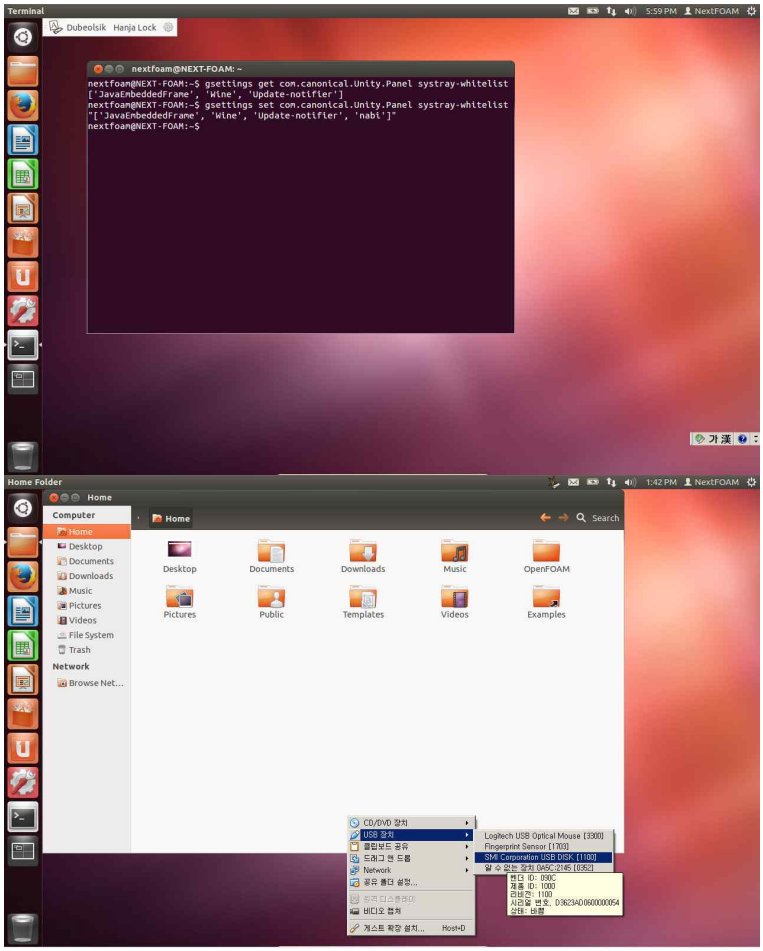
아래 그림과 같이 터미널에 'sudo apt-get install nabi'를 입력하면 비밀번호를 입력하라는 메시지가 터미널에 나타난다. 터미널에 비밀번호를 입력하는 경우는 보안을 위하여 화면에 아무런 표시도 나타나지 않는다. 비밀번호를 입력하면 설치가 진행된다.

설치 완료 후 런처의 'System setting'을 클릭하고 'Language Support'를 클릭하면 아래와 같이 언어를 선택할 수 있는 창이 뜨게 된다. 여기에서 'nabi'를 선택한

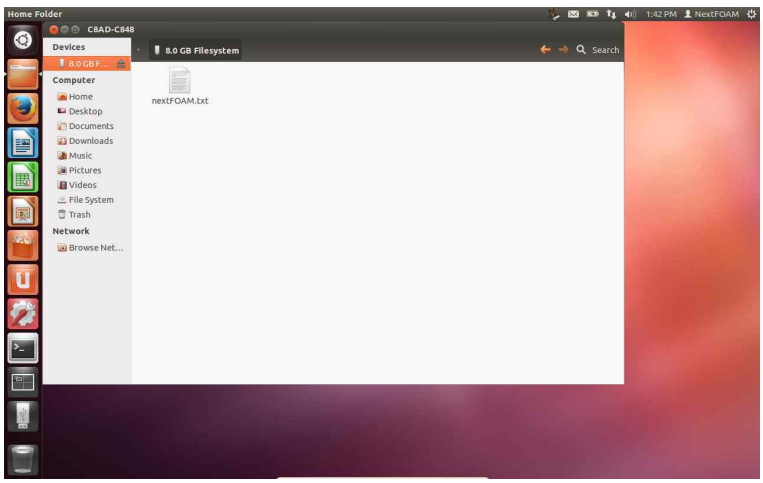


후 재부팅하게 되면 좌측 상단에 한글입력기를 볼 수 있다.

한글입력기가 화면을 가리므로 상단의 패널로 이동 시키려면 그림과 같이 gsettings set com.canonicl.Unity.Panel systray-whitelist "['JavaEmbeddedFrame', 'Wine', 'scp-dbus-service', 'Update-notifier', 'Nabi']" 을 입력 후 재부팅하면 된다.



컴퓨터의 USB를 이용하는 경우 아래 그림과 같이 VirtualBox에서 USB를 선택하면 USB의 내용을 우분투에서 쓰거나 읽을 수 있다.

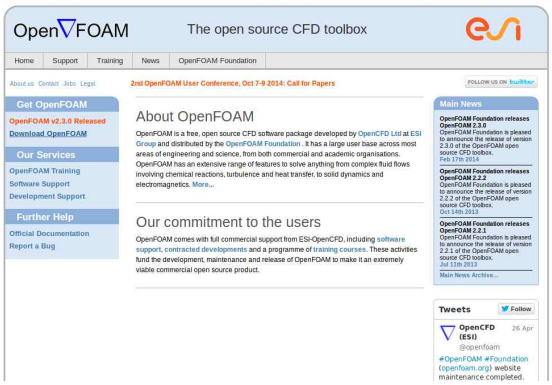


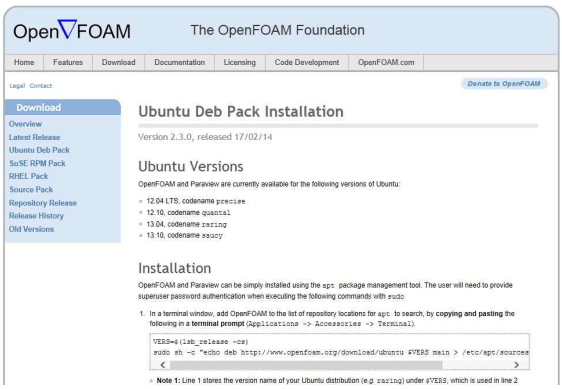
여기까지가 OpenFOAM을 설치하기 위한 환경을 설정한 것이고 다음 절에서는 우분투 리눅스에 OpenFOAM을 설치하는 방법을 설명하겠다.

2. OpenFOAM의 설치 및 구조

2.1 OpenFOAM 설치

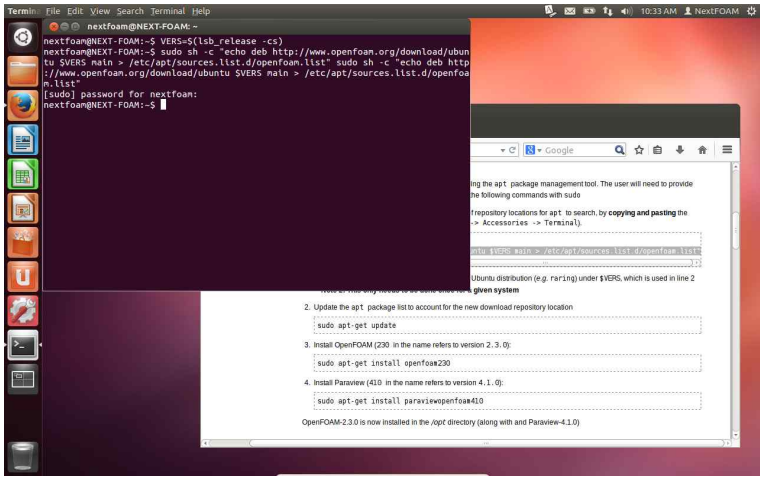
우분투용 OpenFOAM은 'http://www.openfoam.org/download/ubuntu.php'에서 다운 받고 설치 할 수 있다. 본 매뉴얼에서 사용하는 OpenFOAM의 버전은 2.3.0이다.





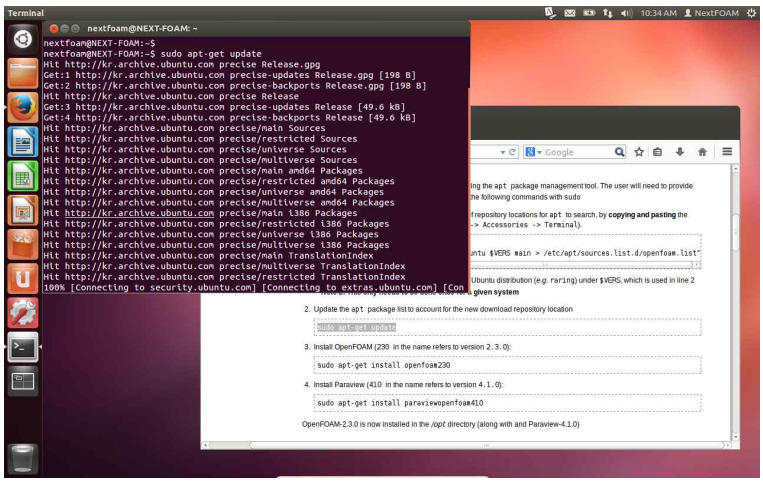
터미널을 실행하고 OpenCFD 홈페이지에 나와 있는 명령어를 차례로 터미널에 입력하면 OpenFOAM을 설치 할 수 있다. 아래의 명령어를 입력하면 OpenFOAM의 설치파일 위치를 컴퓨터에 저장하게 된다. 우클릭하여 '복사', '붙여넣기'를 할 수도 있지만, 리눅스 터미널에서는 조금 다른 방법으로 실행하기도 한다. 복사 할 텍스트를 드래그한 후 붙여 넣을 터미널에서 마우스 휠버튼을 클릭하면 드래그한 텍스트가 입력된다. 터미널이 아니어도 휠버튼을 이용한 텍스트 복사는 가능하다.

```
VERS=$(lsb_release -cs)
sudo sh -c "echo deb http://www.openfoam.org/download/ubuntu $VERS main >
/etc/apt/sources.list.d/openfoam.list"
```



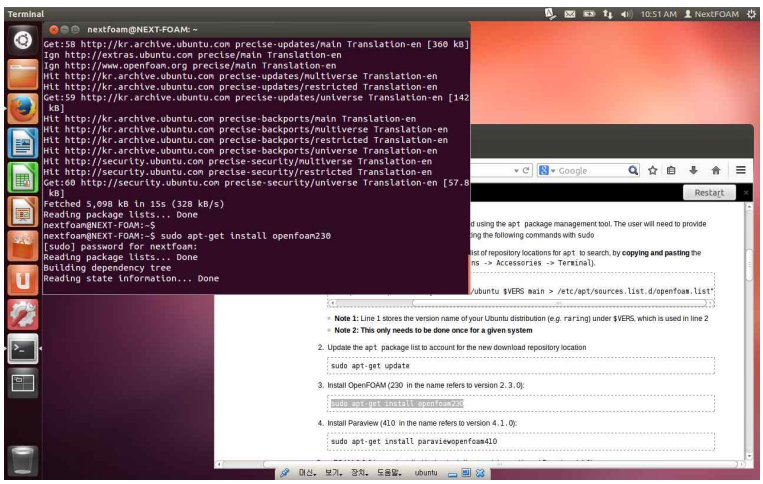
아래의 명령어를 입력하면 그림과 같이 우분투의 update를 체크한다. 필요 시 런처의 'update manager'로 업데이트 할 수 있다.

sudo apt-get update



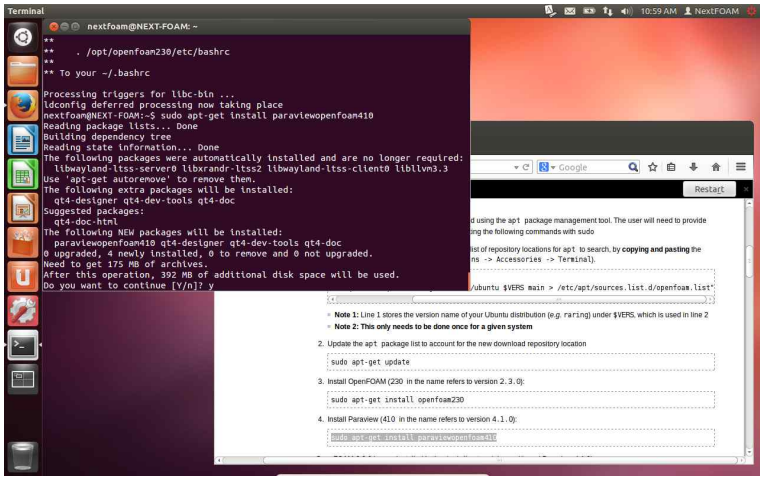
아래 그림과 같이 아래의 명령어를 입력하면 OpenFOAM 설치가 시작된다.

sudo apt-get install openfoam230



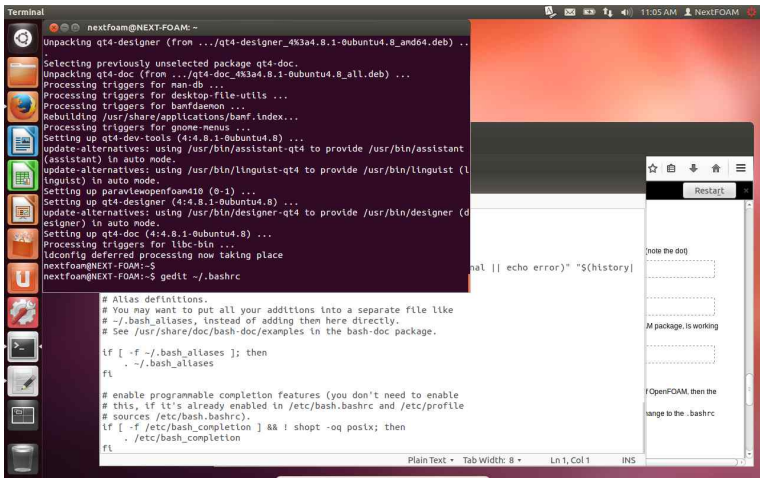
OpenFOAM 설치 후 아래의 명령어를 입력하면 계산결과 가시화에 필요한 paraview(파라뷰)를 설치할 수 있다.

sudo apt-get install paraviewopenfoam410

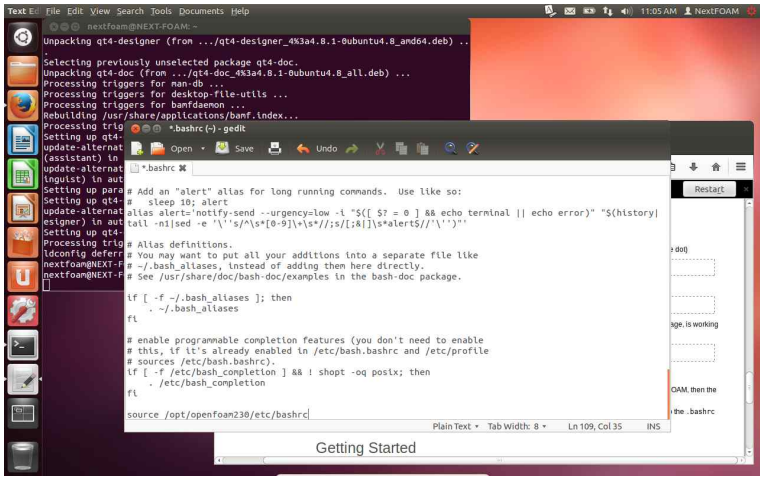


OpenFOAM을 정상적으로 구동 시키기 위해서는 환경설정을 조정해 주워야 한다. 아래의 명령어를 터미널에 입력하면 아래 그림과 같이 환경변수가 저장된 ".bashrc"파일을 gedit라는 프로그램으로 열 수 있다.

gedit ~/.bashrc

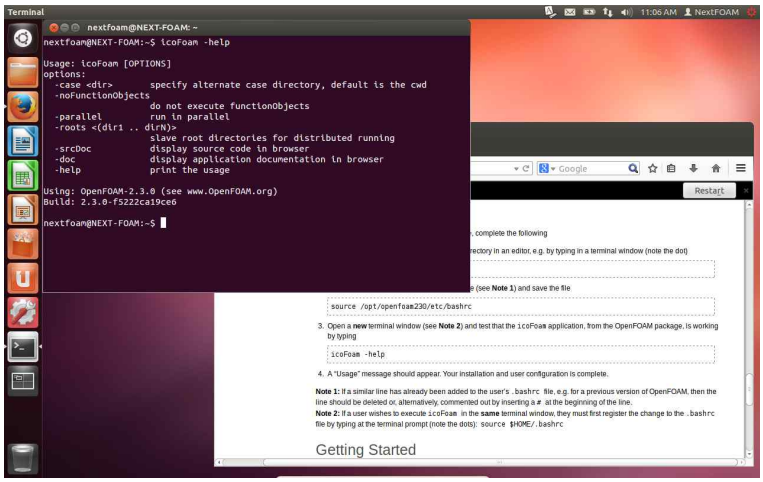


".bashrc"파일의 마지막에 그림과 같이 source /opt/openfoam230/etc/bashrc 을 입력한 후 저장하면 환경 설정도 완료된다.



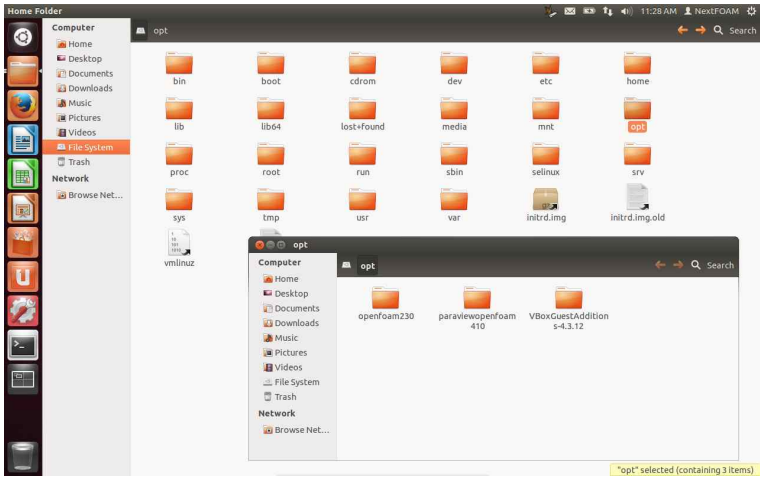
새로 설정한 환경은 새로운 터미널에서부터 적용되기 때문에 새로 터미널을 열어야 한다. OpenFOAM이 정상적으로 설치되었는지 확인하기 위하여 아래의 OpenFOAM 어플리케이션을 실행시켰을 때 그림과 같이 나타나면 정상적으로 설치된 것이다.

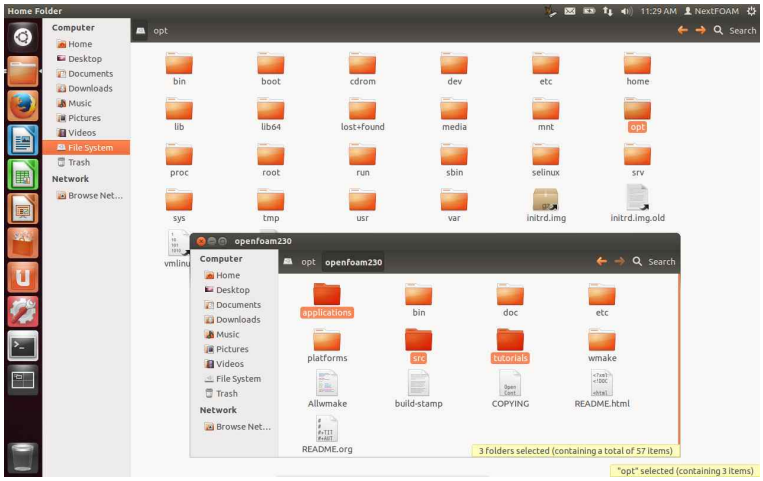
icoFoam -help



2.2 OpenFOAM의 구조 및 리눅스 기본 명령어

설치된 OpenFOAM은 “/opt/openfoam230”에 설치되어있다. OpenFOAM은 앞에서 언급한 것과 같이 C++ 라이브러리와 다양한 application으로 구성되어 있다. C++ 라이브러리의 소스코드는 “/opt/openfoam230/src”에 저장되어 있으며, C++ 라이브러리로 작성한 표준 application의 소스코드는 “/opt/openfoam230/applications”에 저장되어 있다. Tutorial 폴더에는 표준 application을 이용하는 예제들이 저장 되어 있다.



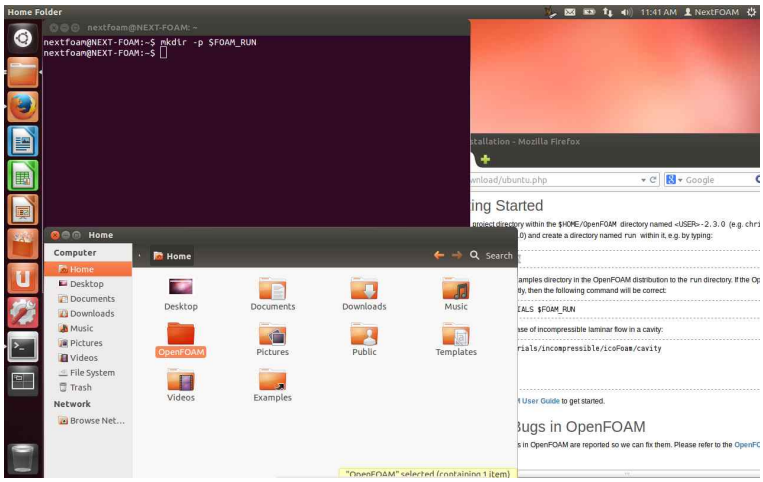


OpenFOAM에서 제공하는 표준 어플리케이션은 다음 그림과 같이 기능에 따라 정리되어 있으며, solver의 경우 어플리케이션의 이름이 Foam으로 끝나는 특징이 있다.


Applications
Solver
basic
laplacianFoam
potentialFoam
scalarTransportFoam
combustion
chemFoam
coldEngineFoam
engineFoam
fireFoam
PDRFoam
reactingFoam
XiFoam
compressible
rhoCentralFoam
rhoPimpleFoam
rhoSimpleFoam
sonicFoam
discreteMethods
dsmc
molecularDynamics
DNS
dnsFoam
electromagnetics
electrostaticFoam
magneticFoam
mhdFoam
financial
financialFoam
heatTransfer
buoyantBoussinesqPimpleFoam
buoyantBoussinesqSimpleFoam
buoyantPimpleFoam
buoyantSimpleFoam
chtMultiRegionFoam
thermoFoam
incompressible
adjointShapeOptimizationFoam
boundaryFoam
icoFoam
nonNewtonianIcoFoam
pimpleFoam
pisoFoam
shallowWaterFoam
simpleFoam
lagrangian
coalChemistryFoam
DPMFoam
icoUncoupledKinematicParcelFoam
reactingParcelFilmFoam
reactingParcelFoam
sprayFoam
uncoupledKinematicParcelFoam
multiphase
cavitatingFoam
compressibleInterFoam
compressibleMultiphaseInterFoam
interFoam
interPhaseChangeFoam
multiphaseEulerFoam
multiphaseInterFoam
potentialFreeSurfaceFoam
settlingFoam
twoLiquidMixingFoam
twoPhaseEulerFoam
stressAnalysis
solidDisplacementFoam
solidEquilibriumDisplacementFoam
utilities
mesh
advanced
conversion
generation
manipulation
miscellaneous
expandDictionary
foamDebugSwitches
foamFormatConvert
foamHelp
foamInfoExec
patchSummary
parallelProcessing
decomposePar
reconstructPar
reconstructParMesh
redistributePar
postProcessing
dataConversion
foamCalc
graphics
lagrangian
miscellaneous
noise
patch
sampling
scalarField
stressField
turbulence
velocityField
wall
preProcessing
applyBoundaryLayer
applyWallFunctionBoundaryConditions
boxTurb
changeDictionary
createExternalCoupledPatchGeometry
dsmcInitialise
engineSwirl
faceAgglomerate
foamUpgradeCyclics
foamUpgradeFvSolution
mapFields
mdInitialise
setFields
viewFactorsGen
wallFunctionTable
surface
surfaceAdd
surfaceAutoPatch
surfaceBooleanFeatures
surfaceCheck
surfaceClean
surfaceCoarsen
surfaceConvert
surfaceFeatureConvert
surfaceFeatureExtract
surfaceFind
surfaceHookUp
surfaceInertia
surfaceLambdaMuSmooth
surfaceMeshConvert
surfaceMeshConvertTesting
surfaceMeshExport
surfaceMeshImport
surfaceMeshInfo
surfaceMeshTriangulate
surfaceOrient
surfacePointMerge
surfaceRedistributePar
surfaceRefineRedGreen
surfaceSplitByPatch
surfaceSplitByTopology
surfaceSplitNonManifolds
surfaceSubset
surfaceToPatch
surfaceTransformPoints
thermophysical
adiabaticFlameT
chemkinToFoam
equilibriumCO
equilibriumFlameT
mixtureAdiabaticFlameT

리눅스에서 사용자는 home 폴더에서만 쓰기 권한이 있기 때문에 '/opt/openfoam230/tutorials'에 있는 예제들을 사용할 수 없다. 따라서 home 폴더의 적당한 위치에 tutorials 폴더를 복사해야 하야 한다. 아래의 명령을 터미널에 입력하면 'home' 폴더에 'OpenFOAM'이라는 폴더가 생성된다. 'OpenFOAM' 폴더에 'nextfoam-2.3.0'폴더가 생성된다. 'nextfoam-2.3.0'은 사용자 이름과 OpenFOAM의 버전을 의미한다. 그리고 그 안에 'run'이라는 폴더가 생성된다.

