Numerical project
Milestone IV

Power to the spectrum

Hans Winther 2020

Putting it all together

- We now have all the ingredients we need to compute theoretical predictions for the key observables: CMB angular power spectra and matter power spectra.

Background

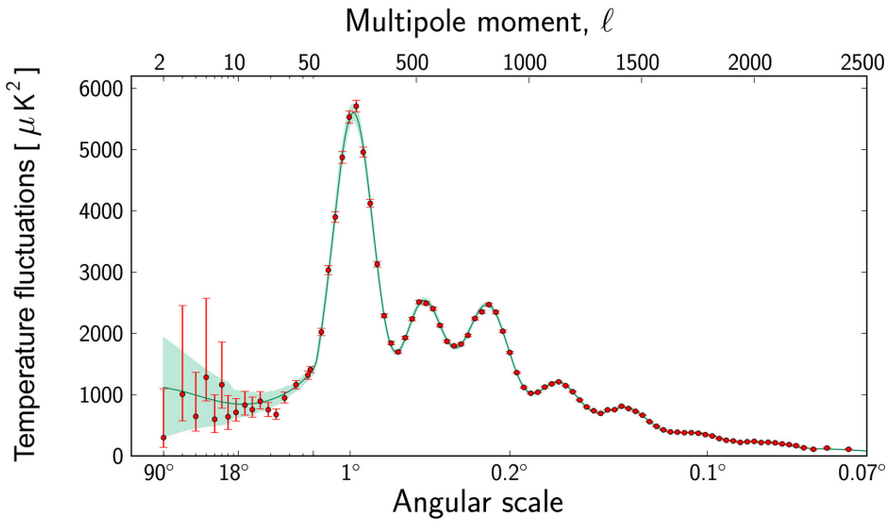
$$\mathscr{H}(x), \eta(x), \Omega_M(x) \ldots$$

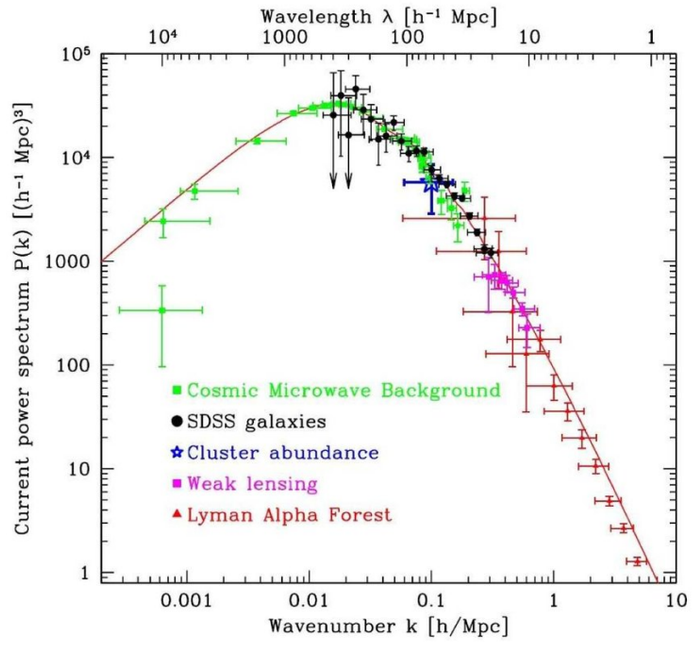
+ Recombination

$$\tau, g, \ldots$$

+ Perturbations

$$\vec{Y} = \begin{pmatrix} \Theta_0 \\ \Theta_1 \\ \Theta_2 \\ \cdots \\ \Theta_{\ell_{\max}} \\ \Phi \\ \delta_b \\ v_b \\ \cdots \end{pmatrix}$$





Theory summary

- The CMB power-spectrum is determined by the photon multipoles

$$C_\ell = 4\pi \int_0^\infty A_s \left(\frac{k}{k_{\rm pivot}}\right)^{n_s-1} \Theta_\ell^2(k) \frac{dk}{k}.$$

