README

This is a fork of the "Weather Research and Forecast model (WRF)" available at: https://github.com/wrf-model/WRF.

WRFlux allows to output time-averaged resolved and subgrid-scale (SGS) fluxes and other tendency components for potential temperature, water vapor mixing ratio, and momentum for the ARW dynamical core. The included post-processing tool written in Python can be used to calculate tendencies from the fluxes in each spatial direction, transform the tendencies to the Cartesian coordinate system, average spatially, and decompose the resolved advection into mean and resolved turbulence components. The sum of all forcing terms agrees to high precision with the model-computed tendency. The package is well tested and easy to install.
It is continuously updated when new WRF versions are released.
The journal article that introduces WRFlux is available here: https://doi.org/10.5194/gmd-15-669-2022.

Usage

Online calculations

During the model run, fluxes, tendencies, and budget variables are averaged over time.
The online calculations can be controlled in the namelist file or the registry file `Registry/registry.wrflux`. The calculations do not affect the model evolution.

The following namelist variables are available:

- **`output_{t,q,u,v,w}_fluxes`** (default: 0): controls calculation and output for each variable;

  0: no output, 1: resolved fluxes + SGS fluxes + other source terms, 2: resolved fluxes only, 3: SGS fluxes only
- **`avg_interval`**: averaging interval in seconds. If -1 (default), use the output interval of the auxhist24 output stream.
- **`output_dry_theta_fluxes`** (default: .true.): if .true., output potential temperature fluxes and tendencies based on dry theta even when the model uses moist theta (`use_theta_m=1`) internally.
- **`hesselberg_avg`** (default and recommended: .true.): if .true., the time averaging of the budget variables is performed with density-weighting (see Theory/Averaging and Decomposition)

with the budget variables potential temperature (t), water vapor mixing ratio (q) and wind speed (u,v,w).

SGS fluxes include horizontal and vertical fluxes from the diffusion module depending on `km_opt` and vertical fluxes from the boundary layer scheme.
The time-averaged fluxes are output in kinematic form (divided by mean dry air mass) and in the Cartesian coordinate system (see Implementation).

The other source terms that are output beside resolved and SGS fluxes for `output_{t,q,u,v,w}_fluxes=1` are the following:

- **t**: microphysics, SW radiation, LW radiation, convection, Rayleight damping + 6th-order diffusion
- **q**: microphysics, convection, 6th-order diffusion
- **u,v**: Coriolis and curvature, pressure gradient (from RK and acoustic step), convection, Rayleight damping + 6th-order diffusion + external-mode filtering
- **w**: Coriolis and curvature, pressure gradient + buoyancy (from RK and acoustic step), Rayleight damping + 6th-order diffusion

Convection includes tendencies from the cumulus and shallow cumulus schemes.

All variables are output to the auxiliary output stream `auxhist24`. The output interval can be set with the namelist variables `auxhist24_interval_m` and `auxhist24_interval_s`. The averaging starts `avg_interval` seconds before each output time of this output stream. If `avg_interval=-1`, the averaging interval is set equal to the output interval.

To calculate the budget in the post-processing, the instantaneous output (history stream) must contain the following variables:

ZNU, ZNW, DNW, DN, C1H, C2H, C1F, C2F, FNP, FNM, CF1, CF2, CF3, CFN, CFN1, MAPFAC_UY, MAPFAC_VX, MU, MUB, PH, PHB, U, V, W, QVAPOR, T, THM.

These variables are all part of the history stream by default in WRF. From the last six variables you only need the ones for which you want to calculate the budget.
The output interval of the history stream (`history_interval`) should be set in such a way that the start and end points of each averaging interval are available. This means that the averaging interval (`avg_interval`) should be a multiple of the output interval.

You also need to set `io_form_auxhist24` and `frames_per_auxhist24`.

