BSPE99673-11958-5

LES 난류기법을 도입한 쇄파대 정밀 퇴적물 이동모델 수립

Establishing Micro-Scale Sediment Transport Model for Surf-zone Dynamics Using LES Turbulence Modeling Scheme

2019.02.20

한 국 해 양 과 학 기 술 원

제 출 문

한국해양과학기술원장 귀하

본 보고서를 “LES 난류기법을 도입한 쇄파대 정밀 퇴적물이동 모델수립”과제의 최종보고서로 제출합니다.

2019. 02. 20

총괄연구책임자 : 장 연 식

참 여 연 구 원 : 진 재 율
도 종 대

보고서 초록

<table>
<tr><td>과제고유
번호</td><td>PE99673</td><td>해당단계
연구기간</td><td>2018.4.1. -
2018.12.31</td><td>단계 구분</td><td></td></tr>
<tr><td rowspan="2">연구사업명</td><td>중사업명</td><td colspan="4"></td></tr>
<tr><td>세부사업명</td><td colspan="4"></td></tr>
<tr><td rowspan="2">연구과제명</td><td>대과제명</td><td colspan="4">주요사업(신진연구자 및 창의적 아이디어 지원)</td></tr>
<tr><td>세부과제명</td><td colspan="4">LES 난류기법을 도입한 쇄파대 정밀 퇴적물이동 모델 수립</td></tr>
<tr><td rowspan="2">연구책임자</td><td rowspan="2">장연식</td><td>해당단계
참여연구원수</td><td>총 : 4 명
내부: 3 명
외부: 1 명</td><td>해당단계
연구비</td><td>정부: 50,000 천원
기업: 천원
계 : 50,000 천원</td></tr>
<tr><td>총연구기간
참여연구원수</td><td>총 : 명
내부: 명
외부: 명</td><td>총
연구비</td><td>정부: 천원
기업: 천원
계 : 천원</td></tr>
<tr><td>연구기관명
및
소속부서명</td><td colspan="2">한국해양과학기술원
연안방재연구센터</td><td>참여기업명</td><td colspan="2"></td></tr>
<tr><td>국제공동연구</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>위탁연구</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="4">요약(연구결과를 중심으로 개조식 500자 이내)</td><td>보고서
면수</td><td>166</td></tr>
<tr><td colspan="6">○ LES 난류기법 도입
- Openfoam 상용모델을 토대로 LES 난류기법을 적용하여 해저면의 난류모델링 정확도 향상.
○ 쇄파대 파랑모델 수립
- VOF 기법을 적용하여 쇄파현상 구현하는 파랑모델 수립.
○ 퇴적물이동모델
- 퇴적물 농도가 아닌 퇴적물 입자의 운동을 직접 계산하는 해저면 라그랑지안 퇴적물 이동모델 수립.</td></tr>
<tr><td rowspan="2">색인어
(각 5개 이상)</td><td>한 글</td><td colspan="4">퇴적물 부유, 쇄파대, OpenFOAM, 퇴적물입자 추적모델</td></tr>
<tr><td>영 어</td><td colspan="4">sediment suspension, surf zone, OpenFOAM,
Sediment particle tracking model</td></tr>
</table>

요 약 문

Ⅰ. 제 목

LES 난류기법을 도입한 쇄파대 정밀 퇴적물이동 모델수립

Ⅱ. 연구개발의 목적 및 필요성

① 쇄파대(surf zone) 퇴적물 부유현상(sediment suspension) 규명에 중요한 난류 모델링(turbulence modeling) 정확도 개선
② CFD (Computational Fluid Dynamics) 모델 기반 쇄파대 파랑 모델 수립
③ 퇴적물 부유현상 규명을 위한 퇴적물입자 추적모형 수립

Ⅲ. 연구개발의 내용 및 범위

① Openfoam CFD 모델 적용 쇄파대 파랑모형 수립
② LES 기반 난류 모델링 기법 수립
③ 쇄파대 정밀 퇴적물 이동 모델 수립

Ⅳ. 연구개발결과

① LES 난류기법 적용한 쇄파대 파랑모델 : 쇄파현상 구현 및 쇄파 시 해저 면 난류에너지 정확도 향상
② 쇄파대 정밀 퇴적물이동 모델 : 라그랑지안 입자 추적기법을 도입한 퇴적물 입자 부유/이동 현상 모델수립

Ⅴ. 연구개발결과의 활용계획

① 연안역 지형변동 모델의 단점인 쇄파대 모델링의 정확도 향상

② 향후 포말대까지 포함한 모델 영역확장

③ 포말대 지하수 효과를 포함한 연안역 지형변동 모델링 시스템으로 개발 가능

S U M M A R Y 및 KEYWORDS

The major aim of this research is to establish the sine-scale sediment transport model in which the accuracy of the prediction of the turbulence energy that is crucial to study the sediment suspension events in the surf-zone will be enhanced. In order to accomplish this, we plan to ① establish surf-zone wave model using Open-FOAM CFD model, ② apply LES turbulence scheme to enhance the accuracy of turbulence modeling, ③ develop Lagrangian sediment particle tracking model combined with OpenFoam.

(KEYWORDS : 퇴적물 부유, 쇄파대, OpenFOAM, 퇴적물입자 추적모델,
sediment suspension, surf zone, OpenFOAM, Sediment particle tracking model)

C O N T E N T S

목 차

제 1 장

서 론

제 1 장 서 론

1. 연구내용

가. 연구개발 목적

본 연구는 연안재해대응 국가역량을 강화에 필수적인 연안역 퇴적물이동 예측의 정확도를 향상시키기 위해 최근 국제동향을 반영하여

① 쇄파대(surf zone) 퇴적물 부유현상(sediment suspension) 규명에 중요한 난류 모델링(turbulence modeling) 정확도 개선 및

② CFD(Computational Fluid Dynamics) 기반 쇄파대 파랑모델과

③ 퇴적물 부유현상 규명을 위한 퇴적물입자 추적모형의 수립을 목적으로 한다.

나. 연구개발 필요성

○ 기술적 측면

(1) 난류 모델링 정확도 향상

- 연안역 퇴적물이동현상을 규명하기 위하여 해저면에서 발생하는 퇴적물 부유현상을 정확하게 이해하는 것이 필요하며 퇴적물 부유현상을 파악하기 위해서는 그 발생기작인 난류에너지 분포를 이해하는 것이 필요하다.
- 기존의 난류모델링에 널리 사용된 Reynolds Averaged Navier-Stokes (RANS) 기법은 해저면 파랑조건하에서 정확도가 낮으며 LES 난류기법이 정확도가 높다는 연구결과가 다수 발표되었다.

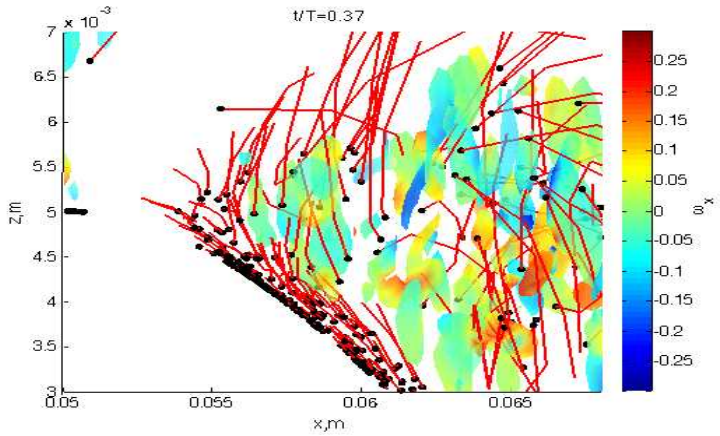


그림 1. 난류(turbulence)가 퇴적물 입자 부유에 미치는 영향 - LES 난류 모델링 기법 적용의 예 (Chang & Park, 2016)

(2) 쇄파대 파랑모델 수립

- 연안역 퇴적물이동 모델을 위해서는 쇄파대에서 천수에 의해 부서지는 파랑 및 이때 전이되는 에너지에 의한 강한 퇴적물 부유/이동 현상을 정확하게 구명하는 것이 필요하다.

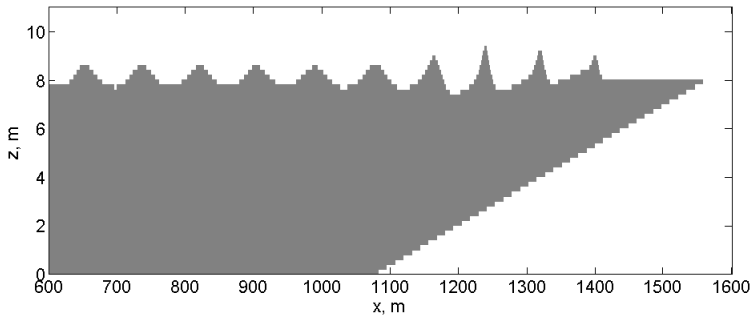


그림 2. Cadmas-Surf 를 사용한 쇄파대 파랑모델 (Chang et al., 2017(a))

- 기존 연구에서 Cadmas-Surf 모델을 사용하여 VOF (Volume of Fluid) 기법을 적용한 쇄파대 파랑연구가 진행 되었으나 (그림 2) 이 경우 난류기법의 구현에 제한이 있어 난류에너지 정확도가 떨어진다.
- OpenFoam CFD 모델의 경우 RANS (Raynolds Averaged Navier-Stokes) 및 LES(LArge Eddy Simulation) 기법 등의 다양한 난류모델링 기법 적용이 가

능하여 난류에너지 구현의 정확도 향상 및 민감도 검증이 가능하다.

(3) 정밀 퇴적물이동 모델 수립

- 쇄파에 의한 해저면 난류에너지 변화를 정확히 구현할 경우 이에 통하여 퇴적물 입자의 부유 및 이동현상을 파악할 수 있다.
- 해저면에서 퇴적물의 부유기작을 정확하게 이해하는 것은 현재 사용 중인 연안역 광역 퇴적물이동 및 지형변동 모델들의 정확도를 향상시키는데 크게 기여할 수 있다.
- 현재까지 LES를 통한 난류에너지와 VOF를 사용한 쇄파현상 및 해저면 퇴적물 입자의 움직임이 모두 검증된 정밀 퇴적물이동 모델은 발표된 바가 없으며 본 연구에서 CFD 모델을 사용하여 수립하고자 하다.
- 이 모델이 수립되면 Openfoam의 높은 확장성을 활용해 향후 포말대 및 지하수 영향 연구 등으로 발전될 가능성이 높다.

○ 경제·산업적 측면

(1) 연안침식 피해 저감

- 태풍 증 자연적인 원인 및 구조물에 위한 인위적인 원인에 의한 연안역 침식 피해가 증가하는 현황이다.
- 정확도가 향상된 정밀 퇴적물 이동 모델 수립을 통한 연안역 퇴적물 이동 예측 정확도 향상이 필수이다.

(2) 연안 퇴적물이동 예측산업 진흥

- 기후 변화로 인한 해수면 상승으로 향후 연안 지형변화에 대한 세계적 관심이 집중되고 있다.
- 따라서 향후 연안지형 변화 예측을 위한 수치모델링 산업이 발전할 것으로 예상된다.
- 최근 선진국에서 개발되고 있는 최신 모델링 기법을 도입하여 퇴적물이동 모델링 분야 선진기술 확보가 필요하다.

○ 사회·문화적 측면

(1) 지역사회 안정

- 연안침식은 지역사회에 경제적 손실뿐 아니라 분쟁 등 사회적 문제를 발생시킬 수 있다.
- 정부의 대국민 신뢰도 향상을 위하여 적절한 연안침식예방연구 성과가 필요하다.

(2) 연구원 경영목표 와의 연계성
- “해양과학기술 및 해양산업 발전에 필요한 원천연구, 응용 및 실용화 연구 분야” 와의 연계성을 통한 원천기술을 확보할 필요가 있다.

다. 기대효과

○ 기술적 측면

(1) 정확도가 향상된 정밀 퇴적물이동 모형을 수립할 경우 다음과 같은 효과를 기대할 수 있다.
- 난류에너지 계산능력을 향상시킬 수 있다.
- 쇄파현상을 수치모델로 구현 할 수 있다.
- 퇴적물 부유/이동 현상을 구현 할 수 있다.

○ 경제·산업적 측면

(1) 향후 연안지형변동 모델로 개발이 가능하다.
- 지형변동모델 개발 성공 시 자체모델을 사용하여 연안침식 문제 해결을 통해 손실을 저감할 수 있다.
- 모델 상업화를 통한 국내외 경제적 수익을 창출 할 수 있다.

○ 사회·문화적 측면

(1) 국내 연안침식 문제 해결을 통해 다음과 같은 사항을 기대할 수 있다.
- 선진 기술 개발로 해외 기술에 의존하지 않는 국내 설계를 통한 설계비 경감 및 국제 경쟁력을 강화할 수 있다.
- 향후 지형변동모델 개발 시 동해안의 심각한 연안침식 문제 해결을 통한 해안환경보전에 기여할 수 있다.

제 2 장

국내외 기술개발 현황

제 2 장 국내외 기술개발 현황

1. 국내외 연구동향

가. 선진국 동향

- Openfoam 모델은 최근 유체역학의 제 분야에 널리 사용되고 있으나 쇄파대 퇴적물이동 연구에 적용된 사례는 아직 발표된 바 없다.
- 최근(2017년) 영국의 Bath 대학(University of Bath)에서 Openfoam을 적용하여 쇄파대 연구를 수행하였으나 이것은 파랑모델이며 퇴적물이동 모델은 아니다.

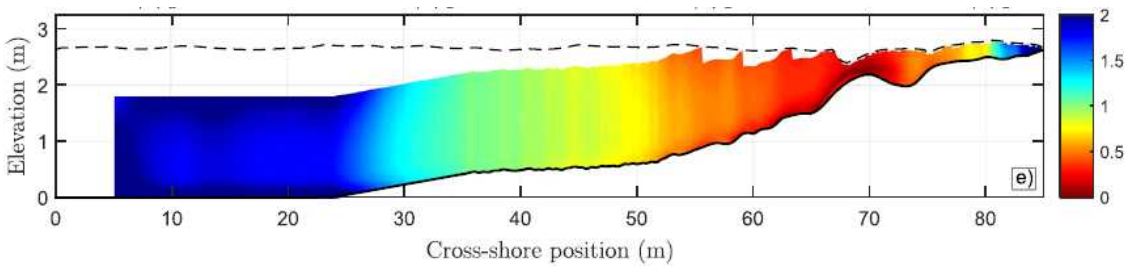


그림 3. Openfoam 모델을 적용한 쇄파대 파랑연구 (Martins et al., 2017).

- 미국 델라웨어 대학의 Tom Hsu 교수팀이 쇄파대 및 포말대 퇴적물이동 연구를 진행 중에 있으나 정밀 난류모델을 사용한 연구 성과가 최근 발표 되고 있다 (Kim et al., 2017).
- 미국 노스캐롤라이나 대학의 Alberto Scotti 교수팀의 경우 LES를 사용한 난류모델링 에 관한 연구를 수행중이나 아직 쇄파대 퇴적물이동에 적용하지는 않은 상태이다.

- 네덜란드 Deltares 연구소의 경우 퇴적물이동 모델링 분야 세계 최고의 권위를 가진 연구기관으로 Delft3D, XBeach 등의 모델을 개발하였으나 이 모델들은 광역 모델로서 쇄파대 정밀 퇴적물이동 연구에 적용하기는 아직 어려운 상황이다.

나. 국내 동향

- 현재 국내의 연안역 퇴적물이동모델 연구는 대부분 모델 범위 수 km 이상의 광역모델을 통한 연구에 집중되어 있다.
- 한 예로 국가 R&D인 “연안침식 대응기술 개발”과제의 경우 XBeach (Roelvink et al., 2009), Delft3D (Elias et al., 2000), UNIBEST (Ruggiero et al., 2010)등의 모델 등을 적용하여 연안역의 단기, 중기, 장기 퇴적물이동 현상을 연구하고 있으나 이 모델들은 모두 광역모델로서 해저면에서 퇴적물 부유현상을 구현하는 정밀 퇴적물이동 연구에 적합한 모델은 아니다.
- 경상대학교의 허동수 교수팀이 일본에서 개발한 Cadmas-Surf 모델에 기반한 쇄파대 파랑 및 퇴적물이동 모형을 개발하여 운영 중에 있으나 구조물 주변의 파랑변동에 중점을 둔 연구를 수행 중이며 이 모델을 통한 해저면 퇴적물 부유/이동 현상 및 이에 필수적인 난류에너지의 정확성은 검증 된 바가 없다.
- 관동대 이광호 교수팀(Lee et al., 2017) 은 최근 OpenFoam을 사용하여 연안구조물 주위에서 파랑변형을 모델링하였으나 이 역시 쇄파현상 및 퇴적물이동에 관한 연구는 아니다.

제 3 장

연구개발수행 내용 및 결과

제 3 장 연구개발수행 내용 및 결과

1. 연구 결과

가. LES 난류기법 도입

(1) OpenFoam 에 LES 난류 기법 적용

① OpenFoam CFD 모델 분석

- OpenFoam은 소스코드가 공개된 CFD 모델로서 많은 개발자들이 작성한 Solver 들의 패키지들로 구성된다.
- 따라서 OpenFoam의 장점은 높은 확장성으로서 Solver 들을 적절하게 사용하거나 또는 필요한 Solver를 직접 개발하여 목적에 맞는 수치모델 패키지를 수립하는데 유리하다.
- 그러나 높은 확장성은 반면 OpenFoam 모델 사용이 쉽지 않음을 반증한다.
- 본 연구의 주요 목표 중 하나는 이러한 OpenFoam의 Solver들을 적절히 결합하여 쇄파대 및 포말대의 파랑현상과 퇴적물부유/이동 현상을 구현하는 정밀 모델을 수립하는 것이다.
- LES 난류모델 Solver를 결합하여 파랑조건 하에서 발생하는 해저면 난류에너지 예측의 정확도를 향상시킬 수 있다.

② LES 난류모델

- LES (Large Eddy Simulation)(Lubins et al., 2006; Zhou et al., 2017) 은 난류모델링 기법으로 유체와 고체의 경계면 근처에서 발생하는 불규칙적인 유체운동인 난류현상을 수치모델을 통해 구현하는 방법이다.

- 자연에서 발생하는 유체운동은 그 규모의 차이가 너무 커서 모든 움직임을 하나의 방정식 (예를 들면 Navier-Stokes 방정식)으로 전부 풀어 낼 수 없으며 따라서 규모가 작은 유체 운동 (즉 난류)는 난류모델 기법을 통해 수치화 시키는 방법을 사용한다.
- 난류모델로 그 동안 계산의 효율성이 상대적으로 높은 RANS (Reynolds-Averaged Navier-Stokes) 모델이 널리 사용되어 왔으나 그 정확도가 의심 받아 왔다.
- RANS 모델은 유속을 평균값과 변동 값으로 나누어 ($u = U + u'$) Reynolds-Averaged 방정식을 구한다.

$$\text{㉮} \quad \frac{\partial U_i}{\partial t} + U_j \frac{\partial U_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial P}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\nu \frac{\partial U}{\partial x_j} - \overline{u_i' u_j'}\right)$$

- 식 ㉮를 계산하기 위해서 마지막 항인 Reynolds stress를 처리해야 하며 직접 계산이 어려운 관계로 Boussinesq approximation을 사용하여 수치화한다.

$$\text{㉯} \quad \overline{u_i' u_j'} = 2\nu_t S_{ij} - 2/3 k \delta_{ij}$$

- 식 ㉯에서 ν_t는 eddy viscosity, S_{ij}는 Strain rate tensor, k는 난류 에너지 (TKE), δ_{ij}는 크로네커 텐서이다.
- 즉 식 ㉯는 RANS 모델은 난류에너지가 평균유속에 비례한다는 가정 하에 난류에너지의 크기와 시공간 분포를 계산한다.
- 반면에 LES 는 기본개념에서 RANS 와 차이를 나타내는데 격자크기 (Filter) 보다 큰 규모의 유체움직임 (Large Eddy)을 직접 계산하여 적정 수준의 난류 운동을 모델로 직접 구현 한다는 점에서 모든 규모의 난류를 수치화 하는 RANS와 차별 된다.
- 이를 위하여 LES 는 Navier-Stokes 방정식에 Filter를 적용한 Filtered Navier-Stokes 방정식의 해를 구한다.

$$\text{㉰} \quad \frac{\partial \tilde{u}_i}{\partial t} + \frac{\partial}{\partial x_j}\left(\tilde{u}_i \tilde{u}_j\right) = -\frac{\partial \tilde{P}}{\partial x_j} + \nu \frac{\partial^2 \tilde{u}_i}{\partial x_j \partial x_j} - \frac{\partial\left(u_i u_j - \tilde{u}_i \tilde{u}_j\right)}{\partial x_j}$$

- 식 ㉰에서 $\tilde{u}$ 는 Filtered 유속 $(\tilde{u}(x)=\int u(x')G(x,x')dx')$ 이며 이것은 수치모델의 격자에 적용하는 Filter 보다 큰 규모의 운동을 직접 계산함을 의미한다.
- 식 ㉰의 마지막 항인 $\tilde{u}_i u_j - \tilde{u}_i \tilde{u}_j$ 은 $(\triangle_g)^2|\tilde{S}|$ 을 사용하여 수치화 한다.
- 여기서 $\triangle_g$는 격자크기로서 즉 격자크기보다 작은 규모의 운동은 식을 사용하여 수치화함을 의미한다.
- 이상에서 설명한 바와 같이 LES 는 모든 규모의 난류운동을 수치화 하는 RANS 에 비해서 적정규모의 난류운동을 직접 계산함으로 모델의 정확도를 향상 시킬 수 있다.
- 특히 해저면 퇴적물 부유와 같이 정밀한 난류운동에 의해 크게 영향 받는 현상규명을 위해서는 정확한 난류모델의 적용이 중요하다.
- 최근 선진국에서는 쇄파대 정밀 퇴적물이동 모델에 LES 를 적용하는 사례가 증가하고 있으나 국내에서는 아직 관련 연구가 부족하다.
- 이에 본 과제를 통해 RANS 대비 LES 의 우수성을 입증하고 향 후 연안역 퇴적물 이동 연구에 적극 적용하고자 한다.

(2) 난류 에너지 (TKE) 계산

① 난류 에너지 계산

- 난류에너지 (Turbulent Kinetic Energy, TKE)는 유체의 난류운동을 정량화 하는 가장 보편적인 방법이다.
- 따라서 TKE의 정확한 계산은 난류모델의 성능 검증과 해저면 퇴적물 부유 현상을 규명함에 있어서 가장 중요한 요소 중 하나이다.
- RANS의 경우 식 (2) 에 나타난 바와 같이 모델에서 TKE를 직접 계산한다.
- 그러나 LES 의 경우 모델에서 TKE를 직접 계산하지 않아 계산된 유속 해를 통해서 구해야 한다.
- $TKE=(\overline{(u')^2}+\overline{(w')^2})/2$ 에서 $u'=u-U$
- 여기서 u는 LES 모델의 해이다.
- 따라서 위 식에서 u'을 계산하기 위해서는 유속의 평균값인 U를 계산해야 한다.
- U를 계산하는 방법으로는 Emsemble Average나 Low pass filtering 의 방법들이 많이 사용된다.
- 그러나 이 방법들은 관측 자료와 같이 모집단이 큰 경우에 정확도가 향상하며

모델결과와 같이 모집단이 작은 경우는 정확도가 낮아진다.

- 본 연구에서는 이웃하는 격자 값들의 평균을 계산하는 Spatial averaging 방법을 적용하여 U 와 TKE 계산한다.

(3) RANS 기법과 성능비교

① LES 와 RANS 성능비교

- LES 와 RANS 의 성능을 비교하기 위하여 수치모델 상에서 난류가 잘 발생하는 쇄파조건 형성한다.
- 잠제를 설치하여 잠제 후면에서 발생하는 쇄파현상 비교한다.

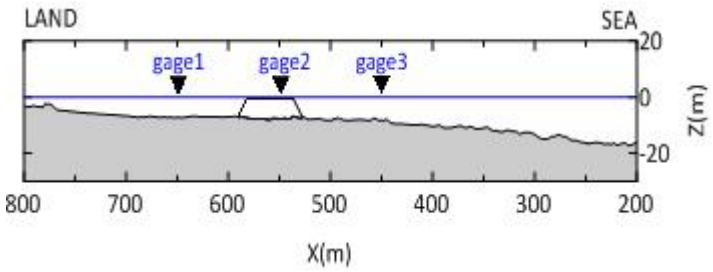
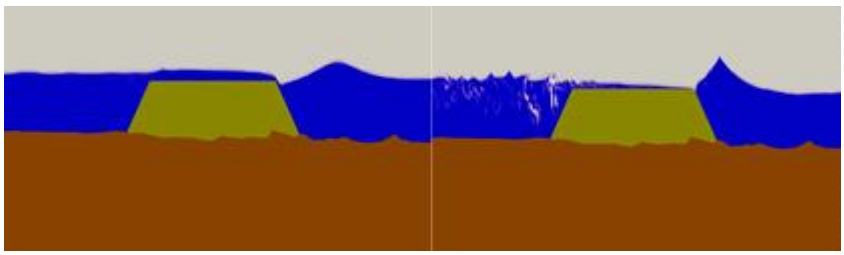


그림 4. 2차원(x-z)수치모델 구성도. Gage 1-3은 잠제 주변 모델자료 측정위치.

- 그림 4은 본 연구를 위해 작성한 2차원 OpenFoam 수치모델 구성도 이며 gage 1-3은 잠제 주변 모델자료 측정 위치를 나타낸다.



RANS LES

그림 5. 잠제 후면 쇄파현상 비교.

- 그림 5는 RANS 와 LES 간 잠제 후면에 발생하는 쇄파현상 비교한다. (두 모

델 간 파랑조건은 동일)

- 그림 5에서 나타난 바와 같이 RANS 기법을 사용하면 쇄파현상이 발생하지 않는 경우에도 LES 기법을 적용하면 활발한 쇄파현상 발생한다.
- 쇄파현상을 통해 발생하는 난류에너지는 LES 가 RANS 보다 높게 나타났다.
- 이 실험 결과를 검증할 관측 자료는 없으나 Chang and Hanes (2004)에 보고된 내용을 참조하면 비교하면 RANS기법의 경우 난류에너지를 실제 현상보다 작게 예측하는 경우가 발생하며 이에 따라 LES 의 상대적으로 높은 난류에너지는 실제현상에 보다 근접한 결과인 것으로 판단된다.
- 그림 6은 그림 1에 표시된 세 지점에서 측정한 파고 비교하며 잠제 후면인 Gage 1에서 쇄파현상으로 인해 불규칙한 파고 형태 관측되고 Gage 1에서 LES의 파고가 RANS 보다 높게 나타난다.

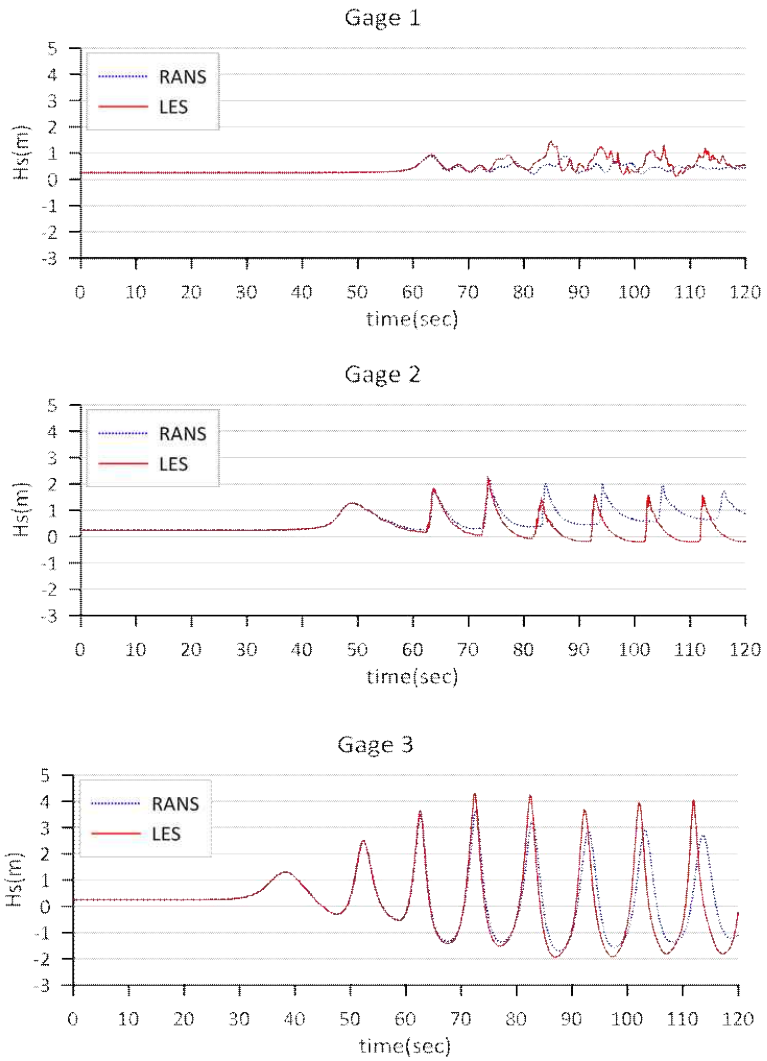


그림 6. 잠제 주변 3개의 Gage에서 측정한 파고 비교.

- 잠제 위인 Gage 2에서 RANS의 파고가 LES 보다 높게 나타나며 잠제 전면인 Gage 3에서 LES의 파고가 RANS 보다 높게 나타난다.
- 그림 2와 3의 결과를 분석하면 LES의 경우 더 많은 파랑 에너지가 잠제 위를 통과 한 것으로 판단되며 LES의 경우 잠제 후면에서 (Gage 1) 더 높은 파고와 더 활발한 쇄파현상이 관측 된다.
- RANS 의 경우 잠제 위에 (Gage 2) 더 많은 파랑 에너지가 축적된 것으로 판단되며 이는 파랑이 잠제를 통과하는 효율성이 RANS 가 더 낮음을 나타낸다.
- Gage 3에서 관측된 RANS 의 더 낮은 파고도 위의 결과를 뒷받침 하며 즉

잠제에 의해 통과되지 못한 파랑이 잠제에 의해 반사되어 잠제 전면에서의 파고를 낮춘 것으로 판단된다.

- 이상의 결과를 확인하기 위해 잠제 전후면(Gage 1 과 3)에서 TKE를 비교한다. (그림 7)
- 그림 7에서 잠제 전 (Gage 3) 에서는 RANS 의 TEK가 LES 보다 최대 약 10배 크게 나타나며 잠제 뒤 (Gage 1) 에서는 LES 의 TKE가 RANS 보다 최대 약 5배 크게 나타난다.
- 그림 7의 TKE 결과도 그림 2와 3의 결과와 일치하며 즉 잠제 후면 (Gage 1) 에서 더 높은 LES의 TKE는 활발한 쇄파 현상에 의해 강한 난류에너지가 발생함을 나타낸다.

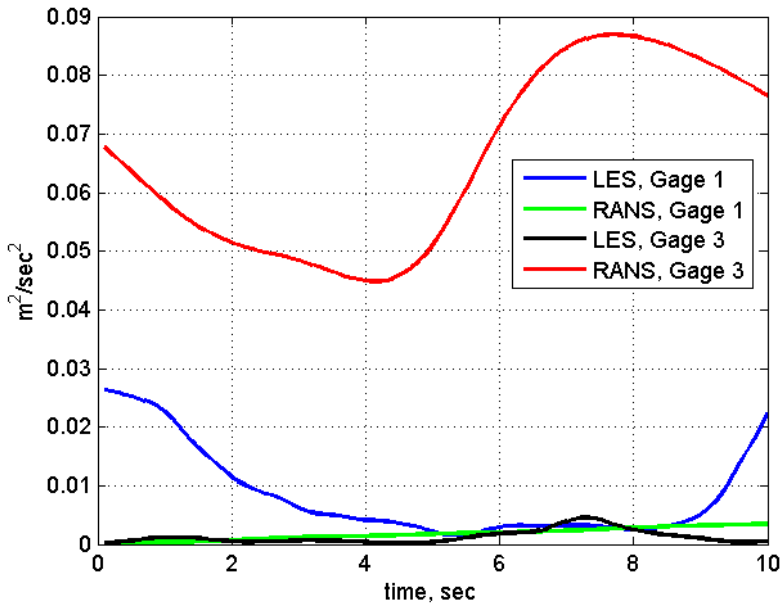


그림 7. 잠제 전후면 TKE 비교.

- 잠제 전면 (Gage 3)에서 RANS 의 더 높은 TKE는 정확한 파랑에너지가 잠제를 통과하지 못하고 반사되기 때문인 것으로 판단된다.

나. 쇄파대 파랑모델 수립

(1) VOF 기법 적용 쇄파현상 구현

① VOF 기법

- Volume of Fluid (VOF) 기법(Noh & Woodward, 1976)은 서로 다른 밀도를 갖는 두 유체 (예를 들어 물과 공기)의 경계면에 위치한 수치모델 격자에서 두 유체의 면적 비율을 실시간으로 계산하여 두 유체의 경계면 변화를 구현하는 기법으로 최근 연안역에 발생하는 중력파를 수치모델로 구현하는 데 많이 사용된다.
- VOF 기법의 장점으로 연안역 파랑 전파에 따른 여울(shoaling) 현상 및 쇄파현상이 구현가능하다.
- 그림 8는 VOF 기법을 적용한 연안역 중력파 구현의 한 예를 보여주며 잠제가 설치되지 않았을 경우 쇄파가 잘 발생하지 않으나 잠제를 설치하면 잠제 후면에서 강한 쇄파 현상이 발생함을 나타낸다.

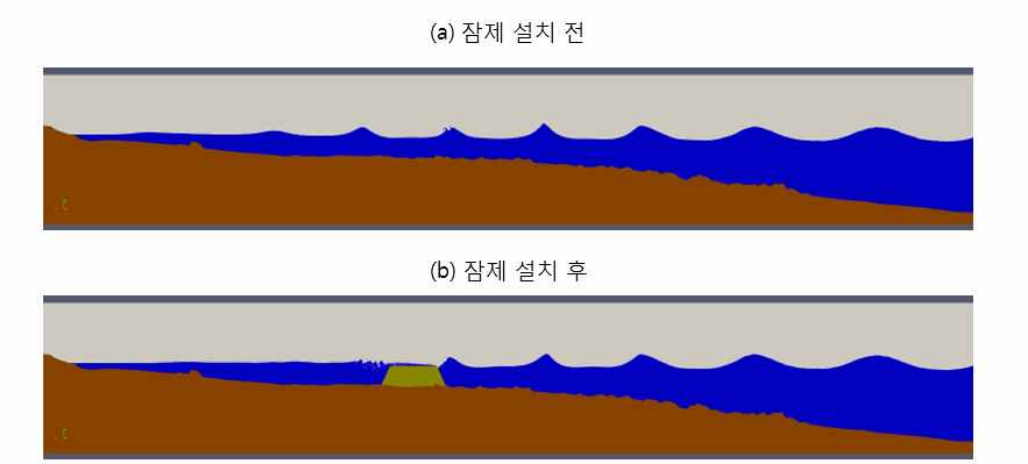


그림 8. VOF 기법에 의한 쇄파현상 구현 (a) 잠제 설치 전, (b) 잠제 설치 후.

(2) OpenFoam 적용 쇄파대 파랑모델 수립

① 파랑모델 대상지역

- 이번 장은 이전 장들에서 기술한 연구결과들을 도출하기 까지 OpenFoam을 적용한 연안역 파랑모델 수립과정을 기술한다.

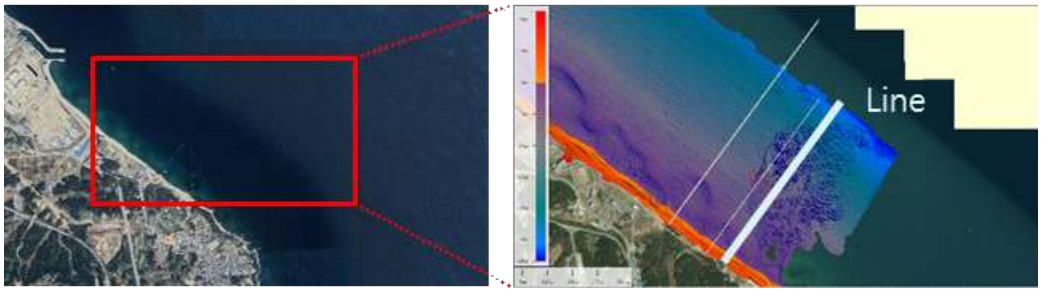


그림 9. 한국해양과학기술원 동해연구소 전면 후정해변 위치(좌) 및 수심도(우). 우측의 Line 은 OpenFoam 모델 설정 구간을 표시.

- 그림 9는 한국해양과학기술원 동해연구소가 위치한 후정해변 및 주변 수심관측 결과를 보여준다.
- 후정해변 남단에 국립해양과학교육관을 건설 중이며 교육관 해변보호를 위해 전면에 잠제건설을 계획 중에 있다.
- 본 연구는 잠제 건설 시 발생하는 파랑변화 예측을 위해 잠제를 통과하는 해안선 수직방향의 OpenFoam 모델 2차원 'Line'을 설정하였다. (그림 9)

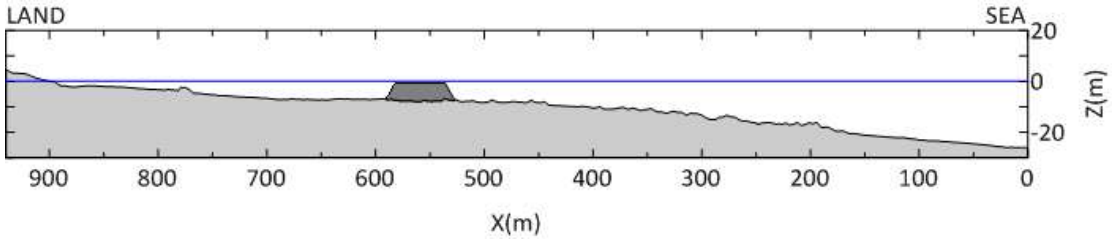


그림 10. 그림 6의 'Line' 상에 수립된 잠제가 설치된 경우의 2차원(x-z) 모델 영역. 파랑색 실선은 평균해수면 위치.

- 그림 10은 'Line'을 따라 수립된 2차원(x-z) 모델 영역과 잠제 및 평균해수면 위치를 나타낸다.
- 잠제는 해안선에서 수직으로 약 350m 떨어진 수심 10m 지역의 평균해수면 아래 약 0.5m에 건설될 예정이다.

② 모델격자 및 파랑조건

- 그림 11은 'Line' 위에 수립된 모델 전체격자망(위)과 잠제 주변 상세격자망(아래)를 보여준다.

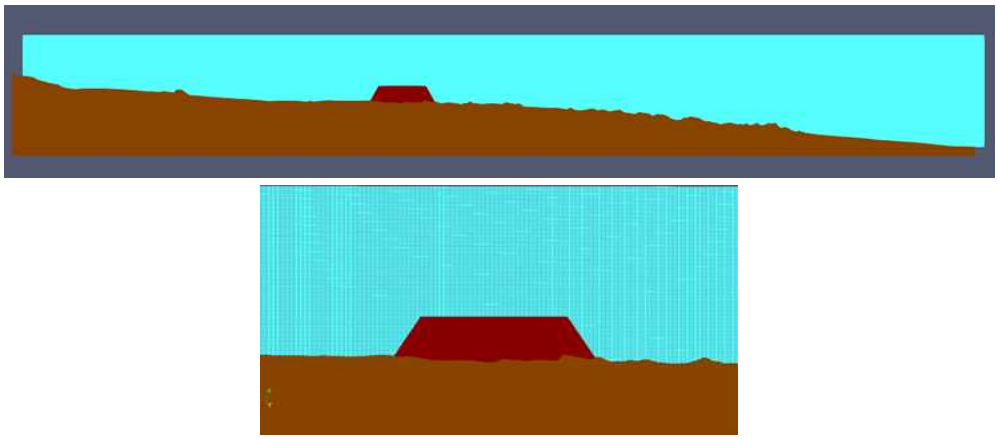


그림 11. 'Line' 위에 수립된 모델 전체격자망(위) 및 잠제 주변 상세격자망(아래)

- 격자크기는 수평/수직방향 모두 0.2m를 사용하였다 (격자수 4700 x 230 = 1,081,000개).
- 모델 시간간격은 최대 0.001초를 사용하였으며 예측 기간은 120초를 사용하였다.
- 파랑조건 : 규칙파(Cnoidal), Hs = 4.0m, T=10초를 사용하였다.

③ 파랑모델 수립결과

- 규칙파 생성 후 약 70초 이후에 안정된 결과를 보여준다 (그림 3).
- 잠제 후면 쇄파현상 및 TKE 생성 결과 등은 이전 장에서 설명한다.

다. 퇴적물 이동모델 수립

(1) 라그랑지안 입자추적 모델 수립

① 모델 수립

- 오일러리안(Eulerian)식 퇴적물 농도 계산은 미국 델라웨어 대학의 Tom Shu 교수 연구팀에 의해서 OpenFoam 모델에 결합된다 (Kim et al., 2017).
- 그러나 위 방법은 퇴적물을 또 다른 유체층으로 표시하여 진흙과 같은 cohesive 퇴적물 계산은 유용하나 모래와 같이 굵은 입자의 퇴적물 움직임을 표현하기에는 한계가 있다.
- 이에 본 연구에서는 Maxey and Riley(1984) 식을 토대로 한 라그랑지안 방식 퇴적물 입자 추적 모델을 수립한다.

$$\text{㉮}\quad \frac{\rho_p}{\rho}\frac{d\vec{V}_p}{dt} = -\left(\frac{\rho_p}{\rho}-1\right)g\vec{n} + V\frac{D\vec{u}_f}{Dt} + C_p V\left(\frac{D\vec{u}_f}{Dt} - \frac{d\vec{V}_p}{dt}\right)$$
$$+\frac{1}{2}C_D A\left|\vec{u}_f - \vec{V}_p\right|\left(\vec{u}_f - \vec{V}_p\right) + \frac{1}{2}C_L A\left(\left|\vec{u}_f - \vec{V}_p\right|_t^2 - \left|\vec{u}_f - \vec{V}_p\right|_b^2\right)\vec{n}$$

- Maxey and Riley 의 식 ㉮에서 ρ 와 ρ_p는 각각 유체와 퇴적물 입자의 밀도, $\vec{V}_p$는 라그랑지안 퇴적물입자 속도벡터, g는 중력가속도, V는 퇴적물입자 부피, $\vec{n}$은 수직방향 단위 벡터, $\vec{u}_f$는 유체 속도벡터, C_p는 added mass 계수, C_D는 견인력(drag force) 계수, A는 퇴적물입자 단면적, C_L 는 양력(lift force) 계수이다.
- 식 ㉮는 퇴적물 입자의 라그랑지안 속도는 1) 부력, 2) 입자 주변에 발생하는 압력변화, 3) 퇴적물 입자 이동 시 주변 유체를 밀어내며 발생하는 added mass, 4) 입자의 견인력 및 4) 양력의 합으로 결정됨을 의미한다.
- 라그랑지안 입자추적 모델을 수립하기 위해서 식 ㉮를 적용하여 해저면에 위치한 모래 입자들의 속도를 계산한 후 이 속도를 적분하여 각 입자들의 시간별 위치변화를 추적한다.
- 따라서 입자추적 모델을 통해 쇄파 시 해저면 상에서 발생하는 강한 난류구조에 의한 퇴적물의 부유현상을 정확하게 계산 가능하다.

(2) OpenFoam 과 결합

① 모델 결합

- 식 ㉮를 토대로 한 입자추적 모델 수립 및 OpenFoam과 결합
- Internal coupling이 아닌 OpenFoam의 결과인 수리자료를 토대로 입자추적을 계산하는 External Coupling을 사용하였다.