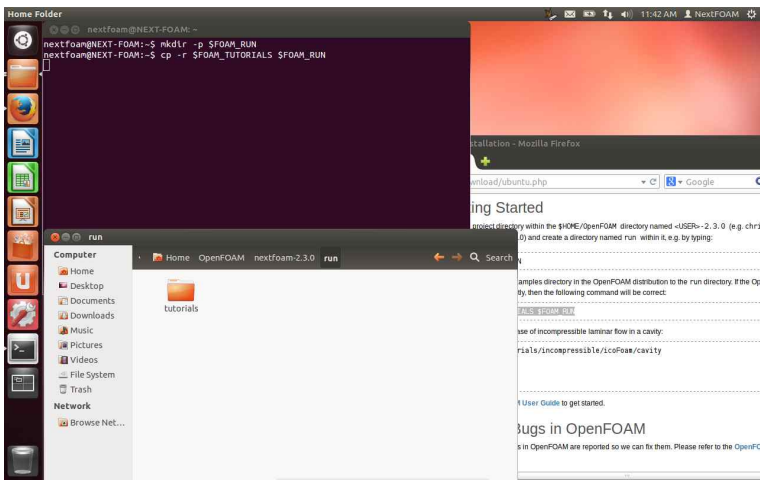
mkdir -p $FOAM_RUN



터미널에 입력한 'mkdir -p $FOAM_RUN'에서 'mkdir'은 make directory를 의미하고 '-p'는 'mkdir'의 옵션으로 '경로 폴더 모두 생성'을 의미한다. '$FOAM_RUN'은 OpenFOAM 환경변수에서 정의된 변수로 '/home/nextfoam/OpenFOAM/nextfoam-2.3.0/run'을 의미한다. 여기서 nextfoam은 사용자 이름이다.

터미널에 아래의 명령어를 입력하면 'tutorials' 폴더의 내용이 'run'폴더로 복사된다.

cp -r $FOAM_TUTORIALS $FOAM_RUN

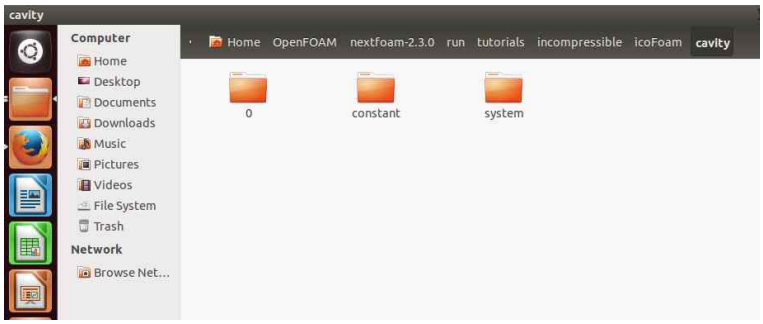


터미널에 입력한 ‘cp –r $FOAM_TUTORIALS $FOAM_RUN’은 ‘$FOAM_TUTORIALS’ 폴더의 하부폴더와 파일을 ‘$FOAM_TUTORIALS’ 폴더로 복사한다는 것을 의미한다. ‘cp’는 복사를 의미하고 ‘-r’은 하위폴더와 파일을 포함한다는 의미이다. $FOAM_TUTORIALS은 ‘/opt/openfoam230/tutorials’을 나타내는 변수이다.

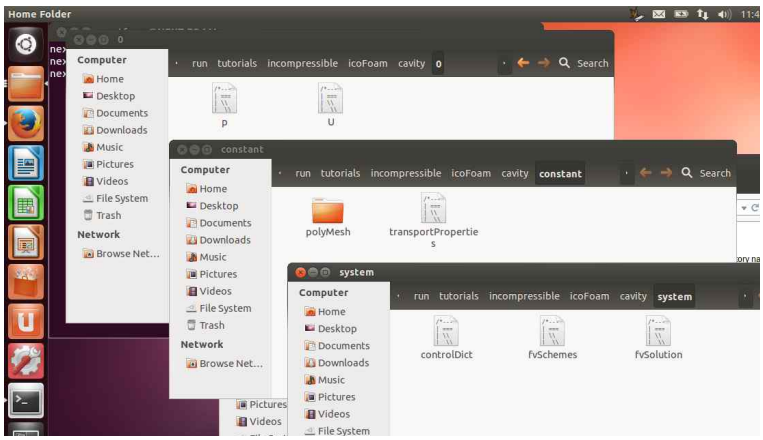
OpenFOAM으로 유동해석을 하기 위해서는 3개의 폴더를 준비해야 한다. 초기조건의 유동장 정보를 갖고 있는 폴더(0 폴더) 그리고 수치기법과 계산조건에 대한 정보를 갖고 있는 폴더(system 폴더), 유체의 물리량과 격자정보를 갖고 있는 폴더(constant 폴더) 이다.

아래 그림은 ‘/Home/OpenFOAM/nextforam-2.3.0/run/tutorials/incompressible/icoFoam/cavity’

폴더를 보여주고 있다. 낮은 레이놀즈 수에서 lead driven cavity문제를 계산하는 예제이다. 세 개의 폴더 ‘0’, ‘constant’, ‘system’을 볼 수 있다.

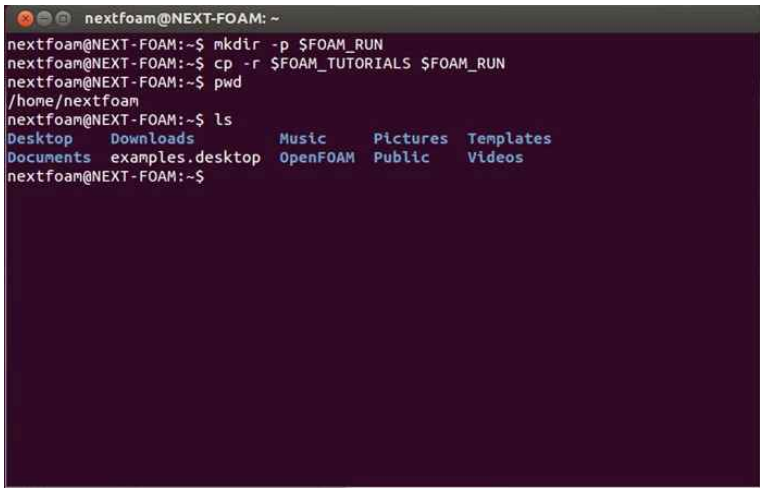


‘0’폴더에는 0초 또는 0번째 time-step(초기상태)에서의 유동장 정보를 갖고 있다. ‘0’폴더에 있는 ‘p’파일은 압력분포를 ‘U’파일은 속도분포에 대한 정보를 갖고 있다. ‘constant’폴더에서 polyMesh’폴더에는 격자계 정보를 갖고 있으며, ‘transProperties’는 유체의 점성계수 정보를 갖고 있다. ‘system’폴더에 있는 ‘controldict’파일은 계산의 시작과 끝, 시간간격, 계산결과의 파일출력 형식 등에 대한 정보를 갖고 있으며 ‘fvScheme’에는 지배방정식의 이산화 방법에 대한 정보가 있고, ‘fvSolution’에는 허용오차와 반복횟수 등에 대한 정보가 있다.



2.2.1 리눅스 기본명령어

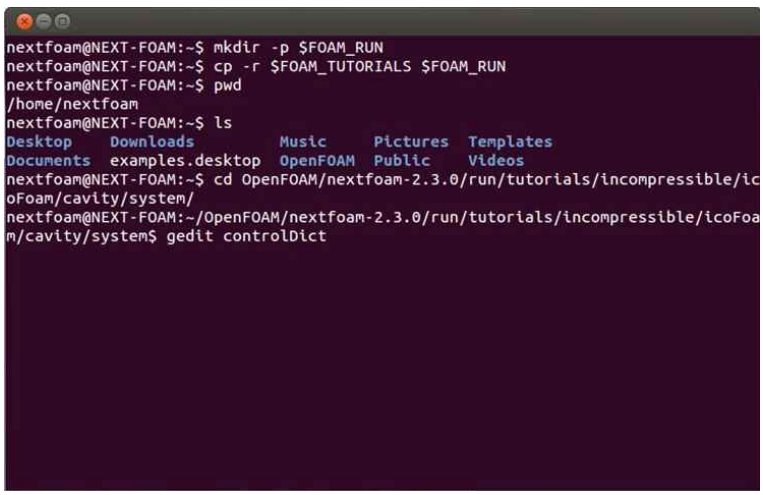
터미널에서 아래 그림과 같이 'pwd'라고 입력을 하면 현재 작업 중인 폴더의 경로를 확인 할 수 있다. 'pwd'는 present working directory의 약자이다. 현재 'Home'폴더의 'nextfoam'폴더 (사용자 폴더)에 있는 것을 볼 수 있다. 터미널에 'ls'라고 입력을 하면 현재 작업 중인 폴더 안의 내용을 화면에 출력시킨다. 화면에 출력된 내용 중 파란색은 폴더를 의미하고 흰색은 파일을 의미한다. 'ls'는 list의 약자이다.



'cd'는 change directory의 약자로 작업폴더를 이동하는 명령어다. 아래와 같이

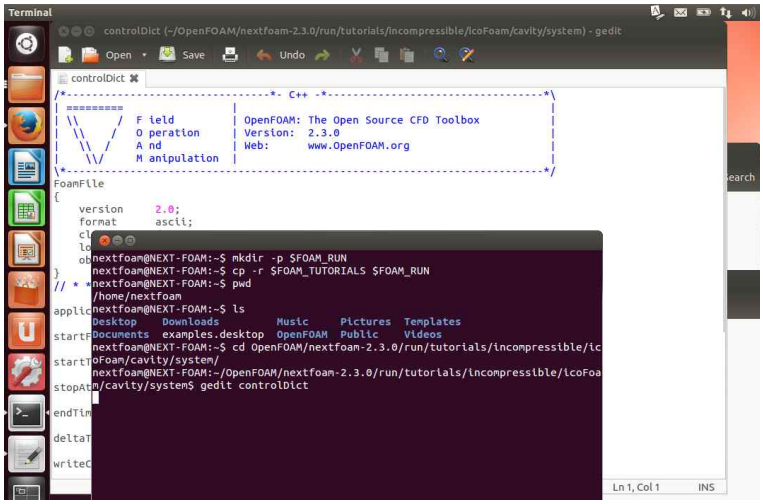
cd OpenFOAM/nextfoam-2.3.0/run/tutorials/incompressible/icoFoam/cavity/system/

을 입력하면 icoFoam cavity예제의 system 폴더로 이동하게 된다. 폴더나 파일 이름을 모두 입력하는 것은 번거롭기 때문에 'Tab'키를 눌러 자동완성을 이용하면 조금 편리하게 입력할 수 있다.

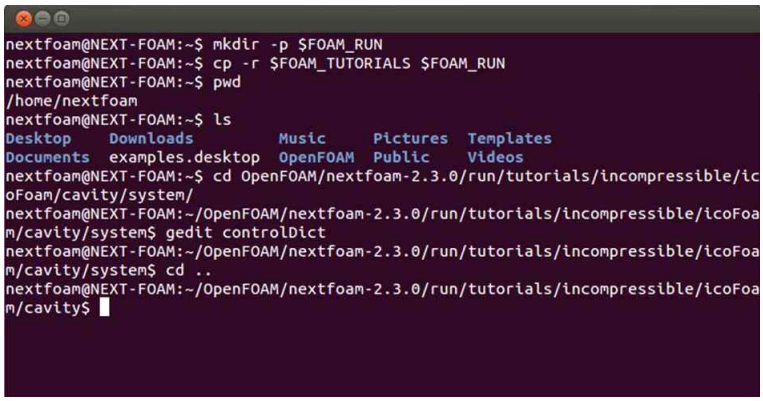


'system'폴더 안의 'controlDict'파일을 열어서 내용을 확인하거나 수정하는 경우 'gedit controlDict'를 입력하면 윈도우의 메모장과 같은 'gedit'라는 프로그램을 이

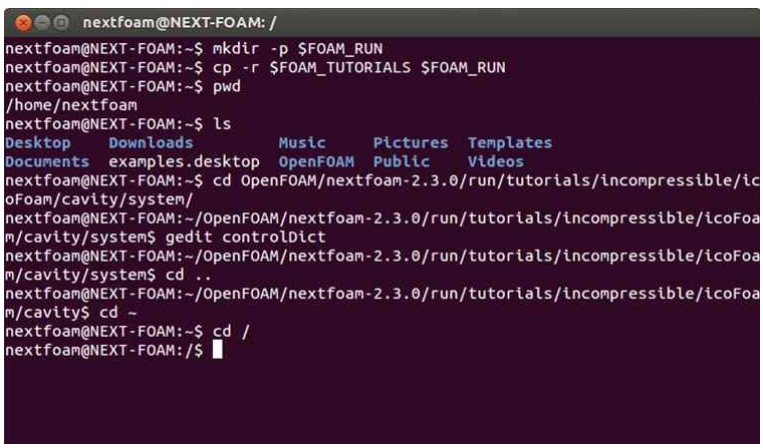
용하여 controlDict파일을 열 수 있다. ‘gedit’가 실행되는 동안에는 터미널에 명령어를 입력할 수 없다.



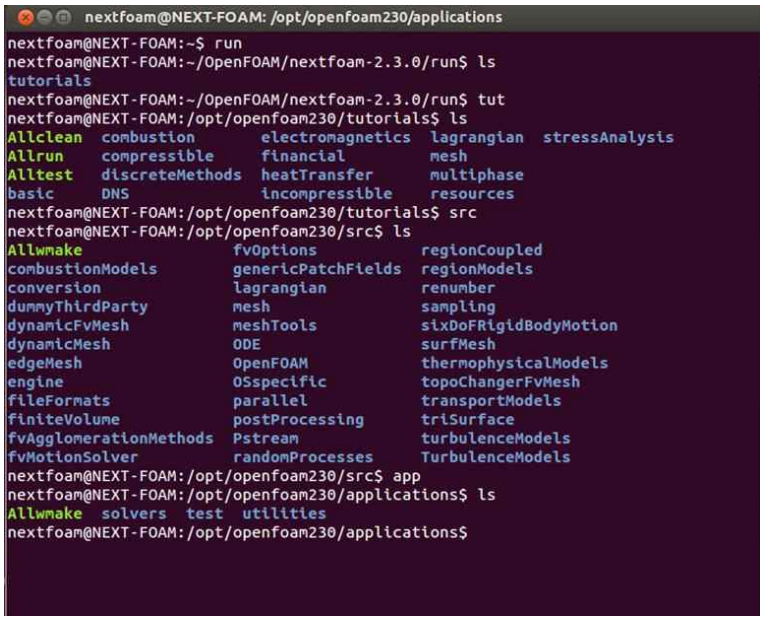
‘gedit’를 종료하고 터미널에 ‘cd ..’을 입력하면 상위 폴더로 이동하는 것을 볼 수 있다. 리눅스에서 ‘..’은 상위폴더를 의미한다.



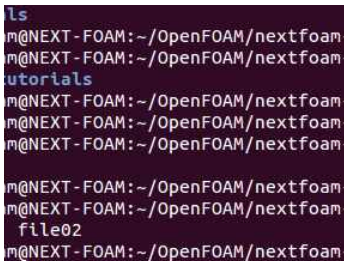
‘cd ~’를 입력하면 ‘Home’폴더로 이동하게 되고, ‘cd /’을 입력하면 ‘root’폴더(리눅스의 최상의 폴더)로 이동하게 된다. 리눅스에서 ‘~’은 ‘Home’폴더을 의미하고, ‘/’은 ‘root’폴더를 의미한다.



OpenFOAM을 설치하게 되면 폴더 이동 시 편리한 몇 가지 명령어(alias)들이 추가되게 된다. 터미널에 ‘run’을 입력하면 OpenFOAM의 ‘run’폴더로 이동하게 되고, ‘tut’를 입력하면 ‘tutorials’폴더로 이동하게 된다. ‘app’를 입력하면 OpenFOAM의 ‘applications’폴더로 이동하게 된다. 아래 그림에서 ‘ls’명령 시 녹색으로 표시되는 내용은 실행파일을 의미한다. ‘Allrun’, ‘Allclean’ 등에 대한 설명은 다음 장에서 설명하게 된다.



폴더를 생성하는 명령어는 ‘make directory’의 약자인 ‘mkdir’이다. 아래 그림과 같이 ‘run’폴더에서 ‘mkdir test’를 입력하여 ‘test’폴더를 생성 할 수 있다. ‘touch’는 빈 파일을 생성하는 명령어 이다. 생성된 ‘test’폴더에서 ‘touch file01’을 입력하여 ‘file01’이라는 파일을 만들 수 있다. ‘copy’의 약자인 ‘cp’를 이용하여 파일 또는 폴더를 복사할 수 있다. ‘cp test01 test02’를 입력하면 ‘test01’파일이 ‘test02’파일로 복사된다. remove의 약자인 ‘rm’ 명령어를 사용하여 파일을 삭제 할 수 있다. 터미널에 ‘rm file02’를 입력하면 ‘file02’파일이 삭제 된다. 폴더를 복사하는 경우 ‘cp’명령어에 ‘-r’이라는 옵션을 붙여 줘야 한다. 터미널에 ‘cp –r test test_cp’를 입력하면 ‘test’폴더가 ‘test_cp’로 복사된다. ‘move’의 약자인 ‘mv’ 명령어를 이용하면 파일이나 폴더를 이동시키거나 이름을 변경할 수 있다. 터미널에 ‘mv test_cp test02’를 입력하면 ‘test_cp’폴더가 ‘test02’폴더로 바뀌게 된다. 파일 삭제 시 파일의 경로를 함께 입력하면 파일이 있는 폴더까지 이동하지 않고 삭제 할 수 있다. 이러한 방법은 다른 명령에도 동일하게 적용된다. 비어있는 폴더의 경우 ‘rmdir' 명령어를 이용하여 삭제 할 수 있지만 폴더 안에 파일이나 폴더가 있는경우 ‘rmdir' 명려어로 삭제 할 수 없다. 폴더와 그 폴더 안의 내용을 동시에 지우기 위해서는 ‘rm’ 명령어와 ‘-r’옵션을 이용하면 된다. 터미널에 ‘rm –r test02’를 입력하면 ‘test02’폴더와 폴더 안의 파일이 모두 삭제된다.



‘test‘ 폴더 생성

‘file01‘ 파일 생성

‘file01’을 ‘file02‘로 파일 복사

‘file02‘ 파일 삭제

‘test‘를 ‘test_cp‘로 폴더 복사

‘test_cp‘를 ‘test02‘로 폴더이름 변경

‘test‘폴더 안의 ‘file01‘파일 삭제

‘test‘폴더 삭제

‘test02‘폴더 삭제 실패

‘test02‘폴더 삭제

3 OpenFOAM applications and utilities

OpenFOAM 을 이용한 예제 실습은 예제 차원에서 끝나는 것이 아니라 solver template를 사용하는 의미가 있다. OpenFOAM은 풀고자 하는 model을 포함한 서로 다른 execution binary를 구동 하여야 하며 각 solver는 서로 다른 형식의 input file과 directory 구조를 요구한다. Tutorial들은 각 solver가 요구하는 요소를 미리 설정해 놓아 향후 사용자가 동일 종류의 solver를 사용하고자 할 때 input template으로서 사용할 수 있다. 이에 본 장에서는 tutorial을 중심으로 하여 solver 및 utility의 구동에 대한 실습을 통해 실제 문제에 OpenFOAM을 적용할 수 있는 방안을 알려주고자 한다.

3.1 icoFoam cavity tutorial

첫 예제는 cavity 유동 해석으로서 OpenFOAM을 구동하기 위해 필요한 준비와 tutorial template을 이용하는 방법에 대해 설명하고자 한다. 우선 solver를 구동하고 이에 쓰인 파일을 다시 review 하는 방법으로 설명을 진행 하고자 한다. OpenFOAM을 설치하고 tutorial을 찾기 위해서는 일반적으로 다음 절차를 따라하면 된다.

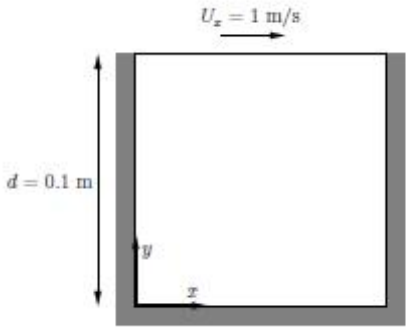
• terminal을 구동하여 사전에 정의된 alias인 tut 명령을 사용하여 $WM_PROJECT_DIR/tutorials folder로 이동한다.

• $WM_PROJECT_DIR/tutorials folder에는 각 solver에 대해 최소한 한 개 이상의 tutorial이 준비되어 있다.

• 이 folder에서 직접 solver를 실행할 경우 불필요한 파일이 생성 되거나 향후 tutorials folder를 template로 사용하기에 곤란한 상황이 벌어질 수 있으므로 필요한 folder를 각 사용자 folder에 복사하여 사용할 것을 권장한다.

3.1.1 tutorial 준비

icoFoam은 OpenFOAM의 가장 간단한 solver중 하나로서 이 solver의 예제를 통하여 OpenFOAM의 기본적인 사용 방법을 알아보기로 한다.



이전에 언급한바와 같이 아래 명령을 통해 tutorial을 사용자의 폴더에 복사 하도록 한다.

```
cp -r $FOAM_TUTORIALS/incompressible/icoFoam/cavity $FOAM_RUN
cd $FOAM_RUN/cavity
```

3.1.2 icoFoam tutorial 구동

tutorial구동을 위해 아래 명령을 실행하여 mesh를 작성한다. 명령을 실행할 때 주의 할 점은 Linux는 OS 특성상 대소문자를 구분하기 때문에 명령에 쓰이는 대소문자를 정확히 써야 한다는 점이다.

blockMesh

mesh 생성 후 아래 명령을 통해 mesh를 check 한다.

checkMesh

명령 실해 후에는 mesh의 수와 geometry의 크기 등을 알 수 있다.
이후 아래 명을을 실행하여 solver를 구동한다.

icoFoam

위 명령을 실행하면 terminal 화면에 계산 진행에 관한 정보가 표시되며 해석이 진행되는 것을 볼 수 있다.
해석이 끝나고 난 후 아래 명령을 실행하면 OpenFOAM의 post-processor인 paraview가 실행된다.

paraFoam

Paraview에서 해석 결과를 확인할 수 있다. (그림 3-1)
자세한 Paraview 사용법은 5.2절 후처리 과정에서 설명되어 있다.

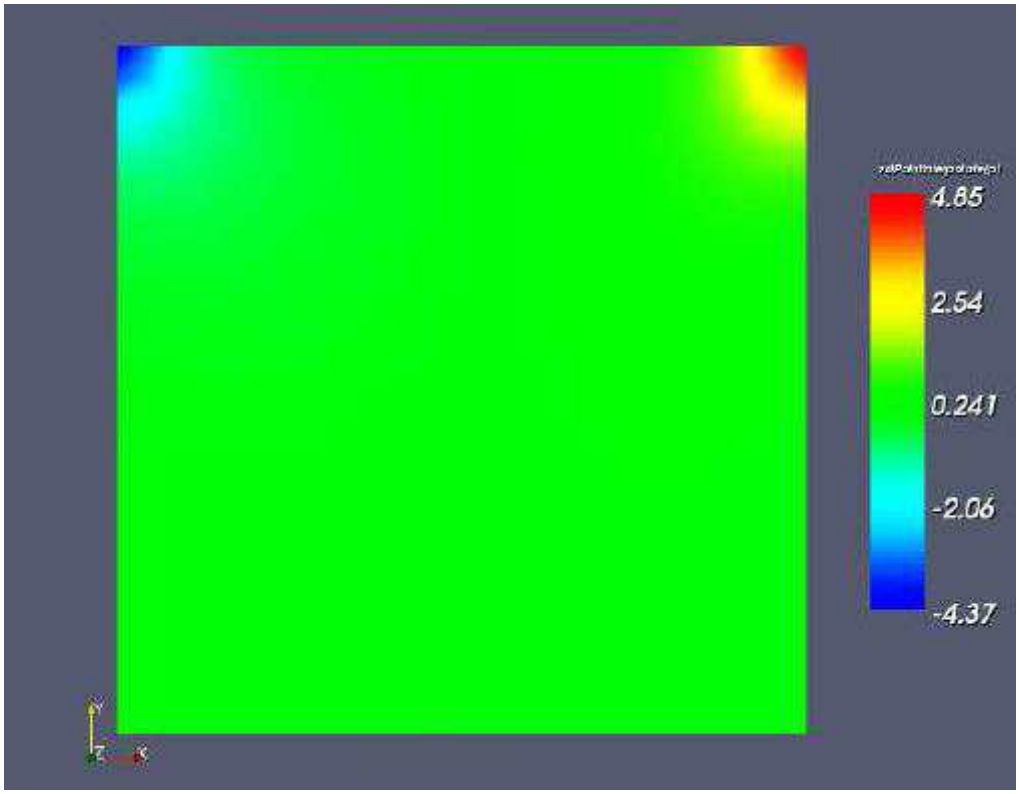


그림 3-1. Paraview를 사용한 cavity 유동 해석 결과

3.1.3 icoFoam input directory 구조

이전에 실행한 icoFoam은 비정상상태, 비압축성, 층류 유동 solver로서 위의 절차를 수행한 후 cavity directory 를 살펴보면 아래와 같은 하부 directory를 볼 수 있다.

0, constant, system, 0.1~0.5

각각의 directory는 다음과 같은 정보를 수록하고 있다.

