OpenCAE
OpenCAE Local User Groups in Japan

Customization of LES turbulence model in OpenFOAM

yotakagi77
Open CAE Local User Groups in Japan
@Kansai

June 13, 2015, Osaka University

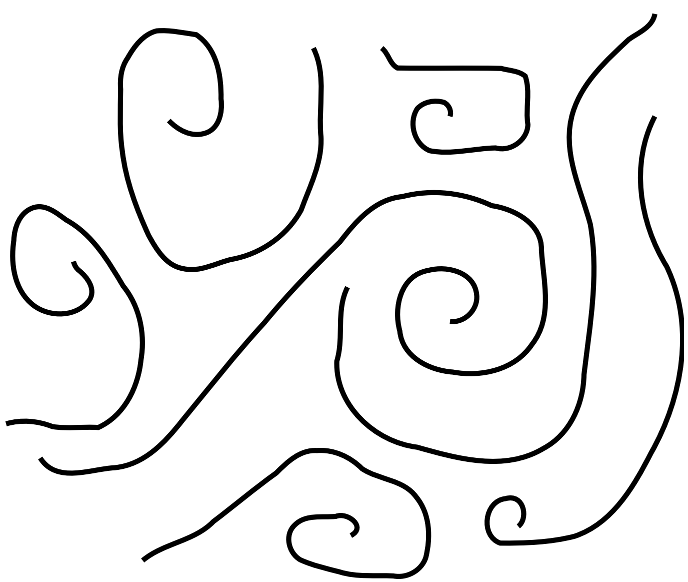
Agenda

- Basic information on turbulence model
- Tensor mathematics
- Exercise 1: Compiling and execution of WALE model
- Exercise 2: Implementation of coherent structure Smagorisky model
- Additional works

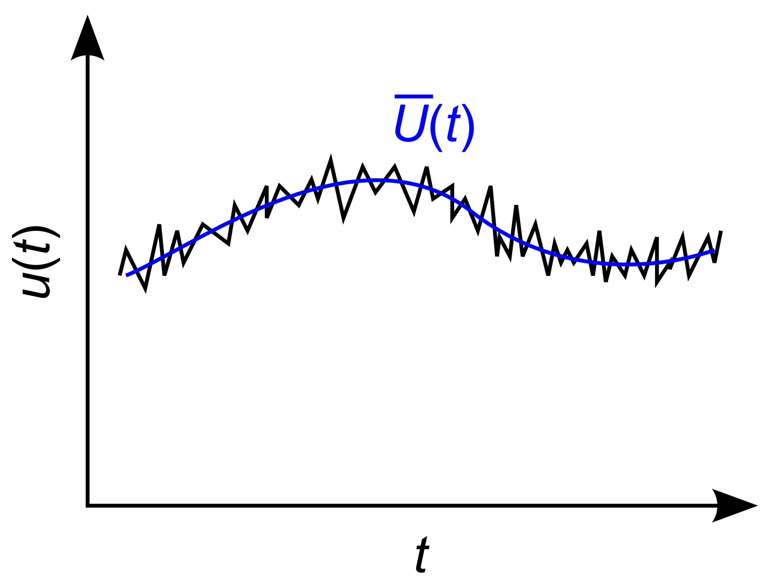
Basic information on turbulence model

Turbulent flow simulation

	DNS	LES	RANS
Modeling	No	Subgrid scale	Reynolds average
Accuracy	◎	○	△
Cost	×	○	◎



