MASTER'S DEGREE IN AERONAUTICAL ENGINEERING

Turbulence: Burgers equation

COMPUTATIONAL ENGINEERING

Authors:
Ricard Arbat Carandell

Professors:
Carlos David Perez Segarra
Francesc Xavier Trias Miquel

Escola Superior d'Enginyeria Industrial, Aeroespacial i Audiovisual de Terrassa (ESEIAAT)
Dated: November 21, 2024

Contents

Index of Figures

1 Introduction

Turbulence is a complex and chaotic phenomenon that appears in fluid flows with a high Reynolds number. It is characterized by an irregular and unpredictable motion, where vortices appear, and it involves a wide range of interacting scales, from very large to tiny ones.

The Navier-Stokes (NS) equations can be used to model turbulence in incompressible flows. However, direct numerical simulation (DNS) of these equations is prohibitive due to the huge number of scales that need to be resolved, as seen in Kolmogorov's scaling laws. Therefore, simplified models, such as the Burgers equation, are employed to capture essential features of turbulence, such as nonlinearity and energy transfer between scales, while being computationally less intensive.

2 Numerical model

2.1 Burgers Equation

The Burgers' equation works as a simplified model that conserves many features of the 3D Navier-Stokes (NS) equations, such as nonlinearity and energy cascade properties. It is defined by:

$$\partial_t u + u\partial_x u = \frac{1}{Re}\partial_{xx} u + f, \tag{1}$$

where Re represents the Reynolds number, and f is an external forcing term.

2.1.1 Fourier Space Representation

To solve the problem, the Burgers' equation in Fourier space is considered in an interval Ω with periodic boundary conditions. This interval defines scales that will be solved. Finally, the Fourier-transformed equation reads:

$$\partial_t \hat{u}_k + \sum_{k=p+q} \hat{u}_p i q \hat{u}_q = -\frac{k^2}{Re}\hat{u}_k + F_k, \quad k = 0, \ldots, N, \tag{2}$$

where $\hat{u}_k(t) \in \mathbb{C}$ denotes the k-th Fourier coefficient of $u(x, t)$

2.1.2 Kinetic Energy Transport Equation

The kinetic energy E_k for the k-th mode can be derived by multiplying $\hat{u}_k$ with its complex conjugate, resulting in:

$$\partial_t E_k = -\frac{2k^2}{Re}E_k - \left(\hat{u}_k C_k(\hat{u}_p, \hat{u}_q) + \hat{u}_k^* C_k(\hat{u}_p, \hat{u}_q)\right) + \hat{u}_k F_k + F_k \hat{u}_k^*, \tag{3}$$

where $C_k(\hat{u}_p, \hat{u}_q)$ is the convective contribution defined by:

$$C_k(\hat{u}_p, \hat{u}_q) = \sum_{k=p+q} \hat{u}_p i q \hat{u}_q. \tag{4}$$

2.1.3 Diffusion and Convection

The diffusive term,

$$D = -\frac{2k^2}{Re} E_k, \tag{5}$$

acts as a damping mechanism, predominantly effective at higher frequencies (small scales). Note how the wave number k has a big effect on its value.

Otherwise, the convective term

$$C = -\left(\hat{u}_k C_k(\hat{u}_p, \hat{u}_q) + \hat{u}_k^* C_k(\hat{u}_p, \hat{u}_q)\right), \tag{6}$$

transfers energy between different scales, facilitating both forward (large to small scales) and backward (small to large scales) energy cascades.

2.1.4 Initial and Boundary Conditions

For the analysis, assume an initial condition where $\hat{u}_k = k^{-1}$ for $k > 0$ and $\hat{u}_0 = 0$ (no mean flow). The forcing term F_k is defined such that $\hat{u}_1(t)$ remains constant and equal to 1 over time, and therefore

$$F = 0 \tag{7}$$

2.2 Discretized form

2.2.1 Euler method

In order to obtain the value of the energy E_k, it is necessary to compute the value of the velocities $\hat{u}_k$ for each mode and for each time step. To do this, equation 3 is discretized and introduced into Euler's time scheme

$$\hat{u}_k^{n+1} = \hat{u}_k^n + \Delta t \; E_k(\hat{u}_k^n) \tag{8}$$

resulting in

$$\hat{u}_k^{n+1} = \hat{u}_k^n + \Delta t \; (-\frac{2k^2}{Re} E_k + (\hat{u}_k \sum_{k=p+q} (\hat{u}_p i q \hat{u}_q)^* + \hat{u}_k^* \sum_{k=p+q} \hat{u}_p i q \hat{u}_q)) = \hat{u}_k^n + \Delta t (D + C) \tag{9}$$

where

$$\hat{u}_k = k^{-1} \qquad \hat{u}_0 = 0 \qquad \hat{u}_1(t) = 1$$

2.2.2 Convective term calculation

The convective terms $C_k(\hat{u}_p, \hat{u}_q)$ need to be calculated taking in account the domain Ω seen in Figure 1, which represents the triadic interactions possible for straight lines bertween $k = 0$ and $k = N$.

