부유체 운동해석을 위한 Euler Overlay Method (EOM) 구현

연성모

삼성중공업 조선해양연구소 선박해양연구센터

2019. 9. 26

Agenda

- Euler Overlay Method 구현
- EOM Zone 자료구조 구현
- (간단한) 구현 방법에 따른 성능 비교

Euler Overlay Method

- Coupling of potential and CFD solutions
  - Purpose
    - Numerical diffusion of wave
    - Reduce computational cost
  - 구현방법
    - Relaxation method
      - » Relaxation method by Jacobsen
        - + waves2Foam
    - Momentum forcing method
      - » Explicit momentum forcing method (EOM) by Jang Kim
        - + Starccm+
      - » Implicit momentum forcing method by Jasak
        - + Naval Hydro Pack

Euler Overlay Method 구현 방법

- Relaxation method
  - waves2Foam
    - 각 governing equation에서 나온 solution에 직접 weight 적용하여 합성
    - $\phi = (1 - w)\phi + w\phi_p$
- Momentum forcing method
  - Governing equation에서 나온 solution이 이미 합성된 solution임
  - Implicit form (Naval Hydro Pack)
    - $(1 - w)Q(\phi) + wR(\phi) = 0$
      - » $Q(\phi)$: implicit form of governing equation
      - » $R(\phi) = (\phi - \phi_p)/\Delta t$: implicit form of boundary condition
    - Δt: governing eq.의 time step과 동일해야함
  - Explicit form (EOM by Jang Kim)
    - $Q(\phi) = wQ(\phi) - wR(\phi)$
    - Δt: governing eq.의 time step과 동일하면 안됨. 사용자 입력. 추천값은 2

Euler Overlay Method 구현 방법

- Relaxation method
  - 각각의 governing equation을 수정할 필요 없음
  - 상대적으로 구현방법이 간단함
- Momentum forcing method
  - Explicit method
    - 각 governing equation의 source항을 이용하므로 source 항을 지원하는 코드에서는 구현 가능
      - » Commercial S/W에 적용가능
    - Missing term이 있어 파제거에 한계가 있을 것으로 예상됨
      - » Two-way couplin을 극복
    - Weight 값이 1일 때도 potential solutio을 강제하지 못함
  - Implicit method
    - 각 governing equation의 구현을 이해해야 적용가능.
    - 가장 강력한 coupling method로 보임

interFoam의 VoF 해석 개요

- OpenFOAM 1.x - OpenFOAM 2.2
  - Explicit solve로 VoF 해석
    - $\frac{\partial \alpha}{\partial t} + \nabla \cdot (\alpha U) + \lambda \nabla \cdot (\alpha(1-\alpha)U_r) = 0$
    - λ 계산을 위해 MULES 알고리즘 적용
      - » Flux limiter scheme
- OpenFOAM 2.3.x – OpenFOAM 7
  - Semi explicit solver 지원
    - $\frac{\partial \alpha}{\partial t} + \nabla \cdot (\alpha U) + \lambda\big(\nabla \cdot (\alpha U) - \nabla \cdot (\alpha U)\big) + \lambda[\nabla \cdot (\alpha(1-\alpha)U_r)] = \frac{\partial \alpha}{\partial t} + \nabla \cdot (\alpha U) + \lambda[\nabla \cdot (\alpha U) + \nabla \cdot (\alpha(1-\alpha)U_r) - \nabla \cdot (\alpha U)] = 0$
    - Operator splitting 기법 적용
      - » Predictor: $\frac{\partial \alpha}{\partial t} + \nabla \cdot (\alpha U) = 0$ by implicit solver with upwind convection scheme
      - » Corrector: $\frac{\partial \alpha}{\partial t} + \lambda[\nabla \cdot (\alpha U) + \nabla \cdot (\alpha(1-\alpha)U_r) - \nabla \cdot (\alpha U)]$ by explicit solver (CMULES)

interFoam의 VoF 해석 개요

- MULES::explicitSolve
  - OpenFOAM 2.2까지 있던 VoF equation solver
  - Subcycling을 하므로 subcycling volume을 반영해야함
  - Temporal term은 직전 solution만을 이용 => backward scheme 지원 못함
  - $-\frac{\partial \phi}{\partial t} + F = S_p \phi + S_u \Rightarrow \phi^n = \frac{\frac{1}{\Delta t}\phi^o + S_u - F}{\frac{1}{\Delta t} - S_p}$

```
if (mesh.moving())
{
    psiIf =
    (
        mesh.Vsc0().field()*rho.oldTime().field()
       *psi0*rDeltaT/mesh.Vsc().field()
      + Su.field()
      - psiIf
    )/(rho.field()*rDeltaT - Sp.field());
}
else
{
    psiIf =
    (
        rho.oldTime().field()*psi0*rDeltaT
      + Su.field()
      - psiIf
    )/(rho.field()*rDeltaT - Sp.field());
}
```

