

# HERCORE: Highly Efficient and quasi-Realistic CORonal and coronal mass Ejection modeling



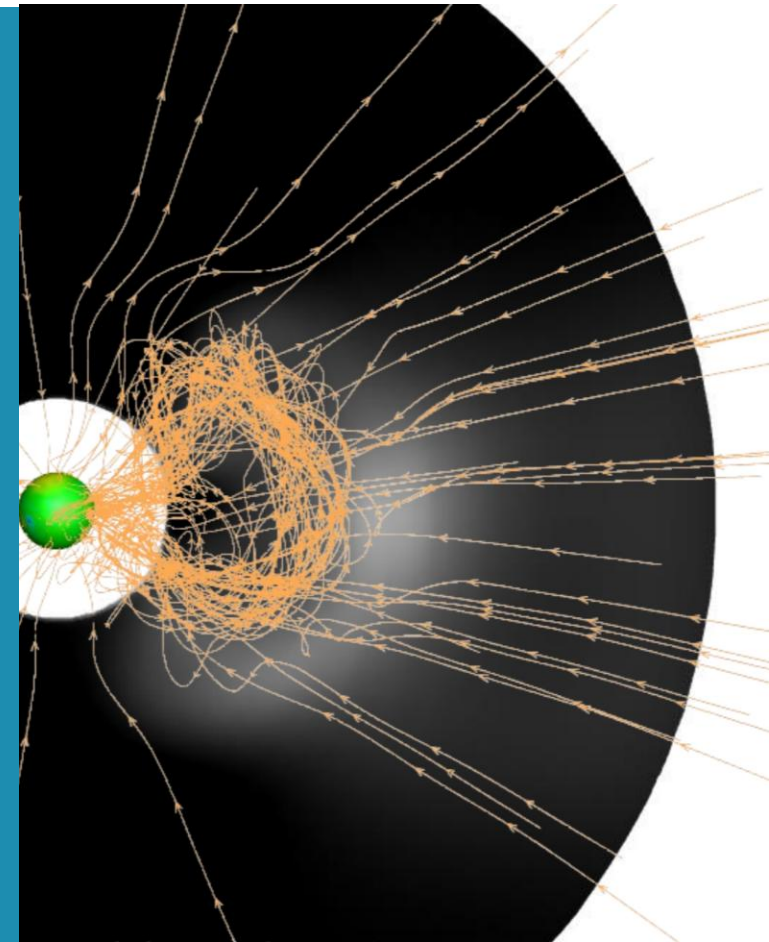
Dr. Haopeng Wang

*KU Leuven, 02/10/2025*

Postdoctoral Researcher, KU Leuven (Supervisor: Stefaan Poedts)

Ph.D., National Space Science Center, UCAS (Supervisor: Xueshang Feng)

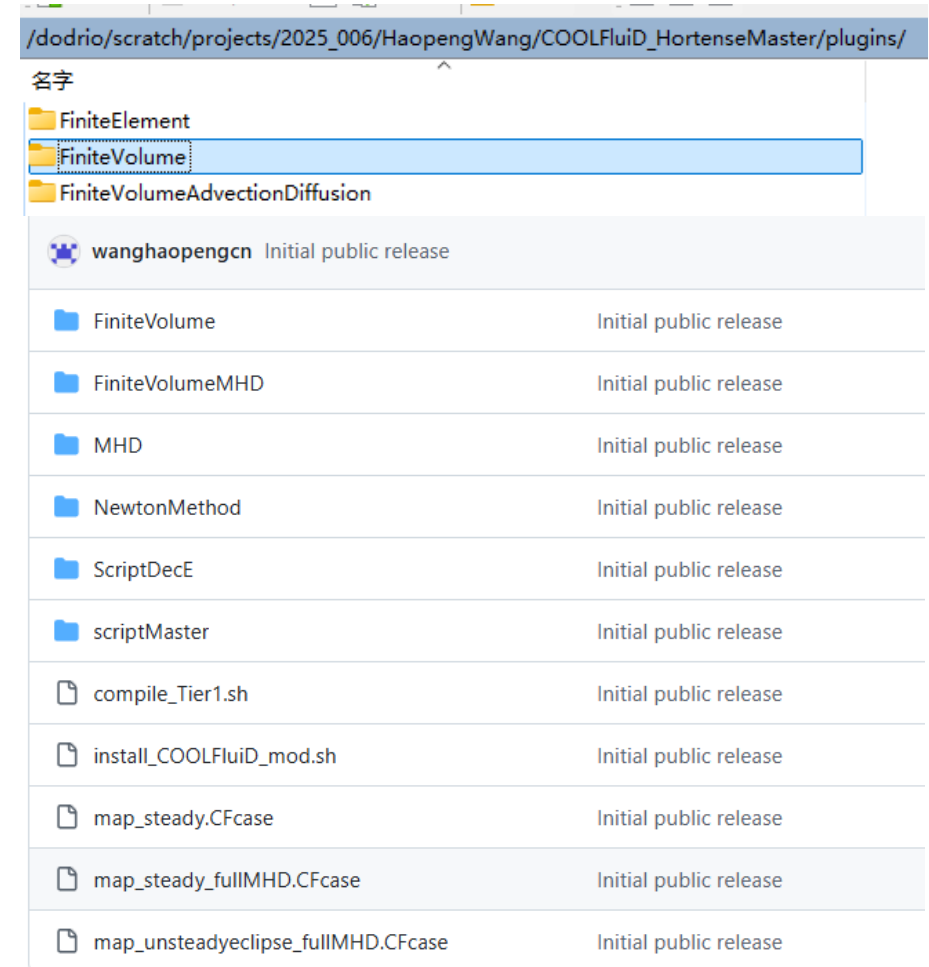
E-mail: [haopeng.wang1@kuleuven.be](mailto:haopeng.wang1@kuleuven.be)



# Installation of COCONUT

## From master version shared on GitHub

- All source code files, job submission scripts, and installation and recompilation scripts are shared via GitHub or OneDrive. You can directly upload the files from the *first four folders* into the corresponding directories in the *plugins* folder of the Master version of COCONUT you just installed.
- Before recompile, you need to open the already replaced files, add a space in any blank area, and then save them. This will ensure the replaced files are recognized as modified and will be recompiled.
- *ScriptDecE* and *scriptMaster* contain job submission scripts corresponds to COCONUT adopting decomposed and full energy equations.
- **Shared via GitHub:** <https://github.com/wanghaopengcn/Start-from-COCONUT-HW>: This repository shares several modifications made to the coronal model COCONUT.
- **Shared via OneDrive:**  
[Start\\_from\\_Master\\_Version\\_COCONUT](#)



