

1 Non-linear contrubution

Martin: We can look at $k = 10$

Just note that, if we dont run the GEvolution from enough large scale, we lose the long mode modulation and the perturbation on quadratic variables would change like $\chi ..$, and on the other hand also we have the large power from non linearities of scalar field. Here we write the full equations and try to solve them numerically in Gevolution, and if we got something interesting or strange we need to solve them in mathematica to see if we did not have make a mistake !

$$\begin{aligned} \pi'' + \mathcal{H}(1 - 3w)\pi' + 3\mathcal{H}\left(-c_s^2 + w\right)\Psi - \Psi' - 3c_s^2\Phi' + \left(3\mathcal{H}^2(c_s^2 - w) + \mathcal{H}'(1 - 3c_s^2)\right)\pi \\ - c_s^2\nabla^2\pi - 2c_s^2\Phi\nabla^2\pi + (1 - c_s^2)\Psi\nabla^2\pi + 3c_s^2\mathcal{H}(1 + w)\pi\nabla^2\pi - (1 - c_s^2)(\mathcal{H}\pi + \pi')\nabla^2\pi \\ + c_s^2\nabla\Phi.\nabla\pi - (2c_s^2 - 1)\nabla\Psi.\nabla\pi + \frac{\mathcal{H}}{2}\left(2 + 3w + c_s^2\right)\nabla\pi.\nabla\pi - 2(1 - c_s^2)\nabla\pi.\nabla(\mathcal{H}\pi + \pi') = 0 \end{aligned} \quad (1)$$

$$\begin{aligned} T_0^0(Gev) &= \Omega_{kess}^0 a^{-3w} \left[1 + \frac{1+w}{c_s^2} \left(-3\mathcal{H}c_s^2\pi - \Psi + (\pi' + \mathcal{H}\pi) - (1 - 2c_s^2)\frac{(\vec{\nabla}\pi)^2}{2} \right) \right] \\ T_0^i(Gev) &= -\Omega_{kess}^0 a^{-3w} (1+w) \left[1 - \left(\frac{1}{c_s^2} - 1 \right) \frac{(\vec{\nabla}\pi)^2}{2} \right] \partial_i\pi \\ T_j^i(Gev) &= w\Omega_{kess}^0 a^{-3w} \left(1 + \frac{1+w}{w} \left[-3\mathcal{H}w\pi - \Psi + (\pi' + \mathcal{H}\pi) - \frac{(\vec{\nabla}\pi)^2}{2} \right] \delta_j^i + \frac{1+w}{w} \delta^{ik} \partial_k\pi \partial_j\pi \right) \end{aligned} \quad (2)$$

we take π and ζ defined as following as new set of variables,

$$\zeta \doteq -\Psi + \pi' + \mathcal{H}\pi, \quad (3)$$

After substitution $\pi' \rightarrow \zeta + \Psi - \mathcal{H}\pi$ we get the following expression for stress tensor according to mathematica, which is very clear,

$$\begin{aligned} \text{T00} &= \frac{1+w}{cs2} (-3 H \text{cs2} \pi\pi[x, t] - \Psi + \pi\text{prime} + H \pi\pi[x, t] - (1 - 2 \text{cs2}) (\partial_x \pi\pi[x, t] \partial_x \pi\pi[x, t])); \\ \pi\text{prime} &= \zeta + \Psi - H \pi\pi[x, t] \\ \zeta &= \zeta + \Psi - H \pi\pi[x, t] \\ \text{T00} &= \frac{(1+w) (\zeta - 3 \text{cs2} H \pi\pi[x, t] - (1 - 2 \text{cs2}) \pi\pi^{(1,0)}[x, t]^2)}{cs2} \end{aligned}$$

So the stress tensor reads,

$$\begin{aligned} T_0^0(Gev) &= \Omega_{kess}^0 a^{-3w} \left[1 + \frac{1+w}{c_s^2} \left(\zeta - 3\mathcal{H}c_s^2\pi - (1-2c_s^2)\frac{(\vec{\nabla}\pi)^2}{2} \right) \right] \\ T_0^i(Gev) &= -\Omega_{kess}^0 a^{-3w} (1+w) \left[1 - \left(\frac{1}{c_s^2} - 1 \right) \frac{(\vec{\nabla}\pi)^2}{2} \right] \partial_i \pi \\ T_j^i(Gev) &= w \Omega_{kess}^0 a^{-3w} \left(1 + \frac{1+w}{w} \left[-3\mathcal{H}w\pi + \zeta - \frac{(\vec{\nabla}\pi)^2}{2} \right] \delta_j^i + \frac{1+w}{w} \delta^{ik} \partial_k \pi \partial_j \pi \right) \end{aligned} \quad (4)$$

After substitution in mathematica we get,

$$\begin{aligned} \zeta' - (\mathcal{H}\pi)' + \Psi' + \mathcal{H}(1-3w)(\zeta - \mathcal{H}\pi + \Psi) + 3\mathcal{H}(-c_s^2 + w)\Psi - \Psi' - 3c_s^2\Phi' + (3\mathcal{H}^2(c_s^2 - w) + \mathcal{H}'(1-3c_s^2))\pi \\ - c_s^2\nabla^2\pi - 2c_s^2\Phi\nabla^2\pi + (1-c_s^2)\Psi\nabla^2\pi + 3c_s^2\mathcal{H}(1+w)\pi\nabla^2\pi - (1-c_s^2)(\zeta + \Psi)\nabla^2\pi \\ + c_s^2\nabla\Phi.\nabla\pi - (2c_s^2 - 1)\nabla\Psi.\nabla\pi + \frac{\mathcal{H}}{2}(2+3w+c_s^2)\nabla\pi.\nabla\pi - 2(1-c_s^2)\nabla\pi.\nabla(\zeta + \Psi) = 0 \end{aligned} \quad (5)$$

The linear part simplification gives the same equation we had before and we can rewrite the equation as following,

$$\begin{aligned} \zeta' - 3w\mathcal{H}\zeta + 3c_s^2(\mathcal{H}^2 - \mathcal{H}')\pi - 3c_s^2(\Phi' + \mathcal{H}\Psi) - c_s^2\nabla^2\pi \\ - 2c_s^2\Phi\nabla^2\pi + (1-c_s^2)\Psi\nabla^2\pi + 3c_s^2\mathcal{H}(1+w)\pi\nabla^2\pi - (1-c_s^2)(\zeta + \Psi)\nabla^2\pi \\ + c_s^2\nabla\Phi.\nabla\pi - (2c_s^2 - 1)\nabla\Psi.\nabla\pi + \frac{\mathcal{H}}{2}(2+3w+c_s^2)\nabla\pi.\nabla\pi - 2(1-c_s^2)\nabla\pi.\nabla(\zeta + \Psi) = 0 \end{aligned} \quad (6)$$

The other equation is like before,

$$\pi' = \zeta + \Psi - \mathcal{H}\pi \quad (7)$$

We actually did nothing except substituting the π' and in non-linear part the substitution is very straightforward. Just note that in Gevolution to compute $\nabla\pi.\nabla(\zeta + \Psi)$ we use the symmetric derivative as following,

$$\nabla\pi.\nabla(\zeta + \Psi) = \frac{1}{4dx^2} \sum_i [\pi(x_i + 1) - \pi(x_i - 1)] [(\zeta(x_i+1) + \Psi(x_i+1)) - (\zeta(x_i-1) + \Psi(x_i-1))] \quad (8)$$

where "1/4" coefficient appear since we are using symmetric derivative and using points with distance two. " x_i " is the lattice coordinate.

