ransX framework

Implementation Guide

Miroslav Mocák, Casey Meakin, Simon Campbell, Cyril Georgy, Maxime Viallet, Dave Arnett

April 7, 2019

Contents

1 Introduction

This implementation guide describes all parts of our analysis framework for multi-fluid compressible hydrodynamic simulation targeted at all of you, who wants to implement it to your hydrodynamic codes and use our post-processing software to display its results.

ransX or rans(eXtreme) framework is a theoretical and programmatic suite allowing for comprehensive analysis of statistical averages of all sort of hydrodynamic properties from 3D simulations in spherical and Cartesian geometry. It consists of three main parts:

- theoretical derivation of 1D Reynolds-averaged Navier-Stokes (RANS) mean-field equations for transport/flux/variance of mass, momenta, kinetic/internal/total energy, temperature, enthalpy, pressure and chemical composition (see ransXtheoryGuide.pdf for more details)
- calculation of required mean-fields for construction of terms in the RANS equations at runtime of hydrodynamic simulations
- construction of terms in the RANS equations and their plotting

We obtain our 1D RANS equations by introducing two types of averaging: statistical averaging and horizontal averaging [Besnard et al., 1992, Viallet et al., 2013]. In practice, statistical averages are computed by performing a time average. Therefore, the combined average of a quantity q is defined for spherical geometry as

$$\overline{q}(r,t) = \frac{1}{T\Delta\Omega}\int_{t-T/2}^{t+T/2} q(r,\theta,\phi,t')\ d\Omega\ dt' \tag{1}$$

where $d\Omega = \sin\theta d\theta d\phi$ is the solid angle in spherical coordinates, T is the averaging time period, and $\Delta\Omega$ is total solid angle being averaged over. The quantity q is defined for cartesian geometry as

$$\overline{q}(x,t) = \frac{1}{T\Delta y\Delta z}\int_{t-T/2}^{t+T/2} q(x,y,z,t')\ dy\ dz\ dt' \tag{2}$$

where Δy and Δz is total number of grid zones in y and z direction.

The flow variables are then decomposed into mean and fluctuation $q = \overline{q} + q'$, noting that $\overline{q'} = 0$ by construction. Similarly, we introduce Favre (or density weighted) averaged quantities by

$$\widetilde{q} = \frac{\overline{\rho q}}{\overline{\rho}} \tag{3}$$

which defines a complimentary decomposition of the flow into mean and fluctuations according to $q = \widetilde{q} + q''$. Here, q'' is the Favrian fluctuation and its mean is zero when Favre averaged $\widetilde{q''} = 0$. For a more complete elaboration on the algebra of these averaging procedures we refer the reader to Chassaing et al. [2010]

2 Calculation of ransX mean fields

We perform the calculation of required mean-fields at runtime of hydrodynamic simulations and exploit various identities between terms in our RANS equations and space-time averaged thermodynamic quantities or their products. For example, mass flux $\overline{\rho' u_r'} = \overline{\rho u_r} - \overline{\rho}\ \overline{u_r}$, so in order to calculate the mass flux, we need $\overline{\rho u_r}$, $\overline{\rho}$, $\overline{u_r}$. The following subsections describe this methodology implemented to the multi-fluid compressible hydrodynamic code PROMPI of [Meakin and Arnett, 2007] and post-processing tool tailored for its output written in Python https://github.com/mmicromegas/ransX. This github repository contains also PROMPI's subroutines that deal with RANS averaging and storage located in directory UTILS/FOR_YOUR_HYDRO. **Feel free to use them in your hydrodynamic codes as well. We hope it speeds up integration of our ransX framework to your research projects.** The directory contains also README file, that will provide you with more details about how and where we use the subroutines in PROMPI.

2.1 Initialization and calculation in hydrodynamic code PROMPI

Our main array for storage of space-time averages between two consecutives data dumps is called $havg$. It is a three dimensional array declared as $havg(4, nrans, qqx)$. First index referes to identification position of horizontal averages of three-dimensional thermodynamic quantities required either at initialization or update stage of hydrodynamic simulation. The seconds index is the number of RANS mean-fields we calculate. The third index is number of grid points in radial or x direction (depending on geometry).

