LESGO tutorial

By Xinyi Huang, Oct. 2023

Caltech

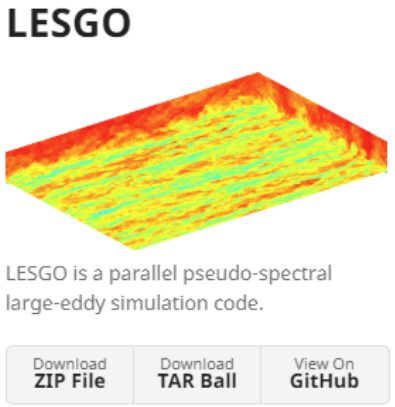
Division of Engineering
and Applied Science

What is LESGO?

- Check out its website.

https://lesgo.me.jhu.edu/

- An open-source code for LES.
- Mainly used by Turbulence Research Group at Johns Hopkins University (PI: Charles Meneveau)
- Also modified and used by his students and collaborators and become a new version.
  - Last major update seems to be around 2017. If you happened to know one of them…
  - (Personal knowledge) Xiaowei Zhu in Portland State University, Johan Meyers in KU Leuven, etc.
- Really easy to use and used in research in the past 30 years.



LESGO solves the filtered Navier-Stokes equations in the high-Reynolds number limit on a Cartesian mesh. Originally designed to simulate flow in the atmospheric boundary layer, LESGO has been extended and used to simulate flow over tree canopies, wall-mounted cubes, and wind turbine arrays, among other things. At its core is the LES flow solver. Built on top of the solver are modules that provide additional functionality such as immersed boundary methods, wind farm modeling, and so on.

LESGO was originally based on the code presented in John D. Albertson's PhD Thesis "Large eddy simulation of land-atmosphere interaction" (University of California, Davis 1996). In the intervening years, many researchers have contributed to LESGO's code base and used LESGO in dozens of scientific publications.

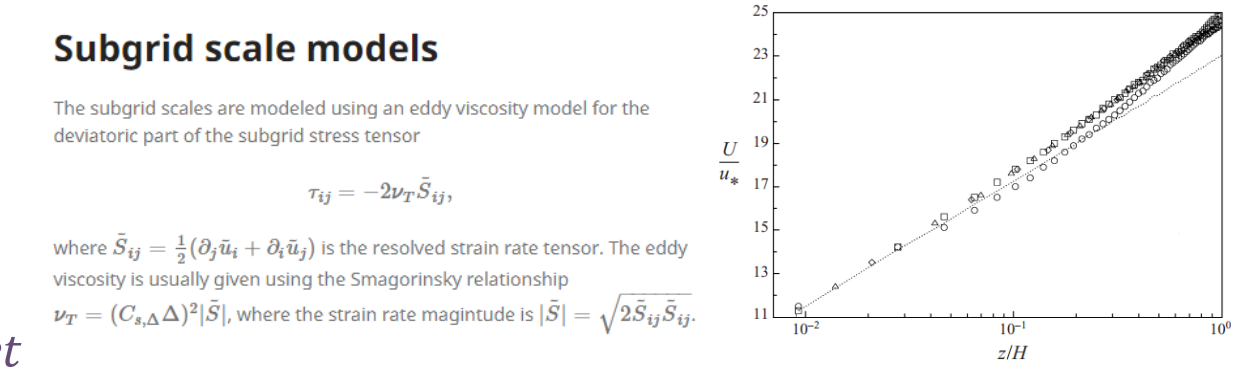
If you use LESGO for any purpose, please cite appropriately.

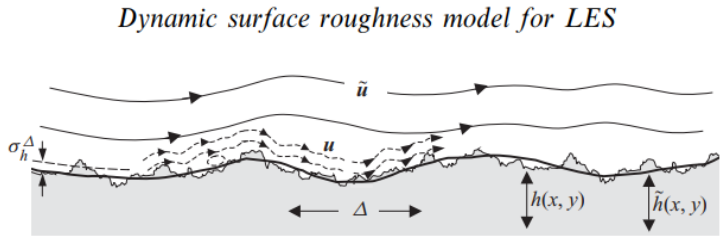
```
2  !!  Copyright (C) 2009-2017  Johns Hopkins University
```