1.1 Solving the equation by leap-frog method

Here we explain how we solve the two equations using leap-frog method,

1.1.1 π equation

for the π equation we have,

$$\pi_{n+1} = \pi_n + \pi'_{n+\frac{1}{2}} \Delta\tau \quad (9)$$

$$\pi'_{n+\frac{1}{2}} = \zeta_{n+\frac{1}{2}} - \mathcal{H}_{n+\frac{1}{2}}\pi_{n+\frac{1}{2}} + \Psi_{n+\frac{1}{2}} \quad (10)$$

According to the definition, since the equation for updating π is linear we do not have any trouble with non-linear terms and this part is done like linear equations.

The background part shoule be updated before this step to have a_{kess} at $(n+1/2)$ and then having $\mathcal{H}(n+1/2)$. Just note that making ζ updating at half steps help us because we need all the terms in $\zeta'(n)$ at integer steps.

Moreover we update ζ first, so we have it in $\zeta_{n+1/2}$ writing,

$$\pi_{n+\frac{1}{2}} = \frac{\pi_{n+1} + \pi_n}{2} \quad (11)$$

and the same for all the other variables at step $n + \frac{1}{2}$.

For π we have,

$$\pi_{n+1} = \pi_n + \Delta\tau \left[\zeta_{n+\frac{1}{2}} - \mathcal{H}_{n+\frac{1}{2}} \left(\frac{\pi_{n+1} + \pi_n}{2} \right) + \Psi_{n+\frac{1}{2}} \right] \quad (12)$$

$$\pi_{n+1} = \frac{1}{1 + \mathcal{H}_{n+\frac{1}{2}} \Delta\tau / 2} \left[\pi_n + \Delta\tau \left[\zeta_{n+\frac{1}{2}} - \mathcal{H}_{n+\frac{1}{2}} \frac{\pi_n}{2} + \Psi_{n+\frac{1}{2}} \right] \right] \quad (13)$$

As it is clear from the formula we don't have access to the $\Psi_{n+\frac{1}{2}}$, so we use the extrapolation to have them in next half steps,

$$\Psi_{n+\frac{1}{2}} = \Psi_n + \Psi'_n \frac{d\tau}{2} \quad (14)$$

Moreover to have Ψ'_n we use the following formula by saving Ψ at two different steps!

$$\Psi'_n = \frac{\Psi_n - \Psi_{n-1}}{d\tau} \quad (15)$$

1.1.2 ζ equation

We chose to update ζ at half steps since then we need its derivative at integer steps and make the computation simplified. Moreover first ζ is updated to have it at step $n + 1/2$, then π is updated to get the values at $n + 1$, just note that although we need ζ at integer steps to be synchronized with particles and π , but since getting ζ_{n+1} makes the ways very complicated we just use the following formula to obtain it, if it is really necessary! Since we guess that putting $\zeta_{n+1/2}$ as ζ_{n+1} in stress tensor is enough.

$$\zeta_{n+1} = \zeta_{n+\frac{1}{2}} + \zeta'_n \Delta\tau / 2 \quad (16)$$

ζ'_n in the equation 25 reads from the differential equation as following,

$$\zeta'_n = 3\mathcal{H}_n(w\zeta_n + c_s^2\Psi_n) - c_s^2(3\mathcal{H}_n^2 - 3\mathcal{H}'_n)\pi_n + 3c_s^2\Phi'_n + c_s^2\nabla^2\pi_n \quad (17)$$

$$\begin{aligned} &+ 2c_s^2\Phi_n\nabla^2\pi_n - (1 - c_s^2)\Psi_n\nabla^2\pi_n - 3c_s^2\mathcal{H}_n(1 + w)\pi_n\nabla^2\pi_n + (1 - c_s^2)(\zeta_n + \Psi_n)\nabla^2\pi_n \\ &- c_s^2\nabla\Phi_n.\nabla\pi_n + (2c_s^2 - 1)\nabla\Psi_n.\nabla\pi_n - \frac{\mathcal{H}_n}{2}(2 + 3w + c_s^2)\nabla\pi_n.\nabla\pi_n + 2(1 - c_s^2)\nabla\pi_n.\nabla(\zeta_n + \Psi_n) \end{aligned} \quad (18)$$

Since we need $\nabla(\zeta_n + \Psi_n)$ we write $\zeta_n = \frac{\zeta_{n+1/2} + \zeta_{n-1/2}}{2}$

$$\begin{aligned}
\zeta_{n+\frac{1}{2}} = & \zeta_{n-\frac{1}{2}} + \Delta\tau \left[3\mathcal{H}_n(w\frac{\zeta_{n+\frac{1}{2}} + \zeta_{n-\frac{1}{2}}}{2} + c_s^2\Psi_n) - c_s^2(3\mathcal{H}_n^2 - 3\mathcal{H}'_n)\pi_n + 3c_s^2\Phi'_n + c_s^2\nabla^2\pi_n \right. \\
& + 2c_s^2\Phi_n\nabla^2\pi_n - (1 - c_s^2)\Psi_n\nabla^2\pi_n - 3c_s^2\mathcal{H}_n(1 + w)\pi_n\nabla^2\pi_n + (1 - c_s^2)\left(\frac{\zeta_{n+1/2} + \zeta_{n-1/2}}{2} + \Psi_n\right)\nabla^2\pi_n \\
& - c_s^2\nabla\Phi_n.\nabla\pi_n + (2c_s^2 - 1)\nabla\Psi_n.\nabla\pi_n - \frac{\mathcal{H}_n}{2}(2 + 3w + c_s^2)\nabla\pi_n.\nabla\pi_n \\
& \left. + 2(1 - c_s^2)\nabla\pi_n.\nabla\left(\frac{\zeta_{n+1/2} + \zeta_{n-1/2}}{2} + \Psi_n\right) \right] \tag{19}
\end{aligned}$$

As one can see the term $\nabla\left(\frac{\zeta_{n+1/2} + \zeta_{n-1/2}}{2} + \Psi_n\right)$ is somehow problematic, since we cannot factorize the $\zeta_{n+1/2}$ and $\zeta_{n-1/2}$ simply or its better to have a field ζ_{int} which lives in the integer steps and in principle we ζ_n and we can use predictor-corrector method to improve this value. Here we use the predictor-corrector method to solve this equation.