The main subroutine, where we calculate *havg* is called **rans_avg.f90**. It requires an input of one parameter called *imode*, which tells it where in hydrodynamic simulation it is being called and whether it should be initializing *havg* or updating it (Fig.1). If in initialization mode (*imode.eq.*0), it initializes also an array called *ransname* (Fig.2), which holds names of horizontally-averaged variables. The *havg* data is stored in a binary file with extension ransdat, whether variables like *ransname* to a text file with extension ranshead together with other variables like resolution and dump times.

```
if(imode.eq.0) then
   rans_tstart = time
   rans_tavg   = 0.d0
   do n=1,nrans
      do i=1,qx
         havg(1,n,i) = 0.d0
         havg(2,n,i) = 0.d0
         havg(3,n,i) = 0.d0
      enddo
   enddo
else if(imode.eq.1) then
   do n=1,nrans
      do i=1,qx
         havg(1,n,i) = havg(2,n,i)
         havg(2,n,i) = 0.d0
      enddo
   enddo
endif
```

Figure 1: Initialization of *havg* array in **rans_avg.f90**

After the initialization, horizontal averages are being computed as show in the example in Fig.2 according to Equation 4 and stored in the *havg* array where the identification position index is set to 2 i.e. $havg(2, ifield, i)$. The scaling factor in

case of spherical geometry $\sin\theta d\theta d\phi/\Delta\Omega$ is the variable $fsteradjk$. In case of cartesian geometry, this factor will be equal to one.

$$\langle q\rangle(r,t) = \frac{1}{\Delta\Omega}\int\int q(r,\theta,\phi,t)\ \sin\theta d\theta d\phi \tag{4}$$

```
do k=1,qz
   do j=1,qy
   fsteradjk = fsterad(j,k)
   do i=1,qx
      dd(i)   = densty(i,j,k)      ! density (g cm-3)
   enddo
   ! dd  (1)
   ifield = 1
   if(imode.eq.0) ransname(ifield) = 'dd'
   do i=1,qx
      havg(2,ifield,i) = havg(2,ifield,i)  + &
           dd(i)*fsteradjk
   enddo
   enddo
enddo
```

Figure 2: Demonstration loop of horizontal averaging (Eq.4), where $\langle q\rangle$(r,t) is $havg(2, ifield, i)$ taking geometry into account with scaling variable $fsteradjk$

After calculation of these horizontal averages, they are turned into a variable that we call running averages defined in Equation 5 and stored in $havg$ array with identification position index set to 3 i.e. $havg(3, ifield, i)$ (Fig.3). These running

averages are horizontal averages, that are additionally averaged further taking into account previous hydro sweep stored in *havg* with identification position index set to 1 and updated as shown in Fig.1.

$$q_{\rm run} = \sum_i \left(\langle q^n \rangle_i + \langle q^n \rangle_{i-1}\right)\ 0.5\ \Delta t_i \tag{5}$$

```
if(imode.eq.1) then
   rans_tavg = rans_tavg + dt
   rans_tend = time
   do n=1,nrans
      do i=1,qx
         havg(3,n,i) = havg(3,n,i) + &
              (havg(2,n,i) + havg(1,n,i))*0.5d0*dt
      enddo
   enddo
else if(imode.eq.0) then
   do n=1,nrans
      do i=1,qx
         havg(4,n,i) = havg(2,n,i) !store first instance in this averaging interval
      enddo
   enddo
endif
```

Figure 3: Calculation of running averages (Eq.5), where $q_{\rm run}$ is $havg(3, n, i)$.