- CMULES::correct
  - Semi implicit-explicit solver에서 corrector에 적용하는 solver
  - Predictor에서 구한 값의 correction 값이므로 subcycling으로 인한 volume 변화를 고려하지않음

interFoam의 VoF 해석 기능 개선

- Backward scheme 지원 추가
  - Backward scheme derivation
    - $\frac{\partial \phi}{\partial t} \approx \frac{1}{h}(a\phi^n + b\phi^o + c\phi^{oo})$ where $b = (a + c)$
    - CMULES::correct

» $\phi^n = \frac{\frac{1}{\Delta t}(b\phi^o - c\phi^{oo}) + S_u - F}{\frac{a}{\Delta t} - S_p}$

Potential solution coupling

- Explicit scheme으로 potential + CFD solution 합성하면 안됨
  - $\phi^n = \frac{\frac{w}{\Delta t}\phi^o + S_u - F}{\frac{1-w}{\Delta t} - wS_p} = \frac{\frac{w}{\Delta t}\phi^o + S_u - F}{\frac{1-w}{\Delta t} - \frac{w}{\Delta t}}$ => weight 값이 0.5이면 divide by zero 오류
  - 따라서
    - relaxation method
      - » MULES, CMULES 사용 가능
    - momentum forcing method는 implicit method로 동작
      - » MULES, CMULES 사용 안함

EOM zone 구현

- relaxationZone (in waves2Foam and Naval Hydro Pack)
  - 문제점
    - fvPatchField 경계조건 내부에서 waveTheory 객체 생성 (waves2Foam)
      - » Initial 생성자: waveAlpha, waveVelocity에서 독립적으로 New selector 실행 생성 => 동종 객체 중복 생성 이슈,
      - » 복사 생성자: time step/iteration마다 New selector실행 생성 => 반복적인 객체 생성/소멸로 인한 오버헤드
    - relaxationShape (waves2Foam, Naval Hydro Pack)
      - » relaxationShape cell collection 기능 자체 구현
      - » Overlapped relaxationShape에 대한 처리 미지원
    - relaxationZone 생성시 별도의 입력필요 (Naval Hydro Pack)
      - » Runtime 실행과 pre-processing에 사용되는 입력파일이 다름
      - » waveProperties + setBatch
    - 경계에서만 potential solution으로 치환됨

문제1: 경계 조건에서 반복적인 객체 생성/소멸

- 비효율적인 객체 관리
  - 생성자 내부에서 wavePropes_ 변수에 객체 할당
  - 경계조건은 계산시 반복적으로 생성/소멸되는 객체임
    - 경계조건 내부에서 waveTheory 객체의 생성 소멸됨
      - » 호출시 객체초기화 후 elapsed time에 대한 elevation을 계산함
        - + waveTheory 객체의 초기화가 반복적임.
        - + waveTheory 초기화 비용이 비싼 경우 계산효율에 영향 미침 (Tehcnip library)

```
class waveAlphaFvPatchScalarField
:
    public mixedFvPatchField<scalar>,
    public convexPolyhedral
{

private:

    //- Private member functions
        autoPtr<waveTheories::waveTheory> waveProps_;
```

```
waveAlphaFvPatchScalarField::waveAlphaFvPatchScalarField
(
    const fvPatch& p,
    const DimensionedField<scalar, volMesh>& iF
)
:
    waveProps_
    (
        waveTheories::waveTheory::New
        (
            this->patch().name(),
            this->internalField().mesh()
        )
    )
{
    this->refValue() = pTraits<scalar>::zero;
    this->refGrad() = pTraits<scalar>::zero;
    this->valueFraction() = 0.0;
}
```

```
waveAlphaFvPatchScalarField::waveAlphaFvPatchScalarField
(
    const fvPatch& p,
    const DimensionedField<scalar, volMesh>& iF,
    const dictionary& dict
)
:
    mixedFvPatchField<scalar>(p, iF),
    waveProps_
    (
        waveTheories::waveTheory::New
        (
            this->patch().name(),
            this->internalField().mesh()
        )
    )
{
    evaluate();
}
```

```
waveAlphaFvPatchScalarField::waveAlphaFvPatchScalarField
(
    const waveAlphaFvPatchScalarField& ptf,
    const DimensionedField<scalar, volMesh>& iF
)
:
    mixedFvPatchField<scalar>(ptf, iF),
    waveProps_
    (
        waveTheories::waveTheory::New
        (
            this->patch().name(),
            this->internalField().mesh()
        )
    )
{
}
```

