UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

Master's In Space And Aeronautical Engineering

Computational Engineering Assignment 4

Burgers Equation in Fourier Space

November, 2023

Author: Iñaki Fernandez

Index

If it is in the reader's interest, all the code done for this report can be found in the following repository: **Code**

1 Introduction

In this work we will solve the Burgers equation in Fourier Space. Burgers equation is a simplification of the Navier-Stokes (N-S) equations, by simplifying them in one dimension, specifically the momentum equation. Therefore, we shall start the discussion by stating the N-S equations, in this case, for an incomprenssible fluid flow the dimensionless N-S equations can be expressed as:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\nabla)\mathbf{u} = \frac{1}{Re}\nabla^2\mathbf{u} - \nabla p \tag{1}$$

$$\nabla \mathbf{u} = 0 \tag{2}$$

where $\mathbf{u}$ is the velocity, Re is the Reynolds number and p is the pressure. Equation (1) is the momentum equation, which is directly obtained from Newton's second law. Equation (2) is the mass conservation equation. The Reynolds number can be expressed as:

$$Re = \frac{V_0 L}{\nu} \tag{3}$$

where V_0 is the free stream velocity, L the characteristic length and ν the kinematic viscosity. The Burgers equation can be obtained by simply taking the N-S momentum equation in one dimension:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \frac{1}{Re}\frac{\partial^2 u}{\partial x^2} + f \tag{4}$$

Here we can notice that the mass conservation gives us that $\partial u/\partial x = 0 \rightarrow u = cte$.

The analysis of the Burgers equation will be done in the Fourier space. This implies that we have to write the $u(x)$ velocity into one dimensional k space. The expansion into k space can be done as it follows:

$$u(x) = \sum_{k=-N}^{k=+N} \hat{u}_k e^{ikx} \tag{5}$$

It can be noticed that the expansion is not done in the hole $k \in [-\infty, +\infty]$, instead, we truncate it at a fixed $2N$ modes. In addition to this, since $u(x,t) \in \mathbb{R}$, we will have to impose:

$$\hat{u}_k = \overline{\hat{u}_{-k}} \tag{6}$$

which implies that if $k < 0$, the value of the velocity in that k will be the complex conjugate of the velocity for the same k but with a positive value. So we will only have to solve $k > 0$ values.

Once we know how to write the velocity into k space, we can write Eq. (4) in k space. The Fourier transformation of the Burgers equation can be read as it follows:

$$\frac{\partial \hat{u}_k}{\partial t} + \sum_{p+q=k} \hat{u}_p i q \hat{u}_q = -\frac{k^2}{Re}\hat{u}_k + \hat{F}_k \tag{7}$$

The second-term is the non-linear part of the Burgers equation, which indeed is the convective term. Basically, the algorithm presented in this work will try to solve the above equation. Since $\hat{u}_k$ can be a complex number, the analysis of the resulys will be done in terms of the energy E_k:

$$E_k = \hat{u}_k \overline{\hat{u}_k} \in \mathbb{R} \tag{8}$$

Direct Numerical Simulations (DNS) of the Burgers equation gives non accurate results for small values of N. In order to obtain better results for this range, we will also study Large-Eddy Simulation (LES). In this case, we will use the proposal of Métais and Lesieu [1], which implies changing the value of ν (remember its relation with the Reynolds number, Eq. (3)):

$$\nu_t\left(\frac{k}{k_N}\right) = \nu_t^{+\infty}\left(\frac{E_{k_N}}{k_N}\right)^{\frac{1}{2}} \nu_t^*\left(\frac{k}{k_N}\right) \tag{9}$$

where

$$\nu_t^{+\infty} = 0.31\frac{5-m}{m+1}\sqrt{3-m}C_K^{\frac{-3}{2}} \tag{10}$$

$$\nu_t^*\left(\frac{k}{k_N}\right) = 1 + 34.5e^{-3.03(k_N/k)} \tag{11}$$

here m is the slope of the energy spectrum, being in our case $m = 2$, E_{k_N} is the energy at the cutoff frequency k_N and C_K is the Kolmogorov constant. So the final kinematic viscosity:

$$\nu_{eff}(k) = \nu + \nu_t(k) \tag{12}$$

2 Problem Specification

Before starting solving Eq. (7), we need to comment some specification around the problem. As it has been mentioned, Eq. (5) have been truncated into $2N$ modes, so that the k space will be divided into small pieces from $-N$ to N as it is depicted in Fig. 1.