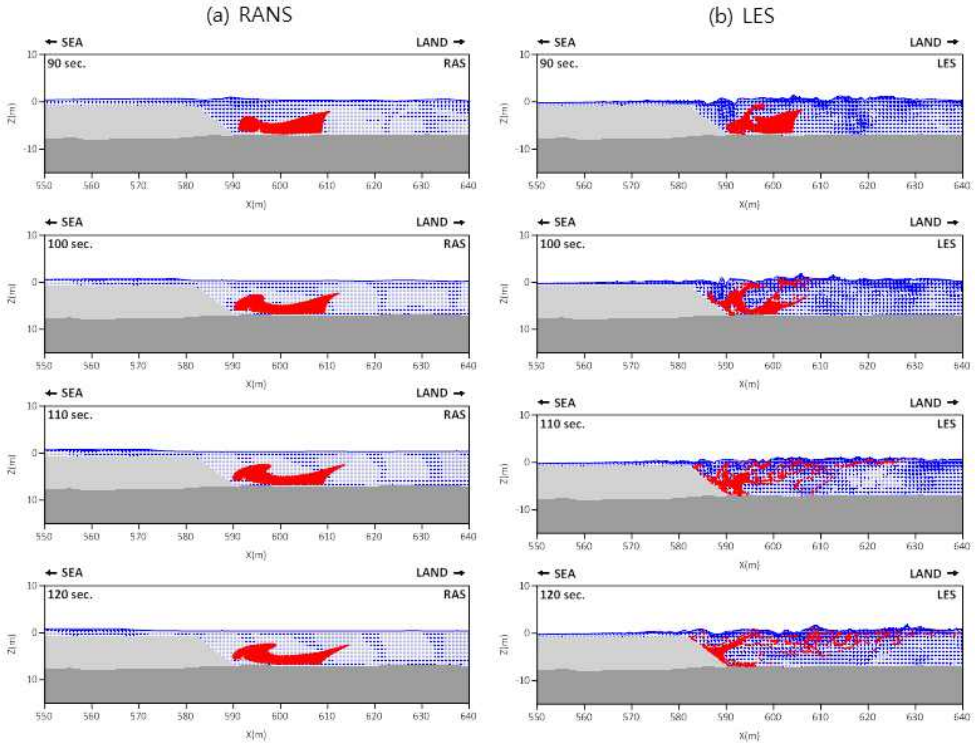


그림 12. 입자추적 모델 결과 (a) RANS , (b) LES

- 그림 12는 RANS 와 LES 간 입자추적 모델결과 비교하면 RANS 의 경우 모래 입자가 활발하게 분산되지 않으나 LES 에 의해 계산된 모래 입자는 잡제 후면에서 활발한 움직임을 보여준다.

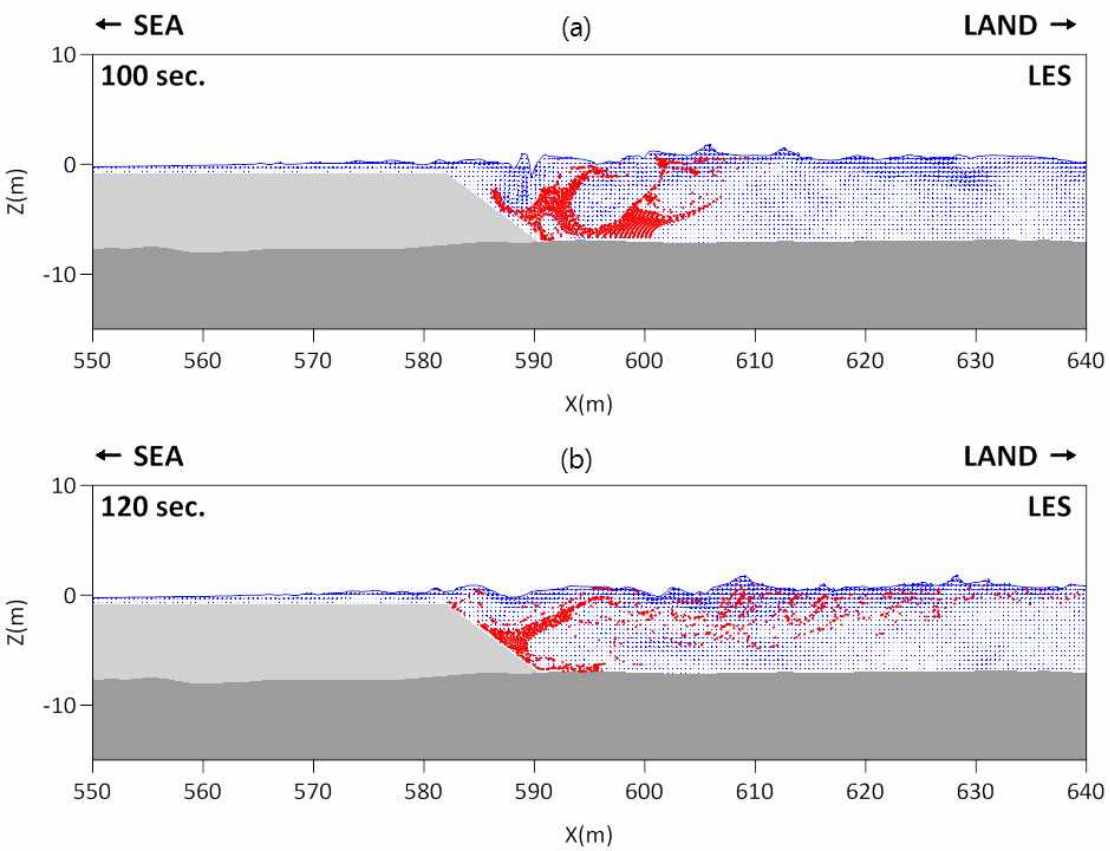


그림 13. LES 의 입자추적 모델 결과 (그림 12 (b) 확대)

- 그림 13은 그림 12(b)에 표시된 LES의 모래입자 분포를 확대하여 보여주며 입자 분포를 통해 잠제 후면에 강한 난류와동 (turbulent eddy)이 형성됨을 확인할 수 있다.
- 이 와동에 의해 해저면 주위의 모래 입자가 해수면 근처로 상승하고 그 후 강한 난류운동에 의해 강하게 확산됨을 확인된다.
- 이상의 결과를 통해 LES 의 경우 난류에너지와 난류에 의한 해수 유동을 보다 잘 구현할 수 있음을 확인할 수 있다.

2. 연구 개발 내용

자문보고서에 기반한 OpenFoam 수립 및 모델개발 내용

가. OpenFoam 소개 및 설치

(1) 개요

- OpenFOAM(Open Field Operation and Manipulation)은 Source Code가 공개된 전산유체역학(CFD, Computational Fluid Dynamics) Code로, 1989년 영국의 Imperial College의 박사과정이던 Hrvoje Jasak과 Henry Weller에 의해 FOAM이라는 이름으로 개발이 시작되었으며, 2004년 Henry Weller가 OpenCFD Ltd.라는 회사를 설립하고 OpenFOAM이라는 이름으로 코드를 공개한다.
- 이후 OpenFOAM은 OpenFOAM Foundation이라는 비영리 회사와 OpenFOAM Extend-Project라는 비영리 커뮤니티 두 곳에 의해 개발 및 관리가 이루어지고 있다.
- 본 연구에서는 이들 중 OpenFOAM Foundation의 OpenFOAM 5.x 버전을 적용한다.
- OpenFOAM은 라이브러리 형태로 개발되며 실제 특정문제를 해결하기 위한 Solver는 독자적으로 개발할 수 있는 환경을 제공한다.
- 이 라이브러리들은 해석하고자 하는 편미분방정식과 최대한 유사하게 만들어져 있어 지배방정식을 잘 이해하고 있다면 C++ 언어에 대해 자세히 알지 못하더라도 쉽게 소스코드를 제어할 수 있다(표 1).
- OpenFOAM은 또한 C++ 라이브러리들의 소스파일 뿐 아니라 사전에 컴파일된 많은 어플리케이션들을 함께 제공하며 이는 Solver와 Utility로 구분된다. 기본적으로 제공되는 표준 Solver들은 특정문제에만 적용되게 만들어져 있으며, 비압축성유동, 압축성유동, 화학반응, 다상유동, 구조해석, 전자기장해석, 분자동역학해석 등 60여 가지를 지원하고, Utility들은 전·후처리, 데이터 파일관리, 병렬연산 등을 지원한다.

표 1. OpenFOAM 내 편미분방정식 코딩방법

수학식	$\frac{\partial \rho U}{\partial t} + \nabla \cdot \Phi U + \nabla \cdot \mu \nabla U = -\nabla p$
입력코드	solve (fvm::dd t(rho,U) + fvm::div(phi,U) - fvn::laplacian(mu,U) = -fvc::grad(p));

- OpenFOAM은 전·후처리 환경을 제공하며 전·후처리 및 모델 수행 구조의 개요는 그림 14와 같다.

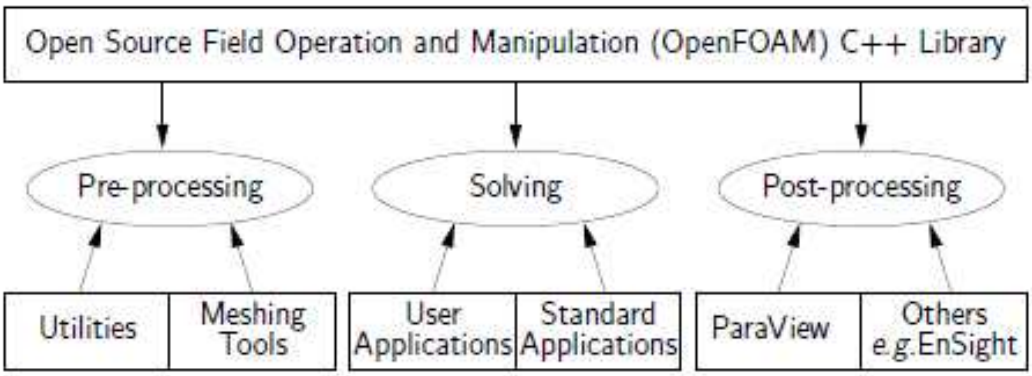


그림 14. OpenFOAM 수행 구조

(2) OpenFoam 설치

- CentOS 7 시스템 환경하에서 OpenFOAM 5.x의 설치과정은 다음과 같음.

① administrator 모드 입력한다.

```
[user@localhost ~]$ su
Password: XXXXXXXXX
```

② 필수 팩키지 설치한다.

```
[root@localhost user]# yum groupinstall 'Development Tools'
[root@localhost user]# yum install zlib-devel libXext-devel \
libGLU-devel libXt-devel libXrender-devel libXinerama-devel \
libpng-devel libXrandr-devel libXi-devel libXft-devel \
libjpeg-turbo-devel libXcursor-devel eadline-devel ncurses-devel \
python python-devel cmake qt-devel qt-assistant \
mpfr-devel gmp-devel
[root@localhost user]# yum upgrade
[root@localhost user]# exit
```

③ 터미널을 종료하고 새로운 터미널창을 오픈한다.

④ OpenFOAM 파일 다운로드 한다.

```
[user@localhost ~]$ cd ~
[user@localhost ~]$ mkdir OpenFOAM
[user@localhost ~]$ cd OpenFOAM
[user@localhost ~]$ git clone \
https://github.com/OpenFOAM/OpenFOAM-5.x.git
[user@localhost ~]$ git clone \
https://github.com/OpenFOAM/ThirdParty-5.x.git
```

⑤ ThirdParty-5.x 폴더 내에 필요 추가 팩키지 다운로드 및 설치한다.

```
[user@localhost ~]$ cd ThirdParty-5.x

[user@localhost ~]$ mkdir download

[user@localhost ~]$ wget –P download \

https://www.cmake.org/files/v3.9/cmake-3.9.0.tar.gz

[user@localhost ~]$ wget –P download \

https://github.com/CGAL/cgal/releases/download/releases%2FCGAL-4.10/CGAL-4.10.tar.xz

[user@localhost ~]$ wget –P download \

https://sourceforge.net/projects/boost/files/boost/1.55.0/boost_1_55_0.tar.bz2

[user@localhost ~]$ wget –P download \

https://www.open-mpi.org/software/ompi/v2.1/downloads/openmpi-2.1.1.tar.bz2

[user@localhost ~]$ wget –P download \

http://www.paraview.org/files/v5.4/ParaView-v5.4.0.tar.gz

[user@localhost ~]$ tar -xzf download/cmake-3.9.0.tar.gz

[user@localhost ~]$ tar -xJf download/CGAL-4.10.tar.xz

[user@localhost ~]$ tar -xjf download/boost_1_55_0.tar.bz2

[user@localhost ~]$ tar -xjf download/openmpi-2.1.1.tar.bz2

[user@localhost ~]$ tar -xzf download/ParaView-v5.4.0.tar.gz
--transform='s/ParaView-v5.4.0/ParaView-5.4.0/'

[user@localhost ~]$ cd ..
```

⑥ 디폴트 Boost, CGAL 버전 수정한다.

```
[user@localhost ~]$ sed -i -e 's/\(boost_version=\)boost-system/\1boost_1_55_0/'
OpenFOAM-5.x/etc/config.sh/CGAL

[user@localhost ~]$ sed -i -e 's/\(cgal_version=\)cgal-system/\1CGAL-4.10/'
OpenFOAM-5.x/etc/config.sh/CGAL
```

⑦ OpenFOAM 설치를 위한 환경변수 설정한다.

```
[user@localhost ~]$ source $HOME/OpenFOAM/OpenFOAM-5.x/etc/bashrc
WM_LABEL_SIZE=64 WM_MPLIB=OPENMPI FOAMY_HEX_MESH=yes
```

⑧ .bashrc 파일내에 다음 내용을 추가한다.

```
[user@localhost ~]$ echo "alias of5x='source
\$HOME/OpenFOAM/OpenFOAM-5.x/etc/bashrc $FOAM_SETTINGS'" >> $HOME/.bashrc
```

⑨ 터미널을 종료하고 새로운 터미널창을 연 후 다음을 입력하여 OpenFOAM 5.x의 환경변수를 활성화 한다.

```
[user@localhost ~]$ of5x
```

⑩ 후처리 프로그램 paraview를 설치한다.

㉮ cmake 3.x을 설치한다.

```
[user@localhost ~]$ cd $WM_THIRD_PARTY_DIR
[user@localhost ~]$ ./makeCmake > log.makeCmake 2>&1
[user@localhost ~]$ wmRefresh
```

㉯ open-mpi를 설치한다.

```
[user@localhost ~]$ cd $WM_THIRD_PARTY_DIR
[user@localhost ~]$ ./Allwmake > log.make 2>&1
[user@localhost ~]$ wmRefresh
```

㉰ paraview를 설치한다.

```
[user@localhost ~]$ cd $WM_THIRD_PARTY_DIR
[user@localhost ~]$ ./makeParaView -mpi -python -qmake $(which qmake-qt4) >
log.makePV 2>&1
[user@localhost ~]$ wmRefresh
```

⑪ OpenFOAM을 설치한다.

```
[user@localhost ~]$ cd $WM_PROJECT_DIR
[user@localhost ~]$ ./Allwmake -j 4 > log.make 2>&1
[user@localhost ~]$ ./Allwmake -j 4 > log.make 2>&1
```

⑫ OpenFOAM 설치 확인은 다음과 같이 한다.

```
[user@localhost ~]$ icoFoam –help
options:
  -case <dir>           specify alternate case directory, default is the cwd
  -fileHandler <handler>
                        override the fileHandler
  -listFunctionObjects
                        List functionObjects
                                        |
```

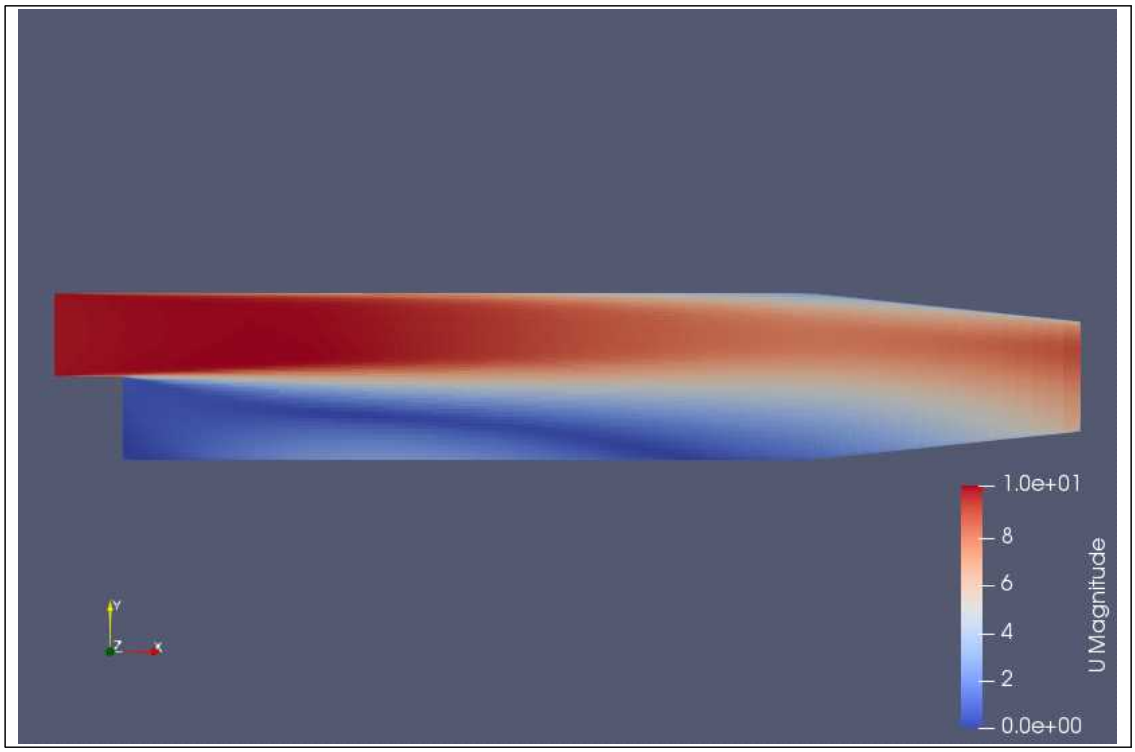
(3) OpenFOAM 예제 Test

① OpenFOAM의 설치를 확인하기 위해 tutorial 내의 pitzDaily case를 test한다.

```
[user@localhost ~]$ cd $FOAM_RUN
[user@localhost ~]$ cp –r $FOAM_TUTORIALS/incompressible/simpleFoam/pitzDaily ./
[user@localhost ~]$ cd pitzDaily
[user@localhost ~]$ blockMesh
[user@localhost ~]$ checkMesh
[user@localhost ~]$ simpleFoam
[user@localhost ~]$ paraFoam
```

② simpleFoam 수행과정은 다음과 같다.

③ paraFoam 수행 예는 다음과 같다.



나. OpenFoam 격자 생성

(1) STL 파일

① OpenFOAM 모델에서 구조물 및 지형정보는 STL 파일 구조로 입력 가능하며, STL 파일의 개요 및 기본구조는 다음과 같다.

② STL 개요

- STL은 STereoLithography의 약자로 물체의 외형정보를 저장하는 3D system 사가 제작한 파일 형식으로 3차원의 표면정보를 제공해주며 Ascii, Binary의 2 가지 형태로 구분된다.

③ STL 구조

- 일반적인 3D 데이터에서 3D 형상을 구성하는 최소단위는 삼각형이나 사각형이며, STL 3D 데이터에서 형상을 구성하는 최소단위는 항상 삼각형이며 이를 패싯(Facet)이라 한다.
- STL은 3D 형상을 구성하는 수많은 패싯들로 구성되어 각 패싯에 대한 정보를 가지며, 패싯은 세 정점과 삼각형을 이루는 면의 법선 벡터로 구성됨. 오류가 없는 STL 포맷을 형성하기 위해서는 물체를 구성하는 패싯들이 오른손법칙과 Vertex-to-Vertex 규칙을 만족하여야만 함. 즉, STL 데이터에서는 법선방향이 항상 외부(바깥)쪽을 향해야 한다.

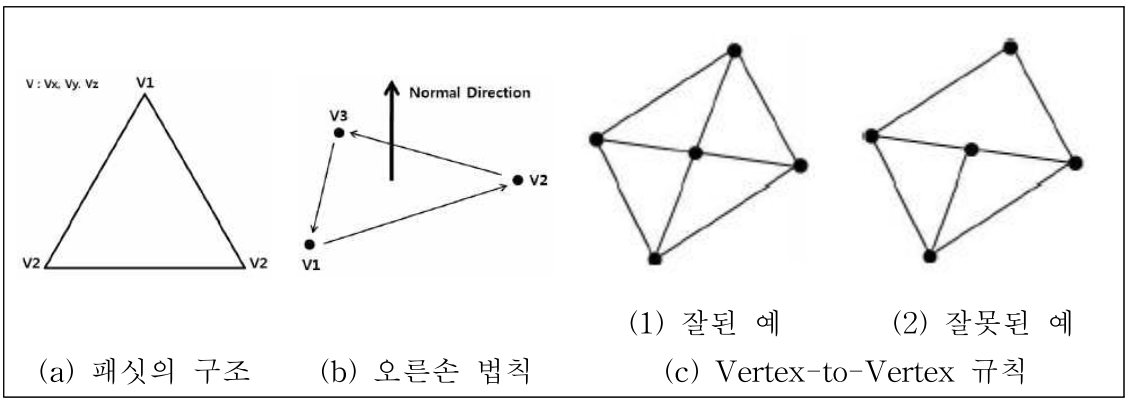


그림 15. STL 구조

- STL 정보는 삼각형의 정점좌표(Vertex)와 패싯을 이루는 법선벡터(Facet

Normal)로 구성되며, 하나의 패싯은 Vertex-to-Vertex 규칙에 맞추어 모든 패싯의 정점은 반드시 이웃한 패싯의 정점과 만나 연결되어야만 한다.

(2) GMSH 프로그램을 이용한 단면지형 STL 파일 제작

① GMSH

- OpenFOAM의 전처리 단계로 구조물에 대한 격자변환에는 Ansys, Salome, ideas, Fluent case, CFX, Star-CD, Gambit, Gmsh 등 다양한 툴이 존재함. 이 중 Gmsh는 1997년부터 Chrisophe Geuzaine와 Jean-François Remacle가 개발하여 무료로 공개한 유한요소해석에 적합한 격자망을 만들어 주는 공개 프로그램이다.
- Gmsh는 기하모델을 표현하기 위해 Boundary representation(b-rep)법을 사용하며, 기하모델은 보텀업(bottom-up) 모델링 기법을 따라 점-선-면-입체의 순으로 형성하게 됨. 또한, 텍스트 형태인 Gmsh 자체의 스크립트 언어를 사용하여 변수를 제언하는 방법으로 기하 모델의 형성이 가능하며, 요소망을 만드는 방법은 Characteristic length에 의해 제어되며, 기하의 형태에 의해 구조화된 요소망, 비구조화된 요소망을 모두 만들 수 있으나, 내부적으로는 모두 비구조화된 요소망으로 인식한다.
- Gmsh는 http://gmsh.info//#Download 에서 다운로드하여 설치할 수 있으며 Gmsh를 사용하여 만든 유한요소망은 STL 파일로 출력 가능하며 이를 이용하여 단면 및 구조물의 지형정보를 반영한 STL 파일을 구축하여 OpenFOAM에 입력·적용할 수 있다.

② 후정 단면 STL 파일구축

1) 점(point) 및 선(line) 정보 생성

- Gmsh 프로그램을 이용하여 기하모델을 생성하기 위해서는 점-선-면을 순차적으로 만드는 bottom-up 방법을 적용하므로 먼저 점을 생성해야만 함(그림 16). 그렇지만, 지형정보의 경우 단면지형이라 할지라고 많은 양의 데이터가 존재하므로 이를 직접 입력함은 시간이 많이 소모되므로 Pyhton 프로그램을 이용하여 점(point) 및 각 점 간의 선(line) 연결정보를 생성하는 script를 이용하여 geo 파일을 생성·적용한다(그림 17).

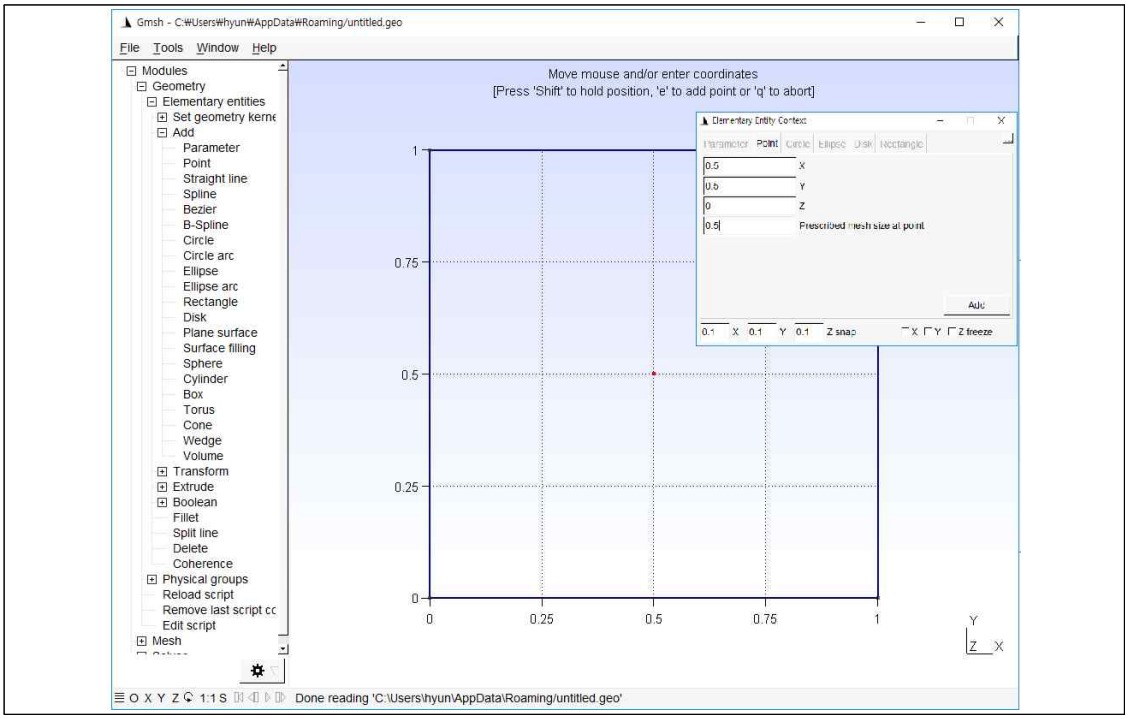


그림 16. Gmsh point 입력 예

```
sec_lc = 1.0;
Point(1) = { 0.00000000, 0.00000000, -30.00000000, sec_lc};
Point(2) = { 0.00000000, 0.00000000, -25.97400000, sec_lc};
Point(3) = { 1.00000000, 0.00000000, -25.97400000, sec_lc};
Point(4) = { 2.00000000, 0.00000000, -25.97400000, sec_lc};
Point(5) = { 3.00000000, 0.00000000, -25.97400000, sec_lc};
                                    |
Point(943) = { 939.00000000, 0.00000000, 4.42300000, sec_lc};
Point(944) = { 940.00000000, 0.00000000, 4.53000000, sec_lc};
Point(945) = { 940.00000000, 0.00000000, -30.00000000, sec_lc};
Line(1) = {1,2};
Line(2) = {2,3};
Line(3) = {3,4};
Line(4) = {4,5};
Line(5) = {5,6};
                                    |
Line(943) = {943,944};
Line(944) = {944,945};
Line(945) = {945,1};
```

그림 17. geo 파일 구축 예(point & line)

- 적용된 script는 그림 18과 같으며, 사용방법은 다음과 같다.

python dat2gmsh.py sect.asc.geo(지형정보 파일)

```
# Python Program for quick gmsh grid gen

import string,sys

def convert_pts_togmsh(filename,outputname,startpoint):

    # Read in data using this bit
    fin=open(filename, 'r')
    i=0
    x = []; y = []; z = [];

    lines = fin.readlines()

    for line in lines:
        # comments indicated by #
            data = str.split(line)
            if data[0] != '#' or data[0] != '':
                i=i+1
                x.append(float(data[0]))
                y.append(float(data[1]))
                z.append(float(data[2]))
                n_lines = int(i)
            else:
                break


    # Write data out with this;

    fout = open(outputname,'w')

    lc_name = "%s_lc" % filename[0:3]
    # Format
    # Point(1) = {0, 0, 0, lc};
    fout.write("%s = 1.0;\n" % lc_name)
    j = startpoint
    for i in range(n_lines):
        outputline  = "Point(%i) = { %8.8f, %8.8f, %8.8f, %s};\n " \
                                    % (j,x[i],y[i],z[i],lc_name )
        j = j + 1
```

```
            fout.write(outputline)

        # gmsh bspline format
        # Write out splinefit line
    #   fout.write("line(%i) = {%i:%i};\n" \
    #               % (startpoint,startpoint,startpoint+n_lines-1))
        j = startpoint
        for i in range(n_lines-1):
            outputline  = "Line(%i) = {%i,%i};\n" \
                                    % (j,j,j+1)
            j = j + 1
            fout.write(outputline)

        fout.write("Line(%i) = {%i,%i};\n" \
                    % (startpoint+n_lines-1,startpoint+n_lines-1,startpoint))
    #
        fout.close
        fin.close

    def main():
      inputfile = sys.argv[1]
      print(sys.argv[1])
      outputfile = inputfile+".geo"
      print(outputfile)
      convert_pts_togmsh(inputfile,outputfile,1)

    if __name__ == "__main__":
        main()
```

그림 18. dat2gmsh.py

- script로 생성된 geo 파일(sect.asc.geo)을 Gmsh로 불러들이면 다음과 같다.

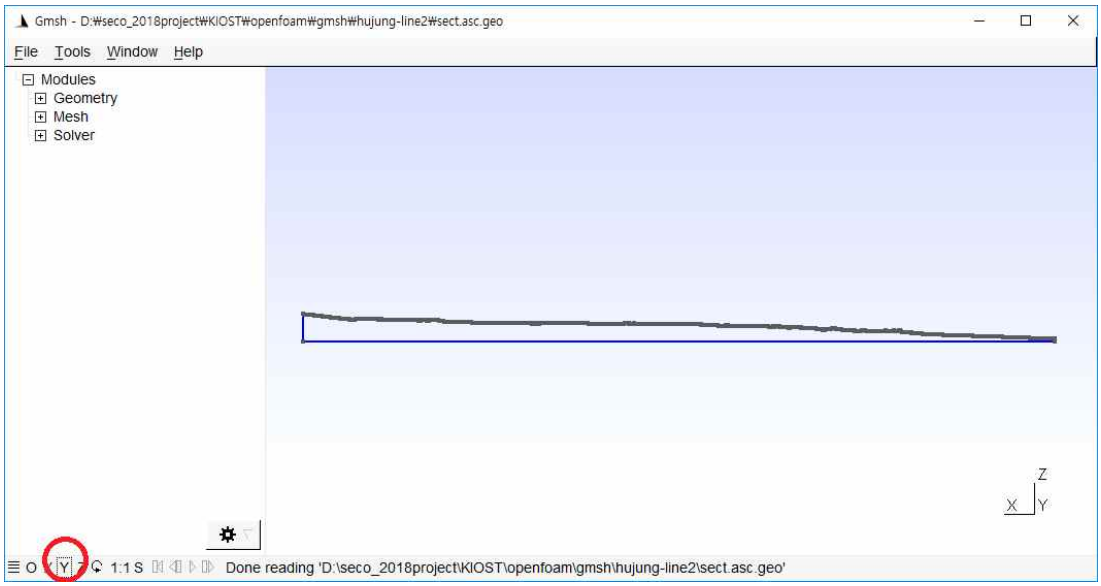


그림 19. 후정 단면 point & line 정보

2) 면(surface) 정보 생성

- Gmsh 프로그램을 이용하여 면 정보를 생성하는 과정은 다음과 같다.
- Gmsh 내 좌측 정보창에서

㉮ Modules-Geometry-Eelementary entities-Add-Plane Surface를 클릭

㉯ 우측 화면에서 마우스로 선(파란색)을 클릭

㉰ 선택된 선이 붉은색으로 변화되면 키보드에서 'e' 를 클릭

㉱ 이후 'q' 를 클릭하면 면 정보가 생성되며 면내에 점선이 나타난다.

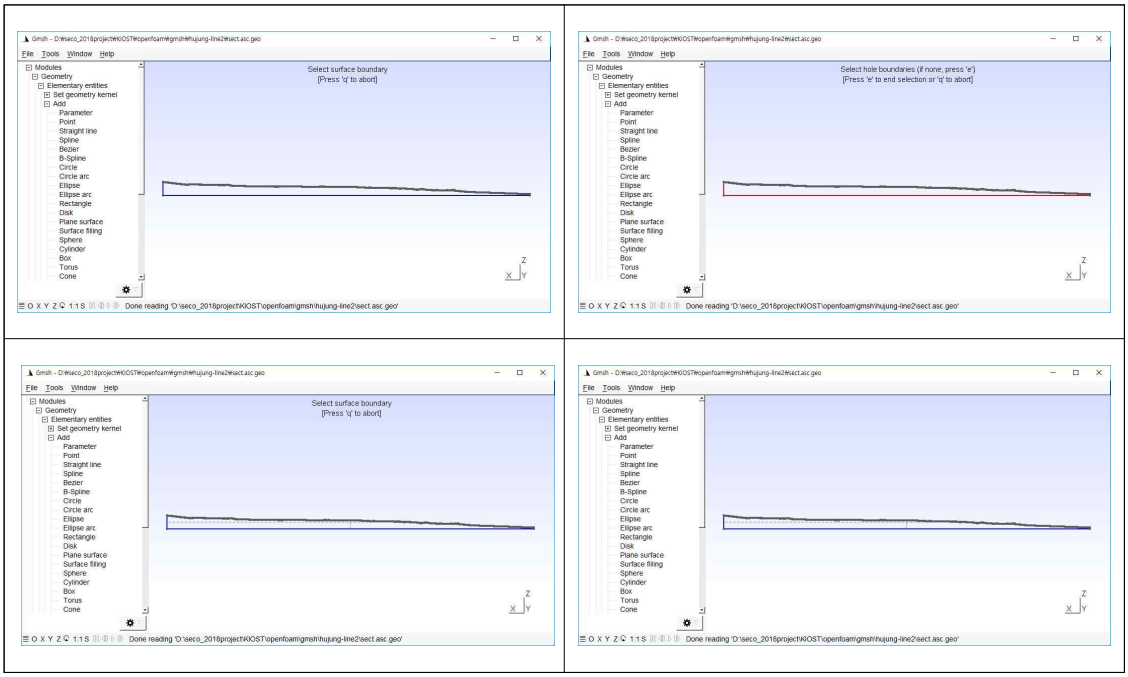


그림 20. 면 정보 생성 과정

- Gmsh 내 좌측 정보창에서 Geometry-Edit script를 클릭하면 Line Loop 및 면(Surface) 정보를 확인할 수 있다.

```
//+
Line Loop(1) = {945, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26,
27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55,
56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84,
85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110,
111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133,
134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156,
157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179,
180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202,
203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225,
226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248,
249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271,
272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294,
295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317,
318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340,
341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363,
364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386,
387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409,
410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432,
433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455,
456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478,
```

```
479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501,
502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524,
525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547,
548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570,
571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593,
594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616,
617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639,
640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662,
663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685,
686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708,
709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731,
732, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754,
755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777,
778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800,
801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822, 823,
824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846,
847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869,
870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 889, 890, 891, 892,
893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915,
916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938,
939, 940, 941, 942, 943, 944};
//+
Plane Surface(1) = {1};
```

그림 21. Line Loop 및 Surface 정보

3) 격자(Mesh) 생성

- OpenFOAM에 적용하기 위해서는 3D 구조의 STL 파일을 생성해야 한다.
- Modules-Geometry-Elementary entities-Extrude-Translate를 선택한다.
- Elementary Operation Context에서 DY = 1, Mesh layer = 1, Extrude mesh를 선택한 후 마우스를 이용하여 격자망을 선택하면 격자망이 붉은색으로 변환 된다.
- Elementary Operation Context창을 닫고 키보드에서 'e' 와 'q' 를 클릭한다.
- Gmsh Gui 창의 좌하단부에서 O X Y Z 중 Z를 마우스로 클릭하면 extrude된 파일 형태를 볼 수 있다.
- Gmsh 내 좌측 정보창에서 Modules-Geometry-Mesh-3D를 선택하면 면내에 유한요소 격자가 형성된다.

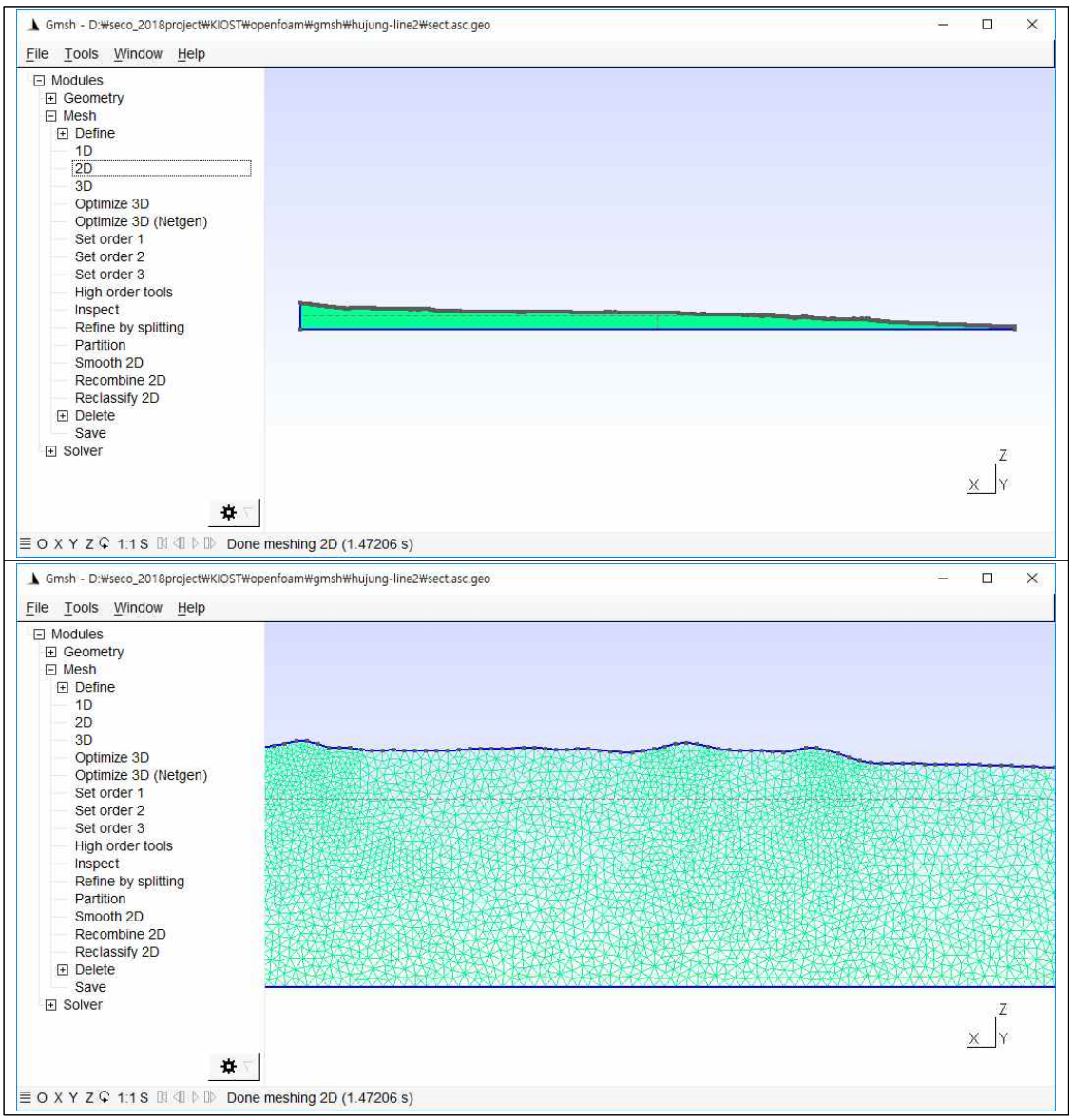
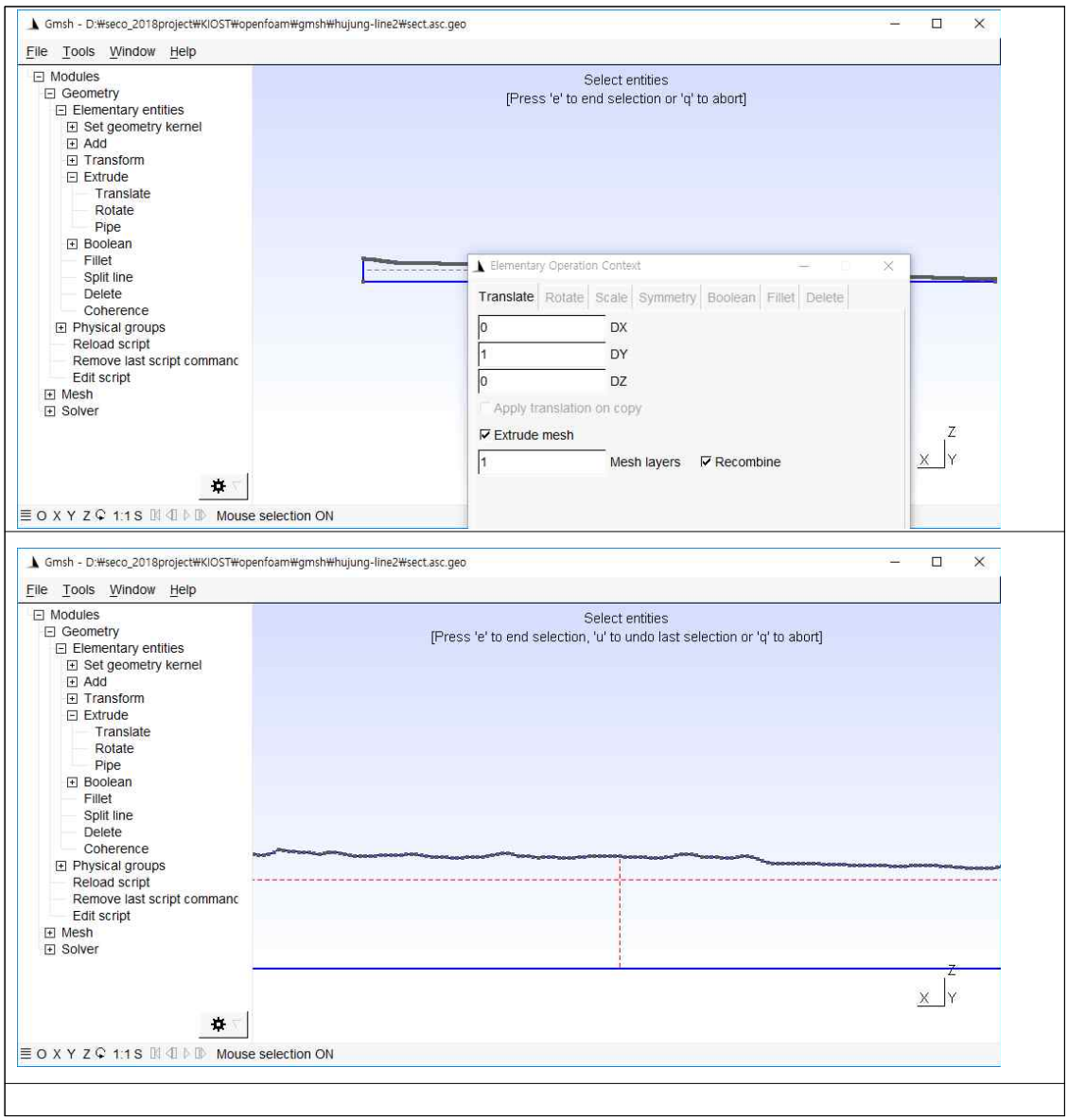


그림 22. 유한 요소 격자

- 생성된 유한요소 격자는 x-z 2D 격자이며, 이를 STL 파일로 변환하여 OpenFOAM에 적용하기 위해서는 3D 구조 파일로 변경해야 한다.
- Modules-Geometry-Elementary entities-Extrude-Translate를 선택한다.
- Elementary Operation Context에서 DY = 1, Mesh layer = 1, Extrude mesh를 선택하고 창을 닫는다.
- 마우스를 이용하여 면 중심의 점선을 선택하면 점선이 붉은색으로 변환됨.