문제1: 경계 조건에서 반복적인 객체 생성/소멸


```
//- Return reference to the object data
inline T& operator()();

//- Return const reference to the object data
inline const T& operator()() const;

//- Return reference to the object data
inline T& operator*();

//- Return const reference to the object data
inline const T& operator*() const;

//- Const cast to the underlying type reference
inline operator const T&() const;

//- Return object pointer
inline T* operator->();

//- Return const object pointer
inline const T* operator->() const;

//- Take over the object pointer from parameter
inline void operator=(T*);

//- Take over the object pointer from parameter
inline void operator=(const autoPtr<T>&);
```

- 해결방법
  - autoPtr Issue
    - autoPtr 대신 참조변수(&)로 대체할 경우
      - » New selector에서 반환된 autoPtr 객체는 lvalue에만 저장할 수 있으므로 사용불가
    - autoPtr에 한번만 객체 생성 시킬 경우
      - » autoPtr은 소유권 이전이 발생하므로 복사생성자에서 (병렬 연산시) 메모리 참조 에러 발생
  - 객체 관리 방법
    - 별도의 공간에 PtrList 로 관리할 경우 (Naval Hydro Pack 방법)
      - » relaxationScheme 객체에 waveTheory 객체 참조를 관리할 변수를 두어 관리
        - + 반복적인 초기화를 피할 수 있음
      - » Looks promising but…
        - + waveTheory와 관계없는 relaxationScheme 자료구조와 강한 의존관계 발생

```
class relaxationZones
:
    public IOdictionary
{
    // Private data

        wordList waveNames_;

        //- HashTable containing wave index for each wave name for easy
        HashTable<label> waveIndices_;

        //- Run-time selectable wave theories
        PtrList<waveTheories::waveTheory> waveTheories_;

        //- List of names of all relaxation zones
        wordList relaxNames_;

        //- Run-time selectable relaxation scheme
        PtrList<relaxationSchemes::relaxationScheme> relaxSchemes_;
```

문제1: 경계 조건에서 반복적인 객체 생성/소멸

– 객체를 전역변수로 관리

- Fortran 또는 in-house code 방식 방법, OOP에서 권장하지 않는 방법, but
- OpenFOAM 자원관리자 (objectRegistry)가 객체를 전역변수처럼 관리함
  - » regIOobjects객체는 objectRegistry 자원관리자에 등록되어 어디서든 접근가능함
    - + objectRegistry에 저장되는 객체는 cache로도 활용가능 (예, grad(U) 등)
    - + waveTheory는 Iodictionary 서브클래스이므로 regIOobjects 서브클래스임
    - + regIOobjects의 class 함수인 store를 이용하여 cache로 저장
    - + 프로그램 종료시까지 메모리에 접근가능
  - » waveTheory를 relaxationScheme과 독립적으로 운영가능함
    - + Swiss Army Knife class 지양

```
autoPtr<waveTheory> waveTheory::New
(
    const word& waveModelName,
    const fvMesh& mesh
)
{
    if (!mesh.foundObject<IOdictionary>("waveProperties"))
    {
        regIOobject::store
        (
            new IOdictionary
            (
                IOobject
                (
                    "waveProperties",
                    mesh.time().constant(),
                    mesh,
                    IOobject::MUST_READ,
                    IOobject::NO_WRITE,
                    true
                )
            )
        );

    }
```

문제1: 경계 조건에서 반복적인 객체 생성/소멸

- waveTheory 이름 관리
  - » IOdictionary의 서브클래스로 만들어져 waveProperties라는 이름으로 등록됨
    - + 복수의 waveTheory 객체 생성시 객체간 구분이 안되는 문제 발생 (name shading)
  - » IOobject의 groupName 함수 사용하여 waveTheory.<waveTheoryName> 형태로 이름 등록
  - » OpenFOAM 8에서는 modelName 함수를 지원하므로 추후 groupName에서 modelName으로 코드 변경 예정