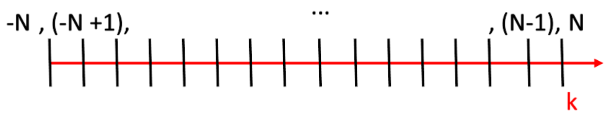

Figure 1: *k space scheme.*

The time evolution of the velocity will be solved using an explicit scheme so that:

$$\frac{\hat{u}_k^{n+1} - \hat{u}_k^n}{\Delta t} \quad ; \quad \Delta t < C_1 \frac{Re}{N^2} \tag{13}$$

where $0 < C_1 < 1$. Using the above expression Eq. (7) can be discretized in time as it follows:

$$\hat{u}_k^{n+1} = \hat{u}_k^n + \Delta t \left[-\frac{k^2}{Re} \hat{u}_k^n + C_{onv}(q,p)^n \right] \tag{14}$$

where $C_{onv}(q,p) \equiv \sum_{p+q=k} \hat{u}_p i q \hat{u}_q$ is the convective term. We will impose an initial state of $\hat{u}_k^0 = 1/k$. We will assume that there is no interaction of $k = 0$ mode so that $\hat{u}_0^n = 0$. In addition to this, we will also impose that $\hat{u}_1^n = 1$.

Finally we will make a brief discussion around the convective term, $C_{onv}(p,q)$. Even though p and q can have values from $-N$ to N, we will only compute the convective terms for those values of p and q lying inside $k = 0$ and $k = N$ lines, in the p vs q diagram, as it is depicted in Fig. 2.

This implies that we will have to compute carefully the convective term.

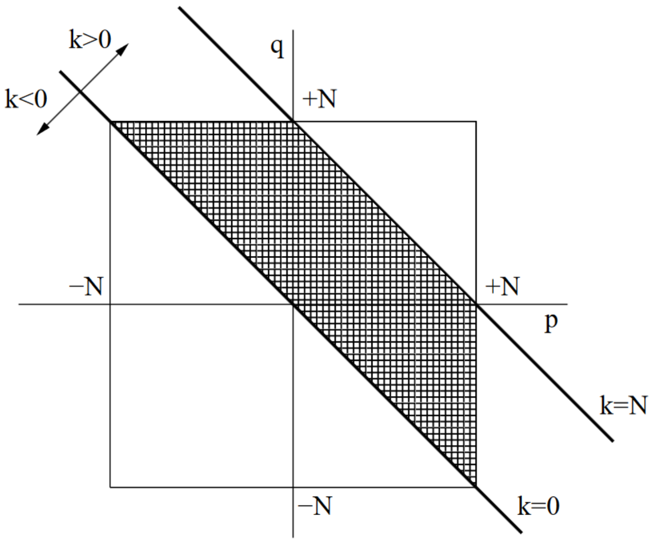

Figure 2: *The values of p and q that will be computed are shown with crossed black solid lines [2].*

3 Code Structure

In this section we introduce the code structure. As it will be shown in Section **7**, the code is written for Python3 language. Further discussion on the code can be found in the repository linked below the **Index**.

1. Establish the input data, which is segregated into two distinct sections: one dedicated to the physical aspects of the problem encompassing parameters like the Reynolds number Re, the kinematic viscosity ν, etc. The other is dedicated to numerical aspects as the modes quantity N, convergence criteria ε, etc.

2. Define velocity vector $\hat{u}_k^n$ and $\hat{u}_k^0$ and the energy spectrum E_k. The first two must be defined as complex vectors.