$$C_k(\hat{u}_p, \hat{u}_q) = \sum_{k=p+q} \hat{u}_p i q \hat{u}_q. \tag{10}$$

When designing the summation algorithm, one needs to take two things in account:

- **Values for q and p:** For each wave number k, a summation from $q_i = k - N$ to $q_f = N$ is done, and for each summation step $p_n = k - q_n$.
- **Velocity $\hat{u}_k$ for negative q and p:** Velocity values that fall in negative p and q values (negative modes) need to be converted by applying the complex conjugate to them

$$\hat{u}_k = \hat{u}^*_{-k}$$

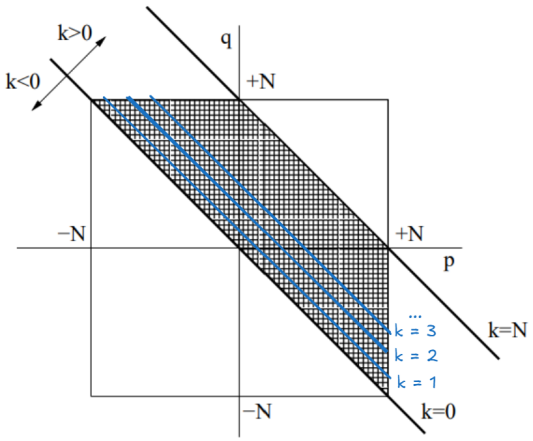


Figure 1. Triadic interactions' domain. For each wave number (k), the velocities along a line parallel to the k = 0 line are added, resulting in the parameter C_k, as seen in Equation 10

2.2.3 Time step

As for the time step size, a simple CFL like condition needs to be satisfied

$$\Delta t < C_1 \frac{Re}{N^2} \tag{11}$$

2.2.4 Energy calculation

And finally, the energy for each wave number k can be easily calculated by multiplying the velocities by their conjugates

$$E_k = \hat{u}_k \hat{u}^*_k \tag{12}$$

2.3 Models

2.3.1 Direct Numerical Soltion (DNS)

DNS solves the Navier-Stokes (NS) equations without any approximation, capturing all scales of turbulence, from the largest energy-carrying eddies to the smallest dissipative scales defined by the Kolmogorov scale. This results in highly accurate representations of turbulent flows. However, the computational cost of DNS is immense due to the wide range of scales that must be resolved, scaling with the Reynolds number Re as:

$$\text{Computational cost} \sim Re^{11/4}. \tag{13}$$

Solving the Burgers equation with DNS implies setting N to a bigger value, such as $N = 100$, thus solving the equations up to a higher frequency. This implies a much longer time required to solve the system of equations.

2.3.2 Large Eddy Simulation (LES)

LES offers a more feasible approach for higher Reynolds number flows by modeling only the largest turbulent scales explicitly, while the smaller scales are approximated using models. This results in significantly reduced computational cost compared to DNS, while still capturing the essential physics of the flow.

For this case, the spectral eddy-viscosity model has been considered. It is a simple model that in our case can be implemented by introducing a turbulent viscosity (ν_t) to the diffusive term

$$D = -2k^2 \left(\frac{1}{Re} + \nu_t \right) E_k, \tag{14}$$

where the turbulent viscosity is defined as

$$\nu_t = \nu_t^{+\infty} \left(\frac{E_{k_N}}{k_N} \right)^{1/2} \nu_t^* \left(\frac{k}{k_N} \right) \tag{15}$$

$$\nu_t^{+\infty} = 0.31 \frac{5-m}{m+1} \sqrt{3-m} C_K^{-3/2} \tag{16}$$

$$\nu_t^* \left(\frac{k}{k_N} \right) = 1 + 34.5 e^{-3.03(k_N/k)} \tag{17}$$

and, for our case, $C_K \approx 0.4523$ is the Kolmogorov constant and $m = -2$ is the slope of the energy spectrum.

3 Code implementation

The developed code simulates the solution of a turbulent flow problem using Fourier modes and evaluates various parameters like kinetic energy over time. Up next it's implementation will be explained:

3.1 Overall algorithm

The overall algorithm follows the typical resolution scheme, where we input the parameters, solve until convergence is accomplished, and finally obtain the results and plots. In a more detailed way, it writes as:

1. **Parameter initialization**: Input of the Fourier modes N, Reynolds number Re, time-stepping constant $C1$, and other simulation-specific parameters like Ck and whether to use LES or DNS.