0 : 유동의 초기 조건과 경계조건 정보

0.1~0.5 : 각 time step의 유동해석 결과 정보

constant : 작동 유체의 동점성 계수 정보

system : 차분법, 행렬 계산법 및 해석의 시작과 종료 등의 정보

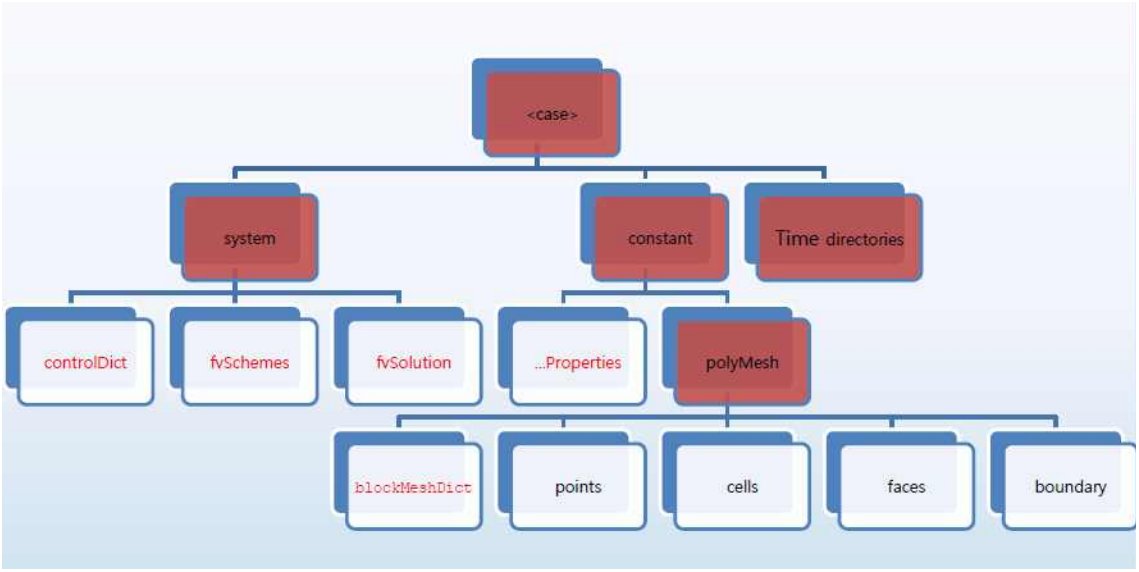


그림 3-2. icoFoam 하부 폴더 구조

3.1.4 icoFoam input file 내용

앞에서 살펴본 각각의 directory에 있는 파일을 하나씩 review하여 solver구동에 필요한 input을 작성하는 방법을 알아보도록 한다.

3.1.4.1 transportProperties

transportProperties 은 constant directory 하단에 있으며 작동 유체의 동점성 계수인 nu를 정의한다.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

nu              nu [ 0 2 -1 0 0 0 0 ] 0.01;
```

```
// ********************************************************* //
```

파일의 1행에서 7행은 의미 없는 헤더이고 8행에서 14행은 파일의 용도에 대해 서술된 header 부분이다. 실재 동점성 계수에 대한 정보는 18행에 기술되어 있다.

3.1.4.2 blockMeshDict

blockMeshDict 파일은 이전에 실행한 blockmesh라는 utility를 실행하기 위한 입력 파일로서 해석영역의 외형 및 boundary 이름 등을 정의하고 있다.

```
/*--------------------------------*- C++ -*----------------------------------*₩
| =========                 |                                                 |
| ₩₩      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  ₩₩    /   O peration     | Version:  2.3.0                                 |
|   ₩₩  /    A nd           | Web:      www.OpenFOAM.org                      |
|    ₩₩/     M anipulation  |                                                 |
₩*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 0.1;

vertices
(
    (0 0 0)
    (1 0 0)
    (1 1 0)
    (0 1 0)
    (0 0 0.1)
    (1 0 0.1)
    (1 1 0.1)
    (0 1 0.1)
);

blocks
```

```
(
    hex (0 1 2 3 4 5 6 7) (20 20 1) simpleGrading (1 1 1)
);

edges
(
);

boundary
(
    movingWall
    {
        type wall;
        faces
        (
            (3 7 6 2)
        );
    }
    fixedWalls
    {
        type wall;
        faces
        (
            (0 4 7 3)
            (2 6 5 1)
            (1 5 4 0)
        );
    }
    frontAndBack
    {
        type empty;
        faces
        (
            (0 3 2 1)
            (4 5 6 7)
        );
    }
);

mergePatchPairs
(
```

);

// ** //

파일의 17행은 향후 기술되는 vertex의 좌표가 실재 해석에서 쓰일 때 SI unit system인 meter 단위로 변환되기 위해 곱해야 할 배수를 나타낸다. 예제와 같이 0.1이라고 되어 있는 경우는 vertex의 좌표가 (1 1 1)로 표시된 경우 실제로는 0.1m, 0.1m, 0.1m위치에 vertex가 위치하는 것을 의미한다.

19행에서 29행은 geometry를 구성하는 vertex의 좌표를 나타낸다. 예제의 경우 총 8개의 꼭지점이 기술된 위치에 놓이며 vertex의 번호는 가장 먼저 쓰인 (0 0 0)이 0번이고 7번까지의 번호가 파일에 쓰여진 순서대로 부여된다.

31행에서 34행은 volume geometry와 mesh를 작성하는 부분이다. 예제의 경우 hex라는 keyword로 육면체를 구성하고 그의 구성은 vertex 순서 0 1 2 3 4 5 6 7를 따르도록 되어 있다. 이때 생성되는 육면체는 아래 그림과 같다.

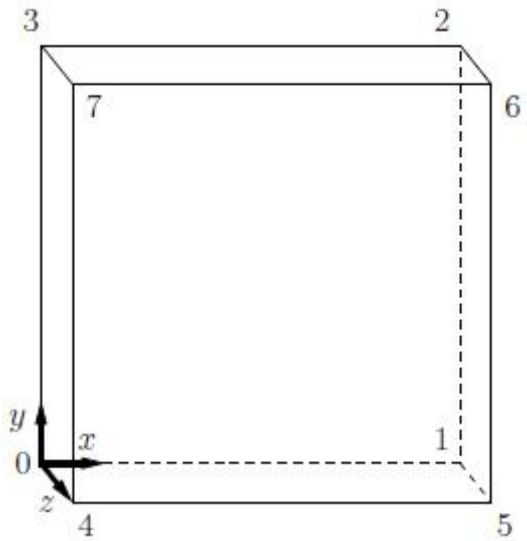


그림 3-3. blockMesh로 생성한 육면체 도메인

이때 vertex의 번호는 오른손 법칙에 따라 쓰여져야 한다.

vertex 순서 이후의 (20 20 1)은 각 edge를 나누는 개수를 의미한다. 첫 번째 20은 vertex 0에서 1로가는 edge를 20개로 나누고 두 번째 20은 vertex 1에서 2로 가는 edge 마지막 1은 vertex 0에서 4로 가는 edge를 1개로 나누는 것을 의미 한다.

그 이후의 simpleGrading (1 1 1)은 각 edge를 나눈 간격을 1배수 즉 동일 간격으로 나누는 것을 의미한다.

boundary는 각 block의 각면에 해당하는 이름과 속성을 나타낸다. cavity 문제는

상자형 공간 상부에 일정 속도를 가했을 때 내부 유동에 대한 해석을 진행하는 예제로 상부 속도를 정의하는 면을 movingWall이라는 이름을 부여하고 이 면의 속성으로 wall 조건을 설정 하였다. 이러한 속성이 부여된 면은 vertex 3, 7, 6, 2로 구성되어 있다. 이 외의 면에 대한 조건도 아래에 나열되어 있다. 특히 이 문제의 경우 2차원 문제이지만 OpenFOAM solver는 2차원 solver가 따로 없는 관계로 깊이 방향에 1개의 cell을 쌓고 양쪽 벽을 empty 조건을 부여하여 해석하게 된다. boundary를 정의하는 vertex는 domain 외부를 향하는 오른손 법칙 순서로 지정하여야 한다.

case directory에서 blockMesh를 실행하면 blockMesh Dict에 서술된 내용을 참조하여 cells, points 등의 mesh에 관련된 정보를 constant/polyMesh 아래에 생성하게 된다.

3.1.4.3 controlDict

system directory 아래에 있는 controlDict file에는 solver 실행시 iteration 횟수, 자동 저장 간격 등 해석 진행에 관한 일반적인 내용이 기술되어 있다.
file의 내용을 살펴보면 다음과 같다.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application icoFoam;
startFrom startTime;
startTime 0;
stopAt endTime;
endTime 0.5;
deltaT 0.005;
writeControl timeStep;
writeInterval 20;
purgeWrite 0;
```

```
writeFormat ascii;
writePrecision 6;
writeCompression uncompressed;
timeFormat general;
timePrecision 6;
runTimeModifiable yes;
// ************************************************************************* //
```

file의 header 부분 이후의 내용은 다음과 같다.

application 은 해당 controlDict를 참조하는 solver의 이름이다. 본 예제의 경우는 icoFoam을 이용하여 해석을 수행하게 된다. 이후 startFrom과 startTime은 해석이 시작되는 조건에 대한 설명으로 예제의 경우 startTime에 기술된 0 초부터 해석을 시작하게 된다. stopAt과 endTime은 해석이 종료되는 시점을 의미하여 0.5 초에 해석을 종료하도록 설정되어 있다. deltaT는 unsteady 계산시 time step size를 나타낸다. writeControl은 해석 수행도중 자동 저장 간격을 어떤 단위로 나타낼지를 결정하며 timeStep으로 되어있는 경우 writeInterval에 지정된 time step 개수에 맞춰 해석 결과가 자동 저장되게 된다. purgeWrite는 해석이 진행되는 동안 과도한 개수의 해석 결과가 자동 저장되는 것을 방지하기 위해 일정 개수의 결과만 남기고 이전의 결과는 지우게 하기 위해 설정하지만 예제와 같이 0 으로 설정된 경우는 모든 저장 결과를 모두 남겨두게 된다. writeFormat은 해석 결과 저장시 용량을 줄이고 저장 시간을 단축하기 위해 binary로 저장 하거나 해석 결과를 text editor에서 바로 확인할 수 있도록 ascii로 저장할 수 있도록 선택할 수 있다. 해석 결과의 크기를 좀더 줄여서 저장 공간을 압축하기 위해서 writeCompression을 적용할 수 있으나 본 예제의 경우 uncompressed로 지정되어 해석 결과를 plane text format으로 저장하게 된다. timeFormat과 time Precision을 이용하여 해석 결과를 저장할 때 시간에 대한 정보를 지수형 혹은 소수형으로 지정할 수 있다. runTimeModifiable은 해석이 진행되는 도중 controlDict 파일을 수정하여 해석을 중단 하거나 해석이 종료되는 시간을 연장할 수 있도록 해석 진행 도중에 system folder의 변경을 감시하는 기능을 켜거나 끌 수 있는 기능이다.

contorlDict에 쓰인 key word의 경우는 이전에 쓰인 항목을 이용하면 되지만 값의 경우는 변경할 수 있다. 하지만 어떤 값을 사용할 수 있는지 궁금할 경우 값을 쓰는 자리에 dummy라는 값을 넣어주면 그 위치에 넣을 수 있는 추가적인 옵션을 볼 수 있다. 이를테면 stopAt에 쓸 수 있는 값의 종류가 궁금하다면 endTime이 쓰여진 자리에 dummy를 적어주고 solver를 구동하면 아래와 같은 메시지를 볼 수 있다.

```
dummy is not in enumeration
4
(
```

```
nextWrite
writeNow
noWriteNow
endTime
)
```

즉 dummy의 위치에 사용할 수 있는 값은 위의 4개의 값이 가능하다는 의미이다. dummy는 어느 위치에서도 작동이 가능하며 위와 유사한 형식으로 dummy가 쓰여 있는 위치에서 사용 가능한 옵션들을 보여준다.

3.1.4.4 fvSchemes

fvSchemes 파일은 solver가 풀고자하는 equation의 discretization scheme을 설정하는 역할을 한다. 파일 안에서는 각 equation의 항 별로 서로 다른 scheme을 적용할 수 있으며 이 file에서도 controlDict에서 사용했던 dummy를 사용할 수 있다.

3.1.4.5 fvSolution

fvSolution은 해석 대상이 되는 equation의 matrix solver를 선택하는 역할을 한다. 또한 solver algorithm에 대한 parameter도 설정할 수 있다.

3.1.4.6 0 directory

예제에 제공되는 0 directory에는 해석의 initial condition과 boundary condition이 서술되어 있다. directory 안에는 p와 U file이 있고 이는 각각 압력과 속도에 대한 값을 포함하고 있다. 이들중 U 파일을 살펴보면 다음과 같다.

```
dimensions [0 1 -1 0 0 0 0];
internalField uniform (0 0 0);
boundaryField
{
        movingWall
        {
                type fixedValue;
                value uniform (1 0 0);
        }
        fixedWalls
        {
                type fixedValue;
                value uniform (0 0 0);
        }
```

```
        frontAndBack
        {
                type empty;
        }
}
```

파일의 가장 윗줄에 있는 dimensions는 해당 파일에서 나타낸 값의 단위를 의미하며 다음과 같이 [질량 길이 시간 온도 몰수 전류량 광량] 의 지수를 나타내며 단위는 SI unit의 경우 [kg m s K mol A cd]을 사용한다. 즉 [0 1 -1 0 0 0 0]은 m의 1제곱 * s의 -1제곱을 의미하여 m/s를 나타낸다.

다음줄의 internalField는 cell value를 의미한다. 예제의 0 folder는 initial condition으로서 domain 내부 모든 cell이 동일하게 정지 상태를 가정하기 위해 uniform (0 0 0);으로 x, y, z 속도가 모두 0으로 설정되어 있다. 해석이 진행되어 0 directory에 추가로 time directory가 생성되면 cell 마다 속도가 다른 값을 갖게 되므로 internalField 부분이 변경되게 된다.

boundaryField는 속도에 대한 boundary condition으로서 해석 영역의 상부에 x방향 1m/s 나머지 벽면은 non-slip condition으로 지정되어 있다. 이전에 설명한 바와 같이 OpenFOAM에서 2차원 문제 해석을 위해서는 깊이방향으로 1층의 cell을 만들게 되고 추가적으로 만들어진 벽면에는 empty라는 type을 지정하면 추가적인 설정이 필요하지 않다.

3.1.5 해석 실행 및 검토

3.1.4에서 살펴본 반와 같이 적정한 위치에 각 파일 작성이 끝난 후 3.1.2의 절차에 따라 utility와 solver를 실행하면 cavity 예제에 대한 실습이 종료 된다.

3.2 회전체 주위 난류유동 해석

회전체 주위의 유동을 해석하는 방법은 물체의 회전을 MRF(Multiple Reference Frame) 기법을 사용하여 정상상태 해석을 하는 방법과 실제 물체를 움직이면서 비정상상태 해석을 하는 방법이 있다. 본 절에서는 OpenFOAM에서 두 가지 방법을 이용하여 해석하는 방법을 소개한다.

3.2.1 임펠러 주위 정상상태 유동 해석

MRF 기법을 이용한 임펠러 주위 정상상태 유동해석을 위해서는 fvOptions 기능을 이용하여 회전영역에 MRF source를 입력하는 방법을 사용한다.

fvOptions는 OpenFOAM v2.2.0에서 처음 도입된 방법으로 소스 코드의 수정 없이 지배방정식에 소스항이나 특정값의 제한(constraint)을 줄 수 있도록 개발된 프레임웍이다. fvOptions를 이용하면 MRF, porous media, actuation disk, fixed temperature, tempertature limit constraint 등을 구현할 수 있다.

fvOptions 기능을 사용하여 MRF 모델을 적용할 때 모든 설정은 3.1절의 비압축성 난류 유동 해석 부분의 내용과 동일하며 단지 system 폴더의 fvOptions 파일에 회전하는 cellzone, 회전축, 회전중심, 각속도 조건을 입력해 주기만 하면 된다.

incompressible/simpleFoam/mixerVessel2D 튜토리얼 문제를 이용해서 임펠러 주위 정상상태 유동 해석 방법을 알아보자. 아래 그림은 튜토리얼 문제의 형상과 격자를 나타내었다. 내부에 회전하는 ‘rotor’가 있고 외부에 정지 상태의 ‘stator’가 있다. 회전하는 영역은 ‘rotor’라는 이름의 cellZone이다. ‘rotor’는 반시계 방향으로 1000RPM의 속도로 회전한다. 유동의 유출입은 없으며 비압축성이다.

standard k-epsilon 난류 모델을 사용하고 simpleFoam solver를 사용한다

3.2.1.1 경계조건 - ‘0’ 폴더 설정

‘0’ 폴더에 필요한 파일은 U, p 파일과 난류 모델에 따라 필요한 파일이다. 이 문제에서는 standard k-epsilon 난류모델을 사용할 것이므로 k와 epsilon 파일이 필요하다.

각 파일에는 초기값과 모든 경계면들에 대한 조건이 주어져야 한다. 경계면은 rotor, stator, front, back 네 개 이다. rotor는 회전하는 경계면, stator는 정지 상태의 경계면이다. front, back은 2차원 계산을 위한 경계면으로 모든 파일에서 ‘empty’ 조건을 사용하면 된다.

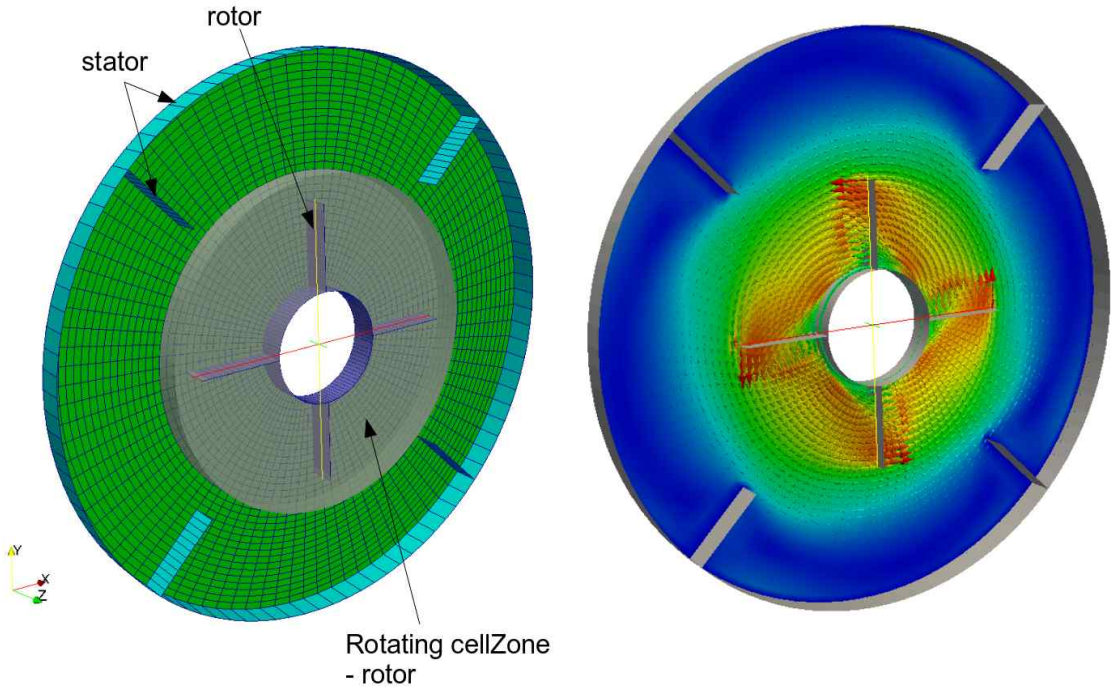


그림 3-4. 임펠러 주위 유동 해석 예제의 격자 및 속도장

- 속도 경계조건, U

U 파일은 다음과 같이 설정된다.

```
/                                                                                    /
*************************************************************************************
//
dimensions      [0 1 -1 0 0 0 0];
internalField   uniform (0 0 0);
boundaryField
{
    rotor
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    stator
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
...
```

```
}
/                                                                                    /
*************************************************************************************
//
```

dimension은 m/s 단위로 설정하고 초기 속도는 (0 0 0)으로 설정한다.
stator는 정지 상태이기 때문에 no-slip조건을 적용하여 fixedValue (0 0 0)을 사용한다.
rotor는 회전하는 면이지만 회전하는 영역('rotor'라는 cellZone) 내부에 있는 면이기 때문에, 회전좌표계 상에서는 정지상태의 면이다. 따라서 속도 조건은 fixedValue (0 0 0)을 사용한다.

- 압력 경계조건, p

p 파일은 다음과 같이 설정된다.

```
/                                                                                    /
*************************************************************************************
//
dimensions      [0 2 -2 0 0 0 0];
internalField   uniform 0;
boundaryField
{
    rotor
    {
        type            zeroGradient;
    }
    stator
    {
        type            zeroGradient;
    }
...
}
/                                                                                    /
*************************************************************************************
//
```

비압축성 유동 solver인 simpleFoam을 사용하기 때문에 압력 p는 밀도로 나누어진 값이며 단위는 m2/s2이 된다. 초기 압력은 0으로 설정한다. rotor와 stator 모두 zeroGradient 조건을 사용한다.

- 난류 경계조건, k

k 파일은 다음과 같이 설정된다.

```
// ********************************************************************************* //
dimensions      [0 2 -2 0 0 0 0];
internalField   uniform 1;
boundaryField
{
    rotor
    {
        type                kqRWallFunction;
        value       $internalField;
    }
    stator
    {
        type                kqRWallFunction;
        value       $internalField;
    }
  ...
// ********************************************************************************* //
```

- 난류 경계조건, epsilon

epsilon 파일은 다음과 같이 설정된다.

```
// ********************************************************************************* //
dimensions      [0 2 -3 0 0 0 0];
internalField   uniform 20;
boundaryField
{
    rotor
    {
        type                epsilonWallFunction;
        value       $internalField;
    }
    stator
    {
        type                epsilonWallFunction;
        value       $internalField;
    }
...
}
// ********************************************************************************* //
```

3.2.1.2 물성값 및 난류모델, 'constant' 폴더 설정

'constant' 폴더에 필요한 파일은 RASProperties, transportProperties 두 개이다.

RASProperties 파일에서는 난류 모델을 설정해 주고, transportProperties 파일에서는 물성값을 설정해 준다.

난류 모델을 standard k-epsilon 모델을 사용할 때 RASProperties 파일은 다음과 같다.

```
// ************************************************************************* //
RASModel        kEpsilon;
turbulence      on;
printCoeffs     on;
// ************************************************************************* //
```

각 난류 모델에 대한 RASModel의 설정 값은 $WM_PROJECT/src/turbulenceModels/incompressible/RAS/ 에서 확인할 수 있다.

비압축성 문제에서 설정해 주어야 하는 물성값은 동점성계수 뿐이다. 동점성계수는 transportProperties 파일에서 설정해 준다. 우선 transportModel에서 뉴턴유체인지 아닌지를 설정한다. 뉴턴유체이면 Newtonian으로 설정하고 동점성계수를 상수로 입력한다. 비뉴턴유체인 경우 transportModel을 CrossPowerLaw 혹은 BirdCarreau 로 설정하고 그에 따라 CrossPwerLawCoeffs나 BirdCarreauCoeffs를 설정해 준다.

```
// ************************************************************************* //
transportModel  Newtonian;
nu              nu [ 0 2 -1 0 0 0 0 ]  1e-05;
// ************************************************************************* //
```

압축성 유동의 경우 점성계수는 Sutherland 모델을 사용하여 온도의 함수로 줄 수 있으며 thermodynamicProperties 파일에서 설정해 준다.

3.2.1.3 수치해석 기법 및 계산조건, 'system' 폴더 설정

'system' 폴더에는 controlDict, fvSolution, fvScheme 파일이 있어야 하고 필요에 따라 여러 가지 파일이 추가로 필요하게 된다. 이 문제에서는 회전체의 운동을 고려하기 위해 fvOption 파일이 추가로 필요하다.

- controlDict

controlDict 파일은 각종 계산 조건을 설정하는 파일이다. 이 예제에서 controlDict 파일은 다음과 같이 설정되어 있다.

```
// ************************************************************************* //
application     simpleFoam;
```

```
startFrom       startTime;
startTime       0;
stopAt          endTime;
endTime         500;
deltaT          1;
writeControl    timeStep;
writeInterval   50;
purgeWrite      0;
writeFormat     ascii;
writePrecision  6;
writeCompression off;
timeFormat      general;
timePrecision   6;
runTimeModifiable true;
// ************************************************************************* //
```

- fvSolution

fvSolution 파일은 각 변수의 행렬연산자, SIMPLE 알고리즘, 완화계수를 설정하는 파일이다.

SimpleFoam solver를 사용할 때는 solvers, SIMPLE, relaxationFactors라는 세 가지 dictionary로 구성된다. 이 예제에서는 다음과 같이 설정되어 있다.

```
// ************************************************************************* //
solvers
{
    p
    {
        solver          GAMG;
        tolerance       1e-08;
        relTol          0.05;
        smoother        GaussSeidel;
        cacheAgglomeration      true;
        nCellsInCoarsestLevel 20;
        agglomerator    faceAreaPair;
        mergeLevels     1;
    }
    "(U|k|epsilon)"
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        nSweeps         2;
        tolerance       1e-07;
        relTol          0.1;
    }
```

```
}
SIMPLE
{
    nNonOrthogonalCorrectors    0;
    pRefCell                    0;
    pRefValue                   0;
}
relaxationFactors
{
    fields
    {
        p                       0.3;
    }
    equations
    {
        U                       0.5;
        "(k|epsilon)"           0.5;
    }
}
// ************************************************************************* //
```

이 문제는 유동의 유출입이 없는 close domain 문제이기 때문에 SIMPLE dictionary에 기준 압력을 설정해 주어야 한다. 기준 위치를 설정하는 방법은 위의 예와 같이 pRefcell을 이용해 cell number를 주는 방법과 pRefPoint를 이용해 좌표를 주는 방법이 있다. 기준 값은 pRefValue를 이용해 설정한다.