3. Initialize the velocity vectors with the parameters defined in Section **2**.

4. Compute a new time step: $t^{n+1} = t^n + \Delta t$.

5. Compute the convective term $C_{onv}(q, p) \equiv \sum_{p+q=k} \hat{u}_p iq\hat{u}_q$.

6. Compute the updated velocity $\hat{u}_k^{n+1}$ using Eq. (14).

7. Check steady state: if $|\hat{u}_k^{n+1} - \hat{u}_k^n| < \varepsilon$ is fulfiled, then the steady state has been obtained and you can go to step 8. If not, $\hat{u}_k^n = \hat{u}_k^{n+1}$ and go back to step 4.

8. Compute the energy spectrum E_k using Eq. (8).

9. Final calculations and print results.

4 Validation of the Code

Before presenting the results obtained in this work, it is crucial to conduct an analysis to ensure the code's proper functionality. The results will be compared with those presented in figure 2 and 3 of the report of the CTTC [2].

First, we performed a DNS calculation for $Re = 40$ and $C_1 = 0.02$ using two distinct mode quantities: $N = 20$ and $N = 100$. The resulting energy spectrum is presented in Fig. 3. It is evident that the solution for $N = 20$ is under-resolved, whereas the solution for $N = 100$ is fully resolved. Same behaviour is also observed in the second figure of the CTTC report. It is interesting to comment how the solution for $N = 100$ at the last mode shows a small peak. This is a numerical problem, where there is a energy transfer to the modes beyond N. This behaviour is also seen in the CTTC report. Lastly, it is notable that for high k values in DNS $N = 100$, we observe energy dissipation, leading to a deviation from the reference slope of $m = -2$. This effect is induced by the diffusive term in Eq. (14), which introduces damping in the energy spectrum.

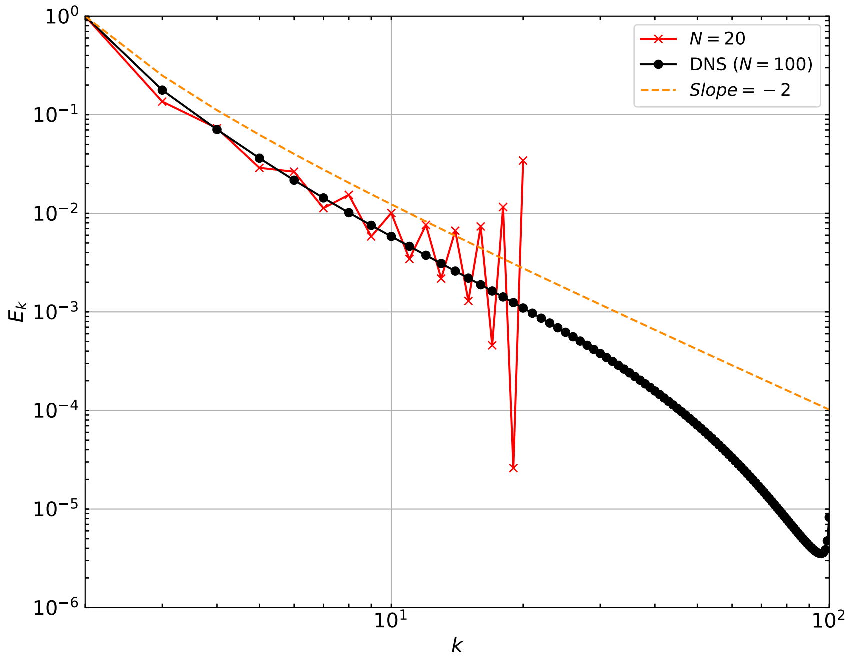

Figure 3: *DNS solution of the energy spectrum for $N = 20$ (red line) and $N = 100$ (black line). In dashed orange line we show m=-2 slope. The results are shown in logarithmic scale.*

Upon comparing our plot with the reference figure, we note that our data for $N = 20$ exhibits larger peaks around the result for $N = 100$ than those reported in the CTTC document. This discrepancy may be attributed to the specific choice of the C_1 value. The number and amplitude of peaks in the $N = 20$ case vary with different values of C_1.