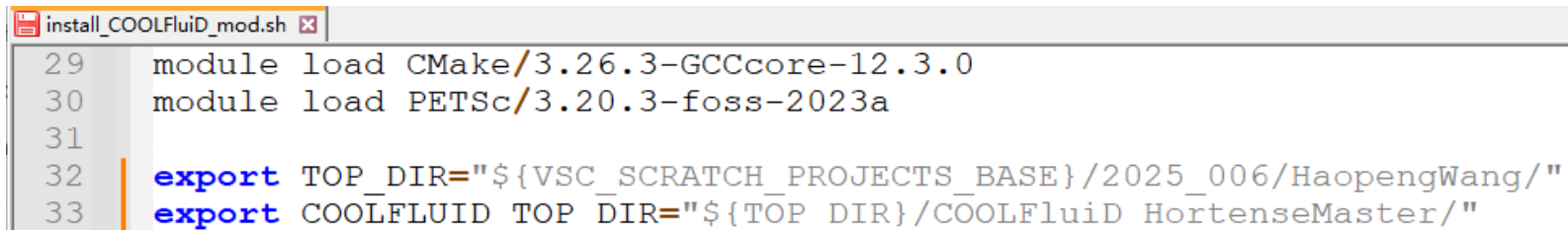
/dodrio/scratch/projects/2025_006/HaopengWang/COOLFluid_HortenseMaster/plugins/	
名字	
FiniteElement	
FiniteVolume	
FiniteVolumeAdvectionDiffusion	
wanghaopengcn Initial public release	
FiniteVolume	Initial public release
FiniteVolumeMHD	Initial public release
MHD	Initial public release
NewtonMethod	Initial public release
ScriptDecE	Initial public release
scriptMaster	Initial public release
compile_Tier1.sh	Initial public release
install_COOLFluid_mod.sh	Initial public release
map_steady.CFcase	Initial public release
map_steady_fullIMHD.CFcase	Initial public release
map_unsteadyeclipse_fullIMHD.CFcase	Initial public release

# Installation of COCONUT

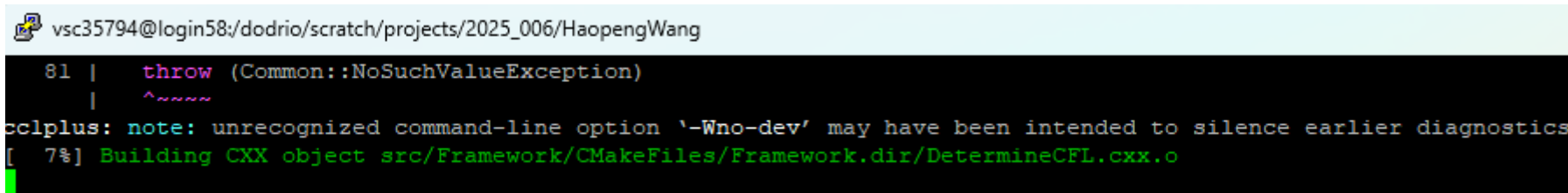
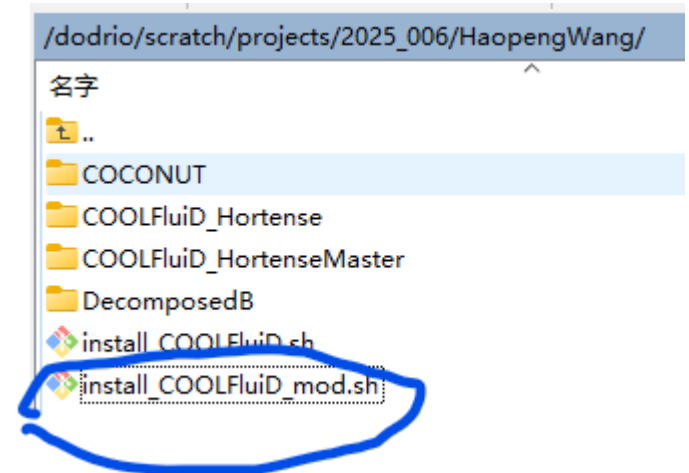
## From master version shared on GitHub

### *Installation of master version COCONUT on Tier1*

1. Modify the path definition of “TOP\_DIR” and “COOLFLUID\_TOP\_DIR” in the script “install\_COOLFluid\_mod.sh” according to your case.
2. Via permission of visiting the installation script and execute installation:  
`chmod a+x ./install_COOLFluid_mod.sh`  
`./install_COOLFluid_mod.sh DEBUG NOCUDA --download=2`



```
install_COOLFluid_mod.sh x
29 module load CMake/3.26.3-GCCcore-12.3.0
30 module load PETSc/3.20.3-foss-2023a
31
32 export TOP_DIR="${VSC_SCRATCH_PROJECTS_BASE}/2025_006/HaopengWang/"
33 export COOLFLUID_TOP_DIR="${TOP_DIR}/COOLFluid_HortenseMaster/"
```



```
vsc35794@login58:/dodrio/scratch/projects/2025_006/HaopengWang
81 | throw (Common::NoSuchValueException)
    | ^~~~~
ccplus: note: unrecognized command-line option '-Wno-dev' may have been intended to silence earlier diagnostics
[ 7%] Building CXX object src/Framework/CMakeFiles/Framework.dir/DetermineCFL.cxx.o
```

# Installation of COCONUT

## From master version shared on GitHub

*Revision of the master version shared on GitHub*

### Venktn3DStrict.hh

- Add `std::vector<CFuint> _NoLimiterID;` (Variables which don't implement limiter)