- 키보드에서 ‘e’ 와 ‘q’ 를 클릭한다.
- Gmsh Gui창의 좌하단부에서 O X Y Z 중 Z를 마우스로 클릭하면 extrude된 형태를 볼 수 있다.



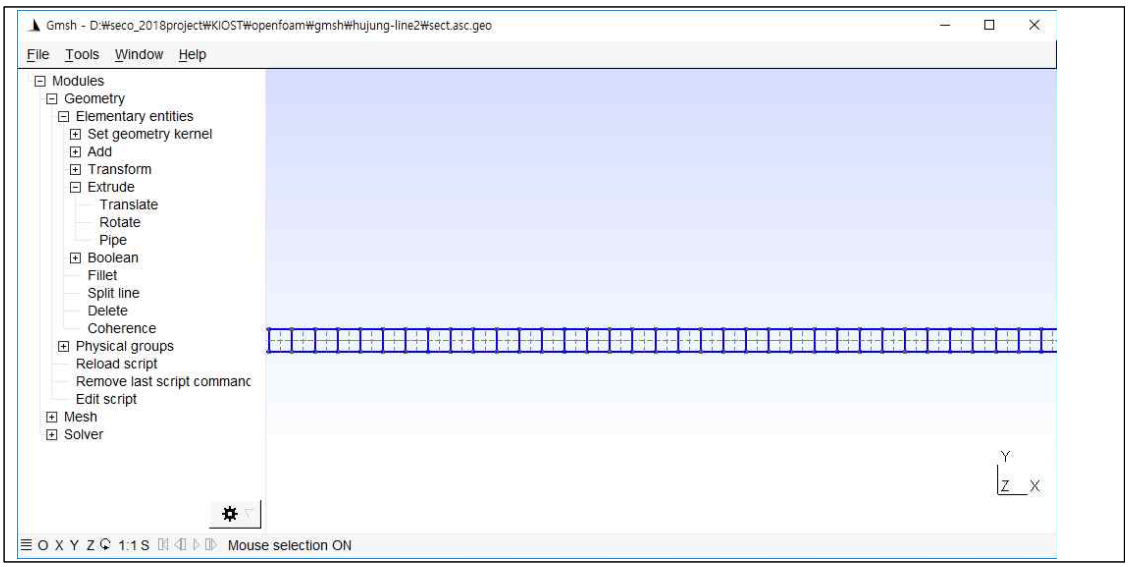


그림 23. Gmsh Extrude 과정

- Gmsh 내 좌측 정보창에서 Geometry-Edit script를 클릭하면 geo 파일 내의 Extrude 정보를 확인할 수 있다.

```
//+
Extrude {0, 1, 0} {
  Surface{1}; Layers{1}; Recombine;
}
```

그림 24. Extrude 정보

- Modules-Mesh-3D를 선택하여 격자 생성

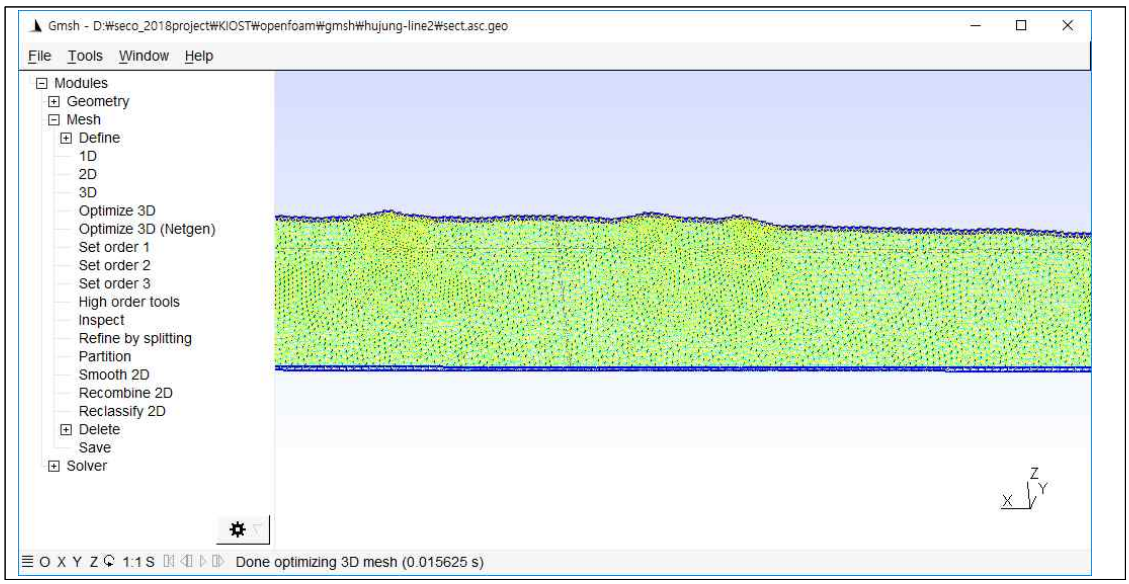
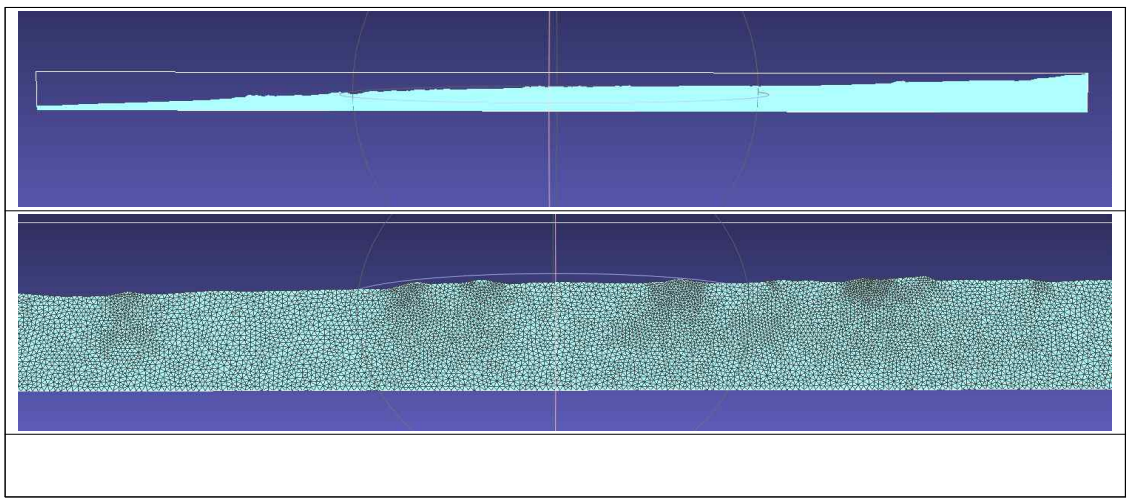


그림 25. 격자 생성 예

4) STL 파일 생성

- Gmsh 내 좌측 정보창에서 File-Export 선택한다.
- 지정된 위치에 확장자를 stl 로 선택하여 저장한다.
- 저장시 stl option은 Ascii 또는 Binary로 선택 가능하다.
- 생성된 STL 파일은 MeshLab 프로그램을 사용하여 확인 가능하다.



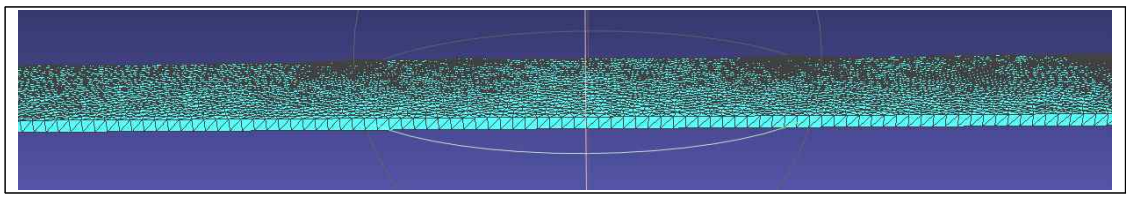


그림 26. MeshLab 표출 예

5) 잠제 STL 파일 생성

- 앞선 동일과정을 통해 잠제의 STL 파일을 구축할 수 있다.

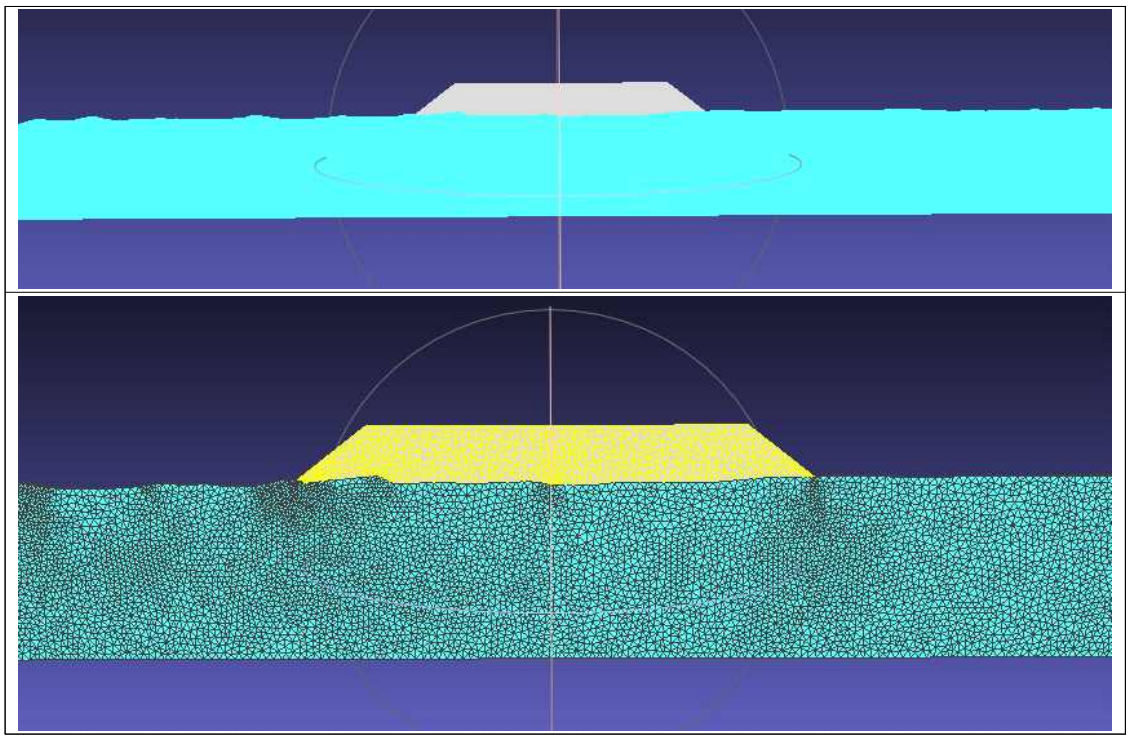


그림 27. 지형 및 잠제 STL 파일의 MeshLab 표출 예

다. OpenFoam 기본 폴더 구조 및 파일내용

- OpenFOAM 모델을 이용한 유동해석에는 기본적으로 '0', 'constant' 및 'system' 이름의 3개의 폴더가 존재한다.

(1) 0 폴더

- 0 폴더에서는 유동해석에 필요한 초기 및 경계조건을 설정한다(그림 28).
- 0 폴더를 기준으로 계산저장간격을 1초라고 설정하고 1분간 모의를 한다면 매 1초 마다 동일내용의 파일을 저장하는 폴더가 생성된다.
- 0 폴더내의 값은 0초에서부터 변화하므로 추후 모델을 새로 돌리는 경우를 대비하여 0.org 등의 폴더명으로 백업해두는 것이 좋다.

그림 28. 0 폴더 파일 예

① U 파일

- 0 폴더내의 특성치 파일 중 먼저 속도에 대한 값을 정의하는 U 파일로부터 파일의 기본 구조를 살펴보면 다음과 같다.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.7.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{    version     2.0;
     format      ascii;
     class       volVectorField;
     location    "0";
     object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
dimensions      [0 1 -1 0 0 0 0];
```

```
internalField   uniform (0 0 0);
boundaryField
{     inlet
      {
          type              waveVelocity;
          waveDictName      waveDict;
          value             uniform (0 0 0);
      }
      atmosphere
      {
          type              pressureInletOutletVelocity;
          value             uniform (0 0 0);
      }
      wall1
      {
          type              fixedValue;
          value             uniform (0 0 0);
      }
      defaultFaces
      {
          type              empty;
      }
      outlet
      {
          type              waveAbsorption2DVelocity;
          absorptionDir     666.0;
          value             uniform (0 0 0);
      }
}
// ************************************************************************* //
```

표 2. dimensions

순서	특성치	단위(SI unit)
1	질량(mass)	kg
2	길이(length)	m
3	시간(time)	s
4	온도(temperature)	K
5	몰수(quantity)	mol
6	전류량(current)	A
7	광량(luminous intensity)	cd

- dimensions는 물리량의 단위를 결정하는 것으로 표 2와 같으며 [0 1 -1 0 0 0 0]은 m의 1제곱 × s의 -1제곱으로 m/s를 표현한 것이다.
- internalField는 cell 값을 지정하는 것으로 0 폴더는 초기조건을 지정하므로 iniform(0 0 0)는 초기 내부의 모든 cell의 속도를 0으로 설정한 것이다.

- 모델 수행 후 속도가 변화되면 새로 생성되는 폴더에서의 internalField 값은 달라지게 된다.
- boundaryField는 blockMeshDict에서 설정된 각 경계면에 속도에 대한 경계조건을 지정하는 것이다.
- 일반적으로 OpenFOAM 모델에 많이 적용되는 boundary type은 <표 2>와 같다.

표 3. Boundary Condition

Boundary Type	Description
fixedValue	Constant value at boundary
zeroGradient	zero gradient(extrapolation)
fixedFluxPressure	the pressure gradient so that the boundary flux matches the velocity boundary condition
totalPressure	total pressure is constant. static pressure is computed
preesureInletVelocity	if known pressure at inlet. velocity is computed from flux
intelOutlet	switching between fixedValue and zeroGradient depending on flow direction
pressureInletOutletVelocity	combination of pressureVelocity and intelOutlet

- 본 연구에 적용된 속도에 대한 경계조건은 OpenFOAM에서 일반적으로 적용되는 경계조건(표 3)이 아닌 olaFLOW의 수행을 위해 수립된 조파경계조건 및 흡수경계조건이다.
- intel에 적용된 waveVelocity 경계조건은 constant 폴더내의 waveDict 파일에서 지정되는 wave theory와 파고, 파주기 등을 읽어 파형을 생성한다.
- outlet에 적용된 waveAbsorption2Dvelocty는 조파된 파형의 유속성분을 반대방향으로 상쇄시켜 반사파를 제거하는 소파조건이다. outlet 조건에서 absorptionDir는 에너지의 흡수방향을 나타내며 360보다 큰 값을 취하게 되면 해당경계면의 수직방향으로 에너지를 흡수한다.
- 3차원 계산에서는 outlet에 waveAbsorption3Dvelocity를 사용하며 옵션 absorptionDir 대신 nPaddles를 사용한다. nPaddles에 지정된 값에 따라 경계면이 분할되게 되고 분할된 면은 각각 다양한 파향에 대해 반사파를 제어하는 역할을 한다. 2차원 계산에서 파향은 일정하므로 nPaddles는 1을 취하며, 3차원의 경우 일반적

으로 25를 적용한다.

- flontANDback은 연직 2차원에서 계산하지 않는 경계면으로 empty로 설정한다.
- wall 경계면에서 유속은 fixedvalue를 사용하여 zero로 설정하였다. uniform(0 0 0)는 각각 x, y, z 방향의 유속을 0으로 설정함을 의미한다.

② p_rgh 파일

- p_rgh 파일은 압력을 설정하는 파일임.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.5-dev                               |
|   \\  /    A nd           | Web:      http://www.OpenFOAM.org               |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p_rgh;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
dimensions      [1 -1 -2 0 0 0 0];
internalField   uniform 0;
boundaryField
{    wall1
    {
        type            fixedFluxPressure;
        value           uniform 0;
    }
    defaultFaces
    {
        type            empty;
    }
    outlet
    {
        type            fixedFluxPressure;
        value           uniform 0;
    }
    inlet
    {
        type            fixedFluxPressure;
        value           uniform 0;
    }
```

```
    atmosphere
    {
        type            totalPressure;
        U               U;
        phi             phi;
        rho             rho;
        psi             none;
        gamma           1;
        p0              uniform 0;
        value           uniform 0;
}
// ************************************************************************* //
```

- inlet, outlet, wall에서 경계조건은 fixedFluxPressure 조건을 사용하여 경계면에 작용하는 유속으로부터 압력을 산정한다.
- 대기경계면 atmosphere에서는 totalPressure 조건을 적용하여 전압력 p0가 일정하게 0이 되도록 하여 정압력을 산정한다.

③ alpha.water 파일

- alpha.water 파일은 물의 VOF에 대한 경계값 및 초기값을 설정하는 것으로 전부 물인 경우 alpha=1, 공기인 경우 alpha=0 이다(추후 setFields를 통해 물과 공기를 구분).

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.5-dev                               |
|   \\  /    A nd           | Web:      http://www.OpenFOAM.org               |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{    version     2.0;
    format      ascii;
    class       volScalarField;
    object      alpha.water;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
dimensions      [0 0 0 0 0 0 0];
internalField   uniform 0;
boundaryField
{    inlet
    {
        type            waveAlpha;
        waveDictName    waveDict;
```

```
        value           uniform 0;
    }
    outlet
    {
        type            zeroGradient;
    }
    "wall."
    {
        type            zeroGradient;
    }
    defaultFaces
    {
        type            empty;
    }
    atmosphere
    {
        type            inletOutlet;
        inletValue      uniform 0;
        value           uniform 0;
    }
}
// ************************************************************************* //
```

- inlet에 적용된 waveAlpha 경계조건은 olaFLOW의 경계조건으로 waveDict파일의 정보로부터 파형에 따른 정보를 생성하는 기능을 한다.

④ porosityIndex 파일

- porous media를 설정한다.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      porosityIndex;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
dimensions      [0 0 0 0 0 0 0];
internalField   uniform 0;
```

```
boundaryField
{      inlet
      {
            type                zeroGradient;
      }
      outlet
      {
            type                zeroGradient;
      }
      "wall."
      {
            type                zeroGradient;
      }
      defaultFaces
      {
            type                empty;
      }
      atmosphere
      {
            type                zeroGradient;
      }
}
// ************************************************************************* //
```

- 본 예제에서는 경계조건은 zeroGradient로 설정한다.

⑤ turbulent viscosity 정의

- nut, nuTilda, nuSgs, k, epsilon의 파일로 경계면의 viscosity를 설정한다.
- Bounday Condition과 Wall function에 대한 설명은 Nextfoam(2017), “Boundary Conditions - OpenFOAM-4.1” 에 보다 자세히 제시되어 있다.

(2) Constant 폴더

- constant 폴더는 중력가속도, 난류모델, porosity, 파랑조건, 수송방정식 등의 물성치 옵션이 정의된 파일이 위치하며, 폴더내의 polyMesh 폴더에는 격자정보와 경계조건정보가, triSurface 폴더에는 stl 파일이 위치한다(그림 29).

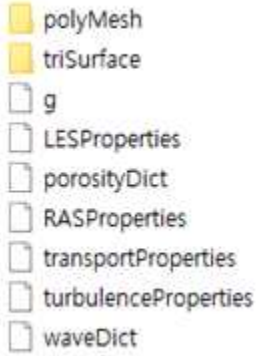


그림 29. Constant 폴더 파일 예

그림 30. polyMesh 폴더 파일 예

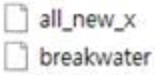


그림 31. triSurface 폴더 파일 예

① 중력가속도(g) 파일

- 연직 2차원을 가정한 본 예제에서 중력의 작용축은 Z축으로 설정하여 계산영역을 설정하였으므로 중력가속도는 다음과 같이 음의 값으로 설정한다.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{     version     2.0;
     format      ascii;
     class       uniformDimensionedVectorField;
     location    "constant";
     object      g;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
dimensions      [0 1 -2 0 0 0 0];
value           ( 0 0 -9.81 );
// ************************************************************************* //
```

② 난류(turbulenceProperties) 파일

- 난류모델의 sumulationType은 RAS, LES, laminar로 구분되며, LES 모델의 설정 예는 다음과 같다.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{     version     2.0;
     format      ascii;
     class       dictionary;
     location    "constant";
     object      turbulenceProperties;
}// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
simulationType  LES;
LES
{     LESModel          Smagorinsky;
     turbulence        on;
     printCoeffs       on;
     delta             smooth;
     cubeRootVolCoeffs
     {
```

```
        deltaCoeff      1;
    }
    PrandtlCoeffs
    {
        delta           cubeRootVol;
        cubeRootVolCoeffs
        {
            deltaCoeff      1;
        }
        smoothCoeffs
        {
            delta           cubeRootVol;
            cubeRootVolCoeffs
            {
                deltaCoeff      1;
            }
            maxDeltaRatio   1.1;
        }
        Cdelta          0.158;
    }
    vanDriestCoeffs
    {
        delta           cubeRootVol;
        cubeRootVolCoeffs
        {
            deltaCoeff      1;
        }
        smoothCoeffs
        {
            delta           cubeRootVol;
            cubeRootVolCoeffs
            {
                deltaCoeff      1;
            }
            maxDeltaRatio   1.1;
        }
        Aplus           26;
        Cdelta          0.158;
    }
    smoothCoeffs
    {
        delta           cubeRootVol;
        cubeRootVolCoeffs
        {
            deltaCoeff      1;
        }
        maxDeltaRatio   1.1;
    }
}//************************************************************************* //
```

③ waveDict 파일

- inlet을 통한 파의 조파조건을 설정한다.

```
/*--------------------------------------------------------------------------*\
| =========                 |                                                |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox          |
|  \\    /   O peration     | Version:  1.3                                  |
|   \\  /    A nd           | Web:      http://www.openfoam.org              |
|    \\/     M anipulation  |                                                |
\*--------------------------------------------------------------------------*/
FoamFile
{    version     2.0;
     format      ascii;
     class       dictionary;
     location    "constant";
     object      waveDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
waveType        regular;
waveTheory      cnoidal;
genAbs          1;
absDir          0.0;
nPaddles        1;
wavePeriod      10.0;
waveHeight      4;
waveDir         0.0;
wavePhase       1.57079633;
tSmooth         0.0;
// ************************************************************************* //
```

④ transportProperties 파일

- 물질의 동점성계수(nu), 밀도 등 특성치를 입력한다.
- 비압축성 물체의 경우 동점성계수만을 설정한다.
- 우선 transportModel에서 뉴턴유체 유·무를 지정한다.
- 뉴턴유체이면 Newtonian으로 설정하고 동점성계수를 상수로 입력한다.
- 비뉴턴유체의 경우는 transportModel을 CrossPowerLaw 또는 BirdCarreau로 설정하고 이에 따른 계수를 CrossPowerLawCoeffs나 BirdCarreauCoeffs에 설정한다.
- 압축성 유체의 경우 점성계수는 Sutherland 모델을 사용하여 온도의 함수로 줄 수 있고, 이는 thermodynamicProperties 파일에서 설정한다.

```
/*--------------------------------*- 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;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
phases (water air);
water
{    transportModel  Newtonian;
    nu              nu [ 0 2 -1 0 0 0 0 ] 1e-06;
    rho             rho [ 1 -3 0 0 0 0 0 ] 1000;
    CrossPowerLawCoeffs
    {
        nu0             nu0 [ 0 2 -1 0 0 0 0 ] 1e-06;
        nuInf           nuInf [ 0 2 -1 0 0 0 0 ] 1e-06;
        m               m [ 0 0 1 0 0 0 0 ] 1;
        n               n [ 0 0 0 0 0 0 0 ] 0;
    }
    BirdCarreauCoeffs
    {
        nu0             nu0 [ 0 2 -1 0 0 0 0 ] 0.0142515;
        nuInf           nuInf [ 0 2 -1 0 0 0 0 ] 1e-06;
        k               k [ 0 0 1 0 0 0 0 ] 99.6;
        n               n [ 0 0 0 0 0 0 0 ] 0.1003;
    }
}air
{    transportModel  Newtonian;
    nu              nu [ 0 2 -1 0 0 0 0 ] 1.48e-05;
    rho             rho [ 1 -3 0 0 0 0 0 ] 1;
    CrossPowerLawCoeffs
    {
        nu0             nu0 [ 0 2 -1 0 0 0 0 ] 1e-06;
        nuInf           nuInf [ 0 2 -1 0 0 0 0 ] 1e-06;
        m               m [ 0 0 1 0 0 0 0 ] 1;
        n               n [ 0 0 0 0 0 0 0 ] 0;
    }
    BirdCarreauCoeffs
    {
        nu0             nu0 [ 0 2 -1 0 0 0 0 ] 0.0142515;
        nuInf           nuInf [ 0 2 -1 0 0 0 0 ] 1e-06;
```

```
        k               k [ 0 0 1 0 0 0 0 ] 99.6;
        n               n [ 0 0 0 0 0 0 0 ] 0.1003;
    }
}sigma          sigma [ 1 0 -2 0 0 0 0 ] 0.07;
// ************************************************************************* //
```

⑤ porosityDict 파일

- 구조물의 porosity를 설정한다.
- 추후 설명할 setFieldsDict에 정의된 regions의 porosty를 설정한다.
- a, b, C는 friction factor이며, D50과 prosity는 입결과 공극률이다.
- 괄호이전의 숫자 2는 setFieldDict에서 설정한 porosity media 영역의 수로 본 예제의 경우 하나의 영역과 할당되지 않은 영역 하나로 총 2개의 영역을 가진다.
- 물체의 특성치는 setFieldsDict에 지정된 porosityIndex 순서에 따르며, 비할당영역인 porosityIndex가 0인 영역에서 공극률은 1이고 porosityIndex가 1인 영역에서 공극률은 0.49로 설정한다.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{     version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      porosityDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// Materials: clear region, core, secondary armour layer, primary armour layer
a                 2(0 50);
b                 2(0 1.2);
c                 2(0 0.34);
D50               2(1 0.01);
porosity          2(1 0.49);
// ************************************************************************* //
```

(3) System 폴더

- system 폴더에는 계산모의 시간, 계산시산 간격, 저장간격, 수치기법, 구조물 설

정 등의 옵션을 가지는 파일이 위치한다(그림 32).

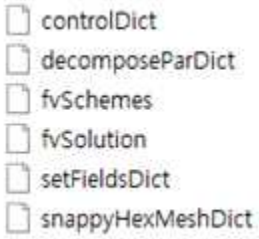


그림 32. system 폴더 파일

① setFieldsDict 파일

- 앞서 설명한 바와 같이 alpha.water 값은 공기인 경우 0, 물인 경우 1로 정의된다.
- setFieldsDict 파일에서 boxToCell은 box를 통해 물을 채우고자 하는 영역을 직육면체로 설정하는 부분이다.
- 설정영역 중 계산영역을 벗어나는 영역은 자동으로 배제된다. Box(-10, 0, -26) (930, 0.02, 0.293)은 (x-시작, y-시작, z-시작) (x-끝, y-끝, z-끝)을 설정하는 내용이다.
- 다공성 구조물의 영역은 surfaceToCell에서 설정된다.
- 대상이 되는 구조물의 형태는 trisurface 폴더내의 stl 파일로 저장되어 있어야 하며, 키워드 file 이후에 경로와 이름을 지정한다.
- 각 구조물은 porosityIndex에 따라 다른 값을 정의할 수 있으며, 여기서 정의되는 구조물들은 0 폴더에서 지저오디는 경계조건에는 포함되지 않는다.
- 이상의 과정에서 지정하지 않은 영역은 defaultFieldValues 옵션을 통해 정의하며 본 예제의 경우는 alpha.water, pososityIndex를 모두 0으로 설정한다.

```
/*--------------------------------------------------------------------------*\
| =========                 |                                                |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox          |
|  \\    /   O peration     | Version:  1.3                                  |
|   \\  /    A nd           | Web:      http://www.openfoam.org              |
|    \\/     M anipulation  |                                                |
\*--------------------------------------------------------------------------*/
FoamFile
{    version     2.0;
    format      ascii;
```

```
    class       dictionary;
    location    "system";
    object      setFieldsDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
defaultFieldValues
(    volScalarFieldValue alpha.water 0
    volScalarFieldValue porosityIndex 0
);
regions
(    boxToCell
    {
        box (-10.0 0.0 -26) (930.0 0.02 0.293);
        fieldValues
        (
            volScalarFieldValue alpha.water 1
        );
    }
    surfaceToCell
    {
        file                "./constant/triSurface/all_new_x_2.stl";
        outsidePoints       ((545.0 0.02 0.0));     // definition of outside
        includeCut          true;                   // cells cut by surface
        includeInside       true;                   // cells not on outside of surf
        includeOutside      false;                  // cells on outside of surf
        nearDistance        -1;                     // cells with centre near surf
                                                    // (set to -1 if not used)
        curvature           -100;                   // cells within nearDistance
                                                    // and near surf curvature
                                                    // (set to -100 if not used)
        fieldValues
        (
            volScalarFieldValue porosityIndex 1
        );
    }
    surfaceToCell
    {
        file                "./constant/triSurface/breakwater_3.stl";
        outsidePoints       ((545.0 0.02 0.0));     // definition of outside
        includeCut          true;                   // cells cut by surface
        includeInside       true;                   // cells not on outside of surf
        includeOutside      false;                  // cells on outside of surf
        nearDistance        -1;                     // cells with centre near surf
                                                    // (set to -1 if not used)
        curvature           -100;                   // cells within nearDistance
                                                    // and near surf curvature
                                                    // (set to -100 if not used)
        fieldValues
        (
```

```
            volScalarFieldValue porosityIndex 1
        );
    }
);
```

② controlDict 파일

- 해석 총 시간, 시간간격 등을 설정한다. deltaT는 해석 시산간격이며, 시간간격 이후의 제어변수는 자료의 출력에 대한 출력간격, 출력형식, 출력형식의 정확도, 출력물의 압축여부 등을 지정하는 것이다. 이후 시간에 대한 형식과 정확도를 지정한다.
- runTimeModifiable은 해석이 진행되는 도중 controlDict 파일을 수정하여 해석을 중단하거나 해석이 종료되는 시간을 연장할 수 있도록 해석 진행 도중에 system 폴더의 변경을 감시하는 기능을 온/오프하는 기능이다.
- maxCo는 최대 Courant 수, maxAlphaCo는 VOF 해석시의 최대 Courant 수, maxDeltaT는 최대시간 간격을 나타낸다.
- controlDict에 사용된 키워드에 어떤 값을 사용할 수 있을지 궁금한 경우 값을 지정하는 자리에 dummy를 넣어주면 넣어줄 수 있는 추가적인 옵션이 출력된다.

```
/*--------------------------------------------------------------------------*\
| =========                 |                                                |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox          |
|  \\    /   O peration     | Version:  1.3                                  |
|   \\  /    A nd           | Web:      http://www.openfoam.org              |
|    \\/     M anipulation  |                                                |
\*--------------------------------------------------------------------------*/
FoamFile
{    version     2.0;
    format      ascii;
    location    "system";
    class       dictionary;
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
application     olaFlow;
startFrom       latestTime;
startTime       0;
stopAt          endTime;
endTime         80;
deltaT          0.001;
writeControl    adjustableRunTime;
```

```
writeInterval   0.1;
purgeWrite      0;
writeFormat     ascii;
writePrecision  6;
writeCompression uncompressed;
timeFormat      general;
timePrecision   6;
runTimeModifiable yes;
adjustTimeStep  yes;
maxCo           0.45;
maxAlphaCo      0.45;
maxDeltaT       0.05;
// ************************************************************************* //
```

③ fvSchemes 파일

- fvSchemes 파일은 solver가 계산하고자하는 방정식의 차분 scheme(공간적인 차분, 내삽기법, 그리고 flux에 관계된 내용)을 지정한다.
- 본 예제에서 ddtSchemes는 시간차분에 Euler 방법을 적용함을 의미한다.
- 이후 gradSchemes는 구배에 Gauss 선형보간을 지정한 것이다.
- 파일 내에서 각 방정식의 항별로 서로 다른 scheme을 지정할 수 있으며, dummy도 적용할 수 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;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
ddtSchemes
{    default          Euler;
}gradSchemes
{    default          Gauss linear;
}divSchemes
{    div(rhoPhi,U)  Gauss limitedLinearV 1;
```

```
    div(U)  Gauss linear;
    div((rhoPhi|interpolate(porosity)),U)  Gauss limitedLinearV 1;
    div(rhoPhiPor,UPor)  Gauss limitedLinearV 1;
    div(rhoPhi,UPor)  Gauss limitedLinearV 1;
    div(rhoPhiPor,U)  Gauss limitedLinearV 1;
    div(phi,alpha)  Gauss vanLeer;
    div(phirb,alpha) Gauss interfaceCompression;
    div((muEff*dev(T(grad(U))))) Gauss linear;
    div(phi,k)      Gauss upwind;
    div(phi,epsilon) Gauss upwind;
    div((phi|interpolate(porosity)),k)      Gauss upwind;
    div((phi|interpolate(porosity)),epsilon) Gauss upwind;
    div(phi,omega) Gauss upwind;
    div((phi|interpolate(porosity)),omega) Gauss upwind;
}laplacianSchemes
{   default         Gauss linear corrected;
}interpolationSchemes
{   default         linear;
}snGradSchemes
{   default         corrected;
}fluxRequired
{   default         no;
    p_rgh;
    pcorr;
    alpha.water;
}
// ************************************************************************* //
```

④ fvSolution 파일

- fvSolution 파일은 해석할 방정식의 matrix solver를 선택하고 algorithm의 파라미터를 설정하는 파일이다.

```
/*--------------------------------*- 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      fvSolution;
}
```

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
solvers
{       "alpha.water.*"
        {
            nAlphaCorr          1;
            nAlphaSubCycles  2;
            alphaOuterCorrectors  yes;
            cAlpha              1;
            MULESCorr           no;
            nLimiterIter       3;
            solver              smoothSolver;
            smoother            symGaussSeidel;
            tolerance           1e-8;
            relTol              0;
        }
        "pcorr.*"
        {
            solver              PCG;
            preconditioner    DIC;
            tolerance           1e-5;
            relTol              0;
        }
        p_rgh
        {
            solver              PCG;
            preconditioner    DIC;
            tolerance           1e-07;
            relTol              0.05;
        }
        p_rghFinal
        {
            $p_rgh;
            relTol              0;
        }
        U
        {
            solver              smoothSolver;
            smoother            symGaussSeidel;
            tolerance           1e-06;
            relTol              0;
        }
        "(k|epsilon|omega|B|nuTilda).*"
        {
            solver              smoothSolver;
            smoother            symGaussSeidel;
            tolerance           1e-08;
            relTol              0;
        }
}PIMPLE
```

```
{      momentumPredictor    no;
       nOuterCorrectors     1;
       nCorrectors          3;
       nNonOrthogonalCorrectors 0;
}relaxationFactors
{      fields
       {
       }
       equations
       {
            ".*" 1;
       }
}
// ************************************************************************* //
```

⑤ decomposeParDict 파일

- decomposeParDict 파일은 병렬수행을 지정하는 파일이다.
- numberOfSubdomains는 적용할 CPU 또는 Core 수를 지정한다.
- method는 영역분할 방법을 지정하며, simple, metis, Scotch 등의 영역분할 방법을 적용할 수 있다.
- simpleCoeffs는 method simple을 사용할 때 설정해야하는 계수들이다.

```
/*--------------------------------------------------------------------------*\
| =========                 |                                                |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox          |
|  \\    /   O peration     | Version:  1.3                                  |
|   \\  /    A nd           | Web:      http://www.openfoam.org              |
|    \\/     M anipulation  |                                                |
\*--------------------------------------------------------------------------*/
FoamFile
{      version             2.0;
       format              ascii;
       location            "system";
       class               dictionary;
       object              decomposeParDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
numberOfSubdomains 4;
method              simple;
simpleCoeffs
{      n                   ( 2 2 1 );
       delta               0.001;
}hierarchicalCoeffs
{      n                   ( 1 1 1 );
```

```
    delta           0.001;
    order           xyz;
}metisCoeffs
{   processorWeights ( 1 1 1 1 );
}manualCoeffs
{   dataFile        "";
}distributed    no;
roots       ();
// ************************************************************************* //
```

라. OlaFlow

(1) 배경

- 오픈소스 기반인 OpenFOAM은 높은 범용성을 목적으로 개발되어 왔지만, 해안 및 항만공학에 적용하기 위해 필요한 조파기능 및 반사파제어기능이 불충분하여 파동역학과 관련된 공학적 문제의 적용에는 한계가 있다.
- 이러한 제한을 극복하고자 Higuera et al.(2013)은 3차원 VARANS(Volume-Averaged Reynolds-Averaged Navier-Stokes) 방정식으로 기초로 OpenFOAM에 조파기능과 반사파제어 기능을 추가적으로 부여한 IHFOAM을 개발하고 이후 IHFOAM에 공극률을 갖는 투과성구조물에서 유체저항을 고려하는 모듈을 추가하고 multi-paddle piston 방식의 조파기능과 조파와 반사파제어를 위한 감쇠영역에서 cutting-egde 기술을 이용하여 CFD 기반의 수치계산에서 단점으로 지적되는 계산비용을 절감한 OlaFOAM을 개발하였으며(이광호 등, 2017), 현재는 OlaFlow로 그 이름이 변경되어 개발이 지속되고 있다.
- 따라서 이 무료 오픈 소스 프로젝트는 파랑의 동적 모의(wave dynamic simulation)의 최신 기술을 OpenFOAM 및 FOAM-extend 커뮤니티에 제공하고자 노력하고 있다.
- OlaFlow는 two-phase solver 로서 이 solver는 경계면에서 작동하는 최신(state-of-the-art) active wave generation 및 감쇠(absorption) 기술, 다공성 매체(porous media)를 통과하는 자유표면흐름(free surface flow) 또는 multi-paddle piston-type wave generation의 재현(replication)과 active wave absorption을 포함하는 최첨단 기법(cutting-edge technologies)을 제공한다.

(2) 개요

- olaFlow는 OpenFOAM에 포함된 solver들 중 하나이다.
- 이 solver는 유한체적차분(finite volume discretization)과 VOF(volume of fluid) 방법을 사용하여 두 비압축성 물질(two incompressible phases)에 대한 3차원 Volume Averaged Reynolds Averaged Navier-Stokes(VARANS) 방정식을 계산한다.
- VOF 에서 각 phase는 격자(cell)내 i번째 물질(i^{th} material)의 유체 부피가 차지하는 비(α_i)로 표현된다.
- 이 방법의 주요 장점은 매우 단순해서 격자 움직임(mesh motion)없이, 아주 복잡한 자유표면 형태를 쉽게 표현할 수 있다는 점이다.
- VOF의 단점은 표면장력 효과가 증가할 때 효율성이 다소 감소한다는 것이다.
- 그렇지만, 대다수 연안 엔지니어링 실무 분야는 비교적 긴 파장의 파를 다루므로 아주 특별한 현상이 아니한 표면장력을 무시할 수 있다.
- olaFlow는 또한 여러 난류모델(e.g. $k-\epsilon$, $k-\omega$ SST, LES)를 지원한다.
- olaFlow는 정적 격자(static mesh)에만 적용이 가능하며 olaFlow의 강화 버전인 olaDyMFlow로 동적 격자(dynamic mesh)를 적용할 수 있다 ("DyM"은 Dynamic Mesh를 지칭).
- 따라서, 이 solver는 부유체의 움직임, 자유표면을 따른 dynamic mesh refinement 또는 이동 경계(moving boundary)에서 파의 생성을 모의할 수 있다.
- 이 두 solver는 이상의 것 이외에는 동일한 방법에 따라 동일한 방정식을 계산한다.
- olaFlow와 olaDyMFlow의 경계조건 모듈은 이 둘을 서로 구분 짓지 않고 작동하도록 구현되어있다.
- 이 두 solver에는 여러 파 이론(wave theories)에 따라 파랑을 생성하는 기능이 추가되어 있다.
- 추가적으로 active wave absorption이 외삽 2차 이론(extrapolated 2D theory)으로 프로그램되어 있어 순흡수경계(pure absorbent boundaries) 또는 파 생성경계(wave generation boundaries)에서 outgoing wave가 적은 파반사(minor reflection)를 가지고 빠져나갈 수 있도록 만들어 준다. 추가적으로, 비 직교 입사파(non orthogonal incident wave)를 다루는 다른 3차원 이론들이 개발되어 있다.

(3) 기본 방정식

- 앞서 언급된 VARANS 방정식은, 연속 방정식과 운동량보존 방정식을 포함하는, 압력(pressure) 및 속도(velocity)를 연결하는 지배 수학 표현식으로 서로 다른 다공성 매체를 통과하는 흐름을 모의할 수 있게 해주며 대부분의 연안 엔지니어링 실무 문제에 적용이 가능한 비압축성유체 가정이 적용된다.

$$\frac{\partial < u_i >}{\partial x_i} = 0$$

$$\frac{\partial \rho < u_i >}{\partial t} + \frac{\partial}{\partial x_i}\left[\frac{1}{\phi}\rho < u_i >< u_j >\right] =$$

$$-\phi\frac{\partial < p^* >^f}{\partial x_i} + \phi g_j X_j \frac{\partial \rho}{\partial x_i} + \frac{\partial}{\partial x_j}\left[\mu_{eff}\frac{\partial < u_i >}{\partial x_j}\right] - [CT] \qquad)$$

- 여기서, closure term $[CT]$는 다음과 같이 정의된다.

$$[CT] = A < u_i > + B|< u >|< u_i > + C\frac{\partial < u_i >}{\partial t}$$

- 이 식에서 마찰계수는 Burcharth and Andersen(1995)에 적용된 Engelund(1953) 식에 의해 계산된다.

$$A = \alpha\frac{(1-\phi)^\beta}{\phi^2}\frac{\mu}{D_{50}^2} \qquad (1.4a)$$

$$B = \beta\left(1 + \frac{7.5}{KC}\right)\frac{1-\phi}{\phi^2}\frac{\rho}{D_{50}} \qquad (1.4b)$$

ρ	Density, which is calculated as presented in equation 1.5
$U,\ u_i$	Velocity vector
p^*	Pseudo-dynamic pressure
$g,\ g_i$	Acceleration due to gravity
X	Position vector
ϕ	Porosity
$\sigma\kappa\nabla\alpha$	Surface tension term
σ	Surface tension coefficient
κ	Curvature of the interface: $\kappa = \nabla \cdot \frac{\nabla\alpha}{\lvert\nabla\alpha\rvert}$
α	Indicator (VOF) function
μ_{eff}	Efficient dynamic viscosity, which takes into account the molecular dynamic viscosity plus the turbulent effects: $\mu_{eff} = \mu + \rho\nu_{turb}$
ν_{turb}	Turbulent kinetic viscosity, given by the chosen turbulence model

- 식(1.2)의 항들은 특정한 성질들을 가지고 있음: 등호의 좌측 항들은 계수 행렬(coefficient matrix)을 조합하기 위해 OpenFOAM®에서 사용되고 우측 항들은 explicit하게 계산되어 방정식의 독립항을 형성한다.
- phase의 이동을 설명하기 위한 추가 방정식도 고려해야만 하며 대다수의 해안공학 분야에서는 물과 공기만이 존재하므로, 다음의 분석은 이 두 phase만을 대상으로 수행한다.
- 이 가정으로 인해, 단지 하나의 indicator phase function(α)만이 필요하게 되며 이 function은 각 cell의 단위 부피당 물의 양으로 정의된다.
- 이는 만약 $\alpha = 1$이면 물이 가득 찬 것이고, $\alpha = 0$이면 공기로 가득 찬 것이며, 다른 경우는 경계면을 가지고 있는 것을 의미한다.
- VOF 함수에 가중치를 적용하여 각 cell 내 유체의 속성을 간단하게 계산할 수 있음. 예를 들어 cell의 밀도는 다음과 같이 계산된다.

$$\rho = \alpha\rho_{water} + (1-\alpha)\rho_{air}$$

- 유체 운동을 추적하는 방정식의 시작점은 고전(classic) 이류방정식이다.

$$\frac{\partial \alpha}{\partial t} + \frac{1}{\phi} \frac{\partial < u_i > \alpha}{\partial x_i} = 0$$

- 물리적 결과를 얻기 위해서는 다음과 같이 몇 가지 제한사항을 적용해야 함: sharp interface가 유지되어야만 하고, α는 conservative해야 하며 0과 1사이에 위치해야만 한다. OpenFOAM®은 compressing differencing scheme의 적용 대신에 artificial compressing term ($\nabla \cdot U_c \alpha(1-\alpha)$)을 적용한다.
- 이 접근법은 conservative하며 단지 interface에서만 non-zero 값을 취하게 된다.
- 또한, U_c가 interface($\frac{\nabla \alpha}{|\nabla \alpha|}$)에 수직이면 유동은 압축되지 않고, 이 점에서 α의 값이 증가하게 되어 공기에서 물의 phase로 변화되게 된다.
- 이로부터 다음의 최종식이 도출된다.

$$\frac{\partial \alpha}{\partial t} + \frac{1}{\phi} \frac{\partial < u_i > \alpha}{\partial x_i} + \frac{1}{\phi} \frac{\partial < u_{ci} > \alpha(1-\alpha)}{\partial x_i} = 0$$

- 여기서, $|U_c| = \min[c_\alpha |U|, \max(|U|)]$이며, 사용자는 factor c_α를 지정할 수 있다.
- 기본 값으로 1 을 사용하지만, interface의 압축성을 강화하기 위해 더 큰 값을 적용할 수도 있다.
- 이 방정식의 경계는 특수 설계된 MULES(Multidimensional Universal Limiter for Explicit Solution)라고 칭하는 solver를 이용하여 계산된다.
- 이 solver는 차분된 발산항(discretized divergence term)의 flux에 대해 제한 팩터(limiter factor)를 사용하여 최종 값이 0과 1사이의 값으로 존재하도록 추구한다.
- 원칙적으로, 해석 알고리즘은 PISO(Pressure Implicit with Splitting of Operators) 이다.

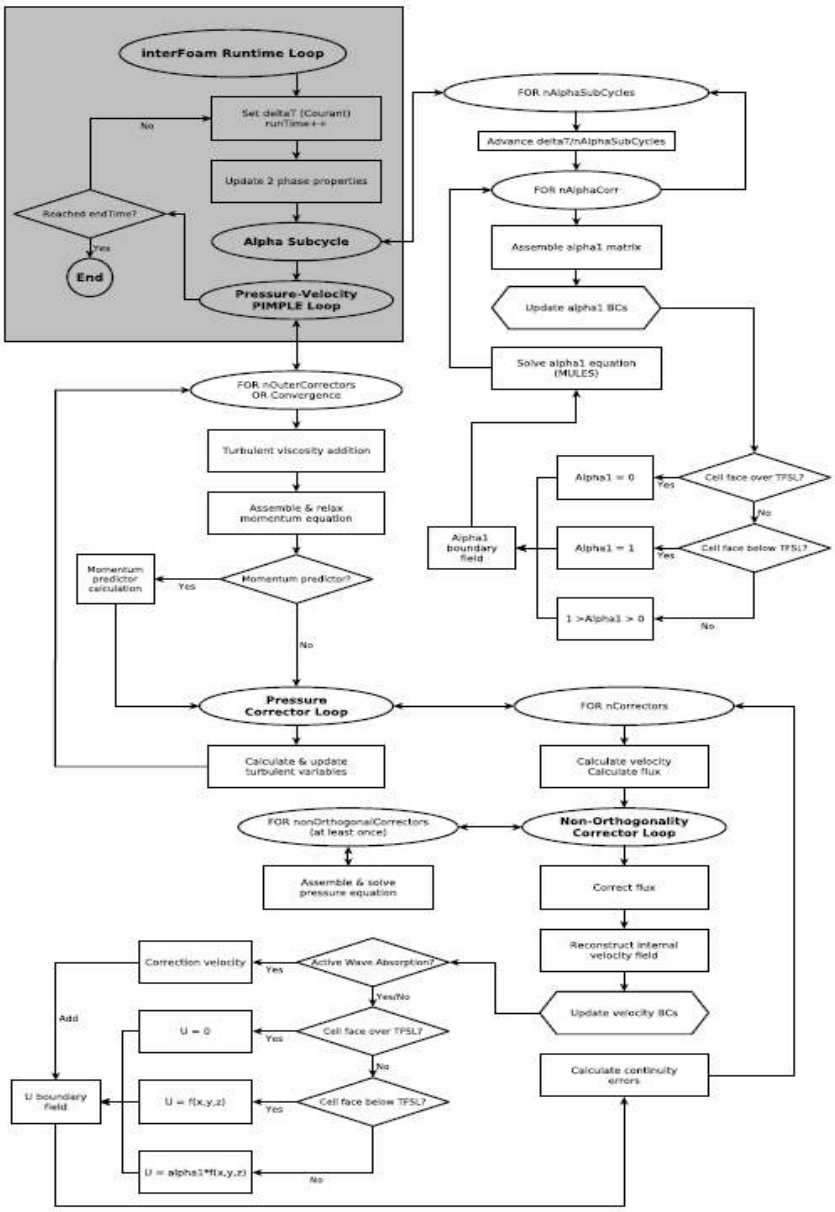


그림 33. “interFoam” solving flow chart. “TFSL” stands for Theoretical Free Surface Level.

- 그림 33은 각 time step에서 계산하는 전체 루프를 보여준다.
- 그림에서 메인 루프는 짙은 배경으로 제시되어 있으며, alpha subcycle과 PIMPLE 루프는 그 외부에 추가로 설명되어 있다.
- 둘 다 새로 개발된 경계조건에 대한 기본 설명을 포함하며 그림에 제시된 일부 변수(e.g. “nAlphaSubCycles”, “nCorrectors” ...)들은 program control file에서 설

정할 수 있고, 이들은 모델 해석 과정(model solving procedure)에서 수행능력(performance)을 지배한다.

(4) OlaFlow 설치 및 컴파일

- 사용자 환경에 따라 olaFlow와 olaDyMFlow를 설치하는 방법은 먼저 사용자 환경에 다음과 같이 OlaFlow를 복사한다.

