Assignment - 2

Dhananjay Chimnani (218070333)

Contents

1 Verification of the Parseval's Theorem

Parseval's theorem in the Discrete Fourier Transform (DFT) states that the total energy of a discrete signal in the time domain is equal to the total energy in the frequency domain. Mathematically, it is expressed as:

$$\sum_{n=0}^{N-1} |x_n|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X_k|^2, \tag{1}$$

where x_n is the time-domain sequence, and X_k is its DFT.

In this problem, we had to verify this theorem for the $y = 0$ data of the u component of the velocity.

On checking, we get the LHS = 428.77845735032963 and the RHS = 428.7784573503308, which are very close. Thus, Parseval's theorem is verified. This result represents the conservation of energy when going from the physical space to the Fourier space.

2 One - Dimensional Energy Spectrum

The energy spectrum $E(k)$ is given by:

$$E(k) = \frac{|\hat{u}|^2 + |\hat{v}|^2}{2}, \tag{2}$$

where $\hat{u}$ is the Fourier Transform of the u velocity field, and k is the wavenumber.

We calculated the energy spectrum along lines in the x-direction and averaged over the y direction. The resulting averaged quantity was plotted against k (the wavenumber).

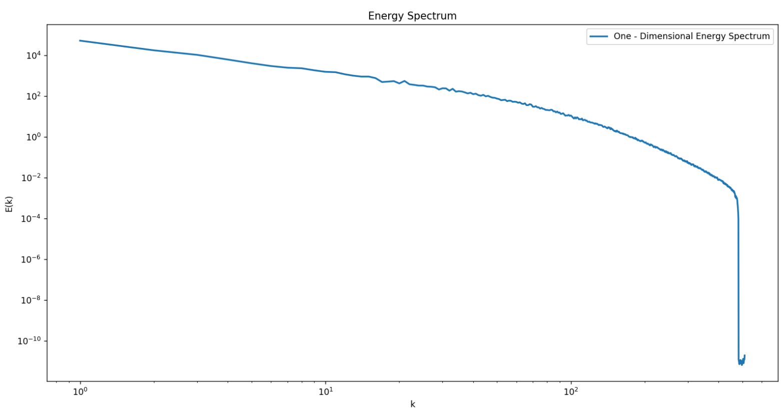


Figure 1: Plot of the averaged energy spectrum against wavenumber k.

After plotting this, we check the slope of the plot in the inertial range, which is defined as $x = [60\eta, l/6]$, where η is the Kolmogorov length scale and l is the integral length scale. To do this, we calculate the corresponding wavenumbers:

$$k_{\text{max}} = \frac{2\pi}{60\eta}, \quad k_{\text{min}} = \frac{2\pi}{l/6}. \tag{3}$$

This defines the range within which we need to check the slope of the above plot.

We perform a linear curve fitting within this range to estimate the slope of the plot. The obtained slope is compared with the theoretical value of -1.67.

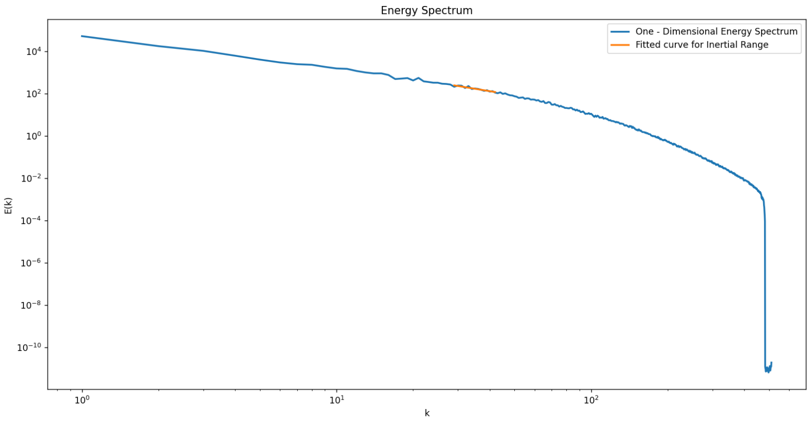


Figure 2: Plot showing the slope of the energy spectrum in the inertial range.