### Venktn3DStrict.cxx

- Add the following snippet to abandon limiter for the selected variables.

```
if (_NoLimiterID.size() > 0){  
for (CFuint iVar = 0; iVar < _NoLimiterID.size(); ++iVar){  
limiterValue[_NoLimiterID[iVar]] = 1.0;  
}  
}
```

```
Venktn3DStrict.cxx  MHD3DConsToPrimElnRef.hh  MH  
options.addConfigOption< CFreal >  
("strictCoeff", "Fix for smooth flow region.");  
options.addConfigOption< bool >  
("psiMinEqual1", "impose psimin = 1 for certain v  
options.addConfigOption< vector<CFuint> >  
("NoLimiterID", "Used to cancel limiter.");  
}
```

```
Venktn3DStrict.hh  Venktn3DStrict.cxx  Venktn3DStrict.hh  Venktn3DStrict.cxx  S  
limiterValue[iVar] = psimin;  
  
const CFreal maxAllowableLimiterFunctionValue = 1.094;  
if (limiterValue[iVar] > maxAllowableLimiterFunctionValue) {  
CFout << "wrong limiterValue = " << limiterValue[iVar] << "\n";  
}  
  
//>> Mark 2025.07.30 by Hp to cancel limiter for several reconstructed variables  
if (_NoLimiterID.size() > 0){  
for (CFuint iVar = 0; iVar < _NoLimiterID.size(); ++iVar){  
limiterValue[_NoLimiterID[iVar]] = 1.0;  
}  
}  
  
//<< Mark 2025.07.30 by Hp to cancel limiter for several reconstructed variables  
}
```

```
Venktn3DStrict.hh  Venktn3DStrict.cxx  map_u  
Venktn3DStrict::Venktn3DStrict(const std::str  
Venktn2D(name),  
socket_uZ("uZ")  
{  
addConfigOptionsTo(this);  
  
_strictCoeff = 1.0;  
setParameter("strictCoeff",&_strictCoeff);  
  
_psiMinEqual1 = false;  
setParameter("psiMinEqual1",&_psiMinEqual1)  
  
_NoLimiterID = vector<CFuint>();  
setParameter("NoLimiterID", &_NoLimiterID);  
}
```

```
Venktn3DStrict.hh*  Venktn3DStrict.cxx  V  
protected:  
CFreal _strictCoeff;  
  
bool _psiMinEqual1;  
  
/// corresponding to variables which do  
std::vector<CFuint> _NoLimiterID;
```

# Installation of COCONUT

## From master version shared on GitHub

*Revision of the master version shared on GitHub*

### StdUpdateSolPP.hh

- Add CFreal \_pBC; & CFreal \_rhoBC; (It corresponds to boundary density and pressure)

### StdUpdateSolPP.cxx

- Add the following snippet to define boundary pressure and density via .Cfcase file.  
options.addConfigOption< CFreal >("pressureBoundaryValue", "the boundary-pressure value for the pressure.");  
options.addConfigOption< CFreal >("densityBoundaryValue", "the boundary-density value for the pressure.");
- Comment out CFreal \_rhoBC=2.0; & CFreal \_pBC = 0.25801090625;

### CMakeLists.txt

Move “StdUpdateSolPP.hh” & “StdUpdateSolPP.cxx” from “LIST ( APPEND NewtonMethodMHD\_files” to “LIST ( APPEND NewtonMethod\_files”

```
StdUpdateSolPP.hh  StdUpdateSolPP.cxx*  + X
void StdUpdateSolPP::defineConfigOptions(Config::OptionList& options)
{
    options.addConfigOption<vector<CFreal>,Config::DynamicOption<>>("Relaxation", "Relaxation factor");
    options.addConfigOption< bool >("Validate", "Check that each update creates variables with physical meaning");
    options.addConfigOption< CFreal >("pressureBoundaryValue", "the boundary-pressure value for the pressure.");
    options.addConfigOption< CFreal >("densityBoundaryValue", "the boundary-density value for the pressure.");
}
```

```
StdUpdateSolPP.hh  StdUpdateSolPP.cxx  + X
//CFreal _rhoBC=2.0;
//CFreal _pBC = 0.25801090625;
CFreal Bmagmax=1.0;

StdUpdateSolPP.hh*  + X  StdUpdate
bool m_validate;

/// boundary pressure val
CFreal _pBC;
/// boundary plasma densi
CFreal _rhoBC;

}; // class StdUpdateSolPP
```

```
StdUpdateSolPP.hh  StdUpdateSolPP.cxx*  + X
StdUpdateSolPP::StdUpdateSolPP(const std::string& name)
    NewtonIteratorCom(name),
    socket_states("states"),
    socket_rhs("rhs"),
    socket_updateCoeff("updateCoeff"),
    socket_invalidStates("invalidStates")
{
    addConfigOptionsTo(this);

    m_alpha = vector<CFreal>();
    setParameter("Relaxation", &m_alpha);

    m_validate = false;
    setParameter("Validate", &m_validate);

    _pBC = MathTools::MathConsts::CFrealEps();
    setParameter("pressureBoundaryValue", &_pBC);

    _rhoBC = MathTools::MathConsts::CFrealEps();
    setParameter("densityBoundaryValue", &_rhoBC);
}
```

# Installation of COCONUT

## From master version shared on GitHub

*Recompile the master version COCONUT on Tier1*

1. Modify the path definition of “TOP\_DIR” and “COOLFLUID\_TOP\_DIR” in the script “compile\_Tier1.sh” according to your case.
2. Go to the OPENMPI directory where you upload the recompile script “compile\_Tier1.sh” and execute recompile via:
  - sh compile\_Tier1.sh

```
compile_Tier1.sh
1 module load CMake/3.26.3-GCCcore-12.3.0
2 module load PETSc/3.20.3-foss-2023a
3 export TOP_DIR="/readonly${VSC_SCRATCH_PROJECTS_BASE}/2025_006/HaopengWang/"
4 export COOLFLUID_TOP_DIR="${TOP_DIR}/COOLFluid_HortenseMaster"
5
6 export BUILD_MODE=optim
7 export CONF_FILE="COOLFluid_Hortense_nocuda.conf"
8
9
10 export COOLFLUID_BASEBUILD_DIR="${COOLFLUID_TOP_DIR}/OPENMPI"
11 export COOLFLUID_CONF_FILE="${COOLFLUID_TOP_DIR}/${CONF_FILE}"
12 export COOLFLUID_INSTALL_DIR="${COOLFLUID_BASEBUILD_DIR}/${BUILD_MODE}/INSTALL"
13 export ALL_ACTIVE=1
14
15 cd $COOLFLUID_TOP_DIR
16 #./prepare.pl --config-file=${COOLFLUID_CONF_FILE} --build=${BUILD_MODE}
17
18 cd ${COOLFLUID_BASEBUILD_DIR}/${BUILD_MODE}
19 make -j 4
20 #make install
```

```
[vsc35794@login58 ~]$ cd /dodrio/scratch/projects/2025_006/HaopengWang/COOLFluid_HortenseMaster/OPENMPI/
[vsc35794@login58 OPENMPI]$ sh compile_Tier1.sh
```