2. **Initial boundary conditions**: Definition of the initial boundary conditions, such as the velocities $\hat{u}_0$ as $\hat{u}_0 = \frac{1}{k}$ for all wave numbers k.

3. **Iterative solver (Gauss-Seidel)**: For each Fourier mode k, the velocity profiles $\hat{u}_k$ are calculated in an iteration loop:

   - **Convective term calculation**: Use nonlinear interactions between modes to compute the convective term C_k.
   - **Diffusive term calculation**: Compute D_k using viscosity (with adjustments if LES is active).
   - **Velocity profile update**: Update $\hat{u}$ using an Euler scheme.
   - **Check Convergence**: Calculate the error as the maximum change in $\hat{u}$ between time steps; stop iterating when below a set tolerance.

4. **Kinetic energy calculation**: Once the solution has converged, compute the kinetic energy spectrum E_k for each mode.

5. **Output and plots**: Return the results, including the kinetic energy spectrum, wave numbers, iteration count, and elapsed time. Generate plots of E_k versus k and other comparisons as needed.

3.2 Detailed resolution pseudocode

Furthermore, a detailed pseudocode resolution has been developed to explain the code:

3.2.1 Gauss Seidel Function

The Gauss Seidel function implements a numerical resolution using the Gauss Seidel method and uses the Euler method to calculate the next loop velocities. Iterations are repeated until the convergence criteria has been reached.

function GAUSS_SEIDEL($\hat{u_0}, C_1, N, Re, LES, Ck, k$)
  $convergence \leftarrow 10^{-5}$
  $error \leftarrow 1$
  $i \leftarrow 0$
  $dt \leftarrow \frac{C_1 \cdot Re}{N^2}$
  $\hat{u}[:, 1] \leftarrow \hat{u_0}$
  Start timer $t\_elapsed$
  while $error > convergence$ **do**
    $i \leftarrow i + 1$
    $C \leftarrow$ CONVECTIVE_TERM($\hat{u}[:, tu], N$)
    $D \leftarrow$ DIFFUSIVE_TERM($\hat{u}[:, tu], N, Re, Ck, LES$)
    $dE \leftarrow D + C$
    $\hat{u}[k, tu + 1] \leftarrow \hat{u}[k, tu] - dt \cdot dE$
    $\hat{u}[1, tu + 1] \leftarrow 1$
    $error \leftarrow \max(|\hat{u}[:, tu + 1] - \hat{u}[:, tu]|)$
    $tu \leftarrow tu + 1$
  end while
  $Ek[k] \leftarrow \hat{u}[k, tu] \cdot \text{conj}(\hat{u}[k, tu])$
  Stop timer $t\_elapsed$
  return $Ek, i, t\_elapsed$
end function

3.2.2 Convective Term Function

To calculate the convective term, the triadic interactions inside a specific domain need to be calculated. See Section 2.2.2 for more details about it.

function CONVECTIVE_TERM($\hat{u}, N$)
  Initialize $Ck \leftarrow 0$ array of size N
  for $k \leftarrow 1$ to N **do**
    for $q \leftarrow k - N$ to N **do**
      $p \leftarrow k - q$

```
            if q < 0 then
                û[q] ← conj(û[−q])
            else if q > 0 then
                û[q] ← û[q]
            else
                û[q] ← 0
            end if
            if p < 0 then
                û[p] ← conj(û[−p])
            else if p > 0 then
                û[p] ← û[p]
            else
                û[p] ← 0
            end if
            Ck[k] ← Ck[k] + û[p] · q · i · û[q]
        end for
    end for
    return Ck
end function
```

3.2.3 Diffusive Term Function

Finally, the diffusive term needs to be calculated taking in account if we are doing a LES or a DNS simulation. The specifics on how to calculate the ν_t can be found in Section 2.3.2

```
function DIFFUSIVE_TERM(u_hat, N, Re, Ck, LES)
    Initialize D ← 0 array of size N
    m ← 2
    for k ← 1 to N do
        if LES = 1 then
            Compute ν∞, νa, Ekn using LES model
            νt ← νinf · (Ekn/N)^(1/2) · νa
        else
            νt ← 0
        end if
        D[k] ← k^2 · (1/Re + νt) · u_hat[k]
    end for
    return D
end function
```

4 Results and discussion

The code results in Figure 2, where different cases have been tested in order to see the behavior of the model. All of them have been tested at a low Reynolds of 40 and a C1 of 0.03, and have different particularities:

- **DNS with N = 20:** With a low N number, the DNS behaves quite poorly. Oscillations are present throughout all the wave numbers, but particularly at the end, where it seems to diverge. At lower k numbers, the value also diverges quite a lot from the expected solution.
- **DNS with N = 100:** As the N value becomes bigger, to a value of N = 100, the solution becomes much more stable, behaving as expected. The constant slope zone is clearly defined, and the final part where E_k drops is also clearly defined.
- **LES with N = 20 and C_k = 0.4523:** For the LES case using the Kolmogorov constant value, the results clearly match the DNS with N = 100, altough only reaching the value of k = 20. At the end of its values, some oscillations can be observed, but taking in account that this calculation is 80 times faster to calculate, it is quite reasonable.
- **LES with N = 20 and C_k = 0.05:** Finally, another LES case has been tested, now with C_k = 0.05, much smaller than it should be. The case starts working fine, but then drops earlier than expected. This is due to an excessive dissipation of energy due to the diffusion and the turbulent viscosity (ν_t), as the Kolmogorov constant is not used.

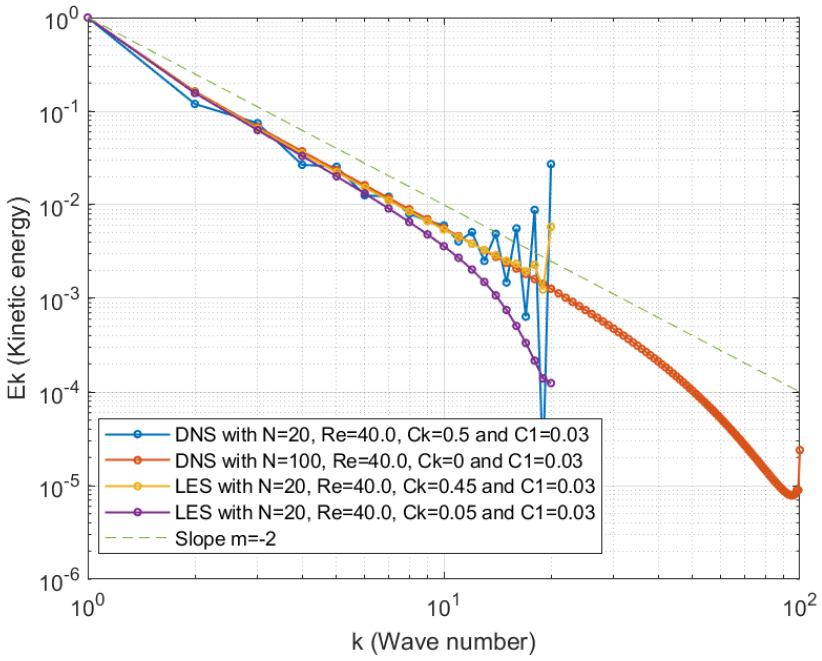
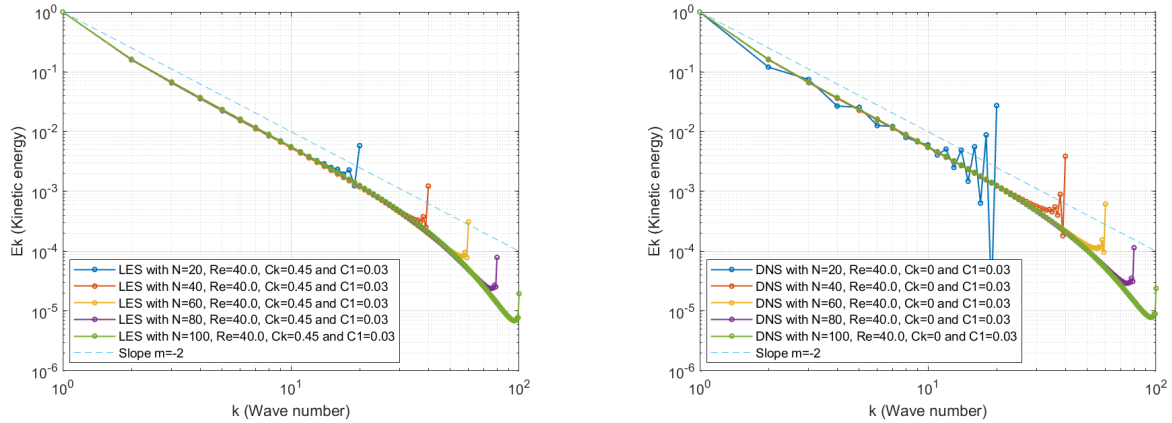


Figure 2. Burgers equation solutions for diferent cases considering DNS/LES, Reynolds number and the parameters C_k and C_1

4.1 N comparison

Tests with different numbers of Fourier modes (N) were also tested. The main difference between tests is how LES is much more stable at lower N numbers than DNS.

As seen in Figure 9a, at lower N numbers, such as $N = 20$, the DNS solution is very unstable, while the LES provides more accurate results.