Next, we proceeded with LES calculations for $N = 20$, $Re = 40$ and $C_1 = 0.02$ while employing two different Kolmogorov constants, $C_k = 0.4523$ and $C_k = 0.05$. The resulting energy spectrum is depicted in Fig. 5, with the energy spectra overlaying the results presented in Fig. 3. It is evident that LES improves the solution for a small number of modes, a trend also observed in the reference result from the CTTC (figure 3).

Despite obtaining similar results, the reference values exhibit smaller errors compared to the DNS $N = 100$ result, which could once again be influenced by the choice of the C_1 value. It is also interesting to note how the LES solution varies with the Kolmogorov constant value. For smaller values of C_k, we observe the results deviating from the DNS $N = 100$ line at smaller E_k values, around $N = 8$. Conversely, with larger C_k values, the deviation occurs at higher E_k values. In this scenario, we obtain three smaller peaks (compared to those of DNS $N = 20$), which is also consistent with the CTTC reference solution.

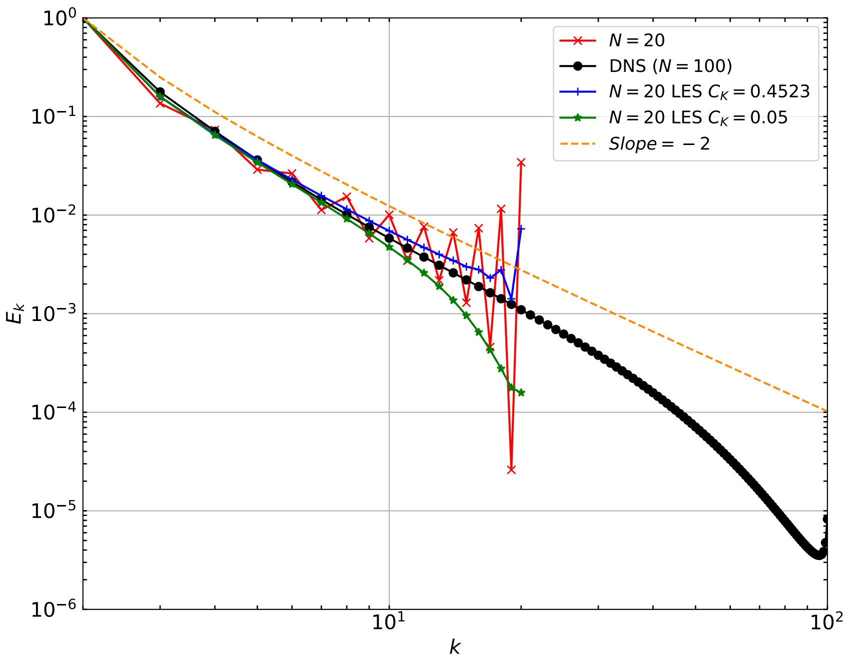

Figure 4: *DNS and LES solution of the energy spectrum. For DNS: $N = 20$(red line) and $N = 100$ (black line). For LES with $N = 20$: $C_k = 0.4523$ (blue line) and $C_k = 0.5$ (green line). In dashed orange line we show m=-2 slope. The results are shown in logarithmic scale.*

5 Results

In this section we present the results for both DNS and LES results. The discussion will be around the Reynolds number and the Kolmogorov constant.

The Reynolds number is associated with the diffusive term in the kinetic energy transport equation, as described in [2]: $(-2k^2/Re)E_k$. Altering its value leads to variations in energy spectrum damping. This phenomenon is visually depicted in Figure 6, where we have presented the DNS results for $N = 100$ with four distinct Reynolds numbers, $Re = [20, 40, 60, 80]$.