The slope in the inertial range comes out to be -1.95, which is near the theoretical value of -1.67.

3 Two Dimensional Energy Spectrum

In this problem, we compute and plot the two-dimensional energy spectrum using the full 2D velocity fields (u, v) to obtain $E(k)$, where $k = k_x\hat{i} + k_y\hat{j}$. This results in a spectral field in wavevector space.

3.1 2D Energy Spectrum Plot

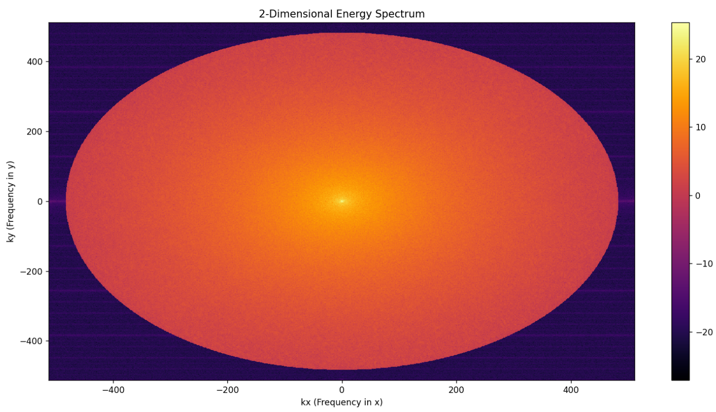


Figure 3: Plot of the two-dimensional energy spectrum.

3.2 Spherically Averaged Energy Spectrum

We perform a shell-averaging over wavenumber shells $k - \frac{1}{2} \leq k < k + \frac{1}{2}$ for $k \in [k_{\min}, k_{\max}]$, to obtain the spherically averaged energy spectrum:

$$E(k) = \sum_{k-1/2 \leq k < k+1/2} E(k), \tag{4}$$

over the scalar wavenumber:

$$k = \sqrt{k_x^2 + k_y^2}. \tag{5}$$

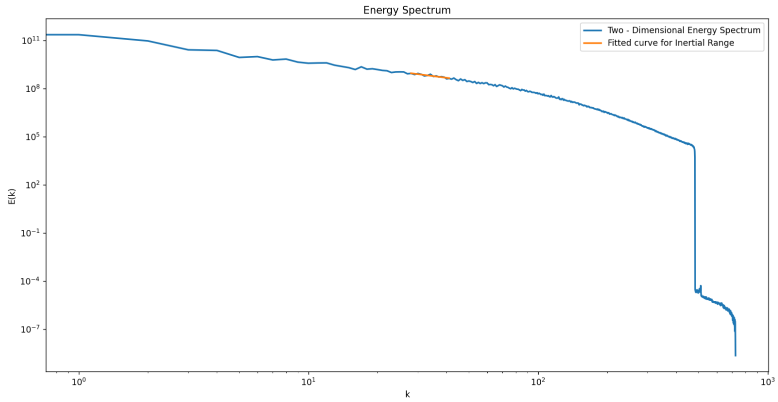


Figure 4: Plot of the spherically averaged energy spectrum.

Again we check the slope of this curve in the inertial range using the same range of k as in the previous problem. We get the slope as -1.82 which is again near the theoretical value of -1.67.

4 Correlation functions of the velocities

Plotting the longitudinal and transverse velocity correlation functions from the velocity fields. The **longitudinal correlation function** is given by:

$$R_L(r) = \frac{\langle u'(\mathbf{x}) u'(\mathbf{x} + r\hat{\mathbf{e}}_r) \rangle}{(u'_{\text{rms}})^2} \tag{6}$$

$$R_T(r) = \frac{\langle u'(\mathbf{x}) v'(\mathbf{x} + r\hat{\mathbf{e}}_r) \rangle}{(v'_{\text{rms}})^2} \tag{7}$$

where:

- u' and v' are the velocities or velocity fluctuations in the respective directions.
- $\mathbf{x}$ is the position vector.
- r is the separation distance.
- $\hat{\mathbf{e}}_r$ is the unit vector along the separation direction.
- $\langle \cdot \rangle$ denotes an ensemble average.

