```
Class Diagram for CosMomentum
Last edited: March 5th, 2019
```

```
Class Universe
public attributes:
private attributes:
  int expansion_or_collapse; // 1 == expansion; 0 == collapse
  int number_of_time_steps;
  double t_initial;
  double eta initial;
  double H initial;
  double H_prime_initial;
  double a_initial;
  double a_final;
  vector<double> t;
  vector<double> eta;
  vector<double> a;
  vector<double> H;
  vector<double> H_prime;
  cosmological_model cosmology;
public methods:
  Universe(cosmological_model cosmo, double a_min, double a_max, int expand_or_collapse);
  ~Universe();
 double return_a_initial();
 double return_a_final();
 double return_eta_initial();
 double return eta final();
 cosmological_model return_cosmology();
 void print_background_cosmology(string filename);
 double f_k(double w);
 double t_at_eta(double e);
 double a at eta(double e);
 double H_at_eta(double e);
 double H_prime_at_eta(double e);
 double eta_at_a(double a);
 vector<vector<double> > return_background_expansion();
 vector<vector<double> > return_background_expansion(int conformal_time_steps);
 double rho_m_of_a(double scale); // All in units of TODAYS critical density
 double rho_r_of_a(double scale);
 double rho_L_of_a(double scale);
 double w L of a(double scale);
 static void expansion_in_flat_matter_dominated_universe(double a, double Omega_m, double *t_phys, double *eta, double *H_conformal, double *H_conformal_prime);
 static void expansion_in_flat_radiation_dominated_universe(double a, double *t_phys, double *eta, double *H_conformal, double *H_conformal_prime);
 static void expansion_in_flat_Lambda_dominated_universe(double a, double *t_phys, double *eta, double *H_conformal, double *H_conformal_prime);
private methods:
  void set_initial_conditions();
  void set background cosmology();
  void set_number_of_time_steps(int n_entries);
  double hubble_from_Friedmann(double a_start);
  double hubble_prime_from_Friedmann(double a_start);
                                                                                                    Class Matter
public attributes:
  Universe* universe;
  vector<double> log_top_hat_radii;
  vector<double> top_hat_sphere_variances;
  vector<double> dtop_hat_sphere_variances_dR;
  vector<double> log_top_hat_radii_for_skewnesses;
  vector<double> top_hat_sphere_skewnesses;
  vector<double> dtop_hat_sphere_skewnesses_dR;
private attributes:
  int number of entries Newton;
  int ell_max;
  double a_initial;
  double a_final;
  double eta_initial;
  double eta_final;
  double D initial;
  double D_prime_initial;
  double norm;
  double f_NL_rescaling_factor;
  double top_hat_radius;
  double second_top_hat_radius;
  INITIALISATION skewness_initialisation;
  vector<double> eta_Newton;
  vector<double> eta_NL;
  vector<double> Newtonian_growth_factor_of_delta;
  vector<double> Newtonian_growth_factor_of_delta_prime;
  vector<double> Newtonian_growth_factor_of_delta_prime_prime;
  vector<double> Newtonian_growth_factor_of_delta_prime_prime;
  vector<double> Newtonian_growth_factor_second_order;
  vector<double> Newtonian_growth_factor_second_order_prime;
  vector<double> wave_numbers;
  vector<double> log_wave_numbers;
  vector<double> transfer_function;
  vector<double> P_L_today;
  vector<double> P_NL_today;
  vector<double> delta_values_for_spherical_collapse;
  vector<vector<double> > spherical_collapse_evolution_of_delta;
  vector<vector<double> > spherical_collapse_evolution_of_delta_ddelta;
  vector<vector<double> > spherical collapse evolution of delta ddelta2;
  vector<double> eta_NL_for_spherical_collapse;
  cosmological_model cosmology;
  /* Variables for halofit (Smith++2003, Takahashi++2012) */
  double current k non linear;
  double current_n_eff;
  double current_C_sm;