An example namelist file based on the `em_les` test case can be found here:
`wrflux/wrflux/test/input/namelist.input.wrflux`

In addition to the namelist variables `output_{t,q,u,v,w}_fluxes`, you can of course control the output by changing the entries in `registry.wrflux` or using an iofields file (namelist setting `iofields_filename`). In this way you can save disk space, for instance, if you do not need the convection scheme tendencies or do not even have a convection scheme activated. Instantaneous fluxes are by default not output.

Post-processing

In the post-processing, the tendencies are calculated by differentiating the fluxes and decomposed into mean, resolved turbulence, and SGS.
To check the closure of the budget, all forcing terms are added and the total model tendency over the averaging interval is calculated. The post-processing is done with a python package located in the directory `wrflux`. The tendency calculations can be done with the function `tools.calc_tendencies`. A template script is given by `tendency_calcs.py`. This script sets the arguments and runs `tools.calc_tendencies` for some example WRF output data located in the directory `example`. Then the output is checked for consistency and a profile plot is drawn.

The budget for each variable can be calculated in several different ways specified by the `budget_forms` argument. This is a list of strings, where each string is a combination of the following settings separated by a space:

- `cartesian`: advective tendencies in Cartesian instead of native (terrain-following) form
- `adv_form` : transform mean advective and total tendencies to advective form using the mass tendencies

For the tendencies in Cartesian form, corrections are applied to the total tendency and to the horizontal derivatives and the vertical flux is using the Cartesian vertical velocity. See Theory/Advection equation transformations for details.

Since WRF uses flux-form conservation equations, the tendencies output by WRFlux are usually of the form (see also Theory):

$$\frac{\partial_t(\rho\psi)}{\rho} = -\frac{\nabla(\rho\mathbf{v}\psi)}{\rho} + \dots$$

With the setting `adv_form` the tendencies are transformed to the advective form:

$$\partial_t\psi = -\mathbf{v}\nabla\psi + \dots$$

If the WRF output data is too large to fit into memory, the domain can be decomposed into xy-tiles and processed tile per tile. The tile sizes are set in the `chunks` argument. In this case, the values at the domain boundaries cannot be computed correctly and are thus set to NaN.
The tiles can also be processed in parallel:

```
mpiexec -n N ./tendency_calcs.py
```

Other arguments of `tools.calc_tendencies` include: averaging directions (x and/or y) for horizontal average, time-averaging interval (if time-averaged data should be coarsened again before processing), and definition of a limited subset (in time and/or space) to process.

File output

For each budget variable a directory is created at the path specified by the argument `outpath` containing the time-averaged postprocessed output in the following netCDF files:

- **tend.nc** : net tendency (model tendency (i.e., left-hand side) and forcing (right-hand side)) and all tendency components including the advection terms decomposed into mean, resolved turbulence and SGS turbulence
- **flux.nc** : mean advective, resolved turbulent and SGS turbulent fluxes in the x, y and z-direction
- **grid.nc** : some grid related variables used in the post-processing, e.g., grid spacings, metric height and its spatial and temporal derivatives, and μ_d and ρ staggered to the grid of the budget variable
- **corr.nc** : only if budget settings include `cartesian`: corrections for the horizontal divergences and the time derivative
- **tend_mass.nc** : only for potential temperature and also for water vapor if budget settings include `adv_form`: mass tendencies

If horizontal averaging is switched on, the averaging dimension(s) are added to the filenames.

Besides the spatial dimensions and the time dimension the following netCDF dimensions occur in the files:

- **budget_form** : the different forms of the conservation equation: selected budget settings (see beginning of section)
- **dir** : spatial directions of the flux derivatives: X, Y, Z, and their sum
- **comp** : components of the Reynold's decomposition (see Theory/Averaging and Decomposition):
  mean advective (mean), resolved turbulent (trb_r), subgrid-scale turbulent (trb_s), and the sum of all (total)
- **side** : side of the conservation equation: tendency (time derivative) or forcing (sum of all forcing terms)

Installation

This repository contains a complete, standalone version of WRF. Since it is a fork of WRF, the whole history of WRF's master branch is included as well. Thus, you can simply merge your changes with the changes that WRFlux introduces using `git`.

