Implicit Large Eddy Simulation: Performance of WENO Schemes for Burger's Equation

Melodie Sloneker

Mentors: Dr. Hsien Shang, Somdeb Bandopadhyay

August 31, 2021

Objectives:

1. Implement high order finite difference WENO schemes for Burger’s Equation
2. Compute and test the energy spectra for Burger’s turbulence
3. Understand the basics of Large Eddy Simulation

Why ILES...?

Consider the Incompressible Navier-Stokes Equation:

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

$$\nabla \bullet \vec{u} = 0$$

where Reynolds Number is: $Re = \dfrac{\rho v_0 L}{\mu}$

$v_0, L \Rightarrow$ characteristic velocity, length

$\rho \Rightarrow$ density

$\mu \Rightarrow$ dynamic viscosity

Why ILES...? ...DNS is costly

$$\frac{\partial \vec{u}}{\partial t} + \boxed{(\vec{u} \bullet \nabla)\vec{u}} = \frac{1}{Re}\nabla^2 \vec{u} - \nabla p$$

convective term: can produce too many dynamically relevant scales

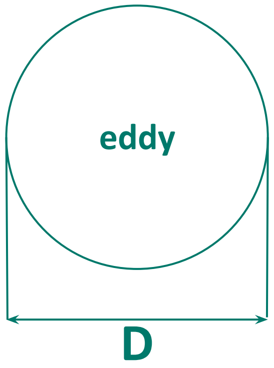
According to Kolmogorov: Reynold's Number Relates to smallest spatial and temporal scale (Frisch 1995)

$$\begin{cases} \delta t \sim Re^{-\frac{1}{2}} \\ \delta x \sim Re^{-\frac{3}{4}} \end{cases}$$

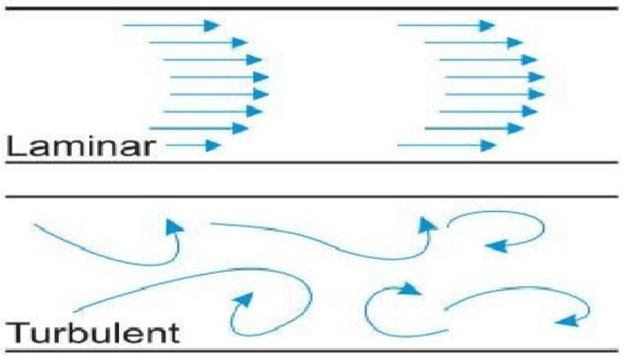
assuming a perfect algorithm scaling, this is often too high a computation/memory cost for DNS…

…we opt for Large Eddy Simulation!

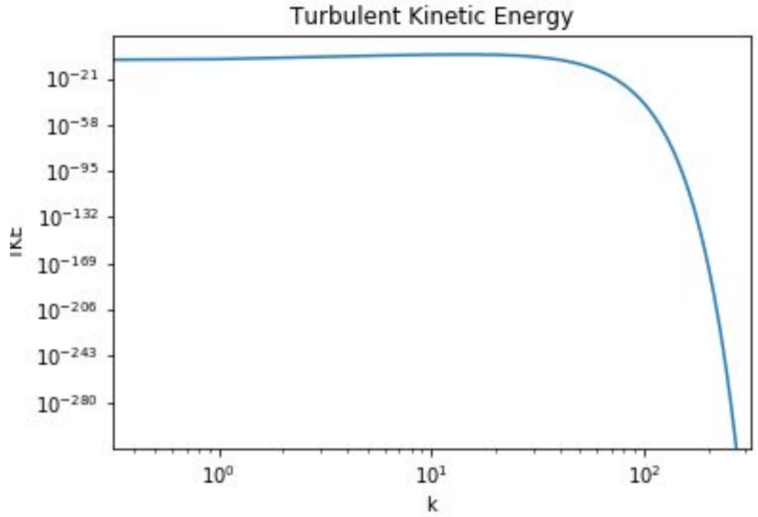
Large Eddy Simulation



turbulent flow: characterized by eddies of different shapes/sizes



[Intholo et al. 2020]



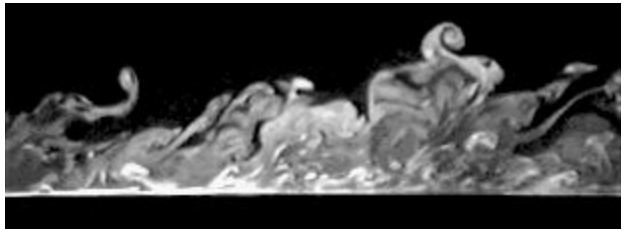
convenient to use Fourier domain to describe eddy size using relationship with wavenumber:

$$k = \frac{2\pi}{D}$$

the low wavenumber eddies contain most turbulent kinetic energy (TKE)

Large Eddy Simulation: we aim to resolve low-mid wavenumber eddies to capture most of the eddies' TKE

LES & ILES



Turbulent Flow
[University of Iowa Fluids Laboratory]

- Explicit Large Eddy Simulation (LES) ⇒ apply low pass filters to Navier-Stokes
  - (Smagorinski 1963, Lilly 1992)
- **Implicit Large Eddy Simulation (ILES)** ⇒ LES without filtering
  - inherent numerical dissipations acts as SGS filter

$$\boxed{\frac{\partial \vec{u}}{\partial t} + \vec{u}\frac{\partial \vec{u}}{\partial x} = \nu \frac{\partial^2 \vec{u}}{\partial x^2}}, \quad \nu \Rightarrow \text{kinematic viscosity}$$

Model Equation: 1D Viscous Burger's Equation

Discretization of Burger's Equation: Flux Splitting

Burger's Eqn. Conserved Form:

$$\frac{\partial \vec{u}}{\partial t} + \underbrace{\frac{\partial}{\partial x}\left(\frac{\vec{u}^2}{2}\right)}_{\boxed{\frac{\partial F}{\partial x}}} = \nu \frac{\partial^2 \vec{u}}{\partial x^2}$$