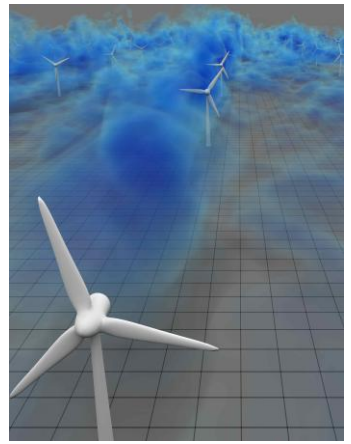
Simulations are performed in SP-Wind, an in-house research code that was developed in a series of earlier studies on LES and wind-farm simulations, and flow optimization (see e.g. Meyers & Sagaut 2007; Delport *et al.* 2009; Meyers & Meneveau 2010). SP-Wind uses a

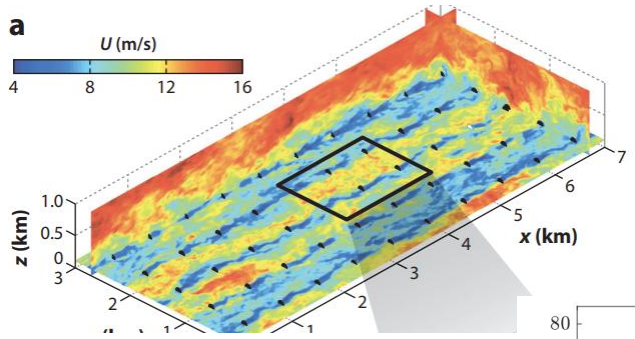
What can LESGO do?

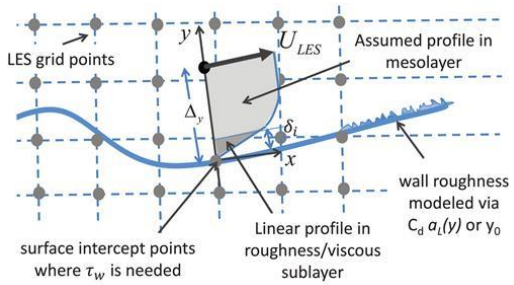
- Tweak subgrid scale models:
  - scale dependent dynamic model (*Porte-Agel et al., 2000, JFM*),
  - Lagrangian scale-dependent dynamic model (*Bou-Zeid et al., 2005, PoF*)
- Simulate flow with roughness:
  - tree canopy (*Chester et al., 2007, JCP*)
  - dynamic roughness model (*Anderson & Meneveau, 2011, JFM*)
- Wind farm simulation:
  - simulate wind turbine array with actuator forcing (*Calaf et al., 2010, PoF*)
  - wind farm flow structures (*Stevens & Meneveau, 2017, ARFM*)
- Wall modelling:
  - integral wall model (Yang et al., 2015, PoF)
  - data-driven wall model (Huang et al., 2019, PoF)

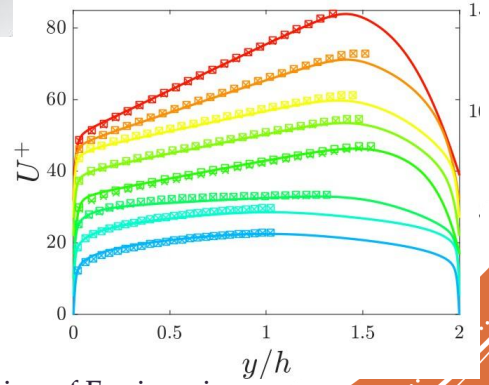












Caltech
Division of Engineering and Applied Science

A little bit of history

*"LESGO solves the filtered Navier-Stokes equations in the high-Reynolds number limit on a Cartesian mesh. Originally designed to simulate flow in **the atmospheric boundary layer**, LESGO has been extended and used to simulate flow over tree canopies, wall-mounted cubes, and wind turbine arrays, among other things. At its core is the LES flow solver. Built on top of the solver are modules that provide additional functionality such as **immersed boundary methods**, **wind farm modeling**, and so on."*

"LESGO was originally based on the code presented in John D. Albertson's PhD Thesis "Large eddy simulation of land-atmosphere interaction" (University of California, Davis 1996). In the intervening years, many researchers have contributed to LESGO's code base and used LESGO in dozens of scientific publications."

- It is first developed based on Albertson & Parlange's work, and mainly used in the field of atmospheric boundary layer.
- Roughness is added by level set IBM; Wind farm is simulated through actuator line model. -> Fully Cartesian mesh!