To install the post-processing package in the directory `wrflux` and its dependencies (numpy, pandas, xarray, matplotlib, netcdf4, and bottleneck), I recommend using `conda` and the provided conda environment file.
Switch to the directory `wrflux` and then:

```
conda env create --file conda_env.yml
conda activate wrflux
pip install -e .
```

This also installs an MPI library. Note that when this conda environment is activated, the commands `mpiexec`, `mpif90`, etc. will point to the binaries in the conda environment. This can cause problems when compiling WRF.

The dependencies could also be installed with `pip`. However, the MPI-enabled version of `netCDF4`, needed if you want to use parallel processing of tiles, is easier to install with `conda`.

Check if everything works as expected by running the example script:

```
python tendency_calcs.py
```

or

```
mpiexec -n 2 ./tendency_calcs.py
```

Implementation

Extracting the fluxes

The SGS fluxes are taken directly out of the diffusion routines in `module_diffusion_em.F`. For momentum fluxes, this is already implemented in the official WRF version with the namelist variable `m_opt`. `m_opt` is automatically turned on when using WRFlux.
If a PBL scheme is activated (including `km_opt=5`), it is responsible for the SGS vertical diffusion in WRF. The vertical SGS fluxes are reconstructed by integrating the resulting tendencies cumulatively from the model top to the surface assuming zero flux at the top.

The resolved fluxes are directly taken from the advection routines in `module_advect_em.F`, except for the vertical fluxes.
The vertical fluxes are output in Cartesian form by multiplying the Cartesian vertical velocity with the correctly staggered budget variable. However, instead of using the vertical velocity calculated by WRF, it is recalculated based on the equation given in Theory/Advection equation transformations.
This recalculated w is almost identical to the prognostic w since we adapted the vertical advection of geopotential (subroutine `rhs_ph` in `module_big_step_utilities_em.F`) to avoid a double staggering of ω. This modification is available as a namelist option (`phi_adv_z`) in WRF v4.3 (see PR 1338 for details).
The vertical component of the diagnostic w equation is still calculated in a slightly different way than in the geopotential equation to be more consistent with the vertical advection of other variables. The horizontal terms (terms 2 and 3 in the equation) are directly taken from the geopotential equation.
For potential temperature, fluxes from the acoustic step (`module_small_step_em.F`) are added.

When decomposing the resolved flux into mean and resolved turbulent components in the post-processing (see Theory/Averaging and Decomposition), the turbulent component is calculated by subtracting the mean advective from the total advective flux. The same is done for the resolved advective tendency.

The potential temperature budget

The online flux averaging uses potential temperature perturbation $\theta_\mathrm{p} = \theta - 300\,\mathrm{K}$ like WRF itself.
In the post-processing, however, the tendency calculations are done for full θ for better interpretation.
To obtain the full θ budget, the advection equation is split up into advection of the perturbation and of the constant base state:

$$0 = \partial_t(\mu_\mathrm{d}\theta) - \nabla\cdot(\mu_\mathrm{d}\boldsymbol{\nu}\theta) = \partial_t(\mu_\mathrm{d}\theta_\mathrm{p}) - \nabla\cdot(\mu_\mathrm{d}\boldsymbol{\nu}\theta_\mathrm{p}) + \theta_0(\partial_t\mu_\mathrm{d} - \nabla\cdot(\mu_\mathrm{d}\boldsymbol{\nu}))$$

with the dry air mass μ_d and the contravariant velocity $\boldsymbol{\nu}$.
To close the budget for both, θ_p and θ, the last term on the RHS needs to vanish. In other words the continuity equation must be closed.
Since WRF does not solve the continuity equation explicitly but instead integrates it vertically, this is not trivial. Therefore, we calculate the temporal and horizontal terms explicitly and take the vertical term as the residual.
Using the residual instead of calculating the vertical component explicitly has only a marginal effect on the vertical component, but when the three directions are summed up, the effect is noticeable. By using the velocities averaged over the acoustic time steps when building the time-averaged velocities, the mass fluxes also include the effect of the acoustic time steps.
In the Cartesian form of the budget, correction terms are added to the mass fluxes analogous to the correction terms of the advective fluxes described in the theory section.