  double current_scale;
  ouble f_1, f_2, f_3, mun, nun, betan, alphan;
  double current_r_sq;
  vector<double> current_P_NL;
  vector<double> current_P_L;
public methods:
  Matter(Universe* uni);
  Matter(Universe* uni, string file_for_transfer_function);
  ~Matter();
  void print_Newtonian_growth_factor(string file_name);
  void return_delta_NL_of_delta_L_and_dF_ddelta_3D(double eta, vector<double> *delta_L_values, vector<double> *delta_NL_values, vector
  double Newtonian_linear_power_spectrum(double k, double e);
  double transfer_function_at(double k);
  double current_P_NL_at(double ln_k);
double current_P_L_at(double ln_k);
  double P_L_today_at(double In_k);
  double return_linear_variance(double z, double R_in_Mpc_over_h);
  double return_non_linear_variance(double z, double R_in_Mpc_over_h);
  vector<double> P_L(double e);
  vector<double> P NL(double e);
  vector<double> return_wave_numbers();
  void print_P_NL(double w, string output_file);
  void set_spherical_collapse_evolution_of_delta(double z_min, double z_max, int n_time);
  double variance_of_matter_within_R_before_norm_was_determined(double R);
  double variance_of_matter_within_R(double R);
  double dvariance_of_matter_within_R_dR(double R);
  double variance_of_matter_within_R_NL(double R);
  /* Needed for PNG calculation: */
  double skewness_of_matter_within_R(double R, double alpha_1, double alpha_2, double alpha_3);
  double dskewness_of_matter_within_R_dR(double R, double alpha_1, double alpha_2, double alpha_3);
  double return_D_of_eta(double eta);
  double return_D_prime_of_eta(double eta);
  vector<vector<double> > return_linear_growth_history(int conformal_time_steps);
  void growth_of_DM_fluctuations_in_flat_matter_dominated_universe(double a, double *eta, double *D, double *D_prime);
  void growth_of_DM_fluctuations_in_flat_radiation_dominated_universe(double a, double *eta, double *D, double *D_prime);
  void growth_of_DM_fluctuations_in_flat_Lambda_dominated_universe(double a, double *eta, double *D, double *D_prime);
  vector<vector<double> > return_power_spectra(double eta, double R);
  vector<vector<double> > compute_phi_of_lambda_3D(double z, double R, double f_NL, double var_NL_rescale);
  int return N of lambda(){return this->delta_values_for_spherical_collapse.size();};
  void return_2rd_moment_and_derivative(double R, double *variance, double *dvariance_dR);
  void set_sphere_skewnesses(int PNG_modus);
  void set_sphere_skewnesses_from_eps3_powerlaw_approximation(int PNG_modus, double R_0_in_Mpc_over_h);
  void set_sphere_skewnesses_from_file(string file_name);
private methods:
  void set_initial_conditions();
  void set_wave_numbers();
  void set_transfer_function_Eisenstein_and_Hu();
  void set_transfer_function_Bond_and_Efstathiou();
  void set_transfer_function_from_file(string file);
  void set_P_today();
  void set_sphere_variances();
  void initialize_linear_growth_factor_of_delta();
  void initialize_up_to_second_order_growth_factor_of_delta(double D, double D_prime);
  /* Methods for halofit (Smith++2003, Takahashi++2012) */
  double sig_sq(double R, double e);
  vector<double> c_and_n_NL(double R, double e);
  double k NL(double k min, double k max, double e);
  double Delta_Q_sq(double k, double e);
  double Delta_H_sq(double k);
  double Delta_H_prime_sq(double k);
  double P_NL_at(double k, double e);
```

void sig_sq_derivs(vector<double> (*a), vector<double> (*dadt)); void norm_derivs(vector<double> (*a), vector<double> (*dadt)); void c_and_n_derivs(vector<double> (*a), vector<double> (*dadt));

struct cosmological model int collapse; double Omega_m; double Omega_r; double Omega_L; double Omega_b; double Omega_k; double theta 27; double w0; double w1; double n_s; double h_100; double sigma_8;