1996-2004: Professor, Department of Geography and Environmental Engineering, School of Engineering, Johns Hopkins University[13]

1990-1996: Assistant and Associate Professor, Department of Land, Air and Water Resources, and Department of Biological and Agricultural Engineering, University of California Davis[14]

John D. Albertson

Professor

Civil and Environmental Engineering
Location: Hollister Hall, Room 113
Phone: 607/255-9671

E-mail: albertson@cornell.edu

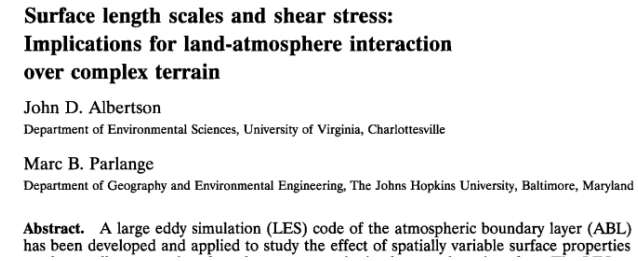
Marc Parlange

12th President of the University of Rhode Island

Contributors

LESGO is the product of work by many research over many years.

William Anderson	UT Dallas, roughness BL, ~2010
Elie Bou-Zeid	Princeton, roughness BL, LDM, ~2004
Joel Bretheim	Princeton, software eng, restricted nonlinear, ~2015
Stuart Chester	?, fractal roughness, ~2001
Jason Graham	?, database, fractal roughness, ~2010
Perry Johnson	UC Irvine, velocity gradient, ~2015
Tony Martinez	NREL, wind turbine wake, ~2016
Fernando Porté-Agel	EPFL, dynamic SGS, ~1998
Carl Shapiro	DOE, wind farm, ~2017
Richard Stevens	U Twente, wind farm, ~2014
Yuheng Tseng	NTU, bluff bodies, ~2004
Claire VerHulst	JHU, eng. innovation, wind farm POD, ~2014
Xiang Yang	PSU, integral wall model, ~2015



Surface length scales and shear stress: Implications for land-atmosphere interaction over complex terrain

John D. Albertson
Department of Environmental Sciences, University of Virginia, Charlottesville

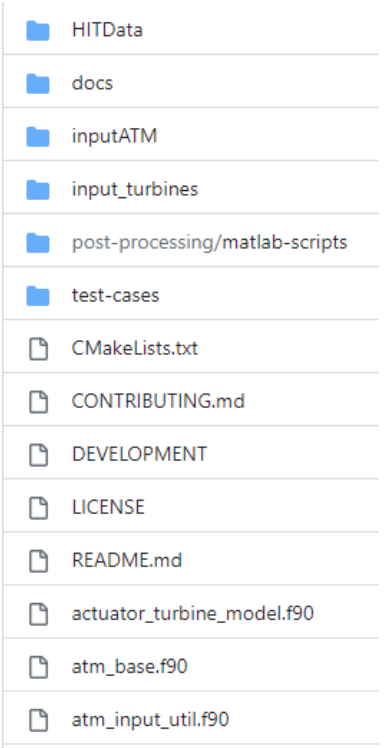
Marc B. Parlange
Department of Geography and Environmental Engineering, The Johns Hopkins University, Baltimore, Maryland

Abstract. A large eddy simulation (LES) code of the atmospheric boundary layer (ABL) has been developed and applied to study the effect of spatially variable surface properties

https://lesgo.me.jhu.edu/

A rough look at the code!

- The code has its own document. It is helpful as a reference and gives some information of the code structure. (Hopefully all details are correct)
- Some test-cases with HITData and other .dat files to help initialization, bash files for running, as runfile, test-lesgo (Never used, no comment)
- Some postprocessing matlab code (Never used, but I would trust that as it is easy Matlab code!)
- A bunch of f90 code (Guess what, it is actually really modern Fortran and uses a lot of pointers!)
- lesgo.conf for full configuration of the test case. (tree.conf if involving level set method for roughness.)



Flow field set up

- Flow field is set on uniform cartesian grid.
  - x for streamwise, y for spanwise, $\boldsymbol{z}$ for wall-normal direction. (Consider that is geophysical society) I personally stick to this coordinate until the moment to publish paper.
  - staggered grid only in z-direction, with w and some derivatives at the wall. This is also for the output velocities. Thus, u resides at $\frac{\Delta z}{2}, \frac{3\Delta z}{2}, \frac{5\Delta z}{2}, \ldots$
- Data input – all in lesgo.conf (tree.conf)
- Data output
  - Set output intervals in lesgo.conf
  - vel.out.XX (for restart); .dat files, as tau_avg.dat, force_avg.dat can be used for convergence check.
  - vel.step_number.cXX, instantaneous flow field for visualization.