As the first guess we take $\zeta_n \approx \zeta_{n-1/2}$, which means we have neglected the term $\zeta'_{n-1/2}d\tau/2$, then we get the following equation,

$$\begin{aligned}
\zeta_{n+\frac{1}{2}} = & \zeta_{n-\frac{1}{2}} + \Delta\tau \left[3\mathcal{H}_n(w\frac{\zeta_{n+\frac{1}{2}} + \zeta_{n-\frac{1}{2}}}{2} + c_s^2\Psi_n) - c_s^2(3\mathcal{H}_n^2 - 3\mathcal{H}'_n)\pi_n + 3c_s^2\Phi'_n + c_s^2\nabla^2\pi_n \right. \\
& + 2c_s^2\Phi_n\nabla^2\pi_n - (1 - c_s^2)\Psi_n\nabla^2\pi_n - 3c_s^2\mathcal{H}_n(1 + w)\pi_n\nabla^2\pi_n + (1 - c_s^2)\left(\frac{\zeta_{n+1/2} + \zeta_{n-1/2}}{2} + \Psi_n\right)\nabla^2\pi_n \\
& - c_s^2\nabla\Phi_n.\nabla\pi_n + (2c_s^2 - 1)\nabla\Psi_n.\nabla\pi_n - \frac{\mathcal{H}_n}{2}(2 + 3w + c_s^2)\nabla\pi_n.\nabla\pi_n \\
& \left. + 2(1 - c_s^2)\nabla\pi_n.\nabla\left(\zeta_{n-1/2} + \Psi_n\right) \right] \tag{20}
\end{aligned}$$

Factorizing $\zeta_{n+1/2}$ gives,

$$\begin{aligned}
\zeta_{n+\frac{1}{2}} \left[1 - 3\mathcal{H}_n w \Delta\tau / 2 - (1 - c_s^2) \nabla^2\pi_n \Delta\tau / 2 \right] = & \zeta_{n-\frac{1}{2}} + \Delta\tau \left[3\mathcal{H}_n(w\frac{\zeta_{n-\frac{1}{2}}}{2} + c_s^2\Psi_n) - c_s^2(3\mathcal{H}_n^2 - 3\mathcal{H}'_n)\pi_n + 3c_s^2\Phi'_n \right. \\
& + c_s^2\nabla^2\pi_n + 2c_s^2\Phi_n\nabla^2\pi_n - (1 - c_s^2)\Psi_n\nabla^2\pi_n - 3c_s^2\mathcal{H}_n(1 + w)\pi_n\nabla^2\pi_n + (1 - c_s^2)\left(\frac{\zeta_{n-1/2}}{2} + \Psi_n\right)\nabla^2\pi_n \\
& - c_s^2\nabla\Phi_n.\nabla\pi_n + (2c_s^2 - 1)\nabla\Psi_n.\nabla\pi_n - \frac{\mathcal{H}_n}{2}(2 + 3w + c_s^2)\nabla\pi_n.\nabla\pi_n \\
& \left. + 2(1 - c_s^2)\nabla\pi_n.\nabla\left(\zeta_{n-1/2} + \Psi_n\right) \right] \tag{21}
\end{aligned}$$

Finally we can write the below equation,

$$\begin{aligned} \zeta_{n+\frac{1}{2}} = & \frac{1}{1 - 3\mathcal{H}_n w \Delta\tau/2 - (1 - c_s^2) \nabla^2 \pi_n \Delta\tau/2} \left[\zeta_{n-\frac{1}{2}} + \Delta\tau \left(3\mathcal{H}_n \left(w \frac{\zeta_{n-\frac{1}{2}}}{2} + c_s^2 \Psi_n \right) - c_s^2 (3\mathcal{H}_n^2 - 3\mathcal{H}'_n) \pi_n + 3c_s^2 \Phi'_n \right. \right. \\ & + c_s^2 \nabla^2 \pi_n + 2c_s^2 \Phi_n \nabla^2 \pi_n - (1 - c_s^2) \Psi_n \nabla^2 \pi_n - 3c_s^2 \mathcal{H}_n (1 + w) \pi_n \nabla^2 \pi_n + (1 - c_s^2) \left(\frac{\zeta_{n-1/2}}{2} + \Psi_n \right) \nabla^2 \pi_n \\ & - c_s^2 \nabla \Phi_n \cdot \nabla \pi_n + (2c_s^2 - 1) \nabla \Psi_n \cdot \nabla \pi_n - \frac{\mathcal{H}_n}{2} (2 + 3w + c_s^2) \nabla \pi_n \cdot \nabla \pi_n \\ & \left. \left. + 2(1 - c_s^2) \nabla \pi_n \cdot \nabla (\zeta_{n-1/2} + \Psi_n) \right) \right] \end{aligned} \quad (22)$$

which at linear order we get the same equation as we had before,

$$\zeta_{n+\frac{1}{2}} = \frac{1}{1 - 3\mathcal{H}_n w \Delta\tau/2} \left[\zeta_{n-\frac{1}{2}} + \Delta\tau \left(3\mathcal{H}_n \left(\frac{w \zeta_{n+\frac{1}{2}}}{2} + c_s^2 \Psi_n \right) - c_s^2 (3\mathcal{H}_n^2 - 3\mathcal{H}'_n) \pi_n + 3c_s^2 \Phi'_n + c_s^2 \nabla^2 \pi_n \right) \right] \quad (23)$$

It is very important to check everything for the first loop specifically! For the first loop we first compute $\zeta^{(-1/2)}$ by $\zeta_{-1/2} = \zeta_0 - \zeta'_0 d\tau/2$ and then ζ is updated to have $\zeta^{(1/2)}$ from the values of $\Psi^{(0)}$ but we take $\Phi'^{(0)} = 0$ which is an approximation in our scheme.

Then we go through the procedure which is the same for all loops (first loop included), which we update π to have it at step 1 from the $\zeta_{1/2}$, during all of these procedures, Φ is assumed to be constant and we use $\Phi^{(1/2)} = \Phi^{(0)}$ since in the first loop Φ' is zero! So this is an approximation that in the first loop we take $\Psi' = 0$ and use the same Φ at 1/2 and 0 step the same. But in the other loops everything seems correct as following,

To compute $\zeta_{-1/2}$ we just use the linear equation for ζ'_0 and also neglect Φ'_0 .