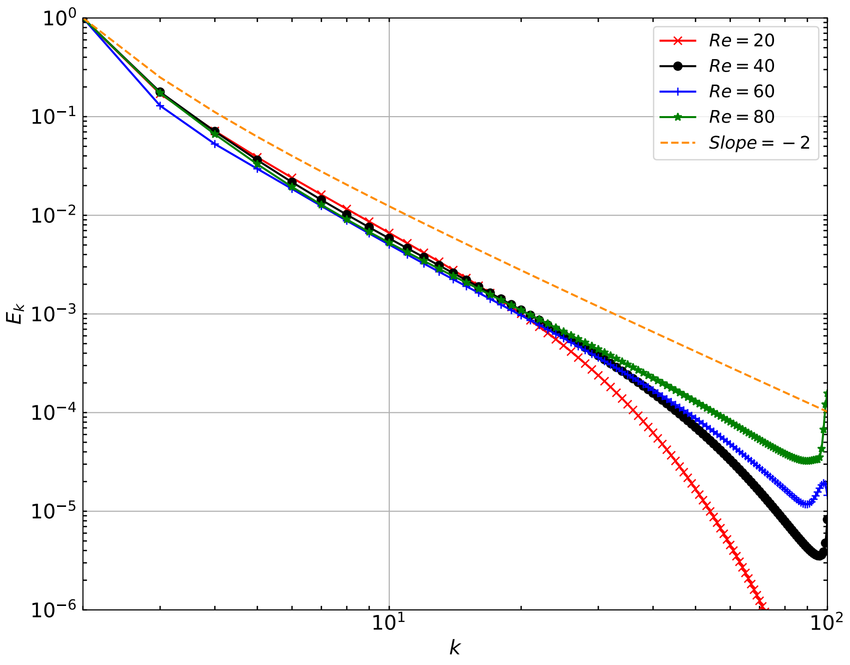

Figure 5: *DNS solution of the energy spectrum for $N = 100$ for different Reynolds numbers. The results are shown in logarithmic scale.*

For high k modes, the energy dissipation exhibits significant variations across different Reynolds numbers, in line with expectations. Notably, the highest Reynolds number corresponds to the lowest dissipation, while the lowest Reynolds number results in the highest energy dissipation. Additionally, the highest Reynolds number outcome aligns more closely with the reference slope of $m = -2$. In conclusion, it becomes evident that changes in the Reynolds number directly impact the diffusive term of the Burgers equation and consequently affect the dissipative component of the energy spectrum.

In the context of LES, we have adopted a similar approach, but with a reduced modes of $N = 20$ and a Kolmogorov constant of $C_k = 0.4523$. This allows us to investigate the Reynolds number's influence on small-scale solutions. The results are depicted in Figure 6, where we consider Reynolds numbers $Re = [20, 40, 60, 80]$.

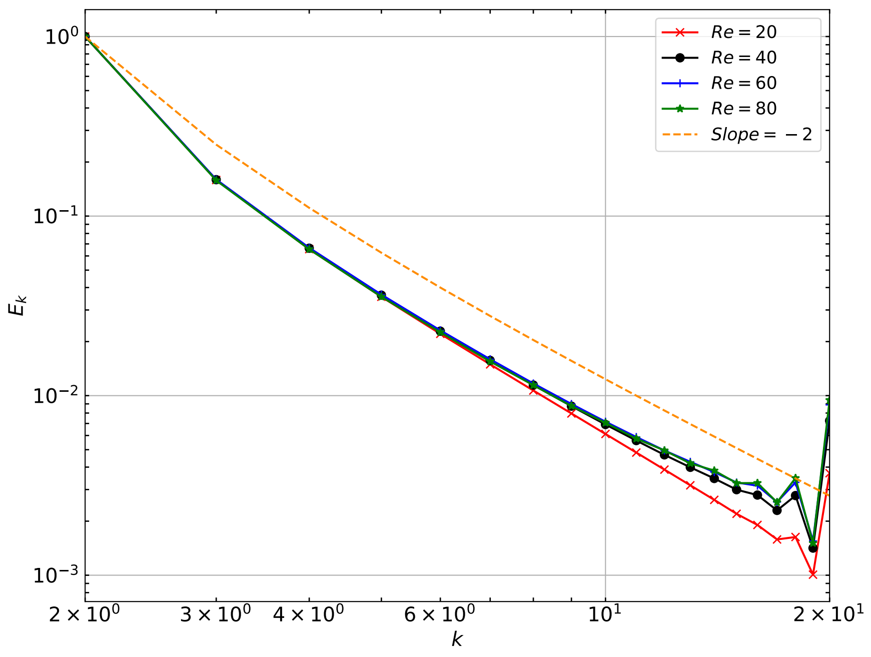

Figure 6: *LES solution of the energy spectrum for $N = 20$ and $C_k = 0.4523$ for different Reynolds numbers. The results are shown in logarithmic scale.*