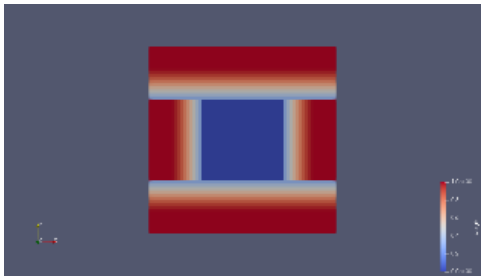
```
waveTheory::waveTheory
(
    const word& subDictName,
    const fvMesh& mesh_
)
:
    IOdictionary
    (
        mesh_.thisDb().lookupObject<IOobject>("waveProperties")
    ),
```

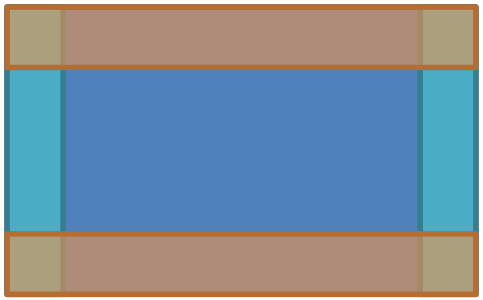
```
waveTheory::waveTheory
(
    const word& waveModelName,
    const fvMesh& mesh
)
:
    IOdictionary
    (
        IOobject
        (
            IOobject::groupName(waveModelName, "waveTheory"),
            mesh.time().constant(),
            mesh,
            IOobject::NO_READ,
            IOobject::NO_WRITE,
            true
        ),
        mesh.lookupObject<IOdictionary>("waveProperties")
    ),
```

문제2: relaxationShape

- relaxationShape 형상
  - OpenFOAM에서 이미 제공하는 기능과 중복

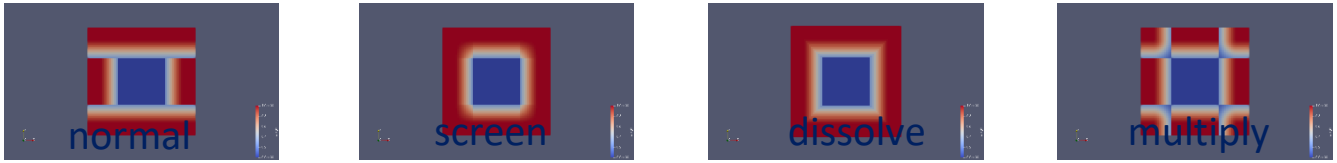
- Wave generation practice
  - 반사파 영향을 줄이기 위해 모든 경계에 relaxation zone 설정
  - Overlapping 구간 존재





문제2: relaxationShape의 overlapping

- 해결방법
  - relaxationShape
    - In-house 코드를 topoSet 객체활용 코드로 대체
  - Naval Hydro Pack의 방법?
    - 모든 relaxationZone마다 전체 relaxationZone의 범위를 명시
      - » 불필요한 좌표가 들어가므로 가독성에 방해
      - » 불필요한 좌표의 내용까지 weight 계산에 들어가므로 초기화 효율 저하
  - Alpha blending 기법 활용
    - Weight field는 [0,1] 사이의 값이므로 이미지 합성 기법 활용가능
      - » Normal, screen, dissolve, multiply 합성 기법 구현



문제3: 입력정보 중복

- relaxationZone 생성
  - setSet 명령으로 사전 생성되도록 구현 (Naval Hydro Pack)
    - setBatch 파일에 topoSet 명령 기록
    - relaxationZone 정보는 waveProperties에도 있음
    - 중복 정보 제거 필요