- fvSchemes

fvSchems 파일은 수치해석 기법을 설정하는 파일이다. 이 예제에서는 다음과 같이 설정되어 있다.

```
// ************************************************************************* //
ddtSchemes
{
    default                 steadyState;
}
gradSchemes
{
    default                 Gauss linear;
}
divSchemes
{
    default                 none;
    div(phi,U)              bounded Gauss limitedLinearV 1;
```

```
    div(phi,k)              bounded Gauss limitedLinear 1;
    div(phi,epsilon)        bounded Gauss limitedLinear 1;
    div((nuEff*dev(T(grad(U)))))    Gauss linear;
}
laplacianSchemes
{
    default                 Gauss linear corrected;
}
interpolationSchemes
{
    default                 linear;
}
snGradSchemes
{
    default                 corrected;
}
fluxRequired
{
    default         no;
    p               ;
}
// ************************************************************************* //
```

- fvOptions

fvOptions은 MRF, Porous media 등의 설정과 각종 소스항 및 각종 값의 제한을 설정하기 위한 파일이다. 임의의 이름을 갖는 dictionary에 필요한 값을 설정한다. MRF 설정을 위해서는 type을 MRFSource로 설정하고 회전영역과 회전중심, 회전축, 각속도를 입력한다. 각속도는 [radians/sec] 단위를 사용하고 반시계방향이 + 값을 갖는다.

이 문제에서는 회전축이 z이고 1000 RPM이기 때문에 axis는 (0 0 1)이 되고 omega는 104.72 radians/sec 가 된다.

```
// ************************************************************************* //
MRF1
{
    type                    MRFSource;
    active                  true;
    selectionMode           cellZone;
    cellZone                rotor;

    MRFSourceCoeffs
    {
        origin              (0 0 0);
```

```
        axis            (0 0 1);
        omega           104.72;
    }
}
// ************************************************************************* //
```

MRF dictionary에서 selectionMode는 points, cellSet, cellZone, mapRegion, all 등의 값을 쓸 수 있다.

3.2.2 임펠러 주위 비정상상태 유동 해석

회전체의 운동을 Sliding mesh 기능을 사용하여 격자를 움직여 구현하는 방법에 대해 알아보자.

Sliding mesh 기능을 사용하기 위해서는 회전하는 영역과 정지 상태의 영역 사이에 각 영역에 해당되는 interface면이 존재해야 한다. OpenFOAM에서는 이런 인터페이스 면을 AMI(Arbitrary Mesh Interface)면이라 한다.(OpenFOAM Extend Project 버전에서는 GGI(General Grid Interface)라고 한다.) 예를 들어 회전 영역에 속해 있는 AMI-1이라는 경계면이 있다면 정지 영역에 속해 있는 AMI-2라는 경계면이 같은 위치에 존재하고 두 면이 인터페이스임을 constant/polyMesh/boundary 파일에서 지정되어야 한다.

MRF 조건을 사용할 때 회전 조건을 fvOptions 기능을 사용하여 주었는데 sliding mesh에서는 dynamicMeshDict 파일을 이용해서 설정한다. 그리고 dynamic mesh 기능을 사용할 수 있는 비정상상태 solver를 사용한다는 것을 제외하면 대부분의 설정은 MRF를 사용하는 경우와 동일하다.

비정상상태 해석은 정상상태 해석에 비해 상당히 많은 시간이 소비된다. 실수에 의한 계산 설정의 오류를 막기 위해 유동 계산 없이 격자의 운동만을 먼저 확인한 다음 유동 계산을 시작하는 것이 효율적이다. 이런 경우에는 설정이 완료된 다음 “moveDynamicMesh” 유틸리티를 실행하면 된다.

아래그림은

$WM_PROJECT/tutorial/incompressible/pimpleDyMFoam/mixerVesselAMI2D

문제이다. 3.2.1의 예제 문제에 Sliding mesh를 적용한 것이다. 이 예제를 사용하여 사용법을 알아보자. solver는 pimpleDyMFoam을 사용한다.

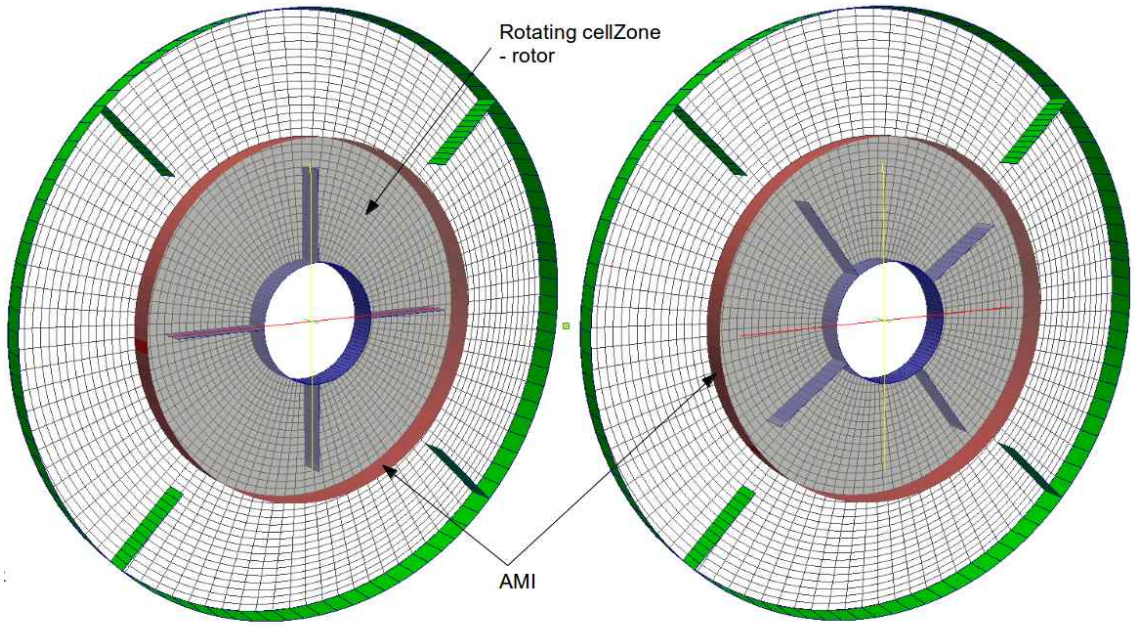


그림 3-5. 임펠러 유동의 비정상 해석을 위한 계산 영역 설정

3.2.2.1 격자의 설정

회전하는 영역과 정지 영역 사이에 두 개의 면이 존재 한다면 두 면이 인터페이스 면이라는 것은 constant/polyMesh/boundary 파일에서 설정한다. 아래의 예와 같이 AMI1과 AMI2라는 이름의 면이 인터페이스라면 neighbourPatch를 대응되는 면의 이름으로 설정해 주면 된다. 상황에 따라 matchTolerance를 조절해 줘야 할 때도 있다.

```
// ************************************************************************* //
...
    AMI1
    {
        type            cyclicAMI;
        inGroups        1(cyclicAMI);
        nFaces          96;
        startFace       6240;
        matchTolerance  0.0001;
        transform       noOrdering;
        neighbourPatch  AMI2;
    }
    AMI2
    {
        type            cyclicAMI;
        inGroups        1(cyclicAMI);
```

```
        nFaces          96;
        startFace       6336;
        matchTolerance  0.0001;
        transform       noOrdering;
        neighbourPatch  AMI1;
    }
...
// ************************************************************************* //
```

준비된 격자 파일이 회전하는 영역은 cellZone으로 지정되어 있으나 3.2.1의 예제와 같이 두 영역 사이에는 경계면이 없는 경우가 있다. 이 경우에는 createBaffle 유틸리티를 이용하여 AMI 면을 생성할 수 있다. CreateBaffle 유틸리티를 사용하기 위해서는 createBafflesDict 파일에서 작동 조건을 설정해야 한다. 다음은 createBafflesDict의 설정 예이다.

```
// ************************************************************************* //
internalFacesOnly true;
baffles
{
    impeller
    {
        type            faceZone;
        zoneName        inter;
        patches
        {
            master
            {
                name            AMI1;
                type            cyclicAMI;
                matchTolerance  0.0001;
                neighbourPatch  AMI2;
                transform       noOrdering;
            }
            slave
            {
                name            AMI2;
                type            cyclicAMI;
                matchTolerance  0.0001;
                neighbourPatch  AMI1;
                transform       noOrdering;
            }
        }
    }
}
```

```
// ************************************************************************* //
```

위의 예는 회전 영역과 정지 영역 사이에 'inter'라는 이름의 faceZone이 있을 때 이를 이용해서 AMI1과 AMI2라는 두 개의 AMI 면을 만드는 것이다. 위와 같이 설정하고 createBaffle 유틸리티를 실행하면 constant/polyMesh/boundary 파일의 설정까지 완성 된다.
위의 예에서 'inter'라는 faceZone도 없다면 topoSet 유틸리티를 이용해서 faceZone을 만들어 주어야 한다.

3.2.2.2 경계조건, '0' 폴더의 설정

'0' 폴더에 필요한 파일은 3.2.1의 MRF 예제와 동일하게 U, p, k, epsilon이다. AMI 면들의 경계조건 type은 모든 파일에서 cyclicAMI 조건을 사용한다.
p, k, epsilon 파일은 새로 추가된 AMI1, AMI2 두 개의 면을 cyclicAMI로 설정한 것 외에는 3.2.1의 MRF 예제와 동일하다.
U 파일에서 회전하는 면의 속도는 movingWallVelocity 조건을 사용한다. 정지하고 있는 면에도 movingWallVelocity 조건을 그대로 사용하여도 no-slip 조건인 fixedValue (0 0 0)과 같은 조건이 된다. 다음은 U 파일의 설정이다.

```
// ************************************************************************* //
    rotor
    {
        type            movingWallVelocity;
        value           uniform (0 0 0);
    }

    stator
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
// ************************************************************************* //
```

3.2.2.3 'constant' 폴더의 설정

'constant' 폴더에 필요한 파일은 3.2.1의 MRF 예제의 RASProperties, transportProperties 파일 외에 turbulenceProperties, dynamicMeshDict 파일이 필요하다.
turbulenceProperties 파일은 solver에 따라 필요한 경우도 있고 아닌 경우도 있다. simulationType을 설정해 주는데 laminar, RASModel, LES 등을 설정할 수 있다.
dynamicMeshDict 파일은 다양한 동적 격자계의 운동을 설정하는 파일이다. 우선

강체 운동(rigid body motion)인지 6자유도 운동인지 등을 설정하고 그에 따라 필요한 설정 값을 준다. 회전체의 운동은 solidbodyMotionFvMesh으로 설정하고 solidBodyMotionFVMeshCoeffs를 다음과 같이 설정한다.

```
// ********************************************************************************** //
dynamicFvMesh               solidBodyMotionFvMesh;
motionSolverLibs ( "libfvMotionSolvers.so" );
solidBodyMotionFvMeshCoeffs
{
    cellZone                rotor;
    solidBodyMotionFunction    rotatingMotion;
    rotatingMotionCoeffs
    {
        origin              (0 0 0);
        axis                (0 0 1);
        omega              6.2832;
    }
}
// ********************************************************************************** //
```

위 예에서 설정된 것은 'rotor'라는 cellZone이 origin, axis, omega에 지정된 회전 중심, 회전 축, 각속도의 조건으로 움직인다는 것이다.

3.2.2.3 'system' 폴더의 설정

'system' 폴더에는 3.1.1의 예제와 같이 controlDict, fvSolution, fvSchemes 파일이 필요하며 fvOptions 파일은 별도의 소스항이나 제한 조건이 없다면 필요 없다.

- controlDict

pimpleDyMFoam은 비정상상태 solver이기 때문에 3.1.1의 예제와 다른 설정이 몇 가지 필요하다. 우선 time step size를 설정해 주어야 하는데 일정한 값을 주는 방법과 adjust time step기법을 사용하는 방법이 있다. Adjust time step은 max Courant number를 설정하고 이 값이 넘지 않도록 time step size를 조절하면서 계산하는 방식이다.

Adjust time step을 사용할 때 writeControl에 대해 adjustableRunTime을 사용하게 되면 설정한 시간 간격마다 데이터를 저장할 수 있다.

이 예제에서 controlDict 파일은 다음과 같이 설정되어 있다. Max Courant number가 0.5를 넘지 않는 time step size를 사용하며 5초까지 계산하는 동안 0.1초 간격으로 데이터를 저장하는 조건이다.

```
// ********************************************************************************** //
application     pimpleDyMFoam;
startFrom       startTime;
startTime       0;
```

```
stopAt          endTime;
endTime         5;
deltaT          1e-3;
writeControl    adjustableRunTime;
writeInterval   0.1;
purgeWrite      0;
writeFormat     ascii;
writePrecision  6;
writeCompression        off;
timeFormat      general;
timePrecision   6;
runTimeModifiable       true;
adjustTimeStep  yes;
maxCo           0.5;
// ************************************************************************* //
```

- fvSolution

fvSolution 역시 비정상상태 solver이기 때문에 3.1.1의 예제와 다른 설정이 몇 가지 필요하다.

3.1.1에서는 'solvers' dictionary에서 p, U, k, epsilon의 서브 dictionary를 설정했는데 여기서는 이와 더불어 pcorr, p, pFinal, UFinal, kFinal, epsilonFinal, cellMotionUx에 대한 서브 dictionary를 설정해 주어야 한다.

simpleFoam에서 설정해 주었던 SIMPLE dictionary 대신 PIMPLE dictionary를 지정해 주어야 한다. PIMPLE dictionary의 설정은 다음과 같다.

```
// ************************************************************************* //
    correctPhi              no;
    nOuterCorrectors        1;
    nCorrectors             2;
    nNonOrthogonalCorrectors 0;
    pRefCell                0;
    pRefValue               0;
// ************************************************************************* //
```

- fvSchemes

fvSchemes에서는 ddtScheme을 'Euler' 혹은 'backward'로 설정하고 fluxRequired dictionary에 pcorr을 추가해 주면 된다.

3.2.3 주기 조건을 사용한 임펠러 주위 유동 해석

회전체의 운동을 계산할 때 계산 용량을 줄이기 위해 회전 날개 하나에 대해서만

계산 격자를 생성하고 주기(cyclic 혹은 periodic) 경계조건을 사용하는 경우가 많이 있다. OpenFOAM에서 주기 경계조건을 사용하는 방법에 대해 알아보자

아래의 그림은 네 개의 회전 날개가 있는 2차원 문제에서 하나의 날개에 대해서만 격자를 생성한 예이다.

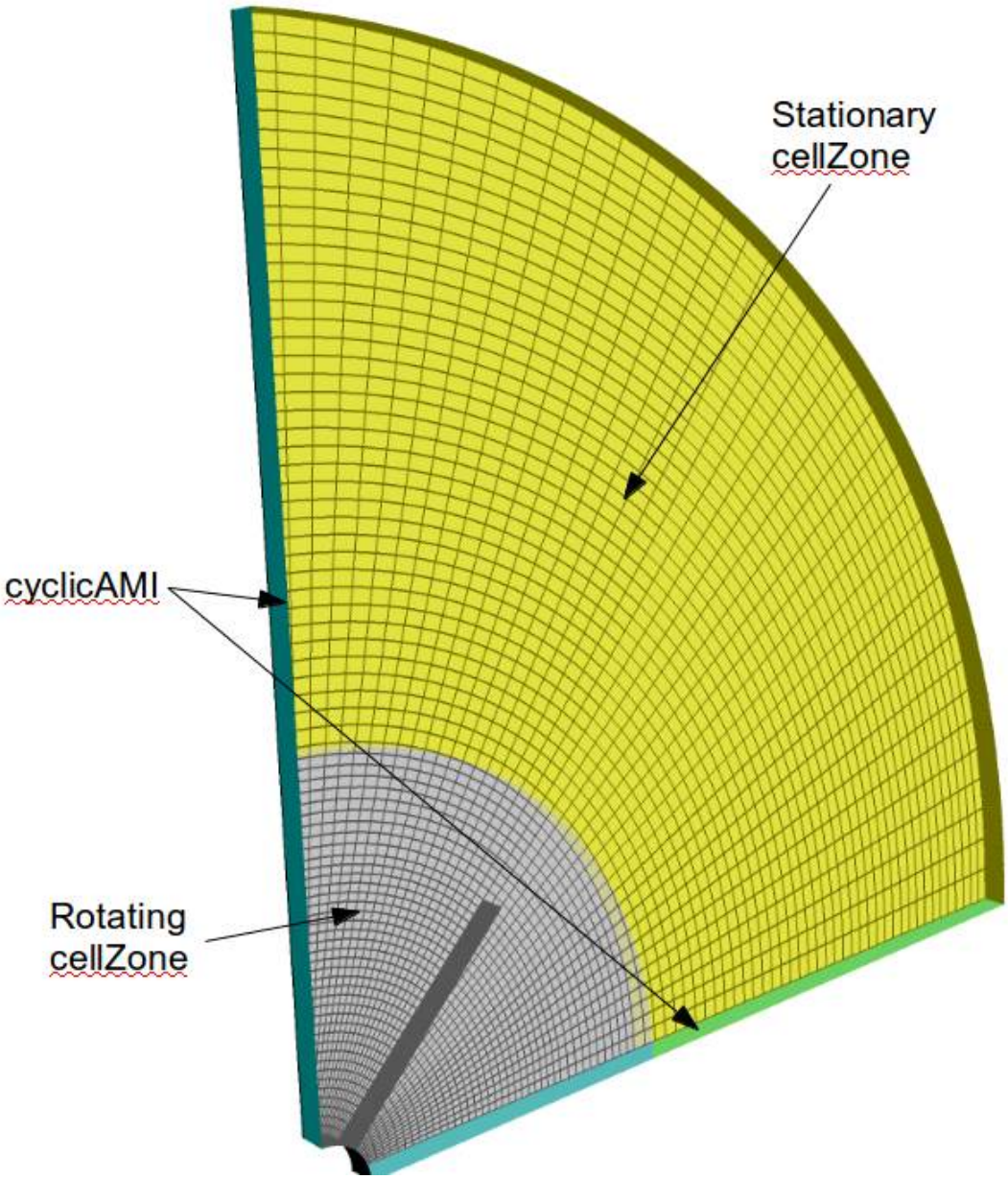


그림 3-6. 주기 조건을 적용하기 위해 계산 영역을 분할한 경우

OpenFOAM에서는 주기 경계조건을 cyclic이라고 부른다. Cyclic 경계조건은 대응되는 두 개의 면에서 격자가 완전히 동일할 때 사용 가능한 조건이다. 그러나 실제

문제에서 대응 되는 두 면의 격자를 완전히 동일하게 제작하는 것은 어려운 일이기 때문에 다른 격자를 갖는 경우에도 사용할 수 있는 조건이 필요하다. 대응 되는 두 면의 격자가 다르기 때문에 두 면은 인터페이스로 처리되어야 하므로 cyclicAMI 조건을 사용할 수 있다.
cyclicAMI 조건은 sliding mesh를 사용할 때 회전 영역과 정지 영역 사이의 인터페이스 처리에 사용한 조건과 같은 조건이며, 설정 내용은 조금 다르다.
위 예제에서 두 주기면의 이름이 cyclic1, cyclic2라고 하면 ../constant/polyMesh/boundary 파일에서 두 면의 설정은 다음과 같다.

```
// ************************************************************************* //
...
    cyclic1
    {
        type            cyclicAMI;
        nFaces          3190;
        startFace       250778;
        matchTolerance  0.0001;
        transform       rotational;
        neighbourPatch  cyclic2;
        rotationAxis    (0 0 1);
        rotationCentre  (0 0 0);
        rotatingAngle   90;
    }
    cyclic2
    {
        type            cyclicAMI;
        nFaces          3190;
        startFace       253968;
        matchTolerance  0.0001;
        transform       rotational;
        neighbourPatch  cyclic1;
        rotationAxis    (0 0 1);
        rotationCentre  (0 0 0);
        rotationAngle   90;
    }
...
// ************************************************************************* //
```

transform에 대한 설정값을 rotational로 주고, rotationAxis와 rotationCenter를 좌표로 입력해 준다. rotationAngle은 주기면이 평면이 아닐 경우에만 필요하며 각도(degree) 값을 입력해 준다.
MRF 소스는 마찬가지로 fvOptions 파일에서 설정하는데 두 개의 주기면을 회전하

지 않는 조건으로 설정해 주어야 한다. 이것은 다음과 같이 nonRotatingPatches를 설정하면 된다.

```
// ************************************************************************************ //
...
MRF
{
    type MRFSource;
    active          true;
    selectionMode           cellZone;
    cellZone        rot;
    MRFSourceCoeffs
    {
        nonRotatingPatches          (cyclic1 cyclic2);
        origin                  (0 0 0);
        axis                    (0 0 1);
        omega                   5;
    }
}
// ************************************************************************************ //
```

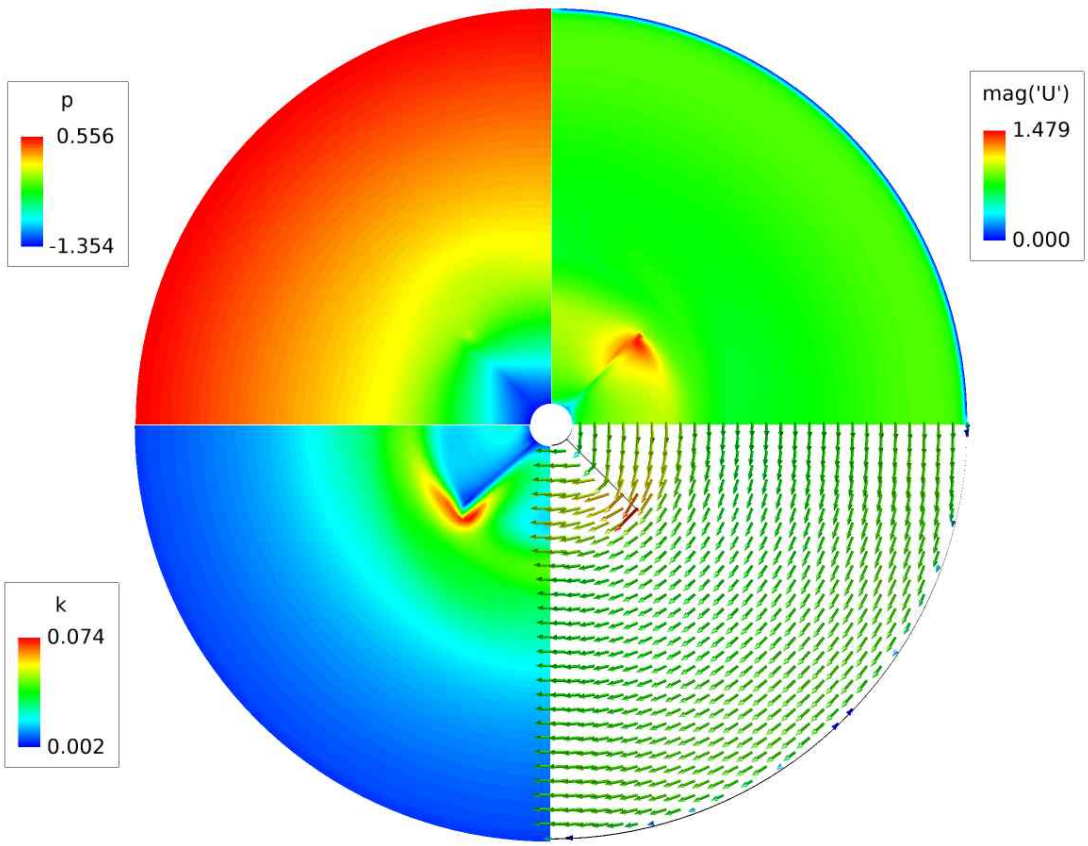


그림 3-7. 임펠러 주위 유동 해석 결과

3.3 열전달 해석

저속 유동의 열전달 해석을 위해 OpenFOAM이 제공하는 표준 solver(standard solver)는 다음과 같다.

BuoyantBoussinesqPimpleFoam
buoyantBoussinesqSimpleFoam
buoyantPimpleFoam
buoyantSimpleFoam
chtMultiRegionFoam
chtMultiRegionSimpleFoam

밀도를 계산할 때 Boussinesq 가정을 사용하는 경우와 thermo module을 사용하는 경우에 대해 각각의 solver가 정상상태와 비정상상태 solver로 구분된다. 그리고 유체와 고체의 열전달을 함께 계산하는 복합열전달(conjugated heat transfer) 문제의 계산을 위한 정상상태 및 비정상상태 solver를 제공한다. 복사열전달 해석 기능은 Boussinesq 가정을 사용하는 solver에서는 제공되지 않는다.

3.3.1 복합열전달 해석

유체와 고체의 열전달을 함께 계산하는 복합열전달 해석은 유체와 고체를 별도의 영역(region)으로 나누어 계산하는 multi-region 방법을 사용한다. 유체의 영역과 고체의 영역을 따로 계산하고 두 영역의 경계면에서 데이터를 공유하는 방식으로 계산한다. 유체 및 고체의 영역은 하나일 필요는 없으며 여러 개의 독립적인 영역(region)이 있을 수 있다.

$FOAM_TUTORIALS/heatTransfer/chtMultiRegionFoam/multiRegionHeater 튜토리얼 문제를 이용해서 복합열전달 문제의 해석 방법을 알아보자

아래의 그림에 계산 영역과 계산 조건을 나타내었다.

