1 δ comparison

1.1 Without sourcing total stress tensor

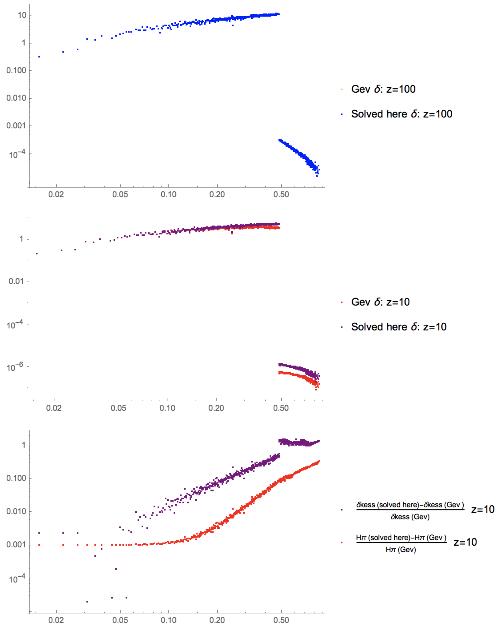
Here to check we are writing the true stress tensor, we compare the Gevolution with mathematica result and we turn on the kessence source gravity flag but keep in mind that since we use new set of variables, the gravitational potentials are not appeared on the stress tensor since we have gravitational potential hidden on the ζ!
So we can easily check Gevolution versus mathematica to just check that the power of Stress tensor is correctly produced, while we already know that it must be correct since we have checked all the variables including like π, ζ and $\mathcal{H}$ so it is kind of making sure that the effect of kessence stress tensor on the total stress tensor is taken into account correctly and plus it is written in the output text rightly.
So we are going to do is first turn off the effect of kessence stress tensor on the total stress tensor and check just the kessence stress tensor is written correctly and is compared with the error we expect. In terms of new variables the stress tensor reads,

$$\begin{aligned} T^0_0(Gev) &= \Omega^0_{kess} a^{-3w} \left[1 + \frac{1+w}{c_s^2} \Big(-3\mathcal{H} c_s^2 \pi + \zeta \Big) \right] \\ T^i_0(Gev) &= -\Omega^0_{kess} a^{-3w} (1+w) \partial_i \pi \\ T^i_j(Gev) &= w\, \Omega^0_{kess} a^{-3w} \left[1 + \frac{1+w}{w} \Big(-3\mathcal{H} w \pi + \zeta \Big) \delta^i_j \right] \end{aligned} \tag{1}$$

So the $\delta_{kess} = \frac{1+w}{c_s^2} \Big(-3\mathcal{H} c_s^2 \pi + \zeta \Big)$ which now we want to compare between Gevolution and mathematica solution for when this stress tensor does not contribute to the total stress tensor in Gevolution which is what we are able to catch in mathematica!
The technical part of mathematica script would be,

```
GevPowerDataz100 =
Table[{datapiGev100[[l, 1]], GevFieldz100[[l]], Gevzetaz100[[l]],  GevDeltaz100[[l]]}, {l, 1, Dimensions
    [GevFieldz100][[1]]}];
MatSolz100 =
Table[{GevPowerDataz100[[k, 1]], Abs[Agrex[[k]][[1, 2]][[1]]],  Abs[Agrex[[k]][[1, 3]][[1]]],  Abs[(1 +
    w)/ cs^2 (-3 cs^2 Agrex[[k]][[1, 2]][[1]]
+  Agrex[[k]][[1, 3]][[1]]  )]}, {k, 1, kmax}]; (*Last column is delta computed*)
Error\[Delta]10 = Table[{MatSolz10[[k, 1]],  Abs[(Abs[MatSolz10[[k, 4]]] - GevPowerDataz10[[k, 4]])]/
   GevPowerDataz10[[k, 4]]}, {k, 1, kmax}] // N;
```

After tracking the δ in mathematica and Gevolution we get the result in the below,

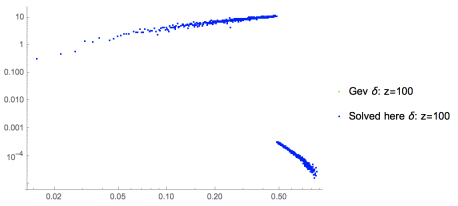


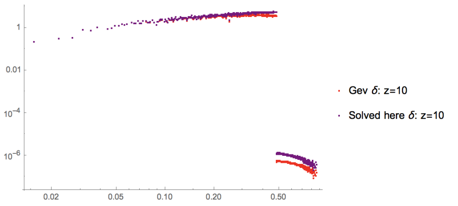
1.2 Kessence sourcing total stress tensor

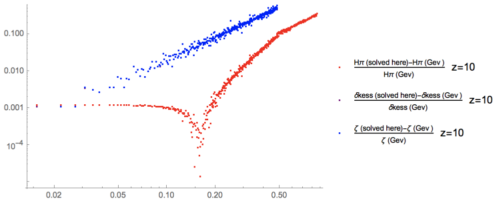
Now what happens if we let also the stress tensor of kessence contribute in the total stress energy tensor? That is a important check which shows that the stress tensor is added in Gevolution correctly, since from previous experience we know that if we add it with wrong coefficients the particles speed up and go over cpus! Again we have the same stress tensor for the kessence but this time we add the components to the total stress tensor as following,

```
if (sim.Kess_source_gravity==1)
{
// Kessence projection Tmunu
        if (sim.vector_flag == VECTOR_ELLIPTIC)
                {
                        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, 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, 0 );
                }

                for (x.first(); x.test(); x.next())
                {
                        // The coefficient is because it wanted to to be source according to eq C.2 of
                            Gevolution paper
                        // Note that it is multiplied to dx^2 and is divived 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);
                }
}
#ifdef BENCHMARK
                kessence_update_time += MPI_Wtime() - ref_time;
                ref_time = MPI_Wtime();
#endif
// Kessence projection Tmunu end
```

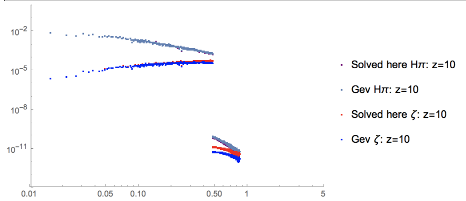
It is clear that we should not get the different solution since we have turned off the effect of potentials on the scalar field, also the below plots show nothing changes in this case, but still was a good check that at least we did not make a bad mistake at coefficients which otherwise it had blown up.

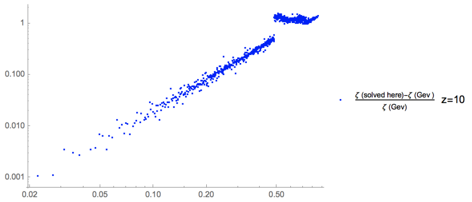


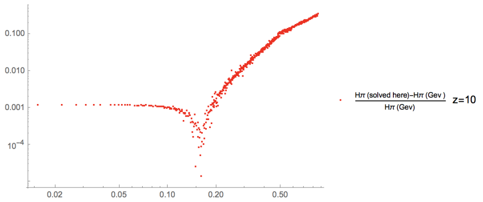




```
Eq1II = 3 cs^2 (H[τ]^2 - H'[τ]) piC[τ] - 3 H[τ] (w ζ[τ]) + ζ'[τ] + cs^2 kwave^2 piC[τ]; Eq2II = piC'[τ] + H[τ] piC[τ] - ζ[τ];
tiling factor = 16 , nKe_numsteps=10, Kessence source gravity= 0 , initial redshift  = 100.0, boxsize  = 400.0  , Ngrid = 64 , Courant factor  = 48.0
time step limit   = 0.04
```







2 Solving the full equation with potentials

2.1 Field dynamics, Kessence stress tensor is turned off

Here we try to compare mathematica with Gevolution in scalar fields solution when the gravitational potentials are turned off and kessence does not source gravity. The set of equations are as following,

$$\zeta' = 3\mathcal{H}(w\zeta + c_s^2\Psi) - c_s^2(3\mathcal{H}^2 - 3\mathcal{H}')\pi + 3c_s^2\Phi' + c_s^2\nabla^2\pi \tag{2}$$

$$\pi' = \zeta - \mathcal{H}\pi + \Psi \tag{3}$$

Now in mathematica we provide Φ and Ψ' from the Gevolution, but we take them as a constant in time, then we solve the full equations and compare with Gevolution. Using the leap frog method as following (like before) we have,

2.1.1 π equation

for the π equation we have,

$$\pi_{n+1} = \pi_n + \pi'_{n+\frac{1}{2}}\Delta\tau \tag{4}$$

$$\pi'_{n+\frac{1}{2}} = \zeta_{n+\frac{1}{2}} - \mathcal{H}_{n+\frac{1}{2}}\pi_{n+\frac{1}{2}} + \Psi_{n+\frac{1}{2}} \tag{5}$$

For the background part it is better to update background 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 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

$$\pi_{n+\frac{1}{2}} = \frac{\pi_{n+1}+\pi_n}{2} \tag{6}$$

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

For π we have,

$$\pi_{n+1} = \pi_n + \Delta\tau\Big[\zeta_{n+\frac{1}{2}} - \mathcal{H}_{n+\frac{1}{2}}(\frac{\pi_{n+1}+\pi_n}{2}) + \Psi_{n+\frac{1}{2}}\Big] \tag{7}$$

$$\pi_{n+1} = \frac{1}{1+\mathcal{H}_{n+\frac{1}{2}}\Delta\tau/2}\left[\pi_n + \Delta\tau\Big[\zeta_{n+\frac{1}{2}} - \mathcal{H}_{n+\frac{1}{2}}\frac{\pi_n}{2} + \Psi_{n+\frac{1}{2}}\Big]\right] \tag{8}$$

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,

$$\Psi_{n+\frac{1}{2}} = \Psi_n + \Psi'_n\frac{d\tau}{2} \tag{9}$$

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

$$\Psi'_n = \frac{\Psi_n - \Psi_{n-1}}{d\tau} \tag{10}$$

2.1.2 ζ equation

$$\zeta_{n+\frac{1}{2}} = \zeta_{n-\frac{1}{2}} + \zeta'_n\Delta\tau \tag{11}$$

where ζ'_n reads from the differential equation as following,

$$\zeta'_n = 3\mathcal{H}_n(w\zeta_n + c_s^2\Psi_n) - c_s^2(3\mathcal{H}_n^2 - 3\mathcal{H}'_n)\pi_n + 3c_s^2\Phi'_n + c_s^2\nabla^2\pi_n \tag{12}$$

Since we need to have ζ_n at ζ'_n so we write $\zeta_n = \frac{\zeta_{n+1/2}+\zeta_{n-1/2}}{2}$

We get,

$$\zeta_{n+\frac{1}{2}} = \zeta_{n-\frac{1}{2}} + \Delta\tau\left[3\mathcal{H}_n(w\frac{\zeta_{n+\frac{1}{2}}+\zeta_{n-\frac{1}{2}}}{2} + c_s^2\Psi_n) - c_s^2(3\mathcal{H}_n^2 - 3\mathcal{H}'_n)\pi_n + 3c_s^2\Phi'_n + c_s^2\nabla^2\pi_n\right] \tag{13}$$

Simplifying the expression gives,

$$\zeta_{n+\frac{1}{2}} = \frac{1}{1-3\mathcal{H}_n w\Delta\tau/2}\left[\zeta_{n-\frac{1}{2}}+\Delta\tau\left(3\mathcal{H}_n\left(\frac{w\zeta_{n-\frac{1}{2}}}{2}+c_s^2\Psi_n\right)-c_s^2(3\mathcal{H}_n^2-3\mathcal{H}_n')\pi_n+3c_s^2\Phi_n'+c_s^2\nabla^2\pi_n\right)\right] \tag{14}$$

It is very important to check everything for the first loop specifically! For the first loop we have $\zeta^{(0)}$, $\zeta^{(-1/2)}$ is computed from the values at step 0 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 update π to have it at step 1 and then we update ζ to have it at step 3/2 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,

$$\zeta^{n+1/2},\zeta^{n},\pi^{n},\Psi^{n},\Psi'^{n}\longrightarrow a_{kess}^{n+1/2},\Psi^{n+1/2},\zeta^{n+1/2},\pi^{n+1}\longrightarrow a_{kess}^{n+1},\pi^{n+1},\Psi^{n+1},\zeta^{n+3/2}. \tag{15}$$

Approximations:

The important approximations here are:

1- At the first loop we take $\Phi'=0$

2- The value of $\Phi^{(n+1)}=\Phi^{(n)}+\Phi'^{(n)}\times d\tau$ to compute $\zeta^{n+3/2}$ which again at the first loop we assume $\Phi^{(1)}=\Phi^{(0)}$

3- Since we do not have $\Phi^{'(n+1)}$ we assume that it does not change fast, so we take $\Phi^{'(n+1)}=\Phi'^{(n)}$ or equivalently $\Phi''^{(n)}\approx 0$

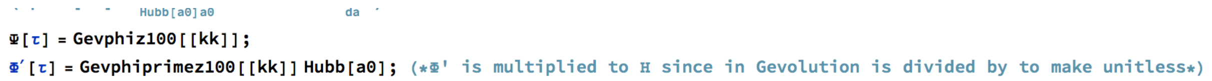
All the top approximations can be suppressed by increasing the number of kessence update or decreasing the time stepping of Gevolution.

