Improve sheet cavitation inception prediction by taking laminar separation into consideration

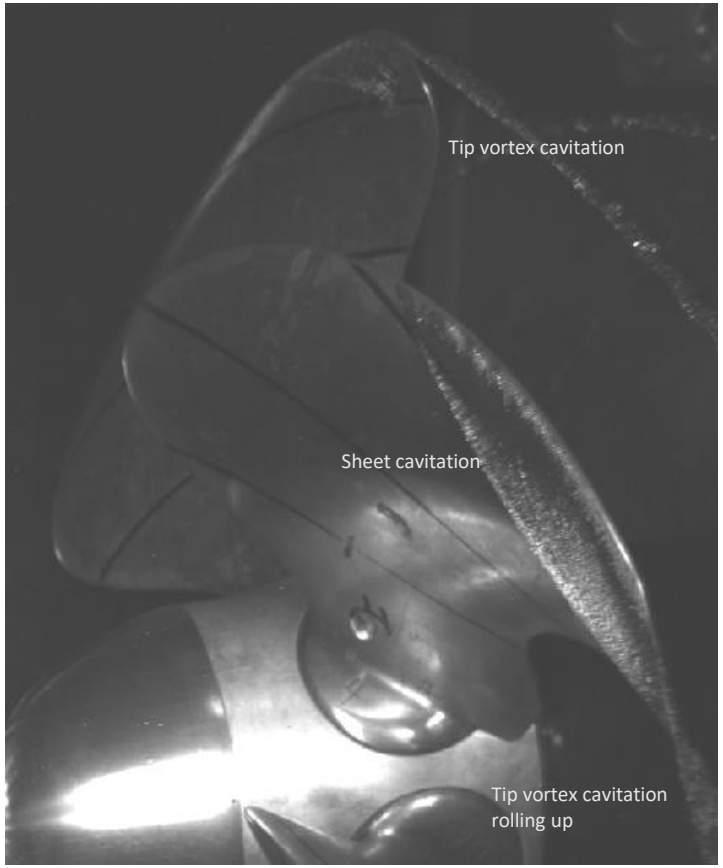
Muye Ge
Marine Technology
Mechanic and Maritime Sciences
Chalmers University of Technology
Gothenburg, Sweden
2018. 11. 28

Outline

- Theoretical background
- Implementation details
- Test Case

Theoretical background

cavitation



- Phase change due to pressure
- Common in marine propellers, impellers, engine nozzles, hydrofoils, etc…
- One of the major source of marine propeller induced pressure pulses and erosion

Theoretical background

$\gamma - Re_\theta$ transition sensitive model

- First published by Langtry in 2006
  - Fully published by Langtry and Mentor in 2009
- Correlation based
  - Based on 2D experiments, influence of pressure gradient, free stream intensity etc.
  - Able to predict natural transition, bypass transition and **separation-induced transition**
- Two additional transport equations
  - $\frac{\partial(\rho\gamma)}{\partial t} + \frac{\partial \rho U_j \gamma}{\partial x_j} = P_\gamma - E_\gamma + \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_f}\right)\frac{\partial \gamma}{\partial x_j}\right]$
  - $\frac{\partial(\rho\widetilde{Re}_{\theta t})}{\partial t} + \frac{\rho U_j \widetilde{Re}_{\theta t}}{\partial x_j} = P_{\theta t} + \frac{\partial}{\partial x_j}\left[\sigma_{\theta t}(\mu + \mu_t)\frac{\partial \widetilde{Re}_{\theta t}}{\partial x_j}\right]$
- Coupled with $k\omega - SST$