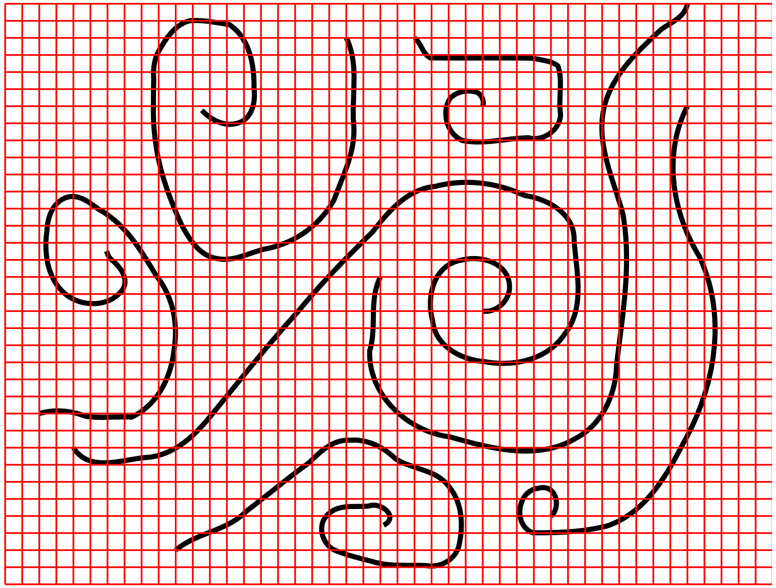
Vortex (eddy) field



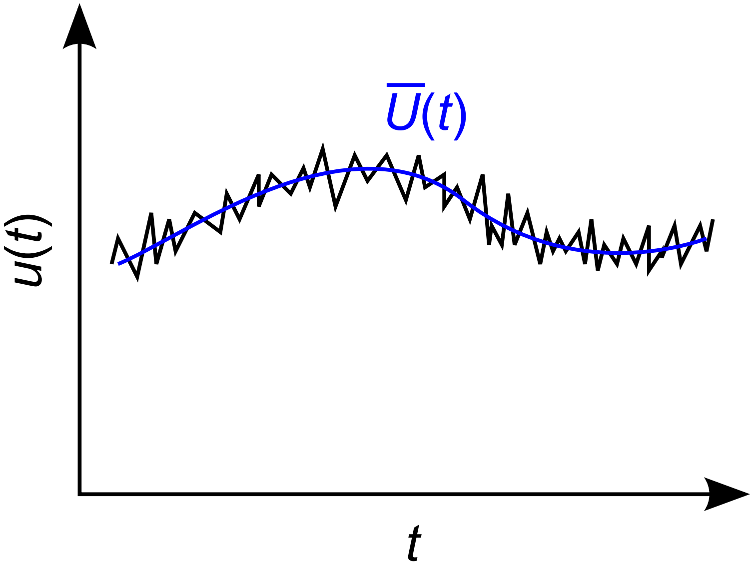
Reynolds average

Turbulent flow simulation

	DNS	LES	RANS
Modeling	No	Subgrid scale	Reynolds average
Accuracy	◎	○	△
Cost	×	○	◎



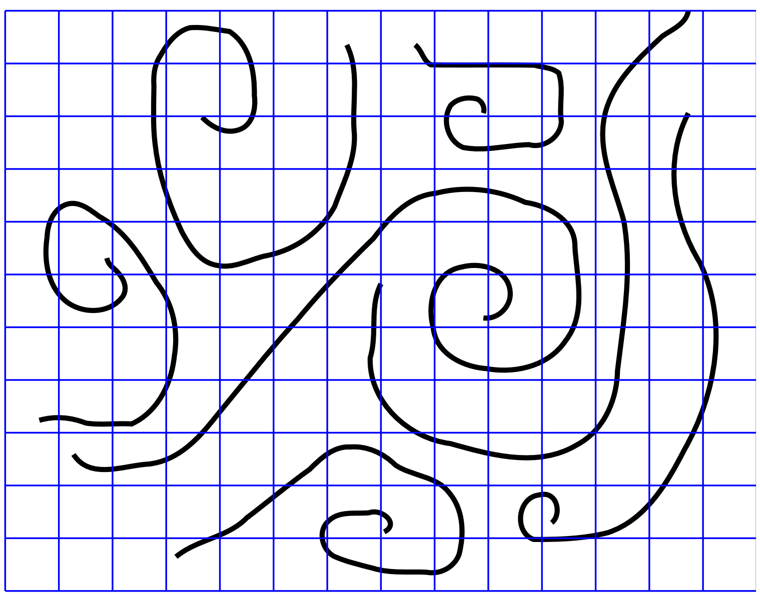
DNS grid, *u*



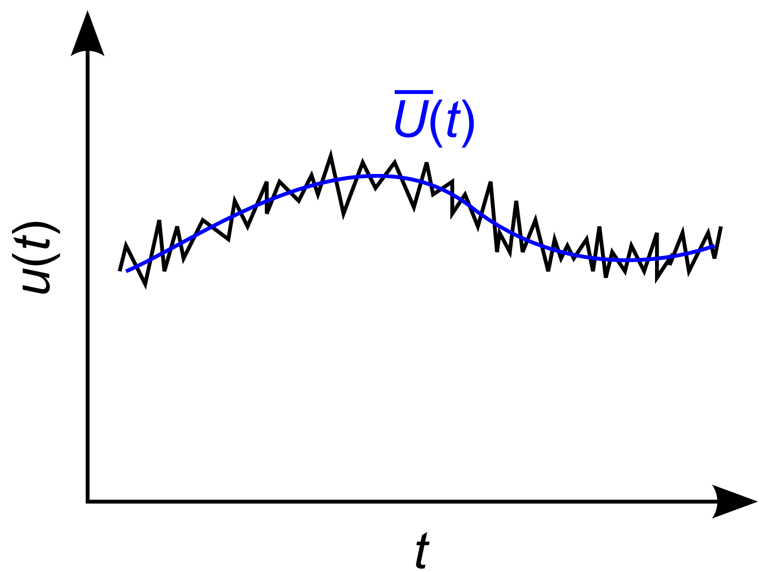
Reynolds average

Turbulent flow simulation

	DNS	LES	RANS
Modeling	No	Subgrid scale	Reynolds average
Accuracy	◎	○	△
Cost	×	○	◎