# Installation of COCONUT

## From master version shared on GitHub

*1<sup>st</sup> step: Polytropic quasi-steady-state simulation on Tier1*

### Revise the following configuration in `map_steady.Cfcase`

- `Simulator.Paths.ResultsDir = ./results-map-res/30Rs_lvl6/steady/polytropic/ADAPT2008/lmax25`
- `Simulator.SubSystem.EM.Data.DistanceBased.FileNameTw= ./MapData/ADAPT2008Lmax25/map_adapt_lmax25_20080803140000.dat`
- `Simulator.SubSystem.Flow.Data.DistanceBased.FileNameTw = ./MapData/ADAPT2008Lmax25/map_adapt_lmax25_20080803140000.dat`
- `Simulator.SubSystem.Flow.Data.Venktn3DStrict.NoLimiterID = 1 2 3 4 5 6`
- `Simulator.SubSystem.CFmesh1.FileName = corona_NoLim_PPDecEBC1_20080803ADAPT.Cfmesh`
- `Simulator.SubSystem.Tecplot1.FileName = corona_NoLim_PPDecEBC1_20080803ADAPT.plt`
- `Simulator.SubSystem.FlowIterator.StdUpdateSolPP.pressureBoundaryValue = 0.108 #0.25801090625`  
`Simulator.SubSystem.FlowIterator.StdUpdateSolPP.densityBoundaryValue = 1.0 #2.0`
- The following boundary conditions should be revised according to your cases.  
`Simulator.SubSystem.Flow.Jet1.rhoBC = 1.0 #1.0`  
`Simulator.SubSystem.Flow.Jet1.pBC = 0.108 #0.00284615174 #0.08592156224 #0.108`  
`Simulator.SubSystem.Flow.InField.Def = \`

### Create a symbolic link to the path of the working folder

- `ln -sf /dodrio/scratch/projects/2025_006/CompleteCopyTest/COOLFluid_Hortense/OPENMPI/optim/apps/Solver/coolfluid-solver* ./`

### Submit a job

- `qsub run_Quasi_steady_Tier1.pbs`

# Installation of COCONUT

## From master version shared on GitHub

*2<sup>nd</sup> step: Restart full MHD quasi-steady-state simulation from the polytropic quasi-steady-state simulation results on Tier1*

### Revise the following configuration in `map_steady_fullMHD.Cfcase`

- `Simulator.Paths.ResultsDir = ./results-map-res/30Rs_1vl6/steady/fullMHD/ADAPT2008/lmax25`
- `Simulator.SubSystem.CFmeshFileReader0.Data.FileName = ./results-map-res/30Rs_1vl6/steady/polytropic/ADAPT2008/lmax25/corona_NoLim_PPDecEBC1_20080803ADAPT.Cfmesh`
- `Simulator.SubSystem.CFmeshFileReader1.Data.FileName = ./results-map-res/30Rs_1vl6/steady/polytropic/ADAPT2008/lmax25/corona_NoLim_PPDecEBC1_20080803ADAPT.Cfmesh`
- `Simulator.SubSystem.Flow.Restart = true`
- The other revisions are like those modifications in `map_steady.Cfcase`

### Submit a job

- `qsub run_Quasi_steadyfullMHD_Tier1.pbs`

```
cd /dodrio/scratch/projects/2025_006/HaopengWang/COCONUT/
```

```
##module swap cluster/dodrio/cpu_rome_512
```

```
module load CMake/3.26.3-GCCcore-12.3.0
```

```
module load PETSc/3.20.3-foss-2023a
```

```
module load vsc-mypirun
```

```
mympirun --universe 360 ./coolfuid-solver --scase /dodrio/
```

# Installation of COCONUT

## From master version shared on GitHub

*3<sup>rd</sup> step: Restart time-evolving coronal simulation from full MHD quasi-steady-state simulation results on Tier1*

**Pay attention to the following modification in *map\_steady\_fullMHD.Cfcase***

- `Simulator.SubSystem.FlowSubSystemStatus.TimeStep = 0.207297` # Corresponds to 5 minutes
- `Simulator.SubSystem.InitialTime = 0.0` # In code hour unite, the restart physical time `*3600.0/1447.2`
- `Simulator.SubSystem.MaxTime.maxTime = 1641.79` # End of the physical time `*3600.0/1447.2`
- `Simulator.SubSystem.Flow.Data.DistanceBased.FileNameTwADF = ./MapData/zqsCR2296Lmax25/map_gong_lmax25_2025 0400.dat 663 1 2025 32919 32918` # Require to be revised according to your cases.
- `Simulator.SubSystem.Flow.Data.Venktn3DStrict.NoLimiterID = 1 2 3 4 5 6` # variables which don't implement limiter
- `Simulator.SubSystem.Flow.Data.HLL.AddLax = true` # true for decomposed energy equation
- The other revisions are like those modifications in *map\_steady.Cfcase*
- Comment out *Simulator.SubSystem.Flow.Jet1.VarIDs = 0* will trun the time-evolving regime to quasi-steady-state regime.
- Pay attention to comments starting with # Decomposed energy
- Also refer to the papers and manuals contained in the **eDocuments** folder for further reference.

### Submit a job

- `qsub run_TimeevolvingfullMHD_Tier1.pbs`

# Installation of COCONUT

## From master version shared on GitHub

*Calculate source terms as Haopeng Wang did and considering decomposed energy equation.*