I have calculated these functions for u, v and their fluctuations (u' and v') in the x and y directions. Also, to calculate this, I used the fact that our domain is periodic in both x and y. This allows us to average our function on all the points in the domain for any particular r value.

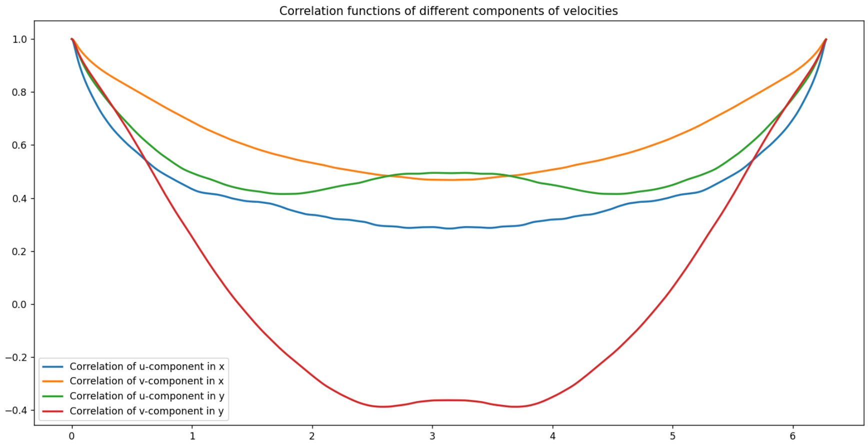


Figure 5: Correlation functions of the velocity components

The above plot contains the correlations of the velocity components and not the fluctuations.

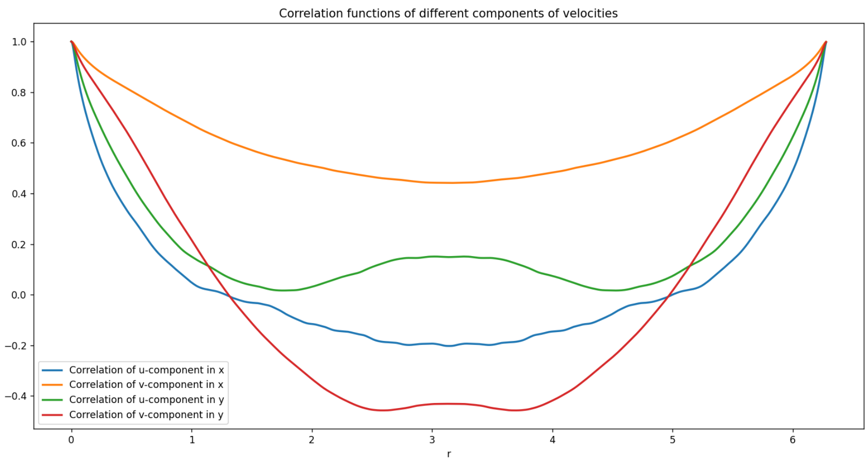


Figure 6: Correlation functions of the velocity fluctuations

5 Structure Function Calculation

Longitudinal structure functions are defined as:

$$S_p(r) = \langle \Delta u^p(r) \rangle = \left\langle \left| (\mathbf{u}(\mathbf{x}+\mathbf{r}) - \mathbf{u}(\mathbf{x})) \cdot \frac{\mathbf{r}}{|\mathbf{r}|} \right|^p \right\rangle \propto r^{\zeta_p} \tag{8}$$

For the calculation i used 26 equidistant values of r in the given range of r and took 12 directions each at an angle of pi/6 to each other for each of these r values. Then I averaged my data over all the data points as my data is periodic for every value of r.

