/\*---------------------------------------------------------------------------\*\

========= |

\\ / F ield | OpenFOAM: The Open Source CFD Toolbox

\\ / O peration |

\\ / A nd | Copyright (C) 2011-2013 OpenFOAM Foundation

\\/ M anipulation |

-------------------------------------------------------------------------------

License

This file is part of OpenFOAM.

OpenFOAM is free software: you can redistribute it and/or modify it

under the terms of the GNU General Public License as published by

the Free Software Foundation, either version 3 of the License, or

(at your option) any later version.

OpenFOAM is distributed in the hope that it will be useful, but WITHOUT

ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or

FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License

for more details.

You should have received a copy of the GNU General Public License

along with OpenFOAM. If not, see <http://www.gnu.org/licenses/>.

**my\_icoFoam に書き換える**

Application

icoFoam

Description

Transient solver for incompressible, laminar flow of Newtonian fluids.

\\*---------------------------------------------------------------------------\*/

#include "fvCFD.H"

// \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* //

int main(int argc, char \*argv[])

{

#include "setRootCase.H"

#include "createTime.H"

#include "createMesh.H"

#include "createFields.H"

#include "initContinuityErrs.H"

// \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* //

Info<< "\nStarting time loop\n" << endl;

while (runTime.loop())

{

Info<< "Time = " << runTime.timeName() << nl << endl;

#include "readPISOControls.H"

#include "CourantNo.H"

fvVectorMatrix UEqn

(

fvm::ddt(U)

Uの式

+ fvm::div(phi, U)

- fvm::laplacian(nu, U)

);

solve(UEqn == -fvc::grad(p));

// --- PISO loop

for (int corr=0; corr<nCorr; corr++)

{

volScalarField rAU(1.0/UEqn.A());

volVectorField HbyA("HbyA", U);

HbyA = rAU\*UEqn.H();

surfaceScalarField phiHbyA

(

"phiHbyA",

(fvc::interpolate(HbyA) & mesh.Sf())

+ fvc::interpolate(rAU)\*fvc::ddtCorr(U, phi)

);

adjustPhi(phiHbyA, U, p);

for (int nonOrth=0; nonOrth<=nNonOrthCorr; nonOrth++)

{

fvScalarMatrix pEqn

(

fvm::laplacian(rAU, p) == fvc::div(phiHbyA)

PISOループ

補正

);

pEqn.setReference(pRefCell, pRefValue);

pEqn.solve();

Tの式を追加

if (nonOrth == nNonOrthCorr)

{

**fvScalarMatrix TEqn**

**(**

**fvm::ddt(T)**

**+ fvm::div(phi, T)**

**- fvm::laplacian(DT, T)**

**);**

**TEqn.solve();**

phi = phiHbyA - pEqn.flux();

}

}

#include "continuityErrs.H"

U = HbyA - rAU\*fvc::grad(p);

U.correctBoundaryConditions();

}

runTime.write();

Info<< "ExecutionTime = " << runTime.elapsedCpuTime() << " s"

<< " ClockTime = " << runTime.elapsedClockTime() << " s"

<< nl << endl;

}

Info<< "End\n" << endl;

return 0;

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* //