Advective form transformation

The components of the continuity equation in the above equation are also used to transform the tendencies from flux-form to advective form when the budget setting `adv_form` is used (see Theory/Advective form).
Only the mean advective tendencies, the total advective tendencies and the final model tendency are transformed to the advective form using the time-averaged continuity equation. This can be done in the postprocessing without changing the online part. The resolved turbulence tendency is left in flux-form.

Miscellaneous

Map-scale factors are taken care of as described in WRF's technical note. Thus, WRFlux is suited for idealized and real-case applications.
All output variables are decoupled from the map-scale factors.

When spatial averaging is switched on in the post-processing, the mean flux in the averaging direction only uses temporal averaging to allow differentiation in that direction. Then, the interior points become irrelevant and only the boundary points matter.

List of modified files

The following WRF source code files have been modified:

- Registry/Registry.EM_COMMON
- Registry/registry.em_shared_collection
- Registry/registry.les
- Registry/registry.wrflux
- dyn_em/module_advect_em.F
- dyn_em/module_avgflx_em.F
- dyn_em/module_big_step_utilities_em.F
- dyn_em/module_diffusion_em.F
- dyn_em/module_em.F
- dyn_em/module_first_rk_step_part1.F
- dyn_em/module_first_rk_step_part2.F
- dyn_em/module_initialize_ideal.F
- dyn_em/module_small_step_em.F
- dyn_em/solve_em.F
- dyn_em/start_em.F
- phys/module_pbl_driver.F
- share/module_check_a_mundo.F
- share/output_wrf.F
- wrftladj/solve_em_ad.F
- wrftladj/solve_em_tl.F

Caveats and limitations

WRFlux has a noticeable impact on the runtime of the model. If all budget variables are considered (`output_{t,q,u,v,w}_fluxes=1` for all variables), the runtime is increased by about 25 %.

Note the following **caveats**:

- If not using the budget setting `cartesian`, the tendencies are calculated in the native form used by the WRF model itself (divided by ρz_η) .Note that this does not correspond to the tendency of the budget variable on model levels since the derivative also includes the coordinate metric z_η. If the distance between the vertical levels changes strongly with time, these tendencies may be hard to interpret. Consider computing the tendencies in Cartesian form.
- When computing the Cartesian flux-form tendency of θ (i.e., $\partial_t \rho\theta$), the budget closure may be poor in some cases. This is due to the fact, that tendencies of ρ and θ are often of opposite sign which leads to a rather small tendency of $\rho\theta$ making the budget closure very challenging. However, the individual flux and budget components are still reliable, only the sum of all counteracting budget components does not match the total tendency very well.

Instead of looking at the flux-form you can compute the tendency in advective form (budget setting `adv_form`) which leads to a better budget in closure in such cases.
- When computing u or v tendencies in advective form together with tiled processing (see Post-processing), errors occur at the tile boundaries. This happens for u if the chunking is in x-direction and for v if the chunking is in y-direction.

and **limitations**:

- The tool does not work with adaptive timestepping
- For hydrostatic simulations (`non_hydrostatic=.false.`) the w-budget is not correct.
- SGS horizontal fluxes can only be retrieved for `diff_opt=2`.
- For non-periodic boundary conditions, the budget calculation for the boundary grid points does not work.
- If using WENO or monotonic advection for scalars, energy is not perfectly conserved. Therefore, when the model uses moist theta (`use_theta_m=1`), the dry theta-budget obtained with `output_dry_theta_fluxes=.true.` is not perfectly closed.