Also I have made my code in such a way that I can change this angle of pi/6 and also the value of 26. I used these values to keep the computation less complex. To get the data for all the 7 structure functions for the above theta and r values took about 4 hours to run.

5.1 Plotting the seven longitudinal structure functions vs r

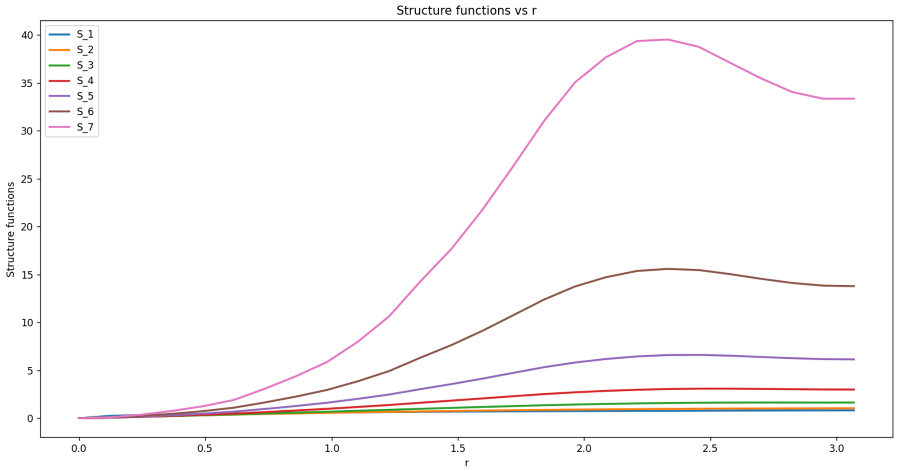


Figure 7: Plot of all the seven longitudinal structure functions.

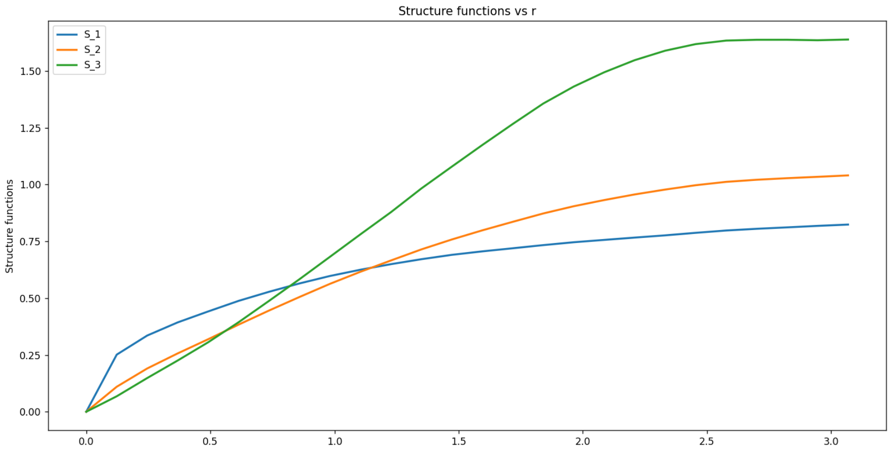


Figure 8: Plots of S_1, S_2, S_3.