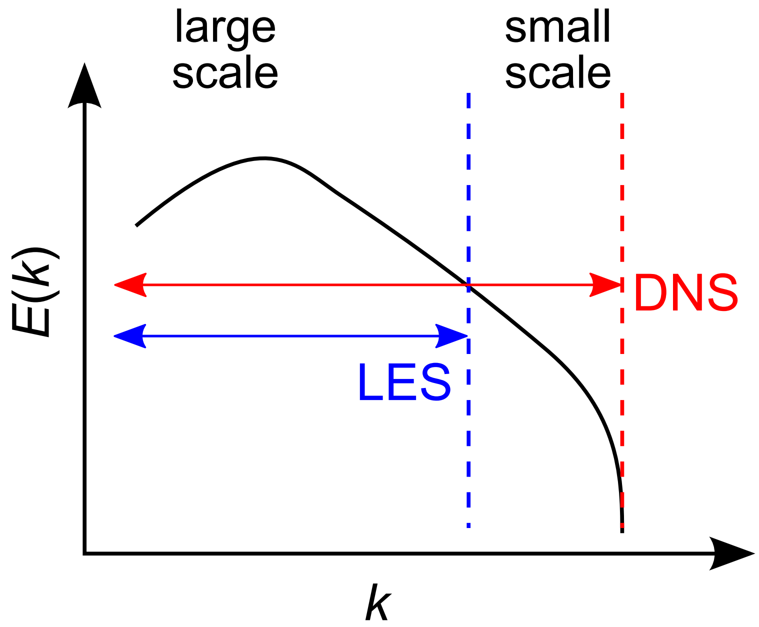
LES grid, $\overline{u} = u - u'$



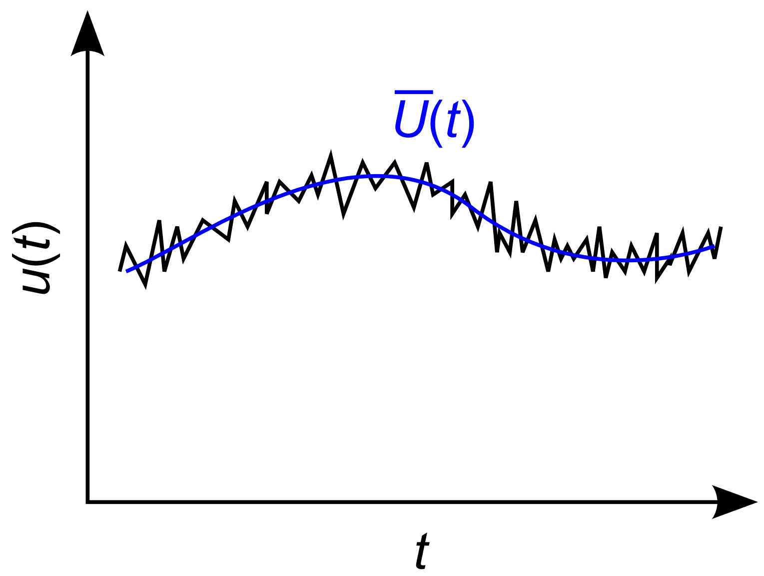
Reynolds average

Turbulent flow simulation

	DNS	LES	RANS
Modeling	No	Subgrid scale	Reynolds average
Accuracy	◎	○	△
Cost	×	○	◎



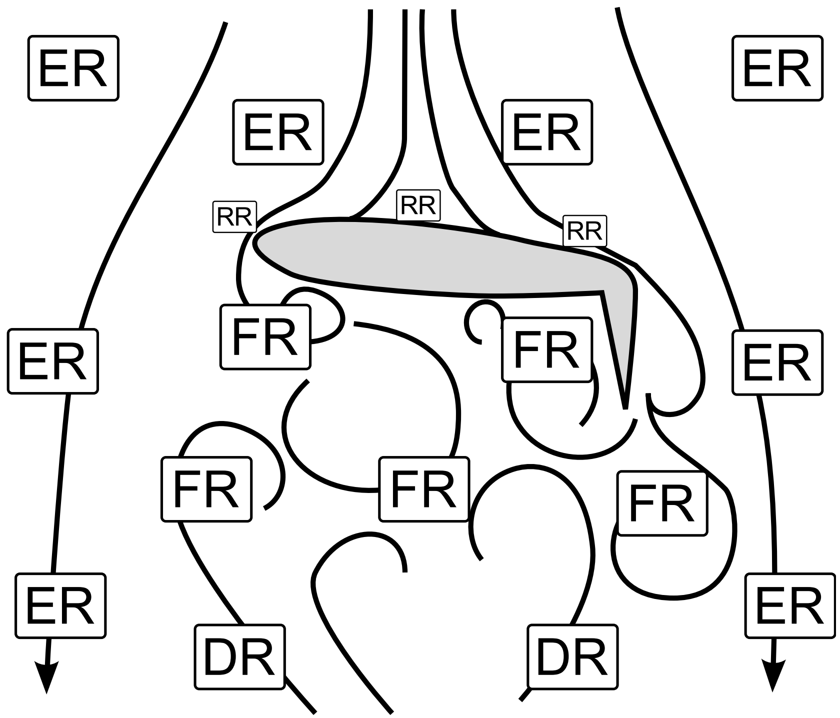
Filtering approach



Reynolds average

Detached-eddy simulation (DES)

- P. R. Spalart (1997):
  - *We name the new approach "Detached-Eddy Simulation" (DES) to emphasize its distinct treatments of attached and separated regions.*

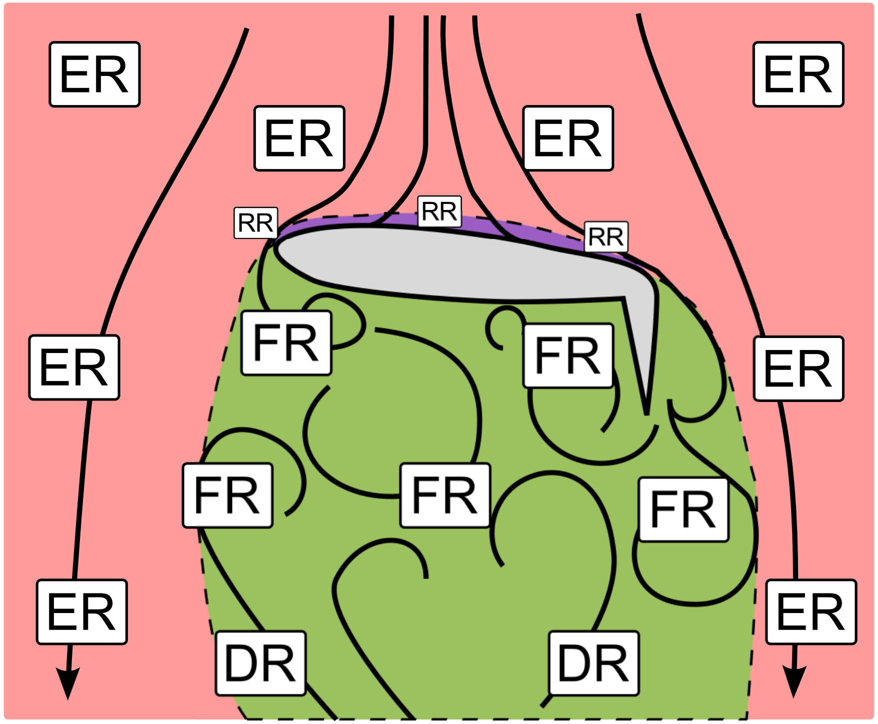


Spalart (2001)