- The photon multipoles are found by doing line of sight integration

$$\Theta_\ell(k, x=0) = \int_{-\infty}^{0} \tilde{S}(k,x) j_\ell[k(\eta_0 - \eta)] dx,$$

$$\tilde{S}(k,x) = \tilde{g}\left[\Theta_0 + \Psi + \frac{1}{4}\Pi\right] + e^{-\tau}\left[\Psi' - \Phi'\right] - \frac{1}{ck}\frac{d}{dx}(\mathcal{H}\tilde{g}v_b) + \frac{3}{4c^2k^2}\frac{d}{dx}\left[\mathcal{H}\frac{d}{dx}(\mathcal{H}\tilde{g}\Pi)\right].$$

- Where the source function is determined by the perturbations you have already computed

The PowerSpectrum class

Inflationary parameters

List of the Cells we are computing Cell (you can cut this down to 1400 if you want)

```
class PowerSpectrum {
  private:

    BackgroundCosmology *cosmo = nullptr;
    RecombinationHistory *rec  = nullptr;
    Perturbations *pert        = nullptr;

    // Parameters defining the primordial power-spectrum
    double A_s        = 2e-9;
    double n_s        = 0.96;
    double kpivot_mpc = 0.05;

    // The k-values we compute Theta_ell(k) etc. for
    const int n_k      = 100;
    const double k_min = Constants.k_min;
    const double k_max = Constants.k_max;

    // The ells's we will compute Theta_ell and Cell for
    Vector ells{
        2,    3,    4,    5,    6,    7,    8,    10,   12,   15,
        20,   25,   30,   40,   50,   60,   70,   80,   90,   100,
        120,  140,  160,  180,  200,  225,  250,  275,  300,  350,
        400,  450,  500,  550,  600,  650,  700,  750,  800,  850,
        900,  950,  1000, 1050, 1100, 1150, 1200, 1250, 1300, 1350,
        1400, 1450, 1500, 1550, 1600, 1650, 1700, 1750, 1800, 1850,
        1900, 1950, 2000};

    //====================================================================
    // [1] Create bessel function splines needed for the LOS integration
    //====================================================================

    // Splines of bessel-functions for each value of ell in the array above
    std::vector<Spline> j_ell_splines;

    // Generate splines of bessel-functions for each ell needed
    // to do the LOS integration
    void generate_bessel_function_splines();

    //====================================================================
    // [2] Do the line of sight integration and spline the result
    //====================================================================

    // Do LOS integration for all ells and all k's in the given k_array
    // and for all the source functions (temperature, polarization, ...)
    void line_of_sight_integration(Vector & k_array);

    // Do the line of sight integration for a single quantity
    // for all ells by providing a source_function(x,k) (can be temp, pol, ...)
    Vector2D line_of_sight_integration_single(
        Vector & k_array,
        std::function<double(double,double)> &source_function);

    // Splines of the reusult of the LOS integration
    // Theta_ell(k) and ThetaE_ell(k) for polarization
    std::vector<Spline> thetaT_ell_of_k_spline;
    std::vector<Spline> thetaE_ell_of_k_spline;
```