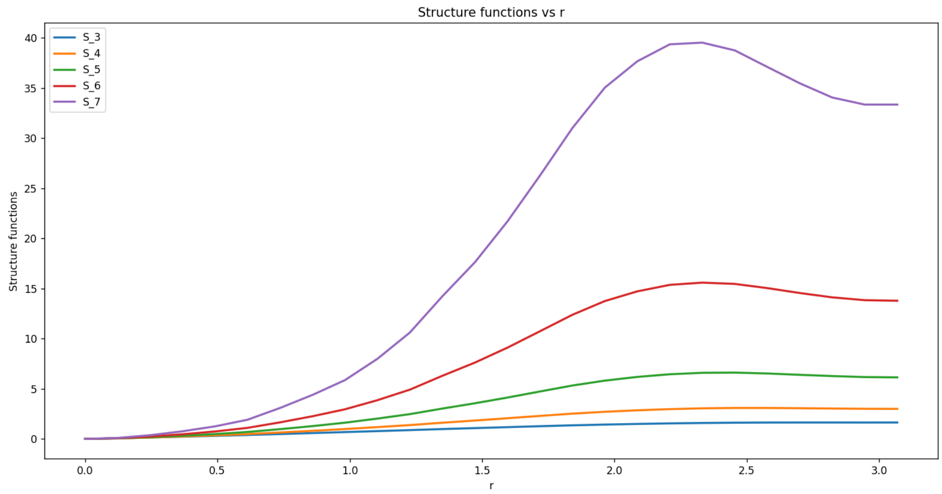


Figure 9: Plots of S_3, S_4, S_5, S_6, S_7.

Figures 7 and 8 have been added as figure 6 is not quite clear.

5.2 Verifying Kolmogorov's $\frac{4}{5}$th Law for $S_3(r)$

In thus problem, I noticed that as I increased the directions over which I am averaging the data, the plot becomes more and more straight. For this I will attach some plots of $S_3(r)$ vs r.

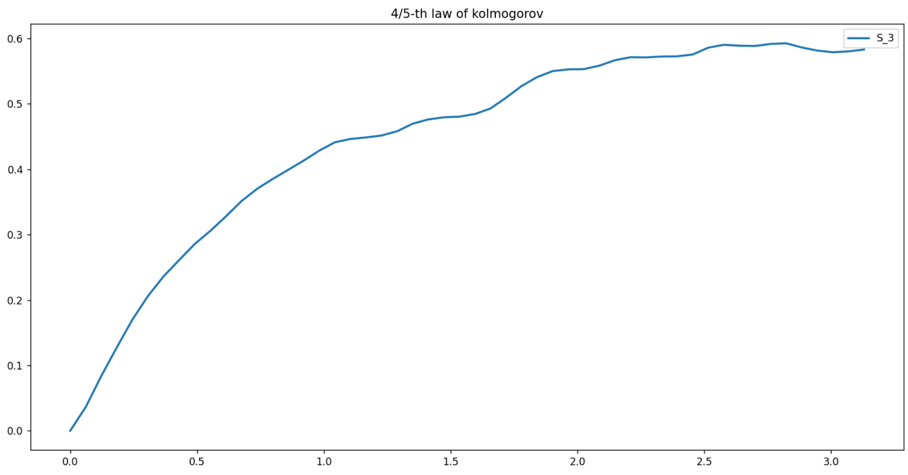


Figure 10: Averaging done over two directions at an angle of pi.

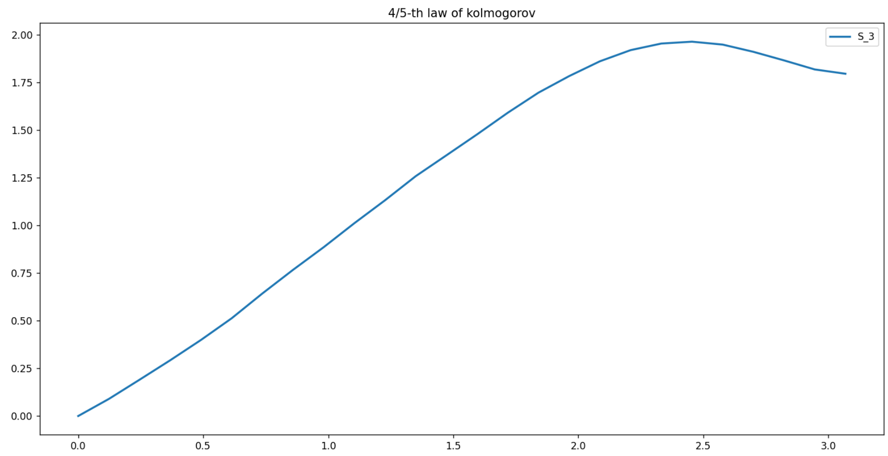


Figure 11: Averaging done over four directions at an angle of pi/2.