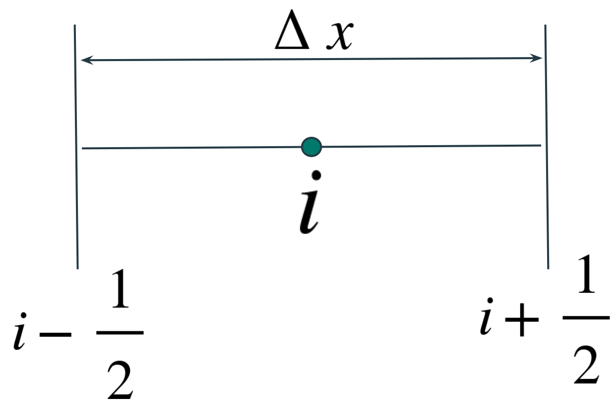
flux calculation

$$\boxed{\begin{aligned} F &= F^+ + F^- \\ F^+ &= \frac{1}{2}(F + \alpha U) \\ F^- &= \frac{1}{2}(F - \alpha U) \end{aligned}}$$

$$\frac{\partial \vec{u}}{\partial t} + \frac{\partial F}{\partial x} = \nu \frac{\partial^2 \vec{u}}{\partial x^2} \Rightarrow \frac{\partial \vec{u}}{\partial t} = \boxed{-\frac{1}{\Delta x}\left[\widehat{F}_{i+\frac{1}{2}} - \widehat{F}_{i-\frac{1}{2}}\right]} + \boxed{\frac{\nu}{(\Delta x)^2}\left[\vec{u}_{i+2} - 2\vec{u}_i + \vec{u}_{i-1}\right]}$$

$$G(u) = R(u) + L(u)$$

Discretization of Burger’s Equation: Flux Splitting (cont.)



For the Calculation of **R(u)**, following Lax-Friedrichs Splitting:

$$\widehat{F}_{i+\frac{1}{2}} = F^{+}_{i+\frac{1}{2}} + F^{-}_{i+\frac{1}{2}}$$

flux reconstruction at each interface

$$\widehat{F}_{i+\frac{1}{2}} = \left[\boxed{\left(F^{+}\right)^{left}_{i+\frac{1}{2}}} + \boxed{\left(F^{-}\right)^{right}_{i+\frac{1}{2}}} \right]$$

Reconstruct using WENO schemes

Discretization of Burger's Equation: Time Integration

For time integration, we use TVD RK3:

$$
\begin{cases}
u^{(1)} = u_j^l + \Delta t G\left(u_j^l\right) \\
u^{(2)} = \dfrac{3}{4} u_j^l + \dfrac{1}{4} u^{(1)} + \dfrac{1}{4} \Delta t G\left(u^{(1)}\right) \\
u^{(l+1)} = \dfrac{1}{3} u_j^l + \dfrac{2}{3} u^{(2)} + \dfrac{2}{3} \Delta t G\left(u^{(2)}\right)
\end{cases}
$$

ENO/WENO Schemes

Weighted Essentially Non-Oscillatory Schemes

discontinuities in solutions of hyperbolic conservation laws ⇒ spurious oscillations:

choose interpolation points over a stencil (avoid initiation of oscillations)

⇓

choose smoothest candidate stencil

⇓

avoids spurious oscillations near discontinuities, obtains information from smooth regions only!

ENO Schemes (Harten et al. 1986, 1987, 1997):

- first successful higher order spatial discretization method for hyperbolic conservation laws that achieves ENO property
- finite-difference ENO scheme (Shu & Osher 1998, 1999)

WENO Schemes (Liu et al. 1994, Jiang & Shu 1996):

- weighted ENO
- uses convex combination of all ENO candidate sub-stencils and assigns weight based on smoothness indicator

We are testing 6 different 5th order WENO Schemes:

WENO5-JS, WENO5-Z, WENO5-ZP, WENO5-NS, WENO5-M, WENO5-YC

Formulation For WENO5-JS

(Jiang & Shu 1996)

$$\boxed{q^L_{i+\frac{1}{2}} = \left(q_{i-2},\ q_{i-1},\ q_i,\ q_{i+1},\ q_{i+2}\right)}$$

smoothness measurement

$$\begin{cases} \beta_1 = \frac{13}{12}\left(q_{i-2} - 2q_{i-1} + q_i\right)^2 + \frac{1}{4}\left(q_{i-2} - 4q_{i-1} + 3q_i\right)^2 \\ \beta_2 = \frac{13}{12}\left(q_{i-1} - 2q_i + q_{i+1}\right)^2 + \frac{1}{4}\left(q_{i-1} - q_{i+1}\right)^2 \\ \beta_3 = \frac{13}{12}\left(q_i - 2q_{i+1} + q_{i+2}\right)^2 + \frac{1}{4}\left(3q_i - 4q_{i+1} + q_{i+2}\right)^2 \end{cases}$$

optimal weights

$$\begin{cases} d_1 = \frac{1}{10} \\ d_2 = \frac{6}{10} \\ d_3 = \frac{3}{10} \end{cases} \xrightarrow{\varepsilon = 10^{-6}} \left.\begin{aligned} \alpha_1 &= \frac{d_1}{(\beta_1+\varepsilon)^2} \\ \alpha_2 &= \frac{d_2}{(\beta_2+\varepsilon)^2} \\ \alpha_3 &= \frac{d_3}{(\beta_3+\varepsilon)^2} \end{aligned}\right\} \begin{aligned} \omega_1 &= \frac{\alpha_1}{\alpha_1+\alpha_2+\alpha_3} \\ \omega_2 &= \frac{\alpha_2}{\alpha_1+\alpha_2+\alpha_3} \\ \omega_3 &= \frac{\alpha_3}{\alpha_1+\alpha_2+\alpha_3} \end{aligned}$$

$$\begin{aligned} q^L_{i+\frac{1}{2}} &= \omega_1\left(\frac{1}{3}q_{i-2} - \frac{7}{6}q_{i-1} + \frac{11}{6}q_i\right) \\ &+ \omega_2\left(-\frac{1}{6}q_{i-1} + \frac{5}{6}q_i + \frac{1}{3}q_{i+1}\right) \\ &+ \omega_3\left(\frac{1}{3}q_i + \frac{5}{6}q_{i+1} - \frac{1}{6}q_{i+2}\right) \end{aligned}$$

other schemes are variations of WENO-JS (calculating smoothness differently)