Super-Region	Region
Euler (ER)	
RANS (RR)	Viscous (VR)
	Outer (OR)
LES (LR)	Viscous (VR)
	Focus (FR)
	Departure (DR)

Detached-eddy simulation (DES)

- P. R. Spalart (1997):
  – *We name the new approach "Detached-Eddy Simulation" (DES) to emphasize its distinct treatments of attached and separated regions.*



Spalart (2001)

Super-Region	Region
Euler (ER)	
RANS (RR)	Viscous (VR)
	Outer (OR)
LES (LR)	Viscous (VR)
	Focus (FR)
	Departure (DR)

Coupling with momentum equation through viscosity

- RANS

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla\cdot\left(\mathbf{U}\mathbf{U}\right) - \nabla\cdot\left(\left(\nu + \underline{\nu_t}\right)\left(\nabla\mathbf{U} + (\nabla\mathbf{U})^{T}\right)\right) = \nabla p$$

Turbulent viscosity

- LES

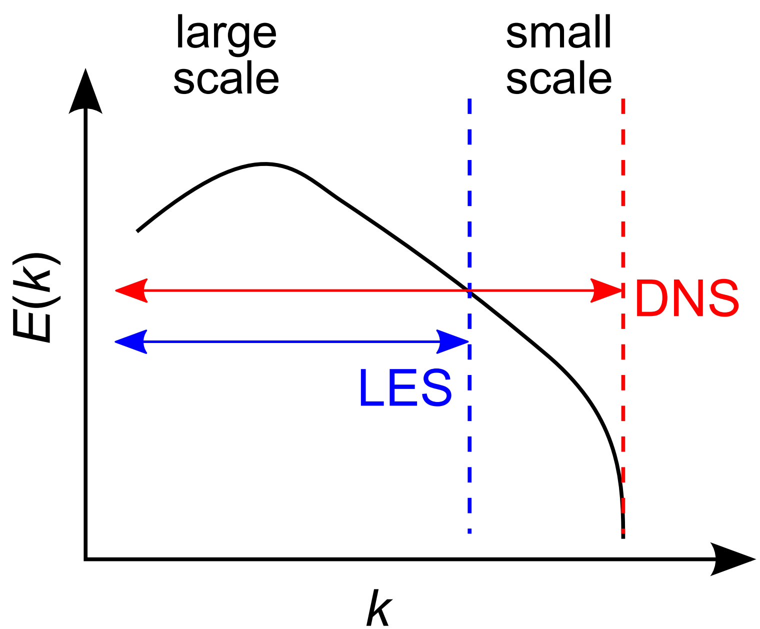
$$\frac{\partial \mathbf{U}}{\partial t} + \nabla\cdot\left(\mathbf{U}\mathbf{U}\right) - \nabla\cdot\left(\left(\nu + \underline{\nu_{SGS}}\right)\left(\nabla\mathbf{U} + (\nabla\mathbf{U})^{T}\right)\right) = \nabla p$$

Sub-grid scale viscosity

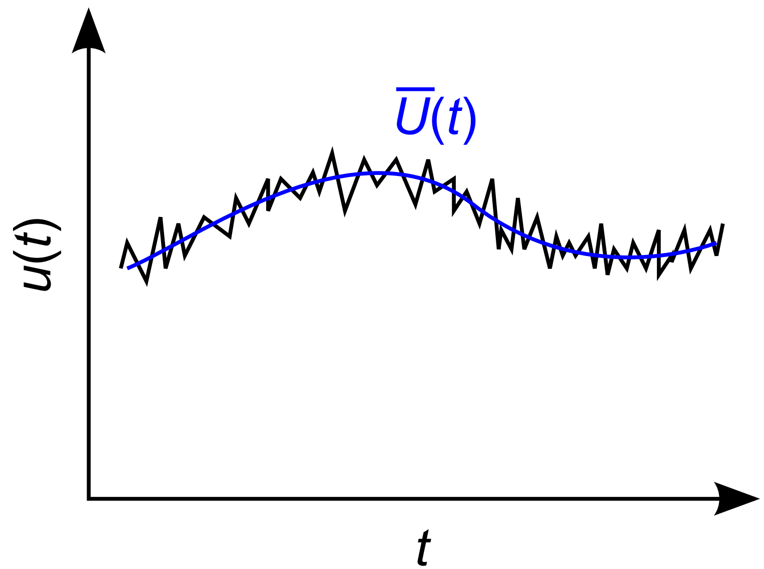
Only change viscosity!