가운데 T 자 형태의 고체영역(heater)가 있고 그 좌우에 고체영역들(leftSolid, rightSolid)이 있다. 위쪽에는 공기가 흐르는 유체영역(topAir)이 있고 아래쪽에는 물이 흐르는 유체영역(bottomWall)이 있다. 공기는 0.1m/s 속도로 유입되며 물은 0.001m/s의 속도로 유입된다. 초기의 온도는 전체가 300K 이며 heater의 아래쪽 면이 500K로 일정하게 주어져 있다.

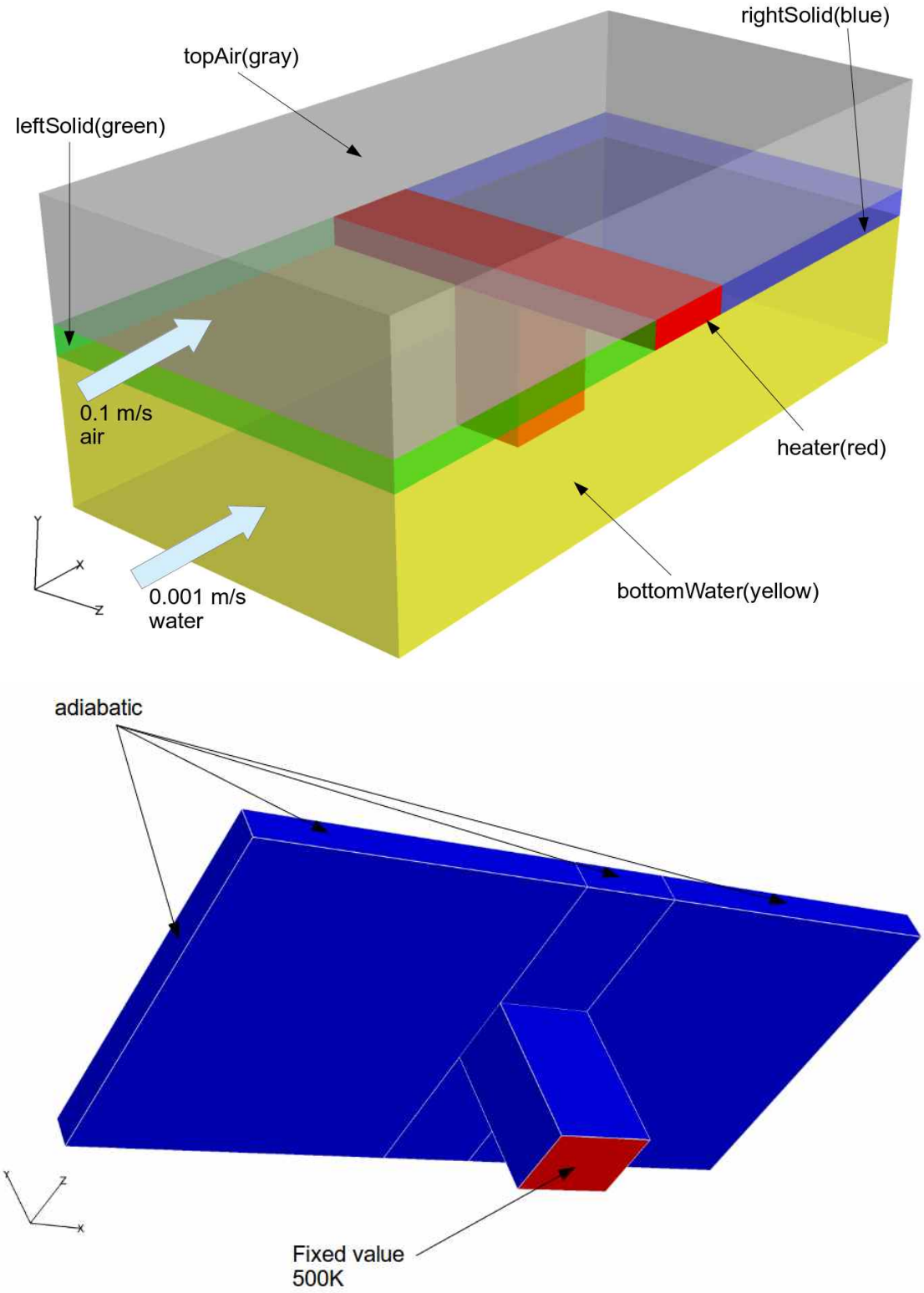


그림 3-8. 복합 열전달 해석 문제의 예

층류(laminar) 조건을 사용하고 chtMultiRegionFoam solver를 사용한다.

3.3.1.1 multiRegion solver의 폴더 구조

multiRegion solver는 OpenFOAM의 일반적인 작업 폴더의 구조와는 조금 다르다. 일반적인 OpenFOAM의 작업 폴더 구조는 “0”, ”constant”, ”system” 세 개의 폴더로 구성되고 그 안에 필요한 파일들이 존재한다. 그러나 multiRegion solver의 경우 “0”, ”constant”, ”system” 폴더 안에 각 영역(region)의 이름을 갖는 폴더가 존재하고 각각의 폴더 아래에 그 영역에 해당하는 “0”, ”constant”, ”system”에 해당되는 파일들이 있어야 한다. “constant” 폴더에는 regionProperties 파일이 있어야 한다. controlDict 파일은 “system” 폴더 아래에 있어야 하고 fvSolution 파일은 “system” 폴더와 각 region 이름의 폴더에도 있어야 한다. 단 chtMultiRegionSimpleFoam인 경우에 “system” 폴더에 fvSolution 파일은 없어도 된다.

본 예제와 같이 topAir, bottomWater, leftSolid, rightSolid, heater라는 다섯 개의 영역이 있다면 폴더 구조는 아래의 그림과 같다.

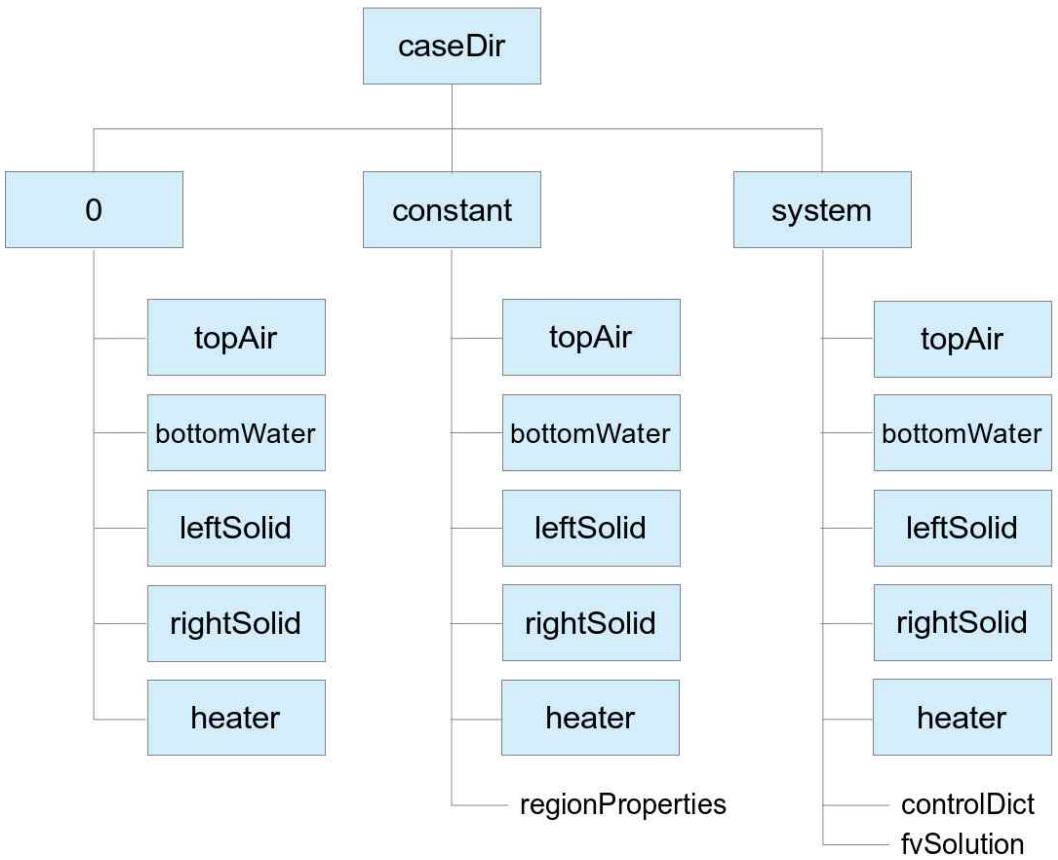


그림 3-9. 복합 영역 문제의 폴더 구조

3.3.1.2 경계조건 - ‘0’ 폴더 설정

‘0’ 폴더 아래에 다섯 가지 region의 이름을 갖는 폴더가 있고 그 아래에 각

region에 해당하는 U, p_rgh, p, T 등의 파일이 있어야 한다. topAir, bottomWater는 유체영역이며, leftSolid, rightSolid, heater는 고체영역이다.
유체영역에서는 p 파일에서 각 경계면은 calculated 조건을 설정하면 된다. 고체영역에서는 불필요한 U, p_rgh, p 파일에서 각 경계면은 calculated 조건을 설정하면 된다.
각 영역에서의 경계조건은 다음과 같다.

topAir

topAir는 minX, maxX, maxY, minz, maxZ, topAir_to_rightSolid, topAir_to_heater, topAir_to_leftSolid의 8개의 경계면이 있다.
minX는 유동의 입구이고 maxX는 유동의 출구이다. maxY, minZ, maxZ는 no-slip wall이며 나머지는 고체영역과 접하는 면으로 mappedWall이라는 patch type을 갖는다.
mappedWall의 경우는 constant/polyMesh/boundary 파일에서 patch의 속성을 정의하는 dictionary에서 sampleMode, sampleRegion, samplePatch의 세 가지를 설정해 주어야 한다. sampleMode는 nearestPatchFace로 설정하고 sampleRegion은 접하는 다른 영역을 설정해 준다. samplePatch는 접하는 영역의 해당되는 경계면의 이름을 설정해 준다. 예를 들어 topAir_to_heat의 경우는 아래와 같다. heater 영역에 topAir_to_heater면과 같은 위치에 heat_to_topAir라는 면이 있을 것이다.

```
//*****************************************
...
    topAir_to_heater
    {
        type                mappedWall;
        inGroups            1(wall);
        nFaces              40;
        startFace           3890;
        sampleMode          nearestPatchFace;
        sampleRegion        heater;
        samplePatch         heater_to_topAir;
    }
...
//*****************************************
```

속도 경계조건은 wall과 mappedWall에서는 fixedValue (0 0 0)을 사용한다. 유동의 입구인 minX에서는 fixedValue (0.1 0 0)을 사용한다. 유동의 출구인 maxX에

서는 inletOutlet 조건을 사용한다. inletOutlet 조건은 유동이 빠져나갈 때는 zeroGradient 조건을 사용하고 들어올 때는 inletValue로 설정한 값을 사용한다.

압력 경계조건(p_rgh)은 wall과 mappedWall에서는 fixedFluxPressure 조건이나 zeroGradient 조건을 사용한다. 유동의 출구에서는 fixedValue 100000를 사용하고 유동의 입구에서는 fixedFluxPressure 조건을 사용한다.

온도 경계조건은 wall에서는 zeroGradient 조건을 사용한다. 유동의 입구에서는 fixedValue 300을 사용하고 유동의 출구에서는 inletOutlet 조건을 사용한다. mappedWall에서는 고체영역의 값과 연결되어야 되기 때문에 compressible::turbulentTemperatureCoupledBaffleMixed 조건을 사용한다. 사용방법은 다음과 같다.

```
//*****************************************
...
    topAir_to_rightSolid
    {
        type                compressible::turbulentTemperatureCoupledBaffleMixed;
        value               uniform 300;
        Tnbr                T;
        kappa               fluidThermo;
        kappaName   none;
    }
...
//*****************************************
```

bottomWater

bottomWater는 minX, maxX, minY, minz, maxZ, bottomWater_to_rightSolid, bottomWater_to_heater, bottomWater_to_leftSolid의 8개의 경계면이 있다. 유동 입구의 속도가 0.001인 것과 유동 출구의 압력이 0인 것을 제외하면 모든 설정은 topAir와 동일하다.

heater

heater에는 minY, minZ, maxZ, heater_to_topAir, heater_to_rightSolid, heater_to_leftSolid, heater_to_bottomWater의 일곱개의 경계면이 있다. minY, minZ, maxZ는 patch type이며 heater_to_topAir, heater_to_rightSolid, heater_to_leftSolid, heater_to_bottomWater는 mappedWall type이다.

고체영역에서는 온도(T)를 제외한 모든 파일은 calculated 조건을 사용한다.

minY는 fixedValue 500을 사용하고 minZ, maxZ는 zeroGradient 조건을 사용한다. mappedWall의 경계조건은 유체 영역에서와 같은 조건을 사용하지만 kappa에

대한 설정값만 solidThermo로 다르다.

```
//*****************************************
...
    heater_to_topAir
    {
        type                compressible::turbulentTemperatureCoupledBaffleMixed;
        value               uniform 300;
        Tnbr                T;
        kappa               solidThermo;
        kappaName   none;
    }
...
//*****************************************
```

leftSolid

leftSolid에는 minX, minZ, maxZ, leftSolid_to_topAir, leftSolid_to_heater, leftSolid_to_bottomWater의 여섯개의 경계면이 있다. minX, minZ, maxZ는 patch type이며 leftSolid_to_topAir, leftSolid_to_heater, leftSolid_to_bottomWater는 mappedWall type이다. minX, minZ, maxZ에는 zeroGradient조건을 사용하고 mappedWall 조건은 heater에서와 같은 조건을 사용한다.

rightSolid

leftSolid와 같은 방법으로 설정한다.

3.3.1.3 'constant' 폴더 설정

'constant' 폴더 아래에 다섯 가지 region의 이름을 갖는 폴더와 regionProperties 파일이 있다.

regionProperties 파일에서 각 영역이 유체인지 고체인지를 설정해 준다. 설정방법은 다음과 같다.

```
//*****************************************
...
regions
(
    fluid       (bottomWater topAir)
    solid       (heater leftSolid rightSolid)
);
...
```

```
//*****************************************
```

유체영역 아래에는 polyMesh 폴더와 g, turbulenceProperties, RASProperties, thermophysicalProperties, radiationProperties 파일이 있고, 고체영역에는 polyMesh 폴더와 thermophysicalProperties, radiationProperties 파일이 있다.

중력 (gravity)

g 파일은 중력을 설정하는 파일인데 다음과 같이 설정한다.

```
//*****************************************
...
dimensions      [0 1 -2 0 0 0 0];
value           (0 -9.81 0);
...
//*****************************************
```

thermophysicalProperties

thermophysicalProperties 파일은 물성값을 설정하는 파일이다. thermoType을 dictionary로 설정하고 그에 따라 mixture dictionary를 설정한다. thermoType은 type, mixture, transport, thermo, equationOfState, specie, energy 등의 키워드를 갖는 dictionary이다.

type : hePsiThermo와 heRhoThermo, heSolidThermo 등이 올 수 있다. hePsiThermo는 compressibility에 기반하여 열역학적 물성값을 계산하는 방법이며, heRhoThermo는 밀도에 기반하여 열역학적 물성값을 계산하는 방법이다.

mixture : 화학종에 대한 설정으로 pureMixture, reactingMixture, multiComponentMixture, homogeneousMixture, inhomogeneousMixture, singleStepReactingMixture 등 여러가지 설정이 가능하며 본 예제와 같은 단일 화학종의 문제에서는 pureMixture로 설정한다.

transport : 점성계수를 설정하는 방법이다. const, sutherland, polynomial 등을 사용할 수 있다.

thermo : 정압비열(Cp)을 설정하는 방법이다. hConst, hPolynomial, hExponential, eConst, janaf 등을 사용할 수 있다.

equationOfState : 상태방정식을 나타내는 방법이다. PerfectGas, incompressiblePerfectGas, icoPolynomial, rhoConst, linear 등을 사용할 수 있다.

specie : 화학종의 특성을 나타내는 것으로 specie를 사용할 수 있다.

energy : 에너지방정식을 엔탈피를 사용할 것인지 내부에너지를 사용할 것인지를 선택한다. SensibleEnthalpy, sensibleInternalEnergy, absoluteEnthalpy

등을 사용할 수 있다.
본 예제에서는 유체영역에서는 다음과 같은 설정을 사용한다.

```
//*****************************************
...
thermoType
{
    type            heRhoThermo;
    mixture         pureMixture;
    transport       const;
    thermo          hConst;
    equationOfState perfectGas;
// equationOfState  rhoConst; // for bottomWater
    specie                specie;
    energy                sensibleEnthalpy;
}
...
//*****************************************
```

위와 같은 설정에서 mixture dictionary는 specie, thermodynamics, transport, equationOfState 네 가지 키워드로 구성된다. specie에는 분자량을, thermodynamics에는 정압비열(Cp)와 생성에너지(Hf)를, transport에는 점성계수와 Prandtl수를 설정한다. equationOfState를 perfectGas로 설정한 topAir의 경우 mixture dictionary에 equationOfState는 필요없지만 rhoConst로 설정한 bottomWater에는 equationOfState에서 밀도를 설정해 준다. 본 예제에서는 다음과 같다.

```
//*****************************************
...
mixture
{
    specie
    {
        nMoles          1;
        molWeight       28.9; // 18; for bottomWater
    }
    thermodynamics
    {
        Cp              1000; // 4181; for bottomWater
        Hf              0;
```

```
    }
    transport
    {
        mu                1.8e-05; // 956e-6; for bottomWater
        Pr                 0.7; // 6.62; for bottomWater
    }
    equationOfState // for bottomWater onpy
    {
        rho               1000;
    }
}
...
//*****************************************
```

고체영역에서는 thermoType과 mixture를 다음과 같이 설정한다.

```
//*****************************************
...
thermoType
{
    type                    heSolidThermo;
    mixture                 pureMixture;
    transport               constIso;
    thermo                  hConst;
    equationOfState   rhoConst;
    specie                  specie;
    energy                  sensibleEnthalpy;
}
mixture
{
    specie
    {
        nMoles          1;
        molWeight    50;
    }

    transport
    {
        kappa    80;
    }
```

```
    thermodynamics
    {
        Hf      0;
        Cp      450;
    }

    equationOfState
    {
        rho     8000;
    }
}
...
//*****************************************
```

radiationProperties

radiationProperties 파일에서 복사열전달에 관한 설정을 한다.

복사열전달을 포함하지 않는 경우 radiation 키워들을 off로 설정한다. 복사열전달을 포함하는 경우 radiation 키워드를 on으로 설정하고 radiationModel을 선택하고 그에 맞는 설정을 한다.

radiationModel에서 선택할 수 있는 것은 P1, fvDOM(finite Volume Discrete Ordinates Model), viewFactor, none의 네 가지 이다.

복사열전달 모델을 선택했다면 solverFreq, absorptionEmissionModel, scatterModel을 설정해 준다. solverFreq는 유동장이 몇 번 iteration할 때 마다 복사열전달을 한 번 계산할 것인지를 설정하는 것이다. absorptionEmissionModel은 매질의 흡수와 방사를 계산하는 방법으로 다음의 네가지 방법이 있다.

constantAbsorptionEmission
binaryAbsorptionEmission
noAbsorptionEmission
wideBandAbsorptionEmission

매질의 산란 모델인 scatterModel은 constantScatter와 none을 사용할 수 있다.

P1 모델을 사용할 때는 Incident radiation intensity, G라는 변수를 계산한다. 따라서 0 폴더에 G라는 파일을 만들어 주어야 한다.

fvDOM 모델은 radiation intensity에 대한 radiation transport equation(RTE)을 계산하는데 0 폴더에 IDefault 파일을 만들어 주어야 한다. viewFactor 모델은 경계면들 사이의 복사열전달만 계산하는 것으로 내부 매질에서의 복사열전달은 고려하지 않는다. 따라서 유동계산에 앞서 경계면들의 viewFactor를 먼저 계산해 준다. viewFactorGen 유틸리티를 사용하여 계산하는데, faceAgglomerate 유틸리티를 먼저 실행하여 계산의 효율성을 높일 수 있다. faceAgglomerate 유틸리티는

viewFactorsDict 파일에서 실행방법을 설정한다.
본 예제에서는 복사열전달은 포함하지 않는다.

3.3.1.4 'system' 폴더 설정

'system' 폴더에는 controlDict, fvSolution, decomposePar 파일과 다섯개의 region 이름을 갖는 폴더가 있다. 각 region 이름의 폴더에는 fvSolution, fvSchemes 파일이 있어야 한다.
'system' 폴더에 있는 fvSolution 파일에는 PIMPLE dictionary 중 nOuterCorrectors 만 설정한다. 다른 fvSolution의 설정들과 fvScheme의 설정들은 각 region 이름의 폴더 안에 있는 파일에서 설정한다.

3.4 다상유동 해석

OpenFOAM의 다상유동은 VOF(Volume of Fuild)를 기반으로 수면(또는 액면)을 포착할 수 있는 능력을 가지고 있으며, 2상 또는 그 이상의 상에 대해서도 적용할 수 있도록 확장이 가능하다. 그리고 수렴가속화를 위한 국소시간전진기법(LTS: local time stepping), Dynamic Mesh 라이브러리를 활용한 적응격자기법 및 격자이동, Multiple reference frame(MRF)이 선택적으로 사용가능하고 다공성 매질에 대한 해석 또한 가능하게 되어 있다. 아래 표는 다상유동 해석을 위한 기본 solver 및 그 특징을 간략히 나타내었다. 이외에 추가 기능으로 응용된 solver들은 OpenFOAM User Guide에 자세히 기술 되어있다.

다상유동의 해석 사례는 설치된 OpenFOAM의 tutorials폴더에 응용 solver의 이름으로 정리되어 있고, 그 중에 업무에 적합한 문제를 직접확인 하여 실행 및 후처리를 통해 확인해야 한다.

3.4.1 VOF 모델을 이용한 weir flow해석

본 해석은 VOF모델을 이용해 보에 물이 넘쳐 흘러나가는 유동을 2차원으로 해석한 예제이다.

- tutorial 복사

자신의 OpenFOAM버전의 run 폴더로 Tutorials의 WeirOverflow를 복사하여 저장한다.

```
$cp -rf $FOAM_TUTORIALS/multiphase/interFoam/ras/weirOverflow $FOAM_RUN
```

복사 후 weirOverflow의 폴더안에 0.org, constant, system 폴더가 구성되어 있다 Allrun를 실행하여 실행결과를 확인 할 수 있으나. 여기에서 Allrun에 해당하는 일련의 절차를 다음과 같이 나타낼 수 있다.

Allrun을 살펴보면 cp -r 0.org 0로 0.org 폴더를 0폴더로 복사하여 준다. 이는 실행에 필요한 초기값을 설정에 필요한 파일들이 들어있다. 혼돈을 피하기 위해 초기파일이 수정되지 않게 하기 위해 0.org라고 만들어 둔 것이다. 다음은 blockMesh를 통해 격자를 생성하고, cp 0/alpha.water.org 0/alpha.water.로 복사한다. 이 또한 초기파일에 덮어쓰기를 방지하기 위한 방법이다. 그리고 setFields를 통해 거동에 대한 초기값을 부여하고, ‘getApplication’이 실행되게 된다. 여기에서 ‘getApplication’에는 system의 controlDict의 application 이름인 interFoam이 들어오게 된다. 이로서 프로그램이 실행하게 된다.

- blockMesh

blockMesh를 수행하게 되면 constant폴더의 PolyMesh폴더에 blockMeshDict로부터 격자의 정보를 읽어 격자를 생성하고 경계조건을 부여하게 된다. 본 예제는 2차원 해석 예제 이지만 OpenFOAM에서는 3차원만 지원한다. 그래서 z 방향으로는 격자를 한 개만 구성하고 경계조건에서 empty로 처리하여 2차원과 같은 해석을 수행하게 한다. 해석격자의 격자점인 vertices는 총 24개로 구성되어 있고 그림 3-9의 개략도에 좌표가 표기되어 있다. 단위는 meter이다. blockMesh는 사각형격자를 만들어 주는 방법이기 때문에 본 예제와 같은 형상은 5개의 사각형 블록으로 나누어 격자를 생성하게 된다. 마지막으로 blockMeshDict에는 각면에 대해 경계조건을 부여한다. 여기에서는 inlet, outlet, lowerWall 그리고 atmosphere로 구성이 되고 나머지 면에 대해서는 자동적으로 empty로 처리된다.