The whole updating loop is as following,

Note that here we can, not assign any ζ in integer steps, since we think it is not necessary, but if we want to calculate it we need to define a new field ζ_{int} which lives on the integer steps and is obtained by

$$\zeta_{int}(n+1) = \zeta_{n+1/2} + \zeta'_{int} d\tau/2 \quad (24)$$

Where ζ'_{int} contains ζ_{int} which comes from the corrected value of ζ_{int} .

1.1.2.1 Predictor-corrector method

We need this method only when we do not have the values of the field at $\zeta(n)$ and we guess $\nabla \zeta(n)$ by $\nabla \zeta(n - 1/2)$ and then we correct it! This is something that can be corrected, but when we are assigning the $\zeta(n)$ by previous loop $\zeta(n) = \zeta_{n-1/2} + \zeta'_{n-1} d\tau$, so we have it and we do not need any corrector method!

This is an important part which should be done to have the errors propagation under the control. As we have explained already in the first loop of updating we take $\nabla \zeta_n = \nabla \zeta_{n-1/2}$ which is just prediction and then we calculate $\zeta_{n+1/2}$ from the predictor. So at the end we have $\zeta_{n+1/2}$ which we did not have in the beginning, so we are going to use $\zeta_{n+1/2}$ as new value to compute ζ_n by taking the average between the two and then again we calculate new corrected ζ_n and we take it as initial value to compute the corrected $\zeta_{n+1/2}$.

This one really does not work! Since we are using the value of ζ_n , if we just use $\nabla\zeta_{n+1/2}$ and then we update it by the differential equation, it works well. At the end since we are using ζ_{int}

Moreover at the same time from the values of ζ_n and ζ'_n we compute ζ_{n+1} to be synched with particles and ... Although it may not improve very much or the stress tensor is not sensitive to this, but still we need to define such a field at integer steps because of predictor-corrector method, so why not we compute ζ_{n+1} too!

The best way to do it is first to check that $\zeta_{n+1/2}$ from the next order corrector is near to original one (like the relative error $2\frac{\zeta_1 - \zeta_2}{\zeta_1 + \zeta_2} < 0.01$, which gives less than 1% error).

The important point is that, if we want to solve it systematically we really need to define ζ_{int} which is the integer ζ , because at the end we want to compute $\nabla\zeta_n$ which is only possible when we define ζ at integer steps, otherwise we do not have access to the $\zeta_n(x+1)$ or we must define 6 variables for it, $\zeta_n(x+0), \zeta_n(x+1)...$. We prefer to define a new field ζ which lives in integer steps because it has the advantage of having it at step (n+1) which is synchronized with the π .

To summerize the predictor-corrector method.

- Calculating $\zeta_{n+1/2}$ from predictor $\nabla\zeta_{n-1/2}$
- using new $\zeta_{n+1/2}$ and $\zeta_{n-1/2}$ to calculate $\zeta_{int}(n)$ by taking average and again calculate $\nabla\zeta_n$ and calculate new $\zeta_{n+1/2}$
-
-
- continue to get the precise enough results
- at the end from the last values at step n compute ζ'_n and from last $\zeta_{n+1/2}$, compute ζ_{n+1} and save as ζ_{int} .

Approximations:

The important approximations here are:

- 1- At the first loop we take $\Phi' = 0$
- 2- To compute $\zeta_{-1/2}$ we neglect non-linear terms, but it should not be important, since we use predictor corrector method in other loops, which correct the values.
- 4- All the top approximations can be suppressed by increasing the number of kessence update or decreasing the time stepping of Gevolution.
- 5- We use predictor-corrector method for couple of times in each loop to correct our bad guess of non-linear term $\nabla\zeta_n$

1.1.3 The loop

- First we provide the initial condition for $\zeta(0), \pi(0), \Psi(0)$
- Then we update ζ by minus half step from the values at step 0 to get $\zeta(-1/2)$, just at the first loop!
- So we are ready to go over the general loop, we first update ζ to have the value at $\zeta(1/2)$ from $\zeta(-1/2)$ and $\Phi(0), \pi(0)$, we compute $\zeta(1) = \zeta(1/2) + \zeta'(0) * d\tau$.

- we compute ζ_1 from the corrected values and by $\zeta_1 = \zeta_{1/2} + \zeta'_0$
- Then we compute $\pi(1)$, from $\pi(0)$ and $\zeta'(1/2)$, $\Phi'(1/2)$ and $\Phi(1/2)$,

So in general, we have,

- First we compute $\zeta_{n+1/2} = \zeta_{n-1/2} + \zeta'(n)$, where in $\zeta'(n)$ we have everything at step n which we know the values!

$$\zeta_{n+\frac{1}{2}} = \zeta_n + \zeta'_n \Delta\tau \quad (25)$$

ζ'_n in the equation 25 reads from the differential equation as following,

$$\zeta'_n = 3\mathcal{H}_n(w\zeta_n + c_s^2\Psi_n) - c_s^2(3\mathcal{H}_n^2 - 3\mathcal{H}'_n)\pi_n + 3c_s^2\Phi'_n + c_s^2\nabla^2\pi_n \quad (26)$$

$$\begin{aligned} &+ 2c_s^2\Phi_n\nabla^2\pi_n - (1 - c_s^2)\Psi_n\nabla^2\pi_n - 3c_s^2\mathcal{H}_n(1 + w)\pi_n\nabla^2\pi_n + (1 - c_s^2)(\zeta_n + \Psi_n)\nabla^2\pi_n \\ &- c_s^2\nabla\Phi_n.\nabla\pi_n + (2c_s^2 - 1)\nabla\Psi_n.\nabla\pi_n - \frac{\mathcal{H}_n}{2}(2 + 3w + c_s^2)\nabla\pi_n.\nabla\pi_n + 2(1 - c_s^2)\nabla\pi_n.\nabla(\zeta_n + \Psi_n) \end{aligned} \quad (27)$$

Where we only need $\Phi(n)$, $\Phi'(n)$, $\pi(n)$, $\zeta(n) = \frac{\zeta(n-1/2)+\zeta(n+1/2)}{2}$. We have all the quantities simply at step n, so we do not need any extrapolation and etc.

So by $\zeta(n) = \frac{\zeta(n-1/2)+\zeta(n+1/2)}{2}$ we again can factorize and for the $\nabla(\frac{\zeta(n-1/2)+\zeta(n+1/2)}{2})$ which cannot be factorized we again use predictor-corrector method by guessing in the first loop that it is simply $\nabla\zeta_{(n-1/2)/2}$, or it is better to use the values of ζ at integer steps, which is approximately correct but we can make it better by predictor-corrector method.