(a) Comparison of Fourier mode number (N) for LES **(b)** Comparison of Fourier mode number (N) for DNS

Figure 3. Side-by-side comparison of Fourier mode number (N) for LES annd DNS cases.

4.1.1 Effect of N on time

The number of Fourier modes evaluated had a large impact on the computation time, as it grew exponentially. There was little difference between cases, but as LES was stable at lower N values, it could result in much more efficient results. For example, a LES at N = 40 took 0.87s, while a DNS at N = 100 required 18.64s, so it took 21.4 times longer to calculate for equivalent results.

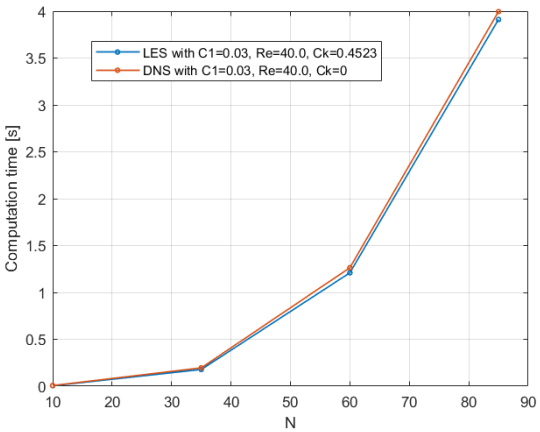


Figure 4. Effect of N number on time

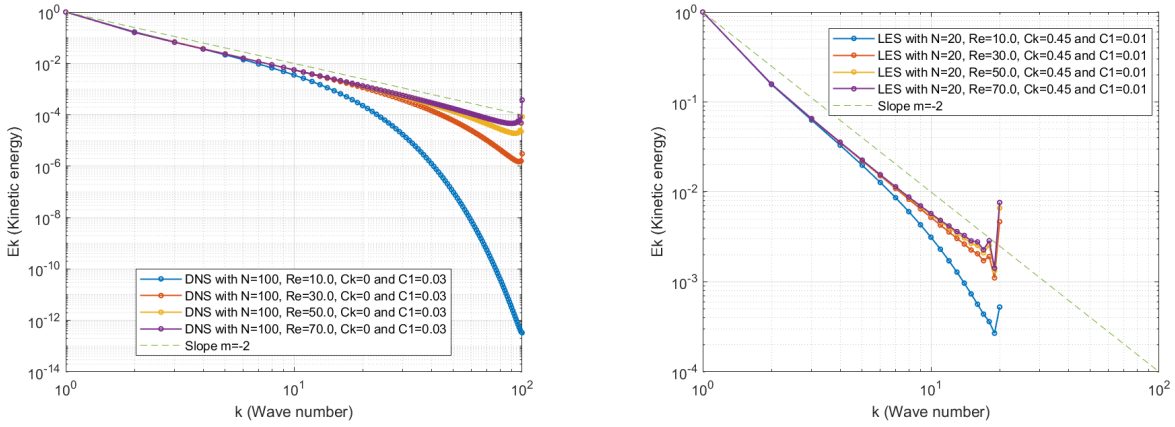
4.2 Reynolds comparison

Some tests were performed for different Reynolds numbers. In these tests, lower and higher Reynolds have been tested, and various behaviors can be seen in Figure 5

4.2.1 Small Reynolds numbers

For the DNS case, as Re increases, the solution becomes closer to the constant slope of $m = -2$. The case of $Re = 10$ is particularly different, as it drops really fast due to the larger impact of the diffusion versus the convection.

For the LES cases, the behavior is quite similar to the DNS, but only reaching $k = 20$. Nonetheless, $Re = 10$ drops much faster, while the rest have closer values between them.



(a) Lower Reynolds comparison for DNS

(b) Lower Reynolds comparison for LES

Figure 5. Side-by-side comparison of Reynolds simulations for DNS and LES.

4.2.2 Higher Reynolds numbers

Some tests have also been performed at higher Re numbers ranging from 10 to 4000 with a LES model. For these cases, a few observations can be made:

- The case of $Re = 10$ completely matches the constant slope case when compared to the other ones.
- The rest of the cases drop faster to lower k numbers, but then come back up.
- For these cases, the C_1 constant required a much lower value to achieve stable results. For bigger values, the higher Re curves would diverge and not output valid values.

As for the DNS model, only the case of $Re = 10$ would output valid values, no matter the value of the other parameters. Therefore, the results were discarded.