```
〉_ git clone git://github.com/phicau/olaFlow.git
```

- 코드 다운로드 이후 업데이트 방법은 다음과 같다.

```
〉_ git checkout
〉_ git pull
```

- 먼저, allMake 스크립트를 실행하여 경계조건을 컴파일 한다.

```
〉_ cd genAbs
〉_ ./allMake
```

- 그리고 사용자는 사용 버전에 맞는 solver를 컴파일 한다.

```
〉_ cd solvers/olaFlowXXXXX
〉_ ./allMake
```

- 여기서, XXXXX는 사용자의 OpenFOAM버전 정보이다.
- 경계조건은 controlDict 파일 내부에서 동적으로 연결되며, 이를 통해 다른 solver와 함께 사용할 수 있다(e.q. interFoam).
- 입력 방법은 다음과 같다.

```
libs
(
    "libwaveGeneration.so"
    "libwaveAbsorption.so"
);
```

마. OpenFoam (OlaFlow) 파 생성 (wave generation)

(1) 배경

- OlaFlow 파 생성 BC는 active wave absorption과 laboratory wavemaker를 재현하기 위한 특수 묘듈과 같은 여러 기능이 새롭게 도입되었고, 다음과 같은 여러 파 이론에 따라 경계에서 사실적으로 파가 생성되도록 코딩되었다. Stokes I, II, cnoidal and streamfunction, regular waves; Boussinesq solitary wave; irregular(random) waves, first and second order; piston-type wavemaker velocity profile replication.
- OlaFlow 파 생성 이론 Le Méhauté(1976)의 그래프 (그림 34)를 통해 선택 가능하다.

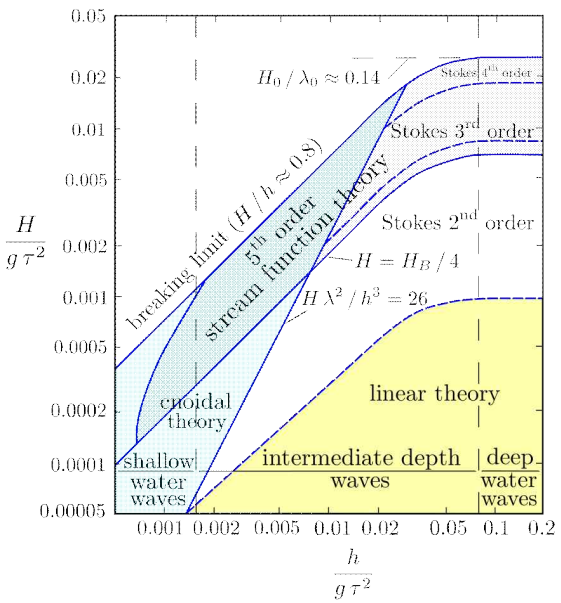


그림 34. Wave theories range of applicability. Le Méhauté(1976)

- 이 그래프는 OlaFlow의 reference 폴더 내 waveTheory 폴더에 위치한 waveTheory.py 라는 전처리 프로그램을 사용하여 쉽게 적용할 수 있다.
- 스크립트를 실행한 후 파고(H), 수심(h), 주기(T)를 입력하면 Le Méhauté(1976)의 그래프 상에 waveTheory에 맞는 점이 표시된다.
- 파 생성 BC가 적절하게 동작하기 위해서는 중력은 Z 축에서 음의 방향으로 작용해야 하며, 각 wave generation boundary의 최저점(lowest point)은 반드시 같은 level에 위치해야 한다.
- 그렇지만, patch마다는 level을 다르게 줄 수는 있으며 파향은 X 축을 기준으로 측정되며, 반시계방향으로 증가한다.
- 표 4는 이후 설명할 파 이론을 요약한 것으로 경계조건에 구현된 solver에 대한 주요 정보와 사용된 reference가 포함되며 이 장에 사용된 일반 변수는 모두 표 5에 제시된다.

표 4. Wave generation references

Theory	Reference	Comments
Stokes I and II	Dean and Dalrymple(1991)	
Stokes V	Skjelbrea and Hendrickson (1960)	
Cnoidal	Svendsen(2006)	Best fil solver.
Streamfunction	Fenton(1988)	No solver programmed. Its input is the output coefficients from Fenton(1988) program.
Solitary wave	Lee et al.(1982)	Boussinesq theory

표 5. 변수 리스트

η	free surface elevation
$[u,v,w]$	velocity components
L	wave length
T	wave period
h	water depth
k	wave number
c	wave celerity
ω	angular frequency
ψ	wave phase shift
θ	wave phase
β	wave propagation direction

(2) OlaFlow에서 생성되는 파 이론 (Wave theories)

① Stokes I

- Stokes I, Airy wave theory, small amplitude waves 또는 linear waves는 water wave에 대한 가장 단순한 이론해이다(analytical solution).
- 이 이론은 Airy(1845)에 의해 개발되었으며, 구현이 쉽고 일부 엔지니어링 근사에서는 충분한 정확성을 가지기 때문에 현재까지도 널리 사용되는 이론이다.
- 이론적인 적용 범위가 좁기는 하지만, 이 이론은 higher order로 확장할 수 있고 추가적 개발이 가능하여 광범위하게 사용된다.
- 이 파 이론의 주요 가정은 다음과 같다.

· 연속, 균질, 비압축성 및 비점성(inviscid) 유체이다.

· 코리올리힘은 무시한다.

· 표면장력 무시한다.

· free surface 상의 균일(uniform)하고 상수(constant)이다.

· 흐름은 비회전성(irrotational)이다.

· 저면은 고정되어있고 비투과정(impervious)이다.

· 상대 파고(relative wave height)는 작다 ($\frac{H}{h} \ll 1$).

- 파는 여러 크기(magnitude) 파와 그들 간의 상호관계로 표현할 수 있다. 가장 중요한 것은 분산 관계(dispersion relation), 어떤 깊이에서 주어진 period에 대한 파장(L)을 구하기 위한 초월방정식(transcendental equation)이다.

$$L = \frac{gT^2}{2\pi}\tanh\left(\frac{2\pi h}{L}\right)$$

$$k = \frac{2\pi}{L}, \quad \omega = \frac{2\pi}{T}, \quad c = \frac{L}{T}, \quad L_0 = \frac{gT^2}{2\pi}$$

- 이 파 이론의 해는 자유표면고도(free suface elevation)과 속도장(velocity field)을 얻을 수 있는 potential function을 기반으로 한다.
- 2차원 파랑에서 X 축으로 이동하는 것을 양의 방향이라고 할 때 표현식은 다음과 같다.

$$\eta = \frac{H}{2}\cos(kx - \omega t + \psi)$$

$$u = \frac{H}{2}\omega\frac{\cosh(kz)}{\sinh(kh)}\cos(kx - \omega t + \psi)$$

$$w = \frac{H}{2}\omega\frac{\sinh(kz)}{\sinh(kh)}\sin(kx - \omega t + \psi)$$

  - 이들 표현식을 적용하기 위해서는 경계의 최저 좌표가 $z=0$에 위치해야만 한다는 가정이 있어야 하나 이는 일반적인 것이 아니며, 어떤 시뮬레이션의 경우 wave generation boundary가 다른 lowest level을 가질 수도 있을 것다.
  - 이를 고려하기 위해서는 이 z 좌표계를 로컬 좌표계, $z = h^* + z^*$로 간주해야 하며, 여기서 z^*에 대한 기준 level은 초기정수면(initial still water level), 그리고 h^*는 현재 경계의 local water depth 이다.
  - 3D 관점에서 2D는 특별한 case임을 고려하면 이 식은 일반적인 case로 확장할 필요성이 있다.
  - 이는 원하는 방향(β)으로 2차원 case를 설정하고, X 와 Y 축의 결과를 투영

(projecting)하며, 중력은 언제나 Z 축으로만 작용한다는 가정으로 간단하게 해결할 수 있다. 이 과정을 통해 다음식이 얻어진다.

$$\eta = \frac{H}{2}\cos(\theta)$$

$$u = \frac{H}{2}\omega\frac{\cosh(kz)}{\sinh(kh)}\cos(\theta)\cos(\beta)$$

$$v = \frac{H}{2}\frac{\cosh(kz)}{\sinh(kh)}\cos(\theta)\sin(\beta)$$

$$w = \frac{H}{2}\omega\frac{\sinh(kz)}{\sinh(kh)}\sin(\theta)$$

- 여기서, $\theta = k_x x + k_y y - \omega t + \psi$ 이며 $k_x = k\cos(\beta)$는 X 축으로 투영된 파수(wave number)이고, $k_y = k\sin(\beta)$는 다른 수평 방향으로 투영된 파수이다.

② Stokes II

- Stokes II는 Stokes I 이론을 보다 발전시킨 것으로, 앞서 언급된 이론에 second order term을 추가한 것이다.
- 단지 분산 관계(dispersion relation)를 반복적(iteratively)으로 계산하기 때문에 이 이론은 여전히 구현이 용이하다.

$$\eta = \frac{H}{2}\cos(\theta) + k\frac{H^2}{4}\frac{3-\sigma^2}{4\sigma^3}\cos(2\theta)$$

$$u = \frac{H}{2}\omega\frac{\cosh(kz)}{\sinh(kh)}\cos(\theta) + \frac{3}{4}\frac{H^2\omega k\cosh(2kz)}{4\sinh^4(kh)}\cos(2\theta)$$

$$w = \frac{H}{2}\omega\frac{\sinh(kz)}{\sinh(kh)}\sin(\theta) + \frac{3}{4}\frac{H^2\omega k\sinh(2kz)}{4\sinh^4(kh)}\sin(2\theta)$$

$$\sigma = \tanh(kh)$$

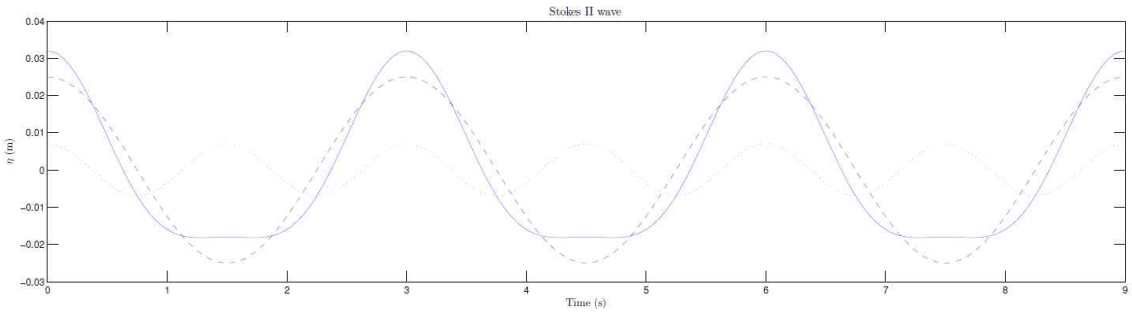


그림 35. Stokes II 파 이론: H = 5 cm; h = 40 cm; T = 3 s. 점선 : Stokes I, 실선: Stokes II.

③ Stokes V

- 위의 이론들은 smaller wave를 잘 묘사하나 파고가 증가하면 이들은 더 이상 small amplitude로 고려할 수 없으며, 따라서 이들 finite height wave를 표현할 수 있는 다른 파 이론이 필요하다.
- fifth order Stoke wave theory는 파 생성에 관한 이전 경험을 바탕으로 Skjelbreia and Hendrickson(1960)에 의해 제시된 이론을 사용한다.
- 두 개의 미지수를 갖는 두 초월 방정식(transcendental equation)의 시스템에서 표현식 전반에 위치하는 λ 파라미터와 파장(L)을 얻기 위해서는 반복적인 계산법(solved iteratively)을 수행한다.

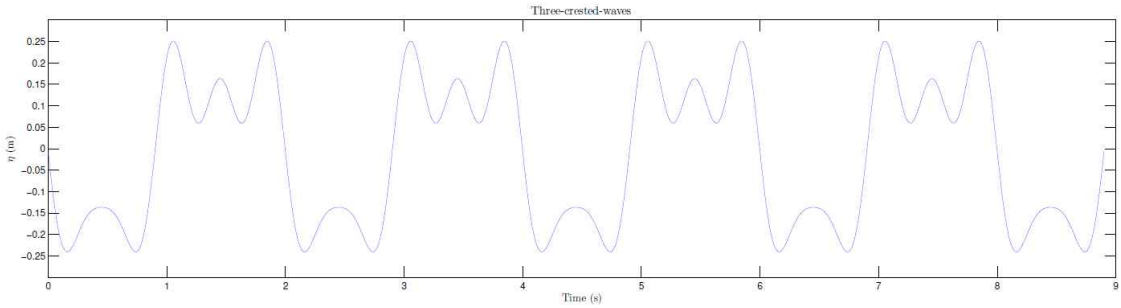


그림 36. Stoke V 파형 (예).

④ Cnoidal

- Cnoidal wave는 특정 형태(distinctive shape)를 가지는 비선형 규칙파의 일종.
- 이 파는 파장이 수심보다 클 때, 자연상태에 존재하는 실제 파랑이다.
- 매우 길고 편평한 파곡(troughs)과 뾰족한 파봉(steep wave crest)을 가지는 매우 특징적인 파형을 가지는 파이다.

⑤ Streamfunction 이론

- Stokes V 또는 cnoidal wave 이론이 적합하지 않을 때는 일반적으로 high order streamfunction solution을 이용할 수 있다.
- 이이 solution은 필요한 계수를 얻기 위한 방정식 시스템을 컴퓨터가 계산하기 때문에 어떤 order로도 확장할 수 있다.
- 이 접근법은 breaking condition에 매우 가까운 파의 생성을 허용한다.
- 이 접근법은 Fourier series의 complex potential solution을 표현하는데 기초한다.
- Solver는 개방경계에 대해 코딩되어 있지 않으며, 대신 “Fourier”라 불리는 프로그램을 사용 하며 이 프로그램은 C 언어로 코딩되어 있고 현재 다운로드할 수 있다 (old version을 다운로드 하는 것이 편리).
- 입력 자료는 무차원 파고(H/h), 주기($T\sqrt{g/h}$), 파장이며 흐름의 크기(current magnitude)를 추가할 수 있다.
- 흐름의 크기는 wave flume 또는 tank를 모델링하는 경우 0을 취할 수 있다.
- 출력은 여러 파 파라미터(wave parameter)와 주어진 수의 component 별 두 개의 세트로 구성된다.
- component는 일반적인 파에서는 10이면 충분하지만, steep한 파에서는 32까지 크게 주어야 한다.
- 이들은 파의 frame reference에서(또한 프로그램에 의해 제공되는) 파장, 평균 유체 속도와 함께 자유표면고도와 속도장을 산출하며, 이들 값은 경계조건을 위한 입력값의 구성요소가 된다.

⑥ Solitary wave

- Korteweg-de Vries 식으로부터 permanent form의 여러 해를 계산할 수 있음: 파가 전파되는 동안 그 형태가 변화하지 않는 것을 의미한다.
- 이들 해의 첫 번째는 solitary wave이며, 이는 oscillatory wave가 아닌

translational wave 이다.

- 파의 형태가 명확한 파곡(wave troughs)없이 언제나 정수면상에 위치하기 때문에, 이는 파의 모든 입자가 전파 방향으로 이동함을 의미한다.
- 다양한 Solitary wave 이론 중 Lee et al.(1982)의 이론을 이용하여 속도와 자유표면에 관한 표현식을 적용하였다. Boussinesq 이론을 선택한다.

⑦ Irregular wave

- 자연 상태의 파는 규칙적이지 않으며 따라서 실제상황을 시뮬레이션 하기 위해서는 불규칙파를 생성하는 것이 필요하다.
- 대부분의 경우는 first order theory가 해상상태(sea state)를 표현하는데 충분할 것이지만, 보다 정확한 결과를 얻기 위해서는 first order component간의 second order interaction을 고려하는 것이 필요하다.

㉮ First order

- First order directional 불규칙파는 주어진 수의 component(N)를 가지고 Stoke I 파를 선형 중첩하여 생성할 수 있다.
- 물리적으로 정확한 방법이며, 대부분의 경우 많은 수의 component로 실제 파랑 스펙트럼을 discretizing하는 것으로, 아주 작은 amplitude들이 구해진다.
- 각각의 component는 자체의 파고(H_i), 파주기(T_i), 파위상(ψ_i) 그리고 전파방향(β_i)에 의해 정의된다. 자유표면과 orbital velocity component는 다음과 같이 주어진다.

$$\eta = \sum_{i=1}^{N} \frac{H_i}{2} \cos(k_{xi}x + k_{yi}y - \omega_i t + \psi_i)$$

$$u = \sum_{i=1}^{N} \frac{H_i}{2} \omega_i \cos^2(\triangle\beta_i) \frac{\cosh(k_i z)}{\sinh(k_i h)} \cos(\theta_i)\cos(\beta_i)$$

$$v = \sum_{i=1}^{N} \frac{H_i}{2} \omega_i \cos^2(\triangle\beta_i) \frac{\cosh(k_i z)}{\sinh(k_i h)} \cos(\theta_i)\sin(\beta_i)$$

$$w = \sum_{i=1}^{N} \frac{H_i}{2} \omega_i \cos^2(\triangle\beta_i) \frac{\sinh(k_i z)}{\sinh(k_i h)} \sin(\theta_i)$$

- 여기서, $\theta_i = k_{ix}x + k_{iy}y - \omega_i t + \psi_i$, $\triangle\beta_i$는 각 component의 방향과 경계면에서 도메인을 향하는 normal 벡터간의 방향의 차이를 의미한다.
- 방향성(directionality)에도 불구하고 경계에서는 외측(outward, $|\triangle\beta| > \pi/2$) 또는 접선 방향(tangential, $|\triangle\beta| = \pi/2$)의 파를 생성할 수 없으므로, 이들을 제거해야만 하며, 단지 각 side에 대해 $\pi/2$보다 작은 각(angle)만을 사용해야만 한다.

㉯ Second order
- First order wave generation 이론으로는 group bound wave가 재현되지 않기 때문에, 정확한 파의 표현을 위해서는 second order effect를 포함하는 것이 필요하다.
- Second order irregular wave의 생성은 개별적인 primary wave component간의 상호작용을 2개씩 고려하여 수행되고, first order method의 상단에 생성된다.
- Longuet-Higgins and Stewart(1960)의 이론이 Torres-Freyermuth et al.(2010)에서와 같이 적용된다.

⑧ Piston wavemaker
- piston-type wavemaker 는 수층(water column) 전체에 대해 constant velocity profile을 생성한다.
- 이 profile은 equilibrium profile이 다른 파(파가 선형장파(linear long wave)인 경우를 제외하고), wavemake근처에 생성되는 evanescent mode라고 불리는 일부 가상파(spurious wave)를 생성하지만, 이 효과는 파가 전파되면서 소멸한다.
- 사용 가능한 자료에 따라 이 경계조건은 4가지의 main case를 가지지만, 시뮬레이션의 first time step에서 단지 2가지의 case로 줄어든다.
- 첫 번째 case(tx)는 wavemaker의 변위와 시간의 시계열을 제공하며, 이는 가장 일반적인 실험 설비의 출력결과이다. 이들로부터 wavemaker의 속도는 아래 식에 제시된 바와 같이 first order forward derivative로 계산할 수 있다.

$$U = \frac{X_{i+1} - X_i}{t_{i+1} - t_i}$$

- 여기서, 시간(t)는 다음과 같이 주어진다. $t_i \leqq t,\ t_{i+1} > t$
- 이 표현식은 homogeneous sampling rate가 필요하지 않기 때문에 상당히 편리하다. 이 결과는 유속의 시계열이 주어질 때, second case(tv)가 된다.
- 세 번째와 네 번째 case는 이전과 동일하나, wavemaker에 추가로 일련의 자유표면고도(free suface level)를 지정한다(txeta, tveta).
- 모든 수리실험의 wavemaker가 이런 feedback을 제공하는 것은 아니지만, 이 방법은 어떤 추가적인 가정 없이 active wave absorption을 trigger할 수 있게 해준다.

(3) 파랑 수치 구현법 (numerical implementation for wave generation)

- 수치 구현법(Numerical implementation)은 모든 파 이론에 대해 같다 (자유표면(free surface)이 제공되지 않을 때 piston-type wavemaker 경우 약간의 변경 존재).
- 세 가지 타입의 cell을 고려한다: wet cells (유체 cell), dry cell (공기 cell, 즉 모든 꼭지점이 free surface level 상단에 위치), partial cell (free surface가 cell의 최저 및 최고 꼭지점 사이에 위치).
- 파 생성은 유속의 값과 VOF 함수(field “alpha1”, α_1이라 지칭)의 설정을 포함하며, 따라서 이들을 구분하여 구현(implementation)해야만 하지만, 대부분의 소스코드에서 이들은 공유된다.
- 압력은 직접적으로 설정하지 않으며 “buoyantPressure” 경계 조건을 사용하여 계산한다.
- OpenFOAM에서 이용할 수 있는 이 특별한 함수는 local density gradient로부터 normal gradient를 계산하며, 이를 통해 경계에 수직인 방향으로 압력의 2차 미분값이 0이 되도록 설정한다.
- total patch area에 대한 patch의 wet area(개별 face area의 합 × α_1)로 patch의 initial still water depth를 측정한다.
- Piston-type wavemaker replication의 경우, 만약 free surface가 제공되지 않는다면, 모든 수층에 상응하는 constant velocity profile이 적용 되며 이는 air velocity가 추가되는 것을 방지하기 위해, 각 cell에서 velocity와 α_1을 곱하여 계산한다.

- Piston-type wavemaker replication의 자유표면고도(free surface elevation)이 제공된다면, 시간에 대한 선형적 보간이 실시된다.
- Piston-type wavemaker를 제외한 파의 경우, free surface 계산을 위한 표현식은 제공되거나 상응하는 파 이론을 사용하며, 결과적으로, 각 time step에서, 이론적 그리고 측정된 free surface level은 active wave absorption을 위해 비교하여 수행한다.
- 반사파가 generation patch에 도달할 가능성을 고려하여 시뮬레이션을 안정화 시킬 필요성을 고려해야 한다.
- "zero gradient"는 boundary face value 의 조건으로 $\frac{\partial q}{\partial x_i} n = 0$ 을 의미하며 n은 face에 대한 수직 방향이다.
- 측정 수면이 이론적 수면(파 이론에 의해 계산된 수면) 보다 높은 것(positive reflected wave, 그림 37의 좌측), 또는 낮은 것(negative reflected wave, 그림 37의 우측)에 따라, 그림 37에 제시된 바와 같이 세 개의 다른 area가 존재한다.
- 이렇게 세 가지 다른 영역("a", "b", "c")과 그들 사이의 두 개의 interface를 생성하며 이들의 구현은 다음에 설명하며, 표 6에 요약하였다.

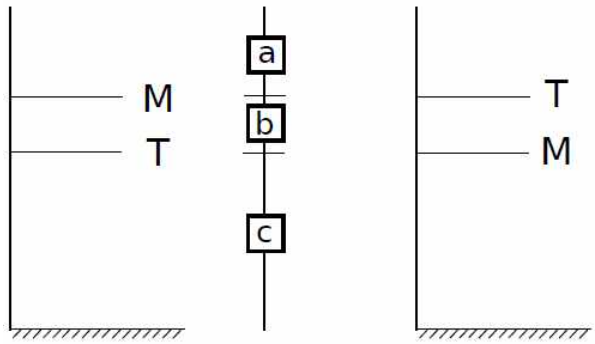


그림 37. Water level in a wave generating patch. "M" stands for measured value, "T" stands for theoretical value. Three zones (a, b and c) and two interfaces are present in each case.

표 6. Overview of the boundary condition values depending on the zones established in 그림 37.

M > T		
	α_1	U
a	0	0
a-b	$\frac{\partial \alpha_1}{\partial x_i} n = 0$	0
b	$\frac{\partial \alpha_1}{\partial x_i} n = 0$	0
b-c	$\frac{\partial \alpha_1}{\partial x_i} n = 0$	$U \cdot \alpha_{1\,calc}$
c	In $\rightarrow$ 1 Out $\rightarrow$ $\frac{\partial \alpha_1}{\partial x_i} n = 0$	U

T > M		
	α_1	U
a	0	0
a-b	$\alpha_{1\,calc}$	$U \cdot \alpha_{1\,calc}$
b	In $\rightarrow$ 1 Out $\rightarrow$ $\frac{\partial \alpha_1}{\partial x_i} n = 0$	U
b-c	In $\rightarrow$ 1 Out $\rightarrow$ $\frac{\partial \alpha_1}{\partial x_i} n = 0$	U
c	In $\rightarrow$ 1 Out $\rightarrow$ $\frac{\partial \alpha_1}{\partial x_i} n = 0$	U

- Zone “a”는 공기에 해당하며($\alpha_1 = 0$) 따라서 속도는 0이다.
- “a”와 “b”의 경계면은 case간에 서로 다름. (파의 속도는 단지 이론적 level 아래에서만 설정되기 때문). M>T 일 때, α_1에 대해 zero gradient 값을 생성하며, 속도는 0. 만약 T>M 일 때는 특별한 과정이 수행된다. 이론적 water level과 cell 간의 교차점(intersection)을 계산하여 cell에 해당하는 α_1을 얻음. 또한, cell wet part의 무게중심(centroid)을 계산하여 그 점의 속도를 계산. 속도는 두 양(quantities)의 곱으로 설정됨. 이는 낮은 Courant number와 안정도를 가지는, interface에서 spurious air velocity가 생기는 것을 방지하기 위함이다.
- Zone “b” 역시 case에 따라 다름. M>T 인 경우, α_1은 zero gradient로 설정되고, 속도는 직접적으로 0임. T>M 일 때, α_1은 water flux가 안쪽이면 1로 설정되고 그렇지 않은 경우는 zero gradient로 설정됨. 속도는 이론값으로 설정된다. 이는 때때로 반사된 진폭이 큰 경우(여러 cell에서 순차적으로), 경계에 water droplet이 나타나게 할 수 있다.
- “b”와 “c”간의 interface는 때에 따라 water above(M>T)와 (T>M)을 가질 수 있지만, 이는 반사파 때문은 아님. 이 조건의 변화는 표 6(좌측과 우측 내용)에 제시된 바와 같이 상황에 따라 행동이 변한 것이다. 첫 번째 case의 경우, α_1은 zero gradient로 설정되고 속도는 앞서 설명된 바와 같은 이유로, T>M인 경우의 zone “a-b”에서와 같은 방법으로 계산됨. 두 번째 case는 만약 물이 흐르고 있다면, 1로 설정되고 그렇지 않으면 zero gradient로 설정됨. 이 case에서 속도는 계

산된 값으로 설정된다.

- 마지막으로, zone “c”는 두 case에서 같으며, 이 값은 측정된 값과 이론적인 값보다 항상 낮기 때문에 물에 해당($\alpha_1 = 1$). 만약, 흐름이 도메인으로 유입된다면, face 상의 α_1은 1로 설정되고, 반대의 경우엔 zero gradient로 설정됨. 이 설정법은 언제나 1로 설정하는 것보다 안정적이며, 이는 “MULES” solver가 0과 1사이의 경계를 보장을 추구한다는 사실에도 불구하고, 이와는 다른 solution이 보다 쉽게 α_1을 음의 값에 도달하게 할 수 있다는 것이 관찰되었기 때문임. 유속은 이론적인 값으로 설정한다.
- Interface가 cell과 일치하면 cell은 자동적으로 interface cell이 된다. 언급된 interface 모두가 cell과 일치하면 (e.q. small reflected waves 또는 T=M), cell의 우선순위는 아래로부터 위로, “c”로부터 “a”로 이다.
- 속도와 α_1은 사용자의 결정에 따라, 다른 방법으로 설정할 수 있음. 첫 번째이자 가장 명확한 방법은 face by face 방법이다. 그들 각각은 다수의 점과 중심(centroid)에 의해 형성된 cell을 소유하며 이들 모두는 서로 다른 좌표계를 보유. face(“a”-“c”)의 belonging zone은 Z 방향의 최고점과 최저점을 사용하여 check되며, 속도는 face의 중심좌표를 사용하여 계산된다. 이 경계조건은 vertical slice에서 생성된 patch의 automatic division을 지원. 이런 zone은 wavemaker 내 개별 paddle의 배열과 비슷하고, 사전에 경계를 수동으로 분할해야 하는 번거로움 없이 이러한 장치에 유사한 복제가 가능하게 하는 정확한 방식으로 행동함. 각 zone 내의 cell들은 paddle의 중심에 맞게 x 및 y 좌표를 잃어버리지만, α_1과 속도계산을 위해 그들의 높이를 유지해야한다.

바. OpenFoam (OlaFlow) 파 흡수 (wave absorption)

- 파의 active absorption은 연안 공학에서 수리 또는 수치 실험의 주요 특징 중 하나이다.
- Prototype scale에서 파는 연구지역으로부터 멀어질 수 있으나 수치실험에서는 이럴 수 없으며, 도메인은 wave basin과 flume같이 dimension이 제한되거나 계산상의 제한으로 인해 무한거리를 가질 수 없다.
- 만약 이를 적절하게 처리되지 않는다면, 실험에 영향을 줄 수 있는 불편한 반사를 유발하여 실험 결과를 왜곡할 수 있다.

(1) 이론

- 과거 outgoing wave를 소멸시키기 위한 첫 번째 방법은 passive wave absorber를 사용하는 것이다.
- 여기에는 dissipative beach, 다공성 물질(porous media), 다공성 판(porous plate) 그리고 인공 수치 감쇠(artificial numerical damping)가 포함된다.
- Absorption은 완벽하게 달성되지 않음. 예를 들어, long wave는 break 되지 않기 때문에, dissipative beach에서도 반사되며, 결과적으로 경계조건에 약한 반사효과를 미친다.
- 수치적으로, passive wave absorber는 다공성 매체(porous media) 또는 relaxation zone으로 재현할 수 있다.
- Passive absorber는 이미 OpenFOAM에 개발되어 있으며 relaxation zone을 사용. 이 기술은 파장의 약 두 배 정도 계산 영역을 증가시키는 파랑 damping zone을 설정해야 하는 명백한 단점을 가지며, 이는 large domain과 prototype application에 상당한 불편을 유발한다.
- 두 번째 방법은 active wave absorption. 이는 실험 기기로 처음 개발되었다.
- 이 시스템은 측정된 크기(feedback)를 기반으로 wavemaker의 움직임을 수정하는 것을 토대로 incoming wave의 재반사를 방지하면서 target wave를 지속적으로 생성한다.
- 이 시스템은 수치모델에도 적용할 수 있으며, 대부분의 경계는 고정되어 있으므로, result wave absorption은 수정된 velocity profile을 경계에 부과하는 것으로 계산할 수 있다.
- Active wave absorption 시스템은 2D, Quasi-3D, 3D의 세 가지로 나눌 수 있다.

(2) 파 흡수 수치 구현법 (numerical implementation)

- Pure active wave absorption의 구현은 단지 속도를 prescribe하는 것만을 필요로 하기 때문에 어렵지 않으며, 압력경계조건은 “buoyantPressure”로 설정하고, α_1은 “zeroGradient”로 설정한다.
- Active wave absorption은 파생성경계(wave generation boundary)를 포함하는 어떤 경계에서도 같은 방법으로 동작한다.
- Absorption 이론의 실제 적용은 경계를 주어진 수의 vertical elements로 나누는

것으로 구성됨. 최소값은 1로 각각의 개별요소에 대해 초기 정수면(initial still water level)이 계산되고 저장됨. 이후 각 time step에서 각 paddle의 실제 수위(actual water level)가 얻어지고 보정속도가 계산된다.

- 보정속도는 이후 각 cell마다 재계산되며, 구해진 값에(포함된 paddle을 확인) 해당 cell의 α_1값을 곱함. 이는 경계에 가까운 air pocket의 전파를 방지하기 위해 필요하다.
- 합성된 속도는 face에 수직인 방향(2D 그리고 full 3D absorption) 또는 주어진 방향(Quasi-3D)에 적용됨. 만약, 이 속도가 positive(in-flux)이면 측정된("M") 수위하의 cell에만 값이 주어지고, 나머지는 0으로 설정된다.
- 반대의 경우(out-flux)에는 모든 cell이 재계산된 속도 값으로 설정됨. 이 방법을 적용하는 주된 이유는 튀는 물방울이 이 patch의 수위보다 높은 곳에 있는 것처럼 하여 안정성을 얻기 위한 것으로, 이것은 물이 도메인 내측으로 흐를 때 전파되지 않게 함. 반대의 경우 이것은 흘러나가게 된다.

사. OpenFoam (OlaFlow) 조파 실습

(1) 조파/흡수 조작방법

- 동시간대에 파를 생성하고 흡수하는데 필요한 여러 파라미터를 소개한다.
- 파의 생성 경계조건은 constant 폴더에 위치한 dictionary file에 의해 제어..
- 기본적으로, 경계조건은 waveDict에서 살펴볼 수 있으며, 이는 수정 가능하다.
- 이는 paddle이 독립적으로 움직임으로써 서로 다른 dictionary name을 가지는 n paddle(즉, directional wave basin)의 움직임을 재현할 때 매우 편리한 방법이다.