Some important point in the Gevolution might be in the Gevolution.hpp

```
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> &,  Field<FieldType> & zeta_half , double
                     Omega_fld ,double w, double cs2, double Hcon, double  H_prime)
                {
double psi, psi_prime, psi_half;
// WRONG! double H_half= Hcon + H_prime *  dtau/2. ; // H(n+1/2) = H(n) + H'(n) dtau/2 WRONG!
//We do not need to write the top equation, since we already have H(n+1/2) by updating the
    background by half step.
//Hcon is at (n+1/2)
double Coeff1 = 1./(1. + Hcon * dtau/2.);

                  Site x(phi.lattice());
                  for (x.first(); x.test(); x.next())
                    {
    psi=phi(x) - chi(x);
    psi_prime= ((phi(x) - chi(x))-(phi_old(x) - chi_old(x)))/dtau;
    psi_half= psi + psi_prime * dtau/2.; //psi_half (n+1/2) = psi(n) + psi_prime'(n) dtau/2

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

    //Full equation
    //pi_k(n+1) = Coeff1 * (pi_k(n) + \Delta T (zeta(n+1/2) + ... ))
                    // pi_k(x)=Coeff1 * (pi_k(x)  + dtau * ( zeta_half(x) - Hcon * pi_k(x)/2. +
                        psi_half ) ); //  pi_k(n+1)


    //NOTE: zeta and psi must be at n+1/2 step according to the formula! So we need to update zeta
         first in the main loop.
                    }
                }
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_half , Field<FieldType> &
                        zeta_integer, double Omega_fld ,double w, double cs2, double Hcon, double
                        H_prime )
                    {
                      double CoeffI, CoeffII, psi, psi_old, psi_prime, phi_prime, Laplacian_pi,
                          zeta_old_half ;
//Hcon(n+1)to calculate zeta(n+1/2)
CoeffI = 1./(1. -  3. * Hcon * w  * dtau/2. );
//Since a_kess is at (n+1) so H_prime is at (n+1) which is needed to calculate zeta(n+1/2)
CoeffII = cs2 * (3. * Hcon * Hcon - 3. * H_prime );

                            Site x(phi.lattice());
                            for (x.first(); x.test(); x.next())
                                {
    //Everything here is at step n except zeta which is at half steps! zeta is like pi_v
                                        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=phi(x) - chi(x); //psi(n)
                                        // psi_prime= ((phi(x) - chi(x))-(phi_old(x) - chi_old(x))
                                            )/dtau; //psi_prime(n)
                                    phi_prime= (phi(x) - phi_old(x))/dtau; //phi_prime(n+1) since we
                                        want to use it to compute zeta (n+3/2)
    //NOTE: We dont have phi'(n+1) since it will be updated by particles later, but we ***ASSUME**
         it remains contant and phi_prime(n) = phi_prime(n+1) or take the second derivative for
        this period approximately to zero!
    psi_prime= ((phi(x) - chi(x))-(phi_old(x) - chi_old(x)))/dtau;
    psi = psi + psi_prime * dtau; //psi(n+1) = psi(n) + psi'(n) dtau
    //***************
    //Linear equation
    //***************
    //Scalar Field only equation:
    zeta_old_half=zeta_half(x); // zeta(n+1/2)
    zeta_half(x)= CoeffI * ( zeta_half(x) + dtau * ( 3. * Hcon * ( w * zeta_half(x)/2. + cs2 * psi
         ) - CoeffII * pi_k(x) + 3. * cs2 * phi_prime + cs2 * Laplacian_pi) ); // zeta(n+3/2) from
         zeta(n+1/2) and zeta'(n+1) so we need to have Phi^{n+1} and Phi'{n+1} which we
        approximate both!
    //***************
    //Linear equation
    //***************
    // computing zeta (n) by taking average ove zeta(n+1/2) and zeta(n-1/2)
    zeta_integer(x)= (zeta_half(x) + zeta_old_half)/2.; //zeta(n+1)
    }
```

Also the important point in mathematica is adding Φ and Φ' as the initial condition from Gevolution as following,

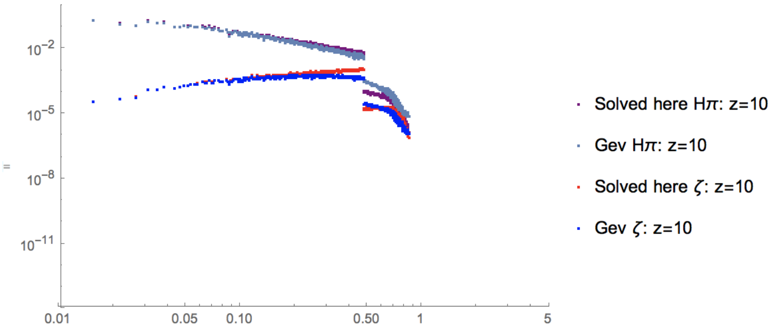


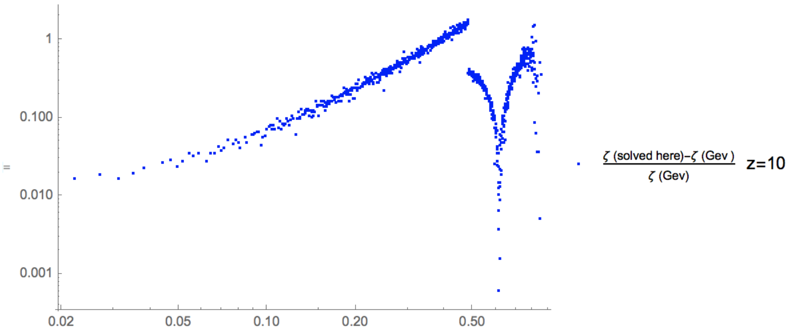
```
Ψ[τ] = Gevphiz100[[kk]];
Φ′[τ] = Gevphiprimez100[[kk]] Hubb[a0]; (*Φ' is multiplied to H since in Gevolution is divided by to make unitless*)
```

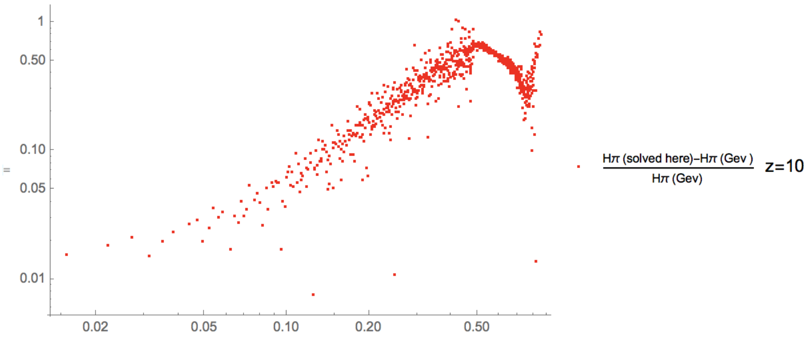
Again providing the initial condition from Gevolution at z=100 and looking at evolution up to z=10 while Φ and Φ' is provided by Gevolution at redshift 100 and is taken constant, we get,

```
Eq1II = 3 cs^2 (H[τ]^2 - H'[τ]) piC[τ] - 3 H[τ] (w ζ[τ] + cs^2 Ψ[τ]) + ζ'[τ] - 3 cs^2 Φ'[τ] + cs^2 kwave^2 piC[τ]; Eq2II = piC'[τ] + H[τ] piC[τ] - ζ[τ] - Ψ[τ];
```

tiling factor = 16 , nKe_numsteps=10, Kessence source gravity= 0 , initial redshift = 100.0, boxsize = 400.0 , Ngrid = 64 , Courant factor = 48.0
time step limit = 0.04



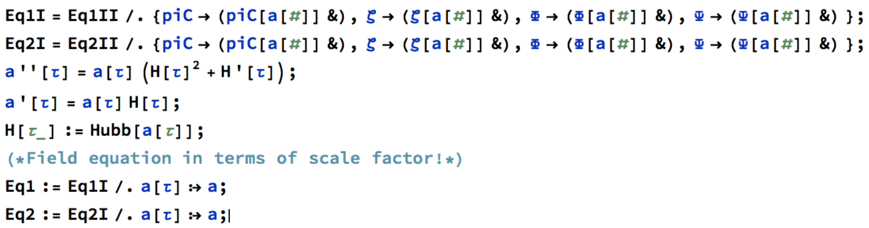




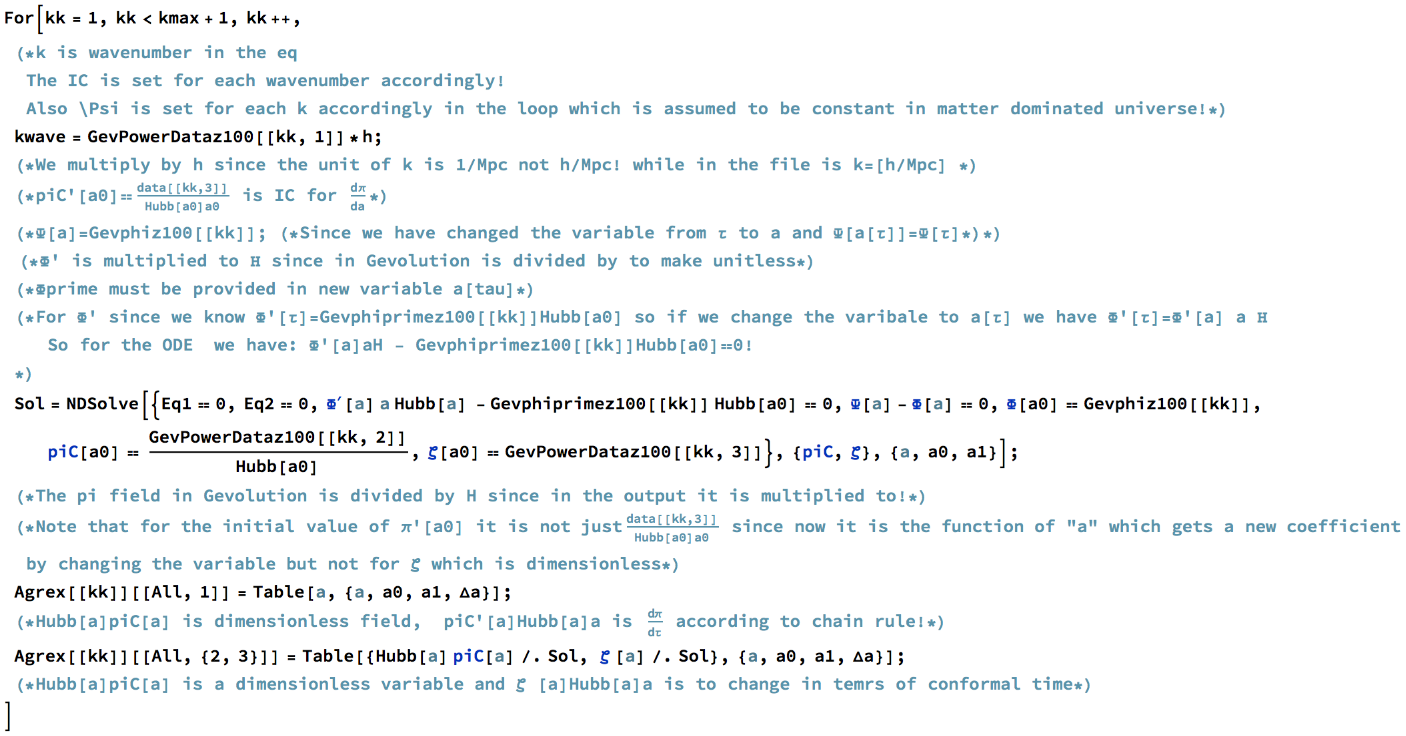
The result is acceptable, but we know that if we increase kessence number of update we improve the approximations and also we see that the difference is more in high wavenumbers which is probably the effect of being near Nyqvist frequency, so we guess we can improve it by increasing number of grids. Moreover the relative error could be because of the fact that we do not solve extra differential equation for Φ and naively we take it independent of Φ' and both as a constant.

