Implementation of non-trivial species in CLASS

decaying dark matter, scalar fields, ...

Thomas Tram (thomas.tram@epfl.ch)

Expanding CLASS

- Adding exotic new physics to CLASS is easy*
- New features will be merged into the main code and will be available in all future versions
- This lecture is essentially a complete step by step tutorial for adding decaying dark matter at the level of the code.

**given sufficient understanding of the underlying physics and provided that the rules of CLASS are respected…*

A word on quintessence

- Implemented in a CLASS branch by Miguel Zumalacárregui (ITP - Uni. of Heidelberg)
- Merged into the development version of CLASS this week, available in v2.4

$$\frac{d\phi^2}{dt^2} = -3H\frac{d\phi}{dt} - \frac{dV}{d\phi},$$

$$V(\phi) = M^4[(\phi - B)^\alpha + A]e^{-\lambda\phi/M}$$

- Changes similar to DCDM case

input.c

- Do we have to deal with boundary conditions? (See Julien's lecture 1)

```
int input_init(…){
  …
 /** These two arrays must contain the strings of names to be searched
      for and the coresponding new parameter */
  char * const target_namestrings[] =
    {"100*theta_s","Omega_dcdmdr","omega_dcdmdr","Omega_scf"};
  char * const unknown_namestrings[] =
    {"h","Omega_ini_dcdm","Omega_ini_dcdm","scf_lambda"};
  enum computation_stage target_cs[] =
    {cs_thermodynamics, cs_background, cs_background, cs_background};
  …
}
int input_try_unknown_parameters()
int input_get_guess()
```

background.c

- Setting flags, defining _bg_ and _bi_ indices:

```
int background_indices(…){
  …
  if (pba->Omega0_dcdmdr != 0.){
    pba->has_dcdm = _TRUE_;
    pba->has_dr = _TRUE_;
  }
  …
  /* - index for dcdm */
  class_define_index(pba->index_bg_rho_dcdm,pba->has_dcdm,index_bg,1);
  …
  /* -> energy density in DCDM */
  class_define_index(pba->index_bi_rho_dcdm,pba->has_dcdm,index_bi,1);
  …
}
```

background.c

- Compute {A} variables from {B} variables:

```
int background_functions(…){
  …
  /* dcdm */
  if (pba->has_dcdm == _TRUE_) {
    /* Pass value of rho_dcdm to output */
    pvecback[pba->index_bg_rho_dcdm] = pvecback_B[pba->index_bi_rho_dcdm];
    rho_tot += pvecback[pba->index_bg_rho_dcdm];
    p_tot += 0.;
    rho_m += pvecback[pba->index_bg_rho_dcdm];
  }
  …
}
```

background.c

- Set initial conditions for {B} quantities:

```
int background_initial_conditions(…){
  …
  if (pba->has_dcdm == _TRUE_){
    /* Remember that the critical density today in CLASS is H0^2 */
    pvecback_integration[pba->index_bi_rho_dcdm] =
      pba->Omega_ini_dcdm*pba->H0*pba->H0*pow(pba->a_today/a,3);
  }
  …
}
```

background.c

- Write e.o.m for {B} quantities:

```
int background_derivs(…){
  …
  if (pba->has_dcdm == _TRUE_){
    /** compute dcdm density rho' = -3aH rho - a Gamma rho*/
    dy[pba->index_bi_rho_dcdm] = -3.*y[pba->index_bi_a]*
      pvecback[pba->index_bg_H]*y[pba->index_bi_rho_dcdm]-
      y[pba->index_bi_a]*pba->Gamma_dcdm*y[pba->index_bi_rho_dcdm];
  }
  …
}
```

perturbations.c

- Initialise perturbation vector:

```
int perturb_vector_init(…){
  /* dcdm */
  class_define_index(ppv->index_pt_delta_dcdm,pba->has_dcdm,index_pt,1);
  class_define_index(ppv->index_pt_theta_dcdm,pba->has_dcdm,index_pt,1);
  …
  if (pba->has_dcdm == _TRUE_) {
    ppv->y[ppv->index_pt_delta_dcdm] =
      ppw->pv->y[ppw->pv->index_pt_delta_dcdm];

    ppv->y[ppv->index_pt_theta_dcdm] =
      ppw->pv->y[ppw->pv->index_pt_theta_dcdm];
  }
}
```

perturbations.c

- Set initial conditions for perturbations.

```
int perturb_initial_conditions(…){
  if (pba->has_dcdm == _TRUE_)
    rho_m += ppw->pvecback[pba->index_bg_rho_dcdm];

}
```

perturbations.c

- Differential equation for perturbations:

```
int perturb_derivs(…){
  if ((pba->has_dcdm == _TRUE_)&&(pba->has_dr == _TRUE_)) {
    /** -> dcdm */
    dy[pv->index_pt_delta_dcdm] =
      -(y[pv->index_pt_theta_dcdm]+metric_continuity);

    dy[pv->index_pt_theta_dcdm] =
      - a_prime_over_a*y[pv->index_pt_theta_dcdm] + metric_euler;
  }
}
```