- Then we compute $\zeta(n + 1)$ which is synchronized with everything by $\zeta(n + 1) = \zeta(n) + \zeta'(n)d\tau$, we have defined two fields one is ζ_{int} and the other ζ_{half} . But note that it is better to update it by $\zeta_{n+1} = \zeta_{n+1/2} + \zeta'_{n+1/2}d\tau/2$, but in this case we need to have $\Phi'(n+1/2) = \frac{\Phi(n+1)-\Phi(n)}{d\tau}$ and of course we need $\Phi(n+1)$ which we don't have access to.
- Background is updated, $a_{kess}(n + 1/2)$ since we need it for updating π .
- After updating $\zeta(n + 1/2)$, we update π from n to n+1 as following,

$$\pi_{n+1} = \pi_n + \pi'_{n+\frac{1}{2}}\Delta\tau \quad (28)$$

$$\pi'_{n+\frac{1}{2}} = \zeta_{n+\frac{1}{2}} - \mathcal{H}_{n+\frac{1}{2}}\pi_{n+\frac{1}{2}} + \Psi_{n+\frac{1}{2}} \quad (29)$$

According to the definition, since the equation for updating π is linear we do not have any trouble with non-linear terms and this part is done like linear equations.

The background part is better to be updated before this step to have a_{kess} at (n+1/2) and then having $\mathcal{H}(n + 1/2)$. Just note that making ζ updating at half steps help us because we need all the terms in $\zeta'(n)$ at integer steps.

Since the scalar field Stress energy tensor must be synchronized with particles stress tensor, we need to have all the variables at the same step which is n, so we need to write all the terms at step $n + \frac{1}{2}$ in terms of the values at step n and n+1 as following,

of course except ζ which we have it at $n + 1/2$. The easiest model to calculate $F_{n+\frac{1}{2}}$ is by taking average of the next and last step, so

$$\pi_{n+\frac{1}{2}} = \frac{\pi_{n+1} + \pi_n}{2} \quad (30)$$

and the same for all the other variables at step $n + \frac{1}{2}$.

For π we have,

$$\pi_{n+1} = \frac{1}{1 + \mathcal{H}_{n+\frac{1}{2}} \Delta\tau / 2} \left[\pi_n + \Delta\tau \left[\zeta_{n+\frac{1}{2}} - \mathcal{H}_{n+\frac{1}{2}} \frac{\pi_n}{2} + \Psi_{n+\frac{1}{2}} \right] \right] \quad (31)$$

As it is clear from the formula we don't have access to the $\Psi_{n+\frac{1}{2}}$ and Ψ_{n+1} , so we use the extrapolation to have them in next half steps,

$$\Psi_{n+\frac{1}{2}} = \Psi_n + \Psi'_n \frac{d\tau}{2} \quad (32)$$

Moreover to have Ψ'_n we use the following formula by saving Ψ at two different steps!

$$\Psi'_n = \frac{\Psi_n - \Psi_{n-1}}{d\tau} \quad (33)$$

As it is clear we do not have any trouble with π updating too, while in previous version of updating we were updating wrongly and ζ was ahead of potentials and we were using the wrong value for potentials.

- One point: since we are using ζ_{int} so we have the value of ζ at step (n), so we basically we do not need the predictor-corrector method since we have what we want, but we have this code for predictor method just to make sure that the approximation by taking the average is good enough, otherwise we need to go over loop for some time to reach our desired precision.
- We turned off the corrector part, since we are doing everything clearly without any approximation in the non-linear part, so we do not need corrector method.
- At the end we need to test if our approximations work well? which is done in the next chapters by comparing the results versus mathematica and class.

1.2 Gevolution code

For the π update we have linear equation and the code part is straightforward,

```
template <class FieldType>
void update_pi_k( double dtau, double dx, double a, Field<FieldType> & phi, Field<
    FieldType> & phi_old, Field<FieldType> & chi, Field<FieldType> & chi_old, Field<
    FieldType> & pi_k, Field<FieldType> & zeta_integer, Field<FieldType> &
    zeta_half, double Omega_fld, double w, double cs2, double Hcon, double
    H_prime, int non_linearity)
{
    double psi, psi_prime, psi_half;
    double Coeff1 = 1. / (1. + Hcon * dtau / 2.);
    Site x(phi.lattice());
    for (x.first(); x.test(); x.next())
    {
        psi=phi(x) - chi(x); //psi(n)
        psi_prime= ((phi(x) - chi(x)) - (phi_old(x) - chi_old(x))) / dtau; //psi'(n)
        psi_half= psi + psi_prime * dtau / 2.; //psi_half (n+1/2) = psi(n) + psi_prime'(n) dtau / 2
        //***** Updating which is linear by definition
        //pi Updating which is linear by definition
    }
}
```

```

//*****
pi_k(x) = Coeff1 * ( pi_k(x) + dtau * ( zeta_half(x) - Hcon * pi_k(x)/2. + psi_half ) ); //*
pi_k(n+1)
//*****
}
}