Now we want to test if we add an extra differential equation in mathematica as $\Phi' = const$, where the constant is the value of Φ' in each redshift and the initial condition of Φ reads from Gevolution, whether we improve the relative error?

The technical points in mahtematica are as following which is change the time variable in Φ and Ψ

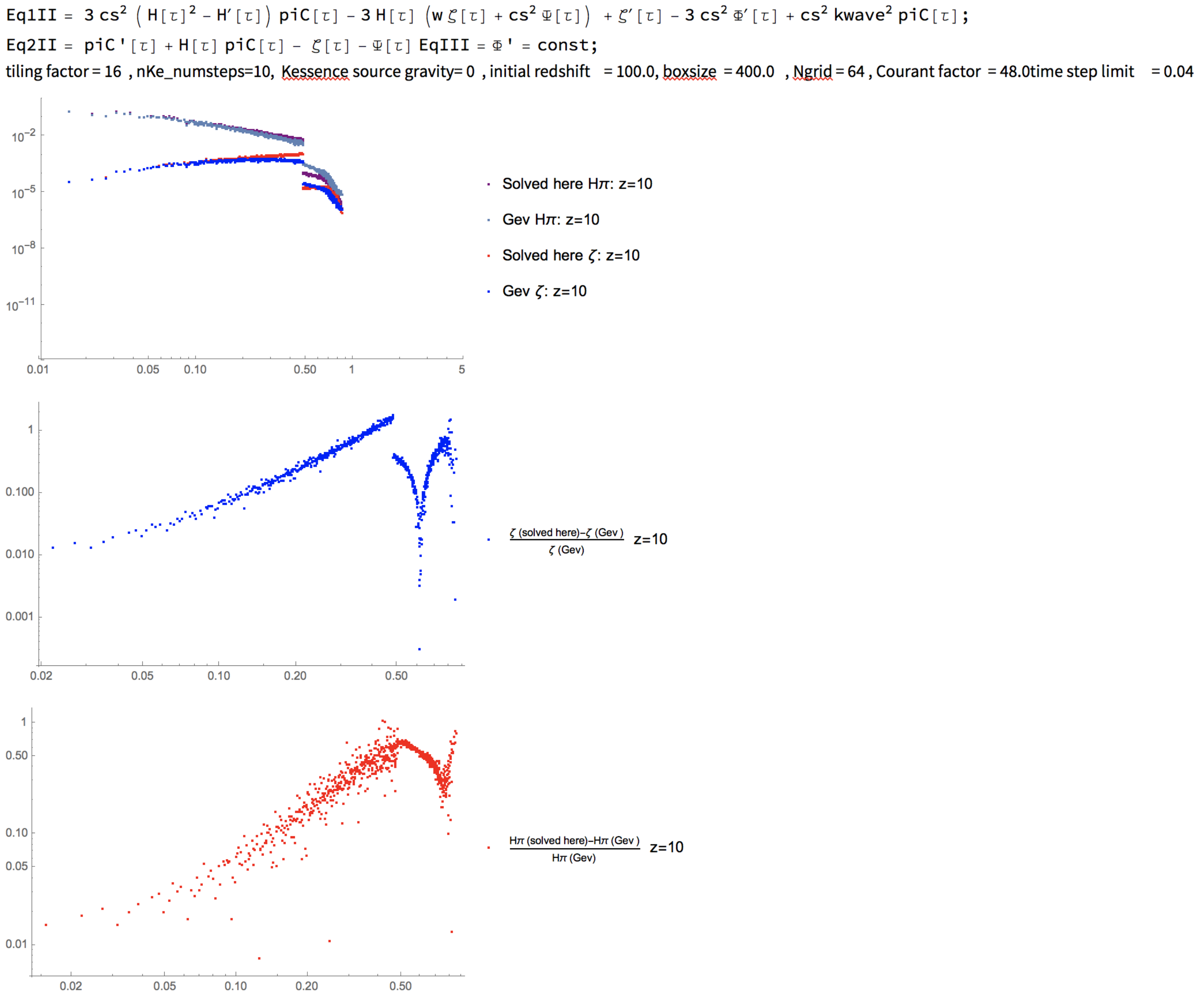


```
Eq1I = Eq1II /. {piC → (piC[a[#]] &), ξ → (ξ[a[#]] &), Φ → (Φ[a[#]] &), Ψ → (Ψ[a[#]] &) };
Eq2I = Eq2II /. {piC → (piC[a[#]] &), ξ → (ξ[a[#]] &), Φ → (Φ[a[#]] &), Ψ → (Ψ[a[#]] &) };
a''[τ] = a[τ] (H[τ]^2 + H'[τ]);
a'[τ] = a[τ] H[τ];
H[τ_] := Hubb[a[τ]];
(*Field equation in terms of scale factor!*)
Eq1 := Eq1I /. a[τ] :→ a;
Eq2 := Eq2I /. a[τ] :→ a;
```



```
For[kk = 1, kk < kmax + 1, kk ++,
 (*k is wavenumber in the eq
  The IC is set for each wavenumber accordingly!
  Also \Psi is set for each k accordingly in the loop which is assumed to be constant in matter dominated universe!*)
 kwave = GevPowerDataz100[[kk, 1]] * h;
 (*We multiply by h since the unit of k is 1/Mpc not h/Mpc! while in the file is k=[h/Mpc] *)
 (*piC'[a0]==data[[kk,3]]/(Hubb[a0] a0) is IC for dπ/da*)
 (*Ψ[a]=Gevphiz100[[kk]]; (*Since we have changed the variable from τ to a and Ψ[a[τ]]=Ψ[τ]*)*)
  (*Φ' is multiplied to ℋ since in Gevolution is divided by to make unitless*)
 (*Φprime must be provided in new variable a[tau]*)
 (*For Φ' since we know Φ'[τ]=Gevphiprimez100[[kk]]Hubb[a0] so if we change the varibale to a[τ] we have Φ'[τ]=Φ'[a] a ℋ
    So for the ODE  we have: Φ'[a]aH - Gevphiprimez100[[kk]]Hubb[a0]==0!
 *)
 Sol = NDSolve[{Eq1 == 0, Eq2 == 0, Φ'[a] a Hubb[a] - Gevphiprimez100[[kk]] Hubb[a0] == 0, Ψ[a] - Φ[a] == 0, Φ[a0] == Gevphiz100[[kk]],
     piC[a0] == GevPowerDataz100[[kk, 2]]/Hubb[a0], ξ[a0] == GevPowerDataz100[[kk, 3]]}, {piC, ξ}, {a, a0, a1}];
 (*The pi field in Gevolution is divided by H since in the output it is multiplied to!*)
 (*Note that for the initial value of π'[a0] it is not just data[[kk,3]]/(Hubb[a0] a0) since now it is the function of "a" which gets a new coefficient
  by changing the variable but not for ξ which is dimensionless*)
 Agrex[[kk]][[All, 1]] = Table[a, {a, a0, a1, Δa}];
 (*Hubb[a]piC[a] is dimensionless field,  piC'[a]Hubb[a]a is dπ/dτ according to chain rule!*)
 Agrex[[kk]][[All, {2, 3}]] = Table[{Hubb[a] piC[a] /. Sol, ξ [a] /. Sol}, {a, a0, a1, Δa}];
 (*Hubb[a]piC[a] is a dimensionless variable and ξ [a]Hubb[a]a is to change in temrs of conformal time*)
]
```

And the result is,

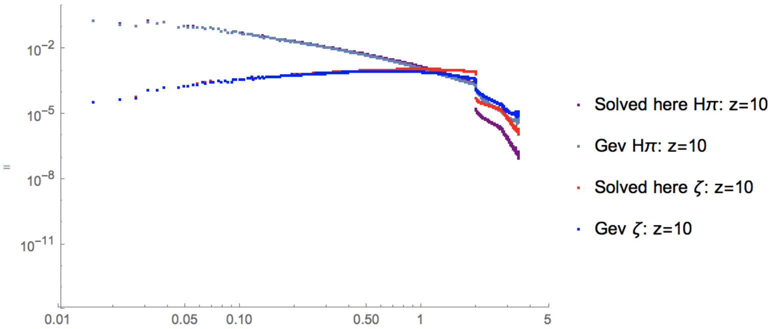


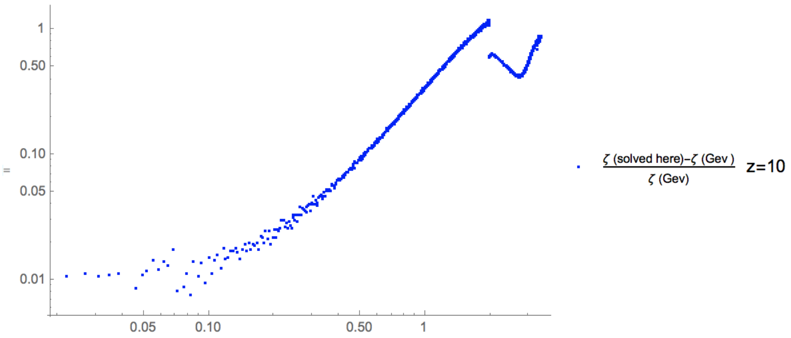
```
Eq1II = 3 cs^2 (H[τ]^2 - H′[τ]) piC[τ] - 3 H[τ] (w ζ[τ] + cs^2 Ψ[τ]) + ζ′[τ] - 3 cs^2 Φ′[τ] + cs^2 kwave^2 piC[τ];
Eq2II = piC'[τ] + H[τ] piC[τ] - ζ[τ] - Ψ[τ] EqIII = Φ' = const;
tiling factor = 16 , nKe_numsteps=10, Kessence source gravity= 0 , initial redshift  = 100.0, boxsize  = 400.0  , Ngrid = 64 , Courant factor  = 48.0time step limit   = 0.04
```

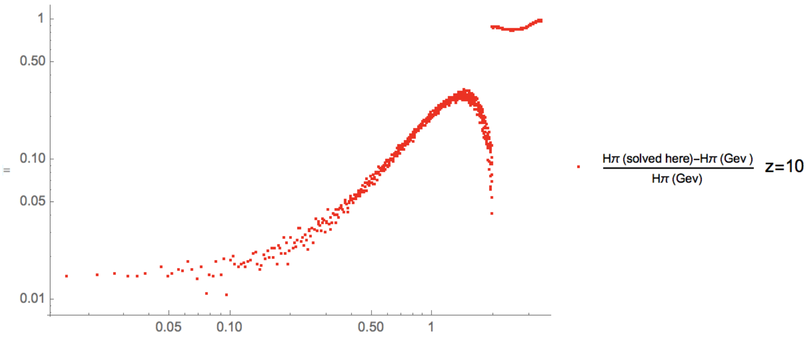
Compared with previous plot we see no difference and the conclusion is that the change of Φ or Ψ during different redshifts based on Φ' does not have so much affect on the ζ and $\mathcal{H}\pi$, on ζ it is clear since the effect of potentials is suppressed with the c_s^2 factor, but on π it has a direct effect, but it seems it is suppressed compared with other terms in equation!
Lets check the improvements by increasing precision parameters, like number of grid and step size. So for a biggest possible number of grids in local computer, 256 and kessence number of update=15 we get the below figure

```
Eq1II = 3 cs^2 (H[τ]^2 - H'[τ]) piC[τ] - 3 H[τ] (w ζ[τ] + cs^2 Ψ[τ]) + ζ'[τ] - 3 cs^2 Φ'[τ] + cs^2 kwave^2 piC[τ];
Eq2II = piC'[τ] + H[τ] piC[τ] - ζ[τ] - Ψ[τ] EqIII = Φ' = const;
```

tiling factor = 64 , nKe_numsteps=15, Kessence source gravity= 0 , initial redshift = 100.0, boxsize = 400.0 , Ngrid = 256, Courant factor = 48.0time step limit = 0.04







The result is acceptable specially referring to the fact that we are using wrong Φ and Φ' in the mathematica!