Variables	'uv' grid	'w' grid
u, v, p	x	
w		x
dudx, dudy, dvdx, dvdy	x	
dudz, dvdz		x
dwdx, dwdy		x
dwdz	x	
dpdx, dpdy	x	
dpdz		x
txx, txy, tyy, tzz	x	
txz, tyz		x
RHSx, RHSy	x	
RHSz		x
divtx, divty	x	
divtz		x

Simulation configuration – lesgo.conf

- Within available SGS models, only 1-Smagorinsky model and 5-Lagrangian scale-similarity model (default) are still good to use. [To re-examine!]
  - Be aware that test filter is always used, including boundary condition as in wall models.
  - molecular viscosity is generally negligible, e.g., in wall modelling.
- nproc, same as the option in '*mpirun –np* ***nproc*** *lesgo-mpi*'
  - *"The flow domain is evenly divided in the vertical (z) direction between the MPI processes. As such, the number of vertical gridpoints (nz) should be evenly divisible by the number of processors (nproc). "*
  - *"Each MPI-process has its "own" z-levels indexed by k = 1 to nz-1. The z-levels k = 0 and k = nz are overlap nodes and are only for holding data that has been copied from the process below or above the current one."*
- Key parameters
  - u_star, nu_molec, z_i are used for nondimensionalization, which will help set up dimensionless numbers, thus its absolute values have no impact on results
  - Re_tau = z_i * u_star / nu_molec,
    is used for calculating viscous stress in momentum equation
  - zo = (nu/(u_star * z_i)) * exp(- vonk B) is dimensionless, directly derived from log law (of rough wall)
    is used for calculating wall shear stress in equilibrium wall model
  - boundary conditions: lbc_mom=2 equilibrium wall model, ubc_mom=0 stress free. (1 viscous can be used for DNS simulation.)
- Other parameters, leave them as default
  - uniform_spacing means dx=dy=dz
  - mean_p_force, channel driving force, balance the wall shear stress
  - model constants; inflow conditions
- I/O parameters – steps to run, checkpoint interval, instantaneous recording start/interval, other averaging start/interval

```
! Domain parameters
DOMAIN {

  ! Specify the number of processors to
  ! compliance with other preprocessing
  nproc = 8

  Nx = 32
  Ny = 32
  ! Total grid size (stored as nz_tot)
  Nz = 24

  ! Dimensional length scale [m]:
  z_i = 1000.0

  ! Non-dimensional domain length
  Lx = 6.28
  Ly = 6.28
  Lz = 1.00
```

LESGO includes five subgrid models to determine the coefficient $C_{s,\Delta}$:

- The **Smagorinsky model** with the Mason wall damping model specifies $C_{s,\Delta}$ using

$$\lambda^{-n} = \lambda_0^{-n} + [\kappa\,(z+z_0)]^{-n}$$

Wall models

Several options are available for the top and bottom boundary conditions:

- The **stress free** condition sets the stresses and vertical derivatives of the velocity equal to zero

$$\tau_{xz} = \tau_{yz} = 0 \qquad \frac{\partial u}{\partial z} = \frac{\partial v}{\partial z} = 0.$$

- The **viscous** wall condition uses molecular viscosity and calculates the wall stress using the values at the first grid point away from the wall.
- The **equilibrium wall model** applies the local law-of-the-wall expression for the wall stress

$$\tau_w = -\left[\frac{\kappa}{\ln(z/z_0)}\right]^2 (\tilde{u}^2 + \tilde{v}^2)$$

where κ is the von Kármán constant, z_0 is the surface roughness height, and velocities have been test filtered at scale 2Δ.

- The **integral wall model** uses a vertical profile similar to the classical integral method of von Kármán and Pohlhausen.

= $C_0\Delta$ is
onstant
are
isotropic
er scale,
lanes.
gorinsky
arger
amic
nvariance.
es the
field at a

- The **Lagrangian scale-dependent model** dynamically changes the Smagorinsky coefficient $C_{s,\Delta}(\mathbf{x}, t)$ by test filtering the flow field at two larger scales and averaging over Lagrangian trajectories. Unlike the Lagrangian scale-similarity model, the Lagrangian scale-dependent model does not assume scale invariance.