```
patchName
{
    type            wave(Alpha|Velocity);
    waveDictName    waveDictName;
    value           uniform 0|(0 0 0);
}
```

- waveDict 파일 내 항목들은 order를 가질 수 있으며, 다른 모든 OpenFOAM 파일과 마찬가지로 //를 사용하여 주석 처리함으로써 줄 끝까지 따르는 내용을 무시할 수 있다.
- 예상치 못한 추가 항목이 포함될 수는 있지만, 이들은 읽혀지지 않는다.
- 여기에는 몇 가지의 값이 필요하며, 이들이 정의되지 않으면 runtime에 self-explanatory error가 발생한다.
- 이는 non-physical value(예를 들어 negative wave height)에서도 마찬가지임.
- 정의되지 않은 파라미터는 기본값이 적용된다.
- 혼동을 피하기 위해서는 dictionary에 모든 파라미터가 항상 포함되는 것이 좋으며, 이는 이후에 나오는 버전에서 기본값이 변경되더라고 계속 같은 결과를 재현할 수 있게 해줄 것이다.
- 이런 파라미터에 대한 값을 설정하는 실제적인 참고자료는 부록 A.2에 제시되었다.

waveType

- 첫 번째 파라미터는 wave의 type이다. 여기서 사용자는 다음 중 하나를 선택할 수 있다.

· **regular** (R) · **solitary** (S)

· **irregular** (I) · **wavemaker** (W)

- 괄호안의 문자는 단지 이 장에서의 내부참조용 문자임. 앞서 설명하였듯, irregular wave case는 단지 Stokes I 파 성분의 선형 합이며, wavemaker는 맞춤(custom-made) wavemaker constant velocity profile을 재현한다.
- 이는 경계조건의 행동을 제어하는 주 스위치이며, 소스코드에서 명확하게 확인할 수 있다.

waveTheory - (R‖S‖W)

- 이 파라미터는 생성되는 파의 type을 제어. 사용자는 기존 tool을 이용하여, case의 특정 파라미터를 기반으로 어떤 값을 사용할지 신중하게 결정해야 한다.
- Irregular wave type의 경우는 Stokes I이 사용되므로, waveTheory는 읽거나 사용되지 않음. 그러나 다른 이론의 사용은 쉽게 구현할 수 있다.

(R)

- 앞서 설명한 바와 같이, 유효한 파랑이론은 다음과 같다.

· **StokesI** · **cnoidal**

· **StokesII**

· **StokesV** · **streamFunction**

(S)

- Solitary wave의 생성은 다음의 한 이론만 제공된다.

· Boussinesq

- 새로운 이론의 추가는 수식을 코딩하고 기존 loop에 다른 else if loop를 추가하면 되므로 그리 어렵지 않다.

(W)

- wavemaker의 경우 제공되는 자료에 따라, 여러 다른 옵션을 적용할 수 있다.

· **tx** · **txeta**

· **tv** · **tveta**

- 이들 모두는 시계열(이름내의 t)을 포함하며, paddle displacement(x), paddle velocity(v), paddle에서 free surface elevation(eta) 등 제공되는 자료에 따라 분류되며, 이들은 보다 정확한 wave absorption을 허용한다.
- 앞서 설명한 것처럼, 처음으로 경계조건이 사용되면 waveTheory가 tveta로 바뀌게 될 것이며, 변경사항을 추적하기 위한, v와 eta(추정된, 이전에 사용하지 않은 경우)와 함께 waveTheoryOrig라고 명명된 추가적인 항목이 dictionary file에 생성된다.

Wave Parameters

- 이것은 항목 자체는 아니지만, 각각의 파 이론에 대한 파를 정확하게 정의하는데 필요한 파라미터를 포함하는 subsection 이다.

(R)

· **waveHeight** in meters · **wavePhase** in radians

· **wavePeriod** in seconds · **waveDir** in degrees

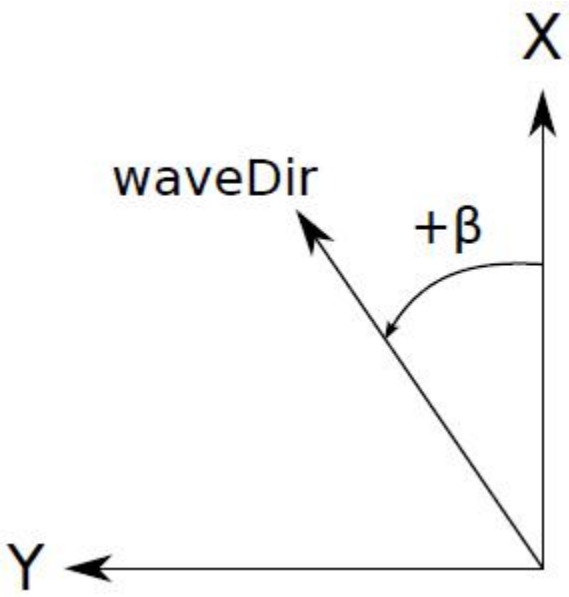


그림 38. Angular reference system

- waveHeight와 wavePeriod가 필요하며, 만약 wavePhase가 주어지지 않는다면, $3\pi/2$가 기본값으로 적용된다. waveDir는 파의 전파 방향으로 degree로 측정되며, X 축으로 0°, Y 축으로 90°이며(그림 38); 기본값은 0 이다.
- 이 설정 파라미터들은 모든 이론에 적용 가능하지만 streamfunction은 Fenton program으로부터 주어지는 추가적인 다음과 같은 정보를 요구한다.

· **waveLength** in meters

· **uMean** in m/s (mean fluid speed in the frame of the wave)

· **Bj** (nondimensional)

· **Ej** (nondimensional)

- Solution.txt에 제공되는 결과들은 모두 무차원이며, 따라서 미터 단위의 파장을 얻기 위해서는 그 값은 수심(h)이 곱해져야 한다.
- 유사하게, 파의 frame에서 mean fluid speed를 계산하기 위해서는 주어진 값이

$\sqrt{gh}$ 와 곱해져야 함. Bj와 Ej는 주어진 값을 사용한다.

- 두 번째 time step에서 경계조건 변수들 사이에 다른 크기의 값이 표시됨. 즉, 주어진 이론에 따라 계산된 waveLength와 첫 번째 time step에서 자동으로 계산되는 waterDepth가 표시된.
- waterDepth는 임의의 값(0보다 큰)으로 설정할 수 있으며 이 값은 absorption에 대한 보정속도의 계산에 이용됨을 주의한다.

(I)

· **waveHeights** in meters · **waveDirs** in degrees

· **wavePeriods** in seconds

· **wavePhases** in radians · **secondOrder** (bool)

- 이전과 이름은 같지만, 숫자의 리스트로써(scalarList type) 단어의 끝에 "s"가 붙어 있음에 주의. 이번에는 이들 모두가 필요하며 추가적으로 secondOrder (bool)로 second order wave generation이 활성화되도록 설정할 수 있다.

(S)

- Solitary wave는 regular type에 대한 limit case로 볼 수 있다.

· **waveHeight** in meters

· **waveDir** in degrees

(W)

- Piston-type wavemaker velocity profile은 또한 waveType에 따라 2 또는 3개 special series가 필요하며, 이들 중 하나는 언제나 timeSeries 이다.

· **timeSeries** in seconds

· **paddlePosition** in meters

· **paddleVelocity** in m/s

· **paddleEta** in meters (free surface at the paddle)

- 중요한 사실은 timeSeries는 주어진 시간 사이에 값이 선형적으로 보간되므로, 균등한 sampling rate를 제공할 필요가 없다는 점이다.
- 추가적으로 tuningFactor라고 불리는 추가적인 변수가 이 wave type에 도입될 수 있음. 이것은 속도와 자유표면고도(free surface elevation)가 곱해진 factor이며, 예상되는 실험 결과(expected laboratory results)와 일치시키는 tuning factor로 이용될 수 있음. 이를 포함하는 이유는 수치 경계는 움직이지 않기 때문에 calibration factor가 필요할 수 있기 때문이며 이것의 기본값은 1이다.

(R‖I‖S‖W)

- 이들 모두에 적용할 수 있는 것은 nPaddles이며, patch가 분할된 vertical slice의 (integer) 개수이다.
- 이들 파라미터는 absorption과 directional wave generation에 중요한 역할을 하며 기본값은 1 이다.
- 전체 patch는 개별적 element로써 동작한다.
- 각각의 paddle 들은 그 중심좌표에 정의되기 때문에, 방향성을 정확하게 재현하고 2D theory를 사용하여 3D 방식으로 동시에 absorb하기 위해서는 충분한 수의 paddle이 필요하다.
- Rule of thumb에 따라 paddle 당 5 vertical column cell을 갖는 것이 효과적이다.
- nPaddles는 cell의 vertical column의 수보다 많이 밀어내면, 기계 정밀도와 round off 에러에 기인한 오류가 발생 될 것이다.
- 때때로 경계에서 보다 부드러운 반응을 얻기 위해서는 초기조건을 약하게(tapered)해야 할 때가 있다.

- 이것은 시뮬레이션의 시작점에서 generation patch 상에 많은 수의 파봉과 파곡을 가질 수 있는 irregular sea state 또는 directional wave의 경우에 적용된다.
- 이것을 위해, 시간이 지나면 사라지는 tSmooth(smoothing time, in seconds)가 도입되었으며, 자유표면고도와 속도는 이 factor에 의해 곱해지면 t=0초에서 0부터 t=tSmooth초에서 1로 선형적으로 변화함. tSmooth의 기본값은 -1이며, 이는 적용되지 않음을 의미한다.

Wave Absorption

- Wave absorption은 boolean variable에 의해 제어됨. genAbs. 이 값이 참이면 wave generation과 active wave absorption이 함께 발생하고 반대의 경우 연결이 끊어짐. 기본적으로는 거짓으로 설정되어 있지만, 파봉과 파곡 사이의 물의 유입-유출의 불균형에 기인한 mean water level의 증가를 막기 위해 언제나 연결하는 것을 추천한다.
- nPaddles는 또한 얼마나 많은 독립적인 absorption zone이 존재하는지를 제어하며, 1보다 큰 값일 때 보다 나은 3D absorption을 허용함. 이전과 같은 규칙이 적용된다.
- Quasi-3D absorption은 absDir에 보정속도가 적용될 방향을 설정하여 적용할 수 있음. 방향은 waveDir에서와 같은 방법에 따라 degree로 주어져야 함. 방향은 도메인의 내측을 가르켜야 하며, 그렇지 않으면 이는 쓸모없고 비생산적이 됨. 만약 값이 360보다 크게 주어진다면, 경계조건은 paddle에 수직인 방향을 선택함. 이것이 기본 적용값이다.
- 파 생성에 필요한 모든 파라미터는 그림 39에 요약되어 있음. 그림에서 사각형 상자내의 단어는 파라미터이고 원형(rounded) 상자의 단어는 옵션들이다. 원(circle) 내의 변수(wavemaker type의 경우에서)는 만약 waveTheory에 이런 문자열이 포함되어 있다면, 그 우측의 파라미터가 필요함을 나타낸다.

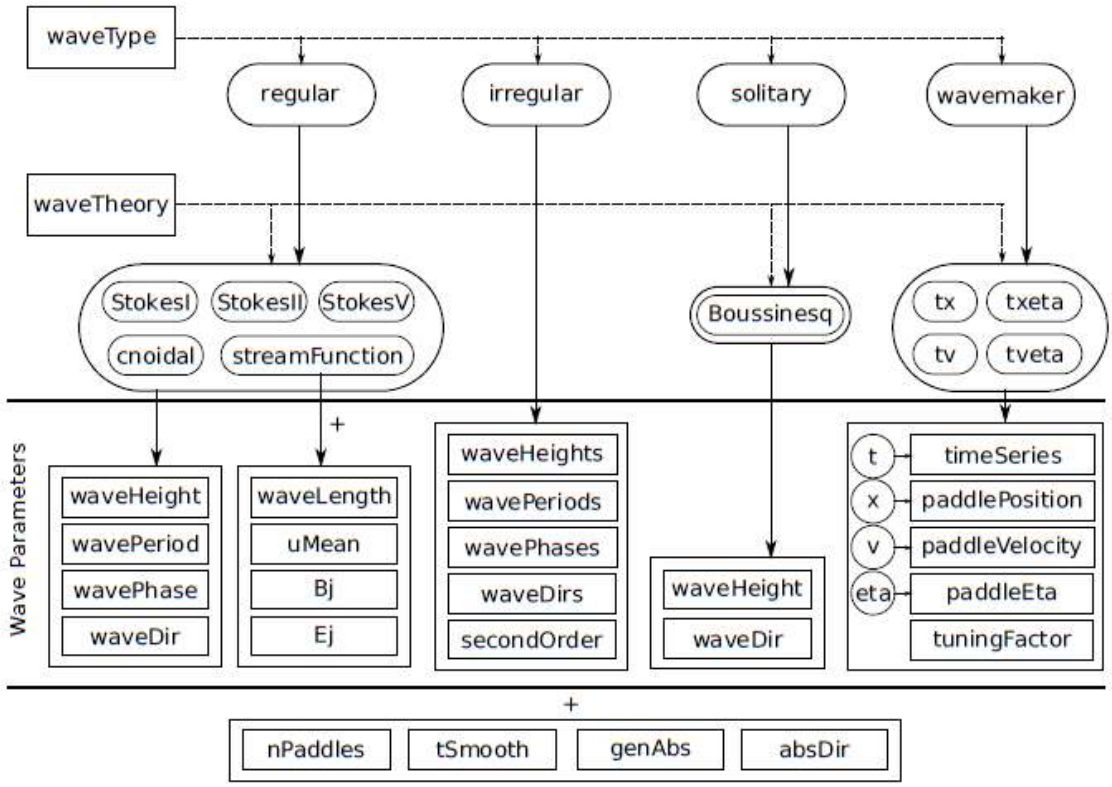


그림 39. Wave generation and absorption parameters

(2) Active 흡수 (absorption)

- Active wave absorption의 설정은 경계조건(즉, U file을 이용한)에서 제어되기 때문에 별도의 dictionary file을 필요로 하지 않는다.
- 필요한 변수들은 absorption의 type에 따라 달라짐, 여기서는 적은 파라미터를 요구하는 full 3D theory를 설명한다.
- nPaddles는 이미 설명한 것처럼, absorption을 위한 개별 요소들의 총 개수이다.
- nEdgeMin과 nEdgeMax는 단지 물만을 추출(3장에 제시된 바와 같이 안정성을 이유로 함)할 수 있는 최소/최대 극치(min/max extreme)에서 paddle의 수이다.
- Min extreme은 만약 patch의 $\Delta X \geq \Delta Y$ 라면 X 좌표의 최소값을 고려하고, 반대의 경우라면 Y 좌표의 최소값이 고려됨. 만약, 값이 정의되지 않았다면, 이들 최신 파라미터들은 0 값을 사용하므로 모든 paddle들은 정상적으로 동작된다.
- 2D absorption theory(3D 도메인에 적용할 경우에도)는 동일한 파라미터를 사용할 수 있음. 만약 Quasi-3D를 활성화해야 한다면 추가적인 파라미터를 정의해야

만 함, absorptionDir로 이는 보정속도가 적용되는 각도이며(그림 38 참조), 도메인의 안쪽을 향한다.

(3) 부록 (조파 관련 부록 모음)

A.1 Boundary conditions

- 경계조건과 각 변수에 사용할 type에 관한 reference guide는 다음과 같음. 대부분의 경우 field value는 0으로 설정됨. 0이 floating point error를 유발할 때는, 0 대신에 아주 작은 값을 사용. 만약 실험이 정지 상태에서 시작한다면 이 값을 사용하는 것은 적절한 방법이다.
- alpha1 또는 p_rgh field의 경우 이러한 상황이 반드시 참값은 아닐 수 있지만, 초기 상태에 이러한 값을 가지는 것은 이들의 계산이 각 time step상에서 독립적으로 수행되기 때문에 계산 결과를 왜곡하지는 않는다. 더욱이 이런 간단한 사실은 ParaView가 t=0일 때 값을 불러들이는 것을 보다 안정적으로 만들어 주며, 그렇지 않으면(충돌이 발생한다면), Skip Zero Time option을 확인해야만 할 수도 있음. zeroGradient type은 field value를 필요로 하지 않는다.

```
alpha1 [0 0 0]
        Inlet:
                type                    waveAlpha ;
                waveDictName            waveDict ;
                value uniform  0;
        Outlet & Wall:
                type                    zeroGradient ;
        Open:
                type                    inletOutlet ;
                inletValue              uniform 0;
                value                   uniform 0;

p rgh [1 -1 -2]
        Inlet & Outlet & Wall:
                type                    buoyantPressure ;
                value                   uniform 0;
```

```
    Open:
        type            totalPressure ;
        p0              uniform 0;
        U               U;
        phi             phi;
        rho             rho;
        psi             none ;
        gamma           1;
        value           uniform 0;

U [0 1 -1]
    Inlet:
        type            waveVelocity ;
        waveDictName    waveDict ;
        value           uniform (0 0 0);
    Outlet:
        type            2DWaveAbsorptionVelocity ;
        absorptionDir   600.0;
        nPaddles        1;
        nEdgeMin        0;
        nEdgeMax        0;
        value           uniform (0 0 0);
    Outlet:
        type            3DWaveAbsorptionVelocity ;
        nPaddles        10;
        nEdgeMin        0;
        nEdgeMax        0;
        value           uniform (0 0 0);
    Open:
        type            pressureInletOutletVelocity ;
        value           uniform (0 0 0);
    Wall:
        type            fixedValue ;
        value           uniform (0 0 0);

k [0 2 -2]
```

```
        Inlet & Outlet:
                type                    zeroGradient ;
        Open:
                type                    inletOutlet ;
                inletValue              uniform 0.0001;
                value                   uniform 0.0001;
        Wall:
                type                    kqRWallFunction ;
                value                   uniform 0.0001;

epsilon [0 2 -3]
        Inlet & Outlet:
                type                    zeroGradient ;
        Open:
                type                    inletOutlet ;
                inletValue              uniform 0.0001;
                value                   uniform 0.0001;
        Wall:
                type                    epsilonWallFunction ;
                value                   uniform 0.0001;

omega [0 0 -1]
        Inlet & Outlet:
                type                    zeroGradient ;
        Open:
                type                    inletOutlet ;
                inletValue              uniform 0.0001;
                value                   uniform 0.0001;
        Wall:
                type                    omegaWallFunction ;
                value                   uniform 0.0001;

nut [0 2 -1]
        Inlet & Outlet:
                type                    calculated ;
                value                   uniform 0;
```

```
Open:
        type            calculated ;
        value           uniform 0;
Wall:
        type            nutWallFunction ;
        value           uniform 0;
```

A.2 waveDict

- 파의 생성과 active wave absorption 경계 조건의 사용법은 필요한 모든 파라미터와 파라미터가 지정되지 않는 경우 기본값이 사용됨과 함께 앞서 제시되었다.
- 이 부록에서는 각 wave type(regular, irregular, solitary, wavemaker)에 대한 하나 또는 그 이상의 다른 dictionary를 설명한다.

```
Regular
        Cnoidal waves:
                waveType                regular ;
                waveTheory              cnoidal ;
                genAbs                  1;
                absDir          0.0;
                nPaddles                1;
                waveHeight              0.45;
                wavePeriod              3;
                waveDir                 0.0;
                wavePhase               4.71238898;

Streamfunction:
                waveType                regular ;
                waveTheory              streamFunction ;
                genAbs                  1;
                absDir          0.0;
                nPaddles                1;
                waveHeight              0.45;
                wavePeriod              3.0;
                waveLength              9.088;
```

```
uMean                   3.102;
waveDir                 0.0;
wavePhase               4.71238898;
Bj
10
(
2.82859350414499 e -01
5.47232613923276 e -02
9.82759610783782 e -03
1.31281446426603 e -03
6.95356271507966 e -05
1.91627990758790 e -05
4.67599434610103 e -06
5.68266097588630 e -07
6.71785159064360 e -07
2.03467109942793 e -07
);
Ej
10
(
1.90754959626264 e -01
7.54514668971716 e -02
2.80425743606401 e -02
1.11437276888261 e -02
4.79770599639068 e -03
2.20447766999100 e -03
1.07033607929517 e -03
5.57098975071516 e - 04
3.34423937410174 e -04
2.71836376724167 e -04
);
```

Irregular

```
waveType                irregular ;
genAbs                  1;
absDir                  0.0;
```

```
    nPaddles                1;
    waveHeights
    3(
    0.000001
    0.000001
    0.000002);
    wavePeriods
    3(
    6.215700
    2.207500
    10.08560);
    wavePhases
    3(
    6.031400
    0.448110
    4.733100);
    waveDirs
    3( 0 );
```

Solitary

```
    waveType                solitary ;
    waveTheory              Boussinesq ;
    genAbs                  0;
    absDir          0.0;
    nPaddles                1;
    waveHeight              0.15;
    waveDir                 0.0;
```

Wavemaker

```
    waveType                wavemaker ;
    waveTheory              tveta ;
    genAbs                  1;
    absDir          0.0;
    nPaddles                1;
    timeSeries
    2( 0 100 );
```

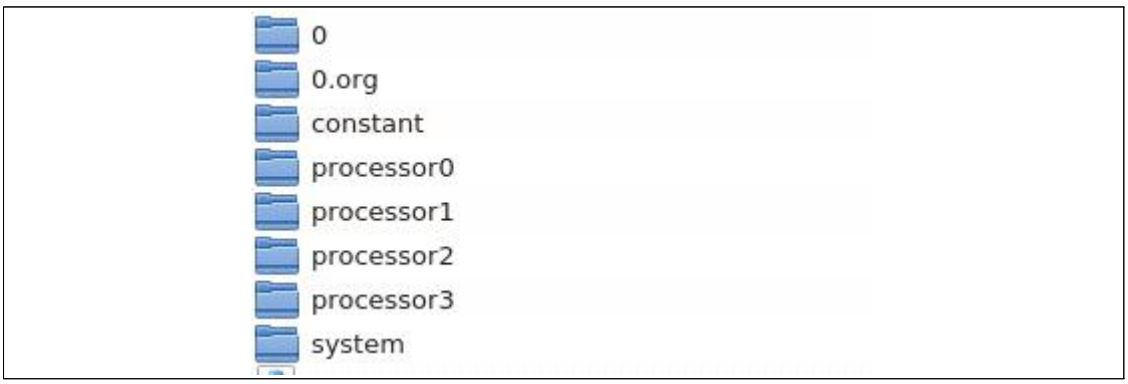
paddleVelocity
2(2 2);
paddleEta
2(0 0);

A.3 OpenFoam 병렬수행

- 병렬수행의 제어는 decomposeParDict 파일을 통해 수행되며, 병렬화 방법은 다음과 같은 방법이 제공된다.

· hierarchical · manual · metis · multilevel · none · scotch · simple · structured

- 제공된 방법 중 추천되는 방법은 scotch 방법이다.
- 이 방법은 단지 subdomains/cores의 수만을 요구하며, processor boundary의 수를 최소화 해주는 방법이다. 병렬화를 위한 decomposition method에 대한 보다 자세한 내용은 $WM_PROJECT_DIR/src/parallel/decompose/ 내의 source code에 제시되어 있다.
- decomposeParDict를 통해 수립된 병렬화 방법은 decomposePar 유틸리티를 사용하여 적용됨. decomposePar 유틸리티를 사용하면 수행 폴더 내에 다음 그림과 같이 processor0~4까지(subdomain이 4인 경우)의 폴더가 자동 생성된다.



- decomposePar 유틸리티를 통해 도메인이 분해되면, 다음의 명령어로 병렬 수행을 실시한다.

```
$> mpirun –np 4 olaFlow –parallel
```

- 병렬 수행을 통해 계산된 결과는 reconstructPar 유틸리티를 사용하여 하나로 모을 수 있다.
- 일반적으로 olaFlow 병렬연산 방법은 다음과 같다.

```
$> blockMesh
$> checkMesh
$> setFields
$> decomposePar
$> mpirun -np 4 renumberMesh -overwrite -parallel
$> mpirun -np 4 olaFlow -parallel
$> reconstrucPar
$> paraFoam 또는 paraFoam -builtin
```

사. 모델결과 가시화

- OpenFOAM 모델을 수행한 후 예측된 결과의 확인을 보다 손쉽게 할 수 있도록 하는 가시화 프로그램을 파이썬 스크립트를 이용하여 구축하고 이의 사용법을 설명한다.

- 윈도우상의 그래픽 프로그램 등에 적용할 수 있도록 OpenFOAM의 예측결과를 변환하는 포트란 프로그램을 작성한다.

(1) 가시화 스크립트

① run_plot.py

- 수위 시계열을 추출하는데 필요한 명령어 및 후처리 코드를 자동으로 실행하는 메인 스크립트로 다음 과정을 수행한다.

- OpenFOAM에서 계산된 결과 파일 중 alpha.water(Air와 Water를 0과 1로 구분한 파일)에서 원하는 지점의 정보를 추출하여 읽고 쓴다.
- 적분을 통해 Air와 Water의 경계 셀(Cell)의 수위를 계산하고 저장한다.
- 저장된 계산 수위 정보를 이용하여 원하는 정점의 시간에 따른 수위 변화시계열을 가시화한다.

- 각각의 코드에 대한 상세설명은 다음 절에 기술하였으며, 각 스크립트는 python2와 python3에서 사용가능하다.

```
## run_plot.py

import os

## postProcess
os.system('postProcess -func sampleDict')

## postSensVOF
os.system('python3 postSensVOF.py')

## plotSensVOF
os.system('python3 plotSensVOF.py')
```

② sampleDict

- sampleDict는 OpenFOAM의 시간대별 출력결과 정보를 추출하기 위한 설정파일로 다음과 같이 실행한다.

$ postProcess -func sampleDict

- sampleDict 파일은 system 폴더내에 위치하며 파일내에 Guage의 종류와 위치정보, 추출하고자하는 field명 등을 지정하여 실행한다.
- sampleDict를 수행하면 postProcessing/sampleDict 폴더가 자동으로 생성되고 생성된 폴더내에 시계열 정보가 저장된다.
- 다음 그림은 smpleDict의 예로 setFormat과 surfaceFormat은 raw형식으로, interpolationScheme은 cellPoint로 설정하여 GaugeVOF01, GaugeVOF02, GaugeVOF03의 세 지점에서 결과를 추출토록 한 것이다.
- 파일 내용 중 axis는 추출 좌표의 순서를 지정하는 것이고, start와 end는 추출 시작 위치와 종료 위치를 입력하는 부분이다.

```
/*---------------------------------*- C++ -*----------------------------------*|
 =========                 |                                                 |
| \      /  F ield          | OpenFOAM: The Open Source CFD Toolbox |
|  \    /   O peration      | Version:  5                                     |
|   \  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/

FoamFile
{
    version          2.0;
    format           ascii;
    class            dictionary;
    location         system;
    object           sampleDict;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
type sets;
libs ('libsampling.so');
setFormat raw;
surfaceFormat raw;
interpolationScheme cellPoint;

sets
(
    GaugeVOF01
    {
        type      midPointAndFace;
        axis      xyz;
        start     (560.0 0.0 -26.0);
        end       (560.0 0.0  20.0);
    }
    GaugeVOF02
    {
        type      midPointAndFace;
        axis      xyz;
```

```
        start    (0.0 0.0 -26.0);
        end      (0.0 0.0  20.0);
    }
    GaugeVOF03
    {
        type     midPointAndFace;
        axis     xyz;
        start    (930.0 0.0 -26.0);
        end      (930.0 0.0  20.0);
    }

);

surfaces          ();

fields            (
                      alpha.water
                  );
```

③ postSensVOF.py

- sampleDict를 수행하여 계산된 정보를 이용하여 각 gauge 상의 수위를 계산하는 스크립트. VOF기법을 적용하여 계산된 정보를 바탕으로 각 위치의 수위를 계산하며, 수행 시 GaugesVOF 폴더가 자동으로 생성되고 계산된 수위시계열이 저장됨. 스크립트의 내용은 다음과 같다.

```
#!/usr/bin/python

import os

pathname = os.path.abspath('.')
savePath = os.path.join(pathname,'gaugesVOF')
if not os.path.isdir(savePath):
    os.makedirs(savePath)

postPath = os.path.join(pathname,'postProcessing')
if os.path.isdir(postPath):
#    postPath = 'postProcessing/sets'
    postPath = 'postProcessing/sampleDict'
else:
    postPath = 'sets'

# List of time dirs in order
a = os.listdir('./'+postPath)
#a.sort(lambda a,b: cmp(float(a), float(b)))
a.sort(key=lambda a: float(a))
```

```
# Get number of sensors
dir1 = os.path.join(pathname,postPath,a[int(len(a)/2.0)])
b = os.listdir(dir1)
nSens = 0
index = []
for i in range(len(b)):
    test1 = b[i].find('VOF') + 1
    test2 = b[i].find('alpha') + 1
    if test1 and test2:
        index.append(i)
        nSens += 1

first = True

for i in range(nSens):
    # Create files to write
    fileName = b[index[i]][0:b[index[i]].find('_')]
    fileW = open(os.path.join(savePath,fileName), 'w')
    print ('Sensor ' + '%i' % int(i+1) + ' of ' + '%i' % nSens + '.')

    # Read files time by time
    for j in range(len(a)):
        directory = os.path.join(pathname,postPath,a[j])
        try:
            fileR = open(os.path.join(directory,b[index[i]]), 'r')
        except:
            print ('WARNING - File not present: ' + os.path.join(directory,b[index[i]]))
        else:
            data = fileR.read()
            fileR.close()
            data = data.split('\n')

            if first: # First time step
                coord = j
                first = False

            x = []
            y = []
            z = []
            alpha = []

            # x y z alpha1 calculation
            for k in range(len(data)-1):
```

```
                line = data[k]
                line = line.split('\t')
                # x = float(line[0]) # y = float(line[1])
                # z = float(line[2]) # pres = float(line[3])

                z.append(float(line[2]))
                alpha.append(float(line[3]))

                if j == coord: # First time step
                    # Create coordinate files
                    fileWXYZ = open(os.path.join(savePath,fileName + '.xy'), 'w')
                    fileWXYZ.write( line[0] + line[1] )
                    fileWXYZ.close()

            # Integrate in Z
            wLevel = z[0]
            for k in range(len(z)-1):
                wLevel = wLevel + alpha[k]*(z[k+1]-z[k])

            # Write to file
            time = a[j]
            fileW.write(time + ' ' + '%.6f' % wLevel + '\n')

    fileW.close()

print ('Done')
```

④ plotSensVOF.py

- postSensVOF.py 스크립트를 이용하여 계산된 결과를 가시화하는 스크립트이며 ./GaugesVOF 폴더 내의 저장결과를 인식하여 시계열 그림을 작성하며 스크립트의 내용은 다음과 같다.

```
#!/usr/bin/python

import os
from pylab import *

pathname = os.path.abspath('.')
readPath = os.path.join(pathname,'gaugesVOF')
a = os.listdir(readPath)
```

```
# Sorting
remove = []
for i in range(len(a)):
    if (a[i].rfind('.')+1): # Includes point
        remove.append(i)
remove.reverse()
for i in remove:
    a.pop(i)
#a.sort(lambda a,b: cmp(int(a.split('F')[1]), float(b.split('F')[1])))
a.sort(key=lambda a: a.split('F')[1])
# Plot
index = 0
indexFig = 0
for gauge in a:
    index = index + 1
    if index >= 4 or index == 1:
        index = 1
        indexFig = indexFig + 1
        figure(num=indexFig)

    subplots_adjust(hspace=1)

    fileR = open(os.path.join(readPath,gauge), 'r')
    data = fileR.read()
    fileR.close()
    data = data.split('\n')
    x = []
    y = []
    for i in range(len(data)-1):
        line = data[i]
        line = line.split(' ')
        x.append(float(line[0]))
        y.append(float(line[1]))

    subplot(3,1,index)
    plot(x,y)
    xlabel('t (s)')
    ylabel('$h + \eta$ (m)')
    title(gauge)

show()
```

⑤ plotSensVector.py

- foamToVTK를 실행하여 생성된 VTK 파일을 이용하여 vector를 가시화하는 스크립트이며 plotSensVector.py 스크립트의 내용은 다음과 같다.

```
#! /usr/bin/python

import vtk
from numpy import zeros
import matplotlib.pyplot as plt
import numpy as np

def data_read(filename):
    print('- Data Read')
    a=vtk.vtkDataSetReader()
    a.SetFileName(filename)
    a.ReadAllScalarsOn()
    a.ReadAllVectorsOn()
    a.Update()
    datas=a.GetOutput()
    d = datas.GetPointData()
    array = d.GetArray('U')
    wtlvl = d.GetArray('alpha.water')
    size = array.GetNumberOfTuples()
    comp = array.GetNumberOfComponents()

    return(datas, d, array, wtlvl, size, comp)

def xyz_out(datas):
    print('    > xyz Read')
    points1=datas.GetPoints()
    npts1 = points1.GetNumberOfPoints()
    x1 = zeros(npts1)
    y1 = zeros(npts1)
    z1 = zeros(npts1)
    for i in range(npts1):
        pt = points1.GetPoint(i)
        x1[i] = pt[0]
        y1[i] = pt[1]
        z1[i] = pt[2]

    return(npts1, x1, y1, z1)
```

```
def uw_out(array):
    print('     > U, alpha.water Read')
    nvls1 = array.GetNumberOfTuples()
    nwls1=wtlvl.GetNumberOfTuples()
    ux1 = zeros(nvls1)
    uy1 = zeros(nvls1)
    uz1 = zeros(nvls1)
    wl1 = zeros(nwls1)
    for i in range(0, nvls1):
        U = array.GetTuple(i)
        ux1[i] = U[0]
        uy1[i] = U[1]
        uz1[i] = U[2]
    for i in range(0, nwls1):
        W = wtlvl.GetTuple(i)
        wl1[i] = W[0]

    return(nvls1, nwls1, ux1, uy1, uz1, wl1)

def sep_airwater(npts1,nwls1,nvls1, x1, y1, z1, ux1, uy1, uz1, wl1):
    xx = zeros(npts1)
    zz = zeros(npts1)
    wl = zeros(nwls1)
    uxx = zeros(nvls1)
    uzz = zeros(nvls1)
    for i, j in enumerate(zip(x1,z1,wl1,ux1,uz1)):
        if j[2] == 1.0:
            xx[i]=j[0]
            zz[i]=j[1]
            wl[i]=j[2]
            uxx[i]=j[3]
            uzz[i]=j[4]
        else:
            xx[i]=j[0]
            zz[i]=j[1]
            wl[i]=j[2]
            uxx[i]=None
            uzz[i]=None

    return(xx, zz, wl, uxx, uzz)

def plot(xx, zz, uxx, uzz):
```

```
    print('- Plot')
    plt.figure()
    Q = plt.quiver(xx[::1000], zz[::1000], uxx[::1000], uzz[::1000], units='width', scale=50)
    qk = plt.quiverkey(Q, 0.9, 0.9, 1, r'$2 \frac{m}{s}$', labelpos='E', coordinates='figure')
    plt.xlim(0, 930)
    plt.ylim(-26, 20)
    plt.show()
    print('- Done')

if __name__ in '__main__':
    # vtkfile path
    filename='/home/shkim/OpenFOAM/olaFlow/test/LES_case/VTK/LES_case_118.vtk'
    # vtkfile read
    datas, d, array, wtlvl, size, comp = data_read(filename)
    # xyz coordinate
    npts1, x1, y1, z1 = xyz_out(datas)
    # u, v, alpha.water read
    nvls1, nwls1, ux1, uy1, uz1, wl1 = uw_out(array)
    # seperate air / water
    xx, zz, wl, uxx, uzz = sep_airwater(npts1,nwls1,nvls1, x1, y1, z1, \
                                ux1, uy1, uz1, wl1)
    # vecter plot
    plot(xx, zz, uxx, uzz)
```

(2) 스크립트 실행 예

① 잠제 전·후면의 시계열 결과 도출

- 다음의 그림과 같이 잠제 주변의 영역에 대해 테스트된 OpenFOAM의 예측결과를 앞서 설명된 스크립트를 이용하여 가시화한 예를 설명한다.
- 결과의 추출은 파입사 경계로부터 650, 550, 450 m 떨어진 지점이다.

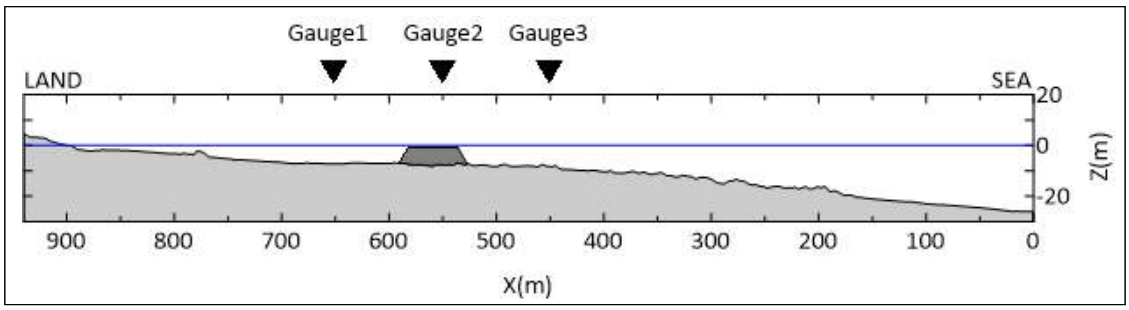


그림 40. 가시화 시계열 정점 위치도

② 가시화 예

- 스크립트의 실행 결과는 다음과 같다.

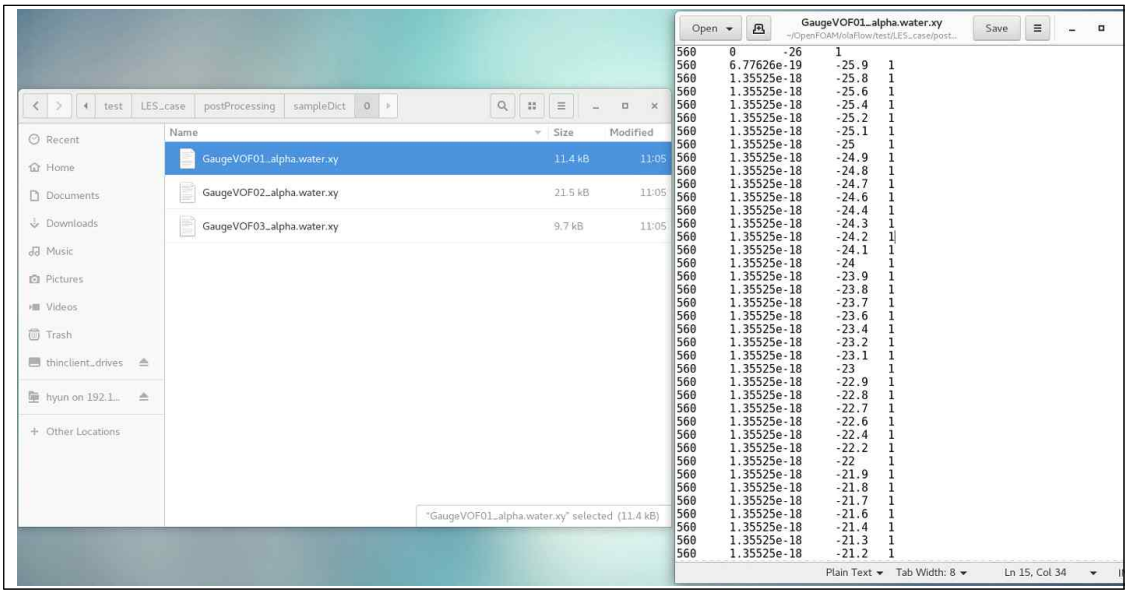


그림 41. postProcessing 결과 파일의 예

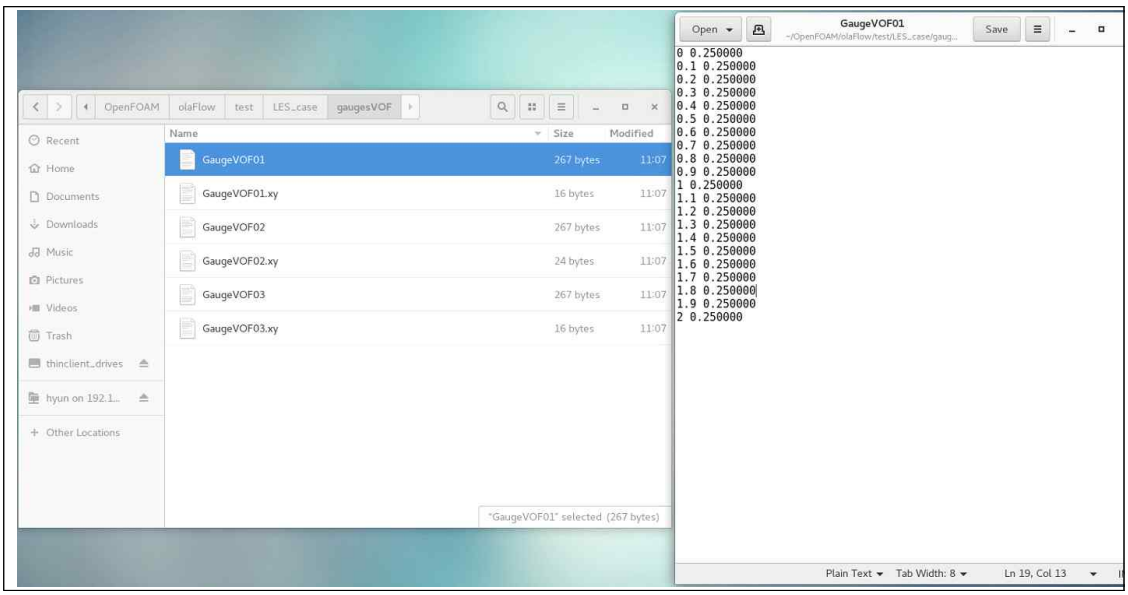


그림 42. postSensVOF.py 결과 파일의 예

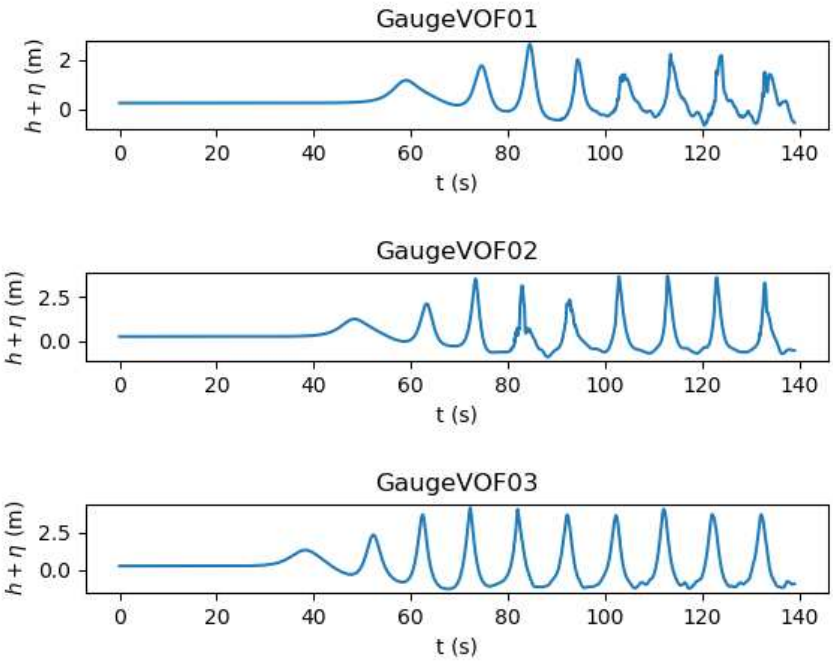


그림 43. 수위 시계열 예

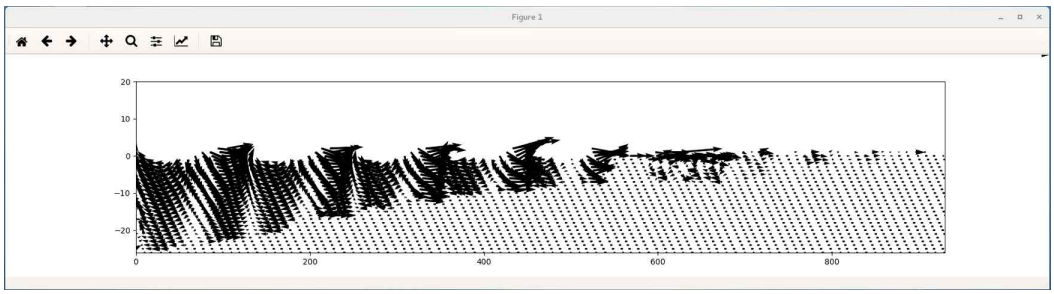


그림 44. 벡터도 예

(2) 예측결과 처리

① OpenFOAM 결과

- OpenFOAM 모델의 예측결과는 지정된 출력시간에 맞추어 독립적인 폴더를 자동으로 생성하고 폴더 내에 시간대별 결과가 저장된다.

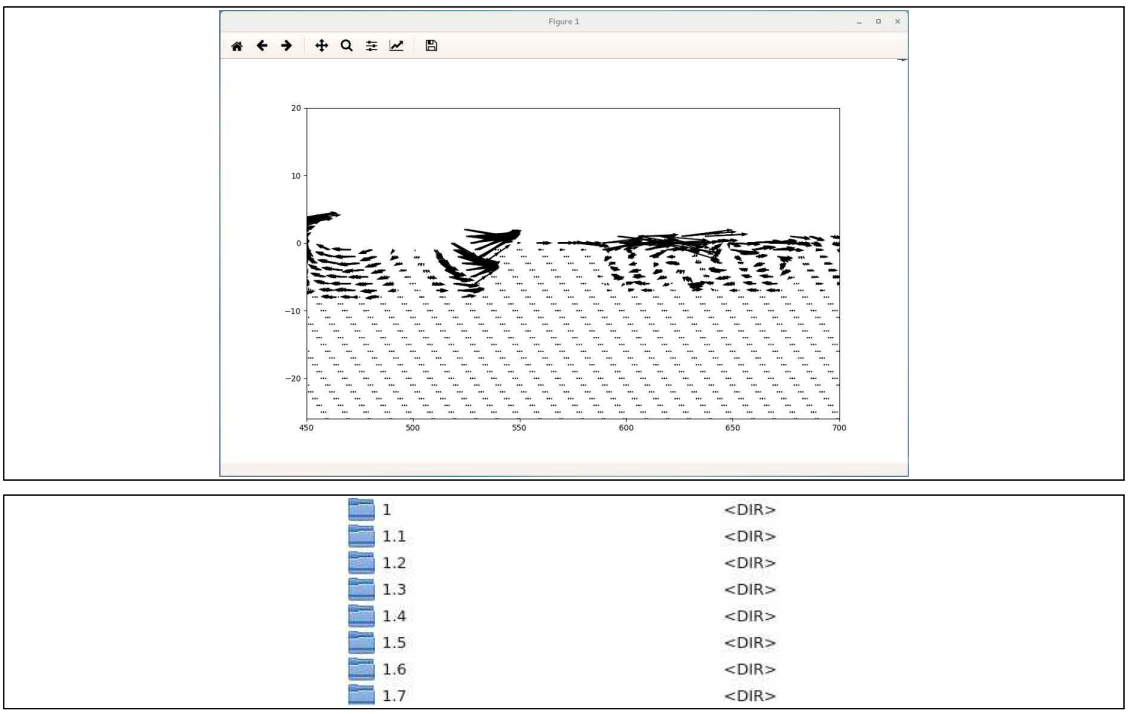
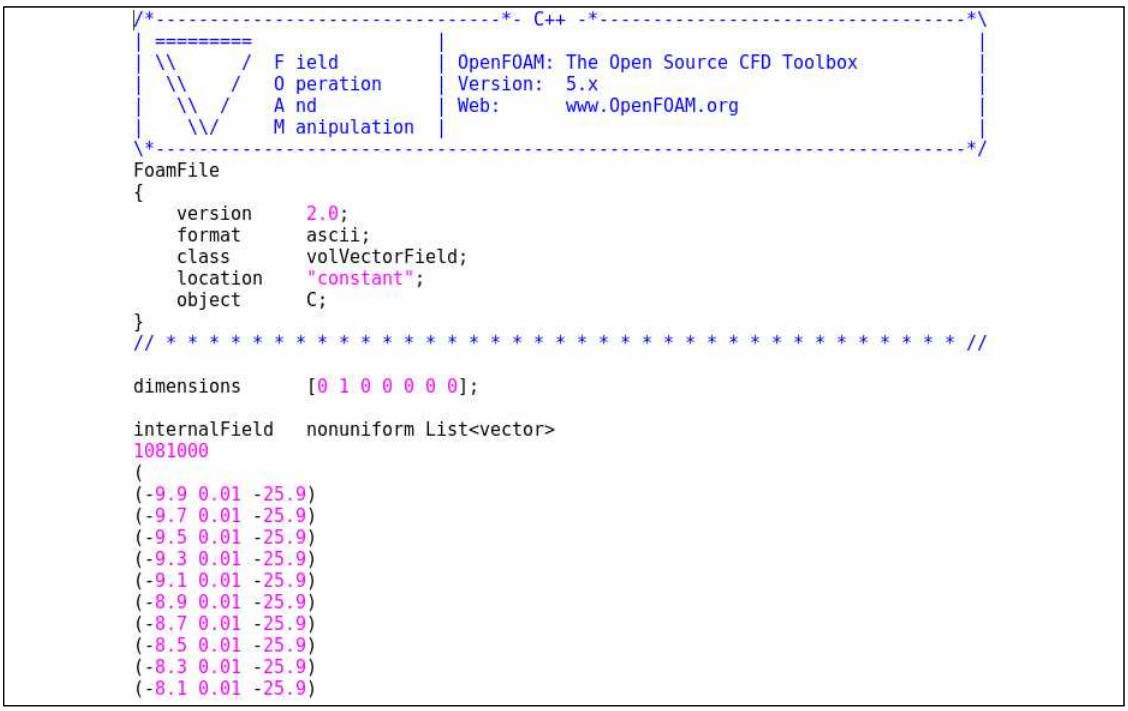


그림 45. OpenFOAM 출력 폴더 구조 예

- 생성된 각 폴더내에는 각 격자의 VOF, 유속, 압력 등의 출력정보가 저장되며 이 정보들을 격자의 중심좌표에 정의된다.
- OpenFOAM의 격의 격자의 꼭지점의 좌표정보는 /constant/polyMesh/points 파일에 지정되어 있지만, 격자의 중심좌표는 지정되어 있지 않으므로, 다음과 같이 후처리 유틸리티를 실행하여 격자의 중심좌표를 정의한다.

$ postProcess -func writeCellCentres

- writeCellCentres를 실행하면 폴더내에 C, Cx, Cy, Cz 등의 좌표정보가 생성되며, 이중 C 파일의 내용은 다음과 같다.