Significant problem: difference of filtering (average) approaches

LES Filtering



Reynolds average



Spatial

Temporal

Inconsistency at the interface between LES and RANS regions

Standard SGS model in OpenFOAM

Library name	Note
Smagorinksy	Smagorinsky model
Smagorinksy2	Smagorinsky model with 3-D filter
homogeneousDynSmagorinsky	Homogeneous dynamic Smagorinsky model
dynLagragian	Lagrangian two equation eddy-viscosity model
scaleSimilarity	Scale similarity model
mixedSmagorinsky	Mixed Smagorinsky / scale similarity model
homogeneousDynOneEqEddy	One Equation Eddy Viscosity Model for incompressible flows
laminar	Simply returns laminar properties
kOmegaSSTSAS	k-ω SST scale adaptive simulation (SAS) model

Standard SGS model in OpenFOAM

Library name	Note
oneEqEddy	*k*-equation eddy-viscosity model
dynOneEqEddy	Dynamic *k*-equation eddy-viscosity model
spectEddyVisc	Spectral eddy viscosity model
LRDDiffStress	LRR differential stress model
DeardorffDiffStress	Deardorff differential stress model
SpalartAllmaras	Spalart-Allmaras model
SpalartAllmarasDDES	Spalart-Allmaras delayed detached eddy simulation (DDES) model
SpalartAllmarasIDDES	Spalart-Allmaras improved DDES (IDDES) model
vanDriestDelta	Simple cube-root of cell volume delta used in incompressible LES models

Tensor mathematics

Tensor

- Rank 0: ‘scalar’, e.g. volume V, pressure p.
- Rank 1: ‘vector’, e.g. velocity vector $\mathbf{u}$, surface vector $\mathbf{S}$. Description: $\mathbf{a} = a_i = (a_1, a_2, a_3)$.
- Rank 2: ‘tensor’, e.g. strain rate tensor S_{ij}, rotation tensor Ω_{ij}.

Description:

$$\mathbf{T} = T_{ij} = \begin{pmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{pmatrix}$$

Symmetric/antisymmetric tensor

- Velocity gradient tensor is decomposed into strain rate tensor (symmetric) and vorticity tensor (antisymmetric, skew).

$$D_{ij} = \frac{\partial u_i}{\partial x_j},\ S_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right),\ \Omega_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i}\right)$$

$$D_{ij} = S_{ij} + \Omega_{ij}$$

- In turbulence modeling, S_{ij} and Ω_{ij} are usually used.

Operations exclusive to tensors of rank 2

$$\mathbf{T} = \frac{1}{2}(\mathbf{T} + \mathbf{T}^{\mathrm{T}}) + \frac{1}{2}(\mathbf{T} - \mathbf{T}^{\mathrm{T}}) = \operatorname{symm}\mathbf{T} + \operatorname{skew}\mathbf{T},$$

$$\operatorname{tr}\mathbf{T} = T_{11} + T_{22} + T_{33},$$

$$\operatorname{diag}\mathbf{T} = (T_{11}, T_{22}, T_{33}),$$

$$\mathbf{T} = \mathbf{T} - \frac{1}{3}(\operatorname{tr}\mathbf{T})\mathbf{I} + \frac{1}{3}(\operatorname{tr}\mathbf{T})\mathbf{I} = \operatorname{dev}\mathbf{T} + \operatorname{hyd}\mathbf{T},$$

$$\det\mathbf{T} = \begin{vmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{vmatrix}$$

OpenFOAM tensor classes

Operation	Mathematical	Class
Addition	$\mathbf{a} + \mathbf{b}$	a + b
Subtraction	$\mathbf{a} - \mathbf{b}$	a – b
Scalar multiplication	$s\mathbf{a}$	s * a
Scalar division	$\mathbf{a} / s$	a / s
Outer product	$\mathbf{a}\,\mathbf{b}$	a * b
Inner product	$\mathbf{a} \cdot \mathbf{b}$	a & b
Double inner product	$\mathbf{a} : \mathbf{b}$	a && b
Cross product	$\mathbf{a} \times \mathbf{b}$	a ^ b
Square	$\mathbf{a}^2$	sqr(a)
Magnitude squared	$\lvert\mathbf{a}\rvert^2$	magSqr(a)
Magnitude	$\lvert\mathbf{a}\rvert$	mag(a)
Power	$\mathbf{a}^n$	pow(a, n)

OpenFOAM tensor classes

Operation	Mathematical	Class
Transpose	$\mathbf{T}^{T}$	T.T()
Diagonal	diag **T**	diag(T)
Trace	tr **T**	tr(T)
Deviatoric component	dev **T**	dev(T)
Symmetric component	symm **T**	symm(T)
Skew-symmetric component	skew **T**	skew(T)
Determinant	det **T**	det(T)
Cofactors	cof **T**	cof(T)
Inverse	inv **T**	inv(T)

Exercise 1: Compiling and execution of WALE model

Governing equation for incompressible LES

- Filtered continuity and Navier-Stokes equations

$$\frac{\partial \overline{u}_i}{\partial x_i} = 0,$$

$$\frac{\partial \overline{u}_i}{\partial t} + \overline{u}_j \frac{\partial \overline{u}_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \overline{p}}{\partial x_i} + \frac{\partial}{\partial x_i}(-\tau_{ij} + 2\nu \overline{S}_{ij})$$

where

$$\tau_{ij} = \overline{u_i u_j} - \overline{u}_i \overline{u}_j$$

SGS eddy viscosity model

- Decomposition of kinetic energy

$$\bar{k} = \frac{1}{2}\overline{u_k u_k} = \underbrace{\frac{1}{2}\bar{u}_k\bar{u}_k}_{k_{GS}} + \underbrace{\frac{1}{2}(\overline{u_k u_k} - \bar{u}_k\bar{u}_k)}_{k_{SGS}}$$

- Conservation of GS energy k_{GS}

$$\frac{\partial k_{GS}}{\partial t} + \bar{u}_j \frac{\partial k_{GS}}{\partial x_j} = \tau_{ij}\bar{S}_{ij} - \varepsilon_{GS} + \frac{\partial}{\partial x_i}\left(-\bar{u}_i \tau_{ij} - \frac{\bar{p}\bar{u}_j}{\rho} + \nu \frac{\partial k_{GS}}{\partial x_j} \right)$$