```

For the stress tensor part we add the following part to the Gevolution.hpp,

```

}
template <class FieldType>
void projection_Tmuu_kessence( Field<FieldType> & T00, Field<FieldType> & T0i, Field<FieldType> & Tij,
double dx, double a, Field<FieldType> & phi, Field<FieldType> & phi_old, Field<FieldType> & chi, Field<
Field<FieldType> & pi_k, Field<FieldType> & zeta_integer, double Omega_fld , double w, double cs2, double
Hcon, double fourpig, int non_linearity ,int method )
{
    Site xField(phi.lattice());
    double coeff1, coeff2, coeff3, Hdot, psi;
    double gradientpi_squared, Dx_pi_Dx_pi, Dx_pi_Dy_pi, Dx_pi_Dz_pi, Dy_pi_Dy_pi, Dy_pi_Dz_pi,
    Dz_pi_Dz_pi;
    Site x(phi.lattice());
    double gradient_pi2;
    coeff1= Omega_fld * pow(a, -3. * w) * (1. + w) / (cs2);
    coeff2= Omega_fld * pow(a, -3. * w) * (1. + w);
    //*****
    //All non-linear terms are zero, except we have non-linearities
    //*****
    gradientpi_squared=0;
    Dx_pi_Dx_pi=0;
    Dx_pi_Dy_pi=0;
    Dx_pi_Dz_pi=0;
    Dy_pi_Dy_pi=0;
    Dy_pi_Dz_pi=0;
    Dz_pi_Dz_pi=0;
    for (xField.first(); xField.test(); xField.next())
    {
        if (non_linearity ==1)
        {
            //*****
            //((D_i pi)^2
            //*****
            gradientpi_squared =0.25*(pi_k(xField+0) - pi_k(xField - 0))*(pi_k(xField + 0) - pi_k(xField-0))
            /(dx*dx);
            gradientpi_squared+=0.25*(pi_k(xField+1) - pi_k(xField - 1))*(pi_k(xField + 1) - pi_k(xField-1))
            /(dx*dx);
            gradientpi_squared+=0.25*(pi_k(xField+2) - pi_k(xField - 2))*(pi_k(xField + 2) - pi_k(xField-2))
            /(dx*dx);
            //*****
            //((X,X) ::::: > Dx_pi_Dx_pi = GradX(pi).GradX(pi)
            //*****
            Dx_pi_Dx_pi =0.25*(pi_k(xField+0) - pi_k(xField-0))*(pi_k(xField+0) - pi_k(xField-0))/(dx*dx);
            //*****
            //((X,Y) ::::: > Dx_pi_Dy_pi = GradX(pi).GradY(pi) = GradY(pi).GradX(pi)
            //*****
            Dx_pi_Dy_pi =0.25*(pi_k(xField+0) - pi_k(xField-0))*(pi_k(xField+1) - pi_k(xField-1))/(dx*dx);
            //*****
            //((X,Z) ::::: > Dx_pi_Dz_pi = GradX(pi).GradZ(pi)
            //*****
            Dx_pi_Dz_pi =0.25*(pi_k(xField+0) - pi_k(xField-0))*(pi_k(xField+2) - pi_k(xField-2))/(dx*dx);
            //*****
            //((Y,Y) ::::: > Dy_pi_Dy_pi = GradY(pi).GradY(pi)
            //*****
            Dy_pi_Dy_pi =0.25*(pi_k(xField+1) - pi_k(xField-1))*(pi_k(xField+1) - pi_k(xField-1))/(dx*dx);
            //*****
            //((Y,Z) ::::: > Dy_pi_Dz_pi = Grady(pi).Gradz(pi)
            //*****
            Dy_pi_Dz_pi =0.25*(pi_k(xField+1) - pi_k(xField-1))*(pi_k(xField+2) - pi_k(xField-2))/(dx*dx);
            //*****
            //((Z,Z) ::::: > Dz_pi_Dz_pi = Gradz(pi).Gradz(pi)
            //*****
            Dz_pi_Dz_pi =0.25*(pi_k(xField+2) - pi_k(xField-2))*(pi_k(xField+2) - pi_k(xField-2))/(dx*dx);
        }
        //*****
        psi= phi(xField) - chi(xField);
        //*****
        //STRESS TENSOR COMPONENTS
        //*****
        // 0-0-component: (Time,Time)
        T00(xField) = - coeff1 * ( -3. * cs2 * Hcon * pi_k(xField) + zeta_integer(xField)
        /*Non-linear*/ - non_linearity * (1. - 2. * cs2) * gradientpi_squared / 2. );
        //*****
        // 1-1-component: (X,X)
        Tij(xField, 0, 0) = + coeff2 * (-3.* w * Hcon* pi_k(xField) + zeta_integer(xField)
        /*Non-linear*/ - non_linearity * (gradientpi_squared / 2. + Dx_pi_Dx_pi) );
        //*****
        // 2-2-component: (Y,Y)
        Tij(xField, 1, 1) = + coeff2 * (-3.* w * Hcon* pi_k(xField) + zeta_integer(xField)
        /*Non-linear*/ - non_linearity * (gradientpi_squared / 2. + Dy_pi_Dy_pi) );
        //*****
        // 3-3-component: (Z,Z)
        Tij(xField, 2, 2) = + coeff2 * (-3.* w * Hcon* pi_k(xField) + zeta_integer(xField)
        /*Non-linear*/ - non_linearity * (gradientpi_squared / 2. + Dz_pi_Dz_pi) );
        //*****
        // 1-2-component: (X,Y)
        Tij(xField, 0, 1) = + non_linearity * coeff2 * /*Non-linear*/ Dx_pi_Dy_pi;
        //*****
        // 1-3-component: (X,Z)
        Tij(xField, 0, 2) = + non_linearity * coeff2 * /*Non-linear*/ Dx_pi_Dz_pi;
    }
}

```

```

// *****
// 2-3-component: (Y,Z)
Tij(xField, 1, 2) = + non_linearity * coeff2 * /*Non-linear*/ Dy_pi_Dz_pi;
// *****

// *****
// In the case of Vector parabolic
// *****
if(method==1) // method=1 Turn on vector elliptic
{
    if (non_linearity ==1)
    {
        // T01:(Time,X)
T0i(xField, 0) = -coeff2 * (1. - /*Non-linear*/ non_linearity * (1./cs2 -1.) *
    gradientpi_squared / 2.) * (pi_k(xField + 0) - pi_k(xField - 0)) / (2. * dx);
        // *****
        // T02:(Time,Y)
T0i(xField, 1) = -coeff2 * (1. - /*Non-linear*/ non_linearity * (1./cs2 -1.) *
    gradientpi_squared / 2.) * (pi_k(xField + 1) - pi_k(xField - 1)) / (2. * dx);
        // *****
        // T03:(Time,Z)
T0i(xField, 2) = -coeff2 * (1. - /*Non-linear*/ non_linearity * (1./cs2 -1.) *
    gradientpi_squared / 2.) * (pi_k(xField + 2) - pi_k(xField - 2)) / (2. * dx);
        // *****
    }
}
}