```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  5.x                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    location    "constant";
    object      C;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 0 0 0 0 0];

internalField   nonuniform List<vector>
1081000
(
(-9.9 0.01 -25.9)
(-9.7 0.01 -25.9)
(-9.5 0.01 -25.9)
(-9.3 0.01 -25.9)
(-9.1 0.01 -25.9)
(-8.9 0.01 -25.9)
(-8.7 0.01 -25.9)
(-8.5 0.01 -25.9)
(-8.3 0.01 -25.9)
(-8.1 0.01 -25.9)
```

그림 46. 중심좌표 출력 예

② OpenFOAM 결과 처리

- 라그란지안 입자추적 모델과 윈도우상의 그래픽 프로그램 등에서 OpenFOAM 모델의 예측결과를 사용하기 위해서는 각 폴더별로 예측된 결과를 변환하는 과정이 필요하므로, 이를 위해 포트란 변환프로그램을 작성한다.
- 변환프로그램은 매 출력폴더내의 U, alpha.water, porisity 정보와 격자좌표 정보를 읽은 후 이를 변환하여 각 격자 중심에서의 유속과 수위 정보 등을 시간대별로 출력하여 주며 간단한 변환작업을 통하여 윈도우상의 그래픽 프로그램을 이용한 가시화를 실시할 수 있다.

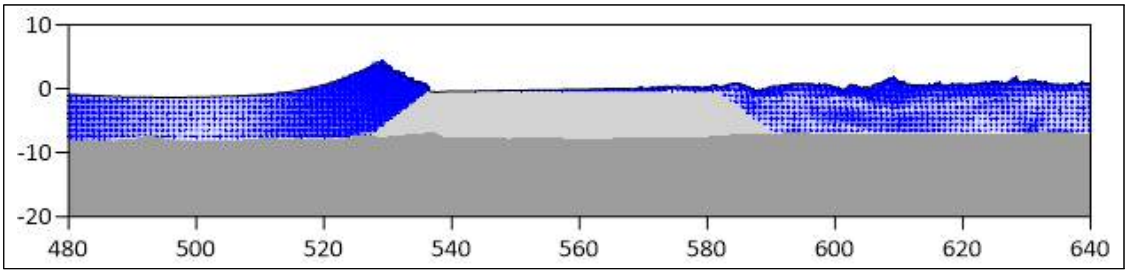


그림 47. 결과 가시화 예

- 후처리프로그램의 내용은 다음과 같음.

```
program convertopenfoam
implicit none
!-------------------------------------------------------------
type OF_CVT
  integer                 :: i,j,k,stat
  integer                 :: x,y,z,notime,fnotime
  integer                 :: xno,yno,zno
  integer                 :: iwskip
  real                    :: stime,dtime,ftime,dx,dy,dz
  character(len=100),dimension(:),allocatable :: fln1,fln2,fln3
  character(len=100) :: fln4,fln5,dir1,dir2
  character(len=1)    :: idx1,idx2,idx
  real,dimension(:,:,:),    allocatable :: xpos,ypos,zpos
  real,dimension(:,:,:),    allocatable :: xnod,ynod,znod
  real,dimension(:,:,:),    allocatable :: velu,velv,velw
  real,dimension(:,:),      allocatable :: wlevel
  real(8),dimension(:),     allocatable :: foltime
  real,dimension(:,:,:),    allocatable :: alphaw
  integer,dimension(:,:,:),    allocatable :: poridx
  real,dimension(:,:),      allocatable :: wdepth
end type OF_CVT
type(OF_CVT),pointer :: Me
allocate(Me)
!-------------------------------------------------------------
Me%xno      =  4700                 ! x grid no
Me%yno      =     1                 ! y grid no
Me%zno      =   230                 ! z grid no
Me%stime    =   0.0                 ! start time
Me%ftime    = 120.0                 ! end time
Me%dtime    =   0.1                 ! printed delta T
```

```
Me%notime  = Me%ftime/Me%dtime
Me%fnotime = Me%notime + 1
Me%idx1    = '('
Me%idx2    = ')'
Me%iwskip  = 1                    ! for writing interval
Me%dx      = 0.2D0
Me%dy      = 0.2D0
Me%dz      = 0.2D0
write(*,*)'Start Time =  ',Me%stime,'End Time  =  ',Me%ftime
write(*,*)'Rsults Time = ',Me%dtime,'Total No. = ',Me%notime
 call convertstep
contains
subroutine convertstep
  call mkpath
  call readnodepoints
  call readcellpoints
  call readporosityidx
  call readwriteresults
end subroutine convertstep
subroutine mkpath
!---------------------------------------------------------------
integer :: i,ctime
!---------------------------------------------------------------

  allocate(Me%foltime(Me%fnotime))
  allocate(Me%fln1(Me%fnotime))
  allocate(Me%fln2(Me%fnotime))
  allocate(Me%fln3(Me%fnotime))
  Me%foltime(1) = 0
  write(Me%dir1,'("./0/")')
  Me%fln1(1) = trim(Me%dir1)//'alpha.water'
  Me%fln2(1) = trim(Me%dir1)//'C'
  Me%fln3(1) = trim(Me%dir1)//'U'
  do i = 2,Me%fnotime
    Me%foltime(i) = Me%foltime(i-1)+Me%dtime
    ctime = int(Me%foltime(i))
    if(mod(Me%foltime(i),float(ctime)) < 0.00001)then
      if     (Me%foltime(i) < 10)then
        write(Me%dir1,'("./",i1.0,"/")') int(Me%foltime(i))
      elseif(Me%foltime(i) < 100)then
        write(Me%dir1,'("./",i2.0,"/")') int(Me%foltime(i))
      elseif(Me%foltime(i) < 1000)then
        write(Me%dir1,'("./",i3.0,"/")') int(Me%foltime(i))
```

```
                endif
              else
                if      (Me%foltime(i) < 10)then
                  write(Me%dir1,'("./",f3.1,"/")') Me%foltime(i)
                elseif(Me%foltime(i) < 100)then
                  write(Me%dir1,'("./",f4.1,"/")') Me%foltime(i)
                elseif(Me%foltime(i) < 1000)then
                  write(Me%dir1,'("./",f5.1,"/")') Me%foltime(i)
                endif
              endif
              Me%fln1(i) = trim(Me%dir1)//'alpha.water'
              Me%fln2(i) = trim(Me%dir1)//'C'
              Me%fln3(i) = trim(Me%dir1)//'U'
              Me%fln4 = './constant/polyMesh/points'
              Me%fln5 = './0/porosityIndex'
!             write(*,'(a,1x,f5.1,1x,a)') 'open file',Me%foltime(i),'sec'
!             write(*,*) trim(Me%fln1(i))
!             write(*,*) trim(Me%fln2(i))
!             write(*,*) trim(Me%fln3(i))
!             write(*,*) trim(Me%fln4)
            enddo
          end subroutine mkpath
          subroutine readnodepoints
          !---------------------------------------------------------------
          integer               :: i,j,k,stat,skip
          integer               :: lstr,mstr,rstr
          character(len=1)   :: idx
          character(len=20) :: lidx,midx,ridx
          !---------------------------------------------------------------
          write(*,*) "reading node position"
          open(10,file=trim(Me%fln4),status='old')

          skip = 0

          do
            read(10,'(a)') idx
              skip = skip + 1
              if(trim(idx) == trim(Me%idx1))then
                exit
              endif
          enddo
          rewind(10)
          do k = 1,skip
```

```
    read(10,*)
  enddo
  allocate(Me%xnod(Me%xno+1,Me%yno+1,Me%zno+1))
  allocate(Me%ynod(Me%xno+1,Me%yno+1,Me%zno+1))
  allocate(Me%znod(Me%xno+1,Me%yno+1,Me%zno+1))
  do k = 1,Me%zno+1
    do j = 1,Me%yno+1
      do i = 1,Me%xno+1
        read(10,*)lidx,midx,ridx
        lstr = len_trim(lidx)
        mstr = len_trim(midx)
        rstr = len_trim(ridx)
        read(lidx(2:lstr),  *) Me%xnod(i,j,k)
        read(midx(1:mstr),  *) Me%ynod(i,j,k)
        read(ridx(1:rstr-1),*) Me%znod(i,j,k)
      enddo
    enddo
  enddo
  close(10)
  end subroutine readnodepoints
  subroutine readcellpoints
  !-------------------------------------------------------------
  integer                :: i,j,k,stat,skip
  integer                :: lstr,mstr,rstr
  character(len=1)   :: idx
  character(len=20) :: lidx,midx,ridx
  !-------------------------------------------------------------
  write(*,*) "reading cell center position"
  open(10,file=trim(Me%fln2(1)),status='old')

  skip = 0

  do
    read(10,'(a)') idx
      skip = skip + 1
      if(trim(idx) == trim(Me%idx1))then
        exit
      endif
  enddo
  rewind(10)
  do k = 1,skip
    read(10,*)
  enddo
```

```
allocate(Me%xpos(Me%xno,Me%yno,Me%zno))
allocate(Me%ypos(Me%xno,Me%yno,Me%zno))
allocate(Me%zpos(Me%xno,Me%yno,Me%zno))
do k = 1,Me%zno
  do j = 1,Me%yno
    do i = 1,Me%xno
      read(10,*)lidx,midx,ridx
      lstr = len_trim(lidx)
      mstr = len_trim(midx)
      rstr = len_trim(ridx)
      read(lidx(2:lstr),  *) Me%xpos(i,j,k)
      read(midx(1:mstr),  *) Me%ypos(i,j,k)
      read(ridx(1:rstr-1),*) Me%zpos(i,j,k)
    enddo
  enddo
enddo
close(10)
end subroutine readcellpoints
subroutine readporosityidx
!-------------------------------------------------------------
integer               :: i,j,k,m,stat,skip
integer               :: lstr,mstr,rstr
character(len=1)  :: idx
character(len=20) :: lidx,midx,ridx
!-------------------------------------------------------------
write(*,'(a,f10.2)') "porosity index"
allocate(Me%poridx(Me%xno,Me%yno,Me%zno))
  open(10,file=trim(Me%fln5),status='old')

  skip = 0

  do
    read(10,'(a)') idx
      skip = skip + 1
      if(trim(idx) == trim(Me%idx1))then
        exit
      endif
  enddo

  rewind(10)

  do k = 1,skip
    read(10,*)
```

```
            enddo
            do k = 1,Me%zno
              do j = 1,Me%yno
                do i = 1,Me%xno
                  read(10,*) Me%poridx(i,j,k)
                enddo
              enddo
            enddo
            close(10)
          end subroutine readporosityidx
          subroutine readwriteresults
          !-------------------------------------------------------------
          integer               :: i,j,k,m,stat,skip,ist
          integer               :: lstr,mstr,rstr
          character(len=1)  :: idx
          character(len=20) :: lidx,midx,ridx
          real                  :: elevation
          !-------------------------------------------------------------
          write(*,*) "readwrite results"
          open(099,file='./ofgrd.grd',status='unknown')
          open(100,file='./ofvel.dat',status='unknown')
          open(101,file='./ofele.dat',status='unknown')
          open(102,file='./depth.dat',status='unknown')

          allocate(Me%alphaw(Me%xno,Me%yno,Me%zno))
          allocate(Me%wlevel(Me%xno,Me%yno))
          allocate(Me%velu(Me%xno,Me%yno,Me%zno))
          allocate(Me%velv(Me%xno,Me%yno,Me%zno))
          allocate(Me%velw(Me%xno,Me%yno,Me%zno))
          allocate(Me%wdepth(Me%xno,Me%yno))
           ist = 1
          do m = 1,Me%fnotime,Me%iwskip
            call readalphawater(m)
            write(102,500) Me%foltime(m)
            write(101,500) Me%foltime(m)
            write(100,500) Me%foltime(m)
! Integrate in Z
            do j = 1,Me%yno
              do i = 1,Me%xno
                  elevation = Me%znod(i,j,1)
                do k = 1,Me%zno-1
                  elevation = elevation
     1              + Me%alphaw(i,j,k)*(Me%znod(i,j,k+1)-Me%znod(i,j,k))
```

```
                enddo
              Me%wlevel(i,j)=elevation
              write(101,501)Me%xpos(i,j,1),Me%ypos(i,j,1),Me%wlevel(i,j)
              enddo
            enddo
!! read vekocity
            if (m == 1)then
              Me%velu=0.0
              Me%velv=0.0
              Me%velw=0.0
            else
              call readvelocity(m)
            endif
            do j = 1,Me%yno
              do i = 1,Me%xno
                do k = 1,Me%zno
!                 if(Me%alphaw(i,j,k) < 0.001)then
                  if(Me%zpos(i,1,k) > Me%wlevel(i,1))then
                    Me%velu(i,1,k) = 0.0D0
                    Me%velv(i,1,k) = 0.0D0
                    Me%velw(i,1,k) = 0.0D0
                  endif
                  write(100,501)
     1                          Me%velu(i,1,k),Me%velv(i,1,k),Me%velw(i,1,k)
!                 write(100,501)Me%xpos(i,1,k),Me%ypos(i,1,k),Me%zpos(i,1,k),
!      1                          Me%velu(i,1,k),Me%velv(i,1,k),Me%velw(i,1,k)
!      1                         ,SQRT(Me%velu(i,1,k)**2+Me%velw(i,1,k)**2)
!      1                    ,atan2(Me%velw(i,1,k),Me%velu(i,1,k))*180./3.141592

                enddo
!               ! find depth for x-direction
!               do k = 1,Me%zno-1
!                 if(Me%poridx(i,1,k)==1 .and. Me%poridx(i,1,k+1)==1)then
!                   Me%wdepth(i,1) = Me%znod(i,1,k)
!                   write(102,502)
!      1                 i,j,Me%xpos(i,1,k),Me%ypos(i,1,k),Me%wdepth(i,1)
!                 exit
!                 endif
!               enddo
              enddo
            enddo
          enddo
            write(099,499) ist,Me%xno
```

```
            write(099,499) ist,Me%yno+2
            write(099,499) ist,Me%zno
            write(099,501) Me%dx,Me%dy,Me%dz
!           do i = 1,Me%xno
            write(099,501)(Me%xpos(i,1,1),i=1,Me%xno)
!           enddo
!           do j = 1,Me%yno
!            write(099,501)(Me%ypos(1,j,1),j=1,Me%yno)
            write(099,*)'0.0   0.2   0.4'                    ! for sediment particle model
!           enddo
!           do i = 1,Me%zno
            write(099,501)(Me%zpos(1,1,k),k=1,Me%zno)
!           enddo
  499   format(10(i4,1x))
  500   format(f12.4)
  501   format(10(f8.3,1x))
  502   format(2i6,10(f8.3,1x))
        end subroutine readwriteresults
        subroutine readalphawater(m)
        !-------------------------------------------------------------
        integer               :: i,j,k,m,stat,skip
        integer               :: lstr,mstr,rstr
        character(len=1)   :: idx
        character(len=20) :: lidx,midx,ridx
        !-------------------------------------------------------------
        write(*,'(a,f10.2)') "read alphawater, time = ",Me%foltime(m)
          open(10,file=trim(Me%fln1(m)),status='old')

          skip = 0

          do
            read(10,'(a)') idx
              skip = skip + 1
              if(trim(idx) == trim(Me%idx1))then
                exit
              endif
          enddo

          rewind(10)

          do k = 1,skip
            read(10,*)
          enddo
```

```
      do k = 1,Me%zno
        do j = 1,Me%yno
          do i = 1,Me%xno
            read(10,*) Me%alphaw(i,j,k)
          enddo
        enddo
      enddo
      close(10)
    end subroutine readalphawater
    subroutine readvelocity(m)
  !-------------------------------------------------------------
    integer                :: i,j,k,m,stat,skip
    integer                :: lstr,mstr,rstr
    character(len=1)   :: idx
    character(len=20) :: lidx,midx,ridx
    !-------------------------------------------------------------
    write(*,*) "reading velocity"
    open(10,file=trim(Me%fln3(m)),status='old')

    skip = 0

    do
      read(10,'(a)') idx
        skip = skip + 1
        if(trim(idx) == trim(Me%idx1))then
          exit
        endif
    enddo
    rewind(10)
    do k = 1,skip
      read(10,*)
    enddo
    do k = 1,Me%zno
      do j = 1,Me%yno
        do i = 1,Me%xno
          read(10,*)lidx,midx,ridx
          lstr = len_trim(lidx)
          mstr = len_trim(midx)
          rstr = len_trim(ridx)
          read(lidx(2:lstr),   *) Me%velu(i,j,k)
          read(midx(1:mstr),   *) Me%velv(i,j,k)
          read(ridx(1:rstr-1),*) Me%velw(i,j,k)
        enddo
```