Spline of Bessel Functions for each ell value in ells

Implement the matter P(k) here

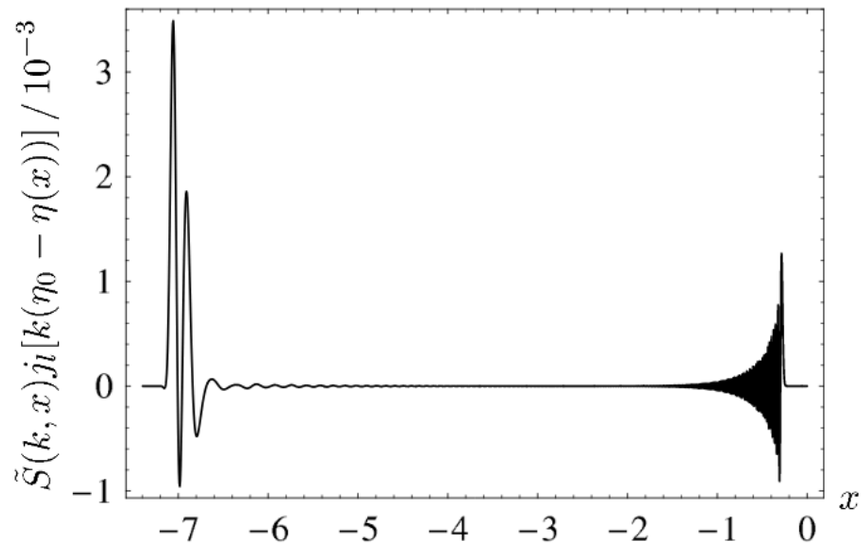
```
    //====================================================================
    // [2] Do the line of sight integration and spline the result
    //====================================================================

    // Do LOS integration for all ells and all k's in the given k_array
    // and for all the source functions (temperature, polarization, ...)
    void line_of_sight_integration(Vector & k_array);

    // Do the line of sight integration for a single quantity
    // for all ells by providing a source_function(x,k) (can be temp, pol, ...)
    Vector2D line_of_sight_integration_single(
        Vector & k_array,
        std::function<double(double,double)> &source_function);

    // Splines of the reusult of the LOS integration
    // Theta_ell(k) and ThetaE_ell(k) for polarization
    std::vector<Spline> thetaT_ell_of_k_spline;
    std::vector<Spline> thetaE_ell_of_k_spline;

    //====================================================================
    // [3] Integrate to get power-spectrum
    //====================================================================

    // General method to solve for Cells (allowing for cross-correlations)
    // For auto spectrum (C_TT) then call with f_ell = g_ell = theta_ell
    // For polarization C_TE call with f_ell = theta_ell and g_ell = thetaE_ell
    Vector solve_for_cell(
        Vector & logk_array,
        std::vector<Spline> & f_ell,
        std::vector<Spline> & g_ell);

    // Splines with the power-spectra
    Spline cell_TT_spline{"cell_TT_spline"};
    Spline cell_TE_spline{"cell_TE_spline"};
    Spline cell_EE_spline{"cell_EE_spline"};

  public:

    // Constructors
    PowerSpectrum() = delete;
    PowerSpectrum(
        BackgroundCosmology *cosmo,
        RecombinationHistory *rec,
        Perturbations *pert);

    // Do all the solving: bessel functions, LOS integration and then compute Cells
    void solve();

    // The dimensionless primordial power-spectrum Delta = 2pi^2/k^3 P(k)
    double primordial_power_spectrum(const double k) const;

    // Get P(k,x) for a given x in units of (Mpc)^3
    double get_matter_power_spectrum(const double x, const double k_mpc) const;

    // Get the quantities we have computed
    double get_cell_TT(const double ell) const;
    double get_cell_TE(const double ell) const;
    double get_cell_EE(const double ell) const;

    // Output Cells in units of l(l+1)/2pi (muK)^2
    void output(std::string filename) const;
};
```

Step I

- The first thing you should do is to go back to the perturbation class and make a 2D spline of the source function

$$\tilde{S}(k,x) = \tilde{g}\left[\Theta_0 + \Psi + \frac{1}{4}\Pi\right] + e^{-\tau}\left[\Psi' - \Phi'\right] - \frac{1}{ck}\frac{d}{dx}(\mathcal{H}\tilde{g}v_b) + \frac{3}{4c^2k^2}\frac{d}{dx}\left[\mathcal{H}\frac{d}{dx}(\mathcal{H}\tilde{g}\Pi)\right].$$

- See the website (will add some plots) or Callin for plots on how this function should look like so you can test that its working



Step II

- We need to do *many* lookups of spherical Bessel functions to compute the line of sight integrals. Best to create a spline of it (well you can try without, but its likely very very slow)

- Compute and plot these functions for different values of ell so that you understand how the look like and how you need to sample it. In the code-template you can use the function

```
double j_ell(const int n, const double x)
```

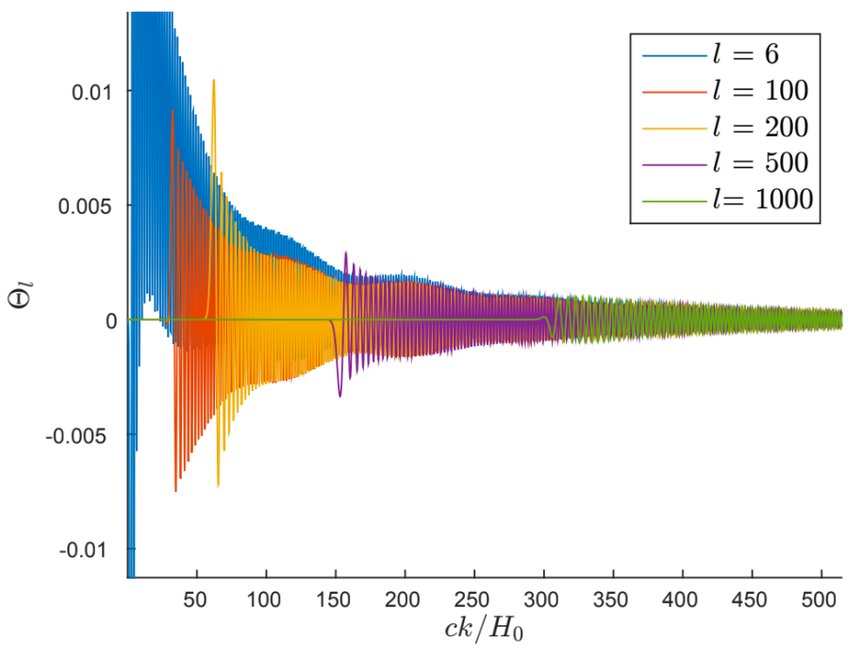
(n is ell)