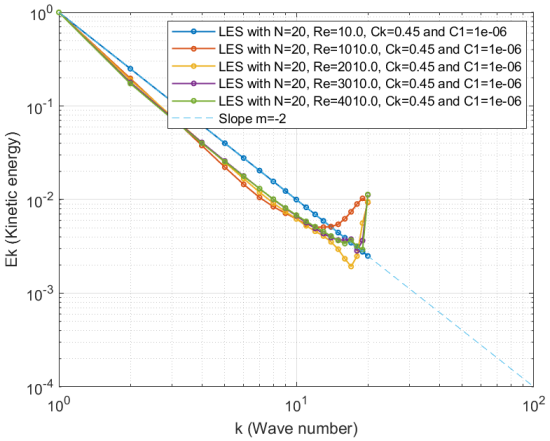


Figure 6. Higher Reynolds comparison for LES

4.2.3 Effect of Re on time

All in all, larger Re numbers produced smaller calculation times, particularly for the lowest Re numbers such as 10. As teh Re got bigger, the difference between values seems to become smaller.

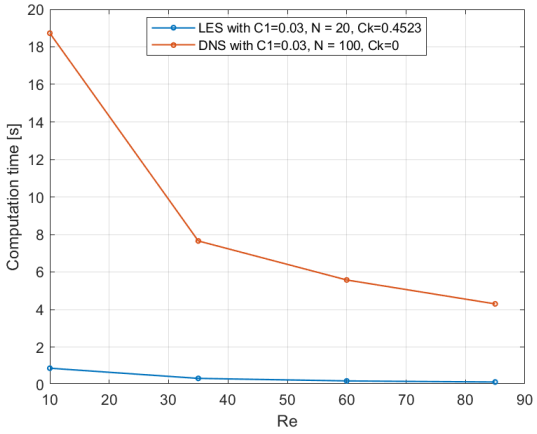


Figure 7. Effect of Reynolds number on time

4.3 C_k comparison

The comparison for different coefficients C_k only made sense with the case of a LES simulation. In Figure 8 it can be seen how, as C_k becomes bigger, the effect of convection becomes stronger, until reaching the Kolmogorov constant value $C_k = 0.4523$. Values bigger than that yield results slightly bigger than expected, but the effect is less important.

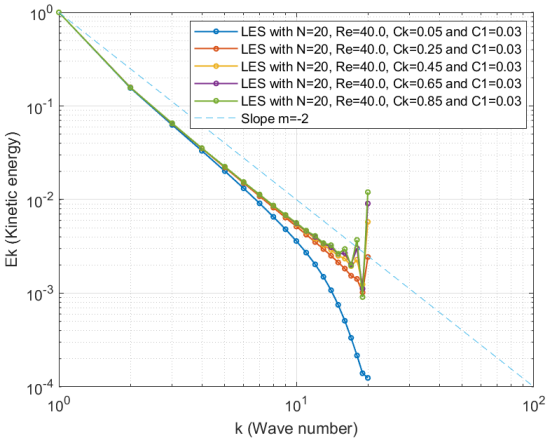
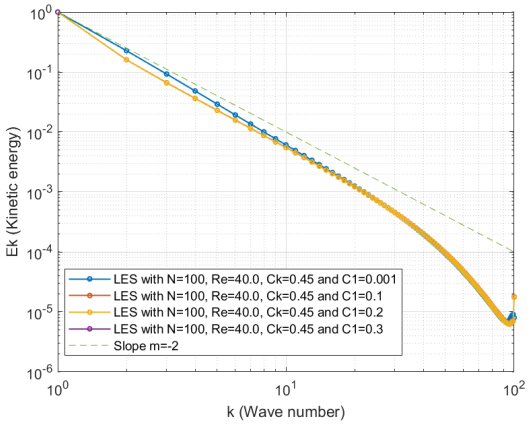


Figure 8. Effect of the parameter C_k in a LES simulation

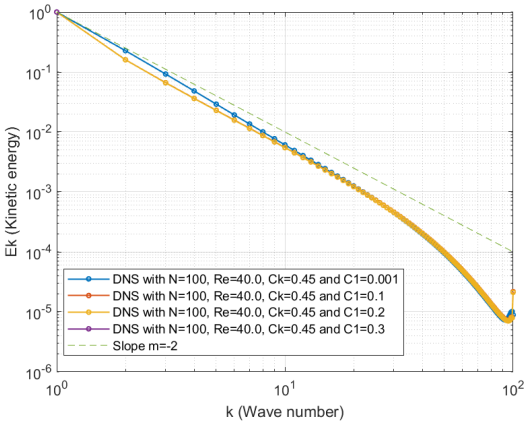
4.4 C_1 comparison

The effect of the time parameter C_1 was also analyzed. For both simulations there was little difference, but a few things could be observed:

- For lower values like $C_1 = 0.001$, both solutions got closer to the constant slope line.
- Values of $C_1 = 0.1 - 0.2$, yielded very similar solutions.
- If the value was higher, $C_1 = 0.3$, the solution diverged.