The running averages are later used in a subroutine **write_rans_data.f90** to calculate statistical average according to Equation 6 at the time of data dump as show in the Figure 4.

$$\overline{q} = \frac{q\text{run}}{\sum_i \Delta t_i} \tag{6}$$

```
do k=1,nrans
   do i=1,qqx
      if(rans_tavg.gt.1.d-10) then
         havg_sum_tot_global(3,k,i) = havg_sum_tot_global(3,k,i)/rans_tavg
      else
         havg_sum_tot_global(3,k,i) = havg_sum_tot_global(2,k,i) !use instantaneous in this case
      endif
   enddo
enddo
```

Figure 4: Calculation of statistical average (Eq.6), where $\overline{q}$ is $havg\_sum\_tot\_global(3, k, i)$. $rans\_tavg$ calculation is shown in Figure 3.

2.2 Post-processing in Python

The statistical averages calculated according to Eq.6 are only between two consecutives data dumps, that typically cover time much shorter than a convective turnover timescale (TO). In order to get robust statistical averages, we need to calculate the averages over at least three TOs. For that we use python script called *rans_tseries.py*, that is using calculated qs stored in ransdat files according to Equation 7 shown in Figure 5.

The script takes advantage of ranshead files, that contain names of the RANS mean fields in exact order as they are stored in *havg* array and stores the final mean fields in a dictionary variable called *eht*. The dictionary is at the end stored in python's *npy* file, that can be easily read by RANS equations classes, which calculate and plot terms of ransX framework equations.

$$\overline{Q} = \frac{\overline{q}\ \Delta t_{\text{dumps}}}{\sum_{\text{dumps}} \Delta t_{\text{dumps}}} \tag{7}$$

```
eht = {}

for s in ts.ransl:
    idx = ts.ransl.index(s)
    tmp2 = []
    for i in range(ntc):
        itavg = np.where((time >= (timec[i]-tavg/2.)) & (time <= (timec[i]+tavg/2.)))
        sumdt = np.sum(dt[itavg])
        tmp1 = np.zeros(ts.rans()['nx'])
        for j in itavg[0]:
            tmp1 += np.asarray(eh[:][j][idx])*dt[j]
        tmp2.append(tmp1/sumdt)
    field = {str(s) : tmp2}
    eht.update(field)
```

Figure 5: Statistical averages as calculated by **ransx_tseries.py** and Eq.7 over time interval between $timec[i] - tavg/2$ and $timec[i] + tavg/2$. The $timec[i]$ is central time around which we can calculate statistical average over time specified by $tavg$. The eht is $\overline{Q}$, the q is loaded into a list called eh, the time between consecutives dumps is dt, the total time over which we need the statistical average is $sumdt$.

3 Implementation of ransX equations

Mean-fields RANS equations are in our framework implemented using python classes each dedicated to one equation. All the classes are stored in directory EQUATIONS. Every class has almost the same structure inheriting some methods from classes CALCULUS.py and ALIMIT.py and consisting of a constructor method *_init_*, where we initialize whole class with data read from the *npy* file and use them to construct all terms in RANS equations using various identities as shown in Section 3.5 at the end of this document. The second method plots mean-field thermodynamic quantity for which we want to see RANS equation implemented in the class. Terms in these equations are then shown by third method [1]. Sometimes, such a class contains fourth method, that calculates integral budget for each of the mean-field of the equations according to volume integral Equation 8

$$I_{\text{budget}} = \int_r 4\pi r^2 \ \overline{Q}_{\text{RANS}}/dr \tag{8}$$

Next subsections contain description of some EQUATIONs classes and implementation/calculation of RANS equations terms for hydrodynamic simulation in spherical geometry. The folder EQUATIONS contain also classes for plotting of some background quantities like temperature gradient, degeneracy or Brunt-Vaisalla frequency, but we skip those for now.