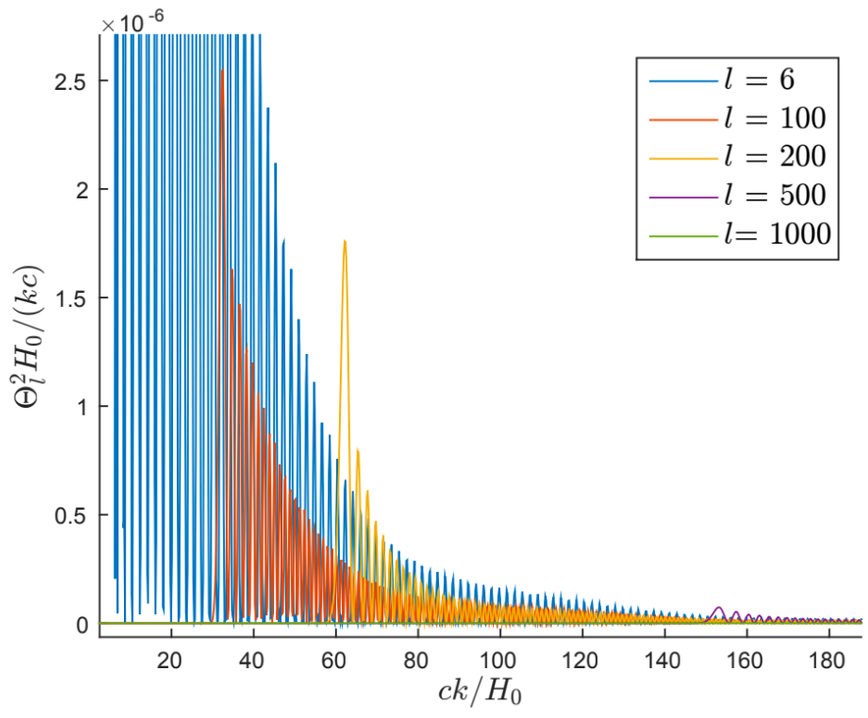
- (For large ell the function is practically 0 for x < 0.2*ell for ell =10-100 and x < 0.7*ell for ell>100 so this sets the lower range you need to compute it at. It has a 2*PI period of oscillation -> what sampling rate you need. It decays as 1/x and the maximum x you need is like 3000-5000. This sets the upper limit.)

Step III

- For each ell do the line of sight integral for several values of k and make a spline

$$\Theta_\ell(k, x=0) = \int_{-\infty}^{0} \tilde{S}(k,x) j_\ell[k(\eta_0 - \eta)] dx,$$