- Conservation of SGS energy k_{SGS}

$$\frac{\partial k_{SGS}}{\partial t} + \bar{u}_j \frac{\partial k_{SGS}}{\partial x_j} = -\tau_{ij}\bar{S}_{ij} - \varepsilon_{SGS} + \frac{\partial}{\partial x_i}\left[\bar{u}_i \tau_{ij} - \frac{1}{2}(\overline{u_i u_i u_j} + \bar{u}_j \overline{u_i u_i}) - \frac{\overline{p u_j} - \bar{p}\bar{u}_j}{\rho} + \nu \frac{\partial k_{SGS}}{\partial x_j} \right]$$

Smagorinsky model

- Local equilibrium between SGS production rate and SGS energy dissipation:

$$\varepsilon_{SGS}\left(\equiv \nu\overline{\frac{\partial u_i}{\partial x_j}\frac{\partial u_i}{\partial u_j}}-\nu\frac{\partial \bar{u}_i}{\partial x_j}\frac{\partial \bar{u}_i}{\partial x_j}\right)=-\tau_{ij}\bar{S}_{ij}$$

- Eddy viscosity approximation:

$$\tau_{ij}^{a}=-2\nu_{SGS}\bar{S}_{ij}$$

- After dimensional analysis and scaling,

$$\nu_{SGS}=(C_S\Delta)^2\,|\bar{S}|,\ |\bar{S}|=\sqrt{2\bar{S}_{ij}\bar{S}_{ij}},\quad C_S:\ \text{Smagorinsky constant}$$

WALE model

(Nicoud and Ducros, 1999)

- Traceless symmetric part of the square of the velocity gradient tensor:

$$S_{ij}^d = \frac{1}{2}(\bar{D}_{ij}^2 + \bar{D}_{ji}^2) - \frac{1}{3}\delta_{ij}\bar{D}_{kk}^2$$

$$= \bar{S}_{ik}\bar{S}_{kj} + \bar{\Omega}_{ik}\bar{\Omega}_{kj} - \frac{1}{3}\delta_{ij}\left[\bar{S}_{mn}\bar{S}_{mn} - \bar{\Omega}_{mn}\bar{\Omega}_{mn}\right]$$

$$S_{ij}^d S_{ij}^d = \frac{1}{6}(S^2S^2 + \Omega^2\Omega^2) + \frac{2}{3}S^2\Omega^2 + 2IV_{S\Omega},$$

$$S^2 = \bar{S}_{ij}\bar{S}_{ij},\ \Omega^2 = \bar{\Omega}_{ij}\bar{\Omega}_{ij},\ IV_{S\Omega} = \bar{S}_{ik}\bar{S}_{kj}\bar{\Omega}_{jl}\bar{\Omega}_{li}$$

- Eddy viscosity of WALE model:

$$\nu_{SGS} = (C_w\Delta)^2 \frac{(S_{ij}^d S_{ij}^d)^{3/2}}{(\bar{S}_{ij}\bar{S}_{ij})^{5/2} + (S_{ij}^d S_{ij}^d)^{5/4}}$$

Model parameters of WALE model

(Nicoud and Ducros, 1999)

	Field a	Field b	Field c	Field d	Field e	Field f
C_w^2/C_s^2	10.81	10.52	10.84	10.55	10.70	11.27

$$\text{If } C_S = 0.18, \quad 0.55 \le C_W \le 0.6.$$

$$\text{If } C_S = 0.1, \quad 0.32 \le C_W \le 0.34.$$

Model parameter C_w is dependent on Smagorinsky constant C_S.

Source code of WALE model

- V&V working group, Open CAE Society of Japan
  https://github.com/opencae/VandV/tree/master/OpenFOAM/2.2.x/src/libraries/incompressibleWALE

- OpenFOAM-dev
  https://github.com/OpenFOAM/OpenFOAM-dev/tree/master/src/TurbulenceModels/turbulenceModels/LES/WALE

Download and compile

1. Download the source code of WALE model from the V&V repository, and compile the WALE model library.

```
$ mkdir -p $FOAM_RUN
$ cd
$ git clone https://github.com/opencae/VandV
$ cd VandV/OpenFOAM/OpenFOAM-BenchmarkTest/
  channelReTau110
$ cp -r src $FOAM_RUN/..
$ run
$ cd ../src/libraries/incompressibleWALE
$ wmake libso
$ ls $FOAM_USER_LIBBIN
```

Simulation of channel flow

2. The standard tutorial case of channel flow at $Re_\tau = 395$: copy the tutorial case file into your run directory.

```
$ run
$ cp -r $FOAM_TUTORIALS/incompressible/pimpleFoam/
  channel395/ ./ReTau395WALE
$ cd ReTau395WALE
```

3. Edit constant/LESProperties and system/controlDict

```
$ gedit constant/LESProperties
```

```
LESModel         WALE;
printCoeffs      on;
delta            cubeRootVol;
...
```

Simulation of channel flow

```
$ gedit system/controlDict
```

```
...
libs   ("libincompressibleWALE.so");
```

This line is necessary to call the new WALE library in solver.

4. After checking other numerical conditions and parameter, run the solver.

```
$ ./Allrun
```

5. If the solver calculation is normally finished, you check the logs and visualize the flow field with ParaView, and plot the fields profile generated by postChannel.