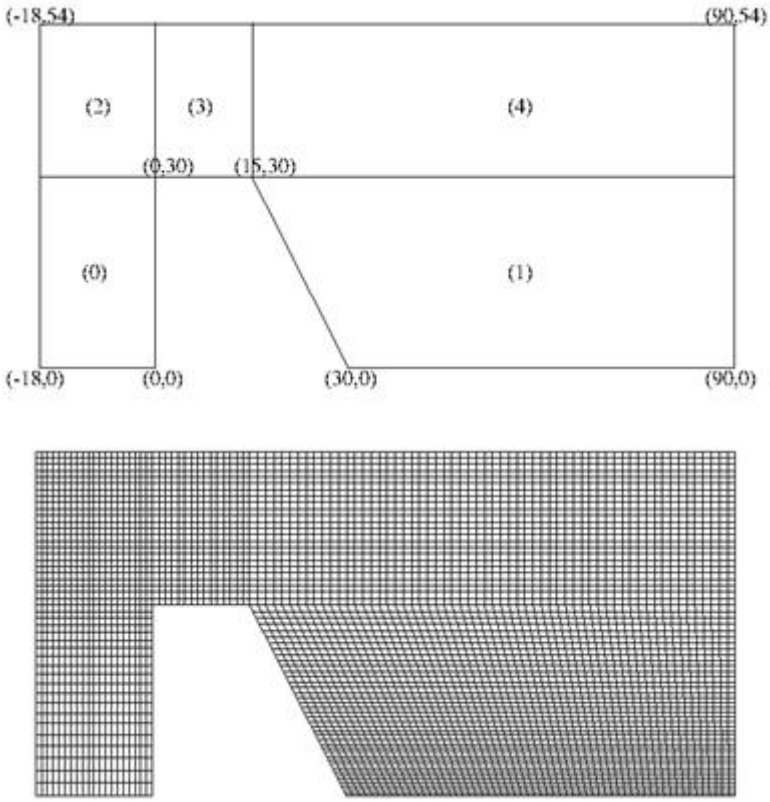


그림 3-10. 개략도

- 물성치 및 경계값 설정

전산유체해석에서 해석을 수행하기 위한 경계조건, 초기값, 해석도구 선정 및 해석도구 작동에 필요한 파라메터를 설정하고 해석을 수행하게 된다. OpenFOAM도 이와 같은 일련의 절차에 따라 해석을 수행하기 위한 기본조건을 만족시켜야 한다. 경계조건은 격자형성 시 constant/polyMesh폴더에 boundary이 생성된다. 본 예제

의 경계조건은 다음과 같이 그림으로 나타낼 수 있다.

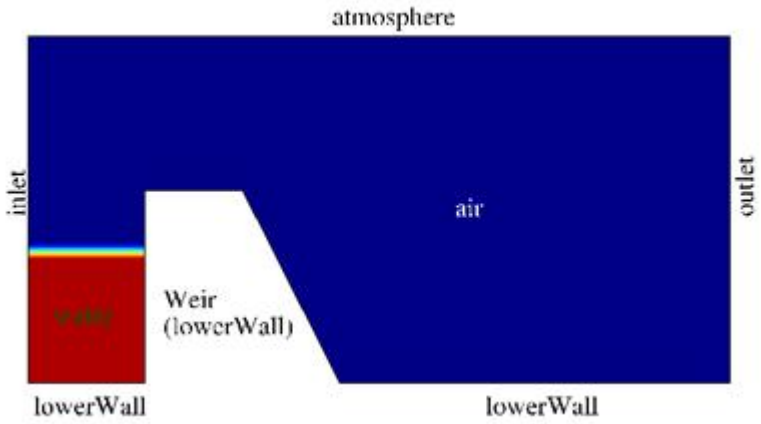


그림 3-11. 경계조건

작동유체의 물성치, 중력, 난류사용에 대해서는 constant폴더에 그에 합당한 파일들이 포함되어 있다. 중력은 g파일에 저장되며, 사용하는 격자계의 중력방향을 설정할 수 있다.

작동유체의 물성치는 transportProperties파일에 수록되며, 본 예제와 같은 다상유동의 경우 상에 따른 다종의 물성치를 입력하게 된다. 여기에서는 물과 공기 그리고 물과 공기 사이의 표면장력에 관련된 계수 sigma가 함께 설정된다.

난류사용에 대한 설정은 turbulenceProperties 파일에 RASModel이라고 작성한다.

그리고 난류모델 선정에 대해서는 RASProperties파일에 작성하게 되며, 본 예제에서는 k-e 난류모델을 사용한다.

초기값 설정은 0폴더에서 필요한 경계값 및 초기값을 읽고 시간의 경과에 따라 해석결과 변수 값을 저장하게 된다. 예로 0폴더에 있는 alpha.water파일은 물의 VOF에 대한 경계값 및 초기값을 설정한다. 전부 물일 경우 alpha=1이고 공기일 경우 alpha=0이다.

dimensions [0 0 0 0 0 0 0];은 alpha의 차원으로 나타내며 전부 0이므로 무차원수 임을 알 수 있다. internalField uniform 0;은 초기값으로 내부에 공기만 있는 것으로 나타낸다. 그 다음은 경계값으로 alpha의 inlet 타입은 variableHeightFlowRate로 유량이 주어지면 유입유량의 높이를 제한할 수 있는 경계조건으로 여기에서는 최저 0.0에서부터 최대 0.9까지 허용하고 있다. Outlet 경계조건의 타입은 zeroGradient로 외삽조건을 부여했다. 같은 경계조건으로 lowerWall도 부여한다. 그리고 atmosphere의 타입은 inletOutlet 경계조건을 부여하였다. 이 경계조건은 기본적으로 유출조건이나 계산영역 밖으로 유출일 경우 외

삽조건이 적용되고 계산영역 안으로 유입일 경우 inletValue로 적용하게 된다. 마지막으로 defaultFaces의 타입은 empty로 부여하여 2차원 해석이 가능하도록 했다. Empty 경계조건은 차원을 감소시켜 1차원 이나 2차원 해석을 가능하도록 한다. 경계조건 중 alpha.water파일의 내용을 살펴보았고 나머지 U, p_rgh, epsilon, k, nut 도 그 물성치와 경계조건에 맞게 수정하여 저장한다.

- SetFields

0폴더에서 경계의 경계값 및 초기값을 입력하고, 해석영역 임의의 위치에 변수의 값을 부여하거나 초기화하기 위해 setFields란 utility가 제공 된다. setFields를 실행하기 위한 setFiledsDict파일은 system폴더 아래 있다. 본 예제의 setFiledsDict를 살펴보면 다음과 같다.

Alpha에 대한 초기값은 영으로 되어 있다. 그리고 임의의 원하는 영역에 alpha를 1로 두면 임의의 원하는 영역만큼 물이 초기에 채워진다. box (-100 0 -100) (0 20 100);는 높이 20까지 물을 채우겠다는 의미이다. Terminal에 setFields를 실행 후 paraFoam으로 확인하면 다음과 같다.

- 해석설정

경계조건, 물성치, 난류 그리고 초기화까지 수행하였고 이제는 해석에 필요한 수치기법들의 설정을 확인 한다. 해석설정은 system폴더안에 controlDict, fvSchemes, fvSolution, setFieldsDict이 있다. controalDict는 해석에 필요한 입출력과 해석시간에 대해 기술한 부분이다. 본 예제에서 다상유동해석 프로그램 중 interFoam을 사용하고 있다.

startFrom은 해석의 시작 time directory를 설정해주는 부분이다. 여기에서는 latestTime으로 되어 있고 마지막 time directory에서 해석을 시작하라는 의미이며 이외에도 startTime, firstTime을 설정할 수 있다.

startTime은 해석 시작을 의미한다. 여기에서 0초라 가정하고 계산을 시작한다. 그리고 stopAt은 언제 계산을 끝낼 것인지를 정하는 옵션이다. 여기에서는 endTime 끝나는 시간까지로 정해져 있다. 이외에도 nextWrite, noWriteNow, writeNow가 사용될 수 있다.

deltaT는 해석 시간간격을 나타내고 여기에서는 0.001초로 두었다. 시간간격이 후의 제어 파라메터는 데이터출력에 대한 출력간격, 출력형식, 출력형식의 정확도, 출력물의 압축여부 묻게된다. 그 다음은 시간에 대한 형식과 정확도를 지정한다. maxCo는 최대 Courant수, maxAlphaCo는 VOF를 해석 시 최대 Courant 수, maxDeltaT는 최대 시간간격을 나타낸다.

다음은 공간차분, 내삽기법 그리고 flux 등에 관계된 내용은 system폴더 아래 fvSchemes파일에 수록되어 있다. 본 예제에서는 다음과 같이 구성된다.

첫 번째 줄은 시간전진기법을 나타내고 있다 여기에서는 Euler를 사용하고 있다.

구배는 Gauss Theorem과 선형 보간을 사용한다. 그리고 대류항과 발산항에 대해서 각 방정식에 적합한 공간차분기법을 적용한다. 여기서 사용된 내삽기법은 선형 타입이다.

fvSchemes에서 각방정식의 공간차분을 주로 다룬다. 그리고 각 방정식을 풀기 위한 설정은 system 폴더 아래의 fvSolution을 통해 설정한다.

본 예제에서는 VOF, 압력, 운동량, 난류를 풀기 위한 수치해법을 개별적으로 나열해 놓았다. Alpha.water는 VOF를 풀기 위한 매개변수 값을 지정한다. 압력은 PCG란 수치해법을 사용하고 있고 이는 Preconditioned Conjugate Gradient Method를 나타낸다. 그래서 preconditioner 설정 부분이 있다. 여기서 사용된 DIC 는 diagonal Incomplete-Cholesky이다. 운동량 방정식과 난류방정식은 smoothSolver로 symmetric Gauss-Seiddel을 사용한다. 여기까지 각 방정식별 수치해법을 설정하고, SIMPLE과 PISO기법을 합친 PIMPLE 기법으로 운동량방정식과 압력을 풀고 연속방정식을 만족하도록 반복적으로 해석을 하는 기법이다. 그래서 운동량예측자 설정 여부를 물어본다. 그리고 수정자는 몇 번 수행할지 물어본다.

- 실행

일련의 절차는 다음과 같다. 1. 격자생성, 2. 물성치 및 경계조건, 3. 초기화, 4. 설정확인, 5. 실행이고 Terminal에

```
BlockMesh
cp 0/alpha.water.org 0/alpha.water
setFields
interFoam
```

의 순서로 실행하게 된다.

- 후처리

OpenFOAM의 기본 post-processer는 paraview를 사용하며, OpenFOAM의 결과를 확인하기 위한 추가기능이 더해진 paraFoam을 사용한다. Terminal 에서 paraFoam을 입력하면 결과를 확인할 수 있다.

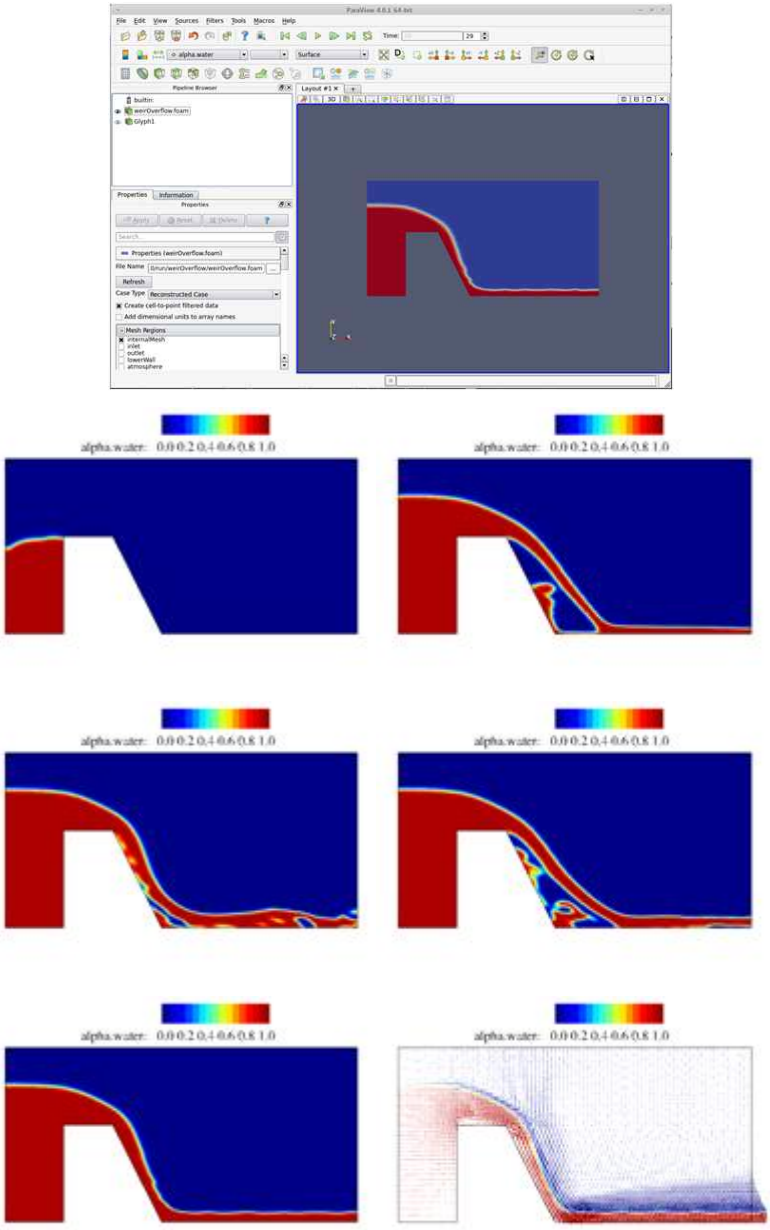


그림 3-12. 해석 결과

paraFoam을 통해 기본적인 유동의 현상 및 시간에 따른 동영상 구현도 가능하다. 본 예제에서처럼 물의 수위가 높아짐에 따라 보를 넘어 물이 흘러가는 유동현상을 잘 볼 수 있다. 특히 물이 보를 넘을 때 공기가 갇혀 발생하는 공동을 확인 할 수 있다. 에서 터미널에서 아래 그림과 같이 'pwd'라고 입력을 하면 현재 작업 중인 폴더의 경로를 확인 할 수 있다.

- 병렬해석 설정 방법

병렬해석을 위해서는 system폴더 안에 decomposeParDict파일이 있어야 한다. 그

구성은 다음과 같다.

numerOfSubdomains 4;는 CPU 또는 Core가 맡아야 하는 해석영역을 나눈 개수이다. 여기에서 부영역이 4로 4개의 CPU또는 Core를 사용하게 된다. method scotch;는 영역분할을 scotch란 방법으로 나눈다고 정의한 부분이며, simple, metis등의 영역분할 방법을 사용할 수 있다. Scotch나 metis를 사용할 경우 method 이하는 설정하지 않아도 된다. Simple을 사용할 경우 아래 부분의 설정을 수정하여야 한다.

병렬해석은 terminal에서

$BlockMesh
$cp 0/alpha.water.org 0/alpha.water
$setFields
$decomposePar
$mpirun -n 4 interFoam -parallel

해석 완료 후 해석결과를 각 프로세서에 받아 하나로 만들기 위해 Terminal에 다음과 같이 입력한다.

$reconstructPar
$paraFoam으로 결과 확인

4. 전·후처리 Utilities

4.1 snappyHexMesh

4.1.1 snappyHexmesh 기능 및 격자 생성과정

snappyHexMesh는 OpenFOAM 내에 있는 격자 생성 유틸리티로 육면체 격자를 기반으로 자동으로 mesh를 만들어 주는 기능을 가지고 있다. Geometry형상으로 stl 포맷으로 된 파일을 가지고 refine 및 snap과정을 거쳐 육면체 및 다면체 cell들을 포함한 격자를 만들어 준다. 또한 추가적으로 prism형태의 layer를 삽입할 수도 있다.

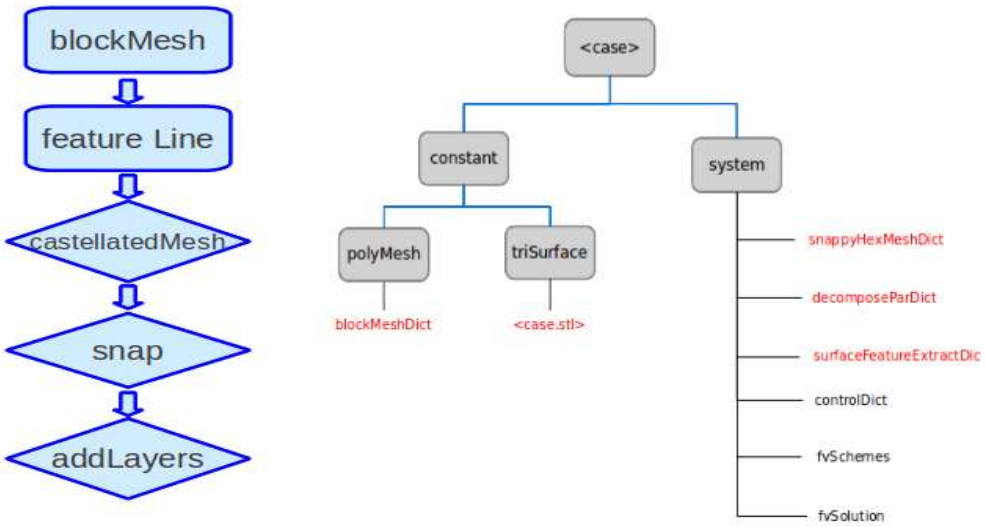


그림 4-1. snappyHexMesh 과정 및 폴더구조

snappyHexMesh를 위해선 먼저 snap과 refine을 위한 blockMesh와 stl 형상을 잘 재현하기 위한 feature line을 생성해야 한다.
snappyHexMesh는 총 3단계로 진행이 된다. castellat, snap, addLayer의 단계를 거쳐 최종적으로 mesh 수렴조건에 맞는 격자를 만들어 주게 된다.
OpenFOAMsolver를 사용하는 것과 마찬가지로 snappyHexMesh를 위해서는 기본적인 폴더구조 및 파일들을 가지고 있어야 한다. 그림 4-1의 오른쪽 그림의 폴더들과 파일들을 기본적으로 구성한 상태에서 격자를 만들 수 있게 된다.
snappyHexMeshDict 파일의 내용에 관해선 4.1.4절에서 자세히 살펴보겠다.

4.1.2 blockMesh 기능 및 내용

blockMesh는 OpenFOAM 내에 있는 간단한 형상의 육면체 격자를 만들 수 있는

유틸리티이며, snappyHexMesh를 수행하기 위해서 그 선행 단계로 blockMesh 유틸리티로 먼저 기본 격자를 만들어 주어야 한다.
blockMesh 유틸리티를 수행하기 위해서는 OpenFOAM case 폴더 안의 constant/polyMesh 폴더 위치에 blockMeshDict 파일이 있어야만 한다. 터미널 창에 blockMesh를 실행하면 blockMeshDict파일에서 설정한 격자정보를 기반으로 격자가 만들어지며 이 격자에 대한 정보가 constant/polyMesh 폴더안에 생성이 된다. 격자정보는 points, faces, cells, owner, neighbour, boundary 파일 등에 저장이 되어있다.
blockMeshDict를 구성하는 내용은 convertToMeters, vertices(꼭지점), edges(모서리), blocks, boundary, mergePatchPairs 등이 있다. blockMesh는 기본적으로 8개의 꼭지점을 가지고 있는 육면체 격자이다. 각각의 꼭지점들은 직선 및 곡선으로 연결할 수 있고 축 방향 edge별로 cell 개수를 지정할 수 있으며, 꼭지점이 8개 미만인 형상(prism, wedge)도 만들 수 있다. 따라서 사용자가 원하는 형상을 각각의 키워드에 맞게 blockMeshDict 파일을 작성해야한다.
각 block은 local coordinate system을 가지고 있다. 0번 꼭지점을 기준으로 0에서 1번 꼭지점 방향을 axis1(a1), 1번에서 2번 꼭지점 방향을 axis2(a2), 0에서 4번 꼭지점 방향을 axis3(a3)으로 정의 된다. 그림4-2는 block의 각 꼭지점 및 축을 나타낸다.

4.1.2.1 convertToMeters

convertToMeters는 격자의 scale을 나타낸다. 원하는 상수를 격자크기에 곱해서 blockMesh를 만든다. 예를들어 mm단위의 blockMesh를 만드는 키워드는 다음과 같다.

```
convertToMeters      0.001;
```

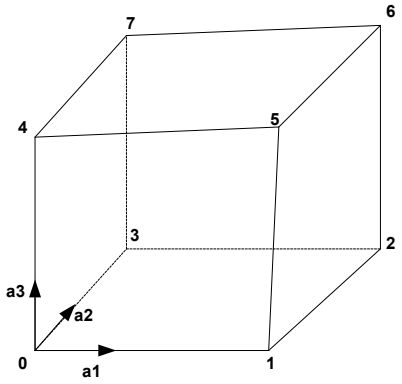


그림 4-2. block의 각 꼭지점 및 축

4.1.2.2 vertices

기본적인 육면체 block을 위해 x, y, z 좌표로 이루어진 8개의 꼭지점이 있어야한다. Dict파일에 나열된 꼭지점은 순서대로 0번부터 7번으로 정의가 된다.
그림4-2의 block을 예로들면 다음과 같고, 위 첫번째가 0번 꼭지점, 마지막 줄이 7번 꼭지점이 된다.

```
vertices
(
        ( 0 0 0 )
        ( 1 0 0 )
        ( 1 1 0 )
        ( 0 1 0 )
        ( 0 0 1 )
        ( 1 0 1 )
        ( 1 1 1 )
        ( 0 1 1 )
);
```

4.1.2.3 edges

두 개의 꼭지점을 연결하는 edge를 직선이나 곡선으로 정의 할 수 있다. edges에 특정 짓지 않은 선은 기본적으로 직선이 된다. 곡선의 종류는 arc, simpleSpline, polyLine, polySpline이 있다. 곡선은 각각 중간 꼭지점들을 포함하고 있어야하며 arc는 두 점을 원으로 연결할 때 원이 지나는 지점의 좌표를 가지고 있어야 한다.
예로 xy평면에 0(0 0 0)과 1(2 0 0)을 arc로 연결 할 때는 원이 지나는 (1 1 0)의 좌표포인트를 입력해야 한다.

```
Edges
(
        arc 0 2 (1 1 0)
);
```

4.1.2.4 blocks

하나의 block을 정의하기 위해서는 block을 이루는 꼭지점과 축방향별로 cell 개수, 각 꼭지점에서의 cell 밀집도를 특정지으면 된다. 하나의 육면체(hex) block을 지정하는 입력 예는 다음과 같다.

```
blocks
```

```
(
        hex (0 1 2 3 4 5 6 7)   (10 10 10)   simpleGrading (1 2 3)
);
```

먼저 block의 종류와 그에 맞는 꼭지점들을 나열한다. 육면체를 만들기 위해서는 8개의 꼭지점들을 순서대로 나열하면 된다. block의 종류에는 hex, wedge, prism 등등 여러 종류가 있으며 더 많은 정보는 OpenFOAM설치 폴더 안의 etc/cellModels 파일을 열어보면 확인 할 수 있다.
다음으로 각 축방향으로 cell 개수를 지정할 수 있다. 차례대로 a1, a2, a3에서의 개수를 나타낸다.
마지막으로 expansion ratios를 특정지어 각 꼭지점에서의 cell 밀집도를 정할 수 있다. 각 edge의 시작점의 cell 크기를 δs,끝점에서의 cell크기를 δe라고 하면 expansion ratios의 값은 다음 식으로 정의 된다.

$$\mathrm{Exp}ansion\,ratio = \frac{\delta_e}{\delta_s}$$

따라서 값이 1이면 cell의 크기가 일정하고 1보다 작으면 끝점에 1보다 크면 시작점에 cell이 밀집된다.
또한, expansion ratios에는 simpleGrading과 edgeGrading 두 가지가 있는데 전자는 3개의 축에 적용하는 것이고 후자는 각각의 edge 마다 expansion ratio를 적용하는 것이다. 따라서 육면체 일 때 edgeGrading은 다음과 같이 정의한다.

EdgeGrading (1 2 3 4 2 2 2 2 1 1 1 1)

4.1.2.5 boundary

OpenFOAM에서는 격자에서 경계조건을 줄 수 있는 영역을 대표적으로 patch라고 부른다. 따라서 경계조건을 주기 위한 모든 patch들을 boundary에 정의해 주어야 한다. 이렇게 정의된 patch들은 0번 폴더의 field values(U, p, turbulence)에서 초기값 및 경계조건을 정의할 수 있다.
하나의 patch는 이름, type, faces를 포함하는 형태로 나타낸다. 이름은 사용자가 원하는 이름을 정의하면 되고, type에는 기본적으로 사용할 수 있는 것들이 있는데 patch, wall, symmetryPlane, empty, wedge, cyclic, processor가 있다. 각각의 type에 대한 내용들은 OpenFOAM UserGuide-5.2 단락을 참고하기 바란다.
faces는 하나의 patch가 포함하는 면들을 꼭지점들로 나타내면 된다. 즉, 하나의 면을 4개의 점들로 묶어 나타내고, 방법은 block의 면에서 외부로 향하는 쪽을 normal 방향으로 했을 때 오른손 법칙으로 꼭지점들을 나타내면 된다.

그림 4-2에서 왼쪽면을 inlet이라는 이름의 patch로 정의하면 다음과 같다.

```
boundary
(
        inlet
        {       type   patch;
        faces
        (       (0 4 7 3);      );
        }
);
```

여기서 (0 4 7 3)은 (4 7 3 0)으로 나타내도 무방하다.
boundary에 정의 되지 않은 면들은 모두 defaultFaces라는 이름의 patch가 되며 type은 empty로 정의된다.

4.1.2.6 mergePatchPairs

한 개 이상의 block을 만들 때 사용하는 키워드이며 face matching과 face merging의 기능이 있다.
face matching은 두 개의 block이 같은 꼭지점을 가지고 있는 면들을 공유할 때 자동적으로 internal face로 인식한다.
face merging은 두 개의 block에 정의된 patch에서 서로 merge 시킬 patch들을 master, slave로 나누어 slave patch를 master patch에 투영 시켜 교차하는 면은 internal로 아닌 곳은 external로 바꾸어 준다.
위에서 설명한 키워드를 blockMeshDict 안에 정의해 주고 Dict파일을 앞서 설명한 특정 위치에 가져다 놓은 후 'blockMesh'를 터미널에서 실행하면 blockMesh 격자를 만들 수 있다.