Just to check that everything is consistent we run Gevolution but instead of comparing the results at z=10 we compare at higher redshifts which we think our approximation ($\Phi'' \approx 0$) works better. Now we compare the results at z=80 from initial condition at z=100,

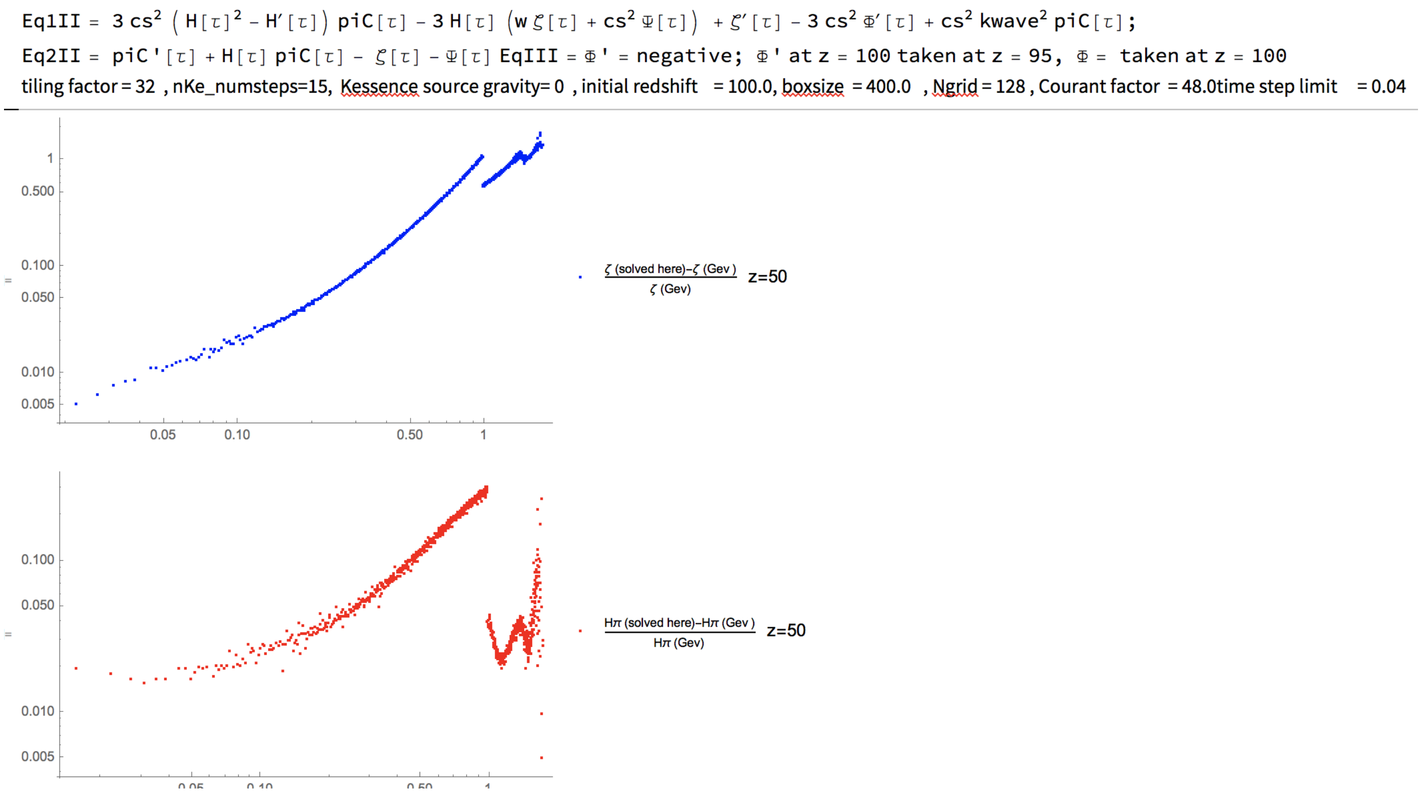
Bug :Up to now we have made a mistake on providing Φ', since at initial redshift it is zero! So we made a mistake on all comparisons with Φ' since it was taken exactly zero! To fix this, we provide Φ' from redshift like z=90which is non-zero!

Important point: Since we are providing the initial condition for mathematica from the power of variables so it does not care about the sign!!! If we take the output of Φ and Φ' from Gevolution we see that both change sign during redshift and depending on the position on the lattice!!! But is it the

same in Fourier space? Not necessarily! in Fourier space Φ is positive and Φ' is negative according to our observation from Class output! Also we observe that if we set $\Phi' = 0$ in mathematica we make less mistake than when we use the positive value at initial redshift! It seems somehow complicated to capture the sign which is removed by computing power!
The way to read the sign of $\Phi(x)$ and $\Phi'(x)$ is as following in Gevolution,

```
for (x.first(); x.test(); x.next())
          {
                      phi_prime(x) =(phi(x)-phi_old(x))/(dtau);
  if(x.coord(0)==2  && x.coord(2)==1)
  {
    if(parallel.isRoot())
    {
      cout<<"Phi_prime: "<<phi_prime(x)<<"Phi: "<<phi(x)<<endl;
    }
  }
          }
```

We must provide the field transfer function as an output which seems very important for tracing small errors! We need it because we are solving different equation in mathematica and Gevolution since providing initial condition from power removes the sign of the field!
In parallel we can believe our comparison, because of the fact that the relative error is small and also we just have added the potentials which we believe are true in Gevolution! So the final comparison which is really important is Class versus Gevolution and we use mathematica just for checking the numerical approximations and if in principle it is true.
But as already mentioned we can find the sign of variables, if they don't change sign and in this case we do not have problem dealing with setting initial condition in mathematica!
According to the fact Φ is positive in Fourier space and Φ' is negative (the potential decays) in high redshifts we set the correspondent initial condition with the correct sign in mathematica and try to compare with Gevolution,
The result when we provide the initial condition for the Φ at z=100 and Φ' at z=98 for taking positive, negative or zero is as following,

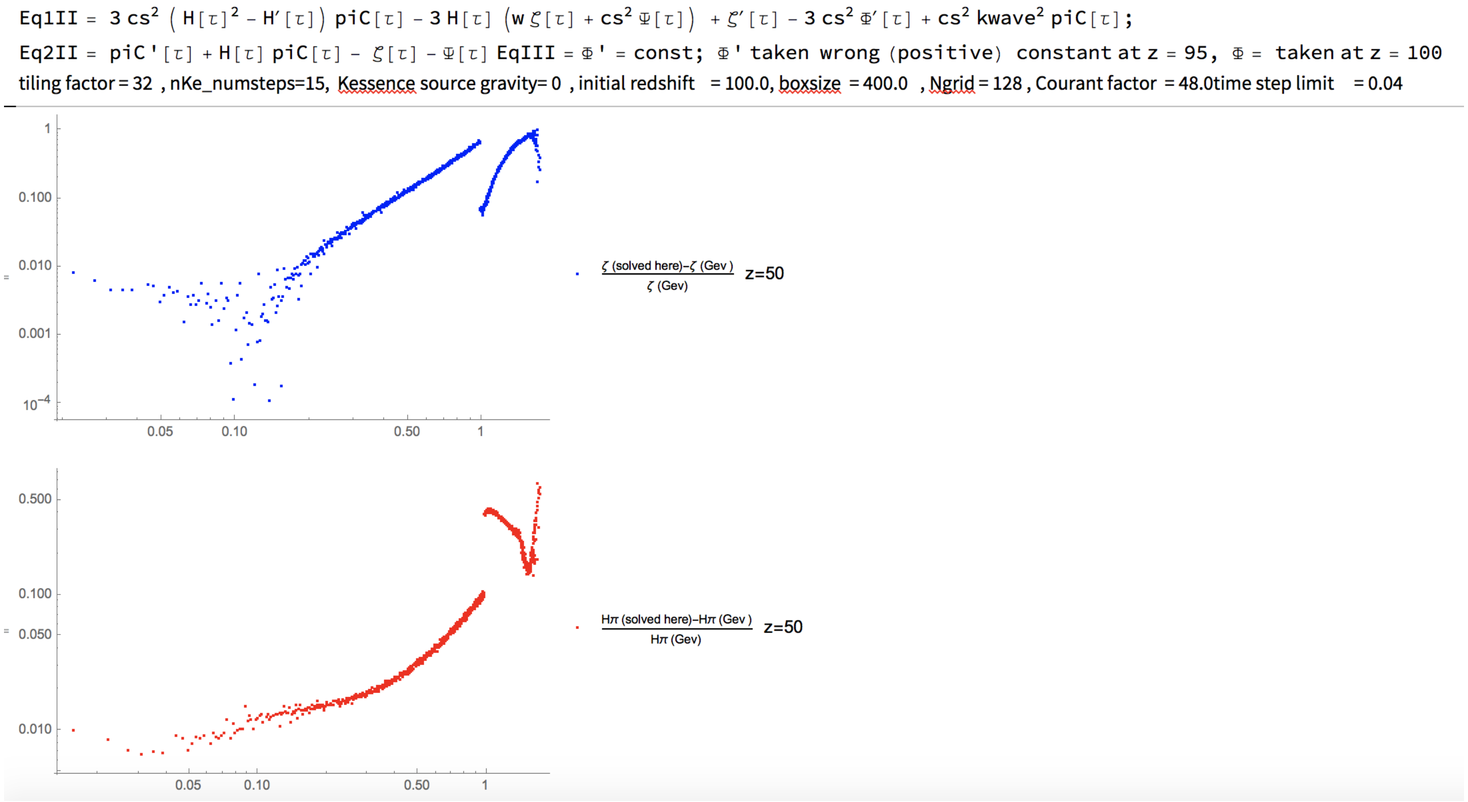


$\mathrm{Eq1II} = 3\,cs^2\left(H[\tau]^2 - H'[\tau]\right)\mathrm{piC}[\tau] - 3\,H[\tau]\left(w\,\zeta[\tau] + cs^2\,\Psi[\tau]\right) + \zeta'[\tau] - 3\,cs^2\,\Phi'[\tau] + cs^2\,\mathrm{kwave}^2\,\mathrm{piC}[\tau];$

$\mathrm{Eq2II} = \mathrm{piC}'[\tau] + H[\tau]\,\mathrm{piC}[\tau] - \zeta[\tau] - \Psi[\tau]$ $\mathrm{EqIII} = \Phi' =$ negative; Φ' at $z = 100$ taken at $z = 95$, $\Phi =$ taken at $z = 100$

tiling factor = 32 , nKe_numsteps=15, Kessence source gravity= 0 , initial redshift = 100.0, boxsize = 400.0 , Ngrid = 128 , Courant factor = 48.0time step limit = 0.04

1
0.500
0.100
0.050
0.010
0.005
0.05
0.10
0.50
1
ζ (solved here)−ζ (Gev) / ζ (Gev) z=50
0.100
0.050
0.010
0.005
0.05
0.10
0.50
1
Hπ (solved here)−Hπ (Gev) / Hπ (Gev) z=50