문제3: 입력정보 중복

- 해결방법
  - relaxationZone 생성시 topoSet 객체 생성하여 runtime에 cellZone 생성

```
void EOMShapeRectangular::findComputationalCells()
{
    word setType = "cellSet";
    word setName = zoneName_; // this should be referred to zone name
    word sourceType = "boxToCell";

    zoneSetPtr_ = topoSet::New(setType, mesh_, setName, mesh_.C().size());
    dictionary dict;
    point corner1(coeffDict_.lookup("startX"));
    point corner2(coeffDict_.lookup("endX"));
    point pmin(point(min(corner1.x(), corner2.x()),min(corner1.y(), corner2.y()),min(corner1.z(), corner2.z())));
    point pmax(point(max(corner1.x(), corner2.x()),max(corner1.y(), corner2.y()),max(corner1.z(), corner2.z())));
    treeBoundBoxList tbb(1, treeBoundBox(pmin, pmax));

    dict.add("boxes", tbb);
    setSource_ = topoSetSource::New(sourceType, mesh_, dict);
    topoSet& zoneSet = zoneSetPtr_();
    topoSetSource::setAction action = topoSetSource::toAction("new");
    setSource_().applyToSet(action, zoneSet);
    cells_.setSize(zoneSet.size());
    label count(0);
    forAllConstIter(cellSet, zoneSet, iter)
    {
        cells_[count++] = iter.key();
    }

    zoneSet.write();
}
```

문제4: 경계에서만 반사파 영향 고려

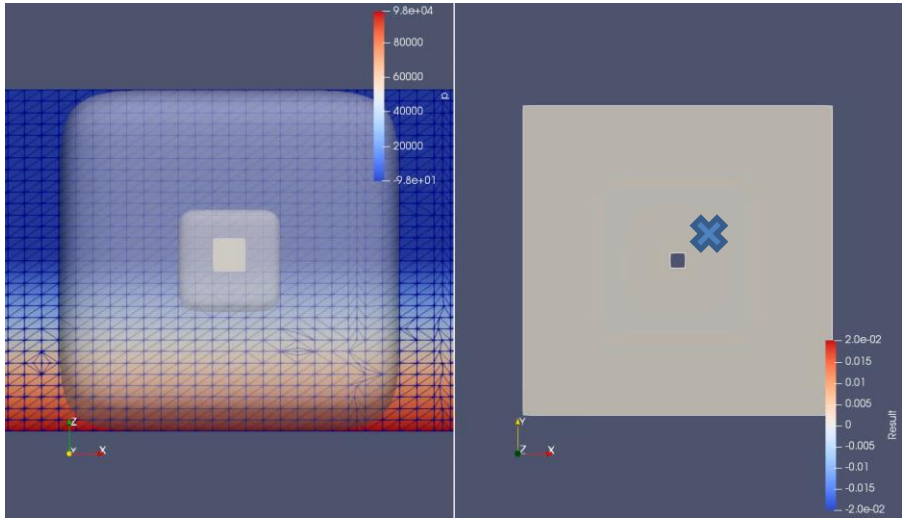
- internalField에서 반사파제거 고려안됨
  - waves2Foam
  - NavalHydroPack

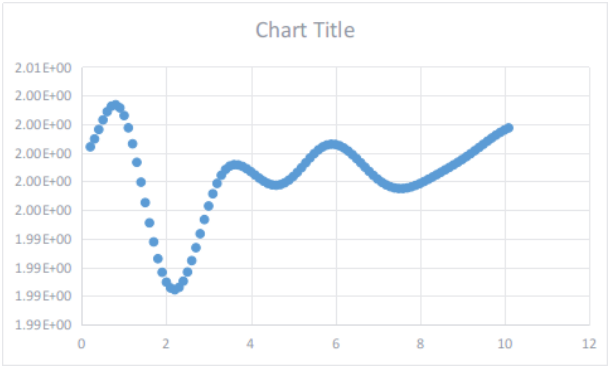
문제4: 경계에서만 반사파 영향 고려

- relaxationShape 재설계
  - Sigma 좌표계에서 shift 변수 추가

파 감쇄 성능 비교 (Radiation Problem)

- Floating object
  - Free fall of cuboid
  - without EOM zone



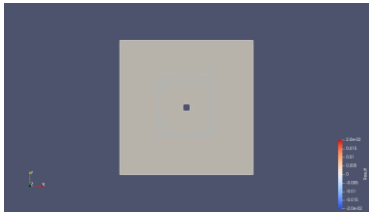


파 감쇄 성능 비교

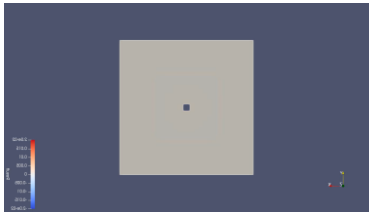
- EOM zone 적용
  - Relaxation method
    - After vof equation, solution relaxation for U and alpha
    - Then, put into momentum equation.
    - Pressure and velocity may not satisfy coupled field
  - Implicit forcing method
  - Explicit forcing method
    - Due to the missing term $wQ(\phi)$, strict enforcing is not possible
      - » $Q(\phi) = wQ(\phi) - wR(\phi)$
    - 파 제거 성능은 떨어짐