In the lower k modes, all four results exhibit consistent behavior, maintaining a shared trend. Around $N = 7$, a noticeable divergence in the trends becomes apparent. In higher wavenumber modes, solutions with larger Reynolds numbers tend to yield higher energy spectrum values, while lower Reynolds numbers result in lower energy spectrum values.

Finally, in Fig. 7 we show LES solutions for different Kolmogorov constant values using $N = 20$ and $Re = 40$. As in the Reynolds variation case, in the lower k modes, all four results exhibit consistent behavior, maintaining a shared trend. Around $N = 7$, a noticeable divergence in the trends becomes apparent. In higher modes, solutions with larger C_k numbers tend to yield higher energy spectrum values, while lower C_k numbers result in lower energy spectrum values. So different C_k values could be used to adjust our results to the desired ones.

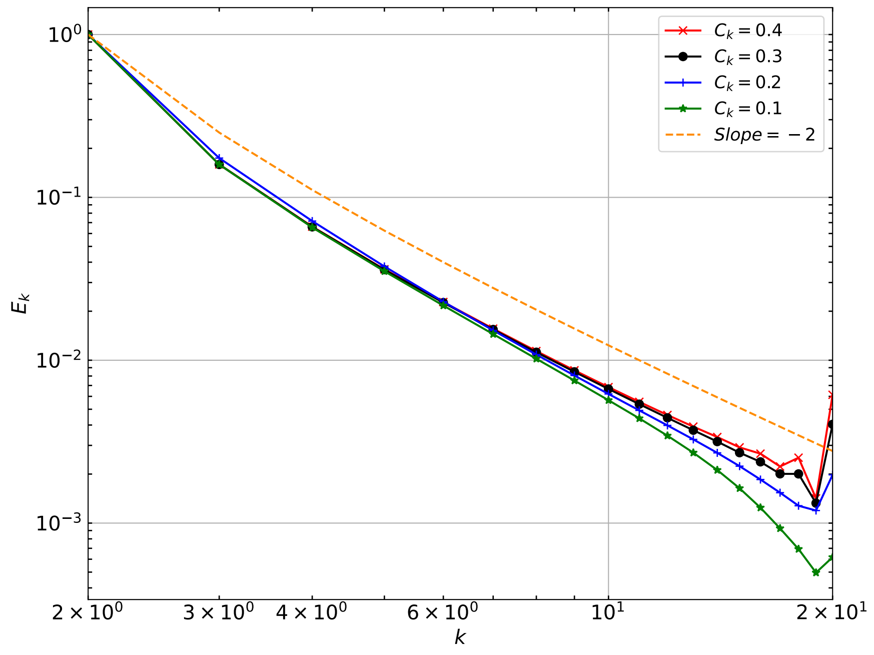

Figure 7: *LES solution of the energy spectrum for $N = 20$ and $Re = 40$ for different C_k values. The results are shown in logarithmic scale.*

6 Conclusions

In this work we have analysed the Burgers equation in Fourier space and we have implemented a code for Direct Numerical Simulation (DNS) and Large-Eddy Simulations (LES).

Before we started, we checked if our code was working properly by comparing our results with data from the CTTC report [2]. We ran DNS and LES simulations with different mode quantities and found differences between under-resolved and fully resolved solutions. But in general, we found that our results where similar to the reference ones. We also found some numerical issues for the highest mode in the DNS solution.

In the first part of the results, we looked at how the Reynolds number affects the Burgers equation. We ran DNS ($N = 100$) simulations for different Reynolds numbers. As expected, we saw that higher Reynolds numbers led to less energy dissipation in high k modes, while lower Reynolds numbers caused more dissipation. The highest Reynolds number matched the trend of the energy spectrum with a slope of $m = -2$, showing how the Reynolds number affects the Burgers equation.