Tests

Test details

This package includes a test suite for automated testing with `pytest`. Idealized test simulations are run for one hour with a large number of different namelist settings to check all parts of the code including different combinations of `km_opt`, `bl_pbl_physics`, `isfflx`, `use_theta_m`, `output_dry_theta_fluxes`, `hesselberg_avg`, `*adv_order`, `*adv_opt`, `mp_physics`, `cu_physics`, `shcu_physics`, `damp_opt`, `diff_6th_opt`, `w_damping`, and boundary conditions. The test simulations are based on the idealized LES test case (`em_les`). To check the output of WRFlux for consistency, the following tests are carried out for these simulations:

- closure of budget (model tendency = summed forcing) in native and Cartesian grid (e > 0.9999)
- resolved turbulent = total - mean advective tendency (e > 0.999995)
- instantaneous vertical velocity very similar to instantaneous diagnosed vertical velocity used in the tendency calculations (e > 0.9995)
- explicit calculation of vertical component of continuity equation very similar to residual calculation (e > 0.99999999)
- no NaN and infinity values appearing except on the lateral boundaries for non-periodic BC

The last test is only done for one simulation.
All simulations, except for the ones with non-periodic BC, contain a 2D mountain ridge to turn on the effect of the Cartesian corrections.
Most simulations use random map-scale factors between 0.9 and 1 to make sure these are treated correctly.

The test statistic used is the coefficient of determination calculated for data d with respect to the reference r:

$$R^2 = 1 - \frac{\mathrm{MSE}(d,r)}{\mathrm{VAR}(r)} = 1 - \frac{\overline{(d-r)^2}}{\overline{(r-\bar{r})^2}}$$

The averaging is over time, height, and along-mountain direction. Afterward, the minimum R^2 value is taken over the remaining dimensions (cross-valley direction, and potentially flux direction, component (mean, resolved turbulent, or total), and budget form). The tests fail if the R^2 score is below the threshold given in brackets for the tests in the list.

For some simulations, the stated thresholds are reduced (see `testing.py` for details): For open and symmetric boundary conditions; when using WENO or monotonic advection for scalars together with `output_dry_theta_fluxes`; and when calculating tendencies for moist θ. The reduction for moist θ is due to the fact that the θ_m-tendencies in the Cartesian coordinate system are very close to 0.

For some simulations, horizontal averaging in the along-mountain direction is tested.

A shorter simulation is run with output at every time step from which turbulent fluxes are calculated explicitly. The thresholds for tests 2 and 3 are reduced in that case.

All test simulations are repeated with a short runtime (2 minutes) in debugging mode (WRF configured with `configure -D`) to detect floating-point exceptions and other issues and with the official WRF version to test for unintended changes of the model dynamics. For the latter, all output variables of the official build and of the debug build are compared bit-for-bit.

Installation

To run the test suite, `pytest` and my other WRF-related package `run_wrf` are required. `run_wrf` facilitates the automatic setup and running of idealized WRF simulations (locally or using a batch system) given a configuration file that defines the simulations to run. It only runs on Linux. Download and install it with `pip`:

```
#install scipy with conda or later with pip
conda install -c conda-forge scipy
git clone https://github.com/matzegoebel/run_WRF.git
cd run_WRF
pip install -e .
```

Then go back to the directory `wrflux` and run:
`pip install -e .[test]`

To run all tests in the test suite you need to have two parallel builds of WRFlux, one with and one without the debugging option (compiled with `configure -D`). To check for differences to the official WRF, a parallel build with debugging option of the *original* branch of this repository is required. This branch always contains the same WRF version as WRFlux is based on. Specify the location of these builds in the configuration file `config/config_test_tendencies_base.py` in the variable `build_path`.
The names of the folders of the builds are specified by the variables `parallel_build`, `debug_build`, and `org_build` in the configuration file.