$\frac{\max(Re_v)}{2.193Re_\theta} \sim H$, $Re_v = \frac{\rho y^2}{\mu} S$

For laminar separation, $H_{sep} = 3.5$

$\frac{Re_v}{3.235Re_\theta} - 1$ to account for H higher than 3.5

Theoretical background

Schenerr Sauer cavitation model

- Assumptions:
  - Initial nuclei uniformly distributed in the water medium
  - All nuclei are spherical and have same size
- Formulations:
  - Phase change decomposed into two terms
    - Condensation term $\dot{m}_{ac} = C_c \alpha_l \frac{3\rho_l \rho_v}{\rho_m R} \sqrt{\frac{2}{3\rho_l}} \sqrt{\frac{1}{|p - p_{threhold}|}} max(p - p_{threhold}, 0)$
    - Vaporization term $\dot{m}_{av} = C_v (1 + \alpha_{nuc} - \alpha_l) \frac{3\rho_l \rho_v}{\rho_m R} \sqrt{\frac{2}{3\rho_l}} \sqrt{\frac{1}{|p - p_{threhold}|}} max(p - p_{threhold}, 0)$

Theoretical background

VoF treatment

$$\frac{\partial \alpha}{\partial t} + \bigtriangledown \cdot \alpha \mathbf{u} = 0$$

$$\rho_m = \alpha_1 \rho_1 + (1 - \alpha_1)\rho_2, \quad \mu_m = \alpha_1 \mu_1 + (1 - \alpha_1)\mu_2$$

Hyperbolic Equation, need to maintain boundness and accuracy
TVD/NVD schemes (**T**otal **V**ariation **D**iminishing, **N**ormalized **V**ariable **D**iminishing, vanleer)
HRS schemes (**H**igh **R**esolution **S**chemes, HRIC and CICSAM)
FCT (**F**lux **C**orrect **T**ransport, MULES (**MU**lti-dimensional **L**imiter for **E**xplicit **S**olution) and CMULES)

$$\frac{\alpha_{i(upwind)}^{n+1} - \alpha_i^n}{\Delta t} V + \phi_{upwind}^n = 0$$

$$\frac{\alpha_i^{n+1} - \alpha_{i(upwind)}^{n+1}}{\Delta t} V + \lambda(\phi_{highOrder}^n - \phi_{upwind}^n) = 0$$

Implementation

VoF treatment

```
cd $WM_PROJECT_USER_DIR
mkdir applications; cd $WM_PROJECT_DIR/applications
cp -r --parents ./solvers/multiphase/interPhaseChangeFoam $WM_PROJECT_USER_DIR/applications
cd $WM_PROJECT_USER_DIR/applications/solvers/multiphase/
mv interPhaseChangeFoam interPhaseChangeFoamMod; cd interPhaseChangeFoamMod
mv interPhaseChangeFoam.C interPhaseChangeFoamMod.C
sed -i s/interPhaseChangeFoam/interPhaseChangeFoamMod/g interPhaseChangeFoamMod.C
sed -i s/interPhaseChangeFoam/interPhaseChangeFoamMod/g Make/*
sed -i s/FOAM_APPBIN/FOAM_USER_APPBIN/g Make/files
```

Implementation
VoF treatment

Modify the *alphaEqnSubCycle.H* to:

```
{
    volScalarField divU(fvc::div(phi));

    if (nAlphaSubCycles > 1)
    {
        dimensionedScalar totalDeltaT = runTime.deltaT();
        surfaceScalarField rhoPhiSum("rhoPhiSum", rhoPhi);

        for
        (
            subCycle<volScalarField> alphaSubCycle(alpha1, nAlphaSubCycles);
            !(++alphaSubCycle).end();
        )
        {
            #include "alphaEqn.H"
            rhoPhiSum += (runTime.deltaT()/totalDeltaT)*rhoPhi;
        }

        rhoPhi = rhoPhiSum;
    }
    else
        {
            #include "alphaEqn.H"
        }

        rho == alpha1*rho1 + alpha2*rho2;
}
```

Implementation

VoF treatment

echo "const dictionary& alphaControls = mesh.solverDict(alpha1.name());" > alphaControls.H

echo "label nAlphaCorr(readLabel(alphaControls.lookup(\"nAlphaCorr\")));" >> alphaControls.H

echo "label nAlphaSubCycles(readLabel(alphaControls.lookup(\"nAlphaSubCycles\")));" >> alphaControls.H