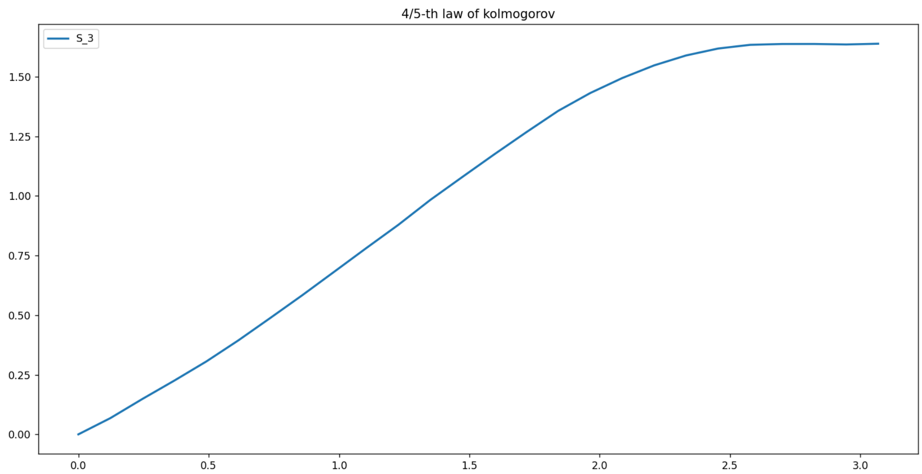


Figure 12: Averaging done over twelve directions at an angle of pi/6.

So calculating the slope of the plot using curve fitting gives:

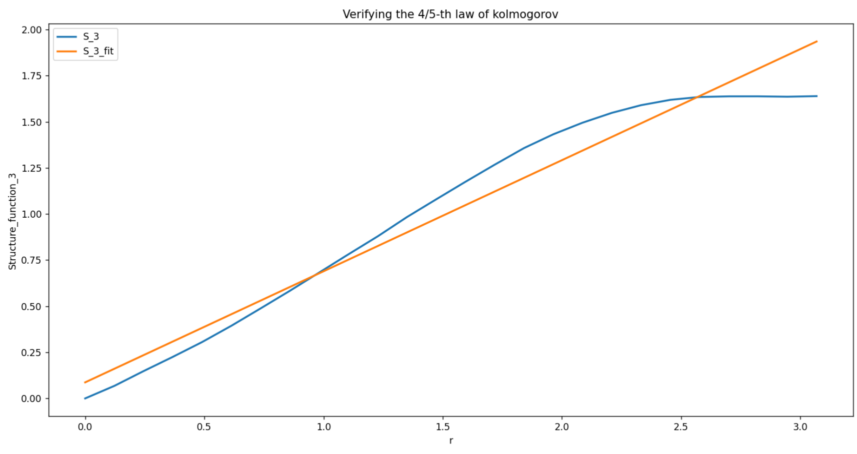


Figure 13: Averaging done over twelve directions at an angle of pi/6 with linear curve fitting.

Slope from fitting: 0.6023750563342332

Now we will calculate the slope as suggested by the 4/5th law:

$$\epsilon = \frac{(u_{\rm rms}^2 + v_{\rm rms}^2)^{1.5}}{l}, \quad \text{slope} = 0.8\,\epsilon \tag{9}$$

This formula has been used as I have used both u and v while calculating the longitudinal structure functions.Using the expression, we get :

Calculated Slope: 0.5873891805023274

As can be seen that the two slopes are quite near. Thus, the Kolmogorov's 4/5th law is verified.