```
      enddo
    enddo
    close(10)
     end subroutine readvelocity
    end program convertopenfoam
```

그림 48. OpenFOAM 후처리 프로그램

아. ParaView

- OpenFOAM 수행 및 예측결과 가시화의 편리성 도모를 위해 “The ParaView Tutorial(version 5.6)”을 기반으로 ParaView의 기본사용 방법을 정리하여 요약·제시한다.

(1) ParaView 개요

① 개요

- ParaView는 2차원 또는 3차원 자료를 가시화하는 공개 프로그램이다.
- ParaView에 사용되는 자료의 크기는 응용 프로그램이 실행되는 아키텍처(architecture)에 따라 광범위하게 적용할 수 있다.
- ParaView가 지원하는 플랫폼은 단일 프로세서 워크스테이션에서 다중 프로세서 메모리 분산 슈퍼컴퓨터 또는 워크스테이션 클러스터까지 다양하며 ParaView는 병렬시스템을 사용하여 매우 큰 데이터 세트를 병렬로 처리할 수 있다.
- ParaView의 디자인은 다른 과학 가시화 솔루션과 차별화되는 다음과 같은 많은 개념적 기능을 포함하고 있다.

  · 오픈 소스, 확장 가능한 다중 플랫폼 가시화 응용 프로그램

  · 대규모 자료 처리를 위한 분산 계산모델 지원

  · 개방적이고 유연하며 직관적인 사용자 인터페이스

  · 개방형 표준에 기초한 확장 가능한 모듈식 아키텍처

  · 유연한 BSD 3-clause 라이선스

  · 상업적 유지보수와 지원

- ParaView는 폭넓은 응용 프로그램을 가지는 범용도구로 작은 데이터에서 큰 데이터까지 확장할 수 있고 특정 과학 분야의 특별한 알고리즘뿐만 아니라 범용

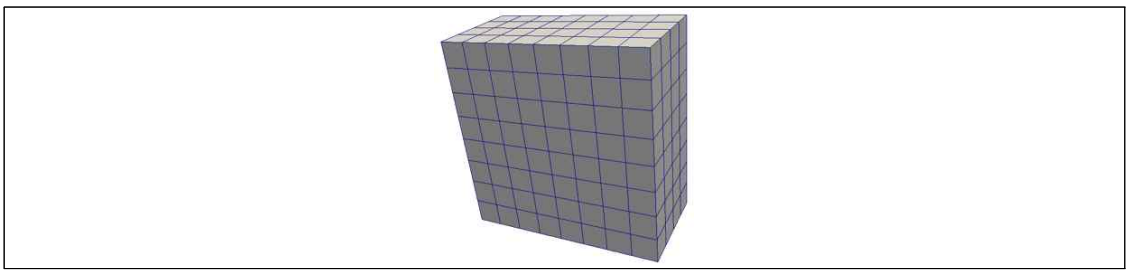
목적의 많은 가시화 알고리즘을 제공하며, 더욱이 ParaView 시스템은 특정 목적에 맞는 맞춤 가시화 알고리즘으로 확장할 수 있다.

② 가시화 기초

- 간단하게 말해서, 가시화는 원시데이터를 가지고 와서 사용자가 보고 이해할 수 있는 형태로 변환하는 것. 이를 통해 사용자는 데이터에 대한 이해도를 높일 수 있다.
- 과학적인 가시화는 특히 2차원 또는 3차원에 정의된 데이터의 형태와 관련이 깊고, 계산 격자로부터 도출된 데이터는 이러한 형태의 분석에 특히 적합하다.
- 사용자가 자료를 가시화하기 위해서는 읽기, 필터링, 렌더링의 세 단계가 필요하다. 먼저 사용자는 ParaView로 데이터를 읽어야 하고 다음으로 필터를 선택하여 데이터로부터 어떤 형태를 생성, 추출, 파생시켜야 하며 마지막으로 볼 수 있는 이미지가 자료로부터 렌더링 된다.
- ParaView는 우선적으로 공간적으로 데이터를 처리할 수 있도록 설계되었다. 따라서 ParaView에서 사용되는 기본 데이터는 격자(mesh) 형태이다.

○ Uniform Rectilinear(Image Data)

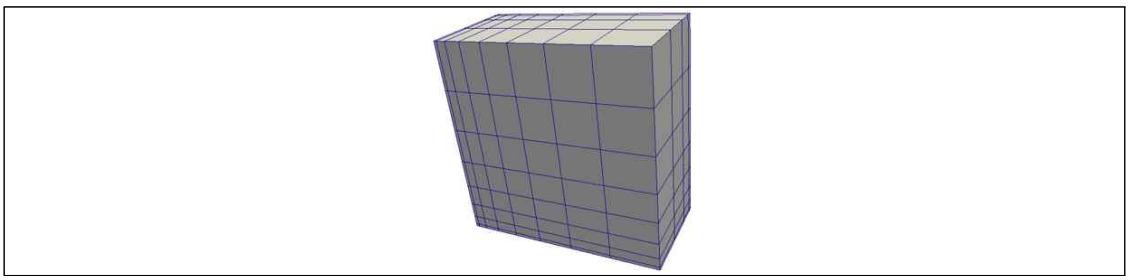
- 1, 2, 3차원 배열의 균일직교 격자로 각 점은 서로 직교하며 각 방향을 따라 일정한 크기를 가진다.



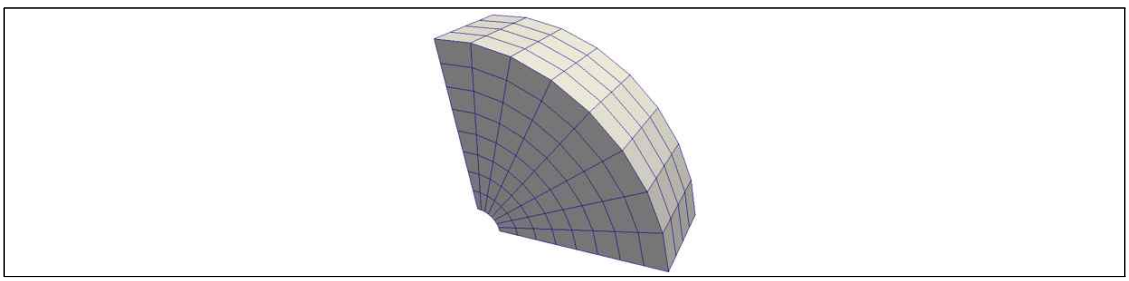
○ Non-uniform Rectilinear(Rectilinear Data)

- 1, 2, 3차원 배열의 균일직교 격자로 각 점은 서로 직교하며 각 방향을 따라 변화하는 크기를 가진다.

○ Curvilinear(Structured Data)

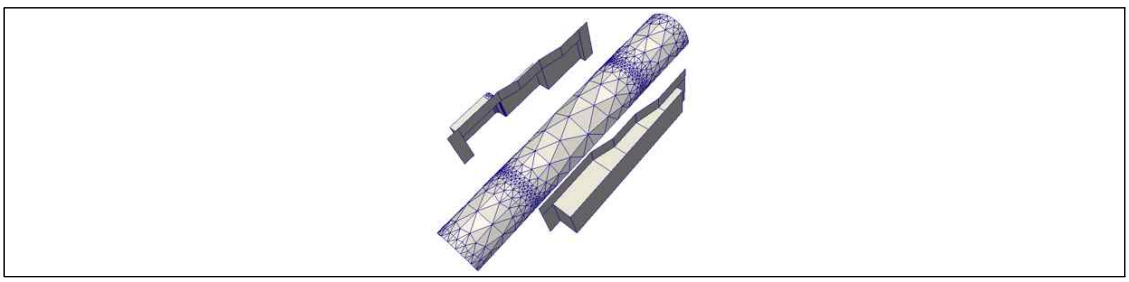


- 곡선 직교 격자는 직교 격자와 같은 topology를 가지지만, 각 점은 임의의 좌표에 배치될 수 있음(단 격자가 중첩되거나 자신을 가로지르는 cell은 허용되지 않음). 곡선 직교 격자는 직교 격자보다 더 적은 메모리 공간과 implicit topology를 제공하고 또한 공간적으로 격자의 형태가 훨씬 더 다양하게 변화할 수 있도록 해준다.



○ Polygonal(Poly Data)

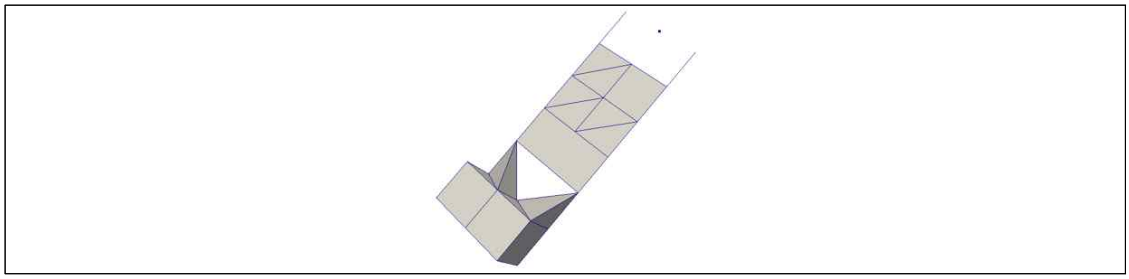
- 다각형 데이터 세트는 점, 선, 2차원 다각형으로 구성. Cell 간의 연결은 임의적이거나 존재하지 않을 수 있음. 다각형 데이터는 기본 렌더링 primitive를 표현하는 것으로 어떤 자료든지 렌더링을 하기 전에는 반드시 다각형 데이터로 변환해야 하며, ParaView는 이를 자동으로 변환한다.



○ Unstructured Grid

- 비구조 데이터 세트는 점, 선, 2차원 다각형, 3차원 사면체 및 비선형 cell로 구성.

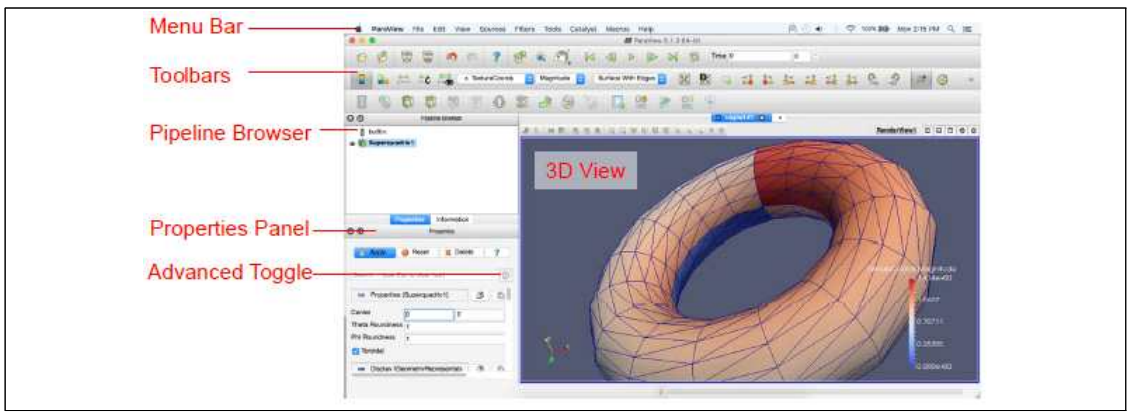
이들은 3차원 사면체와 비선형 cell을 표현설 수 있다는 점을 제외하면 다각형 데이터와 유사하며, 직접 렌더링을 할 수는 없다.

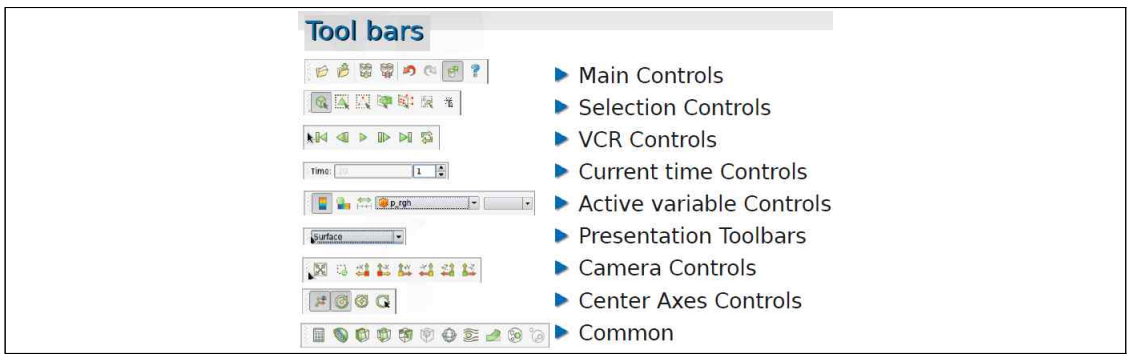


(2) ParaView 기본 사용법

① 사용자 Interface

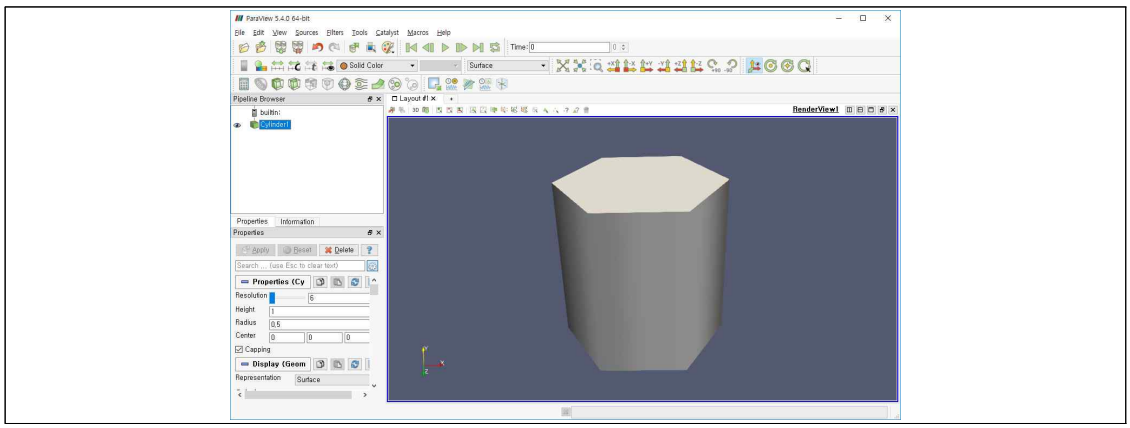
- 다음의 그림은 ParaView를 처음 실행할 때 제공되는 기본 배치로 GUI의 구성 요소는 다음과 같다.





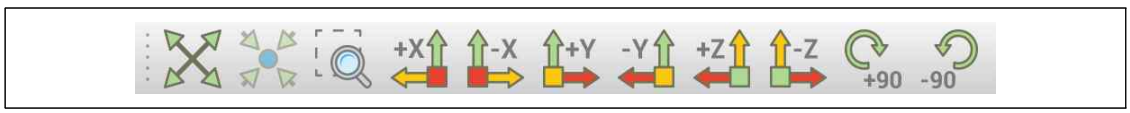
② Source

- Menu Bar의 Sources에서 Cylinder를 선택하고 Apply를 적용하면 다음과 같이 Cylinder를 표현할 수 있다.



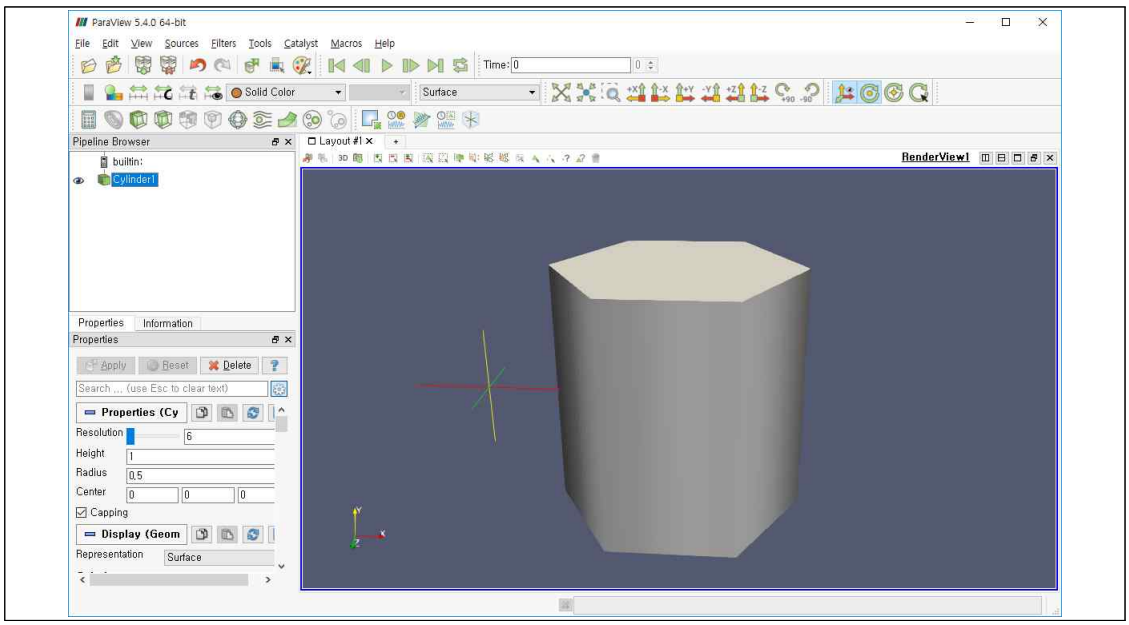
③ Basic 3D Interaction

- 3D View 내 cylinder의 이동, 회전, 확대 및 축소는 마우스의 좌측, 가운데, 우측 버튼을 클릭하여 적용한다. 또한, 다음의 메뉴를 이용하여 확대, 회전, 좌표축에 따른 그림 등을 표현할 수 있다.



- 다음 툴바는 회전 중심과 좌표 orientation을 표시한다.

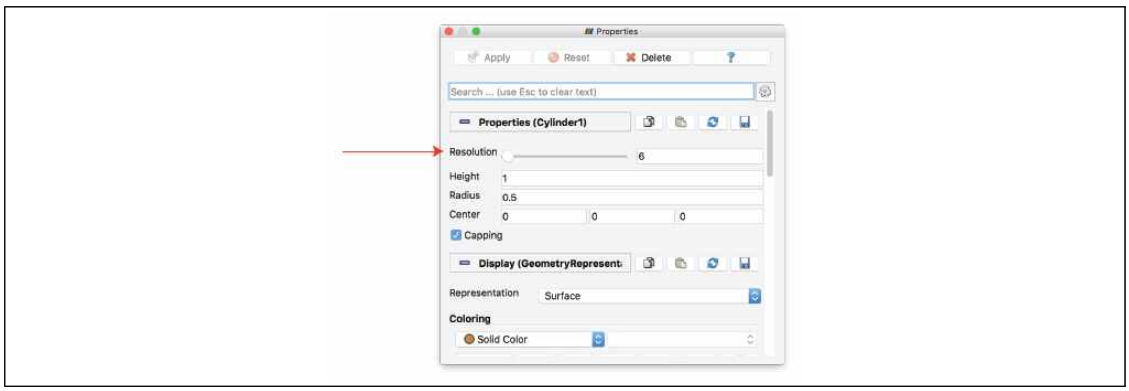
-  는 회전 중심을 정의해주고  는 회전 중심을 물체의 중심으로 원상복귀하게 해주며,  는 회전 중심을 표시하거나 사라지게 해주고  는 좌표축의 orientation을 표시하거나 제거할 때 사용한다.

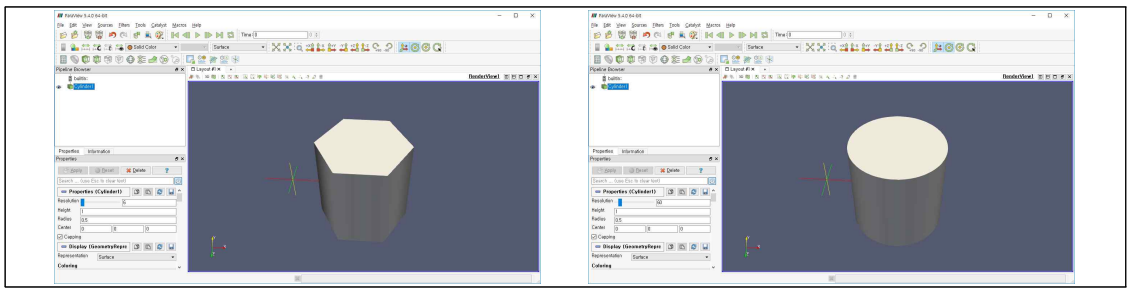


④ Modifying Visualization Parameters

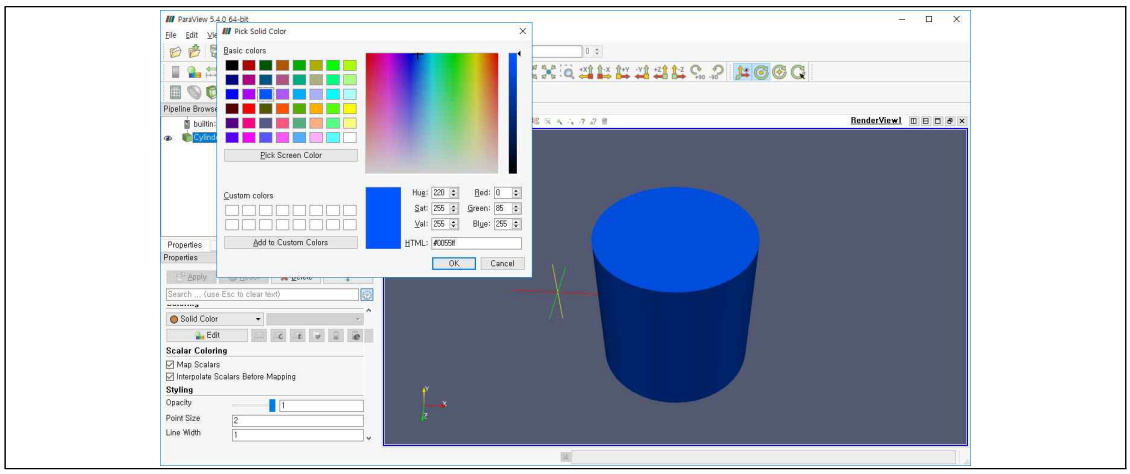
- 앞선 그림에서 cylinder의 형태는 원형이 아니라 6각형의 형태이며, 더욱 cylinder의 형태와 같게 만들기 위해서는 Properties 패널의 Resolution을 증가시켜서 변형할 수 있다.

- 이러한 설정들은 패널 내의 저장 버튼( )으로 저장할 수 있음. 한 object에서 다른 object로의 설정은 복사( )와 붙여넣기( ) 버튼으로 실행한다.

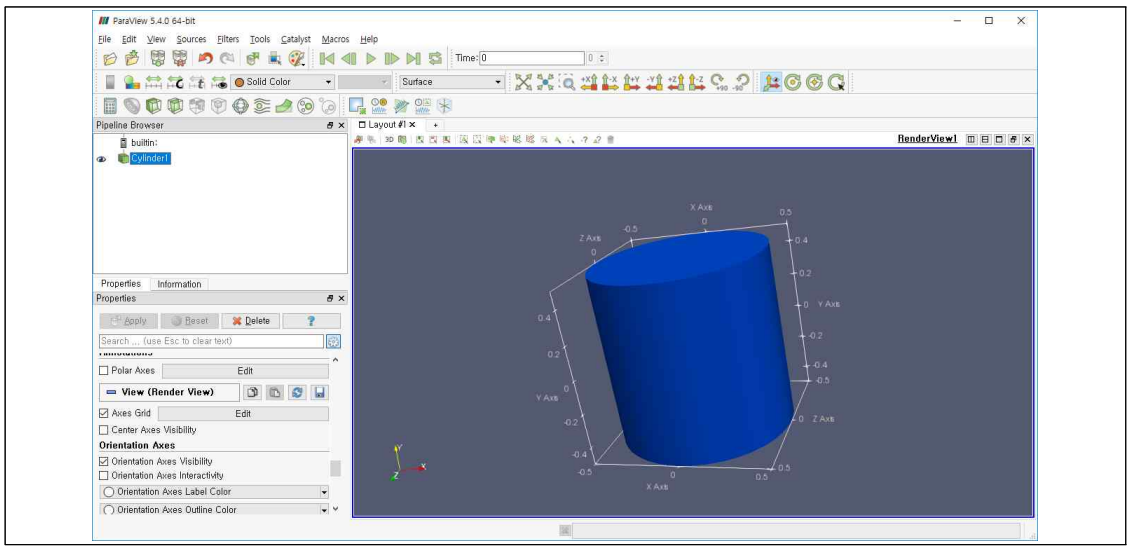




- properties 패널의 Display properties-Edit-Coloring을 선택하여 물체의 색을 바꿀 수 있다.



- properties 패널의 Display properties 하단의 View properties로 Axes Grid를 설정할 수 있으며, 추가적인 properties는  클릭하고 설정할 수 있다.

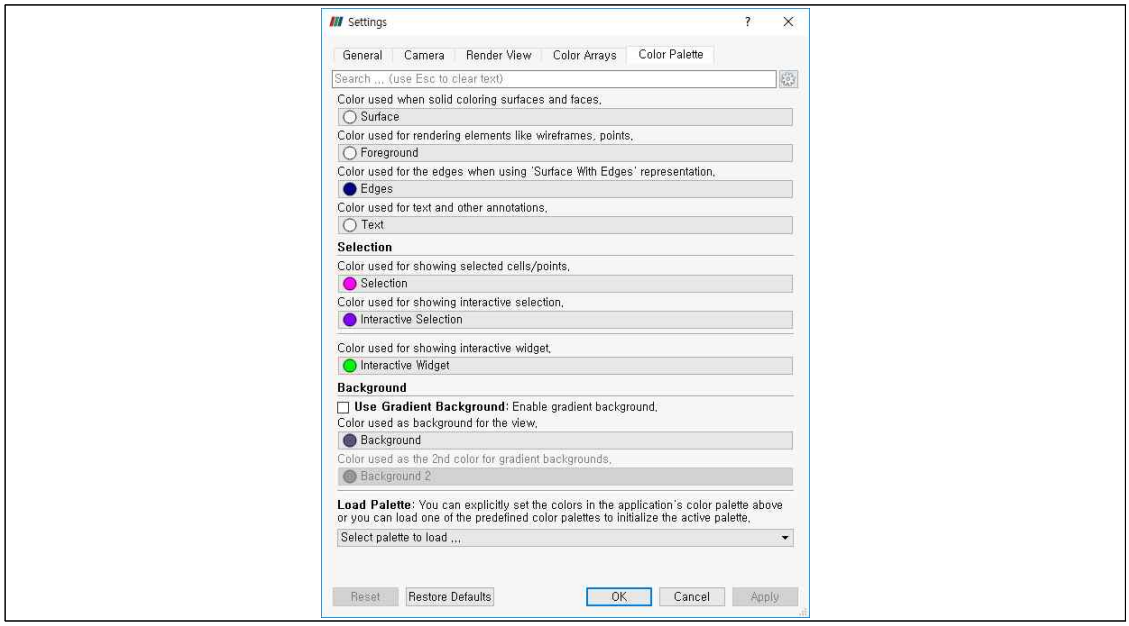


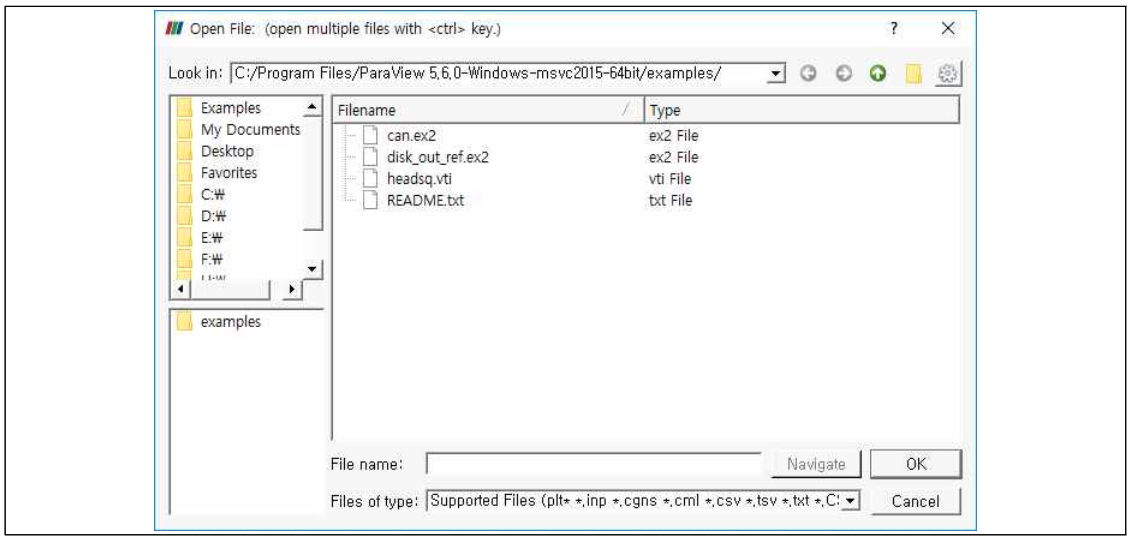
- 상단메뉴의  (auto apply button)을 클릭하여 둔 경우, 변경된 설정은 Apply ()을 클릭하지 않아도 자동으로 적용된다.

- color palette의 변경은 상단 툴바의  버튼을 클릭하거나, 메뉴바의 Edit-Setting-Color Palette를 선택하여 변경할 수 있다.

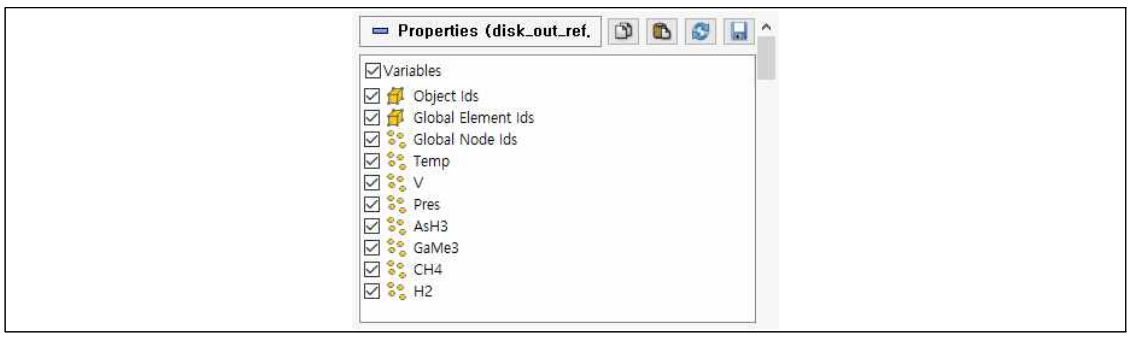
⑤ Loading Data

- 상단메뉴로부터 다음과 같이 파일을 선택하고 불러올 수 있다.

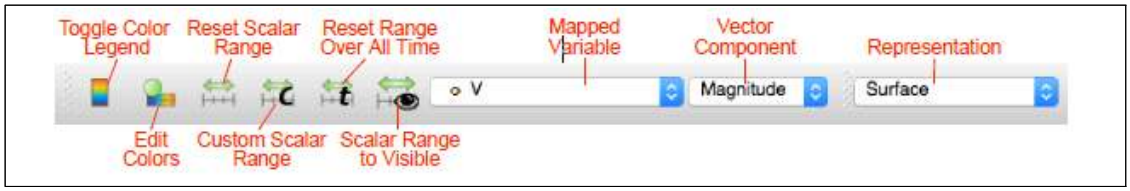




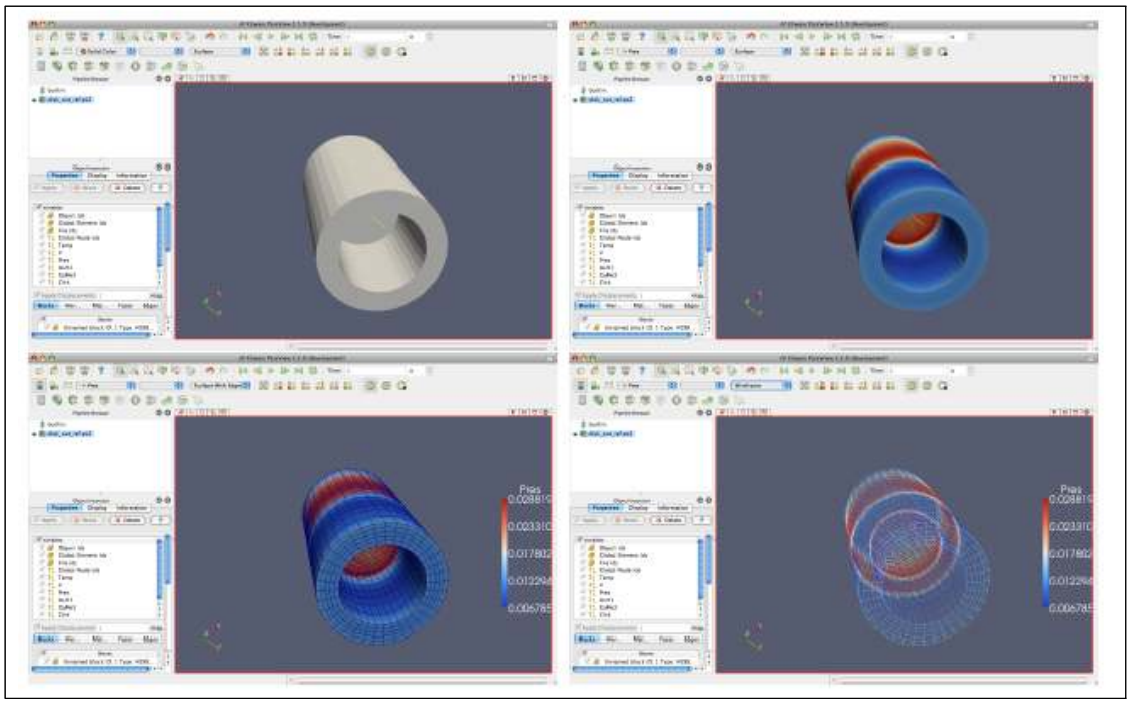
- properties 패널의 Variable을 클릭하여 주고 Apply를 클릭한다.



- 데이터의 필터링에 앞서, 데이터를 표현하는 대부분의 일반적인 파라미터는 다음의 툴바에 있다(이들은 properties 패널 내에도 존재함).



- 앞선 그림의 Mapped Variable(Pres 선택)과 Representation(Surface, Surface with Edge, Wireframe)을 선택하여 물체의 형태와 격자 형태 등을 다음처럼 표현할 수 있다.



⑥ Filters

- ParaView 내의 필터를 적용하여 데이터에 대한 더 많은 정보를 파악할 수 있음. 필터는 데이터를 생성, 추출, 특정 양상을 파생시키는 ParaView의 기능이다. 많이 사용되는 필터는 다음 그림과 같이 툴바에 표시되며, 메뉴에서 Filters를 통해 다른 필터를 선택할 수 있다.

Calculator – point 또는 cell 상의 사용자 정의 계산식 수행. 사용자가 원하는 field 값을 수식을 이용하여 생성. 예를 들어 압력을 무차원화 시킨 압력계수를 계산하고 이를 도시할 수 있다.

Contour – 사용자 정의에 따라 스칼라 필드를 점, 곡선, 표면(surface)으로 추출. 여기서 표면은 종종 isosurface라고 칭해짐. 사용자가 원하는 field 값을 지정된 범위에 대해 도시한다.

Clip – geometry 단면 절단, 사용자 정의 plane의 한 면의 geometry를 제거한다. 즉, 계산 도메인 영역을 분할 하는 기능(plane, box, sphere, scalar 옵션)을 가지고 있다.

Slice – geometry를 plane 형태로 절단, plane이 위치한 geometry의 면만 남김 (plane, box, sphere 옵션). Slice로 plane을 절단하여 surface를 만든 다음 contour를 이용하여 contour line 형태의 그림으로 도시 할 수 있다.

Threshold – 스칼라 필드의 지정된 범위 내에 있는 cell을 추출. 도메인 영역을 field 값의 최대, 최솟값을 기준으로 분할한다.

Extract Subset – 관심항목의 volume 또는 sampling rate의 정의에 따라 grid의 subset을 추출한다.

Glyph – mesh의 각 점에 단순 형태의 glyph를 위치시킴. glyph는 vector로 방향이 정의되고 크기는 vector 또는 scalar로 scaled 됨. 속도벡터의 크기나 개수, 모양, Mode 등은 Properties에서 수정 가능하다.

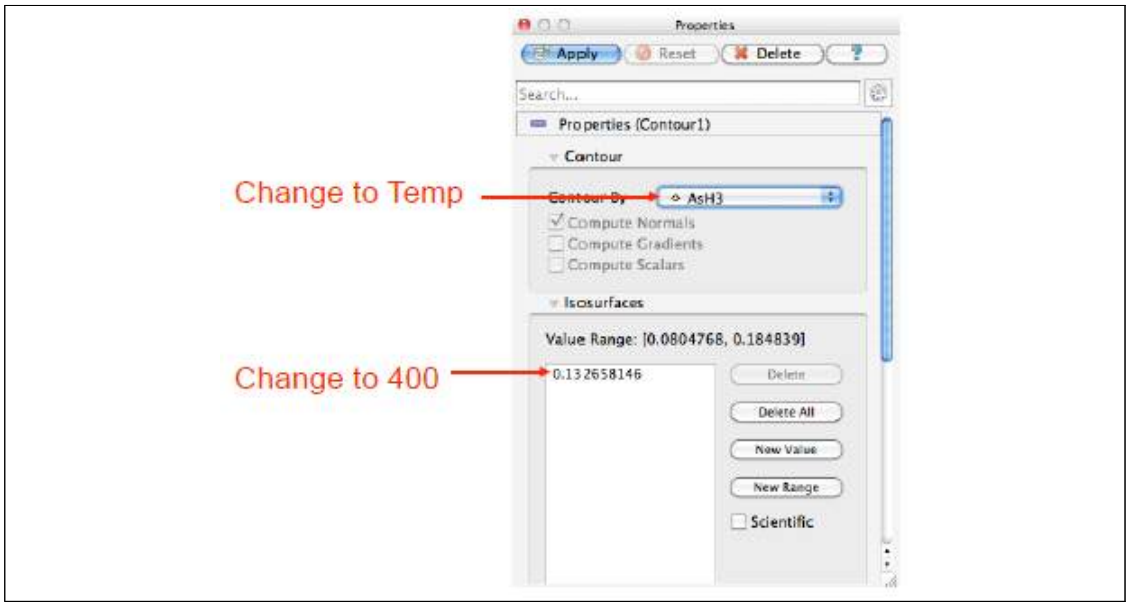
Stream Tracer – Streamline을 그릴 수 있음(point로 vector field를 seed 한 다음, 이들 seed point를 vector field로 trace 함). Properties 내의 seed type에서 point나 line을 선택할 수 있다.

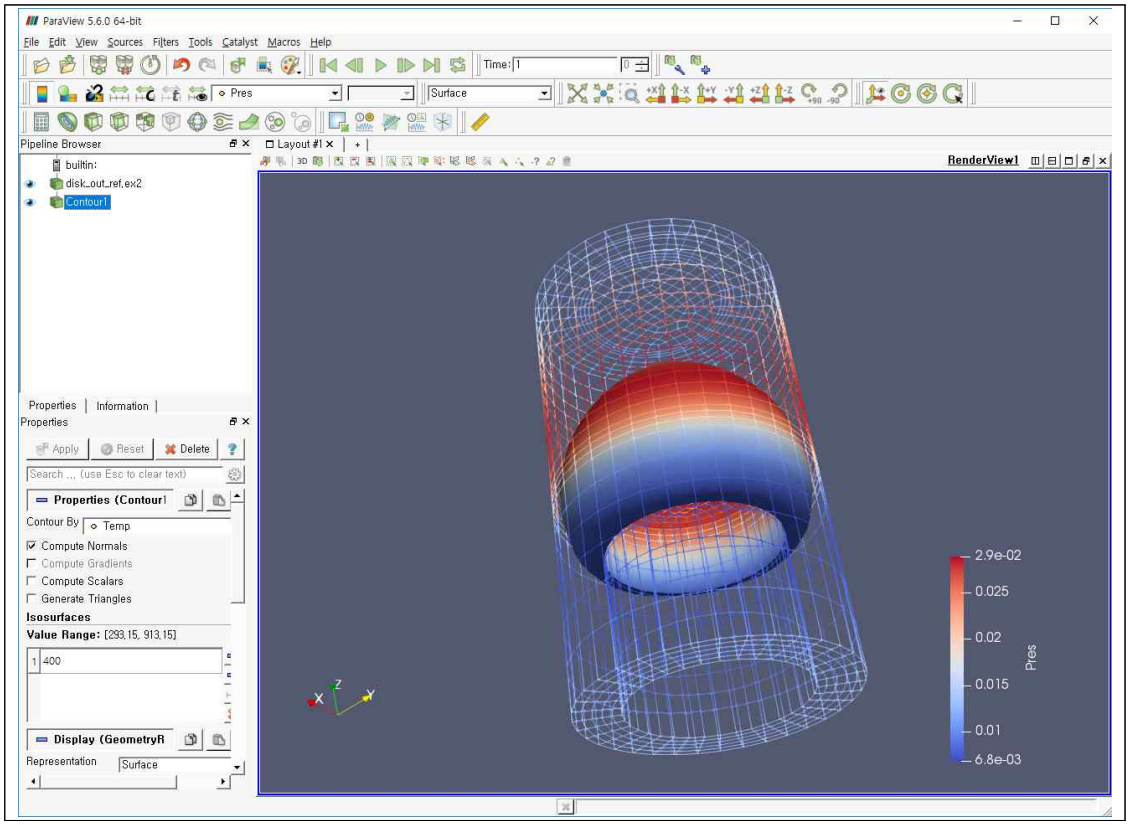
 Wrap (vector)- mesh의 각 점을 주어진 vector field로 대체한다.

 Group Datasets - 여러 pipeline objects의 출력값을 단일 다중 블록 데이터 세트로 결합한다.

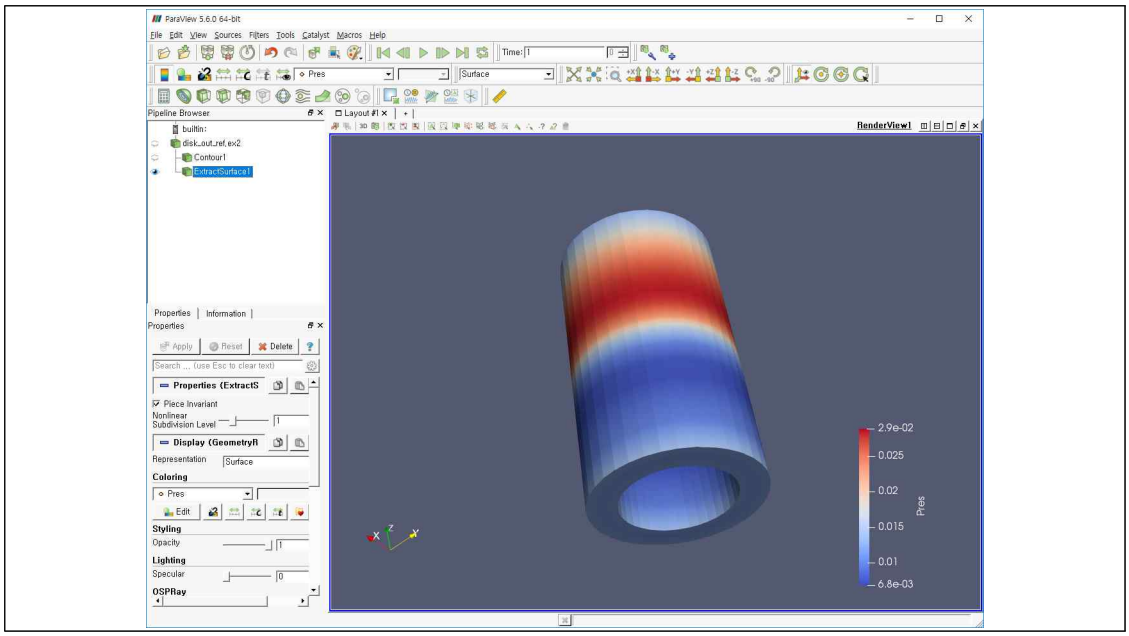
 Extract Level - 다중 블록 데이터 세트로부터 하나 또는 그 이상의 item을 추출한다.

- 추가 필터는 Menu 바의 Filters에서 찾아볼 수 있으며, 키보드상의 Ctrl+space bar -> space bar를 클릭하여 필터를 찾고 선택할 수 있다.
- Contour Filter 적용 :  를 클릭하고 Contour By "TEMP" 선택 후 Isosurfaces의 Value Range에서 값을 400을 선택하고 Apply 하면 다음의 그림이 생성된다.

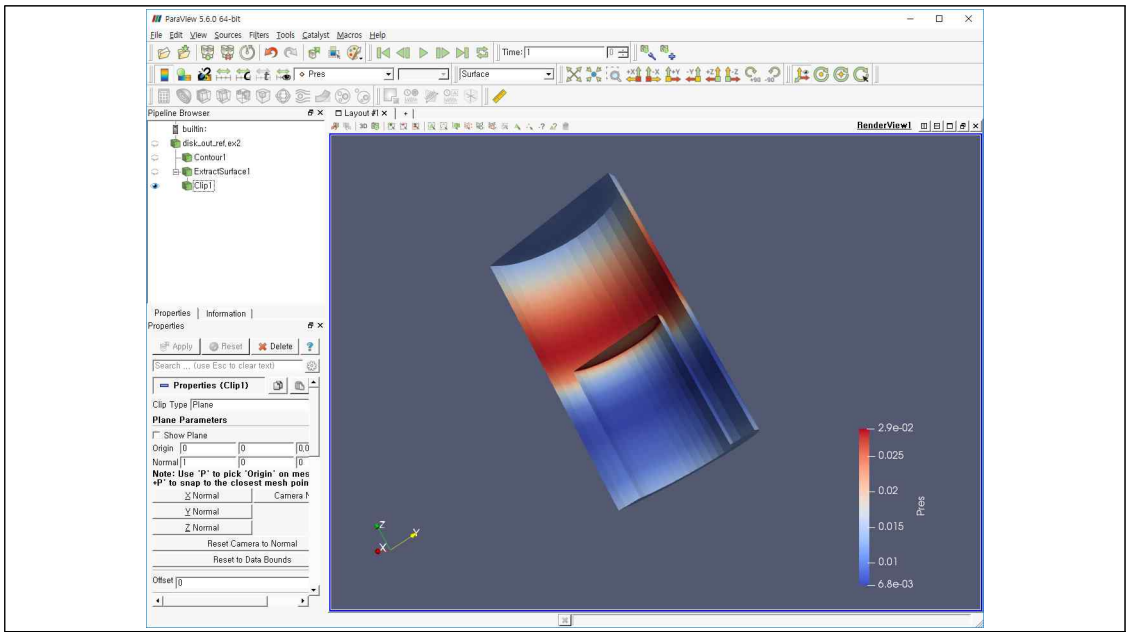




- Visualization Pipeline : Pipeline Browser에서 disk_out_ref.ex2를 선택하고 Filters-Alphabetical-Extract Surface를 선택한다.

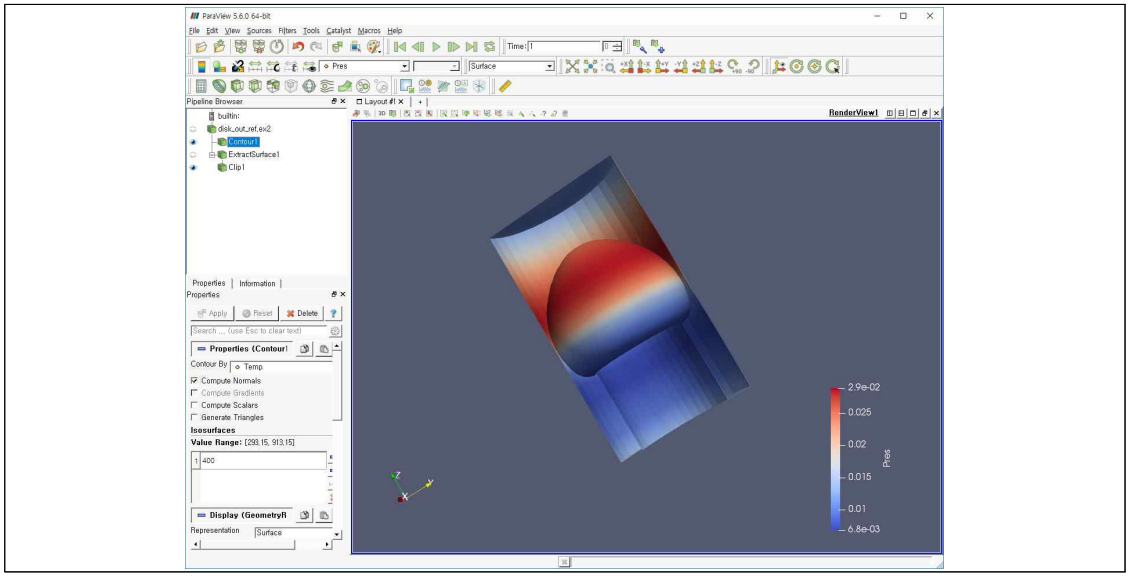


- Pipeline Browser에서 ExtraSurface1을 선택한 상태에서 clip( ) 필터를 선택하고 properties 창에서 show plane(Show Plane)을 해제하면 다음과 같이 그림을 얻을 수 있다.



- 작성된 Contour를 선택하면 다음과 같이 그림을 그릴 수 있다.

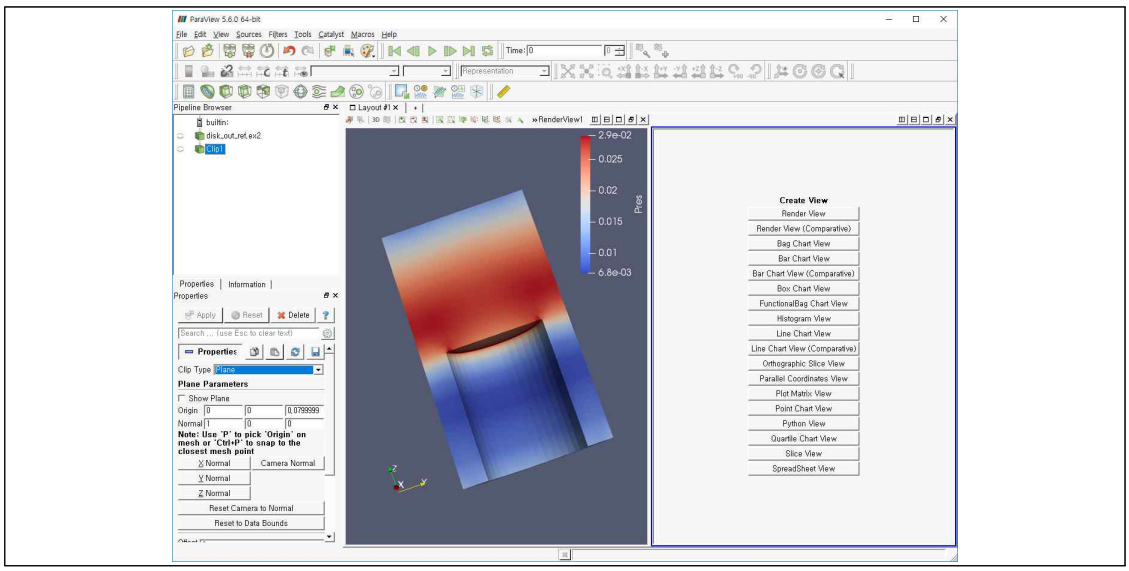
⑦ Multiview

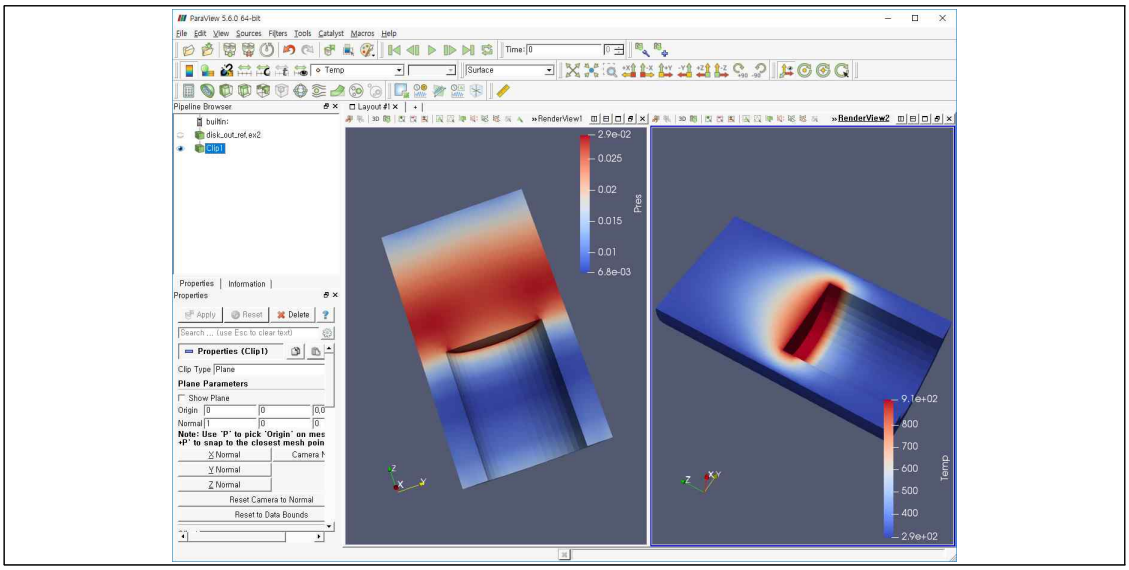


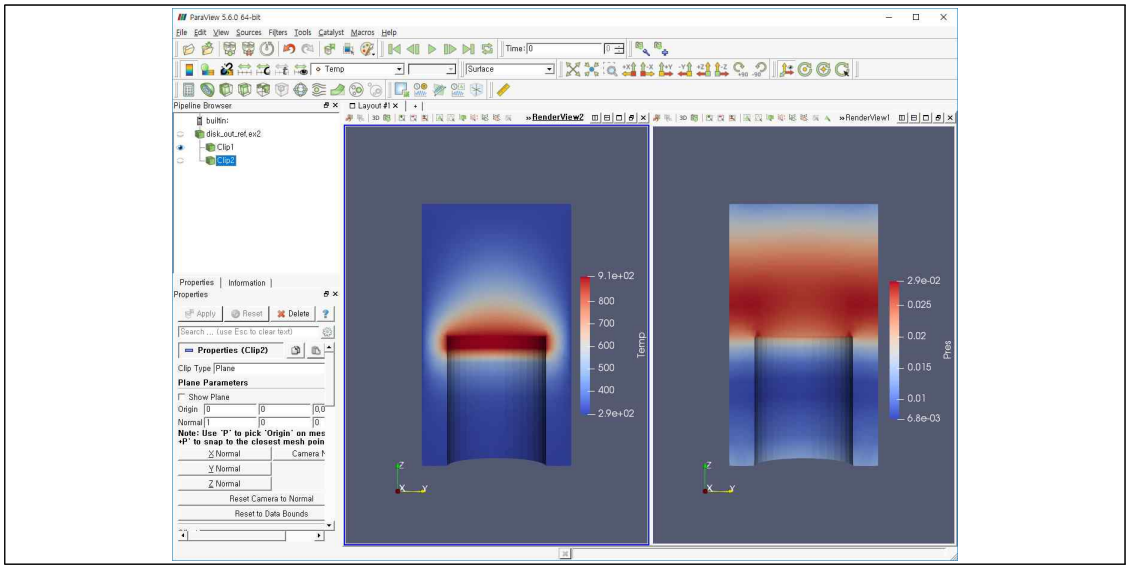
- ParaView 내의 기존에 불러온 파일들을 제거하고 reset 하기 위해서는 Edit-Reset Session(Ctrl+R)을 선택하거나, 툴바에서  를 선택한다.
- 앞서와 같이 disk_out_ref.ex2 파일을 불러오고, Apply를 선택한다.
- 툴바에서 Pres를 선택하고 그림 형태는 Surface를 선택한다.
- 오른쪽 위의  툴바에서  아이콘을 클릭한다.

- 우측 창이 파란색인 상태에서 Pipeline Browser에서 Clip1의 eyeball()을 클릭하고 Solid Color를 Temp로 선택하면 다음과 같이 그림이 그려진다.

- 그림 상단 툴바()를 마우스를 누른 상태에서 상대편 툴바에 옮기는 방법으로 그림의 좌우 위치를 변경할 수 있다.


ParaView 5.6.0 64-bit
File Edit View Sources Filters Tools Catalyst Macros Help
Pipeline Browser
builtin:
disk_out_ref.ex2
Clip1
Properties | Information
Properties
Apply Reset Delete
Properties
Clip Type Plane
Plane Parameters
Show Plane
Origin 0 0 0.0799999
Normal 1 0 0
Note: Use 'P' to pick 'Origin' on mesh or 'Ctrl+P' to snap to the closest mesh point
X Normal Camera Normal
Y Normal
Z Normal
Reset Camera to Normal
Reset to Data Bounds
RenderView1
2.9e-02
0.025
0.02
0.015
0.01
6.8e-03
Press
Create View
Render View
Render View (Comparative)
Bag Chart View
Bar Chart View
Bar Chart View (Comparative)
Box Chart View
FunctionalBag Chart View
Histogram View
Line Chart View
Line Chart View (Comparative)
Orthographic Slice View
Parallel Coordinates View
Plot Matrix View
Point Chart View
Python View
Quartile Chart View
Slice View
SpreadSheet View

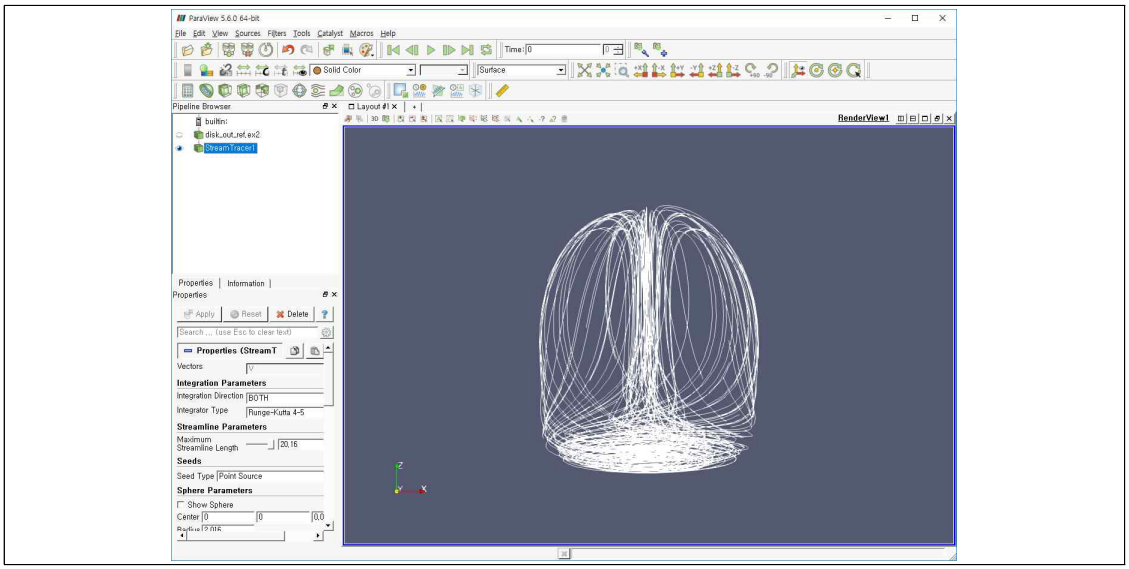

ParaView 5.6.0 64-bit
File Edit View Sources Filters Tools Catalyst Macros Help
Pipeline Browser
builtin:
disk_out_ref.ex2
Clip1
Properties | Information
Properties
Apply Reset Delete
Properties (Clip1)
Clip Type Plane
Plane Parameters
Show Plane
Origin 0 0 0.0
Normal 1 0 0
Note: Use 'P' to pick 'Origin' on mes
'P' to snap to the closest mesh poin
X Normal
Y Normal
Z Normal
Reset Camera to Normal
Reset to Data Bounds
RenderView1
RenderView2
2.9e-02
0.025
0.02
0.015
0.01
6.8e-03
Press
9.1e+02
800
700
600
500
400
2.9e+02
Temp

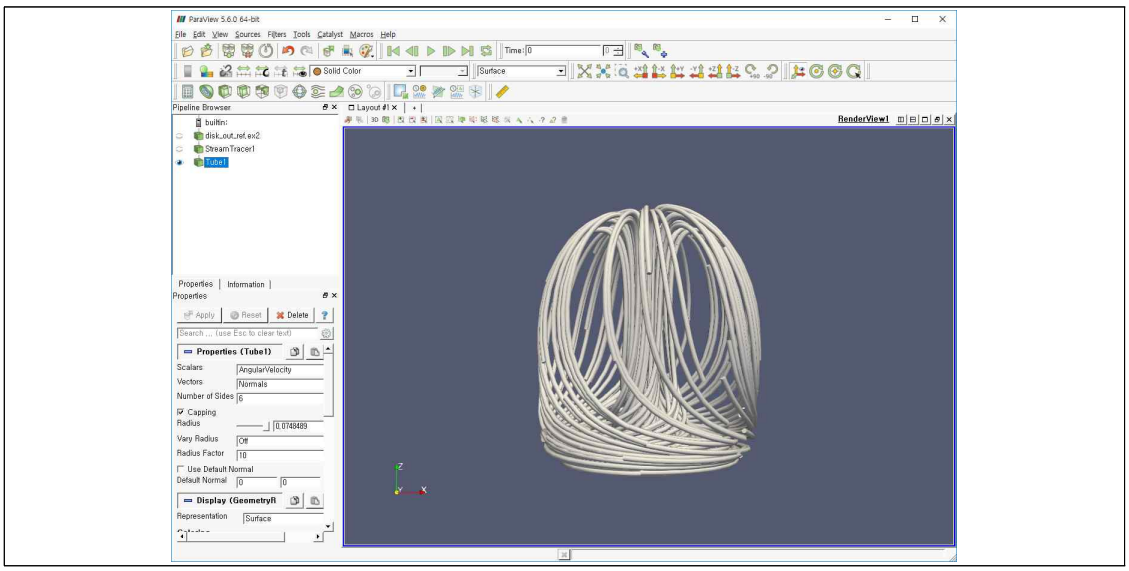


⑧ Vector Visualization

- ParaView 툴바의 Reset Session 실행 후 disk_out_ref.ex2 파일을 새로 불러옴. 이후  를 선택하고 Properties 창에서 Seed Type을 Point Source, Show Sphere를 해제하고 Apply를 클릭하면 다음과 같은 그림을 얻을 수 있다.

- Ctrl+space bar - space bar를 실행하고 Tube 필터를 선택하고 Apply를 적용하면 다음과 같이 그림이 얻어진다.

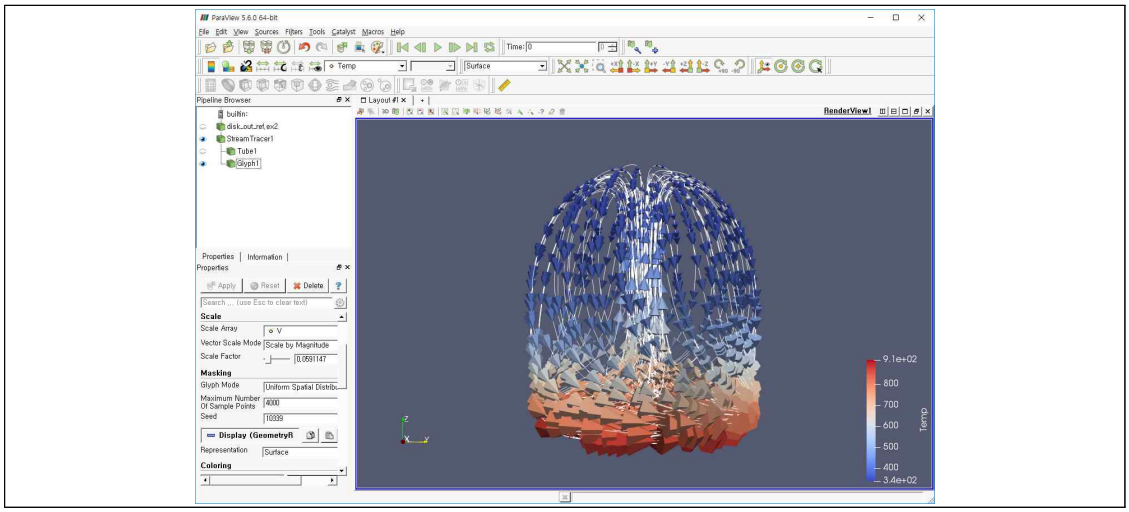




- Pipeline Browser에서 StreamTracer1을 선택하고 glyph filter( )를 클릭하고 properties 패널에서 Glyph Type을 Cone, Orientation Array를 V, Scale Array를 V, 그리고 Glyph Data Range 내의 reset( )을 클릭하고 Apply 하면 다음의 그림이 생성된다.

⑨ Plotting

- ParaView는 plotting 기능을 통해 데이터를 정향 분석하는 기능을 제공한다. Plot은 보통 filter로 생성되고 모든 plotting filter는 filter의 하위메뉴에 위치한다.



- 가장 일반적인 데이터 분석 필터를 포함하는 데이터 분석 툴바가 존재하며 (), 그들 중 일부는 plot을 생성하는 데 이용된다.

 Extract Selection - 자체 object에서 선택된 데이터 추출한다.

 Plot Global Variables Over Time - 단일 point나 cell이 아닌 전체 데이터 세트에 적용되는 “global” 변수의 정보를 캡처. 이 필터는 시간에 따른 전역 정보를 plot 한다.