Lastly modify alphaEqn.H, with following content:

```
{
    {
     word alphaScheme("div(phi,alpha)");
     for (int aCorr=0; aCorr<nAlphaCorr; aCorr++)
     {
         Pair<tmp<volScalarField>> vDotAlphal =
         mixture->vDotAlphal();
         const volScalarField& vDotcAlphal = vDotAlphal[0]();
         const volScalarField& vDotvAlphal = vDotAlphal[1]();
         const volScalarField vDotvmcAlphal(vDotvAlphal - vDotcAlphal);
```

Implementation

VoF treatment

```
        fvScalarMatrix alpha1Eqn
        (
            fvm::ddt(alpha1)
          + fvm::div(phi, alpha1, "div(phi,alpha)")
          - fvm::Sp(divU, alpha1)
          ==
            fvm::Sp(vDotvmcAlphal, alpha1)
          + vDotcAlphal
        );
        alpha1Eqn.solve();
        alpha1 = min(max(alpha1, scalar(0)), scalar(1));
        alpha2 = scalar(1) - alpha1;
    }
    rhoPhi = fvc::flux(phi,alpha1, "div(phi,alpha)")*(rho1 - rho2) + phi*rho2;
    Info<< "Liquid phase volume fraction = "
        << alpha1.weightedAverage(mesh.V()).value()
        << "  Min(" << alpha1.name() << ") = " << min(alpha1).value()
        << "  Max(" << alpha1.name() << ") = " << max(alpha1).value()
        << endl;
}
```

Implementation

Transition model

cd $WM_PROJECT_DIR; cp -r --parents ./src/TurbulenceModels $WM_PROJECT_USER_DIR

cd $WM_PROJECT_USER_DIR/src/TurbulenceModels

Modify the file incompressible/turbulentTransportModels/turbulentTransportModels.C,

add following sentences after makeRASModel (kOmegaSSTLM);

#include "kOmegaSSTLMSep.H"

makeRASModel(kOmegaSSTLMSep);

Then in the terminal type:

sed -i s/FOAM_LIBBIN/FOAM_USER_LIBBIN/g turbulenceModels/Make/files

cd turbulenceModels/RAS; cp -r kOmegaSSTLM kOmegaSSTLMSep; cd kOmegaSSTLMSep

sed -i s/FOAM_LIBBIN/FOAM_USER_LIBBIN/g ./*/Make/files

mv *.C kOmegaSSTLMSep.C; mv *.H kOmegaSSTLMSep.H

sed -i s/kOmegaSSTLM/kOmegaSSTLMSep/g *

Implementation

Transition model

After the intermittency equation, below the sentence gammaIntEff_ = max(gammaInt_(), gammaSep);,

Add:

```
// Modifying the laminar separation indictor
        SepInd_.ref() = gammaSep;
        dimensionedScalar udimk("udimk", dimensionSet(0,-2,2,0,0,0,0), scalar(1.0));
        SepInd_ =  min( (SepInd_ * max((k - kInf*2)*udimk, scalar(0)))*1e6, scalar(1.0));
        volScalarField Rev_y = 1.0/y_ ;
        volVectorField gradRev_y = fvc::grad(Rev_y);
        volVectorField nor_gradRev_y = gradRev_y/mag(gradRev_y);
        surfaceScalarField yPhi
        (
        linearInterpolate(nor_gradRev_y) & this->mesh_.Sf()
        );
        surfaceScalarField yPhio(min(yPhi,scalar(0.0)*yPhi));
       dimensionedScalar udiml("udiml", dimensionSet(0,-1,0,0,0,0,0), scalar(1.0));
```

Implementation

Transition model

```
// Projection equation to the wall
tmp<fvScalarMatrix> SepIndEqn
(
    fvm::div(yPhi, SepInd_)
==
    SepInd_*scalar(100.0)*udiml
);

solve(SepIndEqn);
SepInd_ = min(SepInd_*scalar(1e16), scalar(1.0));
```