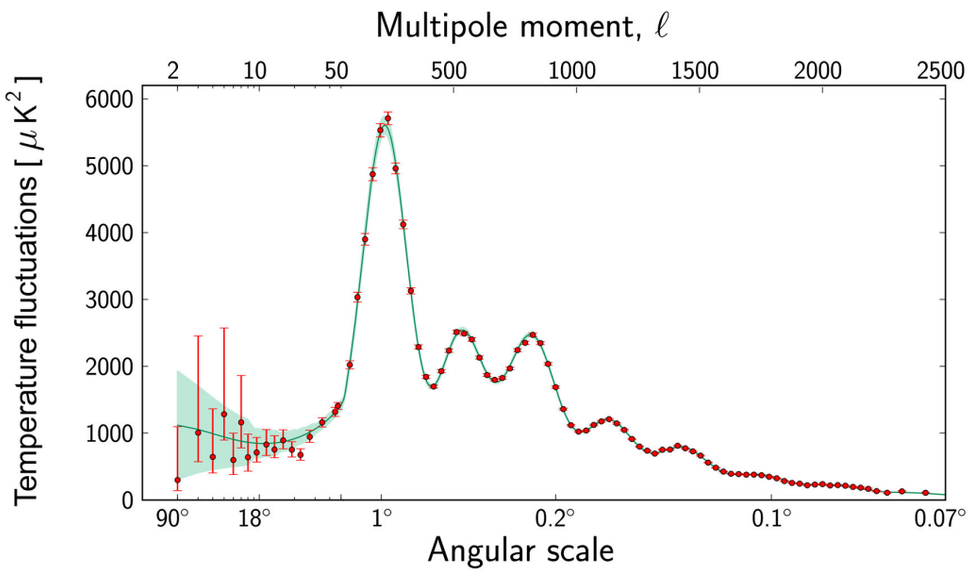


Step IV

- Use the spline for the photon multipole to compute the integral to get the Cell

$$C_\ell = 4\pi \int_0^\infty A_s \left(\frac{k}{k_{\rm pivot}} \right)^{n_s - 1} \Theta_\ell^2(k) \frac{dk}{k}.$$

- Repeat the process for all ell's and make a spline of it to get



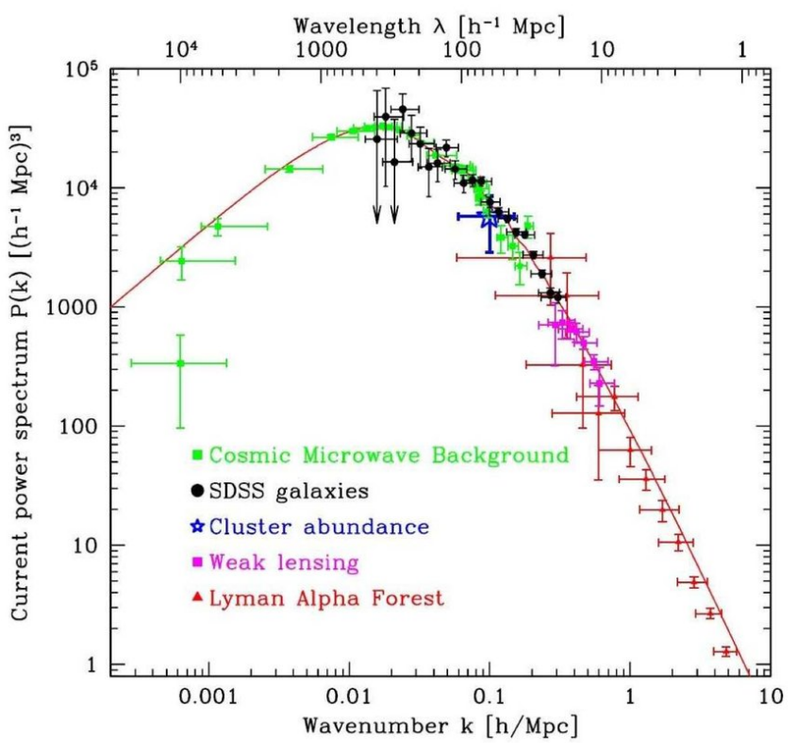
Step V

- The (total) matter power-spectrum is easy! The co-moving matter density perturbation can be computed from the splines you have made of Phi

$$P(k,x) = |\Delta_M(k,x)|^2 P_{\text{primordial}}(k)$$

$$\Delta_M(k,x) \equiv \frac{c^2 k^2 \Phi(k,x)}{\frac{3}{2}\Omega_M a^{-1} H_0^2}.$$

$$\frac{k^3}{2\pi^2} P_{\text{primordial}}(k) = A_s \left(\frac{k}{k_{\text{pivot}}} \right)^{n_s - 1}$$



- (For individual components:)

$$\Delta_i = \delta_i - \frac{3\mathscr{H}}{ck} v_i$$

Good luck

- You are free to figure out for yourself how to do each of the different steps, i.e. how you do the integrations.

- See Callin for some tips. The website also has some tips on how to do this.

- Also for this milestone I recommend you **start early!** It is going to to take a lot of time to get correct so you don’t want to have to deal with this the last week.