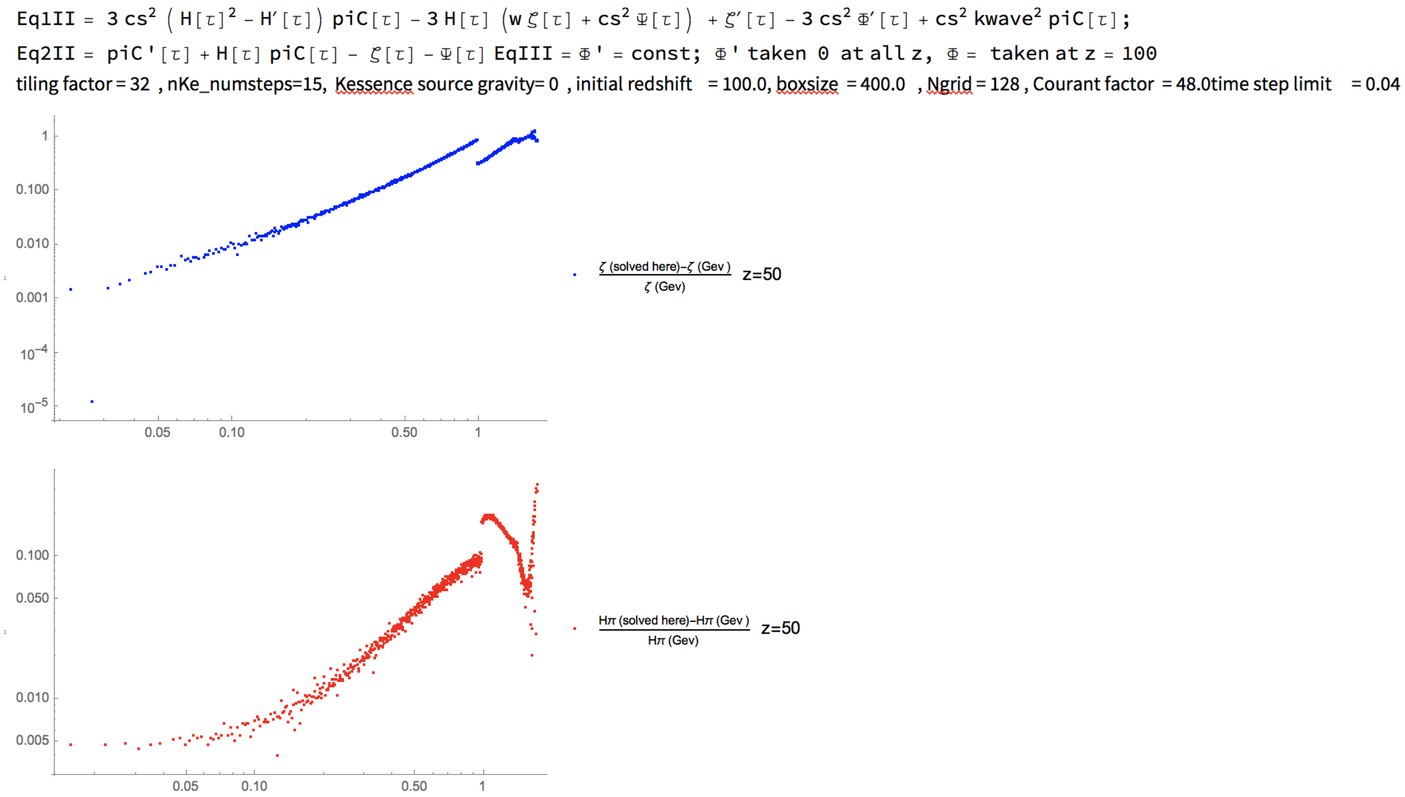


$\mathrm{Eq1II} = 3\,cs^2\left(H[\tau]^2 - H'[\tau]\right)\mathrm{piC}[\tau] - 3\,H[\tau]\left(w\,\zeta[\tau] + cs^2\,\Psi[\tau]\right) + \zeta'[\tau] - 3\,cs^2\,\Phi'[\tau] + cs^2\,\mathrm{kwave}^2\,\mathrm{piC}[\tau];$

$\mathrm{Eq2II} = \mathrm{piC}'[\tau] + H[\tau]\,\mathrm{piC}[\tau] - \zeta[\tau] - \Psi[\tau]$ $\mathrm{EqIII} = \Phi' =$ const; Φ' taken wrong (positive) constant at $z = 95$, $\Phi =$ taken at $z = 100$

tiling factor = 32 , nKe_numsteps=15, Kessence source gravity= 0 , initial redshift = 100.0, boxsize = 400.0 , Ngrid = 128 , Courant factor = 48.0time step limit = 0.04

1
0.100
0.010
0.001
10^{-4}
0.05
0.10
0.50
1
ζ (solved here)−ζ (Gev) / ζ (Gev) z=50
0.500
0.100
0.050
0.010
0.05
0.10
0.50
1
Hπ (solved here)−Hπ (Gev) / Hπ (Gev) z=50



```
Eq1II = 3 cs² ( H[τ]² - H′[τ]) piC[τ] - 3 H[τ] (w ζ[τ] + cs² Ψ[τ]) + ζ′[τ] - 3 cs² Φ′[τ] + cs² kwave² piC[τ];
Eq2II = piC'[τ] + H[τ] piC[τ] - ζ[τ] - Ψ[τ] EqIII = Φ' = const; Φ' taken 0 at all z, Φ = taken at z = 100
tiling factor = 32 , nKe_numsteps=15, Kessence source gravity= 0 , initial redshift = 100.0, boxsize = 400.0 , Ngrid = 128 , Courant factor = 48.0time step limit = 0.04
```

The conclusion from the relative error on $\mathcal{H}\pi$ would be we better take $\Phi' = 0$ to take constant at z=95, since in the first case we are also changing Φ with a constant rate at all redshifts which is not the same as z=95 necessarily! While we improve the relative error on ζ by taking the true behaviour for Φ' which is negative! The reason is not very clear, since these are coupled equations!

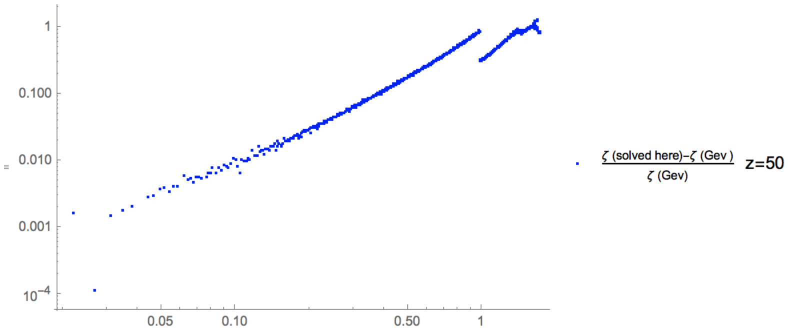
We need to make a better comparison to make sure the one we think is the correct one (negative)?

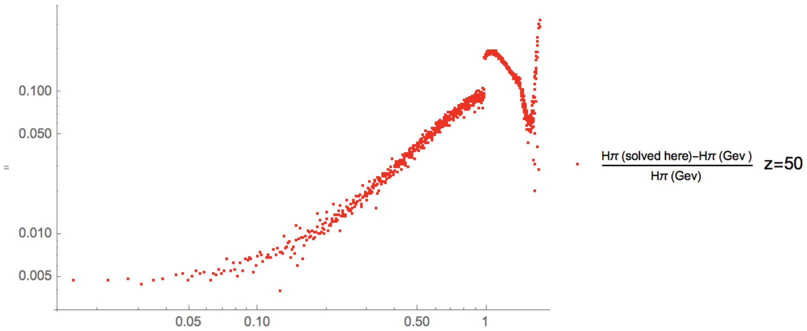
Just note that when we set Φ' a constant and the value at z=50 for example instead of also changing Φ according to $\Phi' = c$, we get better solution which shows that the relative error is not because of making mistake on Φ' and Φ

$$\text{Eq1II} = 3\,\text{cs}^2\left(\text{H}[\tau]^2 - \text{H}'[\tau]\right)\text{piC}[\tau] - 3\,\text{H}[\tau]\left(\text{w}\,\zeta[\tau] + \text{cs}^2\,\Psi[\tau]\right) + \zeta'[\tau] - 3\,\text{cs}^2\,\Phi'[\tau] + \text{cs}^2\,\text{kwave}^2\,\text{piC}[\tau];$$

$$\text{Eq2II} = \text{piC}'[\tau] + \text{H}[\tau]\,\text{piC}[\tau] - \zeta[\tau] - \Psi[\tau],\quad \Phi'\ \text{taken at all}\ z = 50,\quad \Phi =\ \text{taken at}\ z = 100$$

tiling factor = 32 , nKe_numsteps=15, Kessence source gravity= 0 , initial redshift = 100.0, boxsize = 400.0 , Ngrid = 128 , Courant factor = 48.0time step limit = 0.04





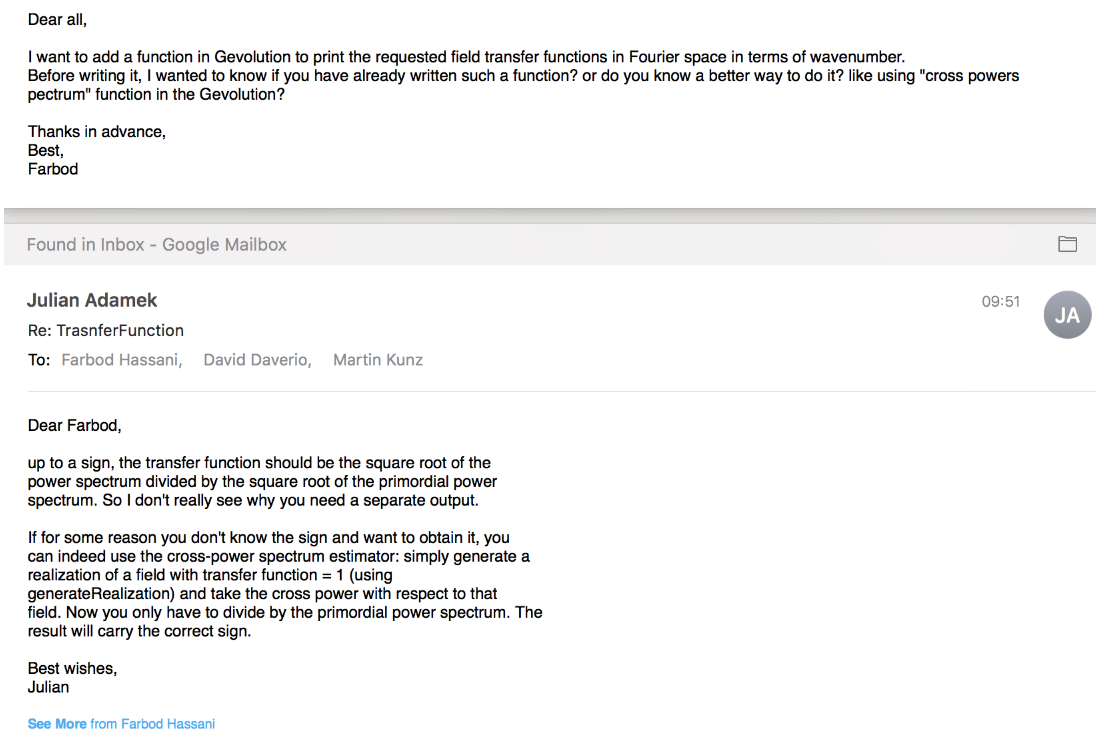
The conclusion after some tests is:
Since we have coupled equations we cannot certainly say what happens if we make Φ or Φ' better estimation. But something which is really clear is that although we do not provide the true initial conditions for Φ and Φ' and Φ'', since its not possible, but the relative error is really acceptable and is clear which comes from the fact that we could not provide complete differential equations. Moreover we have tested our previous argument by letting $\Phi = \Phi = 0$ which we got the right answer in Gevolution and Mathematica. So we do not need to be so sensitive to the decrease the relative error more!!

3 Adding a function for getting field transfer function as the output

Here we try to write a function in output.hpp to print out all the requested field transfer functions in Fourier space into the output text.
This is very important since we want to precisely compare the Gevolution result with Mathematica, while using powerspectrum and extracting trasfer-function from it, does not allow as we have already seen couple of examples which we get wrong results because the field was negative-positive while we take it just positive in mathematica...
instead of writing a new function we can use the extractCrossSpectrum func-

tion in tools.hpp and think of a function which if is crossed with our desired function just gives the Fourier transform of the desired function! It is clear that the other function in real space must be constant (like=1)! So what we do is defining a new constant function which is just used for calculating the transfer function! To do so, we need to make sure if we have understood the cross powerspectrum function correctly, because if it takes average over wavenumbers when we set one function to a constant we get zero since the average of the fields is zero!

The discussion with Julian for the solution is as following,



Dear all,

I want to add a function in Gevolution to print the requested field transfer functions in Fourier space in terms of wavenumber.
Before writing it, I wanted to know if you have already written such a function? or do you know a better way to do it? like using "cross powers pectrum" function in the Gevolution?