Then add the missing constructors for the newly added variables, i.e. SepInd_ and kInf; below the constructor of ReThetat_, add following sentences

```
SepInd_
(
    IOobject
    (
        IOobject::groupName("SepInd", alphaRhoPhi.group()),
        this->runTime_.timeName(),
        this->mesh_,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    this->mesh_
),
```

```
kInf
  (
    dimensioned<scalar>::lookupOrDefault
    (
        "kInf",
        this->coeffDict_,
        dimensionSet( 0, 2, -2, 0, 0, 0, 0),
        0.001
    )
 ),
```

And declare them in kOmegaSSTLMSep.H, after the declaration of ReThetat_:
volScalarField SepInd_;
volScalarField kInf;
Lastly compile the code.
cd $WM_PROJECT_USER_DIR/src/TurbulenceModels;
./Allwmake

Implementation

Cavitation model

```
cd $WM_PROJECT_USER_DIR/applications/solvers/multiphase
cd interPhaseChangeFoamMod/phaseChangeTwoPhaseMixtures/
cp -r SchnerrSauer SchnerrSauerSep; cd SchnerrSauerSep
mv *C SchnerrSauerSep.C; mv *H SchnerrSauerSep.H
sed -i s/SchnerrSauer/SchnerrSauerSep/g *
```

Implementation

Cavitation model

Inside the SchnerrSauerSep.C file, under Member Functions, below the function pCoeff, add

```
Foam::tmp<Foam::volScalarField>
Foam::phaseChangeTwoPhaseMixtures::SchnerrSauerSep::CvSep
(
    const volScalarField& sepind
) const
{
    return
    Cv_*sepind;
}
```

Also modify the sentences in function mDotAlphal(), replace

Cv_*(1.0 + alphaNuc() - limitedAlpha1)*pCoeff*min(p - pSat(), p0_) with:

CvSep*(1.0 + alphaNuc() - limitedAlpha1)*pCoeff*min(p - pSat(), p0_)

The same for the sentence in function mDotP(), replace

(-Cv_)*(1.0 + alphaNuc() - limitedAlpha1)*neg(p - pSat())*apCoeff with:

(-CvSep)*(1.0 + alphaNuc() - limitedAlpha1)*neg(p - pSat())*apCoeff

Implementation

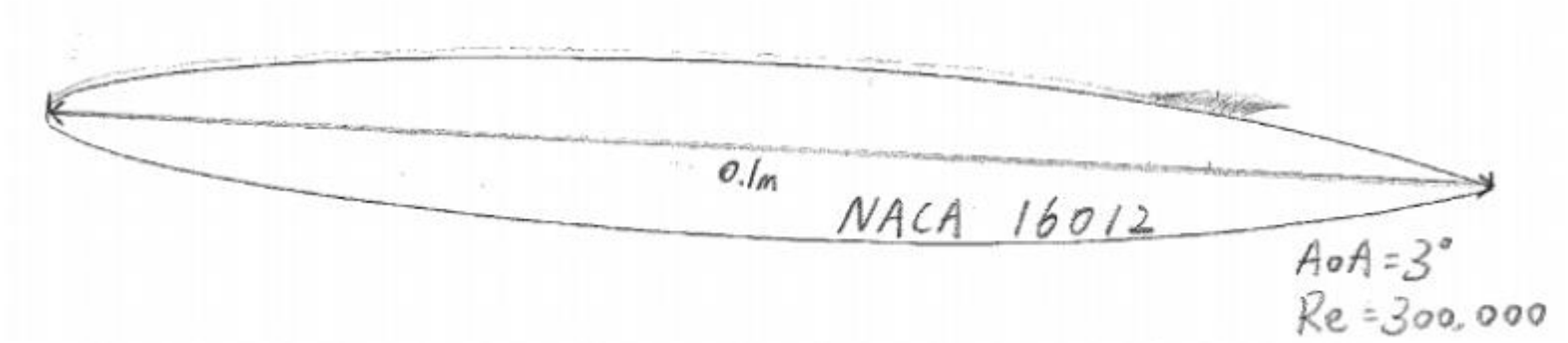
Cavitation model

```
cd $WM_PROJECT_USER_DIR/applications/solvers/multiphase/interPhaseChangeFoamMod
cd phaseChangeTwoPhaseMixtures
rm Make/files; touch Make/files
echo "phaseChangeTwoPhaseMixture/phaseChangeTwoPhaseMixture.C" > Make/files
echo "phaseChangeTwoPhaseMixture/newPhaseChangeTwoPhaseMixture.C" >> Make/files
echo "Kunz/Kunz.C " >> Make/files
echo "Merkle/Merkle.C" >> Make/files
echo "SchnerrSauer/SchnerrSauer.C" >> Make/files
echo "SchnerrSauerSep/SchnerrSauerSep.C" >>Make/files
echo "LIB = \$(FOAM_USER_LIBBIN)/libphaseChangeTwoPhaseMixtures" >> Make/files
cd ..; ./Allwmake
```