3.1 Continuity Equation with Mass FLux

Following lines describe exact mapping between actual physical mean-fields and their counterparts in the ContinuityEquationWithMassFlux.py. A snippet of the code is shown in Figure 6.

$$\begin{aligned}
\widetilde{D}_t\overline{\rho} &= -\nabla_r f_\rho + (f_\rho/\overline{\rho})\partial_r\overline{\rho} - \overline{\rho}\overline{d} \\
\partial_t\overline{\rho} + \widetilde{u}_r\partial_r\overline{\rho} &= -\nabla_r\overline{\rho' u_r'} + (\overline{\rho' u_r'}/\overline{\rho})\partial_r\overline{\rho} - \overline{\rho}\nabla_r\overline{u}_r) \\
\partial_t t_dd + ddux/dd\partial_r dd &= -\nabla_r(ddux - dd * ux) + ((ddux - dd * ux)/dd)\partial_r dd - dd\nabla_r ux) \\
\partial_t t_dd + fht_ux\ \partial_r dd &= -\nabla_r fdd + (fdd/dd)\ \partial_r dd - dd\nabla_r ux)
\end{aligned} \tag{9}$$

[1] Warning: Labels of plot lines are not adjusted for simulation in Cartesian geometry yet.

```
####################################
# CONTINUITY EQUATION WITH MASS FLUX
####################################

# LHS -dq/dt
self.minus_dt_dd = -self.dt(t_dd,xzn0,t_timec,intc)

# LHS -fht_ux Grad dd
self.minus_fht_ux_grad_dd = -fht_ux*self.Grad(dd,xzn0)

# RHS -Div fdd
self.minus_div_fdd = -self.Div(fdd,xzn0)

# RHS +fdd_o_dd gradx dd
self.plus_fdd_o_dd_gradx_dd = +(fdd/dd)*self.Grad(dd,xzn0)

# RHS -dd Div ux
self.minus_dd_div_ux = -dd*self.Div(ux,xzn0)

# -res
self.minus_resContEquation = -(self.minus_dt_dd+self.minus_fht_ux_grad_dd+self.minus_div_fdd+\
                               self.plus_fdd_o_dd_gradx_dd+self.minus_dd_div_ux)

########################################
# END CONTINUITY EQUATION WITH MASS FLUX
########################################
```

Figure 6: Continuity equation with mass flux f_ρ as programmed into ContinuityEquationWithMassFlux.py

3.2 Continuity Equation with Favrian Dilatation

Following lines describe exact mapping between actual physical mean-fields and their counterparts in the ContinuityEquationWithFavrianDilatation.py. A snippet of the code is shown in Figure 7.

$$
\begin{aligned}
\widetilde{D}_t\overline{\rho} &= -\overline{\rho}\widetilde{d} \\
\partial_t\overline{\rho} + \widetilde{u}_r\partial_r\overline{\rho} &= -\overline{\rho}\widetilde{d} \\
\partial_t t\_dd + ddux/dd\ \partial_r dd &= -\ dd * \nabla_r\ ddux/dd \\
\partial_t t\_dd + fht\_ux\ \partial_r dd &= -\ dd * \nabla_r\ fht\_ux
\end{aligned}
\tag{10}
$$

```
##############################################
# CONTINUITY EQUATION WITH FAVRIAN DILATATION
##############################################

# LHS -dq/dt
self.minus_dt_dd = -self.dt(t_dd,xzn0,t_timec,intc)

# LHS -fht_ux Grad dd
self.minus_fht_ux_grad_dd = -fht_ux*self.Grad(dd,xzn0)

# RHS -dd Div fht_ux
self.minus_dd_div_fht_ux = -dd*self.Div(fht_ux,xzn0)

# -res
self.minus_resContEquation = -(self.minus_dt_dd + self.minus_fht_ux_grad_dd + self.minus_dd_div_fht_ux)

##################################################
# END CONTINUITY EQUATION WITH FAVRIAN DILATATION
##################################################
```

Figure 7: Continuity equation with favrian dilatation $\widetilde{q}$ as programmed into ContinuityEquationWithFavrianDilatation.py