5.3 ESS profiles of the Longitudinal Structure functions

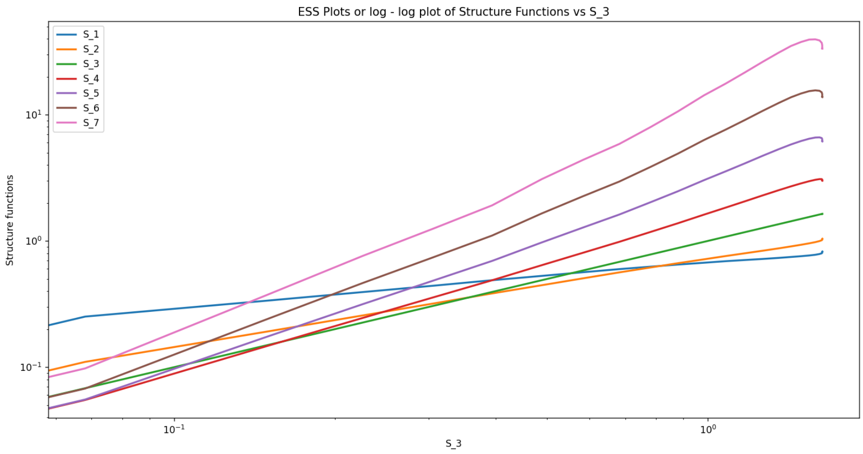


Figure 14: ESS profiles for all the Longitudinal Structure functions in a single plot.

5.4 Deviation from the Kolmogorov's Prediction

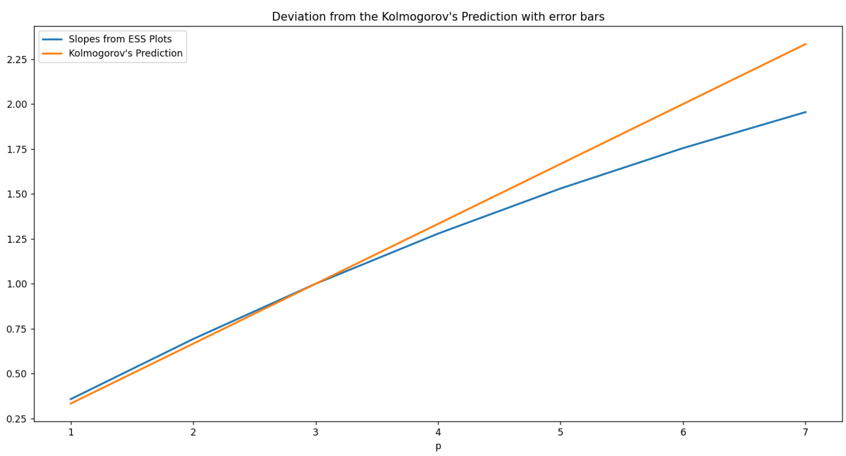


Figure 15: Deviation from the Kolmogorov's Prediction.

The above curve matches quite well with the curve we were shown in the lecture.