Thanks in advance,
Best,
Farbod

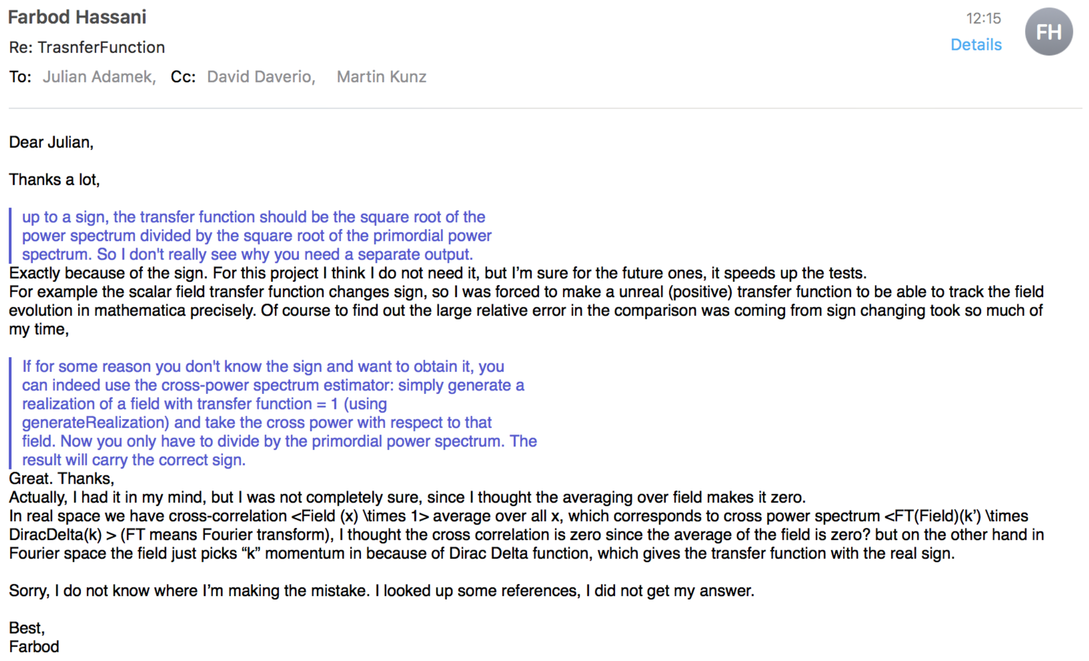
Julian Adamek 09:51
Re: TrasnferFunction
To: Farbod Hassani, David Daverio, Martin Kunz

Dear Farbod,

up to a sign, the transfer function should be the square root of the power spectrum divided by the square root of the primordial power spectrum. So I don't really see why you need a separate output.

If for some reason you don't know the sign and want to obtain it, you can indeed use the cross-power spectrum estimator: simply generate a realization of a field with transfer function = 1 (using generateRealization) and take the cross power with respect to that field. Now you only have to divide by the primordial power spectrum. The result will carry the correct sign.

Best wishes,
Julian



Farbod Hassani
Re: TrasnferFunction
To: Julian Adamek, Cc: David Daverio, Martin Kunz
12:15

Dear Julian,

Thanks a lot,

> up to a sign, the transfer function should be the square root of the power spectrum divided by the square root of the primordial power spectrum. So I don't really see why you need a separate output.

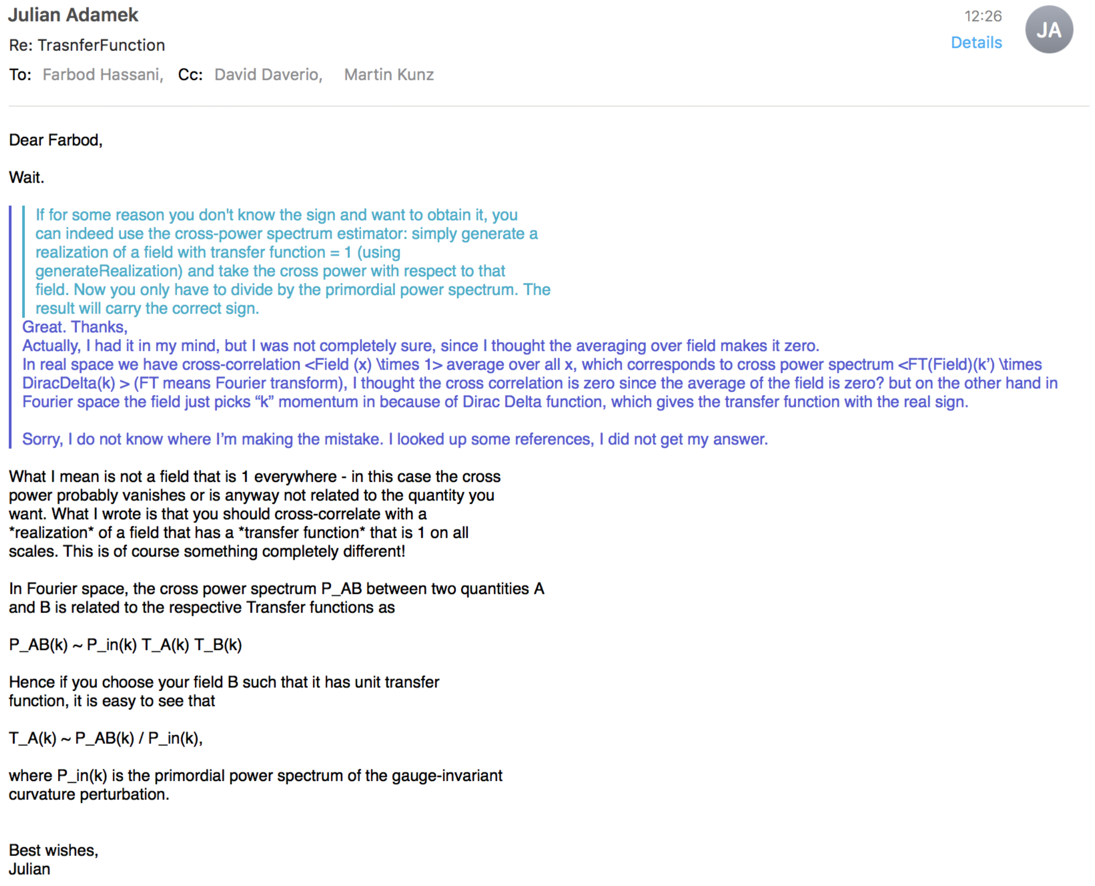
Exactly because of the sign. For this project I think I do not need it, but I'm sure for the future ones, it speeds up the tests.
For example the scalar field transfer function changes sign, so I was forced to make a unreal (positive) transfer function to be able to track the field evolution in mathematica precisely. Of course to find out the large relative error in the comparison was coming from sign changing took so much of my time,

> If for some reason you don't know the sign and want to obtain it, you can indeed use the cross-power spectrum estimator: simply generate a realization of a field with transfer function = 1 (using generateRealization) and take the cross power with respect to that field. Now you only have to divide by the primordial power spectrum. The result will carry the correct sign.

Great. Thanks,
Actually, I had it in my mind, but I was not completely sure, since I thought the averaging over field makes it zero.
In real space we have cross-correlation <Field (x) \times 1> average over all x, which corresponds to cross power spectrum <FT(Field)(k') \times DiracDelta(k) > (FT means Fourier transform), I thought the cross correlation is zero since the average of the field is zero? but on the other hand in Fourier space the field just picks "k" momentum in because of Dirac Delta function, which gives the transfer function with the real sign.

Sorry, I do not know where I'm making the mistake. I looked up some references, I did not get my answer.

Best,
Farbod



Julian Adamek
Re: TrasnferFunction
To: Farbod Hassani, Cc: David Daverio, Martin Kunz
12:26

Dear Farbod,

Wait.

> > If for some reason you don't know the sign and want to obtain it, you can indeed use the cross-power spectrum estimator: simply generate a realization of a field with transfer function = 1 (using generateRealization) and take the cross power with respect to that field. Now you only have to divide by the primordial power spectrum. The result will carry the correct sign.
>
> Great. Thanks,
> Actually, I had it in my mind, but I was not completely sure, since I thought the averaging over field makes it zero.
> In real space we have cross-correlation <Field (x) \times 1> average over all x, which corresponds to cross power spectrum <FT(Field)(k') \times DiracDelta(k) > (FT means Fourier transform), I thought the cross correlation is zero since the average of the field is zero? but on the other hand in Fourier space the field just picks "k" momentum in because of Dirac Delta function, which gives the transfer function with the real sign.
>
> Sorry, I do not know where I'm making the mistake. I looked up some references, I did not get my answer.

What I mean is not a field that is 1 everywhere - in this case the cross power probably vanishes or is anyway not related to the quantity you want. What I wrote is that you should cross-correlate with a *realization* of a field that has a *transfer function* that is 1 on all scales. This is of course something completely different!

In Fourier space, the cross power spectrum P_AB between two quantities A and B is related to the respective Transfer functions as

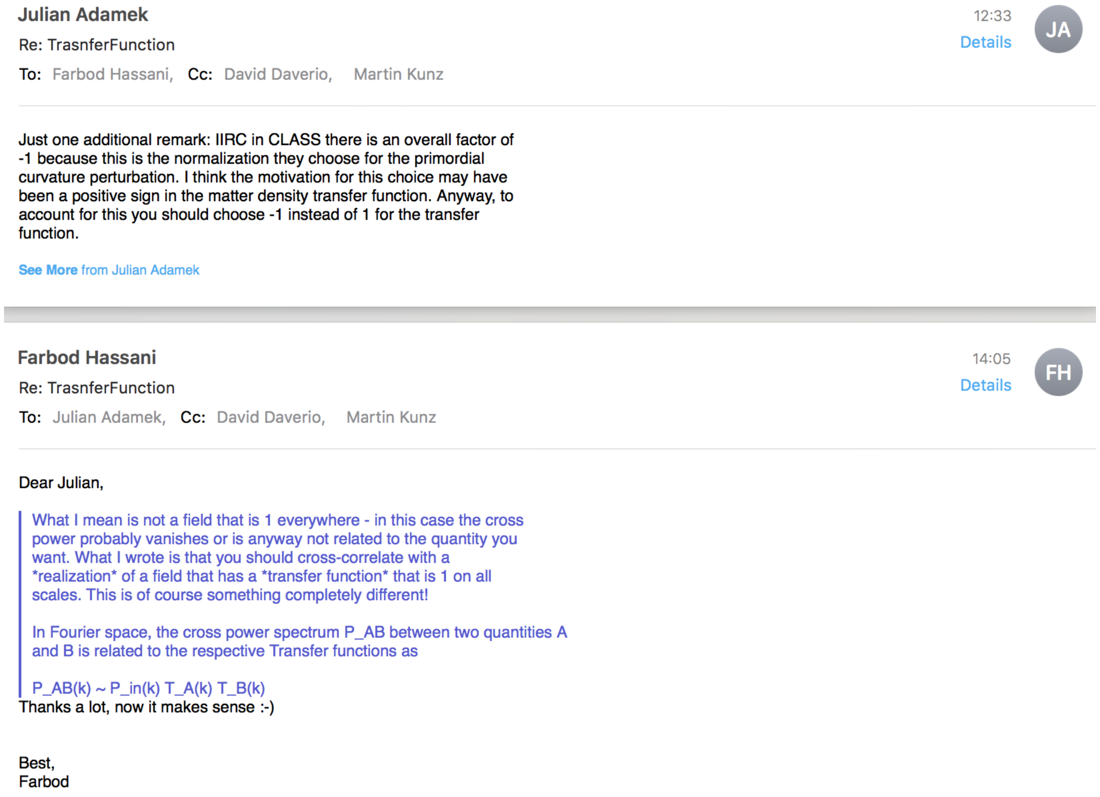
P_AB(k) ~ P_in(k) T_A(k) T_B(k)

Hence if you choose your field B such that it has unit transfer function, it is easy to see that

T_A(k) ~ P_AB(k) / P_in(k),

where P_in(k) is the primordial power spectrum of the gauge-invariant curvature perturbation.

Best wishes,
Julian



Julian Adamek

Re: TrasnferFunction

To: Farbod Hassani, Cc: David Daverio, Martin Kunz

Just one additional remark: IIRC in CLASS there is an overall factor of -1 because this is the normalization they choose for the primordial curvature perturbation. I think the motivation for this choice may have been a positive sign in the matter density transfer function. Anyway, to account for this you should choose -1 instead of 1 for the transfer function.