Formulation for Mapped Weighted ENO Schemes (WENO5-M):

(Henrik, Aslam, & Powers 2005)

$$\boxed{q^L_{i+\frac{1}{2}} = \left(q_{i-2}, q_{i-1}, q_i, q_{i+1}, q_{i+2}\right)}$$

smoothness measurement

$$\begin{cases} \beta_1 = \frac{13}{12}\left(q_{i-2} - 2q_{i-1} + q_i\right)^2 + \frac{1}{4}\left(q_{i-2} - 4q_{i-1} + 3q_i\right)^2 \\ \beta_2 = \frac{13}{12}\left(q_{i-1} - 2q_i + q_{i+1}\right)^2 + \frac{1}{4}\left(q_{i-1} - q_{i+1}\right)^2 \\ \beta_3 = \frac{13}{12}\left(q_i - 2q_{i+1} + q_{i+2}\right)^2 + \frac{1}{4}\left(3q_i - 4q_{i+1} + q_{i+2}\right)^2 \end{cases}$$

optimal weights

$$\begin{cases} d_1 = \frac{1}{10} \\ d_2 = \frac{6}{10} \\ d_3 = \frac{3}{10} \end{cases} \xrightarrow{\varepsilon = 10^{-6}} \begin{aligned} \alpha_{1,js} &= \frac{d_1}{\left(\beta_1 + \varepsilon\right)^2} \\ \alpha_{2,js} &= \frac{d_2}{\left(\beta_2 + \varepsilon\right)^2} \\ \alpha_{3,js} &= \frac{d_3}{\left(\beta_3 + \varepsilon\right)^2} \end{aligned} \Bigg\} \begin{aligned} \omega_{1,js} &= \frac{\alpha_{1,js}}{\alpha_{1,js} + \alpha_{2,js} + \alpha_{3,js}} \\ \omega_{2,js} &= \frac{\alpha_{2,js}}{\alpha_{1,js} + \alpha_{2,js} + \alpha_{3,js}} \\ \omega_{3,js} &= \frac{\alpha_{3},js}{\alpha_{1,js} + \alpha_{2,js} + \alpha_{3,js}} \end{aligned}$$

mapping function

$$\begin{cases} \alpha_1 = f\left(d_1, \omega_{1,js}\right) \\ \alpha_2 = f\left(d_2, \omega_{2,js}\right) \\ \alpha_3 = f\left(d_3, \omega_{3,js}\right) \end{cases}, f(d, \omega) = \frac{\omega\left(d + d^2 - 3d\omega + \omega^2\right)}{d^2 + \omega(1 - 2d)}$$

$$\omega_1 = \frac{\alpha_1}{\alpha_1 + \alpha_2 + \alpha_3}$$

$$\omega_2 = \frac{\alpha_2}{\alpha_1 + \alpha_2 + \alpha_3}$$

$$\omega_3 = \frac{\alpha_3}{\alpha_1 + \alpha_2 + \alpha_3}$$

$$\begin{aligned} q^L_{i+\frac{1}{2}} = &\ \omega_1\left(\frac{1}{3}q_{i-2} - \frac{7}{6}q_{i-1} + \frac{11}{6}q_i\right) \\ &+ \omega_2\left(-\frac{1}{6}q_{i-1} + \frac{5}{6}q_i + \frac{1}{3}q_{i+1}\right) \\ &+ \omega_3\left(\frac{1}{3}q_i + \frac{5}{6}q_{i+1} - \frac{1}{6}q_{i+2}\right) \end{aligned}$$

Defining the Problem: Decaying Burger's Turbulence

$$\frac{\partial \vec{u}}{\partial t} + \frac{\partial}{\partial x}\left(\frac{\vec{u}^2}{2}\right) = \nu \frac{\partial^2 \vec{u}}{\partial x^2}$$

Following the model problem of Maulik & San:
(Maulik & San 2017)

Domain & Boundary:

$x \in [0, 2\pi]$, periodic

Viscosities:

$\nu = 1 \times 10^{-4}$

$\nu = 5 \times 10^{-4}$

$\nu = 1 \times 10^{-3}$

of Grid Points:

512, 1024, 2048

Initialization

Goal: initialize *u(x)* in real space

1. Start from initial energy spectra in Fourier Domain:

$$E(k) = Ak^4 e^{-\left(\frac{k}{k_0}\right)^2}, \int E(k)\, dk = \frac{1}{2}$$

$$A = \frac{2k_0^{-5}}{3\sqrt{\pi}}, \ k_0 = 10$$

2. Energy is related to Fourier Coefficients:

$$E = \frac{1}{2} u^2$$

$$\Downarrow$$

$$|\hat{u}(k)| = \sqrt{2E(k)}$$

3. Our velocity field in Fourier Space has an amplitude and a randomly generated phase:

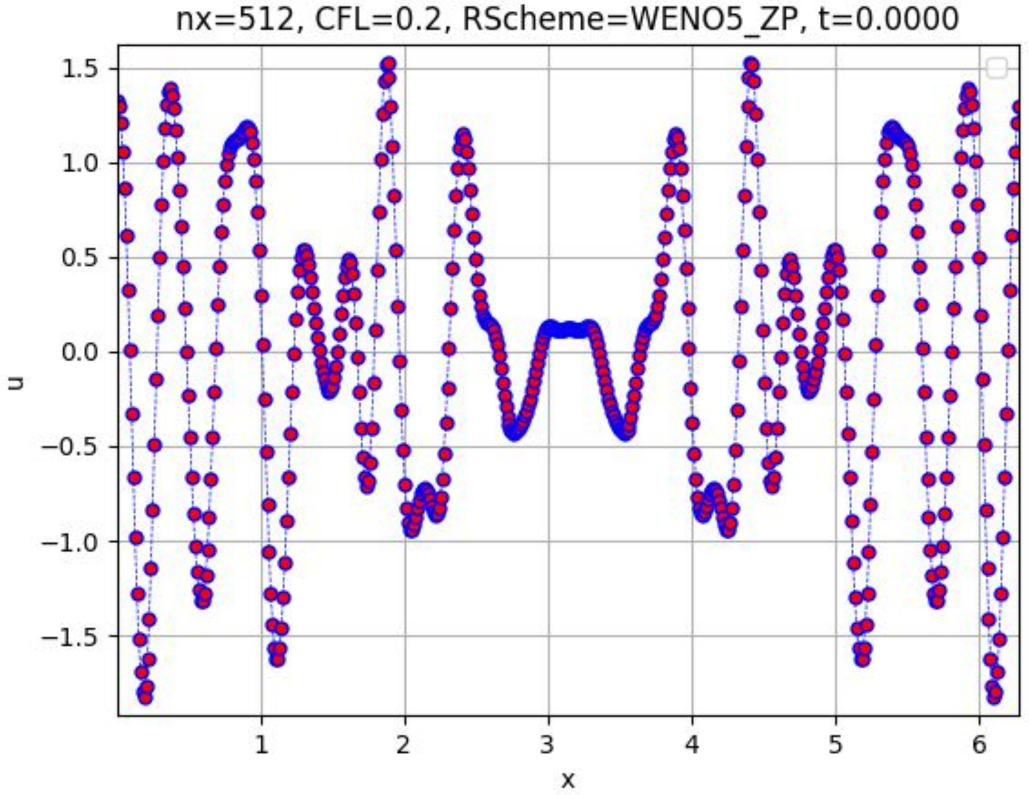
$$\hat{u}(k) = |\hat{u}(k)|\, e^{i \cdot 2\pi \cdot \Psi(k)}$$

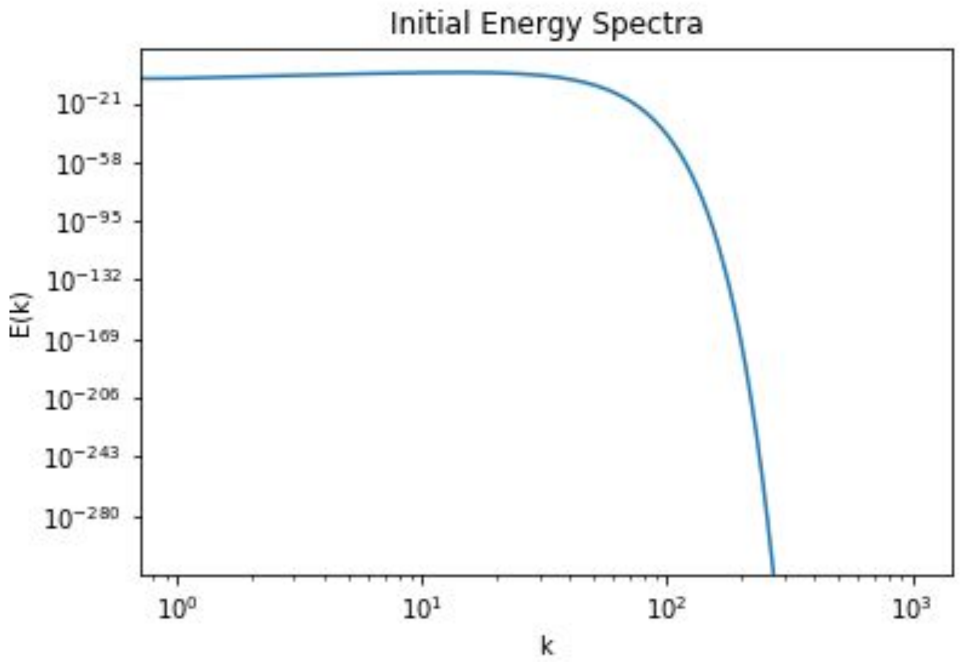
$\Psi(\mathrm{k}) \Rightarrow$ Random Number Generator: [0, 1]

4. Our velocity field is found from the inverse DFT of the Fourier domain field.

Initial Conditions

Initial velocity field





$$E(k) = Ak^4 e^{-\left(\frac{k}{k_0}\right)^2},$$

$$A = \frac{2k_0^{-5}}{3\sqrt{\pi}},\ k_0 = 10$$

Python Code Structure

Selects boundary conditions

Sets domain and creates mesh

Main Driver: Selects problem/simulation to run

Create initial field *u*

1. Defines physical problem (picks calculation of *F* and characteristic speed)
2. Define computation of parabolic (i.e. diffusion) and hyperbolic (i.e. advection) terms

Solves 1D Equations of the form:

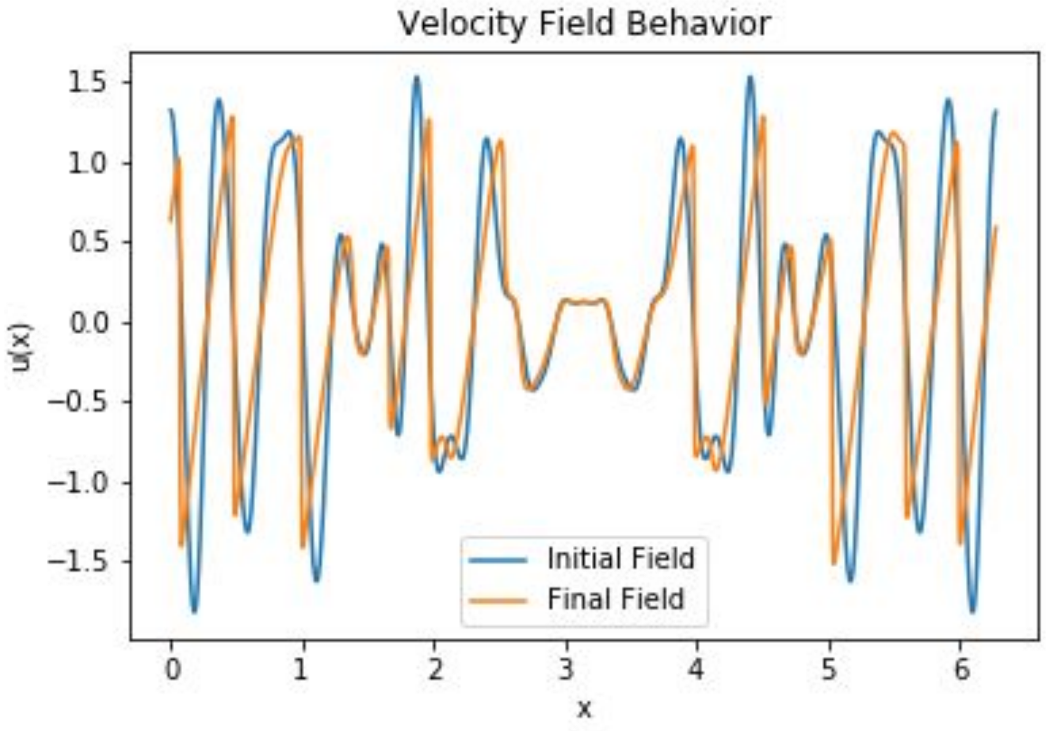
$$\frac{\partial u}{\partial t} + \frac{\partial F}{\partial x} = S$$

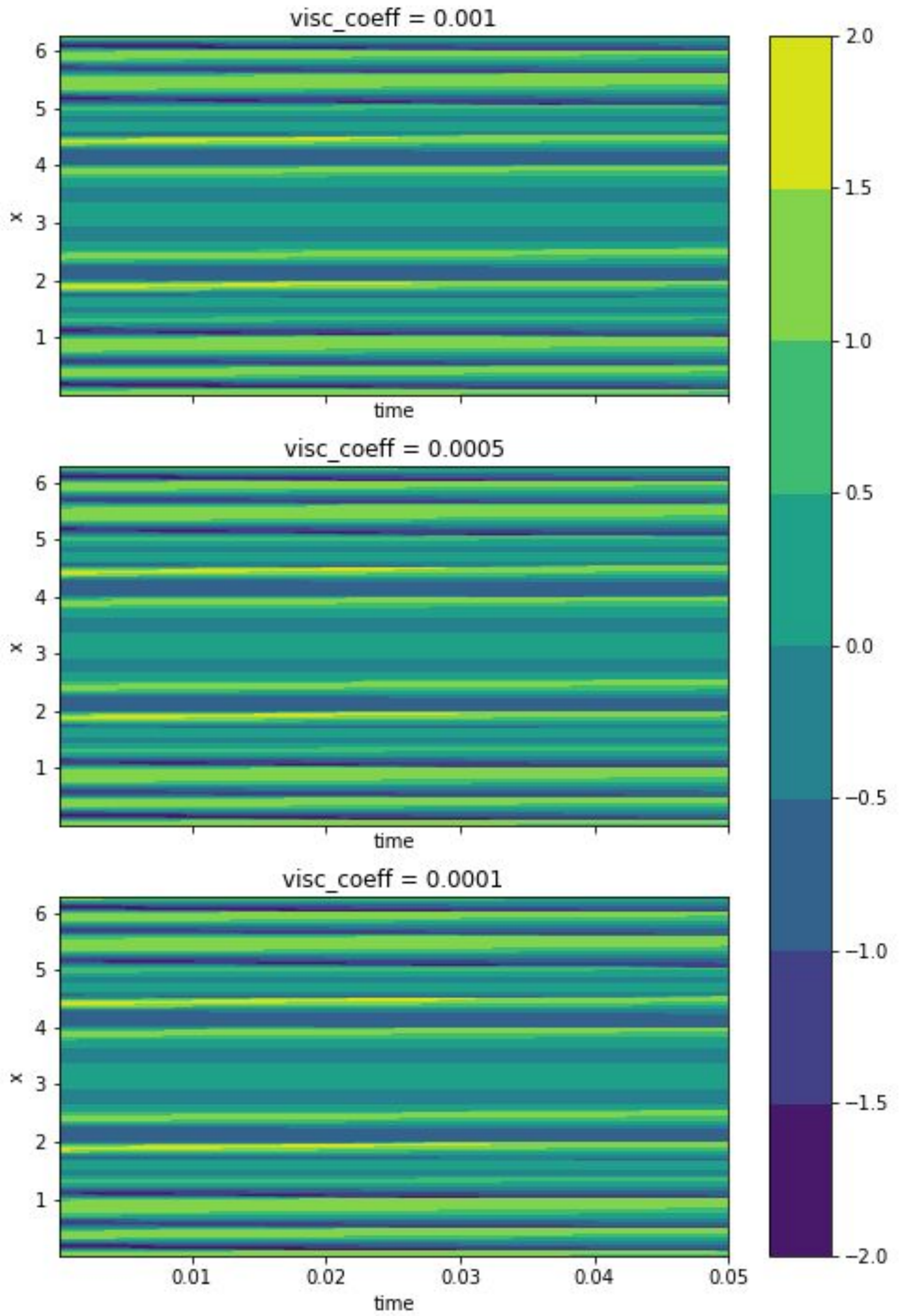
Time Integration Methods (TVD RK3)

- ArrayUtilities.py
- BCFunction_Classes.py
- Definitions.py
- Domain_Classes.py
- Driver.py
- ICSetups.py
- IOUtilities.py
- PhysicsModel_Classes.py
- PlotUtilities.py
- RFunction_Classes.py
- RWUtilities.py
- Solvers1D.py
- StatisticsUtilities.py
- TSFunction_Classes.py

Results

Evolution of the Velocity Fields:



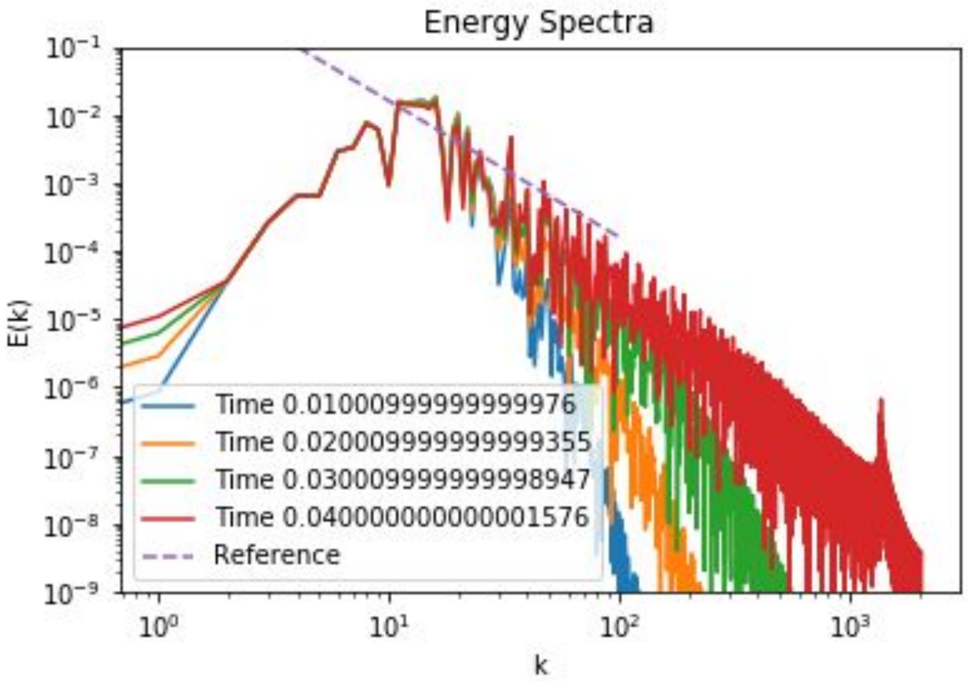


Reference Case - Spectral Methods

Parameters:

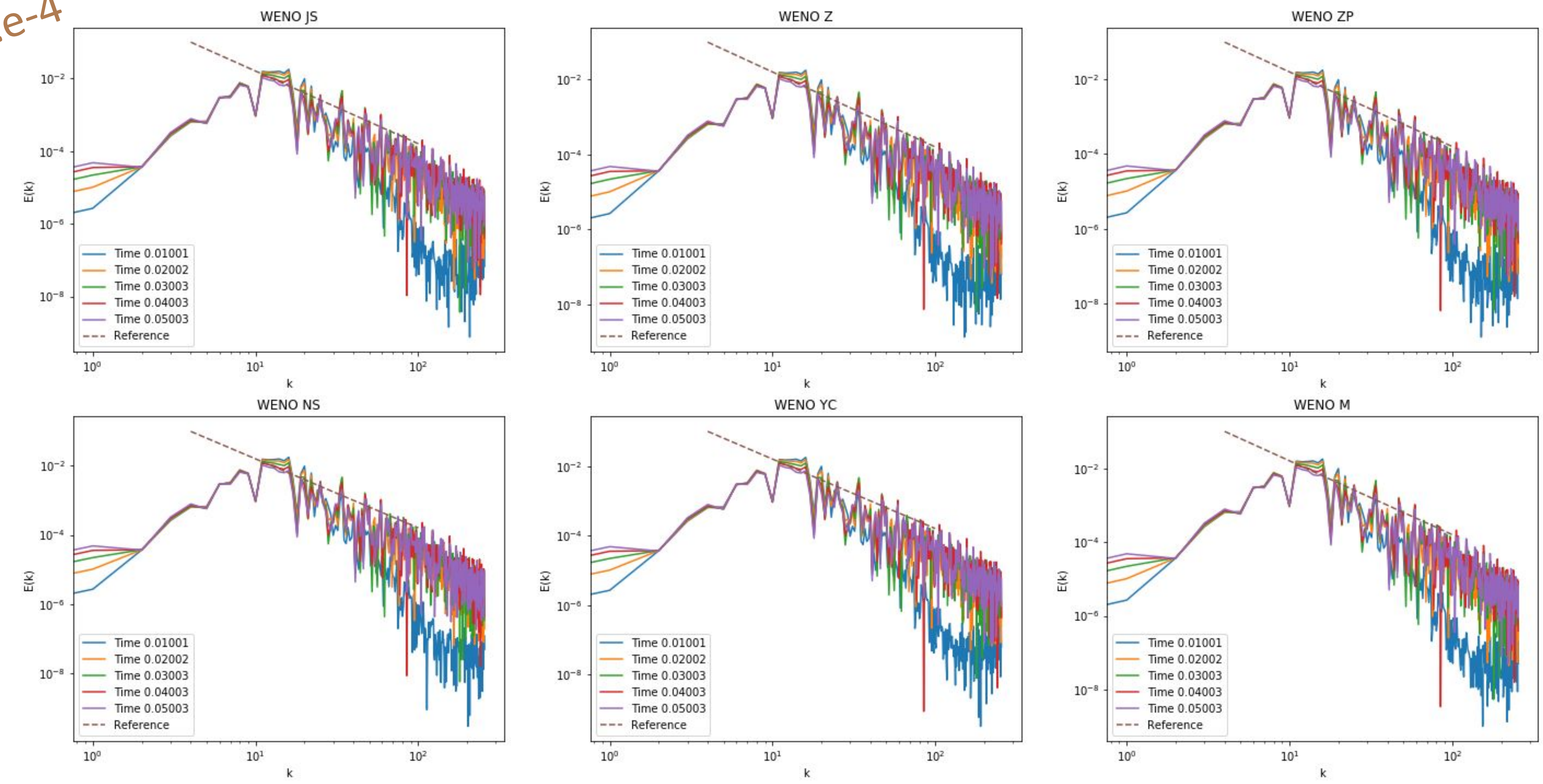
N = 4096

Coefficient of Viscosity: 5.e-4

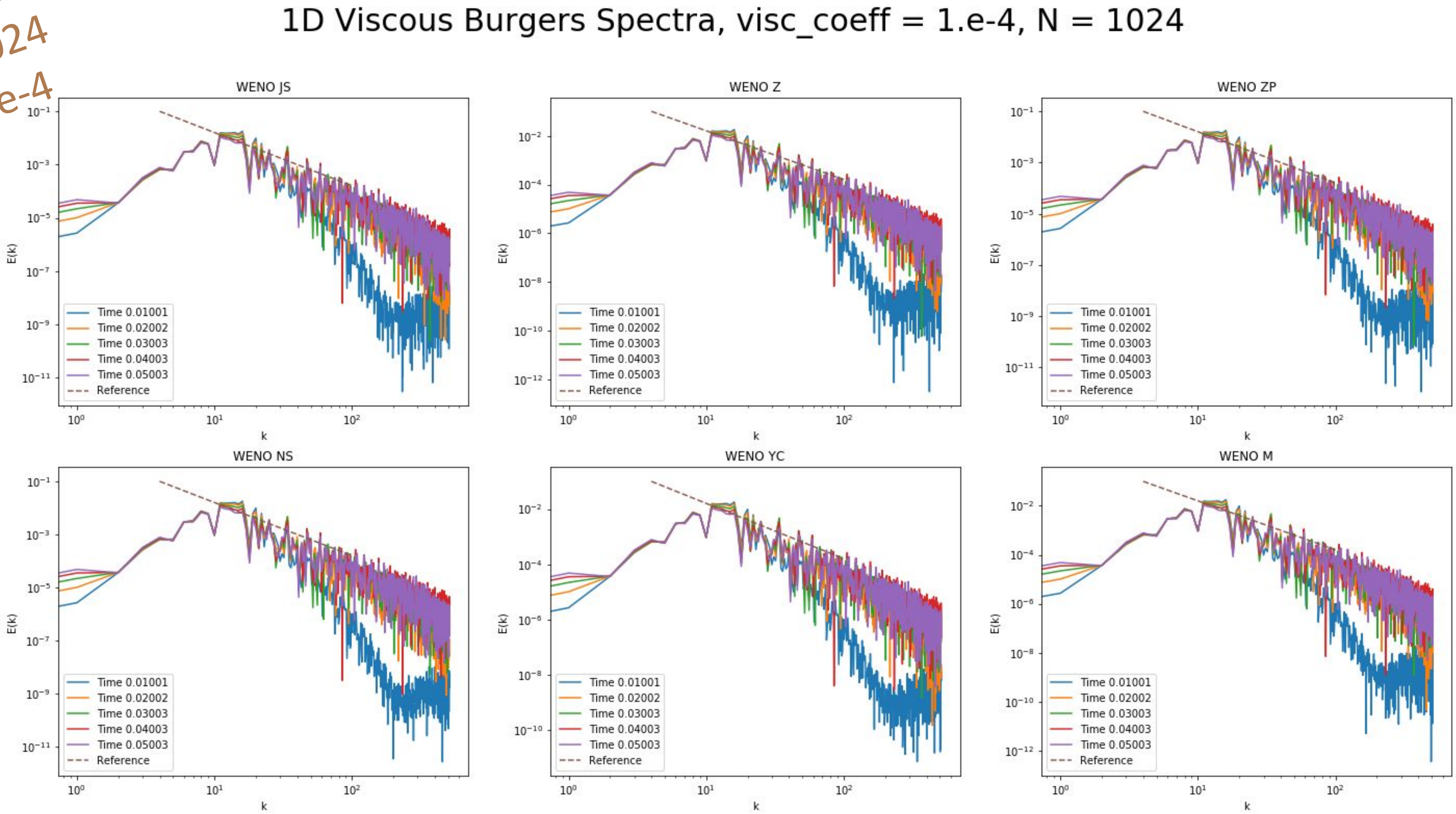


CASE 1:
N = 512
v=1.e-4