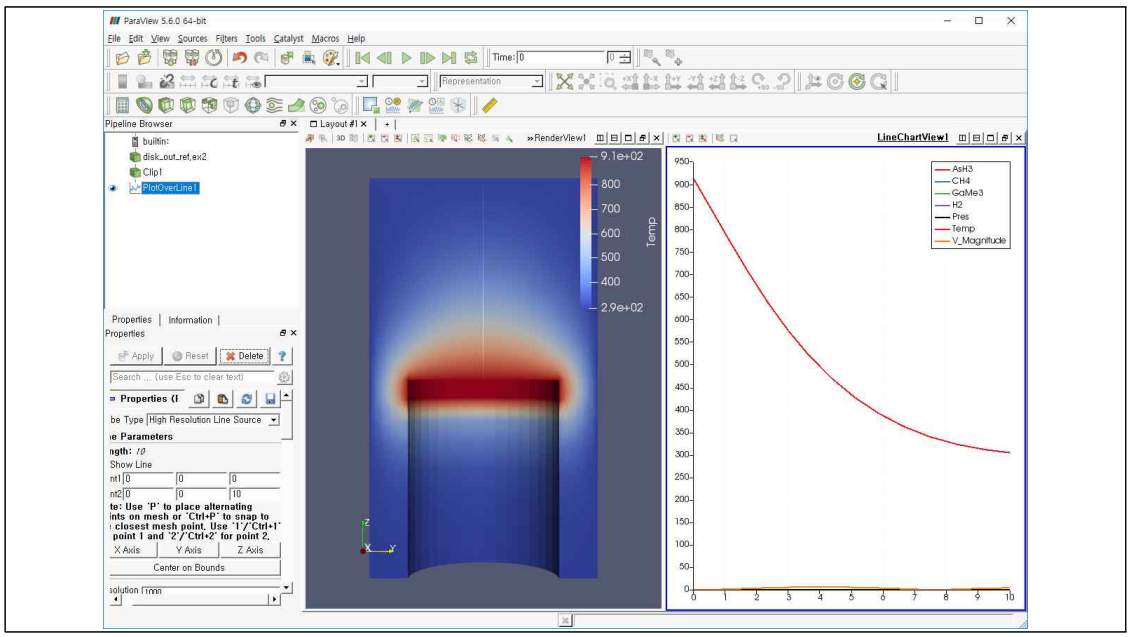
 Plot Over Line - 3차원 공간상에 정의된 선상의 필드 정보를 plot 한다.

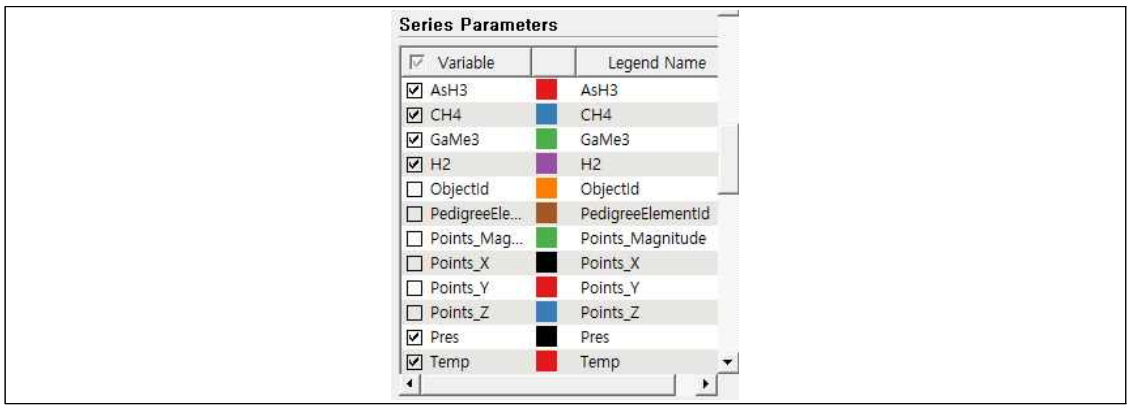
 Plot Selection Over Time - 선택된 점 또는 cell의 필드를 가지고 시간에 따른 변화를 plot 한다.

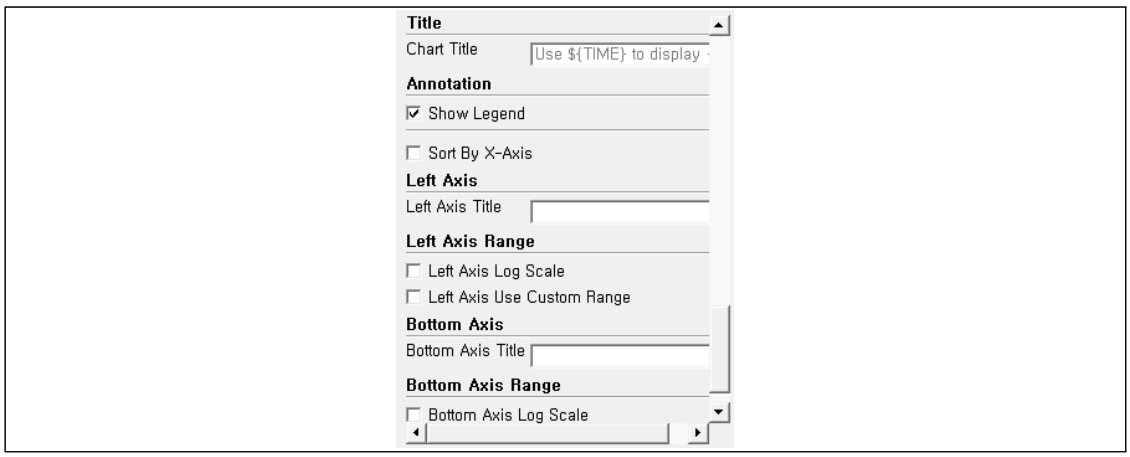
 Probe - Space 내 articular location에서 필드 값을 제공한다.

○ Plot Over a Line in Space

- 앞에서와 같이 disk_out_ref.ex2파일을 열고, Clip filter를 적용한다(Show Plane 해제). 이후 plot over line filter()를 클릭한다.
- Properties 패널에서 point1 (0,0,0), point2 (0.0.10)을 선택하고 Apply를 적용한다.
- Properties 패널의 Series Parameters 창을 통해 우측 plot의 해당 필드에 대한 선택 및 색상 변경 등을 선택할 수 있다.
- Properties 패널의 Series Parameters 창을 하단 메뉴에서 label, legend, axes range 등을 입력·변경할 수 있다.
- 그림의 저장은 File->  Save Screenshot으로 capture 할 수 있고, File->Export Scene을 선택하여 pdf 등의 형태로 저장할 수 있다.

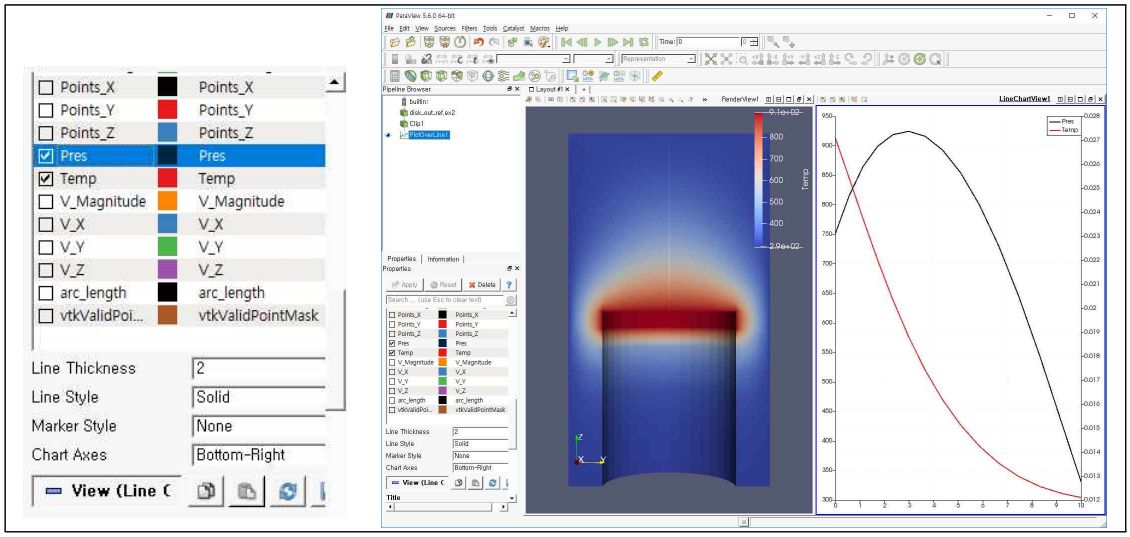




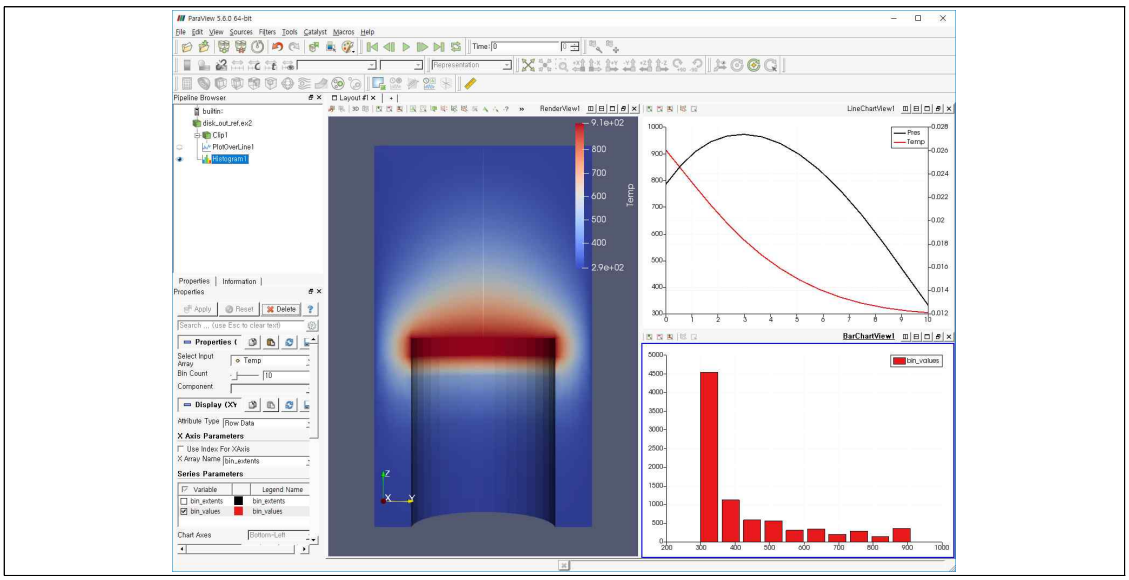


- ParaView 내에서 plot 상에 나타나는 필드의 변경(split, delete, resize, swap 등)이 가능하다.

- Display 창에서 서로 다른 유닛을 가지는 Temp와 Press를 선택하고, 이후 마우스로 Pres를 선택한 상태에서 Chart Axis를 Bottom - Right로 선택하면 좌측에 Temp, 우측에 Press의 단위로 하는 그림이 생성된다.

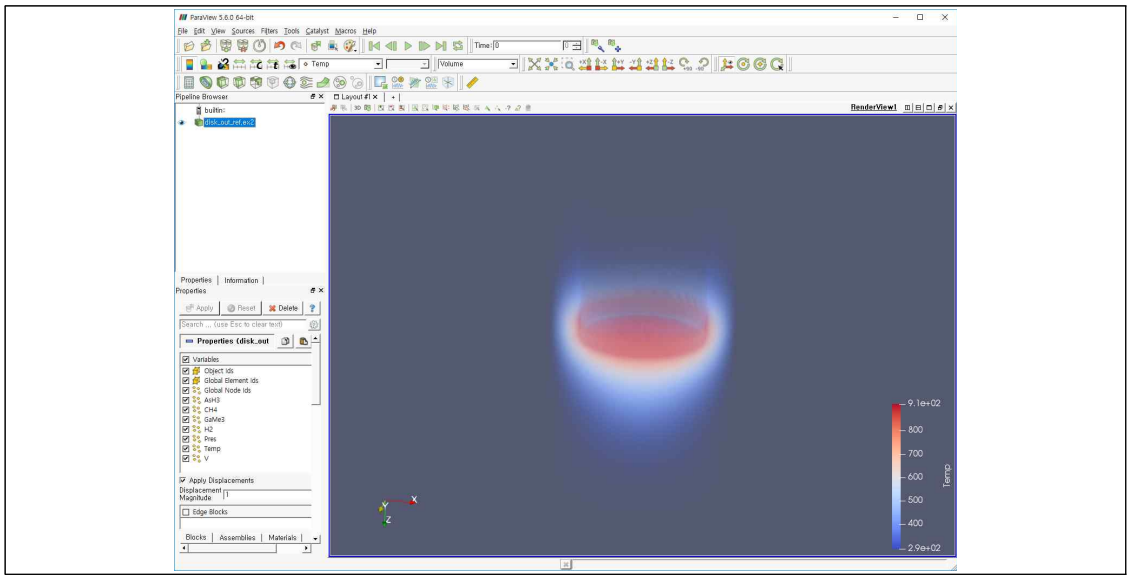


- Histogram 형태의 그림은 Histogram( ) 필터를 이용하여 그릴 수 있다.
- 먼저 pipeline browser에서 disk_out_ref.ex2를 선택하고 Filters->Data Analysis ->Histogram을 선택하면 Temp에 대해 다음과 같은 그림을 얻을 수 있다.

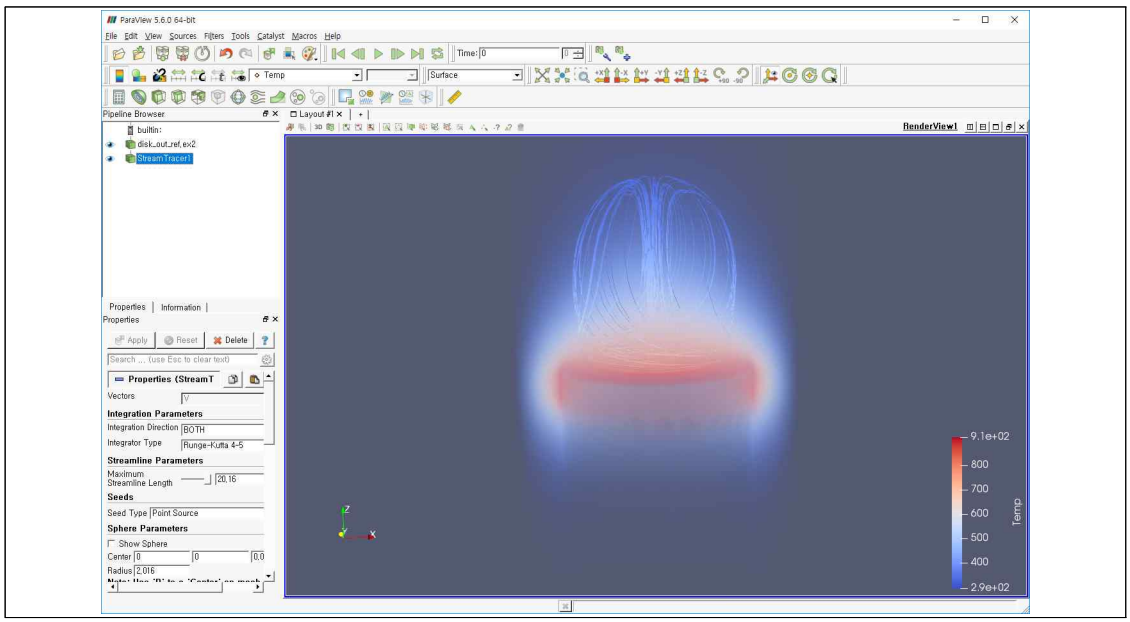


⑩ Volume Rendering

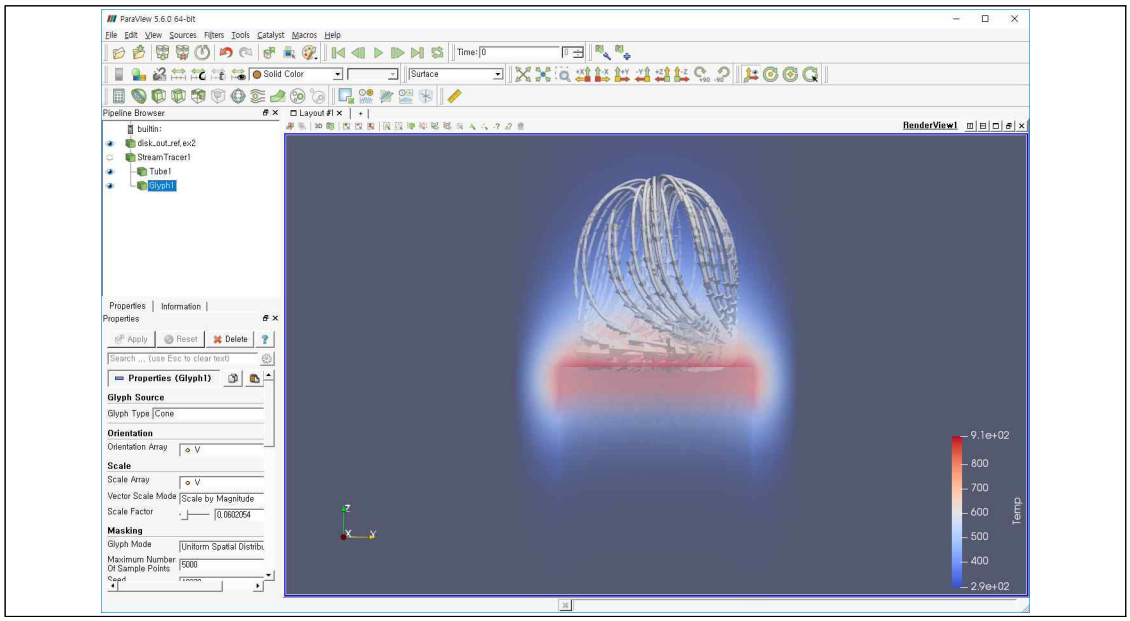
- ParaView는 다양한 데이터 표출방법을 보유하고 있으며(surface, wireframe, points, surface with edges, volume), ParaView에서 데이터를 표출하는 강력한 방법 중 하나는 volume rendering이라고 불리는 기술이다.
- Volume rendering을 사용하면 solid mesh는 cloud의 모든 점에서 color와 density를 결정하는 scalar field를 갖는 반투명의 cloud로 렌더링 된다.
- Surface 렌더링과는 달리 volume rendering은 volume 전체를 살펴볼 수 있게 해준다.
- ParaView 툴바의 Reset Session 실행 후 disk_out_ref.ex2파일을 새로 불러온다. 이후 필드 선택 창에서 Temp를 선택하고 표현방법은 Volume을 선택한 후 Yes를 클릭하면 volume rendering 결과를 표현할 수 있다.



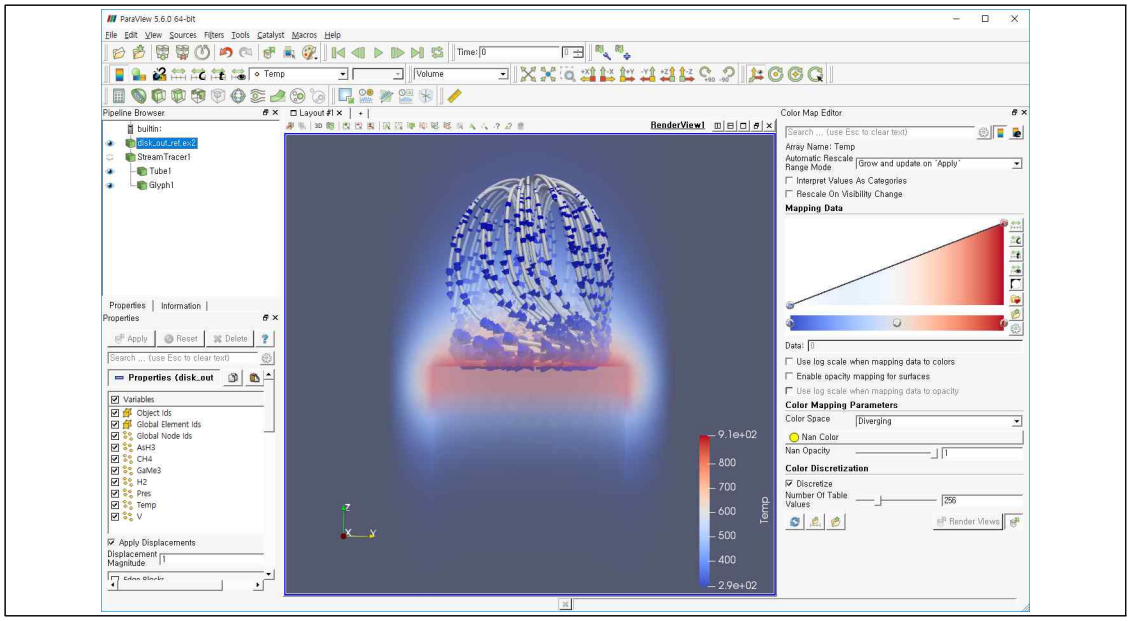
- 이후 stream tracer filter( )를 선택하고 Seed Type을 Point Source, Show Sphere를 해제하고 Apply를 적용한다.



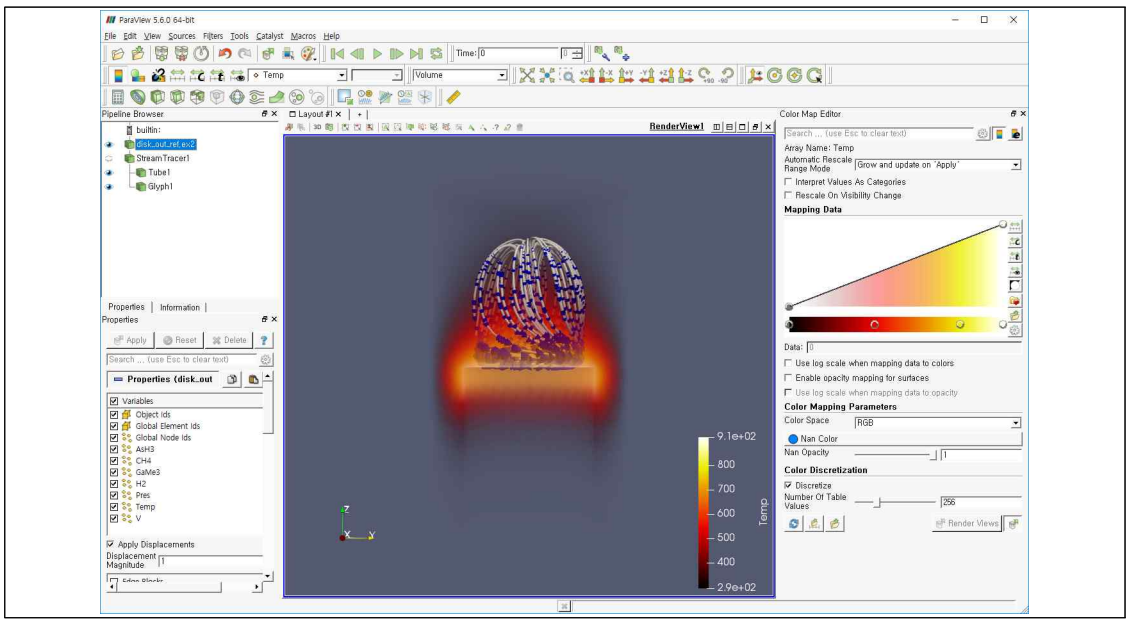
- 빠른 실행(Ctrl+space bar)으로 Tube 필터를 선택하고 적용한다. streamline에서 Temp를 Solid Color로 변경하고, pipeline browser에서 StreamTracer1을 선택한 후 glyph filter( )를 추가한다. Properties 패널에서 Glyph Type을 Cone, Orientation Array는 V, Scale Array는 V Glyph Data Range 아래의 reset ( ) 버튼을 클릭하고 Apply를 적용한다.



- 색상의 변경은 color map editor( )를 이용하여 적용한다.

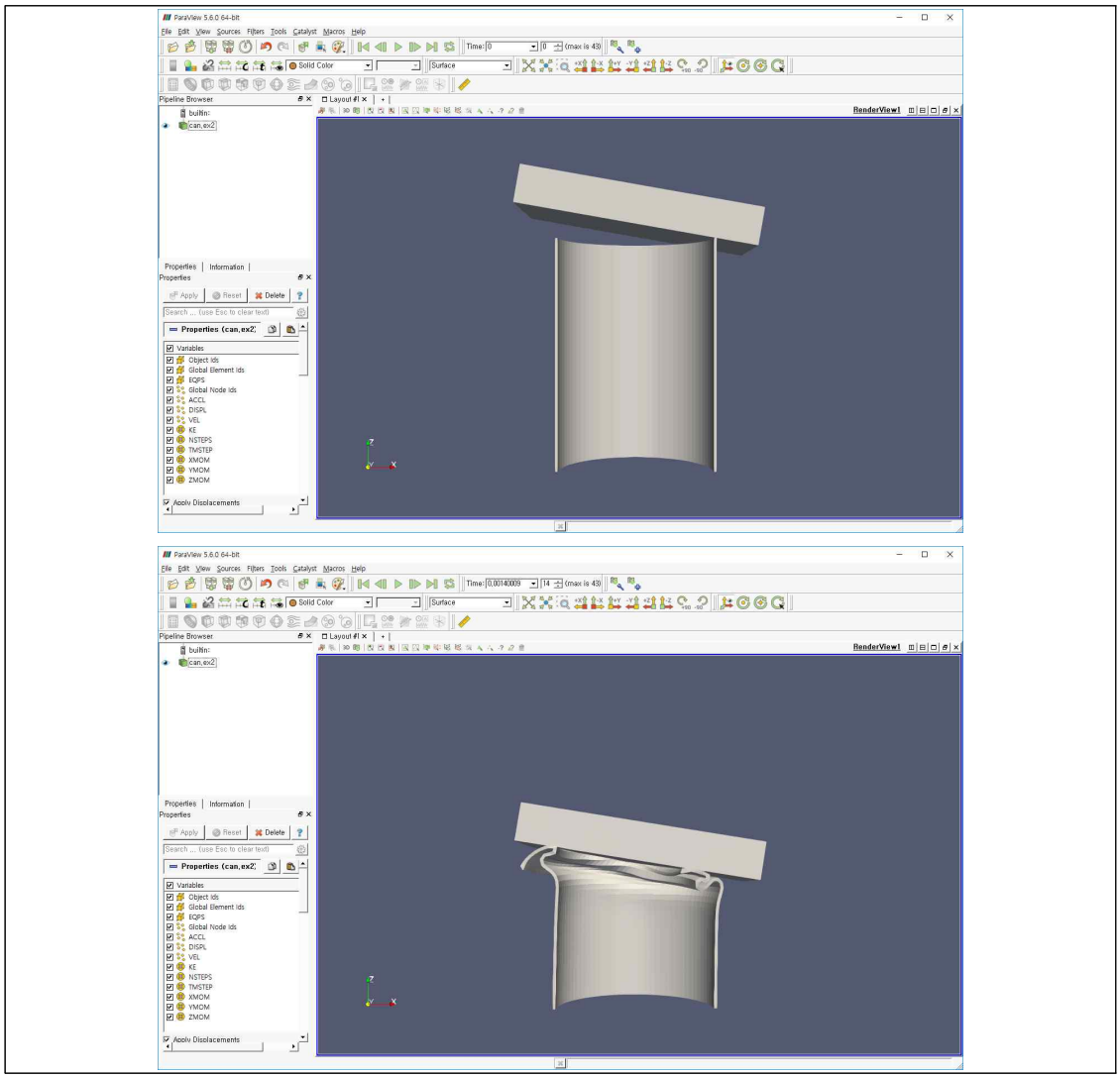


- pipeline browser에서 disk_out_ref.ex2를 선택하고 edit color map 창에서 Choose preset( )을 선택 후 Black-Body Radiation을 선택하고 Apply-Close를 클릭하면 그림은 다음과 같아진다.

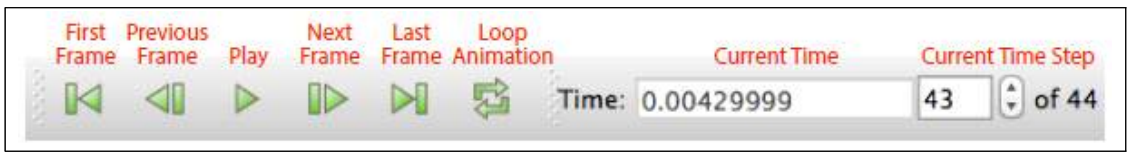


⑪ Time

- 시간에 따라 정의된 데이터를 가시화하는 방법을 다루며 앞서와 같이 Reset Session을 실행하고 example 폴더에서 can.ex2파일을 불러오고 Apply 한다.  를 클릭하여 그림을 정렬한다.

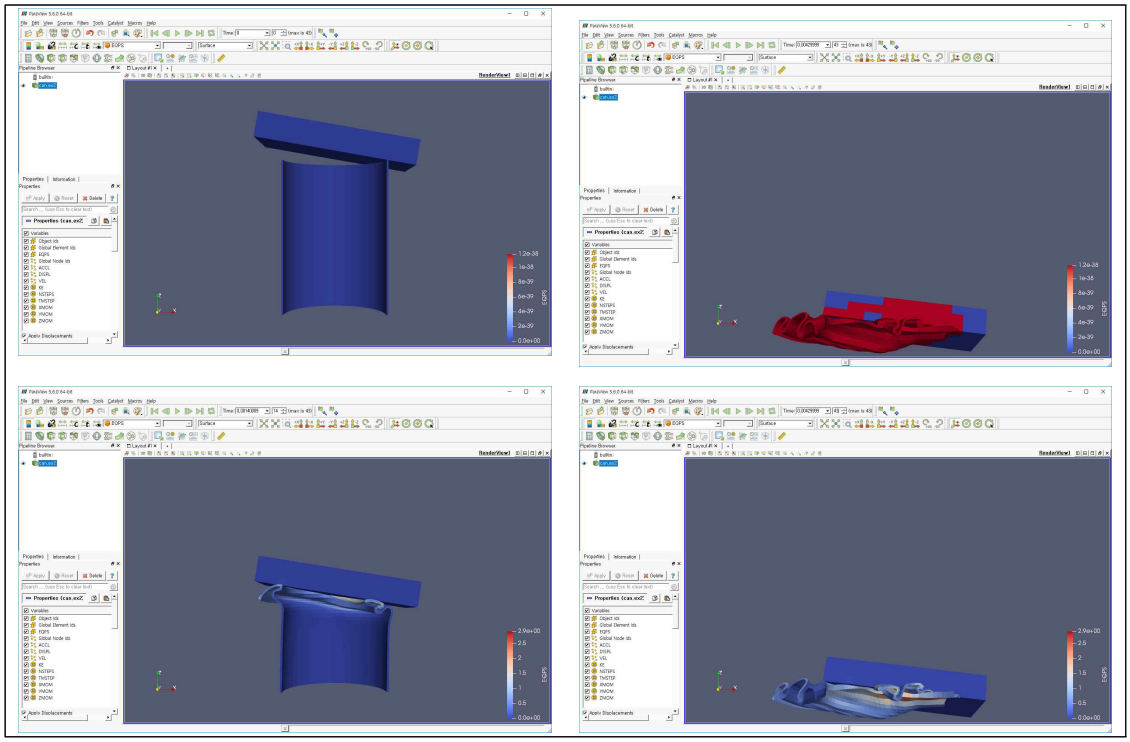


- ParaView에서 시간에 관련된 툴바는 다음과 같다.

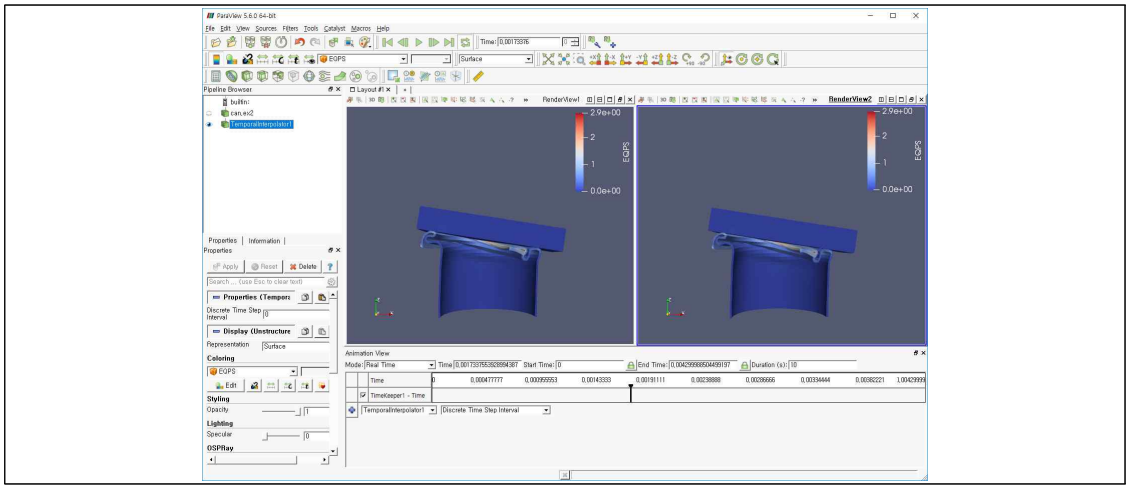


- 동영상의 저장은 File->Save Animation으로 쉽게 할 수 있다.

- 색상의 변화(변수 창에서 Solid Color 대신 EQPS를 선택)는,  툴바를 이용하여 조정할 수 있다.

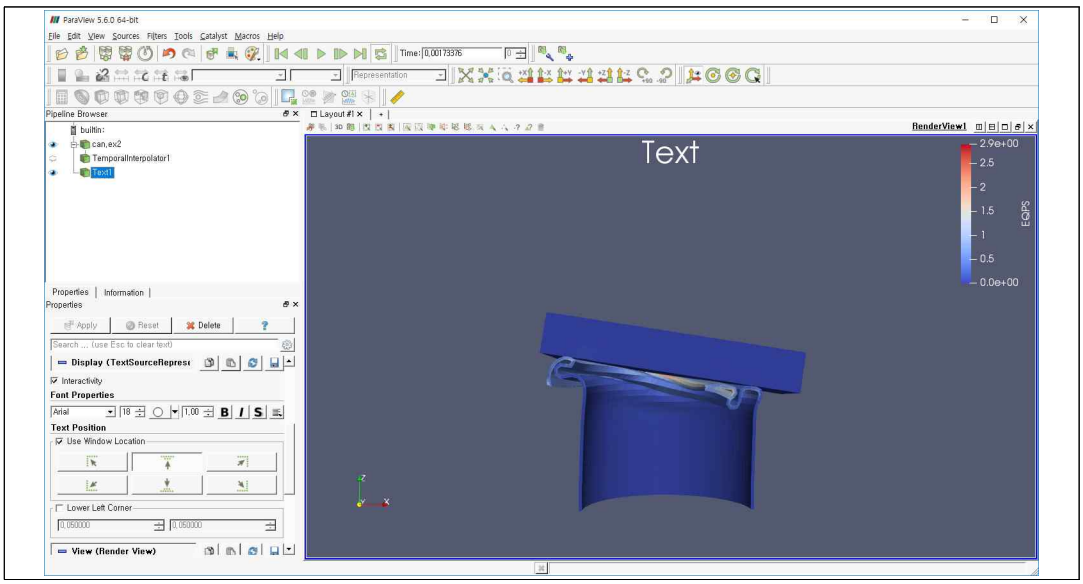


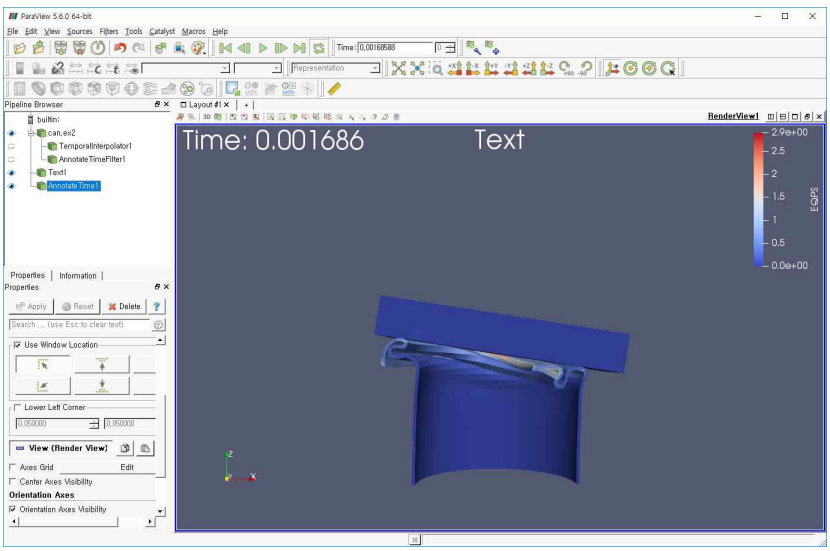
- Animation 시간의 변경은 메뉴 창에서 View-Animation View를 선택하고 Animation View 창에서 Mode를 Real Time으로 선택한 후 Duration을 조절하면 조절된 시간만큼 Animation이 실행된다.
- Filters->Temporal->Temporal Interpolator를 적용한 후 창을 분할하여 필터 전·후의 그림을 표현하고 Play animation을 실행(duration을 10초로 설정)하면 필터 적용 시의 동영상이 훨씬 부드럽게 화면이 넘겨짐을 확인할 수 있다.

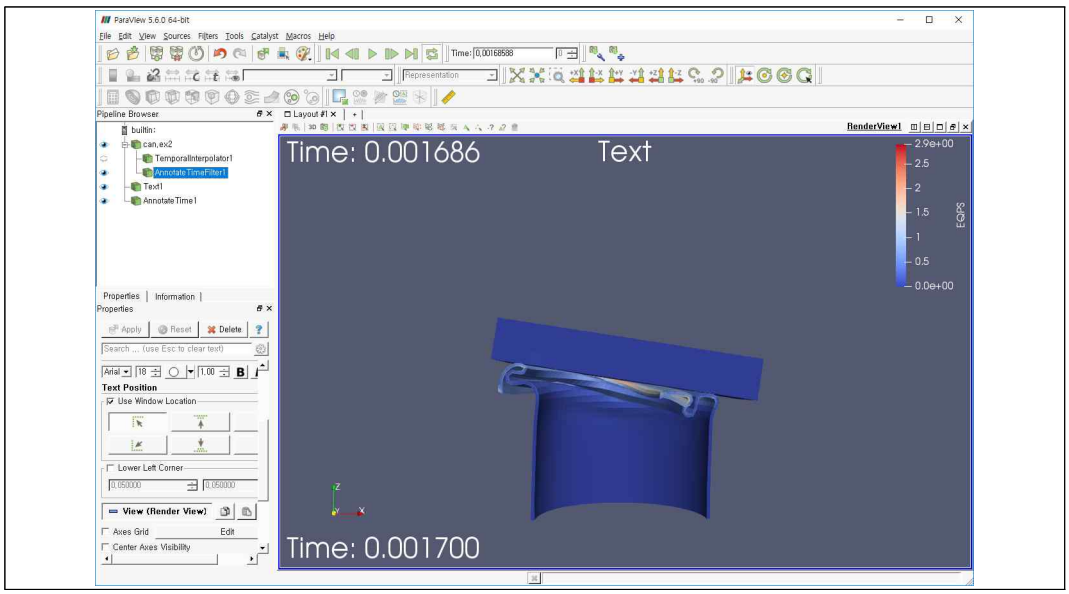


⑫ Text Annotation

- 그림 내 문구의 입력은 Sources->Annotation->Text를 Apply 하여 쉽게 생성할 수 있으며 또한, 문구의 내용 및 위치는 Properties 패널에서 조절이 가능하다.
- 문구 입력과 마찬가지로 시간의 표현은 Sources->Annotation->Annotate Time으로 입력할 수 있다.
- current animation time은 데이터 파일로부터 읽은 time step과 일치하지 않는 경우가 있음. 보통 데이터 파일에 저장된 시간을 아는 것은 중요하며, ParaView에는 필터로 작동하는 annotate time의 특별 버전이 존재한다.
- can.ex2를 pipeline browser에서 선택하고 필터에서 Annotate Time Filter를 적용. 이후 Animation View를 실행하고 Snap to TimeSteps를 Real Time으로 변경하고 Play를 실행하면 서로 다른 시간이 나타남을 확인할 수 있다.


ParaView 5.6.0 64-bit
File Edit View Sources Filters Tools Catalyst Macros Help
Pipeline Browser
builtin:
can.ex2
TemporalInterpolator1
Text1
Properties | Information
Properties
Apply Reset Delete
Display (TextSourceRepres
Interactivity
Font Properties
Text Position
Use Window Location
Lower Left Corner
View (Render View)
RenderView1
Text
2.9e+00
2.5
2
1.5
1
0.5
0.0e+00
EQPS


ParaView 5.6.0 64-bit
File Edit View Sources Filters Tools Catalyst Macros Help
Pipeline Browser
builtin:
can.ex2
TemporalInterpolator1
AnnotateTimeFilter1
Text1
AnnotateTime1
Properties | Information
Properties
Apply Reset Delete
Use Window Location
Lower Left Corner
View (Render View)
Axes Grid
Edit
Center Axes Visibility
Orientation Axes
Orientation Axes Visibility
RenderView1
Time: 0.001686
Text
2.9e+00
2.5
2
1.5
1
0.5
0.0e+00
EQPS



⑬ State

- 유사한 상황에 대해 유사한 후처리를 하고자 하는 경우 최종 작성된 그림의 state 파일을 저장하고(File-Save State) 이를 불러들여 적용(File-Load State) 할 수 있다.

제 4 장

연구개발목표 달성도 및 대외기여도

제 4장 연구개발목표 달성도 및 대외기여도

1. 2018년 목표달성도

구 분	%	성취도 판단		특기사항 (우수성 또는 부진사유 등)
		정상	부진	
당해연도 목표달성도	100%	○		연구 결과를 토대로 한 연구논문 SCIE 저널에 투고 (심사 중)
최종목표 대비 달성도	%			당해 종료 사업

2. 목표달성 내역

성과목표(가중치)	2018년 연구성과	연차목표 달성도 (%)	최종목표 달성도 (%)
1. LES 난류기법 도입 (30%)	- LES 난류기법 도입 - RANS 와 성능 비교	100	
2. 쇄파대 파랑모델 수립 (40%)	- VOF 기법 적용 쇄파현상 구현 - 쇄파대 정밀 파랑모델 수립	100	

3. 퇴적물 이동모델 수립 (30%)	- 라그랑지안 입자 추적 모델 수립 - RANS 와 LES 간 성능 비교	100	
합계 (100%)	- 연구결과 논문 투고 - 향후 실용모델과 정밀모델 결합 토대 마련	100%	%

제 5 장

연구개발결과의 활용계획

제 5 장 연구개발결과의 활용계획

1. 활용방안

○ 쇄파대 정밀 파랑모델 시스템 활용

- 본 연구를 통해 수립된 쇄파대 파랑모형은 향후 포말대 및 지하수 포함하는 쇄파대 정밀모델 시스템으로 개발 가능하다.
- 성공 시 침식현상이 가장 활발히 발생하는 포말대 퇴적물이동 예측의 정확성 향상이 기대된다.

○ 실용모델과의 결합

- 쇄파대 정밀 모델의 장점은 높은 정확성이고 단점은 계산의 비효율성이다.
- 향후 컴퓨터 계산능력의 발전에 따라 쇄파대 퇴적물이동 실용모델의 정밀화가 요구될 것으로 예상된다.
- 이에 따라 본 정밀모델을 공학적 용도의 실용모델과 결합하는 방안 계획 중이다.
- 성공 시 연안역 지형변화 정밀-실용모델 개발이 기대된다.

○ 오일러리안-라그랑지안 겸용 퇴적물이동모델 구축

- 본 과제를 통해 수행중인 라그랑지안 퇴적물 입자추적모델 완성 시 오일러리안 퇴적물농도 계산모델과 성능비교 가능하다.
- 또한 오일러리안-라그랑지안 겸용 퇴적물이동 모델 시스템 수립을 계획 중이다.

2. 기대성과 및 예상파급효과

○ 기술적 측면

- 오일러리안-라그랑지안 겸용 퇴적물이동 모델 개발 시 오일러리안과 라그랑지안 방식으로 동시에 퇴적물이동을 예측하는 세계 최초의 모델이 될 것으로 전망하며 이에 따라 퇴적물이동 모델 분야 선진국 수준의 기술력을 구비할 것으로 전망된다.

○ 경제·산업적 측면

- 연안역 지형변화 정밀-실용 모델 시스템 개발 성공 시 이 모델은 현재 사용 중인 연안역 지형변화 예측 모델들에 비해 뛰어난 정확도를 구비할 것으로 기대되며 이에 따라 연안역 구조물 설계 시에 널리 사용될 것으로 전망되며 상용화를 통한 수익 창출도 가능할 것으로 예상된다.

제 6 장

참고문헌

제 6 장 참고문헌

Chang, Y.S., Hane, D.M., 2004. Suspended sediment and hydrodynamics above mildly sloped long wave ripples. J. Geophys. Res., 109(C07022).

Chang, Y.S., Park, Y-.G., 2016. Suspension of sediment particles over a ripple due to turbulent convection under Unsteady Flow Conditions. Ocean Sci. J. 51(1), 127-135.

Chang, Y.S., Do, J.D.,, Kim, S-.S., Ahn, K., Jin, J-.Y., 2017. Measurement of turbulence properties at the time of flow reversal under high wave conditions in Hujeong Beach. J. Korean Soc. Coast. Eng., 29(4), 206-216.

Elias, E.P.L., Walstra, D.J.R., Roelvink, J.A., Stive, M.J.F., 2000, Hydrodynamic validation of Delft3D with field measurements at Egmond. International conference; 27th, Coastal engineering, 2000, Sydney.

Higuera, P., Lara, J. L., Losada, I. J., 2013. Realistic wave generation and active wave absorption for Navier-Stokes models: Application to OpenFOAM, Coast. Eng., 71, 102-118.

Kim, Y. Zhou Z. Hsu, T.-J.,Puleo, J.A., 2017. Large eddy simulation of dam-break-driven swash on a rough-planar beach. J. Geophys. Res.,122(2), 1274-1296.

Lee, K.-H., Bae, J.-H., Kim, S.-K., Kim, D.-S., 2017. Three-dimensional Simulation of Wave Reflection and Pressure Acting on Circular Perforated Caisson Breakwater by OLAFOAM. J. Korean Soc. Coast. Ocean Eng., 29(6),

286-304.

Lubin, P., Vincent, S., Abadie, S., Caltagirone, J.-P., 2006. Three-dimensional Large Eddy Simulation of air entrainment under plunging breaking waves. Coast. Eng., 53(8), 631-655.

Martins, K., Blenkinsopp, C.E., Almar, R., Zang, J., 2017. The influence of swash-based reflection on surf zone hydrodynamics: a wave-by-wave approach. Coast. Eng. 122, 27-43.

Maxey, M.R., Riley, J., 1983. Equation of motion for a small rigid sphere in a non-uniform flow. Phys. Fluids, 26(4), 883 - 889.

Noh W.F., Woodward P., 1976, SLIC (Simple Line Interface Calculation). In: van de Vooren A.I., Zandbergen P.J. (eds) Proceedings of the Fifth International Conference on Numerical Methods in Fluid Dynamics June 28 - July 2, 1976 Twente University, Enschede. Lecture Notes in Physics, vol 59. Springer, Berlin, Heidelberg.

Roelvink, J. A., Reniers, A. J. H. M., Van Dongeren, A. R., Van Thiel de Vries, J. S. M., McCall, R. T., and Lescinski, J. M., 2009. Modelling storm impacts on beaches, dunes and barrier islands. Coastal Engineering, 56(11-12), 1133 - 1152.

Ruggiero, P., Buijsman, M., Kaminsky, G.M., Gelfendaum, G., 2010. Modeling the effect of wave climate and sediment supply variability on large-scale shoreline change. Mar. Geol., 273(1-4), 127-140.

Zhou, Z., Hsu, T.-J., Cox, D., Liu, X., 2017. Large eddy simulation of wave-breaking induced turbulent coherent structures and suspended sediment transport on a barred beach. J. Geophys. Res. Oceans, 122, 207-235.

이광호, 배주현, 안성욱, 김도삼, 배기성, 2016. 파-흐름 공존장내 잠제 주변에서 OLAFOAM에 의한 파랑특성의 수치해석. 한국해안·해양공학회 논문집, 28(6). 332-349.

주 의

1. 이 보고서는 한국해양과학기술원에서 수행한 기본연구사업의 연구결과보고서입니다.

2. 이 보고서 내용을 발표할 때에는 반드시 한국해양과학기술원에서 수행한 기본연구사업의 연구결과임을 밝혀야 합니다.

3. 국가과학기술 기밀유지에 필요한 내용은 대외적으로 발표 또는 공개하여서는 안됩니다.