To run the test suite, execute `pytest` in the folder `wrflux/wrflux/test`. Make sure you do not have a python installation activated with an MPI library if you did not compile WRF with it. This would cause a problem when running the test simulations.
The test results are written to csv tables in the subdirectory `test_results`. For failed tests scatter plots are created in the subdirectory `figures`.

Running all tests on four cores takes about 3 hours (further parallelization is not yet supported). If the simulations do not need to be repeated because the WRF code has not been changed, only the post-processing is tested. This takes less than 10 minutes.

Theory

The theory is explained in detail in the paper: https://gmd.copernicus.org/preprints/gmd-2021-171/.

We repeat it here in a slightly less mathematically rigorous notation.

Advection equation transformations

Flux form

The advection equation for a variable ψ in the Cartesian coordinate system in flux-form reads:

$$\partial_t \left(\rho\psi\right) = \sum_{i=1}^{3} -\partial_{x_i} \left(\rho u_i \psi\right)$$

Like many other atmospheric models, WRF uses a coordinate transformation from the Cartesian coordinate system (t, x, y, z) to (t, x, y, η) with the generalized vertical coordinate $\eta = \eta(x, y, z, t)$. In WRF η is a hybrid terrain-following coordinate.

The transformed advection equation reads:

$$\partial_t \left(\rho z_\eta \psi\right) = \sum_{i=1}^{2} \left[-\partial_{x_i} \left(\rho z_\eta u_i \psi\right)\right] - \partial_\eta \left(\rho z_\eta \omega \psi\right)$$

with the contra-variant vertical velocity ω and the vertical coordinate metric z_η.
In this and the following equations, all horizontal and temporal derivatives are taken on constant η-levels.
Note that in WRF the coordinate metric appears as part of the dry air mass $\mu_\mathrm{d} = -\rho_\mathrm{d} g z_\eta$ in the equations.

In the following, I will derive a form of the advection equation in which the individual parts are the same as in the Cartesian advection equation for improved interpretability, but nevertheless, horizontal derivatives are taken on constant η-levels for ease of computation.

The relationship between the contra-variant vertical velocity ω in the η-coordinate system and the Cartesian vertical velocity w is given by the geopotential equation:

$$w = \frac{\mathrm{d}z}{\mathrm{d}t} = \left(\partial_t \phi + u\partial_x \phi + v\partial_y \phi + \omega \partial_\eta \phi\right)/g = z_t + z_x u + z_y v + z_\eta \omega$$

Solving for $z_\eta \omega$, inserting in the previous equation, and rearranging leads to:

$$\partial_t \left(\rho z_\eta \psi\right) - \partial_\eta \left(\rho z_t \psi\right) = \sum_{i=1}^{2} \left[-\partial_{x_i} \left(\rho z_\eta u_i \psi\right) + \partial_\eta \left(\rho z_{x_i} u_i \psi\right)\right] - \partial_\eta \left(\rho w \psi\right)$$

Dividing by z_η yields the back-transformed advection equation:

$$z_\eta^{-1} \partial_t \left(\rho z_\eta \psi\right) - \partial_z \left(\rho z_t \psi\right) = \sum_{i=1}^{2} \left[-z_\eta^{-1} \partial_{x_i} \left(\rho z_\eta u_i \psi\right) + \partial_z \left(\rho z_{x_i} u_i \psi\right)\right] - \partial_z \left(\rho w \psi\right)$$

Using the product rule and the commutativity of partial derivatives we can show that this equation is analytically equivalent to the following:

$$\partial_t\left(\rho\psi\right) - \partial_z\left(\rho\psi\right) z_t = \sum_{i=1}^{2}\left[-\partial_{x_i}\left(\rho u_i\psi\right) + \partial_z\left(\rho u_i\psi\right) z_{x_i}\right] - \partial_z\left(\rho w\psi\right)$$

This equation follows straight from the Cartesian advection equation. The advective tendency (left-hand side) and the horizontal advection components have a correction term added that accounts for the derivatives being taken on constant η instead of constant height levels.
Numerically however, this form is not perfectly consistent with the original transformed advection equation (see also Alternative Corrections).