(a) Comparison of time parameter (C_1) effect for LES



(b) Comparison of time parameter (C_1) effect for DNS

Figure 9. Side-by-side comparison of Fourier mode number (N) for LES annd DNS cases.

4.4.1 Effect of C_1 on t

The parameter C_1 also had a big effect on the time. Clearly, as C_1 became smaller, and therefore also Δt, the time increased particularly. However, when too big values were reached, at

around $C_1 = 0.3$, the solutions started to diverge.

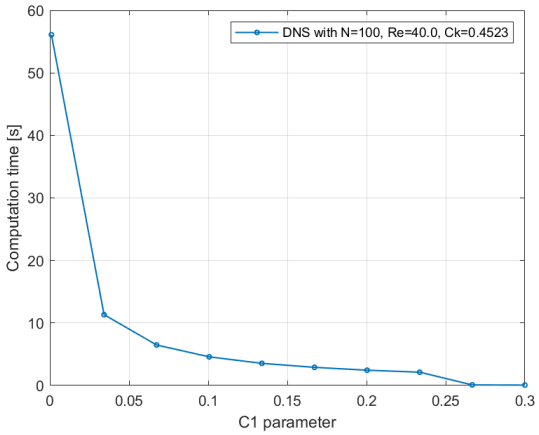


Figure 10. Effect of the parameter C_1 in the time of a LES simulation

5 Conclusions

The implementation and analysis of the Burgers' equation using Fourier modes, as outlined in this report, provide significant insights into the computational modeling of turbulence:

- **Accuracy vs. Efficiency**: The Direct Numerical Simulation (DNS) and Large Eddy Simulation (LES) models both offer unique benefits. While DNS provides highly accurate results by capturing all turbulence scales, it demands significant computational resources. LES, on the other hand, strikes a balance between accuracy and efficiency by modeling smaller scales, allowing for faster computations with reasonable accuracy.

- **Parameter sensitivity**: The simulations demonstrate that the choice of parameters, such as the number of Fourier modes N and the constants $C1$ and Ck, significantly affects both the stability and accuracy of the results. For instance, setting Ck close to the Kolmogorov constant enhances LES stability, while $C1$ directly influences the time step and thus the convergence behavior.

- **Energy transfer mechanisms**: The results confirm the expected energy cascade properties, with the convective and diffusive terms playing crucial roles in distributing energy across scales. The simulations effectively show how larger scales transfer energy to smaller ones (and vice versa), reproducing the characteristics of turbulent flows.

- **Computation times**: The comparison of computational times shows that using LES with optimal parameter tuning can drastically reduce simulation time compared to DNS, making LES suitable for preliminary or large-scale studies where exact detail is not critical.

References

[1] CTTC. (2024, March). *Burgers equation in Fourier space* (tech. rep.). CTTC.

[2] Wikipedia contributors. (2024a). *Burgers' equation — wikipedia, the free encyclopedia* [Accessed: 2024-11-16]. https://en.wikipedia.org/wiki/Burgers%27_equation

[3] Wikipedia contributors. (2024b). *Euler method — wikipedia, the free encyclopedia* [Accessed: 2024-11-16]. https://en.wikipedia.org/wiki/Euler_method

A Matlab Code

The provided code simulates the solution of a turbulent flow problem using Fourier modes and evaluates various parameters like kinetic energy over time. The main file, `solve_case.m`, serves as the primary function called within test cases that are executed through scripts such as `burgers.m`. The main computational logic resides in the auxiliary functions: `gauss_seidel.m`, `convective.m`, and `diffusive.m`.

The `burgers.m` will not be added here, as it only makes use of the output of the functions in order to make some plots and does not provide any additional data. However, it will be available in a folder delivered altogether with the report.

A.1 Function Breakdown

A.1.1 solve_case.m

`solve_case.m` initializes parameters for a specific problem case and calls the solver function `gauss_seidel.m`:

- Initializes the wave numbers k and the initial velocity profile $\hat{u}_0$.
- Calls `gauss_seidel.m` to compute the energy E_k, number of iterations i, and elapsed time t.
- Returns a legend string to identify the simulation type (e.g., DNS or LES).

```
function [Ek,k,i,legend,t] = solve_case(N,C1,Ck,Re,LES)

    % Modes and initial velocities
    k = 1:N;
    u_hat_0 = 1./k';

    % Solver
    [Ek,i,t] = gauss_seidel(u_hat_0, C1, N, Re, LES, Ck, k);

    % Legend string
    if LES==1
        name = "LES";
    else
        name="DNS";
    end

    legend = sprintf("%s with N=%i, Re=%.1f, Ck=%.2g and C1=%.2g",name, N,
   Re, Ck,C1);

```

```
end
```

Listing 1. Solve case function

A.1.2 gauss_seidel.m

`gauss_seidel.m` implements an explicit time-stepping integration scheme:

- Computes convective (`convective.m`) and diffusive (`diffusive.m`) terms iteratively.
- Uses an Euler time scheme to update the velocity field $\hat{u}$ at each time step.
- Iterates until the solution converges based on a defined tolerance.
- Returns the computed kinetic energy E_k for each mode, number of iterations, and elapsed time.

```
function [Ek,i,t_elapsed] = gauss_seidel(u_hat_0, C1, N, Re, LES, Ck, k)

% Explicit time integration scheme

% Numerical variables
convergence = 1e-5; % convergence error
error = 1;
i = 0;
dt = C1*Re/N^2; % Time step

% Preallocation
tu = 1; % Initial time unit. Starting from second tu, as tu=1 is known
u_hat(:,tu) = u_hat_0; % Start updating at this point

% Getting time
tic;

% Iterating until error is
while error > convergence
    i = i+1;

    % Convective and diffusive terms calculation
    C = convective(u_hat(:,tu),N);
    D = diffusive(u_hat(:,tu),N,Re,Ck,LES);
    dE = D+C;

    % Next velocity calculation
    u_hat(k,tu+1) = u_hat(k,tu)-dt*dE; % Euler time scheme
```

```
30      u_hat(1,tu+1) = 1; % Setting u0(t) = 1 to force the solution and
        instead of Fk
31
32
33      % Error calculation.
34      error = max(abs(u_hat(:,tu+1) - u_hat(:,tu))); % When u is stable
        finish iterating
35
36      % Advance time unit
37      tu = tu+1;
38
39  end
40
41  % Calculating kinetic energy after iterating
42  Ek(k) = u_hat(k,tu).*conj(u_hat(k,tu));
43
44  % Elapsed time
45  t_elapsed = toc;
46
47  end
```

Listing 2. Gauss Seidel function

A.1.3 convective.m

`convective.m` calculates the convective term for each Fourier mode:

- Iterates over possible mode combinations q and p such that $p + q = k$.
- Handles conjugate symmetry conditions for negative indices.
- Updates the convective term C_k using the nonlinear interaction between modes.

```
1
2   function [Ck] = convective(u_hat,N)
3
4   % Preallocation
5   Ck = zeros(N,1);
6
7   for k=1:N % Solving for each Fourier mode
8
9       for q=k-N:N
10      p = k-q;
11
12          % If q is negative, get the conjugate of the positive
13          if q < 0
14              u_hat_q = conj(u_hat(-q));
```

```
        elseif q > 0
            u_hat_q = u_hat(q);
        else
            u_hat_q = 0;
        end

        % If p is negative, get the conjugate of the positive
        if p < 0
            u_hat_p = conj(u_hat(-p));
        elseif p > 0
            u_hat_p = u_hat(p);
        else
            u_hat_p = 0;
        end

        % Calculation of the convective term
        Ck(k) = Ck(k) + u_hat_p*q*1i*u_hat_q;


    end
end

end
```

Listing 3. Convective term calculation function

A.1.4 diffusive.m

`diffusive.m` calculates the diffusive term:

- Determines whether the LES (Large Eddy Simulation) model is applied.
- If LES is enabled, calculates a spectral eddy viscosity ν_t.
- Computes the diffusive term $D_k = k^2 \left(\frac{1}{Re} + \nu_t\right) \hat{u}_k$.

```

function [D] = diffusive(u_hat,N,Re,Ck,LES)
% Preallocation
D = zeros(N,1);
m = 2;
% Solving for each Fourier mode
for k = 1:N

    if LES == 1 % LES model case
        nu_inf = 0.31*((5-m)/(m+1))*sqrt(3-m)*Ck^(-3/2);
```

```
        nu_a = 1 + 34.5*exp(-3.03*N/k); % Spectral non_dimensional eddy
    viscosity
        E_kn = u_hat(N)*conj(u_hat(N));  % Energy cutoff frequency

        nu_t = nu_inf*(E_kn/N)^(1/2)*nu_a; % Turbulent viscosity calculated

    else % DNS model case
        nu_t = 0; % No turbulent viscosity
    end

    % Diffusive term calculation
    D(k) = k^2*(1/Re + nu_t)*u_hat(k);
end

end
```

Listing 4. Diffusive term calculation function

A.2 Parameters

- **N**: Number of Fourier modes.
- **Re**: Reynolds number, indicating the relative effect of inertial forces to viscous forces.
- **C1, Ck**: Constants for time-stepping and LES model scaling.
- **LES**: Logical flag indicating whether to use LES or DNS.

A.3 Outputs

The outputs of the solver include:

- E_k: Kinetic energy spectrum.
- k: Wave numbers for plotting.
- i: Number of iterations until convergence.
- t: Elapsed computation time.