### MHDConsACAHSWSorceTerm.cxx

- Add the following snippet for decomposed energy equation

### .Cfcase

- Switching to “Simulator.SubSystem.Flow.Data.SourceTerm = MHDConsACAHSWSorceTerm” to calculate source terms as Haopeng Wang did.
- Adding “Simulator.SubSystem.Flow.Data.MHDConsACAHSWSorceTerm.deCompEorNot = 0” means doesn’t calculate the source term caused by decomposed energy equation.

```
onsACAHSWSorceTerm.hh  MHDConsACAHSWSorceTerm.cxx*  map_steady.Cfcase
options.addConfigOption< CFint >("RadiativeLossTerm", "Switch on optically thin approximation for radiation loss
options.addConfigOption< CFint >("deCompEorNot", "Calculate source term caused by decomposed energy equation.")
```

```
HDConsACAHSWSorceTerm.hh  MHDConsACAHSWSorceTerm.cxx  map_steady.Cfcase
setParameter("RadiativeLossTerm", &_RadiativeLossTerm);
_deCompE = 0;
setParameter("deCompEorNot", &_deCompE);
```

```
MHDConsACAHSWSorceTerm.hh  map_steady.Cfcase
CFint _wave_pressure;
CFint _deCompE
```

```
onsACAHSWSorceTerm.hh  MHDConsACAHSWSorceTerm.cxx  map_steady.Cfcase
}

//>> energy Source terms caused by decomposed energy equation
//CFuint deCompE = 1;
CFreal Q_deCompE = 0.0;
//if (deCompE == 1 && r_dimless>1.05){
if (_deCompE == 1){
    const CFuint BXID = 4;
    const CFuint BYID = 5;
    const CFuint BZID = 6;
    const CFuint gradBXID = elementID*nbEqs + BXID;
    const CFuint gradBYID = elementID*nbEqs + BYID;
    const CFuint gradBZID = elementID*nbEqs + BZID;
    _gradBx[XX] = this->m_ux[gradBXID];
    _gradBx[YY] = this->m_uy[gradBXID];
    _gradBx[ZZ] = this->m_uz[gradBXID];
    _gradBy[XX] = this->m_ux[gradBYID];
    _gradBy[YY] = this->m_uy[gradBYID];
    _gradBy[ZZ] = this->m_uz[gradBYID];
    _gradBz[XX] = this->m_ux[gradBZID];
    _gradBz[YY] = this->m_uy[gradBZID];
    _gradBz[ZZ] = this->m_uz[gradBZID];
    std::vector<CFreal> BdotLambaB(3, 0.0);
    std::vector<CFreal> VdotLambaB(3, 0.0);
    VdotLambaB[0] = Vx*_gradBx[XX] + Vy*_gradBx[YY] + Vz*_gradBx[ZZ];
    VdotLambaB[1] = Vx*_gradBy[XX] + Vy*_gradBy[YY] + Vz*_gradBy[ZZ];
    VdotLambaB[2] = Vx*_gradBz[XX] + Vy*_gradBz[YY] + Vz*_gradBz[ZZ];
    BdotLambaB[0] = (Bx*_gradBx[XX] + By*_gradBx[YY] + Bz*_gradBx[ZZ]) / B0;
    BdotLambaB[1] = (Bx*_gradBy[XX] + By*_gradBy[YY] + Bz*_gradBy[ZZ]) / B0;
    BdotLambaB[2] = (Bx*_gradBz[XX] + By*_gradBz[YY] + Bz*_gradBz[ZZ]) / B0;
    Q_deCompE = Vx*BdotLambaB[0] + Vy*BdotLambaB[1] + Vz*BdotLambaB[2] -
    (Bx*VdotLambaB[0] + By*VdotLambaB[1] + Bz*VdotLambaB[2]) / B0;
    source[7] += Q_deCompE*volumes[elementID];
}

//<< energy Source terms caused by decomposed energy equation
```

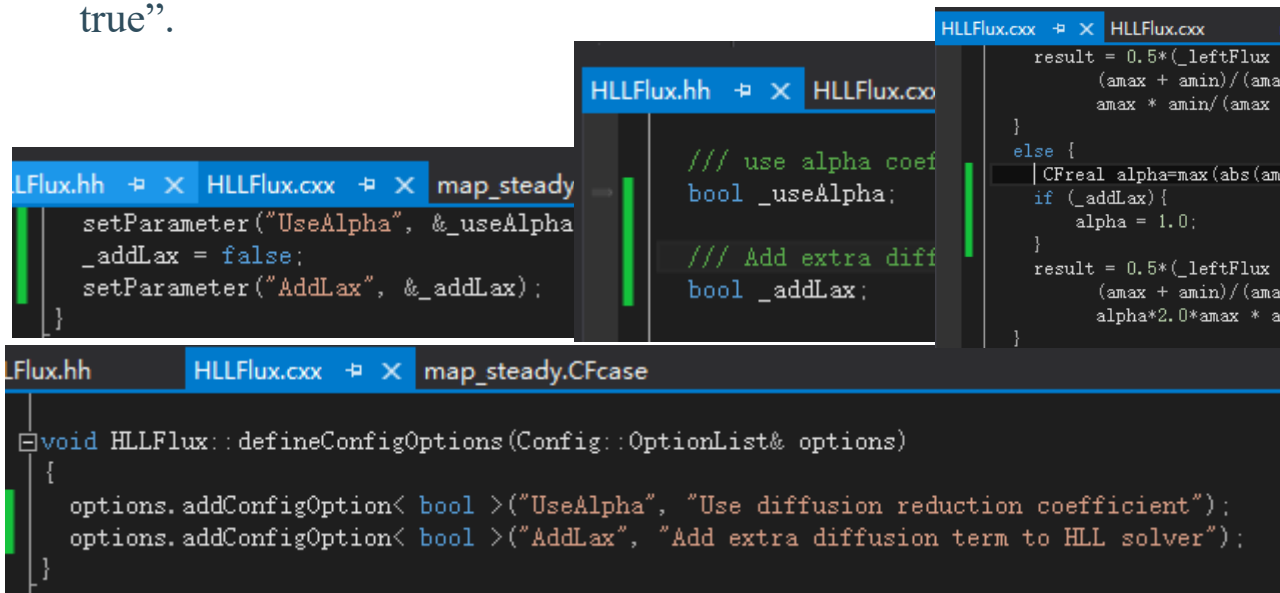
# Installation of COCONUT

## From master version shared on GitHub

*Considering extra dissipation term for decomposed energy equation.*

### HLLFlux.cxx & HLLFlux.hh

- Add the following snippet to add extra diffusion term to HLL solver for decomposed energy equation
- Adding “Simulator.SubSystem.Flow.Data.HLL.AddLax = false” to .Cfcase means doesn’t consider the extra dissipation term.
- Always keep “Simulator.SubSystem.Flow.Data.HLL.UseAlpha = true”.