Caltech

More details of the numerics from the website

- *"The LES flow solver uses a pseudospectral discretization in the longitudinal and spanwise directions (x and y, respectively) and second-order centered finite differencing in the wall-normal (z) direction.*
- *"The equations are the filtered N-S equations in the high-Re limit in which molecular viscosity effects are generally neglected. An eddy viscosity closure model is applied for the sub-grid scale stresses.*
- *"Boundary conditions along the x-y perimeter are periodic. The wall-normal boundary condition options include various wall models which prescribe the surface stresses (for LES), no-slip velocity (for DNS), and stress-free. The default wall-normal configuration is a **half-channel** (wall model or no-slip at bottom wall, stress-free at the top to mimic a full channel's center plane), but a full-channel can readily be simulated for many (but not all) combinations of wall model and SGS model and DNS.*
- *"Solid objects can be represented using a level set immersed boundary method. In addition to the bottom wall, solid surfaces represented by the immersed boundary method also apply a log-law stress boundary condition.*
- *"For temporal integration, the explicit second-order Adams-Bashforth method is used. Time stepping occurs using either a static time step or CFL-based dynamic time step.*
- *"Continuity is preserved by solving the pressure Poisson equation using a direct TDMA solver."*

- convec is calculation the Lamb vector.
  - The location of ω_2 is at w-grid, 1st grid is at uv-grid, i.e, the z-grid is $0.5\Delta z, \Delta z, 2\Delta z$, etc.
  - $-\boldsymbol{u} \times \boldsymbol{\omega}$, in x-dir is $-v\omega_3 + w\omega_2$.
    v is on uv-grid, ω_3 is on uv-grid;
    w is on w-grid, ω_2 is usually on w-grid, requires interpolation of $w\omega_2$;
    But for the first layer, ω_2 is on uv-grid, only w needs interpolation.

Some basics before we start

- Some of the very basics for G1 (maybe):
  - know linux commands, basic knowledge of the general process of compilation, a basic idea of mpi (and even fftw), some experience in working with remote system.
  - basic knowledge of Fortan and deep understanding of at least ONE programming language, including data type, features as OOP. Some knowledge of CFD.
  - If you have none, you need a handy senior student (who have them)!
  - If not, use online resources wisely: https://www.cfd-online.com/Wiki/Main_Page, https://www.fftw.org/fftw3_doc/, https://mpitutorial.com/tutorials/.
- Requires CMake, FFTW3 for basic compilation. It has MPI implementation.
  - It does not have HDF5 I/O as it claims. It may be a necessary feature for large data (<2GB).
  - It seems that it implements CGNS feature, but not sure if it works. It may be helpful if to visualize in commercial software as Tecplot (has great document of data format), Ensight, Pointwise.