4.1.3 surfaceFeatureExtractDict

surfaceFeatureExtract 유틸리티를 이용해서 stl 형상에 맞는 feature line을 추출할 수 있다. snappyHexMesh를 만들때 feature line을 이용하면 형상을 더 깨끗하게 구현할 수 있는데, snappy하는 과정 중에 castelate 단계에서 feature line에 있는 blockMesh의 cell들을 분할(refine)하게 된다. feature line은 stl파일을 기반으로 추출한다. 예를들어 surfaceFeatureExtractDict안에는 다음과 같이 정의되어 있다.

```
flange.stl
{
        extractionMethod      extractFromSurface;
```

```
    extractFromSurfaceCoeffs
    {       includedAngle   150;    }
    writeObj                yes;
}
```

이는 flange.stl이라는 파일을 가지고 includedAngle에 0~180도(각이 클수록 더 좋은 line을 만들어 준다)의 각을 설정하면 feature line을 만들어 준다.
surfaceFeatureExtractDict 파일을 system 폴더 안에 위치시킨 후 터미널 창에서 'surfaceFeatureExtract'를 실행하면 <stl이름>.eMesh(예, flange.eMesh)파일이 constant/triSurface 폴더 안에 생성된다.

4.1.4 snappyHexMeshDict

snappyHexMesh는 stl파일을 기반으로 하는 3차원 육면체 격자이다. 유틸리티를 실행하면 3단계를 거쳐 자동으로 격자를 만들어 주게 되며 각각 castellatedMesh, snap, addLayers로 명명되어있다.
snappyHexMeshDict 파일을 구성하는 키워드를 그림 4-3에 나타내었다.

Keyword	Description	Example
castellatedMesh	Create the castellated mesh?	true/false
snap	Do the surface snapping stage?	true/false
doLayers	Add surface layers?	true/false
mergeTolerance	Merge tolerance as fraction of bounding box of initial mesh	1e-06
debug	Controls writing of intermediate meshes and screen printing — Write final mesh only — Write intermediate meshes — Write volScalarField with cellLevel for post-processing — Write current intersections as .obj files	 0 1 2 4
geometry	Sub-dictionary of all surface geometry used	
castellatedMeshControls	Sub-dictionary of controls for castellated mesh	
snapControls	Sub-dictionary of controls for surface snapping	
addLayersControls	Sub-dictionary of controls for layer addition	
meshQualityControls	Sub-dictionary of controls for mesh quality	

그림 4-3. snappyHexMesh의 주요 키워드

먼저 snappyHexMesh 유틸리티를 실행하기 전에 선행 단계로 geometry파일인 stl(stereolithography)을 특정 위치에 가져다 두어야 하며, 사용자가 원하는 해석 도메인 크기에 맞는 blockMesh와 stl형상에 맞는 feature line을 만들어 주어야 한

다. blockMesh를 만들 때 주의할 점은 block이 stl 형상을 포함하는 크기이어야 하며 block의 cell들이 적어도 하나라도 stl 면과 교차하는 지점이 있어야 한다. 그리고 종횡비가 1에 근접하게 만들어 주는게 격자 수렴성에 도움이 된다.
먼저 snappyHexMesh는 3가지 단계로 나눌 수 있는데 각 단계의 실행 여부를 선언한다. 즉, castellatedmesh, snap과 addLayers 단계를 true/false로 선택한다.
다음으로 중요한 키워드들은 그림 4-3에 굵은 글씨로 표시해 놓았다. 이러한 키워드들이 포함하는 내용과 설명들은 OpenFOAM tutorials 내에 snappyHexMeshDict 파일을 찾아 열어보면 설명과 함께 사용될 값을 예제로 제공하고 있어서 참고할 수 있다.

4.1.4.1 geometry

snappyHexMeshDict 파일에서 형상과 볼륨region을 정의하는 부분이다. 격자를 만들 때 사용하는 형상인 stl파일을 geometry안에 정의해주면 된다. 이 밖에 사용자가 유동장 내에 격자를 더 조밀하게 가져자고자 하는 부분을 region으로 설정해서 refine을 할 수 있다. refine은 snappy 과정 중에 castellated하는 작업으로 blockMesh의 cell들을 분할하는 것을 말한다. 그림 4-4에 refine level에 따른 cell의 분할된 모습을 나타내었다.
다음은 geometry를 정의 한 예이다.

```
geometry
{
        <case>.stl      // STL filename
        {
        type    triSurfaceMesh;
        name    <case-wall>; // patch name
        }
        <case>.stl
        {
        type    triSurfaceMesh;
        regions {       <region_name> {name  <case-wall>;}  }
        }
        box
        {
        type    searchableBox;
        min     (1.5 1 -0.5);
        max     (3.5 2 0.5);
        }
};
```

snappyMesh에 사용되는 stl파일을 정의하는 방법은 첫줄에 triSurface 안에 있는 stl 파일의 이름을 써주고 하부 키워드에 type과 name을 지정하면 name에 지정한 이름이 patch가 되고 patch type이 wall로 생성이 된다. type은 기본적으로 triSurfaceMesh를 사용하면 되고 이 밖에 용량이 큰 stl이면 distributedTriSurfaceMesh를 사용할 수 있다. 이런 방법으로 여러 개의 stl파일을 지정할 수도 있으며, stl파일이 하나의 면이 아닌 여러 면으로 이루어져 있으면 각각의 면들을 따로 regions로 이름만 정의해주면 된다.

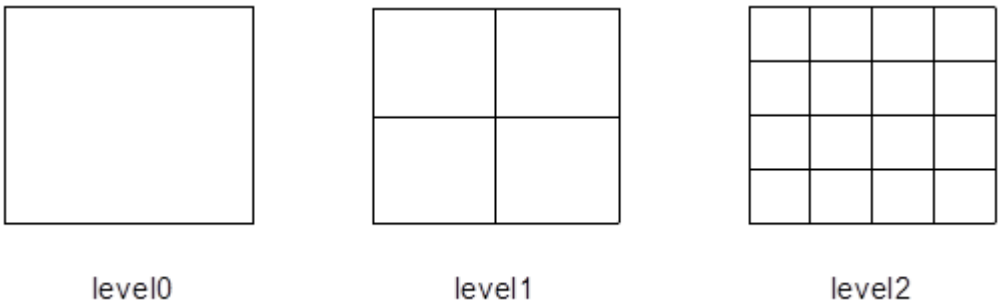


그림 4-4. refine level

형상에 대한 정의가 끝나면 geometry 안에 특정영역을 볼륨 region으로 지정할 수 있다. 그림4-5에 나타난바와 같이 비행기 꼬리 부분에 흑백영역으로 된 부분을 박스의 형태로 region을 지정할 수 있으며, 방법은 region의 이름을 선언하고 type과 그에 맞는 입력값을 넣어 주면 된다.

region에 사용할 수 있는 type에는 searchableBox, searchableCylinder, searchablePlane, searchablePlate, searchableSphere 등이 있다. searchableBox의 경우 육면체의 꼭지점 두개를 min, max로 정의 할 수 있으며, searchableCylinder는 두 점과 반경(radius)을 정의하면 된다.

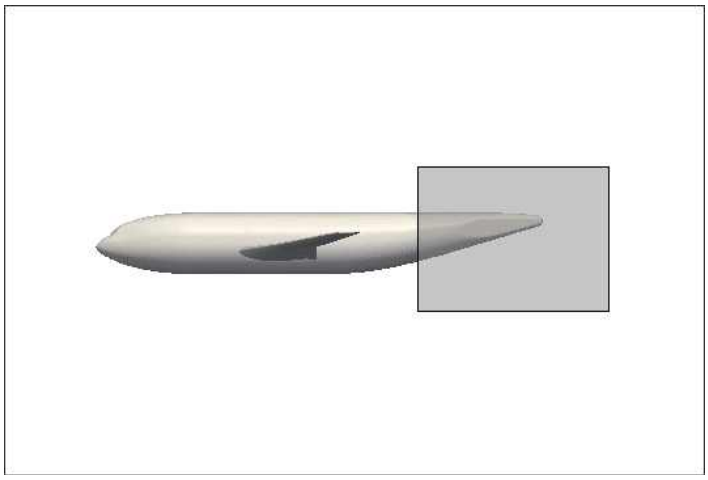


그림 4-5. refinement region

4.1.4.2 castellatedMesh

castellate는 blcokMesh를 refine하는 과정으로 feature line, surface, region을 순서대로 refine 하게 된다. refine level은 자유롭게 정의가 가능하며, refine을 한 예는 그림 4-4와 같으며, CastellatedMeshControls에 들어가는 내용은 다음과 같다.

```
maxLocalCells 20000000;
병렬로 snappyMesh를 만들 때 하나의 코어에서의 최대 cell 수
maxGlobalCells 30000000;
snappyMesh를 만들 때 전체 cell의 최대 수
minRefinementCells 10;
refine을 해야하는 격자의 기준 최소 개수. refine을 해야 하는 격자의 수가 기준
값 보다 작으면 refine을 하지 않는다.
nCellsBetweenLevels 3;
refinelevel이 다를 때 그 사이에 들어가는 격자의 수
features
(
{   file "Body.eMesh";          level 7;    }
)
refine을 해야 할 feature line의 목록과 각 refine level을 정의한다.
<case>.eMesh 파일은 constant/triSurface 폴더 안에 있다.
refinementSurfaces
{
        body            {   level (6 6);    }
)
refine을 해야 할 면들의 목록과 각 refine level의 min, max 값을 정의 한다. 먼
저 min level로 refine을 하고, 면들이 교차하는 각이  resolveFeatureAngle에서
정의한 각 보다 크면 max level로 refine을 한다.
resolveFeatureAngle 30;
refinementRegions
{
        refineBox
{
    mode inside;
    levels ((1E15 3));
}
}
```

geometry에서 정의한 영역의 refine level을 설정한다. mode에는 inside, outside, distance가 있으며 각각 거리와 level로 정의 한다. inside와 outside는 거리 값은 무시를 하고 region의 내부나 외부영역을 refine한다. distance는 region의 면에서

정의한 거리까지 refine을 한다.
locationInMesh (1 0 0);
형상의 내부 또는 외부 중 어디에 격자를 만들지를 포인트 좌표로 정의한다. 외부의 영역에 격자를 만들려면 외부영역의 임의의 포인트 좌표를 넣어주면 된다.
features, surface를 사용자가 원하는 크기 수준으로 refine을 한 후 locationInMesh에서 정의한 영역의 격자만을 남기고 반대 영역의 격자는 삭제를 한 후 region을 refine한다.

allowFreeStandingZoneFaces true;

그림4-6은 간단한 airfoil을 가지고 snappyHexmesh의 castellatedMesh가 완성된 격자를 보여주고 있다.

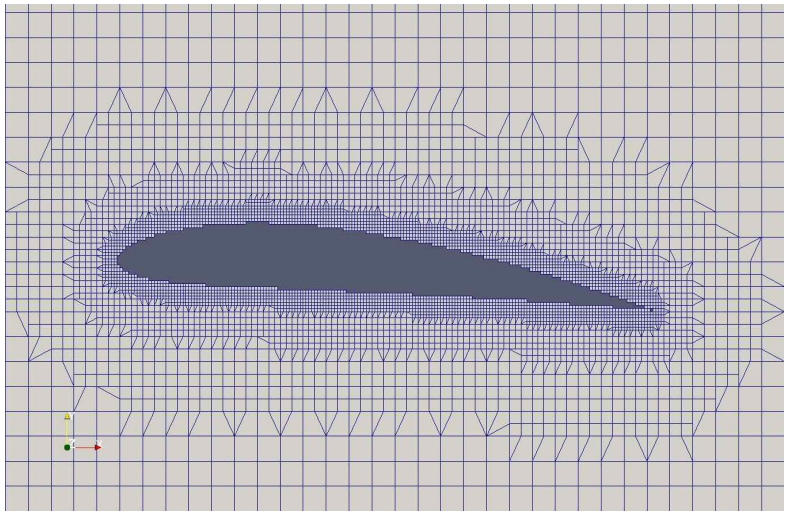


그림 4-6. castellatedMesh 격자

4.1.4.3 snap

castellated 된 격자의 꼭지점 좌표들을 형상의 면들로 이동시키는 과정을 반복적으로 수행해서 원하는 형상을 갖는 격자를 만드는 과정이다. Snap이 진행되는 와중에 meshQuality의 기준 값에 부합하는 지를 보고 반복적으로 cell을 바꿔가며 격자를 완성한다.

nSmoothPatch 3;
실제 형상의 위치로 옮겨질 점들의 위치를 계산하기 위한 반복계산의 회수
tolerance 1.0;
실제 형상의 위치로 옮겨질 점들의 현재 위치와 실제 형상 간의 상대거리 비율
nSolveIter 10;
meshdisplacementrelaxation반복계산 회수
nRelaxIter 5;

snappingrelaxation최대 반복계산 회수
nFeatureSnapIter 5;
featurelinesnapping반복계산 회수
implicitFeatureSnap false;
snapping중에 feature line을 찾아서 작업을 수행
explicitFeatureSnap true;
.eMesh파일로 작업을 수행
multiRegionFeatureSnap true;
여러 면에서 feature line을 찾는다.

그림 4-7은 airfoil의 castellatedMesh와 snap이 완성된 격자를 보여주고 있다.

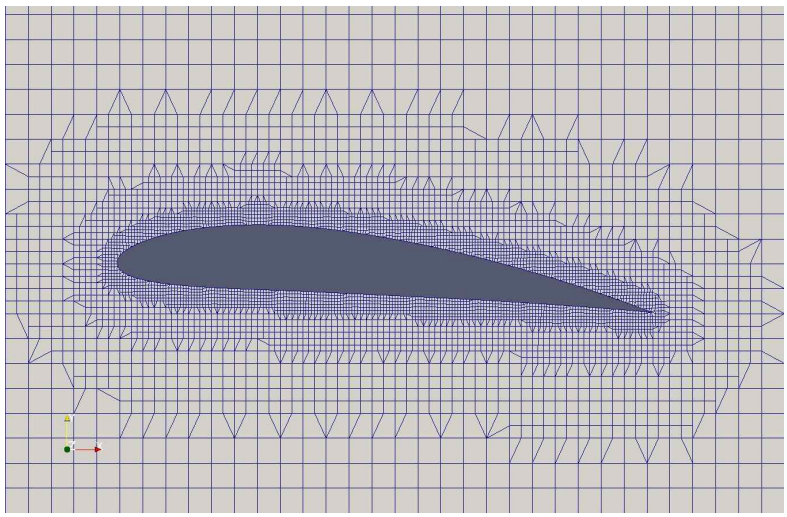


그림 4-7. snap 격자

4.1.4.4 addLayers

사용자가 원하는 patch 면(geometry에서 정의)에 특정 개수의 경계층을 삽입할 수 있다. 경계층은 면에서의 격자를 면의 수직 방향으로 늘려서 삽입하고 그 과정에서 2번의 meshQuality를 확인해서 기준에 부합하는 격자를 반복적인 작업으로 만들어 간다.

relativeSizes false;//true;
경계층을 삽입하는 방법으로 상대값과 절대값으로 지정할 수 있다. 상대값(true)은 patch 면에서 격자 크기의 상대적인 크기로 경계층을 삽입하며, 절대값(false)은 사용자가 특정지은 두께로 경계층을 삽입하며 단위는 미터(m)이다.
layers
{
body

{ nSurfaceLayers 5; }
}
사용자가 원하는 patch에 몇 개의 경계층을 삽입 할지를 정의한다
expansionRatio 1.0;
면에서 멀어지는 경계층의 두께 비율을 정의 한다
firstLayerThickness 0.3;
patch면에 인접한 경계층의 두께를 정의 한다. 예를 들어 경계층을 상대값으로 firstLayerThickness=0.3이면 patch면에서의 격자크기의 30% 두께의 첫 번째 경계층이 삽입된다.

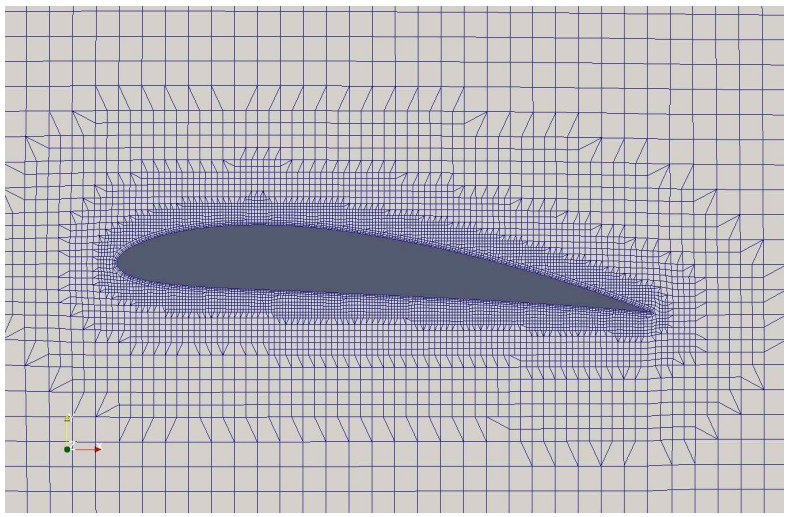


그림 4-8. addLayers 격자

경계층은 expansionRatio와firstLayerThickness의 조합으로 삽입이 이루어진다. 이 밖에도 expansionRatio와finalLayerThickness, thickness와 firstLayerThickne-ss, thickness와 finalLayerThickness의 조합으로도 경계층 삽입이 가능하다.

minThickness 0.25;
전체 경계층의 최수 두께. 이 두께 보다 얇은 경계층은 만들어지지 않는다.
nGrow 0; //1;
featureAngle 60;
여기에 정의한 각에서 경계층 삽입을 하지않는다.
nRelaxIter 5;
snappingRelaxation최대 반복계산 회수
nSmoothSurfaceNormals 1;
면의 노멀 방향으로의 포인트 smoothing 반복계산 수
nSmoothNormals 3;
내부 볼륨영역의 smoothing 반복계산 수
nSmoothThickness 10;

경계층 두께의 smoothing 반복계산 수
maxFaceThicknessRatio 0.5;
warped된 cell을 정의하며, 기준값에 부합하지 않으면 경계층을 삽입하지 않는다.
maxThicknessToMedialRatio 0.3;
경계층 두께와 면에서 medial axis까지의 거리 비율이 정의한 값 보다 크면 경계층 두께를 줄인다.
minMedianAxisAngle 90;
면에서 떨어진 격자를 이동시킬 때 medial axis를 정의할 때 사용하는 각
nLayerIter 50;
경계층 삽입을 위한 반복계산 회수. 정의한 반복계산 수에서도 수렴된 격자가 생성이 안되면 마지막 반복계산에 삽입한 경계층으로 격자를 만들어 준다.

4.1.4.5 meshQuality

meshQuality는 snappyMesh를 하는 과정에서 격자의 수렴 판정의 기준이 되는 부분이다. 격자 생성하는 단계에서 snap과 layer 삽입 시 이 기준을 만족하는 지를 확인하면서 반복적인 과정으로 격자를 만들어 준다. 그 기준들로는 cell의 볼륨값, orthogonal, skewness 등등 이 있으며 기준 값들은 수정 없이 사용하면 된다.
또한, 수렴된 격자가 만들어진 후 'checkMesh' 유틸리티로 격자의 질을 확인할 수 있으며, 이 때 격자 질의 판정 기준으로도 사용이 된다.

```
maxNonOrtho 65;
maxBoundarySkewness 20;
maxInternalSkewness 4;
maxConcave 80;
minVol 1e-13;
minTetQuality 1e-15;
minArea -1;
minTwist 0.02;
minDeterminant 0.001;
minFaceWeight 0.05;
minVolRatio 0.01;
minTriangleTwist -1;
relaxed
{       maxNonOrtho   75;     }
nSmoothScale 4;
errorReduction 0.75;
```

4.2 paraFoam

4.2.1 Paraview

Paraview는 미국 Sandia 국립연구소에서 개발한 공개 후처리 프로그램으로서 OpenFOAM을 설치할 때 자동으로 설치할 수 있다. 소스를 다운받아 설치할 때는 OpenFOAM 버전에 맞는 Thirdparty를 다운받아 컴파일 해주면 된다. 2014.09월 기준으로 ParaView-4.1이 최신버전이다.

Paraview는 VTK(visualization toolkit) 포맷으로 데이터를 처리한다. OpenFOAM은 foamToVTK 유틸리티를 사용해서 해석 결과를 VTK 포맷으로 변환시킬 수 있다. 또한, paraview는 OpenFOAM 해석결과를 직접 불러올 수 있으며 이 때 사용하는 유틸리티가 paraFoam이다. 해석 결과가 있는 폴더 위치에서 터미널 창에 paraFoam을 입력하면 '<case>.OpenFOAM' 이라는 이름을 가지고 있는 임시 파일을 자동 생성해서 불러오게 된다. OpenFOAM 해석 결과를 볼 수 있는 또 다른 방법은 <case>.foam이라는 파일을 만들어 paraview를 실행시킨 후 불러오면 해석 결과를 확인할 수 있다.

4.2.2 GUI의 구성

Paraview GUI의 구성 화면을 그림 4-9에 나타내었다.

GUI는 크게 4부분으로 나누어 보면 빨강색 숫자1이 나타내는 부분은 toolbar가 모여있는 부분이다. 여기서 contour나 streamline, vector, cutting plane 등 후처리 작업을 위한 아이콘이 있으며, 카메라 각도 조절 및 후처리 하고자 하는 해석 결과의 time directory(예, 0.1 또는 1000)폴더의 선택과 같은 사용 빈도가 높은 아이콘들을 정리해 놓은 곳이다.

2번으로 표시된 영역은 Pipeline Browser로 paraview에서 불러들인 해석 case 및 후처리 작업 중인 기능을 표시해 준다.

3번 영역은 Properties와 Information을 나타내는데 사용자가 후처리를 하고자 하는 영역 및 field values(속도, 압력, 난류, 온도 등등)의 선택, 데이터의 가공과 데이터 및 영역의 정보를 확인 할 수 있다.

4번은 사용자가 작업하는 데이터를 시각적으로 확인할 수 있는 display 창이다. Display 창에서는 camera undo, view settings을 할 수 있으며, 창을 여러 개로 분할해서 여러 작업을 동시에 진행 할 수도 있다. Display창에서 마우스 동작은 다음과 같다. 마우스 왼쪽버튼은 회전, 휠버튼은 이동, 오른쪽 버튼은 확대 및 축소 기능을 한다.

본 절에서는 OpenFOAM이 설치 된 폴더 안에 있는 예제 파일인 pitzDaily($WM_PROJECT_DIR/tutorials/incompressible/simpleFoam/pitzDaily)를 이용하여 수행하였으며 해석 방법은 터미널 창에 다음과 같이 입력하면 된다.

```
cp -r $FOAM_TUTORIALS/incompressible/simpleFoam/pitzDaily $FOAM_RUN
cd $FOAM_RUN/pitzDaily
blockMesh
simpleFoam
paraFoam
```

첫 줄은 pitzDaily 예제를 FOAM_RUN 폴더로 복사하는 명령어 이며, 두 번째 줄은 복사해온 폴더로 이동하는 명령어이다. 이동 후 blockMesh로 격자를 만들고 simpleFoam 솔버로 해석을 수행하면 된다. 마지막 줄은 Paraview를 실행하는 유틸리티이다.

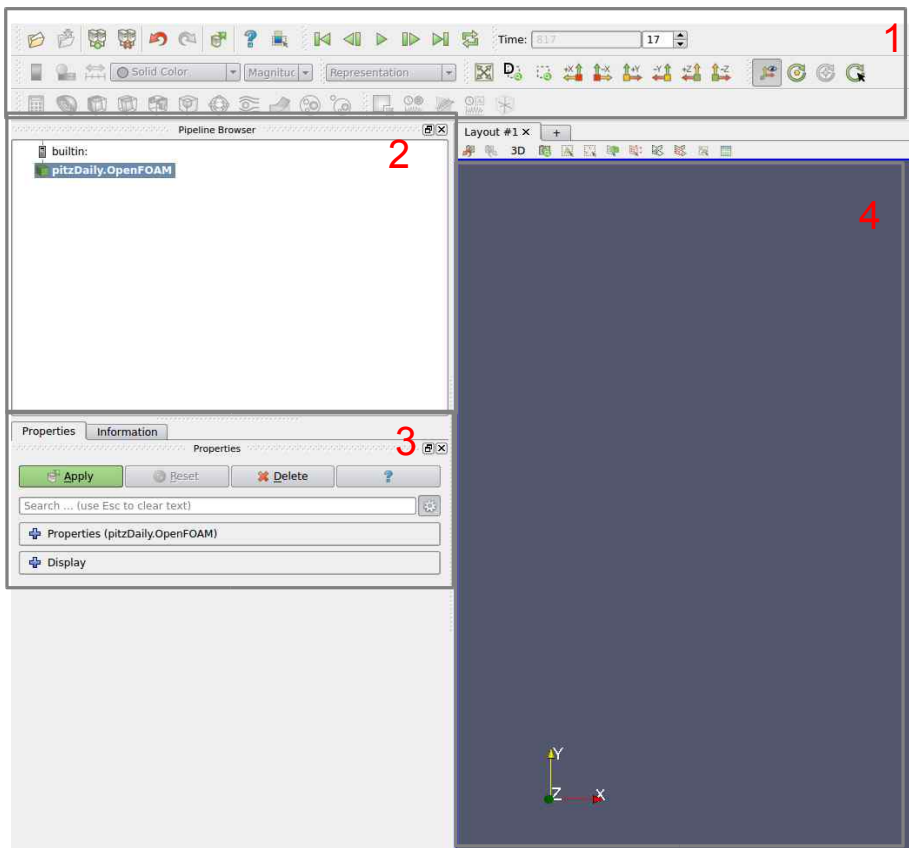


그림 4-9. Paraview

4.2.3 Pipeline Browser

Pipeline Browser는 사용자가 불러들인 <case>.OpenFOAM, <case>.foam 파일 및 작업하는 내용을 보여준다. 구성은 상하위 구조로 되어있고 동시에 여러 파일을 같은 창에서 불러들여서 작업하는 것도 가능하다. 창에 보이는 목록에서 왼쪽에 보

이는 눈모양의 버튼으로 display창에서 해당 목록을 켜고 끌 수 있다. 파랑색으로 음영처리가 되어있으면 현재 작업하고 있는 목록을 나타낸다.