The screenshot shows three code editors. The top editor, 'map\_steady.Cfcase', contains the following code:

```
setParameter("UseAlpha", &_useAlpha);
_addLax = false;
setParameter("AddLax", &_addLax);
```

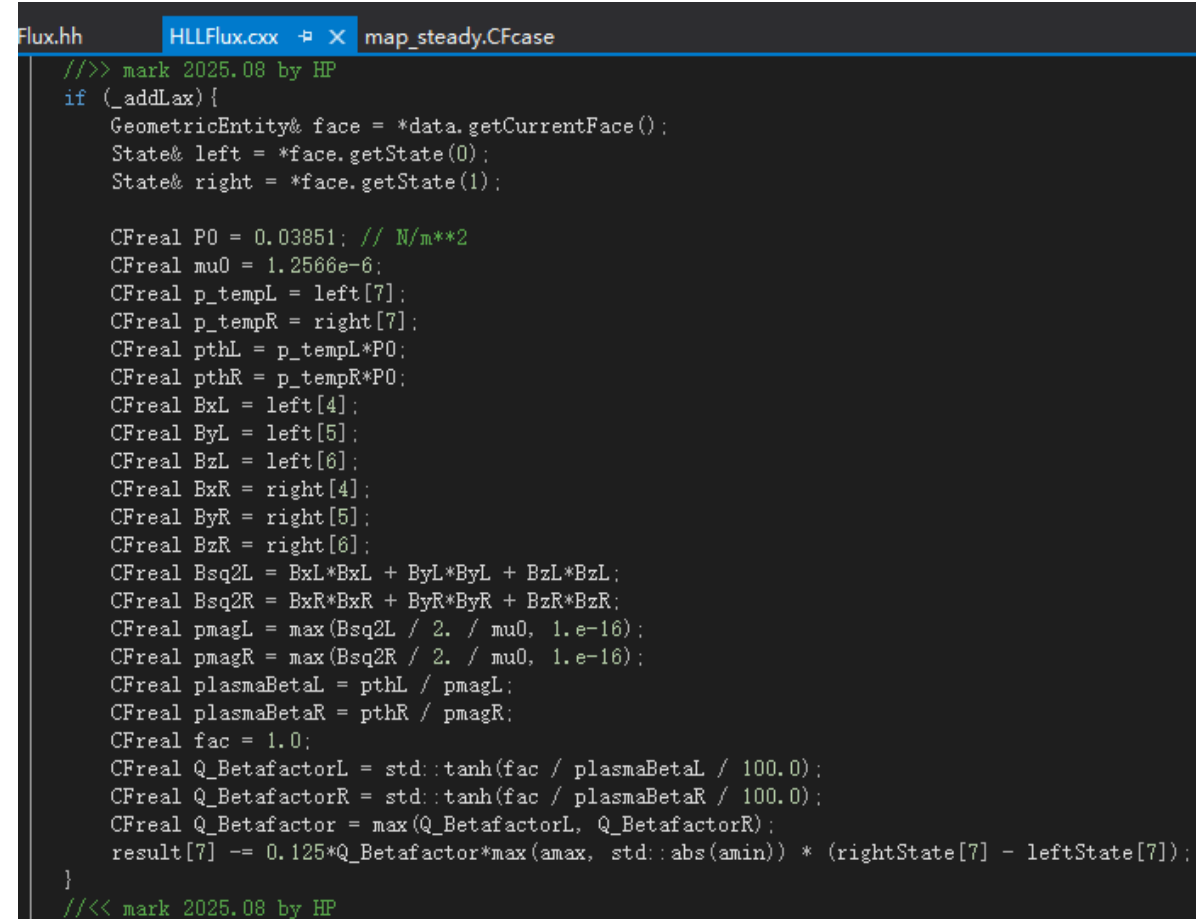
The middle editor, 'HLLFlux.cxx', shows the following code:

```
/// use alpha coef
bool _useAlpha;

/// Add extra diff
bool _addLax;
```

The bottom editor, 'HLLFlux.cxx', shows the following code:

```
void HLLFlux::defineConfigOptions(Config::OptionList& options)
{
    options.addConfigOption< bool >("UseAlpha", "Use diffusion reduction coefficient");
    options.addConfigOption< bool >("AddLax", "Add extra diffusion term to HLL solver");
}
```



The screenshot shows the 'HLLFlux.cxx' file with the following code:

```
if (_addLax) {
    GeometricEntity& face = *data.getCurrentFace();
    State& left = *face.getState(0);
    State& right = *face.getState(1);

    CFreal P0 = 0.03851; // N/m**2
    CFreal mu0 = 1.2566e-6;
    CFreal p_tempL = left[7];
    CFreal p_tempR = right[7];
    CFreal pthL = p_tempL*P0;
    CFreal pthR = p_tempR*P0;
    CFreal BxL = left[4];
    CFreal ByL = left[5];
    CFreal BzL = left[6];
    CFreal BxR = right[4];
    CFreal ByR = right[5];
    CFreal BzR = right[6];
    CFreal BsqrL = BxL*BxL + ByL*ByL + BzL*BzL;
    CFreal BsqrR = BxR*BxR + ByR*ByR + BzR*BzR;
    CFreal pmagL = max(BsqrL / 2. / mu0, 1.e-16);
    CFreal pmagR = max(BsqrR / 2. / mu0, 1.e-16);
    CFreal plasmaBetaL = pthL / pmagL;
    CFreal plasmaBetaR = pthR / pmagR;
    CFreal fac = 1.0;
    CFreal Q_BetafactorL = std::tanh(fac / plasmaBetaL / 100.0);
    CFreal Q_BetafactorR = std::tanh(fac / plasmaBetaR / 100.0);
    CFreal Q_Betafactor = max(Q_BetafactorL, Q_BetafactorR);
    result[7] -= 0.125*Q_Betafactor*max(amax, std::abs(amin)) * (rightState[7] - leftState[7]);
}
```

# Installation of COCONUT

## From master version shared on GitHub

*Considering decomposed energy equation.*

### MHD3DProjectionPrimE.cxx

- Correct “\_fluxArray[7] = Vn\*(E + P);” to “\_fluxArray[7] = Vn\*(E + p);” for decomposed energy equation