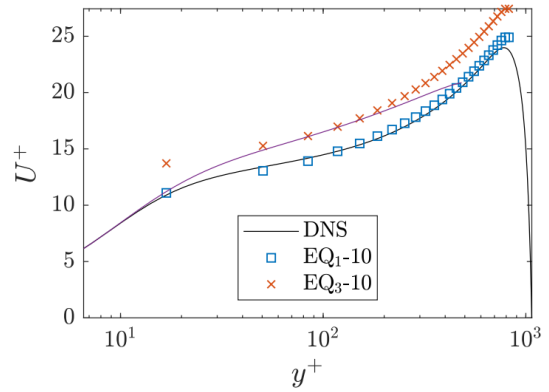
Compilation and running

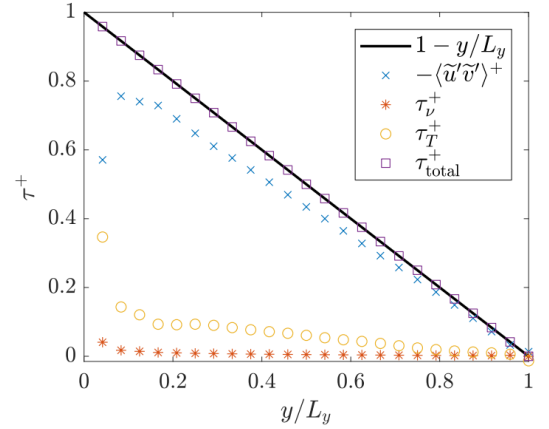
- Compilation, readied in existing bash file.
  - Modules: fortran compiler, MPI; on Resnick, it would be '*module purge; module load mpich/4.0.0*' (GNU) or '*module purge; module load intel/20.1 openmpi/4.1.1_intel-20.1*' (Intel, Resnick cores)
  - Change settings in CMakeList.txt, including the compiler (GNU or Intel), and optional features. Turn all of them off except MPI and others you need.
  - *./build-lesgo,* built executable lesgo-mpi-**
  - *rm –r bld* for clean up.
- Running
  - Load the module
  - *mpirun -np 4 ./lesgo-mpi >& $SLURM_SUBMIT_DIR/log.$SLURM_JOB_ID,* for running

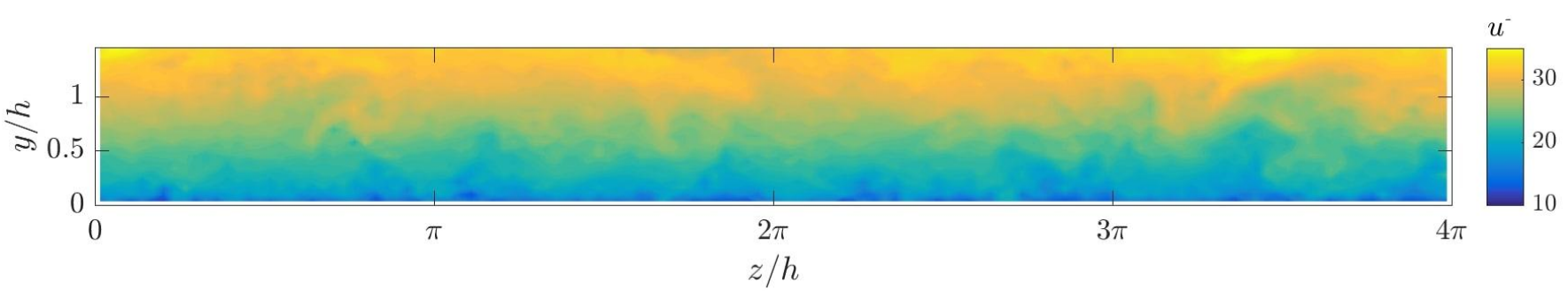
(need to load the corresponding compiler for compiling mpi in the system to compile the program (mpifort), which is by default gcc/4.8.5 in Resnick.
gcc/4.8.5 (and fftw/3.3.3) is by default loaded so only mpich or openmpi needs to be loaded. gcc version can be changed if have corresponding mpi package.
loading intel automatically includes mpi realization and is used through mpiifort instead of mpifort.
LESGO automatically uses ifort or gfortran instead of using wrapper compiler mpifort.)

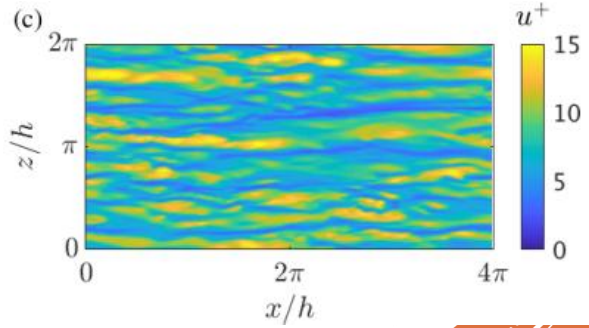
Postprocessing

- Matlab suffices!
- This is the same as any type of code.
- Some in-code averaging (in plane or along time) is provided.
- Both instantaneous flow field and other statistics can be obtained through output files vel.time_step.cXX.
- When saving an archive version, keep lesgo.conf, log file, lesgo_param.out, postprocessed results, even some instantaneous flow fields. Put all parameters in the .mat file to save.