perturbations.c

- Add contribution to $\delta T^{\mu,\nu}$:

```
int perturb_total_stress_energy(…){
  /* dcdm contribution */
  if (pba->has_dcdm == _TRUE_) {
    ppw->delta_rho += ppw->pvecback[pba->index_bg_rho_dcdm]*
                      y[ppw->pv->index_pt_delta_dcdm];
    ppw->rho_plus_p_theta += ppw->pvecback[pba->index_bg_rho_dcdm]*
                             y[ppw->pv->index_pt_theta_dcdm];
  }
  if (ppt->has_source_delta_m == _TRUE_) {
    /* include decaying cold dark matter */
    if (pba->has_dcdm == _TRUE_) {
      delta_rho_m += ppw->pvecback[pba->index_bg_rho_dcdm]*
        y[ppw->pv->index_pt_delta_dcdm];
      rho_m += ppw->pvecback[pba->index_bg_rho_dcdm];
    }
  }
}
```

perturbations.c

- Print perturbations? (Note: will be moved to `output.c` in an upcoming release.)

```
int perturb_prepare_output_file(…){
  /* Decaying cold dark matter */
  class_fprintf_columntitle(ppw->perturb_output_file, "delta_dcdm",
                            pba->has_dcdm, colnum);
  class_fprintf_columntitle(ppw->perturb_output_file, "theta_dcdm",
                            pba->has_dcdm, colnum);
}
int perturb_print_variables(…){
  /* Decaying cold dark matter */
  class_fprintf_double(ppw->perturb_output_file, delta_dcdm, pba->has_dcdm);
  class_fprintf_double(ppw->perturb_output_file, theta_dcdm, pba->has_dcdm);
}
```

perturbations.c

- Store source functions $S_i(k, \tau)$ for later use?

```
int perturb_indices_of_perturbs(…){
  ppt->has_source_delta_dcdm = _FALSE_;
  ppt->has_source_theta_dcdm = _FALSE_;
  if (ppt->has_density_transfers == _TRUE_) {
    if (pba->has_dcdm == _TRUE_)
      ppt->has_source_delta_dcdm = _TRUE_;
  }
  if (ppt->has_velocity_transfers == _TRUE_) {
    if (pba->has_dcdm == _TRUE_)
      ppt->has_source_theta_dcdm = _TRUE_;
  }
  class_define_index(ppt->index_tp_delta_dcdm, ppt->has_source_delta_dcdm,…);
  class_define_index(ppt->index_tp_theta_dcdm, ppt->has_source_theta_dcdm,…);
}
```

perturbations.c

- Store the sources $S_i(\tau, k)$:

```
int perturb_sources(…){
  /* delta_dcdm */
  if (ppt->has_source_delta_dcdm == _TRUE_) {
    _set_source_(ppt->index_tp_delta_dcdm) = y[ppw->pv->index_pt_delta_dcdm];
  }
  /* theta_dcdm */
  if (ppt->has_source_theta_dcdm == _TRUE_) {
    _set_source_(ppt->index_tp_theta_dcdm) = y[ppw->pv->index_pt_theta_dcdm];
  }
}
```

spectra.c

- Define indices for transfer functions $T_i(k, z)$:

```
int spectra_indices(…){
  /* indices for species ass. with a matter trsf. function in Fourier space*/
  class_define_index(psp->index_tr_delta_dcdm,ppt->has_source_delta_dcdm,…);
  /* indices for species ass. with a vel. trsf. function in Fourier space */
  class_define_index(psp->index_tr_theta_dcdm,ppt->has_source_theta_dcdm,…);
}
```

spectra.c

- Compute matter transfer functions $T_i(k, z)$:

```
int spectra_matter_transfers(…){
  /* T_dcdm(k,tau) */
  if (pba->has_dcdm == _TRUE_) {
    rho_i = pvecback_sp_long[pba->index_bg_rho_dcdm];
      if (ppt->has_source_delta_dcdm == _TRUE_) {
        delta_i = ppt->sources[][(…)+ppt->index_tp_delta_dcdm][(…)+index_k];
        psp->matter_transfer[(…) + psp->index_tr_delta_dcdm] = delta_i;
        delta_rho_tot += rho_i * delta_i;
        rho_tot += rho_i;
      }
      if (ppt->has_source_theta_dcdm == _TRUE_) {
        theta_i = ppt->sources[][(…)+ppt->index_tp_theta_dcdm][(…)+index_k];
        psp->matter_transfer[(…) + psp->index_tr_theta_dcdm] = theta_i;
        rho_plus_p_theta_tot += rho_i * theta_i;
        rho_plus_p_tot += rho_i;
      }
}}}
```

output.c

- Write transfer functions if requested:

```
int output_open_tk_file(…){
  class_fprintf_columntitle(*tkfile,"d_dcdm",pba->has_dcdm,colnum);
  class_fprintf_columntitle(*tkfile,"t_dcdm",pba->has_dcdm,colnum);
}

int output_one_line_of_tk(…) {
  class_fprintf_double(tkfile,
                       tk[psp->index_tr_delta_dcdm],
                       ppt->has_source_delta_dcdm);
  class_fprintf_double(tkfile,
                       tk[psp->index_tr_theta_dcdm],
                       ppt->has_source_theta_dcdm);
}
```