3.3 Composition Transport Equation

XtransportEquation.py

$$
\begin{aligned}
\overline{\rho}\widetilde{D}_t\widetilde{X}_i &= -\nabla_r f_i + \overline{\rho}\widetilde{\dot{X}}_i^{\text{nuc}} \\
\overline{\rho}\partial_t\widetilde{X}_i + \overline{\rho}\widetilde{u}_r\partial_r\widetilde{X}_i &= -\nabla_r\overline{\rho}\widetilde{X_i''u_r''} + \overline{\rho}\widetilde{\dot{X}}_i^{\text{nuc}} \\
\partial_t\overline{\rho}\widetilde{X}_i + \partial_r\overline{\rho}\widetilde{u}_r\widetilde{X}_i &= -\nabla_r\overline{\rho}\widetilde{X_i''u_r''} + \overline{\rho}\widetilde{\dot{X}}_i^{\text{nuc}} \\
\partial_t\overline{\rho}\widetilde{X}_i + \partial_r\overline{\rho}\widetilde{u}_r\widetilde{X}_i &= -\nabla_r\overline{\rho}(\widetilde{X_iu_r} - \widetilde{X}_i\widetilde{u_r}) + \overline{\rho}\widetilde{\dot{X}}_i^{\text{nuc}} \\
\partial_t\,\overline{\rho}\,\overline{\rho X_i}/\overline{\rho} + \partial_r\,\overline{\rho}\,\overline{\rho u_r}/\overline{\rho}\,\overline{\rho X_i}/\overline{\rho} &= -\nabla_r\left(\overline{\rho X_i u_r} - \overline{\rho X_i}\overline{\rho u_r}/\overline{\rho}\right) + \overline{\rho\dot{X}_i^{\text{nuc}}} \\
\partial_t\left(t\_dd * t\_ddxi/t\_dd\right) + \partial_r\left(dd * ddux/dd * ddxi/dd\right) &= -\nabla_r\left(ddxiux - ddxi * ddux/dd\right) + ddxidot \\
\partial_t\left(t\_dd * t\_fht\_xi\right) + \partial_r\left(dd * fht\_ux * fht\_xi\right) &= -\nabla_r fxi + ddxidot
\end{aligned}
\tag{11}
$$

3.4 Density-specific Volume Covariance

DensitySpecificVolumeCovarianceEquation.py

$$
\begin{aligned}
\overline{D}_t b &= +\overline{v}\nabla_r\overline{\rho}\overline{u_r''} - \overline{\rho}\nabla_r(\overline{u_r'v'}) + 2\overline{\rho}\overline{v'd'} \\
\partial_t b + \overline{u}_r\partial_r b &= \overline{v}\nabla_r\overline{\rho}(\overline{u}_r - \widetilde{u}_r) - \overline{\rho}\nabla_r(\overline{u_rv} - \overline{u}_r\overline{v}) + 2\overline{\rho}(\overline{vd} - \overline{v}\overline{d}) \\
\partial_t\overline{v'\rho'} + \overline{u}_r\partial_r(\overline{v'\rho'}) &= \overline{v}\nabla_r\overline{\rho}(\overline{u}_r - \widetilde{u}_r) - \overline{\rho}\nabla_r(\overline{u_rv} - \overline{u}_r\overline{v}) + 2\overline{\rho}(\overline{vd} - \overline{v}\overline{d}) \\
\partial_t(\underbrace{\overline{v\rho}}_{1} - \overline{v}\ \overline{\rho}) + \overline{u}_r\partial_r(\underbrace{\overline{v\rho}}_{1} - \overline{v}\ \overline{\rho}) &= \overline{v}\nabla_r\overline{\rho}(\overline{u}_r - \widetilde{u}_r) - \overline{\rho}\nabla_r(\overline{u_rv} - \overline{u}_r\overline{v}) + 2\overline{\rho}(\overline{vd} - \overline{v}\overline{d}) \\
-\partial_t(\overline{v}\ \overline{\rho}) - \overline{u}_r\partial_r(\overline{v}\ \overline{\rho}) &= \overline{v}\nabla_r\overline{\rho}(\overline{u}_r - \widetilde{u}_r) - \overline{\rho}\nabla_r(\overline{u_rv} - \overline{u}_r\overline{v}) + 2\overline{\rho}(\overline{vd} - \overline{v}\overline{d}) \\
\partial_t(1 - t\_sv * t\_dd) - ux\ \partial_r(1 - sv * dd) &= sv * \nabla_r dd * (ux - ddux/dd) - dd\ \nabla_r(svux - sv * ux) + 2 * dd * (svdivu - sv * divu) \\
\partial_t\ t\_b - ux\ \partial_r\ b &= sv * \nabla_r\ dd * (ux - ddux/dd) - dd\ \nabla_r(svux - sv * ux) + 2 * dd * (svdivu - sv * divu) \\
\partial_t\ t\_b - ux\ \partial_r\ b &= sv * \nabla_r\ dd * (ux - fht\_ux) - dd\ \nabla_r(svux - sv * ux) + 2 * dd * (svdivu - sv * divu)
\end{aligned}
\tag{12}
$$