Features

- Flow solver
- Subgrid scale models
- Wall models
- Level set immersed boundary method
- Wind turbine modeling
- Concurrent precursor simulations

How to develop with it

- No need to go into details of files not relevant to the code. Many are not relevant to the main program.
  - The solver itself almost only uses main.f90, convec.f90, divstress_uv.f90, divstress_w.f90, finalize.f90, forcing.f90, initial.f90. initialize.f90, press_stag_array.f90.
  - Other files can be seen as a part of SGS modeling, wall modeling, inflow conditions, I/O processing, and function tools. They are well capsuled in modules.
- Start with running a plain case as half channel to see if you obtain the log law correctly.
  - Basic visualization tools can be developed at the same time.
  - Learn regular behavior of the output. Monitor how well the code converges.
- Follow existing patterns for I/O and code development.
  - Always be suspicious. This is a code passed among generations of grad students so there can be mistakes!
  - (I personally caught a small error on buffer layer communication, but I did not document it so I forgot it. Always make a lot of notes when reading a code!)
  - (In case you are interested, I developed my notes on code reading, which includes general description of every file and the whole program structure.)

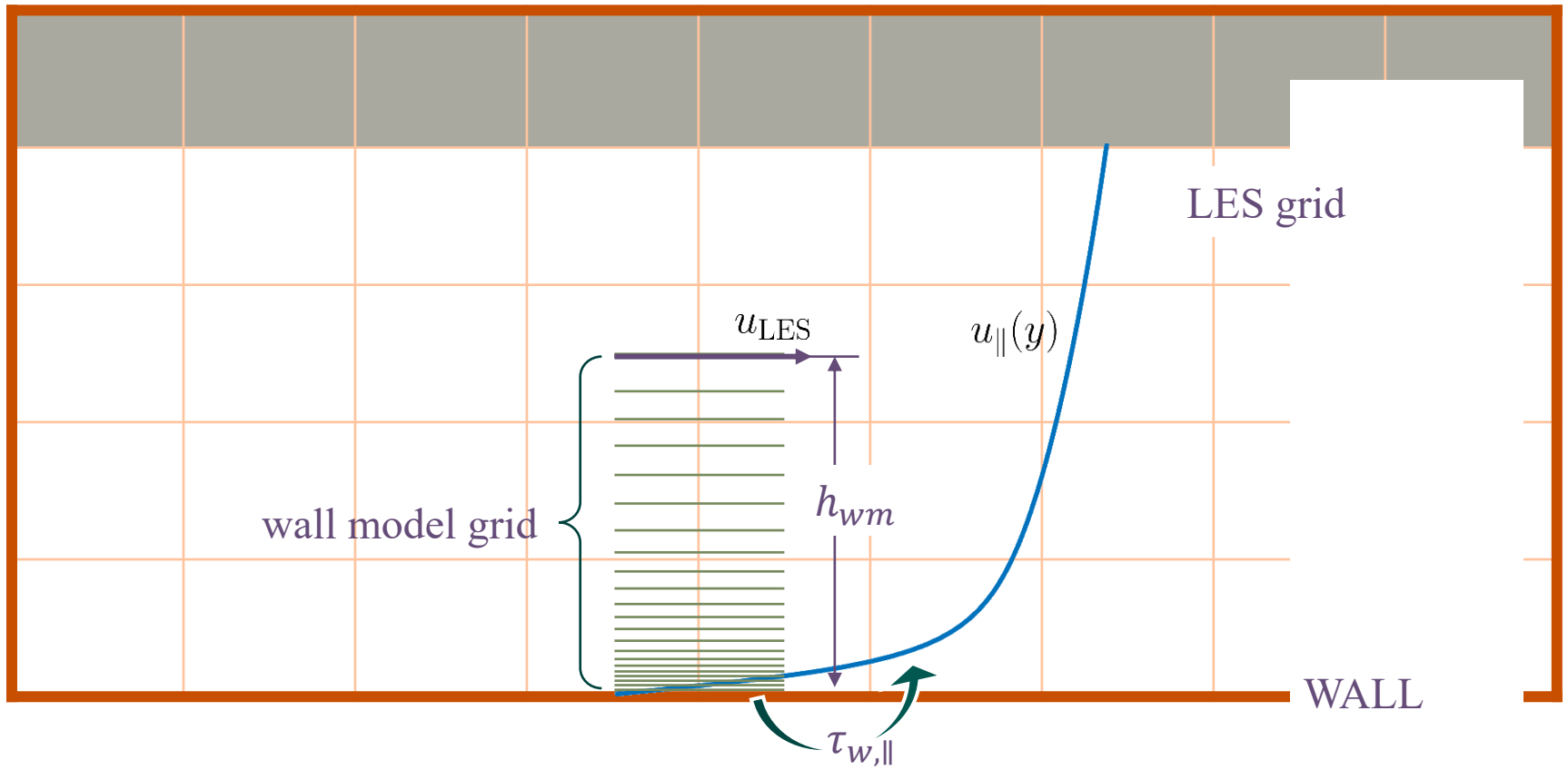
Thank you for listening!