See More from Julian Adamek

Farbod Hassani

Re: TrasnferFunction

To: Julian Adamek, Cc: David Daverio, Martin Kunz

Dear Julian,

> What I mean is not a field that is 1 everywhere - in this case the cross power probably vanishes or is anyway not related to the quantity you want. What I wrote is that you should cross-correlate with a *realization* of a field that has a *transfer function* that is 1 on all scales. This is of course something completely different!
>
> In Fourier space, the cross power spectrum P_AB between two quantities A and B is related to the respective Transfer functions as
>
> P_AB(k) ~ P_in(k) T_A(k) T_B(k)

Thanks a lot, now it makes sense :-)

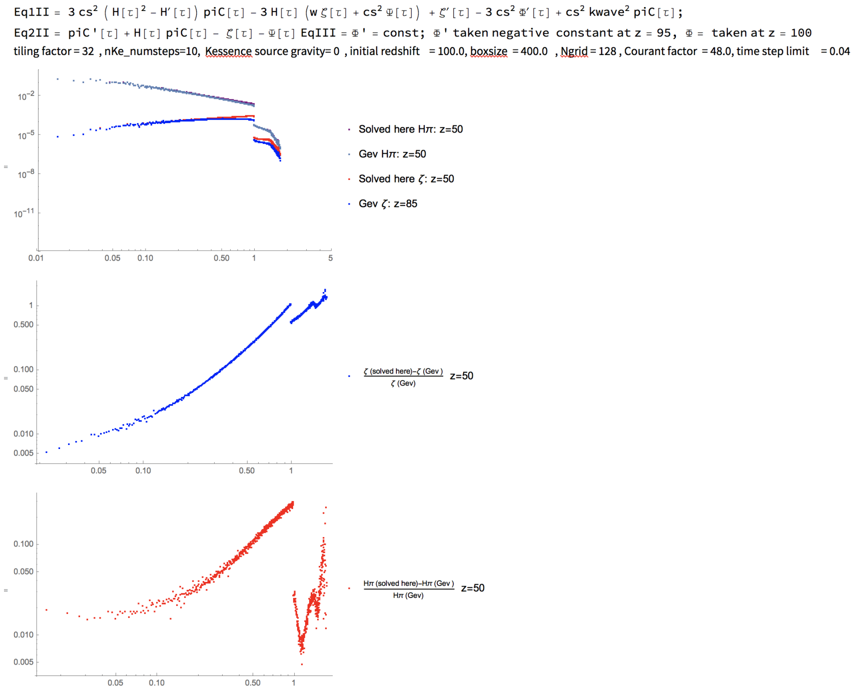
Best,
Farbod

3.1 Field dynamics and stress tensor while kessence source gravity

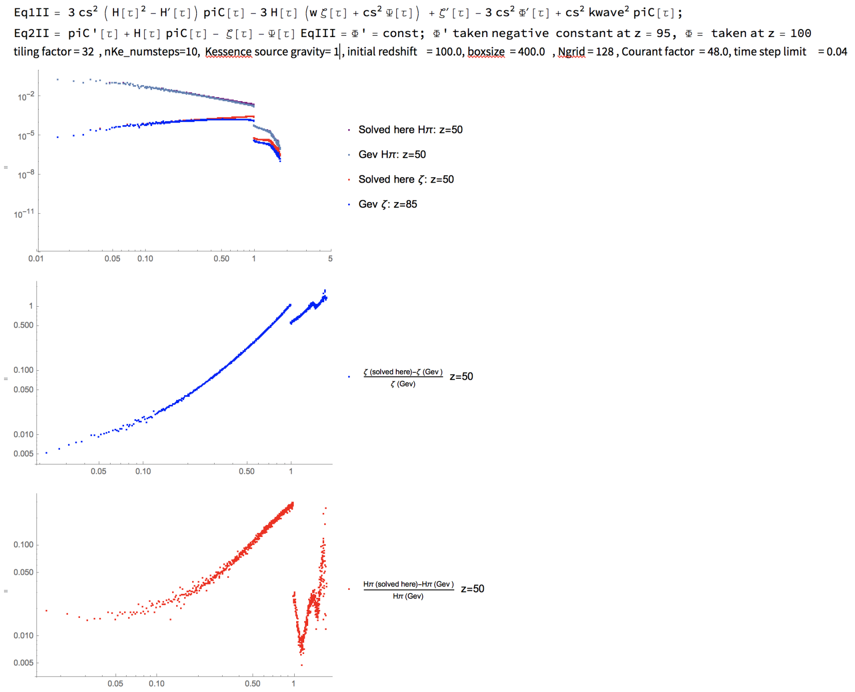
Accepting the fact that we are solving the equations correctly, we can turn on kessence source gravity to see the effect of kessence on the total stress tensor is resonable and can we get something? We again provide the initial condition at z=100 except Φ' which is provided at z=95, and look at the solution of field and stress tensor. The fields relative error would be as following which shows that the backreaction effect from the fields to themselves is negligible, since we get more or less the same plot when it was off!

Note that if we get at z=10 we see a lot of difference because we make a mistake on Φ and Φ'! So we compare at z=50

The comparison at z=50 for when the kessence is turned on and off is as following,



```
Eq1II = 3 cs^2 ( H[τ]^2 – H′[τ]) piC[τ] – 3 H[τ] (w ζ[τ] + cs^2 Ψ[τ]) + ζ′[τ] – 3 cs^2 Φ′[τ] + cs^2 kwave^2 piC[τ];
Eq2II = piC'[τ] + H[τ] piC[τ] – ζ[τ] – Ψ[τ] EqIII = Φ' = const; Φ' taken negative constant at z = 95, Φ = taken at z = 100
tiling factor = 32 , nKe_numsteps=10, Kessence source gravity= 0 , initial redshift = 100.0, boxsize = 400.0 , Ngrid = 128 , Courant factor = 48.0, time step limit = 0.04
```



```
Eq1II = 3 cs^2 ( H[τ]^2 – H′[τ]) piC[τ] – 3 H[τ] (w ζ[τ] + cs^2 Ψ[τ]) + ζ′[τ] – 3 cs^2 Φ′[τ] + cs^2 kwave^2 piC[τ];
Eq2II = piC'[τ] + H[τ] piC[τ] – ζ[τ] – Ψ[τ] EqIII = Φ' = const; Φ' taken negative constant at z = 95, Φ = taken at z = 100
tiling factor = 32 , nKe_numsteps=10, Kessence source gravity= 1, initial redshift = 100.0, boxsize = 400.0 , Ngrid = 128 , Courant factor = 48.0, time step limit = 0.04
```

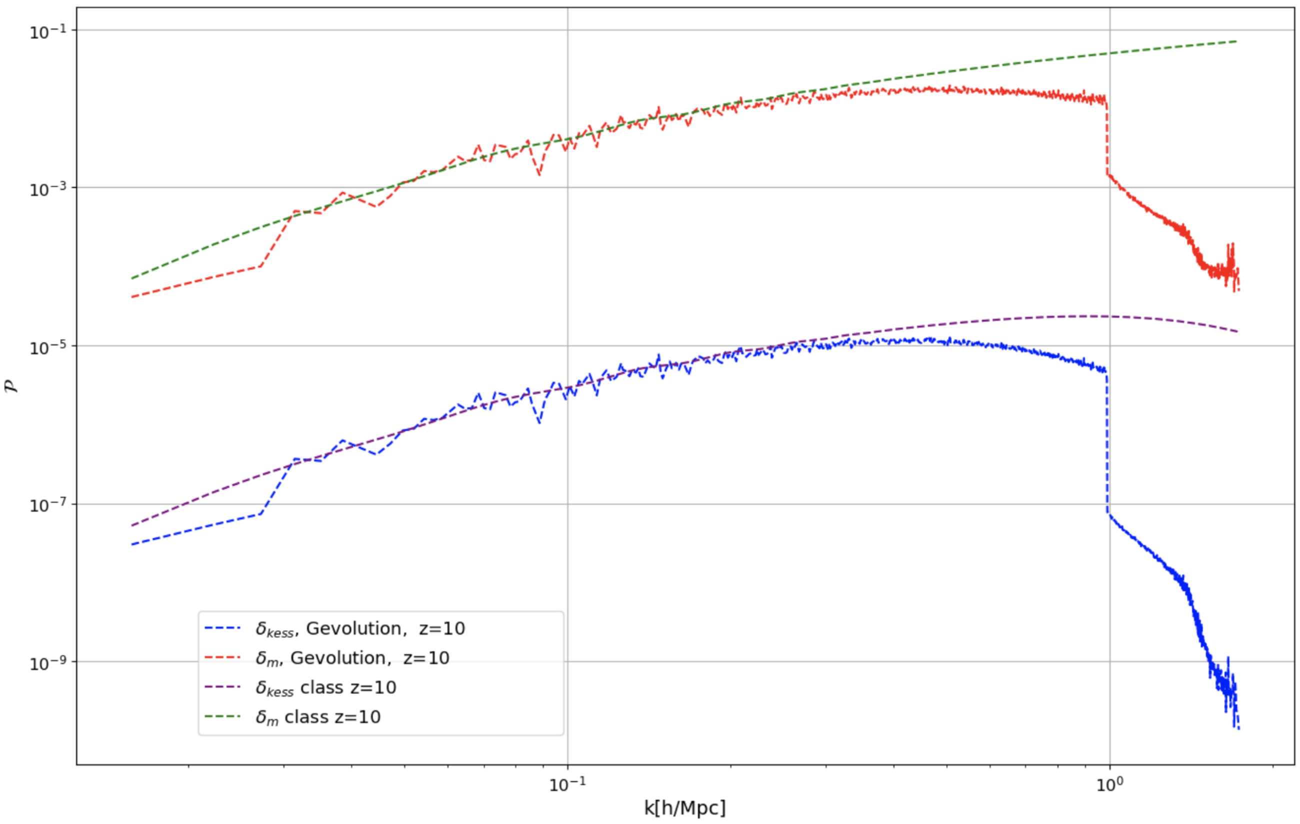
As it is clear we do not see any difference on the field behaviour! On the other hand the backreaction from kessence stress tensor to the fields is negligible!
Comparing the δ_{kess} fro when it sources gravity and when it does not, give the same solution! To make sure we are getting the right solution we need to compare Gevolution δ_{kess} versus class or some other tests..

4 The ratio of δ_{kess} in Gevolution and total stress tensor compared with class for different c_s^2 limits

To see the effect of kessence on total stress tensor we measure the $\delta_{kess}/\delta_{tot}$, this is a very good check specially for the limit $c_s^2 \rightarrow 0$ which we expect the same behaviour from both!
We now shift to the python and try to compare the plots from class with Gevolution, first we interpolate class data on Gevolution wavenumbers for a better comparison as following in the python,