3.5 Usefull Identities

$$\overline{a''} = \overline{a - \widetilde{a}} = \overline{a} - \widetilde{a} \tag{13}$$

$$\widetilde{a''b''} = \widetilde{(a - \widetilde{a}) * (b - \widetilde{b})} = \widetilde{ab} - \widetilde{a}\widetilde{b} \tag{14}$$

$$\overline{a'b'} = \overline{(a - \overline{a}) * (b - \overline{b})} = \overline{ab} - \overline{a}\,\overline{b} = \overline{a'b''} \tag{15}$$

$$\widetilde{a''b''c''} = (a - \widetilde{a}) * \widetilde{(b - \widetilde{b}) * (c - \widetilde{c})} = \widetilde{abc} - \widetilde{a}\widetilde{bc} - \widetilde{b}\widetilde{ac} - \widetilde{c}\widetilde{ab} + 2\widetilde{a}\widetilde{b}\widetilde{c} \tag{16}$$

$$\overline{a'b'c''} = \overline{(a - \overline{a}) * (b - \overline{b}) * (c - \widetilde{c})} = \overline{abc} - \overline{ac}\;\overline{b} - \overline{a}\;\overline{bc} + \overline{a}\;\overline{b}\;\overline{c} - \overline{ab}\;\widetilde{c} + \overline{a}\;\overline{b}\widetilde{c} \tag{17}$$

$$\overline{a''b'c''} = \overline{(a - \widetilde{a}) * (b - \overline{b}) * (c - \widetilde{c})} = \overline{abc} - \overline{ac}\overline{b} - \widetilde{a}\overline{bc} + \widetilde{a}\overline{b}\overline{c} - \overline{ab}\widetilde{c} + \overline{a}\overline{b}\widetilde{c} \tag{18}$$

$$\overline{a''bc} = \overline{(a - \widetilde{a})bc} = \overline{abc} - \widetilde{a}\overline{bc} \tag{19}$$

$$\overline{a''\partial_r b'} = \overline{(a - \widetilde{a})\partial_r b'} = \overline{a\partial_r b'} - \cancelto{0}{\widetilde{a}\partial_r\overline{b'}} = \overline{a\partial_r b} - \overline{a}\partial_r\overline{b} \tag{20}$$

Bibliography

Didier Besnard, FH Harlow, RM Rauenzahn, and C Zemach. Turbulence transport equations for variable-density turbulence and their relationship to two-field models. Technical report, Los Alamos National Lab., NM (United States), 1992.

P. Chassaing, R.A. Antonia, F. Anselmet, L. Joly, and R. Sarkar. *Variable Density Fluid Turbulence*. Kluwer Academic Publishers, 2010.

C. A. Meakin and D. Arnett. Turbulent Convection in Stellar Interiors. I. Hydrodynamic Simulation. *Apj*, 667:448–475, September 2007. doi: 10.1086/520318.

M. Viallet, C. Meakin, D. Arnett, and M. Mocák. Turbulent Convection in Stellar Interiors. III. Mean-field Analysis and Stratification Effects. *Apj*, 769:1, May 2013. doi: 10.1088/0004-637X/769/1/1.