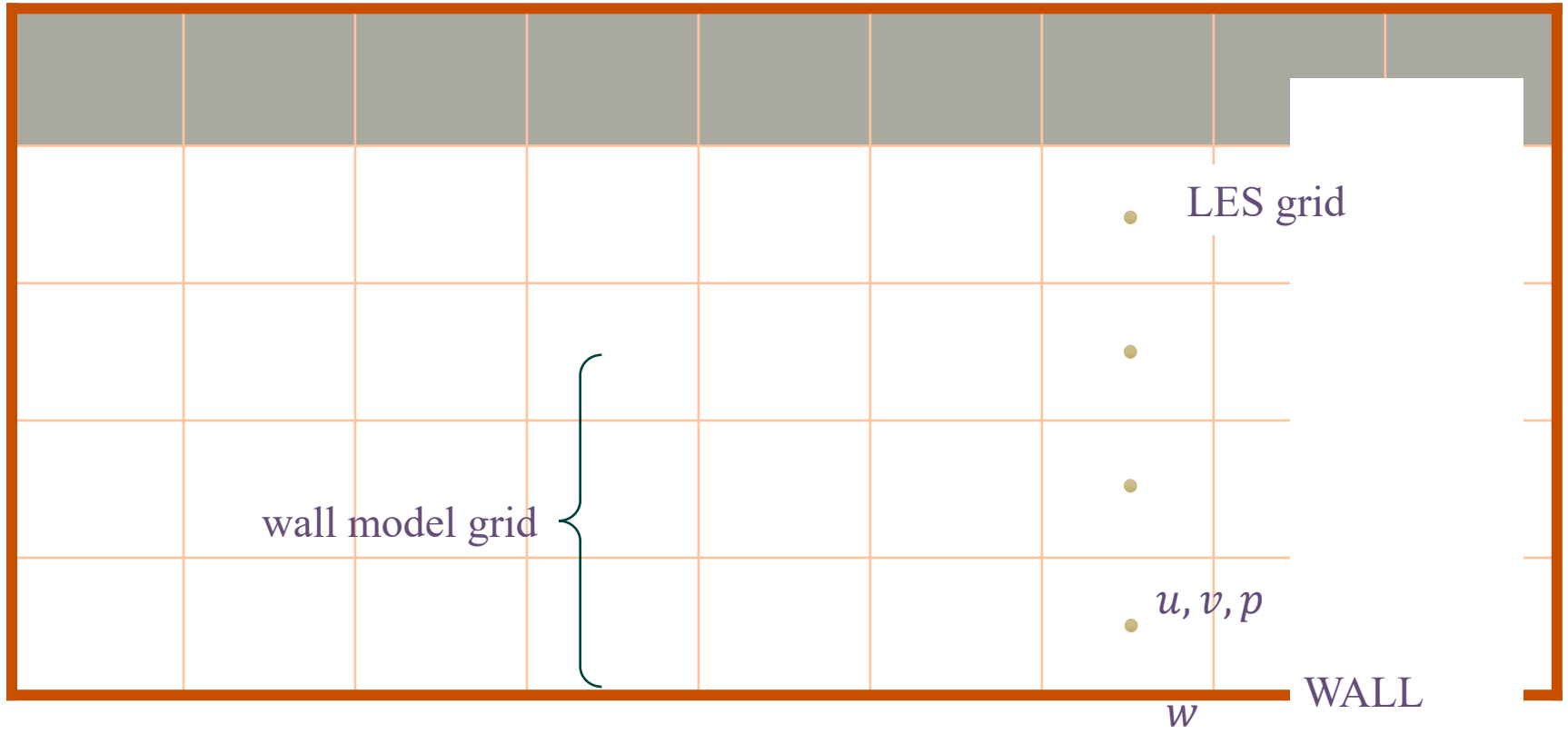
Questions?

Xinyi Huang, xinyih@caltech.edu

Caltech

Division of Engineering
and Applied Science


LES grid
u_{LES}
$u_{\parallel}(y)$
wall model grid
h_{wm}
WALL
$\tau_{w,\parallel}$


LES grid
wall model grid
u, v, p
WALL
w