```

For the ζ update, we have turned off the predictor-corrector part, we have

```

template <class FieldType>
void update_zeta(double dtau, double dx, double a, Field<FieldType> & phi, Field<
    FieldType> & phi_old, Field<FieldType> & chi, Field<FieldType> & chi_old, Field
    <FieldType> & pi_k, Field<FieldType> & zeta_integer, Field<FieldType> &
    zeta_half, double Omega_fld, double w, double cs2, double Hcon, double H_prime
    , int non_linearity )
{
    double Gradphi_Gradpi, Gradpsi_Gradpi, Gradpi_Gradpi, GradPsiZeta_Gradpi, Dx_psi, Dy_psi, Dz_psi;
    double C1, C2, C3, psi, psi_old, psi_prime, phi_prime, Laplacian_pi,
    zeta_old_half, zeta_old_integer;
    // Since a_kess is at n so H_prime is at n which is needed to calculate zeta(n+1/2)
    // *****
    // Coefficient two, H(n), H_prime(n) since BG already updated
    // *****
    C2 = cs2 * (3. * Hcon * Hcon - 3. * H_prime );
    // *****
    // Coefficient three, H(n), H_prime(n)
    // *****
    C3 = (2. + 3. * w + cs2 ) * Hcon/2.;

    // *****
    // When non-linearities are turned off we put non-linear terms zero
    // *****
    Gradphi_Gradpi=0.;
    Gradpsi_Gradpi=0.;
    Gradpi_Gradpi=0.;
    GradPsiZeta_Gradpi=0.;

    Site x(phi.lattice());
    for (x.first(); x.test(); x.next())
    {
        // *****
        // Laplace pi, pi(n) since pi is not updated yet
        // *****
        Laplacian_pi= pi_k(x-0) + pi_k(x+0) - 2. * pi_k(x);
        Laplacian_pi+=pi_k(x+1) + pi_k(x-1) - 2. * pi_k(x);
        Laplacian_pi+=pi_k(x+2) + pi_k(x-2) - 2. * pi_k(x);
        Laplacian_pi= Laplacian_pi/(dx*dx);

        // *****
        // psi(n)
        // *****
        psi = phi(x) - chi(x);
        // *****
        // Coefficient one, H( at n)
        // *****
        C1 = 1./(1. - 3. * Hcon * w * dtau/2. - non_linearity * (1. - cs2) * Laplacian_pi * dtau/2.);
        ;
        // *****
        // phi'(n)
        // *****
        phi_prime= (phi(x) - phi_old(x))/dtau; //phi_prime(n) since we want to
        use it to compute zeta (n+1/2);
        // *****
        // psi'(n)
        // *****
        psi_prime= ((phi(x) - chi(x)) - (phi_old(x) - chi_old(x))/dtau;
        if (non_linearity ==1)
        {
            // *****
            // Grad_i_Psi
            // *****
            Dx_psi = ((phi(x + 0) - chi(x + 0)) - (phi(x - 0) - chi(x - 0)));
            Dy_psi = ((phi(x + 1) - chi(x + 1)) - (phi(x - 1) - chi(x - 1)));
            Dz_psi = ((phi(x + 2) - chi(x + 2)) - (phi(x - 2) - chi(x - 2)));
            // *****
            // Grad_phi . Grad_pi
            // *****
        }
    }
}

```

```

Gradphi_Gradpi= 0.25 * (phi(x + 0) - phi(x - 0)) * (pi_k(x + 0) - pi_k(x - 0)) / (dx * dx);
Gradphi_Gradpi+=0.25 * (phi(x + 1) - phi(x - 1)) * (pi_k(x + 1) - pi_k(x - 1)) / (dx * dx);
Gradphi_Gradpi+=0.25 * (phi(x + 2) - phi(x - 2)) * (pi_k(x + 2) - pi_k(x - 2)) / (dx * dx);
//****************************************************************
//Grad_psi . Grad_pi
//****************************************************************
Gradpsi_Gradpi= 0.25 * (Dx_psi) * (pi_k(x+0) - pi_k(x-0)) / (dx * dx);
Gradpsi_Gradpi+=0.25 * (Dy_psi) * (pi_k(x+1) - pi_k(x-1)) / (dx * dx);
Gradpsi_Gradpi+=0.25 * (Dz_psi) * (pi_k(x+2) - pi_k(x-2)) / (dx * dx);
//****************************************************************
//Gradpi.Gradpi
//****************************************************************
Gradpi_Gradpi= 0.25 * (pi_k(x + 0) - pi_k(x - 0)) * (pi_k(x + 0) - pi_k(x - 0)) / (dx * dx);
Gradpi_Gradpi+=0.25 * (pi_k(x + 1) - pi_k(x - 1)) * (pi_k(x + 1) - pi_k(x - 1)) / (dx * dx);
Gradpi_Gradpi+=0.25 * (pi_k(x + 2) - pi_k(x - 2)) * (pi_k(x + 2) - pi_k(x - 2)) / (dx * dx);
//****************************************************************
//GradPsiZeta_Gradpi = Grad_pi . Grad_ (zeta + psi)
//Grad_pi . Grad_ (zeta + psi) = Grad_pi (Grad_zeta + Grad_Psi)
//zeta_integer from previous step is at n
//****************************************************************
GradPsiZeta_Gradpi= 0.25* (zeta_integer(x+0) - zeta_integer(x-0) + Dx_psi) * (pi_k(x+0) - pi_k(x-0)) / (dx * dx);
GradPsiZeta_Gradpi+=0.25* (zeta_integer(x+1) - zeta_integer(x-1) + Dy_psi) * (pi_k(x+1) - pi_k(x-1)) / (dx * dx);
GradPsiZeta_Gradpi+=0.25* (zeta_integer(x+2) - zeta_integer(x-2) + Dz_psi) * (pi_k(x+2) - pi_k(x-2)) / (dx * dx);
}
//****************************************************************
// Having the values at previous steps
//****************************************************************
zeta.old_half = zeta_half(x); //zeta(n-1/2)
//****************************************************************
// FULL Updating equation
//****************************************************************
// zeta(n+1/2) = zeta(n-1/2) + zeta'(n)
//****************************************************************
zeta_half(x) = C1 * ( zeta_half(x) + dtau * (
/*Linear(1,2,3)*/
+ 3. * Hcon * ( w * zeta_half(x)/2. + cs2 * psi ) - C2 * pi_k(x)
/*Linear(4,5)*/
+ 3. * cs2 * phi_prime + cs2 * Laplacian_pi
/*Non-linear terms*/
+ non_linearity *
/*Non-linear(1,2)*/
+ 2. * cs2 * phi(x) * Laplacian_pi - (1. - cs2) * psi * Laplacian_pi
/*Non-linear(3)*/
- 3. * cs2 * Hcon * (1. + w) * pi_k(x) * Laplacian_pi
/*Non-linear(5,6,7)*/
- cs2 * Gradphi_Gradpi + (2. * cs2 - 1.) * Gradpsi_Gradpi - C3 *
Gradpi_Gradpi
/*Non-linear(8)*/
+ 2. * (1. - cs2) * GradPsiZeta_Gradpi
)
);
//****************************************************************
//Computing zeta(n+1)
//****************************************************************
double zeta_prime_int_n0 ;
//****************************************************************
// zeta'(n) from the new values at n, zeta(n)...
// like zeta_integer(x) (n)
//****************************************************************
zeta_prime_int_n0 =
/*Linear(1,2,3)*/
+ 3. * Hcon * ( w * zeta_integer(x) + cs2 * psi ) - C2 * pi_k(x)
/*Linear(4,5)*/
+ 3. * cs2 * phi_prime + cs2 * Laplacian_pi
/*Non-linear(1,2)*/
+ 2. * cs2 * phi(x) * Laplacian_pi - (1. - cs2) * psi * Laplacian_pi
/*Non-linear(3)*/
- 3. * cs2 * Hcon * (1. + w) * pi_k(x) * Laplacian_pi
/*Non-linear(4)*/
+(1. - cs2) * (zeta_integer(x) + psi) * Laplacian_pi
/*Non-linear(5,6,7)*/
- cs2 * Gradphi_Gradpi + (2. * cs2 - 1.) * Gradpsi_Gradpi - C3 *
Gradpi_Gradpi
/*Non-linear(8)*/
+ 2.* (1. - cs2) * GradPsiZeta_Gradpi;
//****************************************************************
// When we get the precision and it goes out of loop we need to update zeta_integer for
// being synched with particles and ....
//zeta(n+1) = zeta(n+1/2) + zeta'(n)dtau/2
//****************************************************************
zeta_integer(x) = zeta_half(x) + zeta_prime_int_n0 * dtau/2.;

}
}
}