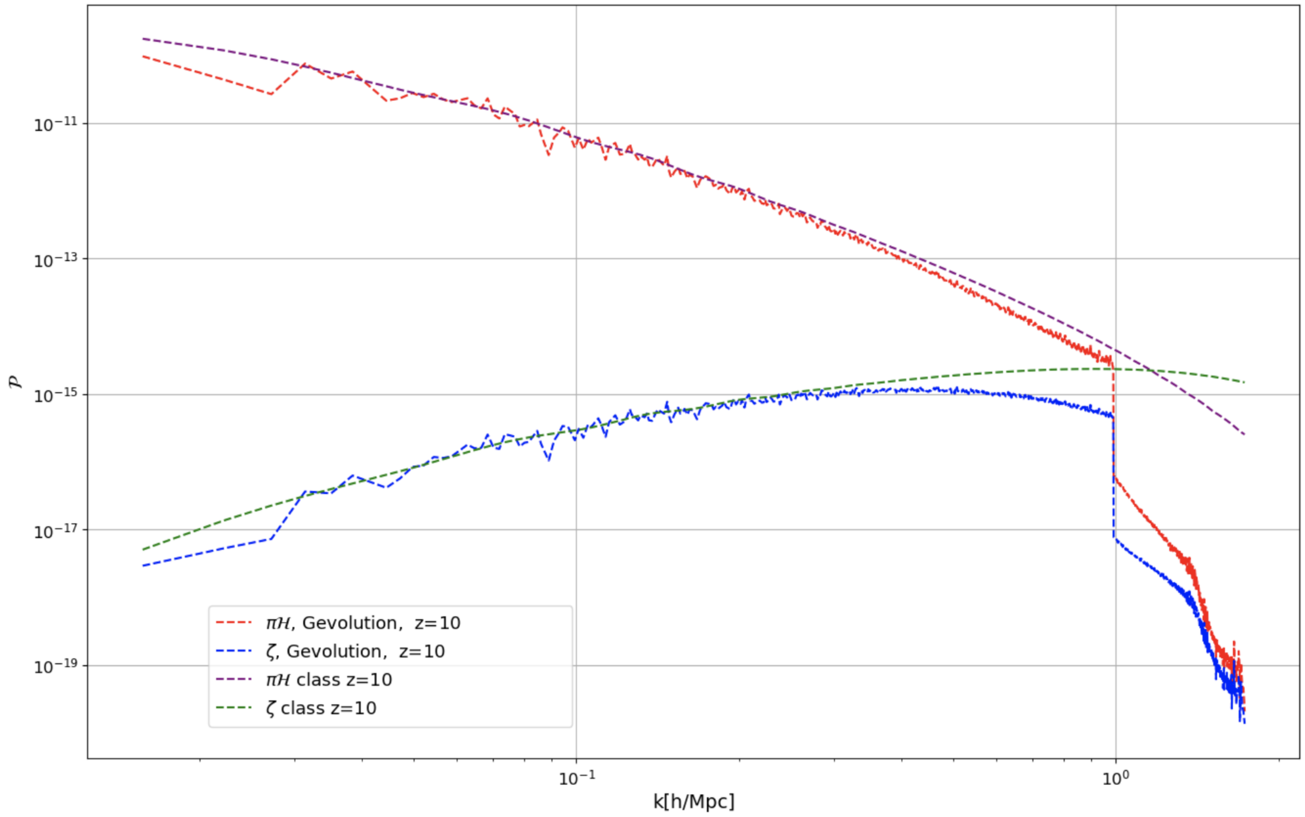
```
  # params for makin power dimensionless
# H_0 in Gevilution unit.
h=0.67556
Boxsize=320.;
c=2997.; #[100km/s]
HGev=np.sqrt(Boxsize**2/c**2) #0.10677, H0 Gevolution in code's unit
def Hubble_conf_Mpc(a):
    H0=0.00022593979933110373;w=-0.9;h=0.67556;
    Omega_b=0.022032/h/h; Omega_cdm=0.12038/h/h;
    Omega_m=Omega_b+Omega_cdm; Omega_Lambda=0.0;
    Omega_rad=9.16681e-05; Omega_kessence=1.-Omega_m-Omega_rad;
    return H0*np.sqrt(Omega_m*(a**-3)+Omega_rad*(a**-4)+Omega_Lambda+Omega_kessence*(a**(-3*(1+w))))*a
#################################
#Class Hubble factor, H in unit 1/Mpc!
# It is phsyical hubble, to make it conformal need to multiply to a. Hconf = H_phys * a
H_conf_class_z100=Hubble_conf_Mpc(1./(1.+100.)); # Unit=1/Mpc Hconf=a*H
H_conf_class_z10=Hubble_conf_Mpc(1./(1.+10.)); # Unit=1/Mpc Hconf=a*H
H_conf_class_z1=Hubble_conf_Mpc(1./(1.+1.)); # Unit=1/Mpc Hconf=a*H
H_conf_class_z0=Hubble_conf_Mpc(1./(1.+0.)); # Unit=1/Mpc Hconf=a*H
#################################
# Parameters for converting to dimensionless power.
As=2.19*10**-9;
h=0.67556
kp=0.05/h;
ns=0.96;
cs2=1.e-6;
#################################
#Making power of class field to compare with Gev
Class_power_z100=np.zeros((np.shape(Gev_deltakess_z02)[0],8))
Class_power_z10=np.zeros((np.shape(Gev_deltakess_z02)[0],8))
Class_power_z1=np.zeros((np.shape(Gev_deltakess_z02)[0],8))
Class_power_z0=np.zeros((np.shape(Gev_deltakess_z02)[0],8))
# Making interpolation, for delta!
for i in range (1,7):
    interp_class_100 =interp1d(class_newt_z100[:,0],class_newt_z100[:,i])
    interp_class_10 =interp1d(class_newt_z10[:,0],class_newt_z10[:,i])
    interp_class_1 =interp1d(class_newt_z1[:,0],class_newt_z1[:,i])
    interp_class_0 =interp1d(class_newt_z0[:,0],class_newt_z0[:,i])

    Class_power_z100[:,i]=As*((interp_class_100(Gev_deltakess_z02[:,0]))**2)*((Gev_deltakess_z02[:,0]/kp)
         **(ns-1.));
    Class_power_z10[:,i]=As*((interp_class_10(Gev_deltakess_z02[:,0]))**2)*((Gev_deltakess_z02[:,0]/kp)**(
         ns-1.));
    Class_power_z1[:,i]=As*((interp_class_1(Gev_deltakess_z02[:,0]))**2)*((Gev_deltakess_z02[:,0]/kp)**(ns
         -1.));
    Class_power_z0[:,i]=As*((interp_class_0(Gev_deltakess_z02[:,0]))**2)*((Gev_deltakess_z02[:,0]/kp)**(ns
         -1.));
```

After comparison, we get the following result which is completely consistent!



4.1 Comparing class and Gevolution with initial condition from class Must be completed for more redshift ranges and also big runs!

Giving initial condition from Class, comparing Class with mathematica at redshift z=10 gives the following result which shows full agreement, Note that the initial condition of class and Gevolution are slightly different



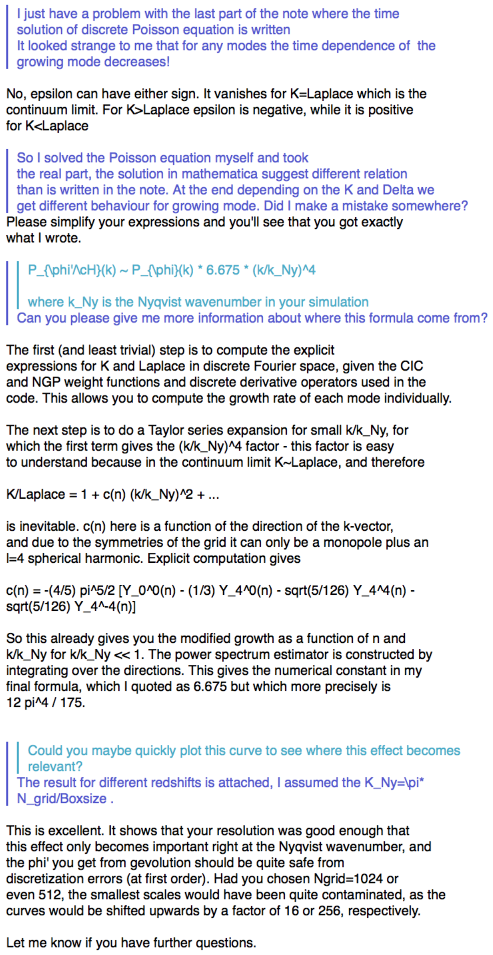
Question: why we get less power at high wavenumbers in Gevolution than Class? Nyqvist frequency?

4.2 Checking the limit $c_s^2 \to 0$ that we get kessence density as matter density

-For the limit $c_s^2 \to 0$ check we get matter power from Gevolution compared with class!
We have checked that we get such a behaviour in class, do we get in Gevolution too? For not very small c_s^2 yes, but it is also a good check to see this in smaller sound speeds. This is a consistency check but not so much necessary for our purposes!

5 Non-linear contrubution

Here we write the full equations and try to solve them numerically in Gevolution, and if we got something intreesting or strange we need to solve them in mathematica to see if we did not have make a mistake !
To solve in mathematica we must solve with Non -linear solve command and to exactly compare with Gevolution we can get some symmetric situations. The discussion with Julian is as following,



> I just have a problem with the last part of the note where the time
> solution of discrete Poisson equation is written
> It looked strange to me that for any modes the time dependence of the
> growing mode decreases!

No, epsilon can have either sign. It vanishes for K=Laplace which is the
continuum limit. For K>Laplace epsilon is negative, while it is positive
for K<Laplace

> So I solved the Poisson equation myself and took
> the real part, the solution in mathematica suggest different relation
> than is written in the note. At the end depending on the K and Delta we
> get different behaviour for growing mode. Did I make a mistake somewhere?

Please simplify your expressions and you'll see that you got exactly
what I wrote.

> > P_{\phi'/\cH}(k) ~ P_{\phi}(k) * 6.675 * (k/k_Ny)^4
> >
> > where k_Ny is the Nyqvist wavenumber in your simulation
>
> Can you please give me more information about where this formula come from?

The first (and least trivial) step is to compute the explicit
expressions for K and Laplace in discrete Fourier space, given the CIC
and NGP weight functions and discrete derivative operators used in the
code. This allows you to compute the growth rate of each mode individually.

The next step is to do a Taylor series expansion for small k/k_Ny, for
which the first term gives the (k/k_Ny)^4 factor - this factor is easy
to understand because in the continuum limit K~Laplace, and therefore

K/Laplace = 1 + c(n) (k/k_Ny)^2 + ...

is inevitable. c(n) here is a function of the direction of the k-vector,
and due to the symmetries of the grid it can only be a monopole plus an
l=4 spherical harmonic. Explicit computation gives

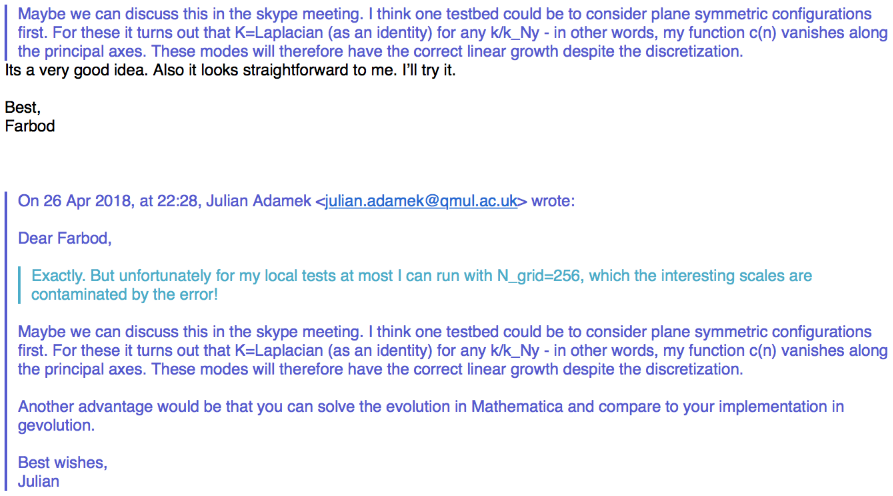
c(n) = -(4/5) pi^5/2 [Y_0^0(n) - (1/3) Y_4^0(n) - sqrt(5/126) Y_4^4(n) -
sqrt(5/126) Y_4^-4(n)]

So this already gives you the modified growth as a function of n and
k/k_Ny for k/k_Ny << 1. The power spectrum estimator is constructed by
integrating over the directions. This gives the numerical constant in my
final formula, which I quoted as 6.675 but which more precisely is
12 pi^4 / 175.

> > Could you maybe quickly plot this curve to see where this effect becomes
> > relevant?
>
> The result for different redshifts is attached, I assumed the K_Ny=\pi*
> N_grid/Boxsize .

This is excellent. It shows that your resolution was good enough that
this effect only becomes important right at the Nyqvist wavenumber, and
the phi' you get from gevolution should be quite safe from
discretization errors (at first order). Had you chosen Ngrid=1024 or
even 512, the smallest scales would have been quite contaminated, as the
curves would be shifted upwards by a factor of 16 or 256, respectively.

Let me know if you have further questions.



> Maybe we can discuss this in the skype meeting. I think one testbed could be to consider plane symmetric configurations first. For these it turns out that K=Laplacian (as an identity) for any k/k_Ny - in other words, my function c(n) vanishes along the principal axes. These modes will therefore have the correct linear growth despite the discretization.

Its a very good idea. Also it looks straightforward to me. I'll try it.

Best,
Farbod

> On 26 Apr 2018, at 22:28, Julian Adamek <julian.adamek@qmul.ac.uk> wrote:
>
> Dear Farbod,
>
> > Exactly. But unfortunately for my local tests at most I can run with N_grid=256, which the interesting scales are contaminated by the error!
>
> Maybe we can discuss this in the skype meeting. I think one testbed could be to consider plane symmetric configurations first. For these it turns out that K=Laplacian (as an identity) for any k/k_Ny - in other words, my function c(n) vanishes along the principal axes. These modes will therefore have the correct linear growth despite the discretization.
>
> Another advantage would be that you can solve the evolution in Mathematica and compare to your implementation in gevolution.
>
> Best wishes,
> Julian

6 Sensitivity to initial conditions

What would be error if we set the initial condition at z=100 to zero?

7 The effect of non linearities on gravitational potential at some different redshifts Ψ and matter power spectrum

8 Trace the average of the perturbation to be consistent

9 Vector elliptic and vector parabolic consistency check