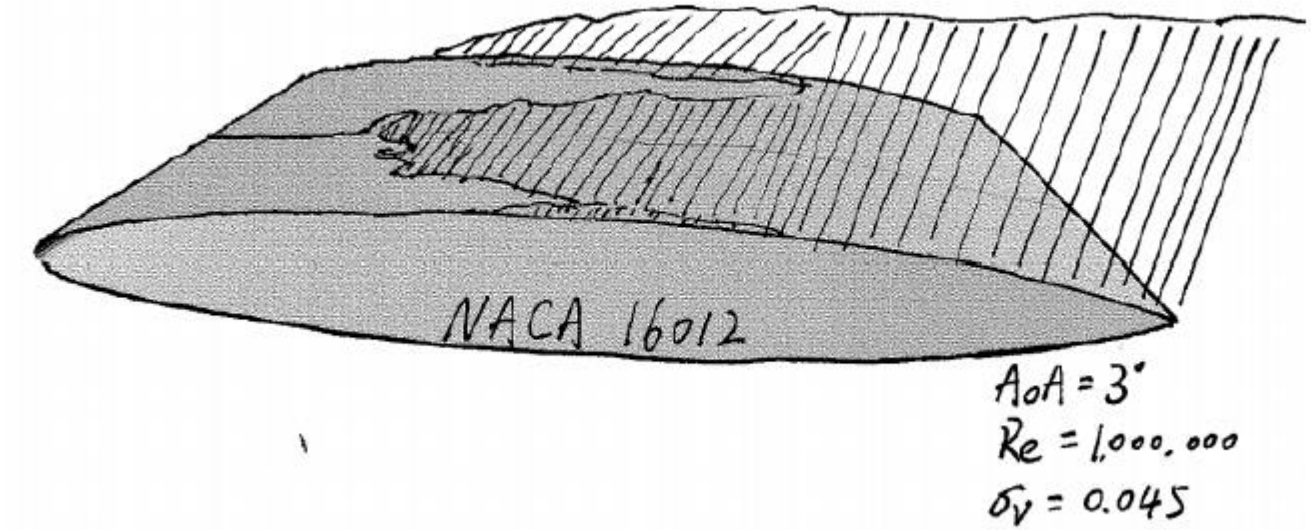
Test Case

- NACA 16012 hydrofoil
- Performed in 1985 by J. P. Franc and J. M. Michel.
- Dye injection at the leading edge for boundary layer visualization
- AoA = 3deg; Re = 300,000 (non-cavitating)
- AoA = 3deg; Re = 1,000,000; cavitation number 0.045 (cavitating)