In terms of LES, in the lower k modes, all four results displayed consistent behavior, maintaining a shared pattern. As we moved into higher k modes, solutions with larger Reynolds numbers tended to produce higher energy spectrum values, while lower Reynolds numbers led to lower energy spectrum values. Furthermore, we extended our analysis to consider different values of C_k. Similar to the Reynolds number variation, we observed that in the lower k modes, all four results exhibited consistent behavior. In higher k modes, we noted that solutions with larger values of C_k tended to yield higher energy spectrum values, while lower C_k values resulted in lower energy spectrum values.

References

[1] O. Métais and M. Lesieur, "Spectral large-eddy simulation of isotropic and stably stratified turbulence," *Journal of Fluid Mechanics*, vol. 239, pp. 157–194, 1992.

[2] CTTC, "Burgers equation in fourier space," November 17 th 2014.

7 Appendix: Python3 Code

Here we show the code in Python3 language for the Burgers problem. Further discussion around the code can be found in the link below the **Index** of this report.

```
#=======================================
# Code for Burgers equation
# Master in Space and Aeronautical
# Engineering.
# Computational Engineering: Assginment
# 4.
# Author: I aki Fernandez Tena
# email: inakiphy@gmail.com
#=======================================


#=======================================
# Import modules
#=======================================

import matplotlib.pyplot as plt
import numpy as np
from matplotlib.colors import ListedColormap
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm


#=======================================
# Physical input data
#=======================================
L  = 1.0                                  # Characteristic lenght
U  = 1.0                                  # Characteristic viscosity
Re = 40.0                                 # Reynolds number
nu = (U * L) / Re                         # Kinematic viscosity
i  = 1j                                   # Complex number i

#=======================================
# Numerical input data
#=======================================
N        = [20,100,20,20]                 # Vector length
Ck       = [0.4523,0.05]                  # Ck parameter
C        = 0.02                           # Cosntant for explicit scheme integration
time_max = 100000000000                   # Max loop time
eps      = 1.0E-06                        # Convergence parameter
m        = 2                              # Slope
LES      = 0                              # If LES=1 we compute LES


#=======================================
# Start solver for different N-s
#=======================================
for Z in range(len(N)):
    if Z==2:
        LES = 1
    delta_T  = (C * Re) / N[Z]**2          # Time step

    #=======================================
    # Define vectors
    #=======================================
```

```python

    uk      = np.zeros(N[Z], dtype=complex)      # Define u_k vectors as complex
    uk_zero = np.zeros(N[Z], dtype=complex)      # Define u_k at the first time step
    Ek      = np.zeros(N[Z])                     # Energy spectrum
    #=======================================
    # Initialize velocities
    #=======================================
    # We only compute k>0 values

    uk[0]      = 0.0                          # Impose it
    uk_zero[0] = 0.0                          # Impose it
    uk[1]      = 1.0                          # Impose it
    uk_zero[1] = 1.0                          # Impose it
    for k in range(2, N[Z]):
        uk_zero[k] = 1.0 / float(k+1)

    print("=========================")
    print(f"Starting loops for N= {N[Z]} & LES={LES}")
    print("=========================")
    #=======================================
    # Begin time step (t+delta_t)
    #=======================================
    for time in range(time_max):
        t0 = np.abs(np.sum(uk_zero))
        for k in range(2, N[Z]):
            # Compute the convective term first