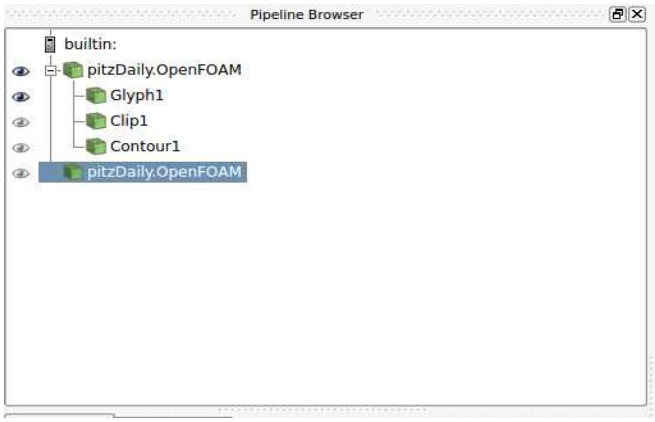


그림 4-10. Pipeline Browser

4.2.4 Properties & Information

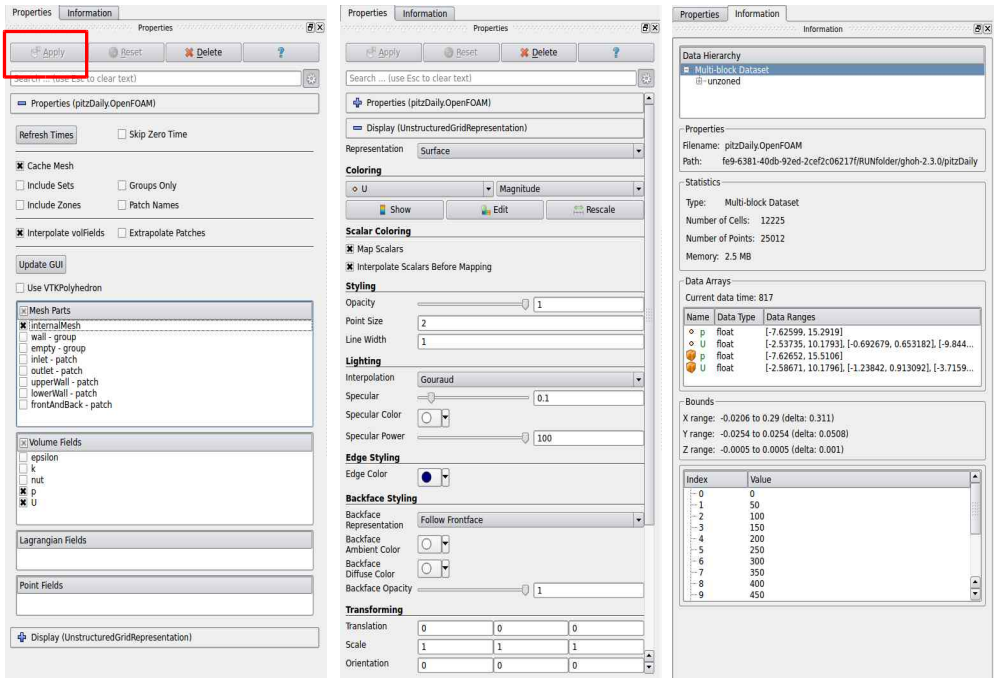


그림 4-11. Properties & Information

Properties는 Pipeline Browser에서 선택한 목록의 기본 정보를 포함하는 창이다. <case>.OpenFOAM파일을 예로들면 Properties에는 사용자가 포스트하고자 하는 영역과 field values를 선택할 수 있다.

그림 4-11 왼쪽그림에서 보이는 것처럼 .OpenFOAM파일은 'Mesh Parts'와

‘Volume Fields’가 있다. Mesh Parts는 격자의 볼륨도메인 전체를 나타내는 internalMesh와 사용자가 경계조건 및 후처리를 위해 만들어 놓은 경계면들을 포함하고 있다. 즉, OpenFOAM에서의 경계면들을 여기서 활성화 및 비활성화 시킬 수 있다. 방법은 왼쪽의 작은 체크 박스를 체크로 설정한다.
Volume Fields는 계산에 사용된 field values를 포함한다. 즉, <case>안에 0번 부터 time 폴더에 있는 field values를 여기에 나타낸다. 참고로 사용자가 OpenFOAM 유틸리티를 사용하여 만들어 준 values도 나타낼 수 있다. 예로 yPlusRAS유틸리티를 사용해서 wall에서의 yplus값을 만들면 Volume Fields에 데이터를 가져올 수 있다.
처음 데이터를 불러오거나 contour같은 기능을 사용할 때 실행시키기 위해서는 ‘Apply’버튼을 눌러줘야 한다. 그림 4-11에 왼쪽그림에서 빨강색 박스 부분이 Apply 버튼이며 처음에는 녹색으로 되어있고 이 버튼을 눌러줘서 파일 및 기능을 실행 시켜줘야 한다. 또한, Properties의 어느 하나라도 변경이 되면 Apply버튼이 녹색으로 변하게 된다.

Properties에서 작업 할 수 있는 내용으로 Display가 있다. 그림 4-11을 보면 Properties에는 Properties이외에 Display를 포함하고 있다. 여기서는 데이터의 가공을 할 수 있는 영역이다.

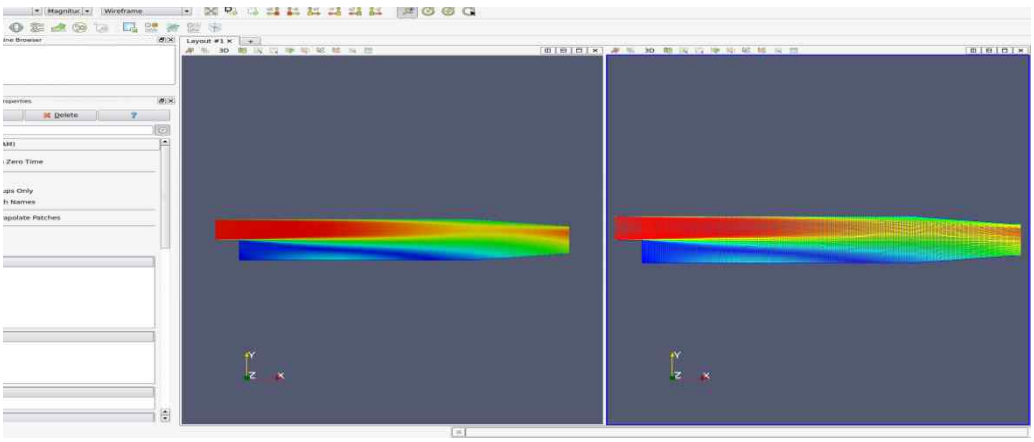


그림 4-12. Representation

Representation은 데이터를 가시화하는 옵션으로 해석 도메인 영역을 Outline, Point, Surface, Surface With Edges, WireFrame으로 나누어져 있다. 그림 4-12는 각각 Surface와 WireFrame을 보여주고 있다.
Coloring에서는 field values의 선택 및 legend를 수정하거나 설정하고 데이터의 범위를 지정할 수 있다. ‘Edit’ 버튼을 누르면 그림 4-13의 오른쪽 창과 같은 Color Map Editor 창이 나타나고 여기서 작업 할 수 있다.

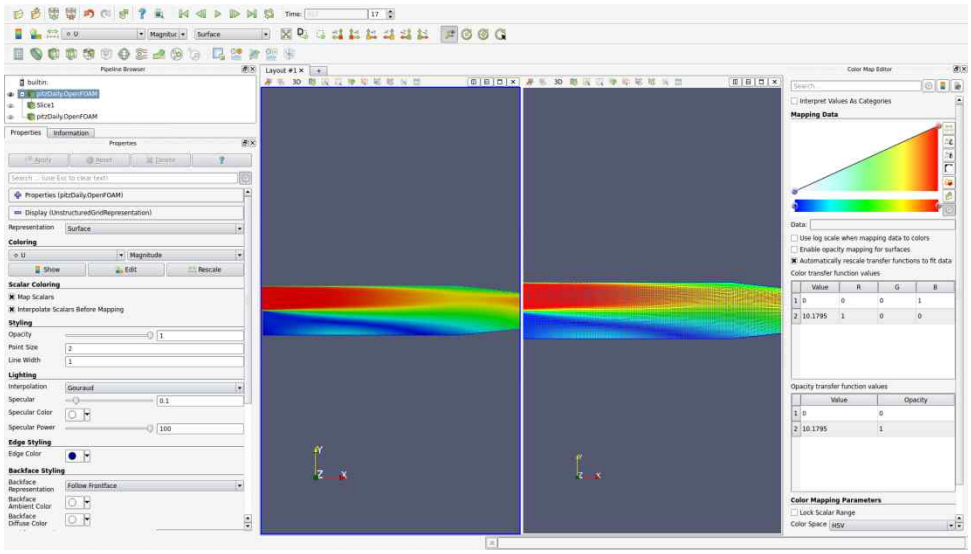


그림 4-13. Color Map Editor

Styling
Lighting
Edge Styling
Backface Styling
Transforming

이 밖에 Display의 나머지 기능들은 간단한 예제를 통해 쉽게 확인이 가능하다.

Information은 격자정보 및 field values의 최대, 최소값 등 Pipeline에서 활성화된 목록의 일반적인 정보를 포함하고 있다. 예로, 격자수, 격자의 도메인 크기, 계산된 time 폴더, fleld values 등등이 있다.

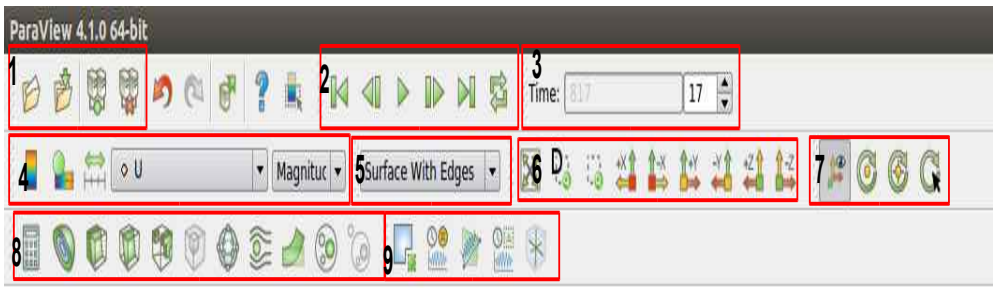


그림 4-14. Toolbars

4.2.5 Main Controls

paraview는 GUI 상단에 자주 사용하는 기능들을 아이콘의 형태로 모아놓았다. 그림 4-14에 1번 박스는 Main Controls이다. 파일을 열고 저장할 수 있는 아이콘이 있으며 바로 옆에 undo/redo 아이콘이 놓여있다.

참고로 Toolbars는 메인 메뉴에서도 활성화가 가능하다. 최상단의 View - Toolbars에서 각각의 해당 Toolbars를 GUI상에 나타나게 할 수 있으며, Toolbars들이 모여 있는 곳에서 마우스 우측버튼으로도 가능하다.

4.2.6 VCR및 Current time controls

그림 4-14에 3번 박스는 현재 창에 보이는 데이터의 iteration(steady)이나 time(unsteady) 폴더를 나타내는 것으로 OpenFOAM의 <case>폴더를 불러오면 저장된 모든 time 폴더를 다 가지고 온다. 따라서 사용자는 iteration이나 time 폴더들이 나열되어 있을 때 특정 폴더의 후처리 된 데이터를 보거나 작업을 선택할 수 있다.

2번 박스는 iteration이나 time 폴더를 이동할 수 있는 VCR controls이다. 비디오의 영상재생 아이콘과 비슷하게 생겼으며 0번 폴더에서 순서대로 이동할 수 있고 마지막 폴더로 한번에 이동이 가능하다. 또한 모든 time 폴더를 순차적으로 연속해서 보여주는 아이콘도 포함하고 있다.

4.2.7 Active Variable Controls & Representation

그림 4-14의 4, 5번 박스는 Active Variable Controls과 Representation을 나타낸다. 이 아이콘들은 4.2.4절의 Properties의 Display창에서 설명했던 Representation과 Coloring과 같은 기능을 GUI창에 나타낸 것이다.

4.2.8 Camera와 CenterAxesControls

Camera는 창에 보여지는 그림을 언제나 동일한 크기로 Reset하는 버튼과 특정 지점을 확대할 수 있는 아이콘이 들어가 있으며 그림 4-14의 6번 박스의 왼쪽에서 첫번째와 세번째 버튼이 그것이다. 박스안에 x, y, z와 함께 축 모양으로 그려진 아이콘들은 각각 그림을 축 방향에서 볼 수 있게 해준다.

7번 박스인 Center Axes Controls는 왼쪽부터 Show Orientation Axes, Show Center, Reset Center, Pick Center 아이콘을 나나태며, 화면상에 local coordinate를 on/off 할 수 있거나 센터 포인트를 옮길 수가 있다.

4.2.9 Common

8번 박스는 Common을 나타낸다. 후처리 할 때 가장 빈번하게 사용하는 기능들을 아이콘으로 모아놓았다. 상단의 메인메뉴의 Filters에서도 찾을 수 있다. 마우스버튼을 아이콘 위로 가져다 놓으면 아이콘의 이름을 확인할 수 있다.

4.2.9.1 Calculator

사용자가 원하는 field value를 수식을 이용해 만드는 기능이다. 예를 들어 압력을 무차원화 시킨 압력계수를 만들어서 후처리를 할 수 있다.

4.2.9.2 Contour

Isosurface를 만들 수 있는 기능이다. 사용자가 원하는 field value의 지정된 범위의 값을 그려준다. Properties 창에서 수정이 가능하다.

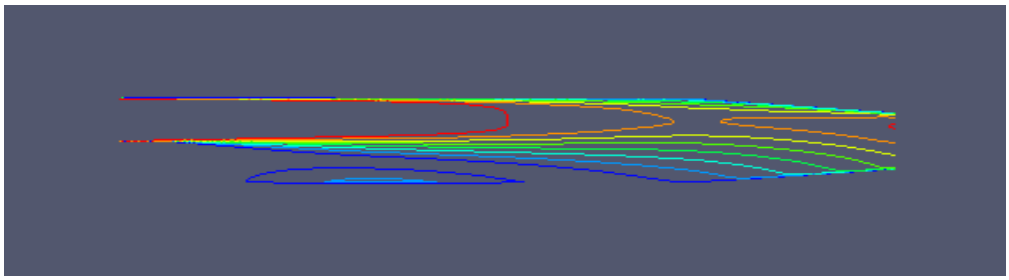


그림 4-15. Contour

4.2.9.3 Clip

계산 도메인 영역을 분할하는 기능이다. Plane, Box, Sphere, Scalar의 옵션을 사용할 수 있다.

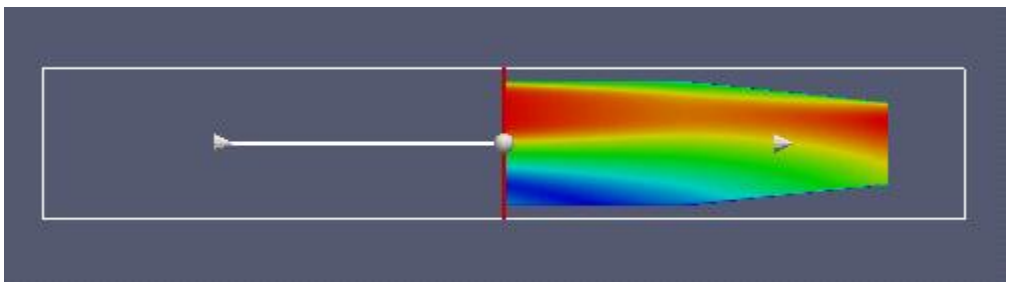


그림 4-16. Clip

4.2.9.4 Slice

Slice는 도메인 영역에 surface를 만드는 기능이다. Plane, Box, Sphere 옵션이 있으다. Slice로 도메인영역에 surface를 만든 후 contour를 이용하여 그림 4-15와 같은 contour line을 만들 수 있다.

4.2.9.5 Threshold

도메인 영역을 field value 의 최대, 최소 값을 기준으로 분할할 수 있다.

4.2.9.6 Glyph

도메인 영역에 속도 벡터를 그릴 수 있다. 벡터의 크기나 개수, 모양, Mode는 Properties에서 수정할 수 있다.

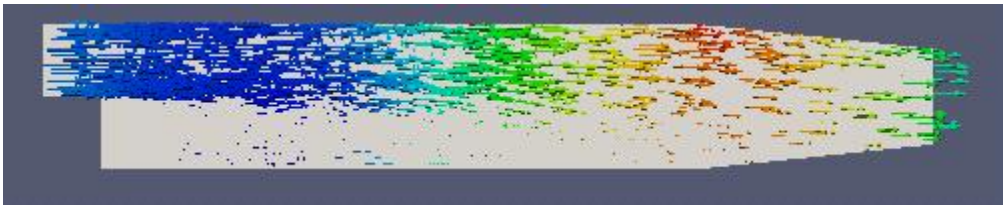


그림 4-17. Glyph

4.2.9.7 Stream Tracer

stream line을 그릴 수 있다. Properties에 seeds type에서 point나 line을 선택할 수 있다. 그림 4-18는 line source를 이용한 stream line을 보여주고 있다.

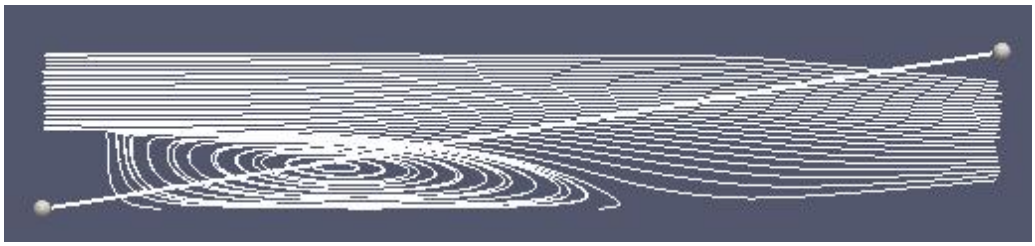


그림 4-18. Stream Tracer

4.2.10 DataAnalysis

그림 4-14의 9번 박스에는 field value를 이용해서 그래프를 그리거나 특정 포인트 지점에서의 값을 추출할 수 있다. 즉, plot과 probe 기능을 이용해서 해당 작업이 가능하다. 이러한 기능들은 메인 메뉴의 Filters에서도 찾을 수 있다.

4.2.11 그림 내보내기와 동영상 만들기

후처리 된 작업 화면을 그림파일로 저장할 수 있다. 상단의 메인 메뉴에서 File - Save Screenshot에서 가능하다.

<case>.OpenFOAM 파일을 불러오면 저장된 time폴더를 모두 불러들인다. 이러한 데이터를 이용하여 동영상을 만들 수 있다. 상단의 메인 메뉴에서 File - Save Animation에서 초당 프레임수 및 Resolution을 선택하고 동영상을 만들 수 있다.

4.2.12 State 파일 사용하기

반복적인 후처리 작업을 위해서 state 파일을 이용할 수 있다. 비슷한 해석 케이스에서 유사한 후처리를 하고자 한다면 후처리가 완료된 해석 케이스에서 state 파일을 저장한 후 다른 해석 케이스에서 state 파일을 불러들이면 가능하다.

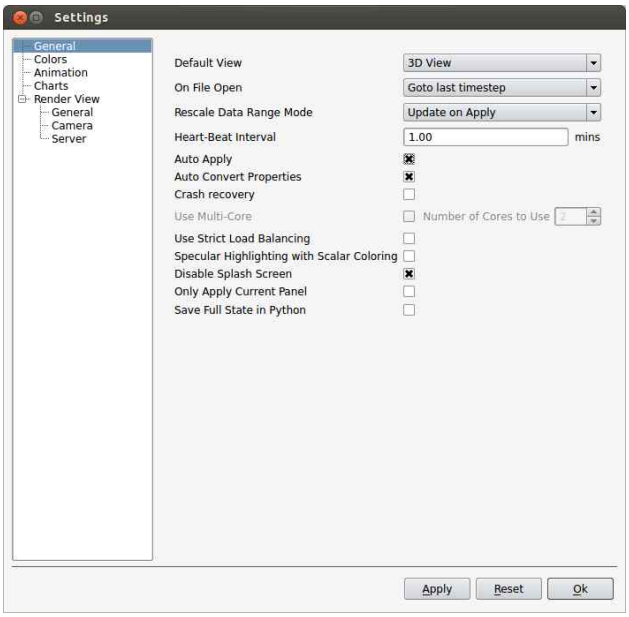
메인 메뉴에서 File - Save State에서 후처리 된 state 파일을 저장할 수 있고, File - Load State로 state 파일을 불러올 수 있다. State 파일을 불러올 때 동일한 후처리를 하고자 하는 <case>.OpenFOAM파일을 선택해주면 된다.

4.2.13 Settings & View Settings

메인 메뉴의 Edit - Settings , Edit - View Settings에서 paraview 환경설정을 해줄 수 있다.

Settings의 General에서는 Auto Apply기능이 있다. 이것은 Properties에서 Apply 버튼을 자동으로 적용하는 기능을 가지고 있다. Colors는 배경이나 surface의 기본 색깔을 지정할 수 있다. Render에서는 display창에서 마우스 버튼의 기능을 지정할 수 있다.

View Settings은 display창의 배경 색깔을 선택할 수 있고, 'Use Parallel Projection'의 체크 박스를 이용해서 원근감을 켜고 끌 수 있다.



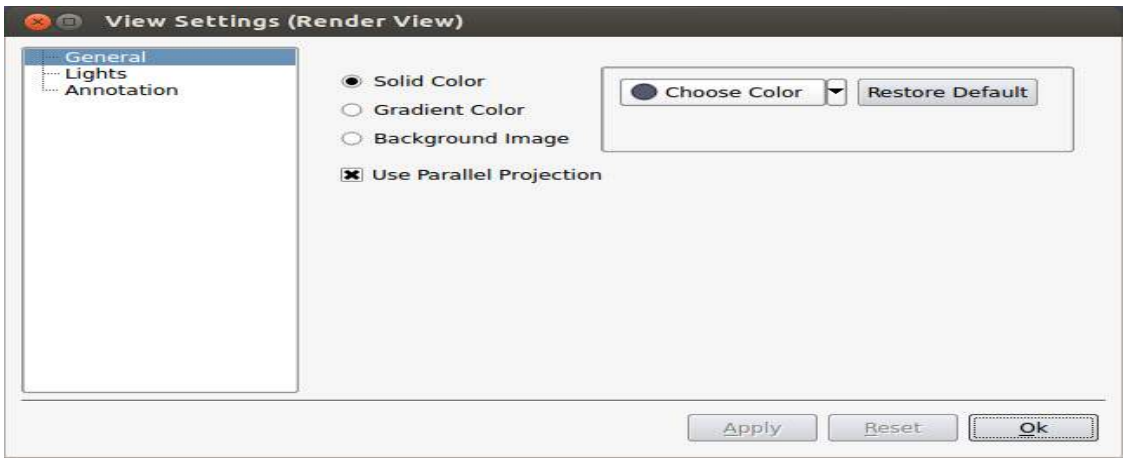


그림 4-19. Settings & View Settings

[참고문헌]

[1] http://www.openfoam.com

[2] http://www.nextfoam.co.kr

[3] A.Bansal, "K-Distribution Models for Gas Mixture in Hypersonic Nonequilibrium Flows" Ph-D Dissertation, Mechanical Engineering, PennState University, 2011 December

[4] 김태우, 오세종, 이관중 "외부 유동 해석에 대한 오픈 소스 코드, OpenFOAM의 검증" 한국항공우주학회지 제 39권 제 8호, 2011.8

[5] 박준권, 강관형, "Source 공개 코드 OpenFOAM 소개", 한국전산유체공학회지 제 15권 제3호, pp.46-53, 2010.9

[6] ParaView User's Guide (v3.10)

[7] OpenFOAM User Guide (v1.7.1), August 2010

[8] J. Allen and T. Hauser, "Parallel, Steady/Unsteady DSMC Simulations of the Aerodynamics of Sounding Rockets", 45th AIAA Aerospace Sciences Meeting and Exhibit, 8-11 January 2007, Reno, Nevada

[9] H. Tani, N. Tani and H. Negishi, "Numerical Simulation of Hypersonic Re-entry Flows using DSMCFoam", 8th International OpenFOAM Workshop, June 11-14, 2013, Jeju Korea.

[10] J. Gill, B. Kim, G. Kim, H. Shin S. Jung and G. Kim, "Development and Validation of a Density-Based Implicit Solver Using LU-SGS Algorithm", 8th International OpenFOAM Workshop, June 11-14, 2013, Jeju Korea.

[11] K. Han and K. Y. Huh, "Development of Coal Combustion Libraries based on Structural Behavior of the Particle", 8th International OpenFOAM Workshop, June 11-14, 2013, Jeju Korea.

[ISBN 978-89-294-0613-4]

손일엽 • 김병윤 • 노현석

전산열유체 공학해석 분야에서의 OpenFOAM 활용

2014년 12월 16일 인쇄
2014년 12월 16일 발행

발 행 처

대전광역시 유성구 대학로 245
㉾ 305-806
전화 : 042-869-1004
등록 : 1991년 2월 12일 제 5-259호

발행인
한 선 화

인쇄처
㈜ 미래미디어