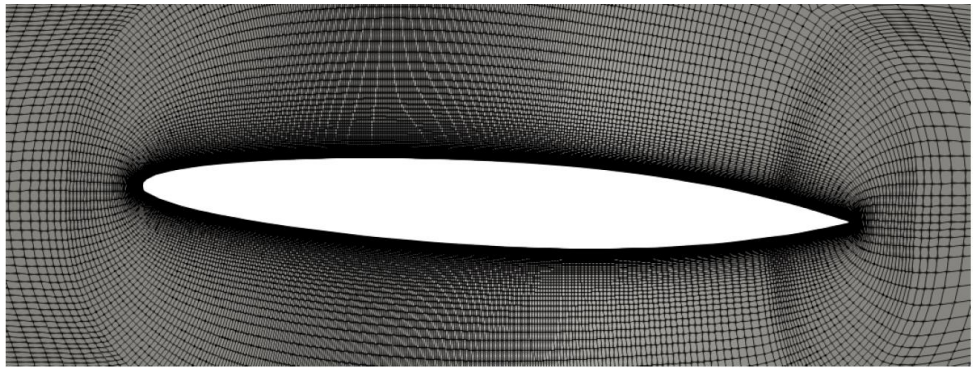
Test Case



Non-cavitating



Cavitating



Test Case

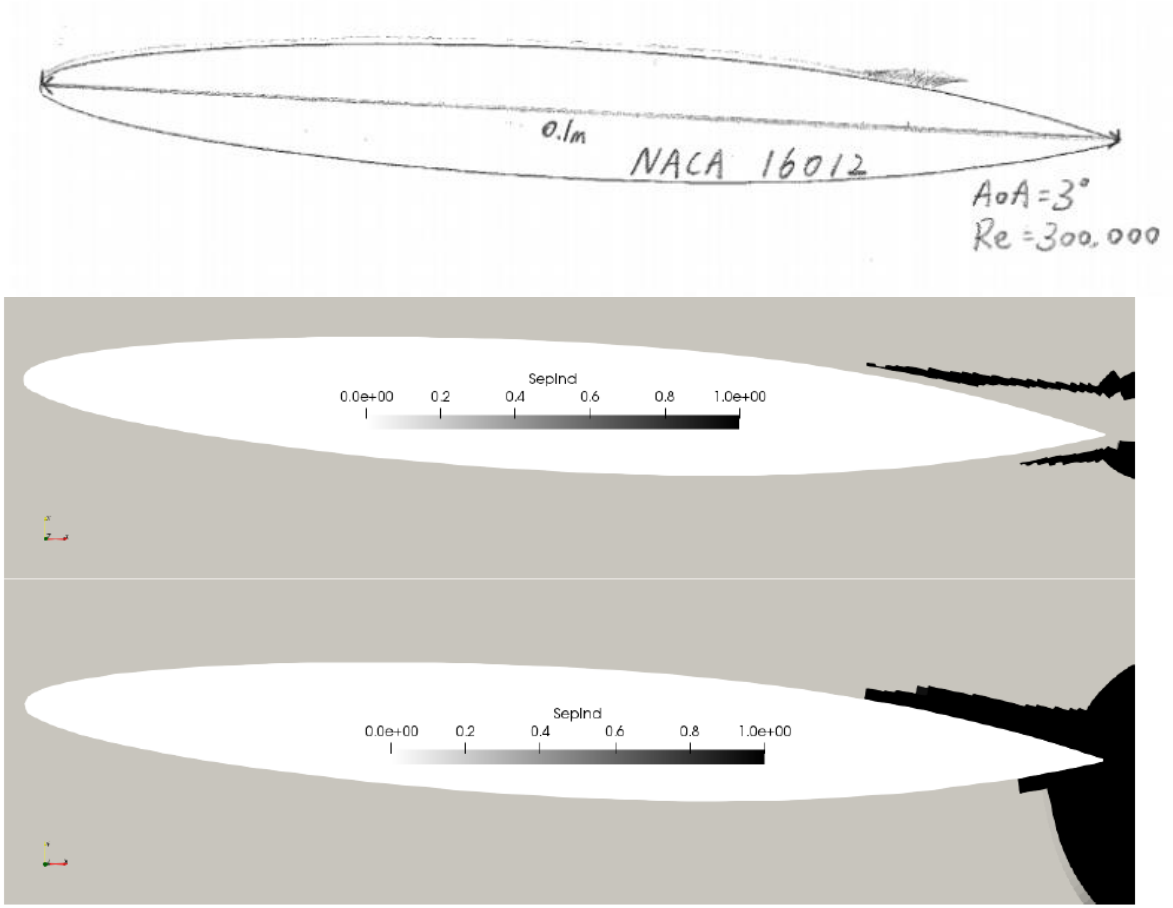


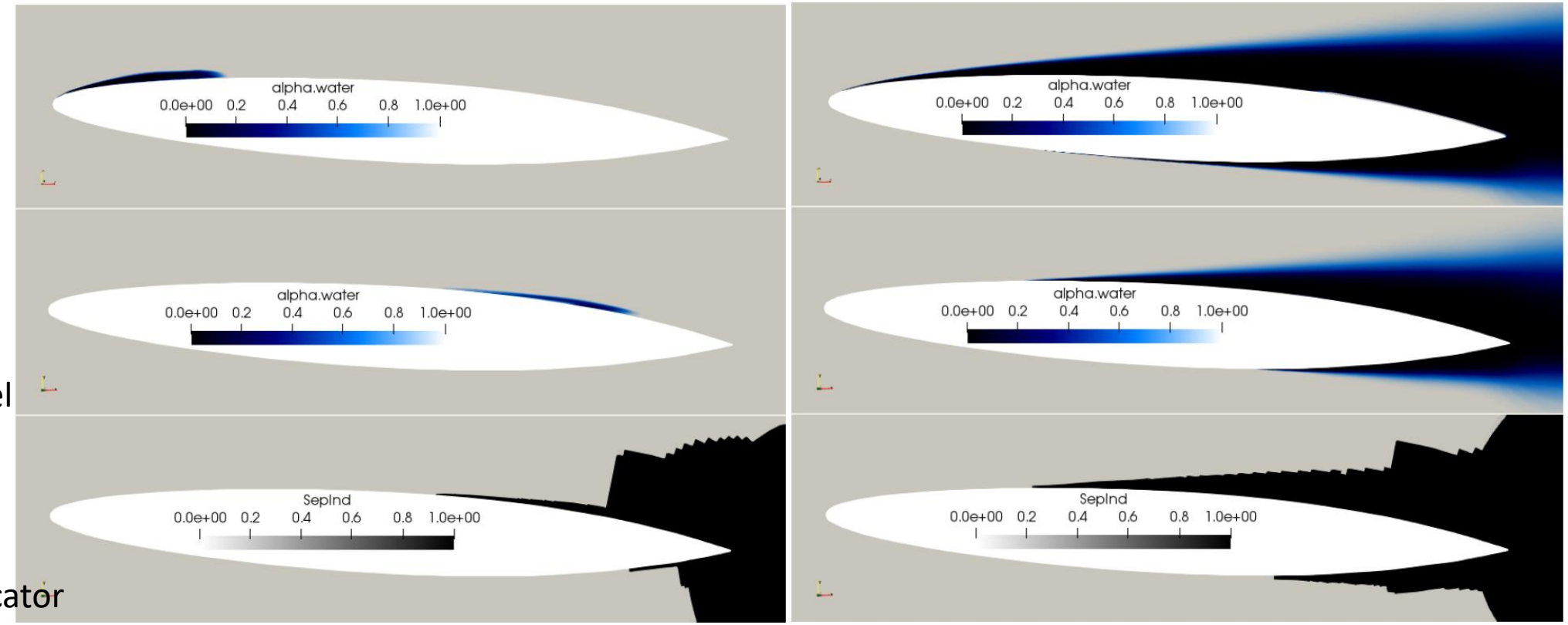
Figure 3.5: The laminar separation indicator, top: original predicted region; bottom: predicted region with modified code. Non-cavitation condition, Re = 300,000, AoA = 3deg.