Simulation of channel flow at Re_τ = 110

6. If you use the test case of channel flow supplied in the V&V repository, copy the template case and edit the setting.

```
$ run
$ cp -r ~/VandV/OpenFOAM/OpenFOAM-BenchmarkTest/
  channelReTau110/template $FOAM_RUN/ReTau110WALE
$ cd ReTau110WALE
$ gedit caseSettings
```

```
controlDict
{
  deltaT          0.002;
  endTime         0.022;
  libs            "libincompressibleWALE.so";
}
```

Simulation of channel flow at Re_{τ} = 110

```
turbulenceProperties
{
  simulationType LESModel;
}

LESProperties
{
  LESModel WALE;
  delta cubeRootVol;
}
```

The original caseSettings is for DNS simulation on large parallel machine. You had better to change other parameters in blockMeshDict and decomposeParDict.

Simulation of channel flow at $Re_\tau = 110$

7. After checking other numerical conditions and parameter, run the solver.

```
$ ./Allrun
```

8. If the solver calculation is normally finished, you check the logs and visualize the flow field with ParaView. If the integration time is not sufficient for the flow field to become fully developed state, run longer simulations.

Exercise 2: Implementation of coherent structure Smagorisky model

Original source codes for SGS model

1. Check the original source code for SGS model.

```
$ src
$ cd turbulenceModels/incompressible/LES/
$ ls
```

2. Glance the codes of Smagorinsky model.

```
$ gedit Smagorinsky/Smagorinsky.*
```

3. In this exercise, we look the codes of dynamic models.

```
$ ls *[Dd]yn*
```

4. Compare the structures and statements of the related codes (*.C and *.H).

Private member functions: updateSubGridScaleFields

In Smagorinsky.C

```
void Smagorinsky::updateSubGridScaleFields
(const volTensorField& gradU)
{   nuSgs_ = ck_*delta()*sqrt(k(gradU));
    nuSgs_.correctBoundaryConditions();      }
```

In dynLagrangian.C

```
void dynLagrangian::updateSubGridScaleFields
(const tmp<volTensorField>& gradU)
{   nuSgs_ = (flm_/fmm_)*sqr(delta())*mag(dev(symm(gradU)));
    nuSgs_.correctBoundaryConditions();      }
```

In dynOneEqEddy.C

```
void dynOneEqEddy::updateSubGridScaleFields
(   const volSymmTensorField& D,
    const volScalarField& KK     )
{   nuSgs_ = ck(D, KK)*sqrt(k_)*delta();
    nuSgs_.correctBoundaryConditions();      }
```

Understanding formulation with codes

- What calculation, mathematical operation, and variable are necessary for coherent structure Smagosinsky model (CSM)? Compare the formulation of models with the related source codes.
- In CSM, the second invariant of velocity gradient is used:

$$Q = \frac{1}{2}\left(\overline{\Omega}_{ij}\overline{\Omega}_{ij} - \overline{S}_{ij}\overline{S}_{ij}\right) = -\frac{1}{2}\frac{\partial \overline{u}_j}{\partial x_i}\frac{\partial \overline{u}_i}{\partial x_j}$$

where

$$S_{ij} = \frac{1}{2}\left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i}\right), \Omega_{ij} = \frac{1}{2}\left(\frac{\partial \overline{u}_i}{\partial x_j} - \frac{\partial \overline{u}_j}{\partial x_i}\right)$$

Coherent structure Smagorinsky model for non-rotating flow (NRCSM)

- Smagorinsky model (SM) based on an eddy-viscosity,

$$\tau_{ij}^{a} = -2C\Delta^2 |\bar{S}| \bar{S}_{ij}$$

$$(\tau_{ij}^{a} = -2\nu_t \bar{S}_{ij},\ \nu_t = C\Delta^2 |\bar{S}|)$$

- The model parameter C is determined as follows:

$$C = C_1 |F_{CS}|^{3/2}$$

with

$$C_1 = \frac{1}{20},\ F_{CS} = \frac{Q}{E}$$

where

$$E = \frac{1}{2}\left(\overline{\Omega}_{ij}\overline{\Omega}_{ij} + \bar{S}_{ij}\bar{S}_{ij}\right) = \frac{1}{2}\left(\frac{\partial \bar{u}_i}{\partial x_j}\right)^2$$

NRCSM model is invalid for rotating flow.

Coherent structure Smagorinsky model (CSM)

- Smagorinsky model (SM) based on an eddy-viscosity,

$$\tau_{ij}^{a} = -2C\Delta^2 |\bar{S}| \bar{S}_{ij}$$

$$(\tau_{ij}^{a} = -2\nu_t \bar{S}_{ij}, \nu_t = C\Delta^2 |\bar{S}|)$$

- The model parameter C is determined as follows:

$$C = C_2 |F_{CS}|^{3/2} F_{\Omega}$$

with

$$C_2 = \frac{1}{22}, F_{CS} = \frac{Q}{E}, F_{\Omega} = 1 - F_{CS}$$

where

$$E = \frac{1}{2}\left(\bar{\Omega}_{ij}\bar{\Omega}_{ij} + \bar{S}_{ij}\bar{S}_{ij}\right) = \frac{1}{2}\left(\frac{\partial \bar{u}_i}{\partial x_j}\right)^2$$

Improved CSM model is valid for rotating flow.