Thus, we stay with the previous back-transformed equation, in which the individual components plus their corrections are equivalent to the components of the Cartesian advection equation.

Advective form

To transform the advection equation from the flux-form to the advective form, we can use the mass tendencies introduced in the section about the potential temperature budget.

The left-hand side of the equation can be transformed as:

$$\partial_t\psi = \left(\rho z_\eta\right)^{-1}\left(\partial_t(\rho z_\eta\psi) - \psi\partial_t(\rho z_\eta)\right)$$

and the components of the right-hand side as:

$$u_i\partial_{x_i}\psi = \left(\rho z_\eta\right)^{-1}\left(\partial_{x_i}\left(\rho z_\eta u_i\psi\right) - \psi\partial_{x_i}\left(\rho z_\eta u_i\right)\right)$$

In the Cartesian coordinate system correction terms for the mass tendency components are introduced analogously to the correction terms for the ψ tendency.

Numerical implementation

WRF uses a staggered grid, where the fluxes of ψ are staggered with respect to ψ. If ψ is on the mass grid (potential temperature and mixing ratio) the equation with staggering operations indicated reads:

$$z_\eta^{-1}\partial_t\left(\rho z_\eta\psi\right) - \partial_z\left(\rho z_t\overline{\psi}^z\right) = \sum_{i=1}^{2}\left[-z_\eta^{-1}\partial_{x_i}\left(\rho z_\eta u_i\overline{\psi}^{x_i}\right) + \partial_z\left(\rho z_{x_i}\overline{u_i}^{x_i z}\overline{\psi}^z\right)\right] - \partial_z\left(\rho w\overline{\psi}^z\right)$$

where the staggering operations are denoted with an overbar and the staggering direction. For momentum, the equation looks a bit differently, since also the velocities in the fluxes need to be staggered. The staggering of ψ depends on the advection order as described in WRF's technical note. For the back-transformed advection equation to be numerically consistent with the original transformed advection equation, all derivatives need to use the correct advection order. The correction terms derive from the vertical advection term and thus must use the order of the vertical advection.

Averaging and decomposition

When the advection equation is averaged over time and/or space, the fluxes and resulting tendency components can be decomposed into mean advective and resolved turbulent components which is useful for LES simulations that partly resolve the turbulent spectrum.
Since WRF is a compressible model, we use a density-weighted average (Hesselberg averaging) unless `hesselberg_avg=.false.`. The effect of the density-weighting is rather small.

Means and perturbations are defined by:

$$\widetilde{\psi} = \frac{\langle \rho\psi \rangle}{\langle \rho \rangle}, \quad \psi'' := \psi - \widetilde{\psi}$$

$\langle \psi \rangle$ denotes the time and/or spatial average, $\widetilde{\psi}$ is the density-weighted average, and ψ the perturbation.

The flux decomposition then reads:

$\langle \rho u_i \psi \rangle = \langle \rho \rangle \, \widetilde{u}_i \widetilde{\psi} + \langle \rho u_i'' \psi'' \rangle$ for $i = 1, 2, 3$.

The correction flux is decomposed like this:

$\langle \rho Z_i \psi \rangle = \langle \rho \rangle \, \widetilde{Z}_i \widetilde{\psi} + \langle \rho Z_i'' \psi'' \rangle$ for $i = 1, 2$

with $Z_i = z_{x_i} u_i$.

Note that the time average is a block average, not a running average.

Contributing

Feel free to report issues on Github.
You are also invited to fix bugs and improve the code yourself. Your changes can be integrated with a pull request.

Acknowledgments

Thanks to Lukas Umek who provided the code used as a starting point for WRFlux: https://github.com/lukasumek/WRF_LES_diagnostics.