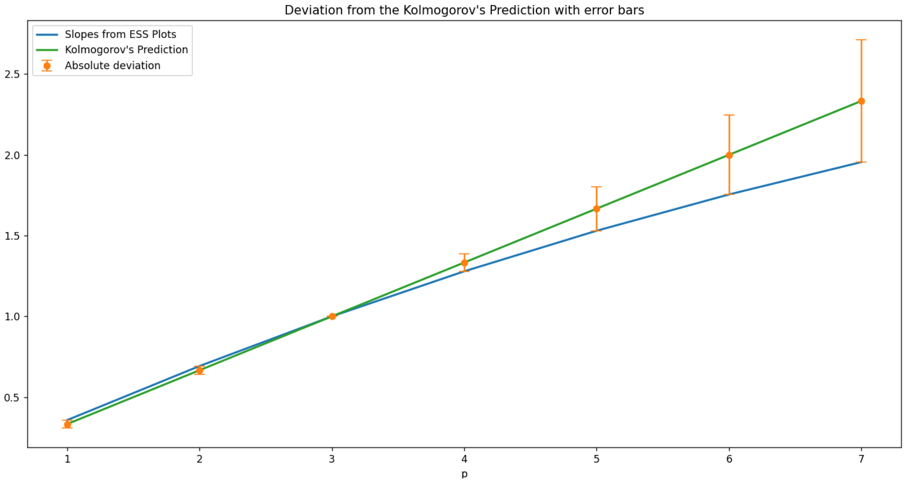
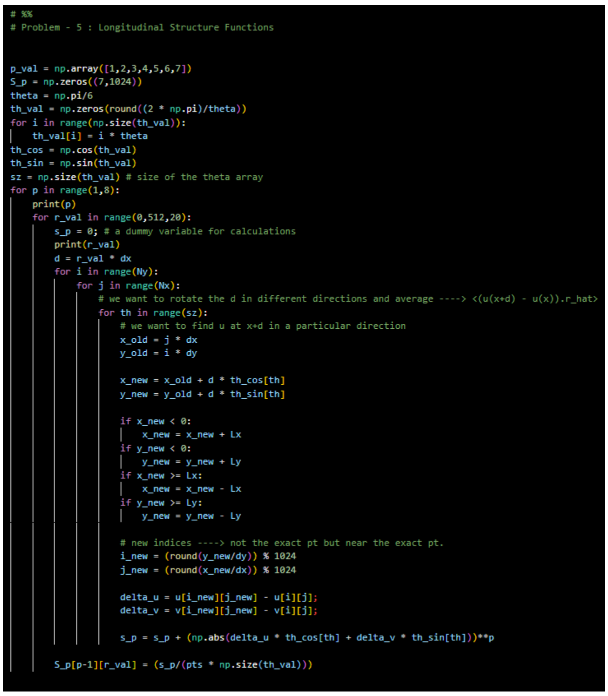


Figure 16: Deviation from the Kolmogorov's Prediction shown using error bars.

I don't really have a reason for why the deviation is seen at larger values of p and not at the smaller ones.

6 Important Notes

For the fifth problem, my code took a long time to run. So, to save my time, I have saved the data in the file `"matrix.npy"`, which I will put in my zip file so that my results can be easily verified. My code still contains the section through which I calculated my data. I am providing an image of this cell. You can save the file `"matrix.npy"` and directly run the next cell, ignoring this cell to save time.



```python
# %%
# Problem - 5 : Longitudinal Structure Functions


p_val = np.array([1,2,3,4,5,6,7])
S_p = np.zeros((7,1024))
theta = np.pi/6
th_val = np.zeros(round((2 * np.pi)/theta))
for i in range(np.size(th_val)):
    th_val[i] = i * theta
th_cos = np.cos(th_val)
th_sin = np.sin(th_val)
sz = np.size(th_val) # size of the theta array
for p in range(1,8):
    print(p)
    for r_val in range(0,512,20):
        s_p = 0; # a dummy variable for calculations
        print(r_val)
        d = r_val * dx
        for i in range(Ny):
            for j in range(Nx):
                # we want to rotate the d in different directions and average ----> <(u(x+d) - u(x)).r_hat>
                for th in range(sz):
                    # we want to find u at x+d in a particular direction
                    x_old = j * dx
                    y_old = i * dy

                    x_new = x_old + d * th_cos[th]
                    y_new = y_old + d * th_sin[th]

                    if x_new < 0:
                        x_new = x_new + Lx
                    if y_new < 0:
                        y_new = y_new + Ly
                    if x_new >= Lx:
                        x_new = x_new - Lx
                    if y_new >= Ly:
                        y_new = y_new - Ly

                    # new indices ----> not the exact pt but near the exact pt.
                    i_new = (round(y_new/dy)) % 1024
                    j_new = (round(x_new/dx)) % 1024

                    delta_u = u[i_new][j_new] - u[i][j];
                    delta_v = v[i_new][j_new] - v[i][j];

                    s_p = s_p + (np.abs(delta_u * th_cos[th] + delta_v * th_sin[th]))**p

        S_p[p-1][r_val] = (s_p/(pts * np.size(th_val)))
```

Figure 17: This code section can take around 4 hours to run.

I have taken help from CHATGPT for making this latex file as I had no expirience of writing code in LATEX.