Setting for making new library

1. Copy the source code of WALE model. Compile them.

```
$ run
$ cd ../src/libraries
$ cp -r incompressibleWALE/WALE/ ./NRCSM
$ cp –r incompressibleWALE/Make ./NRCSM
$ cd NRCSM
$ rename WALE NRCSM *
$ rm –r NRCSM.dep
$ rm –rf Make/linux64Gcc47DPOpt
$ gedit Make/files
```

```
NRCSM.C
LIB = $(FOAM_USER_LIBBIN)/libNRCSM
```

```
$ sed –i ‘s/WALE/NRCSM/g’ NRCSM.C
$ sed –i ‘s/WALE/NRCSM/g’ NRCSM.H
```

Setting for making new library

```
$ wmake libso
$ ls $FOAM_USER_LIBBIN
```

If you find the renamed and recompiled library (libNRCSM.so), you are ready to make a new library for the NRCSM.

2.You can easily learn the codes for calculating the *Q* and *E* terms from the postProcessing utilities.

```
$ util
$ cd postProcessing/velocityField/Q
$ gedit Q.C &
```

There are two ways of calculating *Q*, that is, with velocity gradient tensor and with *SS* and $\Omega\Omega$ terms.

Introducing model coefficient C_1

3. Replace all ‘cw’ with ‘c1’ (gedit or sed), and change the value to 0.05.

```
$ run
$ cd ../src/libraries/NRCSM/
$ gedit NRCSM.C NRCSM.H
```

NRCSM.C

```
    c1_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "c1",
            coeffDict_,
            0.05
        )
    )
```

```
$ wmake libso
```

Q and E calculations

4. In NRCSM.C, insert the calculation of Q and E. Copy&Paste the corresponding section from Q.C. Save and Compile them.

```
$ gedit NRCSM.C
```

NRCSM.C

```
volScalarField Q
(
   0.5*(sqr(tr(gradU)) - tr(((gradU)&(gradU))))
);
volScalarField E
(
   0.5*(gradU && gradU)
);
```

```
$ wmake libso
```

F_{CS} and C calculations

5. In NRCSM.C, insert the calculation of F_{CS} and C (coefficient of eddy viscosity model). Save and Compile them.

```
$ gedit NRCSM.C
```

NRCSM.C

```
volScalarField Fcs
(
   Q/
   max(E,dimensionedScalar("SMALL",E.dimensions(),SMALL))
);
volScalarField ccsm_
(
    c1_*pow(mag(Fcs),1.5)
);
```

```
$ wmake libso
```

ν_{SGS} calculation

6. In NRCSM.C, modify the nuSGS_ calculation. Look the other updateSubGridScaleFields functions in the dynamic models.

```
$ gedit NRCSM.C
```

NRCSM.C

```
nuSgs_ = ccsm_*sqr(delta())*mag(dev(symm(gradU)));
```

Save and compile them.

```
$ wmake libso
```

7. Finally, comment out or delete unnecessary statements (the calculations for WALE model). Save and compile them.

```
$ wmake libso
```

k_{SGS} calculation

8. The calculation of k_{SGS} is invalid, but the value of k_{SGS} is not actually used in LES with NRCSM model. If you requires a proper k_{SGS}, consult the paper of Kobayashi (PoF, 2005).

NRCSM.H

```
//- Return SGS kinetic energy
//  calculated from the given velocity gradient
tmp<volScalarField> k(const tmp<volTensorField>& gradU) const
{
   return (2.0*c1_/ce_)*sqr(delta())*magSqr(dev(symm(gradU)));
}
```

Validation with channel flow

9. The standard tutorial case of channel flow at $Re_\tau = 395$: copy the tutorial case file into your run directory.

```
$ run
$ cp -r $FOAM_TUTORIALS/incompressible/pimpleFoam/
  channel395/ ./ReTau395NRCSM
$ cd ReTau395NRCSM
```

10. Edit constant/LESProperties and system/controlDict

```
$ gedit constant/LESProperties
```

```
LESModel        NRCSM;
printCoeffs     on;
delta           cubeRootVol;
...
```

Validation with channel flow

```
$ gedit system/controlDict
```

```
...
libs   ("libNRCSM.so");
```

This line is necessary to call the new NRCSM library in solver.

11. After checking other numerical conditions and parameter, run the solver.

```
$ ./Allrun
```

12. If the solver calculation is normally finished, you check the logs and visualize the flow field with ParaView, and plot the fields profile generated by postChannel.

Additional works

1. Compile and test the WALE model supplied from openfoam-dev. Prepare a Make directory by yourself.
2. Implementation of CSM model. Add F_{Ω} term and C_2 coefficient.
3. Calculation of Q and E terms with SS and $\Omega\Omega$ terms. Compare the results with the solution of Exercise 2.
4. Validation of customized model with other flow fields such pipe, backstep, cylinder, and rotating flow.

References

- OpenFOAM User Guide
- OpenFOAM Programmer's Guide
- 梶島, 乱流の数値シミュレーション 改訂版, 養賢堂 (2014).
- P. R. Spalart et al., "Comments on the Feasibility of LES for Wings, and on a Hybrid RANS/LES Approach", 1st ASOSR CONFERENCE on DNS/LES (1997).
- P. R. Spalart, "Young-Person's Guide to Detached-Eddy Simulation Grids", NASA CR-2001-211032 (2001).
- F. Nicoud and F. Ducros, "Subgrid-scale modelling based on the square of velocity gradient tensor", Flow, Turbulence and Combustion, 62, pp.183-200 (1999).

References

- 小林, “乱流構造に基づくサブグリッドスケールモデルの開発”, ながれ, 29, pp.157-160 (2010).
- H. Kobayashi, “The subgrid-scale models based on coherent structures for rotating homogeneous turbulence and turbulent channel flow”, Phys. Fluids, 17, 045104 (2005).
- H. Kobayashi, F. Ham and X. Wu, “Application of a local SGS model based on coherent structures to complex geometries”, Int. J. Heat Fluid Flow, 29, pp.640-653 (2008).