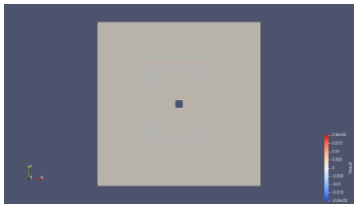
CFD-based Numerical Wave Basin for FPSO in Irregular Waves, OMAE2019-96838

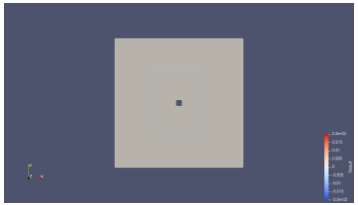
MULES



CMULES

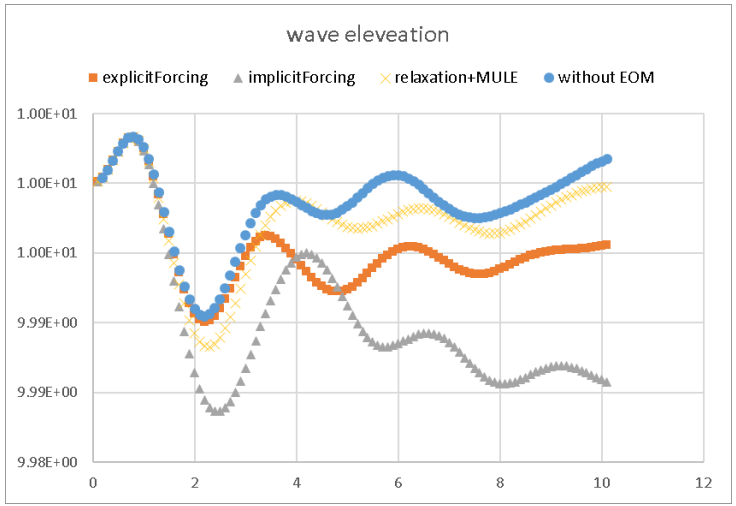


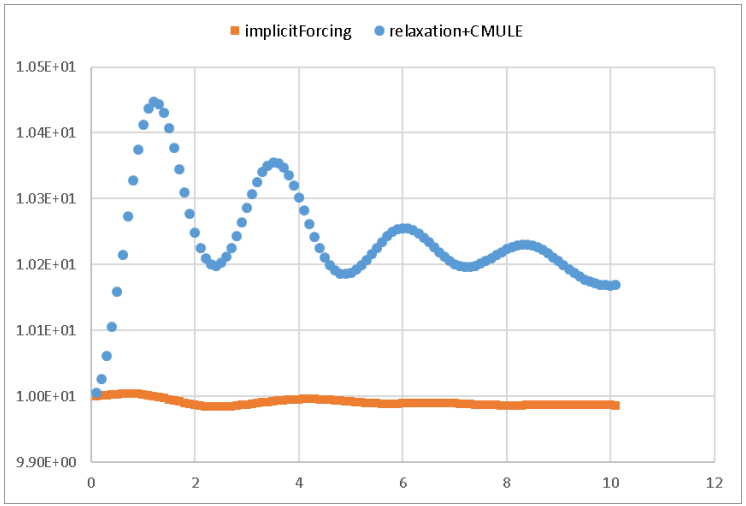




파 감쇄 성능 비교

- EOM zone 적용
  - Relaxation+CMULE, implcitForcing이 반사파 영향이 거의 없음
    - Relaxation+CMULE의 경우 다른 방법보다 radiation wave elevation 크기가 다름





Conclusion

- 다양한 Euler Overlay Method 구현
  - Relaxation method, momentum forcing methods
- relaxationZone 재설계
  - 빈번한 객체생성/삭제 제거
  - relaxationShape 합성 방법 구현
  - Weight field 제어 기능 추가
- 파 제거 기능
  - Relaxation, implicit momentum forcing method
    - 파제거 효과
  - Explicit momentum forcing method
    - 파제거에 한계 있음


Thank you very much!