### MHD3DConsToPrimEInRef.xx

- Modify “MHD3DConsToPrimInRef.xx” to “MHD3DConsToPrimEInRef.xx”

```
MHD3DProjectionPrimE.cxx* MHD3DProjectionDiffVarSet.cxx MHD3
// Haopeng: check this
_fluxArray[0] = Vn*rho;
_fluxArray[1] = Vn*rho*u - Bn*Bx + P*nx;
_fluxArray[2] = Vn*rho*v - Bn*By + P*ny;
_fluxArray[3] = Vn*rho*w - Bn*Bz + P*nz;
_fluxArray[4] = (v*Bx - By*u)*ny + (w*Bx - Bz*u)*nz + phi*nx;
_fluxArray[5] = (u*By - Bx*v)*nx + (w*By - Bz*v)*nz + phi*ny;
_fluxArray[6] = (u*Bz - Bx*w)*nx + (v*Bz - By*w)*ny + phi*nz;
_fluxArray[7] = Vn*(E + p);
_fluxArray[8] = refSpeedSq*Bn;
}
```

```
MHD3DConsToPrimEInRef.hh MHD3DConsToPrimEInRef.cxx map_unst
void MHD3DConsToPrimEInRef::setMatrixFromRef()
{
    cf_assert(!_model.isNull());

    const RealVector& linearData = _model->getPhysicalData();

    const CFreal ovrbar = 1./linearData[MHDTerm::RHO];
    const CFreal gm1 = _model->getGamma() - 1.;
    const CFreal ubar = linearData[MHDTerm::VX];
    const CFreal vbar = linearData[MHDTerm::VY];
    const CFreal wbar = linearData[MHDTerm::VZ];
    const CFreal Exbar = linearData[MHDTerm::EX];
    const CFreal Bybar = linearData[MHDTerm::BY];
    const CFreal Bzbar = linearData[MHDTerm::BZ];

    _transMatrix(0,0) = 1.;
    _transMatrix(1,0) = -ubar*ovrbar;
    _transMatrix(1,1) = ovrbar;
    _transMatrix(2,0) = -vbar*ovrbar;
    _transMatrix(2,2) = ovrbar;
    _transMatrix(3,0) = -wbar*ovrbar;
    _transMatrix(3,3) = ovrbar;
    _transMatrix(4,4) = 1.;
    _transMatrix(5,5) = 1.;
    _transMatrix(6,6) = 1.;
    _transMatrix(7,0) = gm1*0.5*(ubar*ubar + vbar*vbar + wbar*wbar);
    _transMatrix(7,1) = -gm1*ubar;
    _transMatrix(7,2) = -gm1*vbar;
    _transMatrix(7,3) = -gm1*wbar;
    _transMatrix(7,4) = 0.0;
    _transMatrix(7,5) = 0.0;
    _transMatrix(7,6) = 0.0;
    _transMatrix(7,7) = gm1;
}
```

# Installation of COCONUT

## From master version shared on GitHub

*Considering decomposed energy equation.*

### **.Cfcase**

- Simulator.SubSystem.Flow.Data.HLL.UseAlpha = true # Use self-adjustable dissipation term for HLL flux solver.
- Simulator.SubSystem.Flow.Data.HLL.AddLax = true # true for decomposed energy equation with addition dissipation term to HLL flux solver
- Simulator.SubSystem.Flow.Data.SourceTerm = MHDConsACAHWST # calculate source terms as Haopeng did.
  
- Simulator.SubSystem.Tecplot1.Data.outputVar = PrimeE # adopt decomposed energy equation
- Simulator.SubSystem.ParaView1.Data.updateVar = PrimeE # adopt decomposed energy equation
- Simulator.SubSystem.Flow.Data.UpdateVar = PrimeE # adopt decomposed energy equation
- Simulator.SubSystem.Flow.Data.MHDConsACAHWST.deCompEorNot = 1 # 1 for decomposed energy equation with source term caused by decomposed energy equation