```

In the main file we have stress tensor of kessence and main update of the fields

```

        }
        if (sim.Kess_source_gravity==1)
{
// Kessence projection Tmunu
// In the projection zeta_integer comes, since synched with particles..
    if (sim.vector_flag == VECTORELLIPTIC)
    {
        projection_Tmunu_kessence( T00_Kess,T0i_Kess,Tij_Kess , dx, a, phi, phi.old,
                                chi, pi_k, zeta_integer, cosmo.Omega_kessence, cosmo.w_kessence, cosmo.
                                cs2_kessence, Hconf(a, fourpiG, cosmo), fourpiG, sim.NL_kessence ,1 );
    }
    else
    {
        projection_Tmunu_kessence( T00_Kess,T0i_Kess,Tij_Kess , dx, a, phi, phi.old,
                                chi, pi_k, zeta_integer, cosmo.Omega_kessence, cosmo.w_kessence, cosmo.
                                cs2_kessence, Hconf(a, fourpiG, cosmo), fourpiG, sim.NL_kessence , 0 );
    }
    for (x.first(); x.test(); x.next())

```

```

{
    // The coefficient is because it wanted to be source according to eq C.2 of
    // Gevolution paper
    // Note that it is multiplied to dx^2 and is divided by -a^3 because of definition
    // of T00 which is scaled by a^3
    // We have T00 and Tij according to code's units, but source is important to
    // calculate potentials and moving particles.
    // There is coefficient between Tij and Sij as source.
    source(x) += (fourpiG * dx * dx / a) * T00_Kess(x);
    if (sim.vector_flag == VECTOR_ELLIPTIC) for(int c=0;c<3;c++) Bi(x,c) += (2. *
        fourpiG * dx * dx / a) * T0i_Kess(x,c);
    for(int c=0;c<6;c++) Sij(x,c) += (2. * fourpiG * dx * dx / a) * Tij_Kess(x,c);
}
#endif BENCHMARK
    kessence_update_time += MPI_Wtime() - ref_time;
    ref_time = MPI_Wtime();
#endif
// Kessence projection Tmunu end

//Kessence
#ifndef BENCHMARK
    ref_time = MPI_Wtime();
#endif
    for (x.first(); x.test(); x.next())
    {
        phi_prime(x) = (phi(x)-phi_old(x))/(dtau);
    }
    // We just need to update halo when we want to calculate spatial derivative or use some neighbours
    // at the same time! So here we do not need to update halo for phi_prime!
//*****Kessence - LeapFrog:START
//*****
double a_kess=a;
for (i=0;i<sim.nKe.numsteps; i++)
{
    //First we update zeta_integer to have it at 0-1/2 just in the first loop
    if(cycle==0)
    {
        //computing zeta_half(-1/2) and zeta_int(-1) but we do not work with zeta(-1)
        update_zeta(-dtau/(2. * sim.nKe.numsteps), dx, a_kess, phi, phi_old, chi, chi_old, pi_k,
            zeta_integer, zeta_integer, cosmo.Omega_kessence, cosmo.w_kessence, cosmo.cs2_kessence, Hconf(
            a_kess, fourpiG, cosmo), Hconf_prime(a_kess, fourpiG, cosmo), sim.NL_kessence);
    }
    //*****Upgrading zeta_integer to get zeta_integer(n+1/2) and zeta_integer(n+1), in the first loop is getting
    //zeta_integer(1/2) and zeta_integer(1)
    // In sum: zeta_integer(n+1/2) = zeta_integer(n-1/2)+ zeta_integer'(n)dtau which needs background to
    // be at n with then
    //Note that here for zeta_integer'(n) we need background to be at n and no need to update it.
    //\zeta_integer(n+1/2) = \zeta_integer(n-1/2) + \zeta_integer'(n) dtau
    //We also update zeta_int from n to n+1
    //*****
    update_zeta(dtau/sim.nKe.numsteps, dx, a_kess, phi, phi_old, chi, chi_old, pi_k, zeta_integer,
        zeta_integer, cosmo.Omega_kessence, cosmo.w_kessence, cosmo.cs2_kessence, Hconf(a_kess, fourpiG,
        cosmo), Hconf_prime(a_kess, fourpiG, cosmo), sim.NL_kessence);
    //*****Since we have pi(n+1)=pi(n) + pi'(n+1/2), and in pi'(n+1/2) we have H(n+1/2) we update the
    //background before updating the pi to have H(n+1/2). Moreover zeta(n+1) = zeta(n+1/2) + zeta'(n
    //+1/2), so we put zeta.int updating in the pi updating!
    //*****
    rungekutta4bg(a_kess, fourpiG, cosmo, dtau / sim.nKe.numsteps / 2.0);
    //*****we update pi to have it at n+1 (at first loop from the value at (0) and the value of zeta_integer at
    //1/2 and H(n+1/2) we update pi at (1))
    //In the pi update we also update zeta_int because we need the values of a_kess and H_kess at step n
    //+1/2
    //By the below update we get pi(n+1) and zeta(n+1)
    //*****
    update_pi_k(dtau/sim.nKe.numsteps, dx, a_kess, phi, phi_old, chi, chi_old, pi_k, zeta_integer,
        zeta_integer, cosmo.Omega_kessence, cosmo.w_kessence, cosmo.cs2_kessence, Hconf(a_kess, fourpiG,
        cosmo), Hconf_prime(a_kess, fourpiG, cosmo), sim.NL_kessence); // H.old is updated here in the
        function
        pi_k.updateHalo();
    zeta_integer.updateHalo();
    //*****Now we have pi(n+1) and a_kess(n+1/2) so we update background by halfstep to have a_kess(n+1)
    //*****
    rungekutta4bg(a_kess, fourpiG, cosmo, dtau / sim.nKe.numsteps / 2.0 );
}

#endif BENCHMARK
    kessence_update_time += MPI_Wtime() - ref_time;
    ref_time = MPI_Wtime();
#endif
//*****Kessence - LeapFrog: End
//*****

```