            conv = 0                              # Set the convective term to 0
            for p in range(-N[Z]+1,N[Z]):
                q = k - p
                if (q < -N[Z] + 1) or (q >= N[Z]):    # If it enters, we are out of p
                    domain
                    uq = 0.0
                    up = 0.0
                else:
                    uq = np.conjugate(uk_zero[np.abs(q)]) if q < 0 else uk_zero[q]
                    up = np.conjugate(uk_zero[np.abs(p)]) if p < 0 else uk_zero[p]
                conv = conv + q * i * uq * up
            # Compute n+1 velocity
            if LES == 0:
                uk[k] = uk_zero[k] - delta_T * (nu * k**2 * uk_zero[k] + conv)
            elif LES == 1 :
                nu_inf = 0.31 * ((5 - m) / (m + 1)) * np.sqrt(3 - m) * Ck[Z-2] ** (-3.0 /
                     2.0)
                EkN    = (uk_zero[-1] * np.conjugate(uk_zero[-1])).real
                nu_a   = 1.0 + 34.5 * np.exp(-3.03 * (N[Z] / k) )
                nu_t   = nu_inf * ((EkN / N[Z]) ** (0.5) * nu_a)
                nu_eff = nu + nu_t
                uk[k] = uk_zero[k] - delta_T * (nu_eff * k**2 * uk_zero[k] + conv)

        t = np.abs(np.sum(uk))
        if np.abs(t-t0) < eps:
            print("======================")
            print(f"Time loop converged for N={N[Z]}")
            print("The requiered steps has been:" +  " " + str(time))
            print("======================")
            break
        else:
            uk_zero = uk
    print("=========================")
    print(f"Computing Energy spectrum for N={N[Z]} & LES={LES}")
    print("=========================")
    #=======================================
    # Compute energy spectrum
    #=======================================
    for k in range(N[Z]):
        Ek[k] = (uk[k] * np.conjugate(uk[k])).real
    if Z == 2 and LES == 1:
        k_cv1LES = np.linspace(1, N[0], N[0])               # Generate k points with the same spacing
        Ek1LES   = np.zeros(N[0])
        Ek1LES   = Ek
    elif Z == 3 and LES == 1:
        k_cv2LES = np.linspace(1, N[1], N[1])               # Generate k points with the same spacing
        Ek2LES = np.zeros(N[1])
        Ek2LES = Ek
    elif Z == 0 and LES == 0 :
        k_cv1 = np.linspace(1, N[0], N[0])               # Generate k points with the same spacing
        Ek1   = np.zeros(N[0])
        Ek1   = Ek
    else:
        k_cv2 = np.linspace(1, N[1], N[1])               # Generate k points with the same spacing
        Ek2 = np.zeros(N[1])
        Ek2 = Ek

print("=================================")
print("Starting plots")
print("=================================")

#=======================================
# Starting plots
#=======================================
output_plot = f"Results/Burguer_Re{Re}_C1{C}-DNS-LES.pdf"

# Create the references line
X = np.linspace(1, N[1], N[1])
Y = np.linspace(1, N[1], N[1])
for i in range(1,N[1]):
    Y[i] = (X[i])**-2

# Figure specifications
fontsize=15

# Start first plot
plt.figure(1)
plt.figure(figsize = (10,8))
plt.tick_params(axis='both', which='both',length=3, width=1.0,
labelsize=15, right=True, top=True, direction='in') # For ticks in borders

# Figure labels
plt.ylabel(r"$E_{k}$", fontsize=fontsize)
plt.xlabel(r"$k$", fontsize=fontsize)

# Limits
plt.xlim(k_cv1[1], N[1])
plt.ylim(10E-07,1)
plt.grid()

```

```python
# Plot
plt.loglog(k_cv1[1:], Ek1[1:],'-x', color='red', label=r"$N=20$")
plt.loglog(k_cv2[1:], Ek2[1:], '-o', color='black', label=r"DNS ($N=100$)")
plt.loglog(k_cv1LES[1:], Ek1LES[1:],'-+', color='blue', label=r"$N=20$ LES $C_{K}=0.4523$")plt.loglog(k_cv1LES[1:], Ek2LES[1:], '-*', color='green', label=r"$N=20$ LES $C_{K}=0.05$")
plt.plot(X + 1, Y, linestyle="dashed", color="darkorange", label=r"$Slope=-2$")

# Legend specifications
plt.legend(fontsize=fontsize-2, loc='upper right')

# Save figure
plt.savefig(output_plot, bbox_inches='tight')


plt.close(1)
print("=================================")
print("Code has finished succesfully")
print(f"You can find the results in: {output_plot}")
print("=================================")
```