# Installation of COCONUT

## From master version shared on GitHub

*Considering full energy equation.*

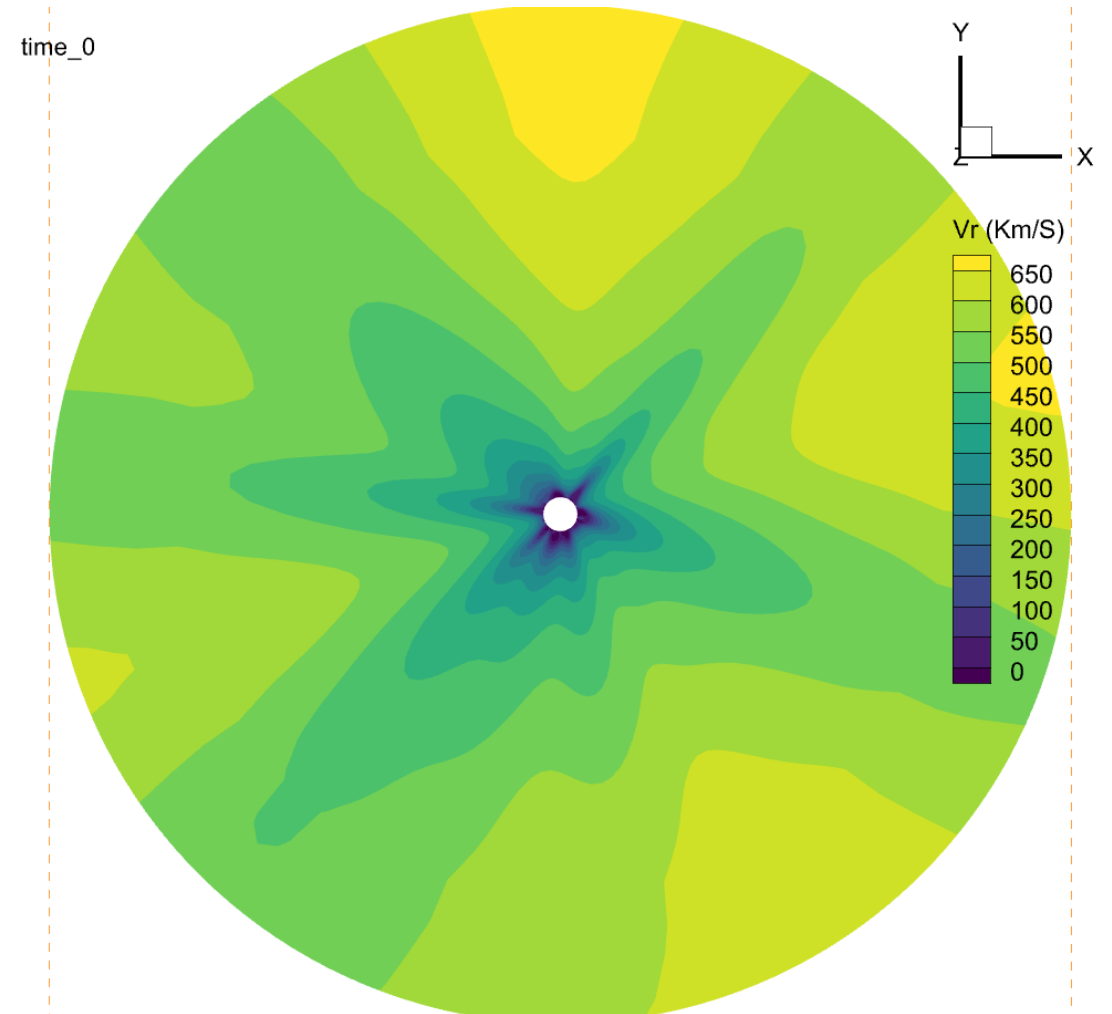
### **.Cfcase**

- Simulator.SubSystem.Flow.Data.HLL.UseAlpha = true
- Simulator.SubSystem.Flow.Data.HLL.AddLax = false # false for full energy equation without addition dissipation term to HLL flux solver
- Simulator.SubSystem.Flow.Data.HLL.AddLax = true # true for full energy equation with addition dissipation term to HLL flux solver
- Simulator.SubSystem.Flow.Data.SourceTerm = MHDConsACAHWST # calculate source terms as Haopeng did.
  
- Simulator.SubSystem.Tecplot1.Data.outputVar = Prim # adopt full energy equation
- Simulator.SubSystem.ParaView1.Data.updateVar = Prim # adopt full energy equation
- Simulator.SubSystem.Flow.Data.UpdateVar = Prim # adopt full energy equation
- Simulator.SubSystem.Flow.Data.MHDConsACAHWST.deCompEorNot = 0 # 0 for full energy equation without source term caused by decomposed energy equation

# Installation of COCONUT

## From master version shared on GitHub

*Calculated by full energy equation  $f$   
or CR2296 (GONGzqs  $L_{\max}=25$ ).*

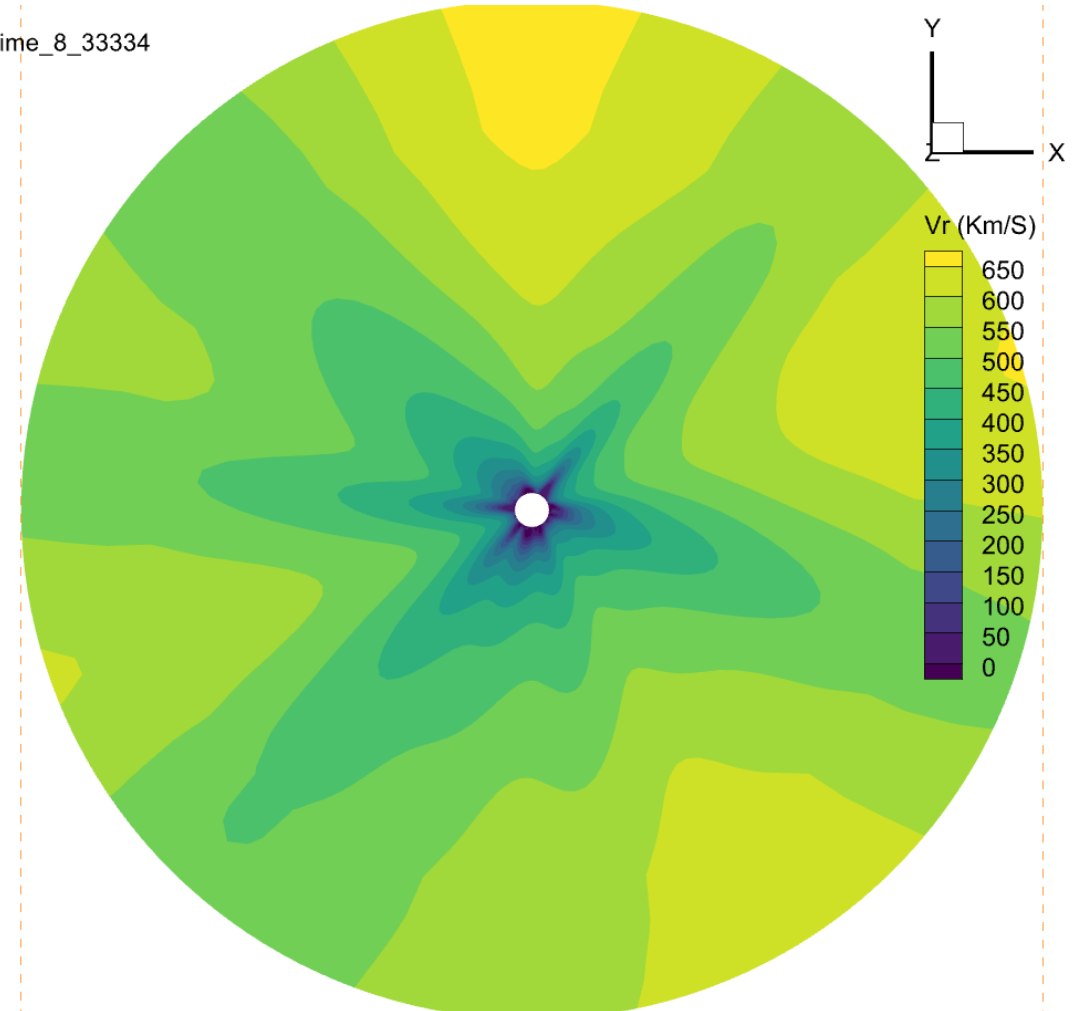


# Installation of COCONUT

## From master version shared on GitHub

*Calculated by full energy equation  $f$   
or CR2296 (GONGzqs  $L_{\max}=25$ ).*

time\_8\_33334



## Call for more collaborations

1. The implicit time-evolving coronal models **COCONUT** and **SIP-IFVM** show great promise for **practical space weather forecasting** due to their efficiency and stability. (Providing inner-boundary conditions for inner-heliosphere models, improve efficiency of CME simulations with required accuracy, ...)
2. The extended magnetic field decomposition strategy and energy decomposition strategies improve the numerical stability of the MHD coronal and CME models in addressing **time-evolving low- $\beta$  issues**.
3. We are **extending** the coronal model to 1 AU or **coupling** the coronal model with an inner heliosphere model to conduct some **faster-than-real-time** and more **realistic** CME simulations from the solar surface to 1 AU.
4. We plan to integrate active region models into global coronal model COCONUT.

# Looking forward to more good ideas and comments

*Thanks*

E-mail: [haopeng.wang1@kuleuven.be](mailto:haopeng.wang1@kuleuven.be)