Test Case

Original model

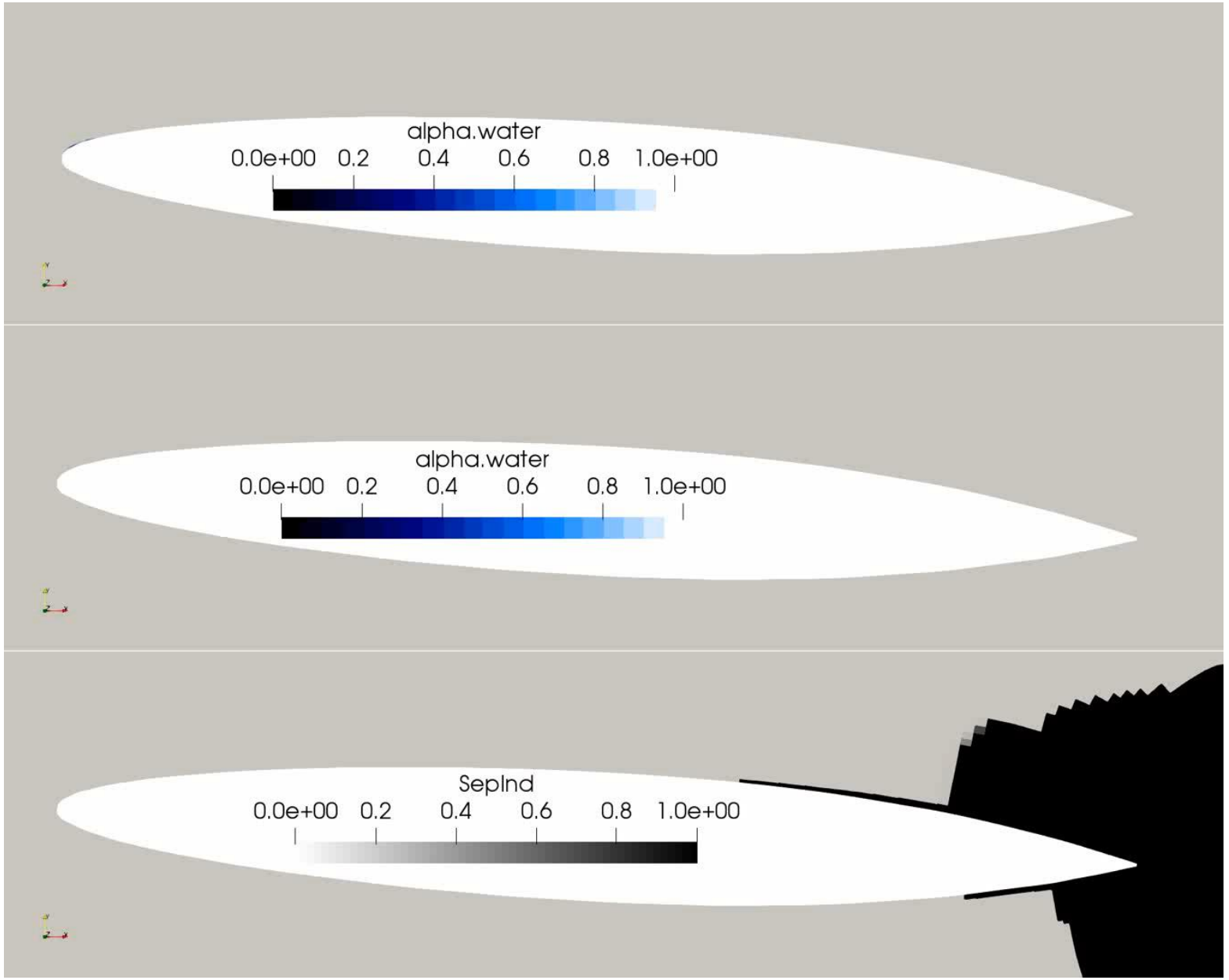
Modified model

Cavitation indicator



Time = 0.002s

Time = 0.1s



Original model

Modified model

Cavitation indicator

Thank you for your attention!