1D Viscous Burgers Spectra, visc_coeff = 1.e-4, N = 512

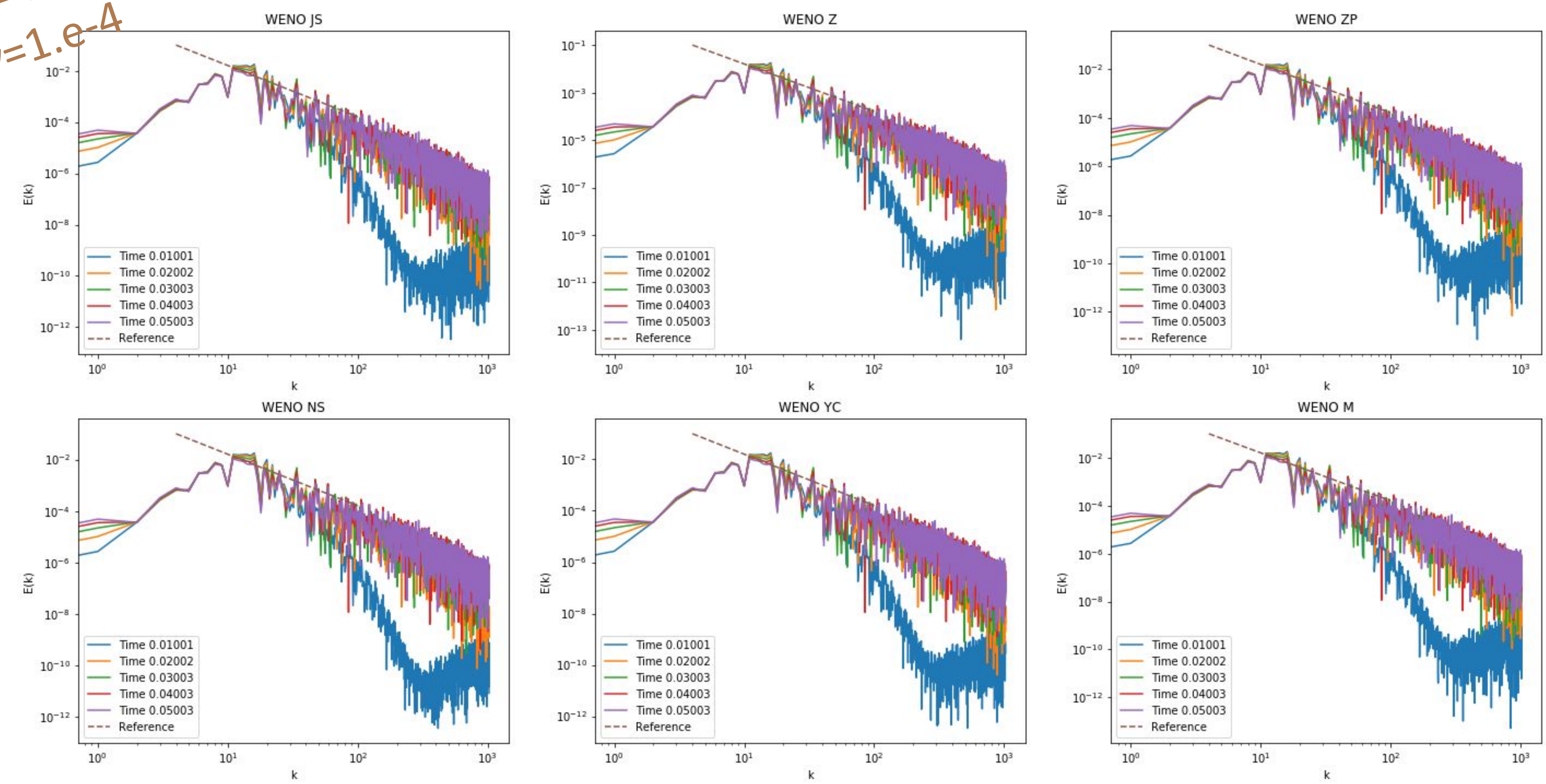


CASE 2:
N = 1024
v=1.e-4

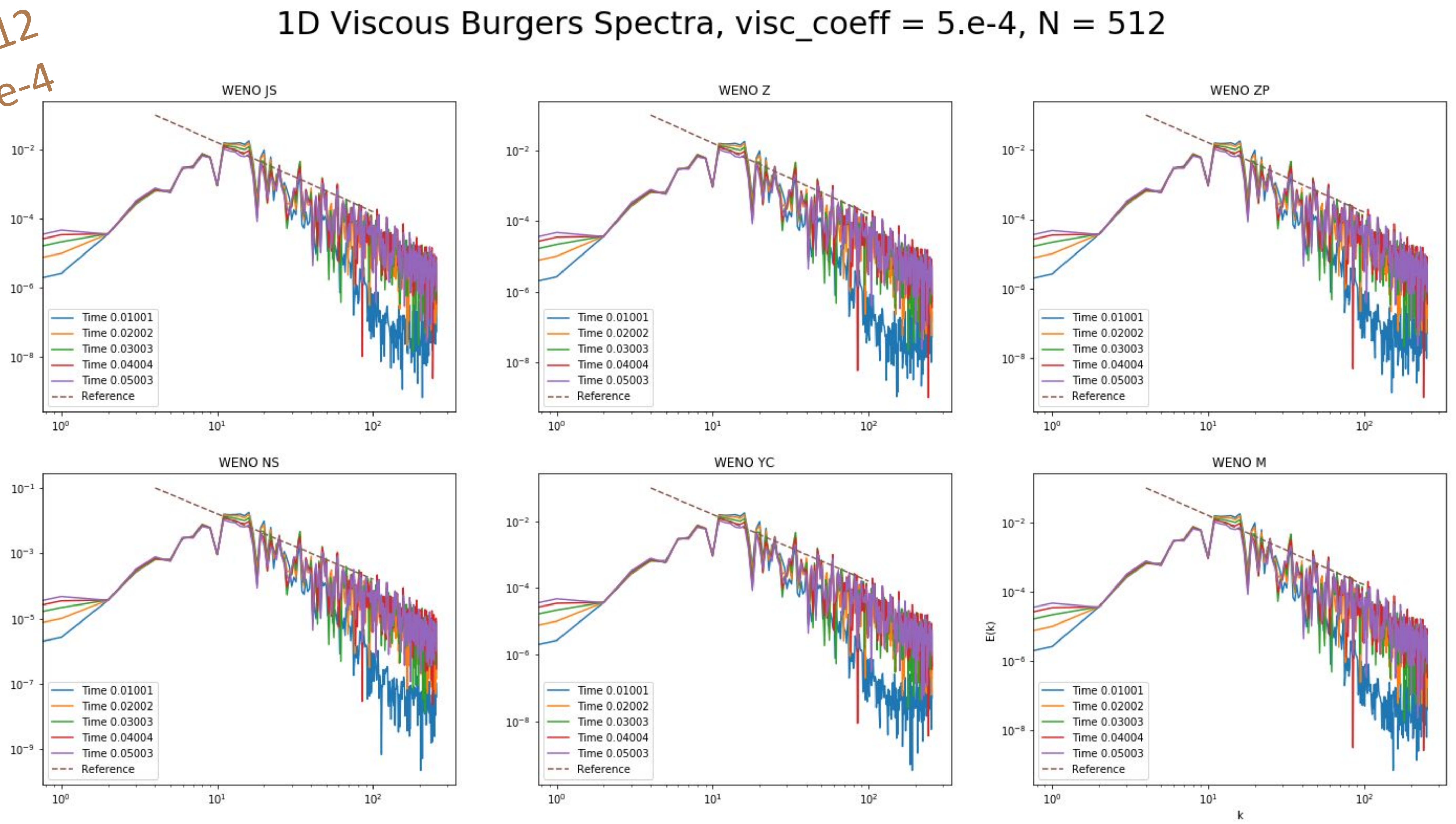


CASE 3:
N = 2048
v=1.e-4

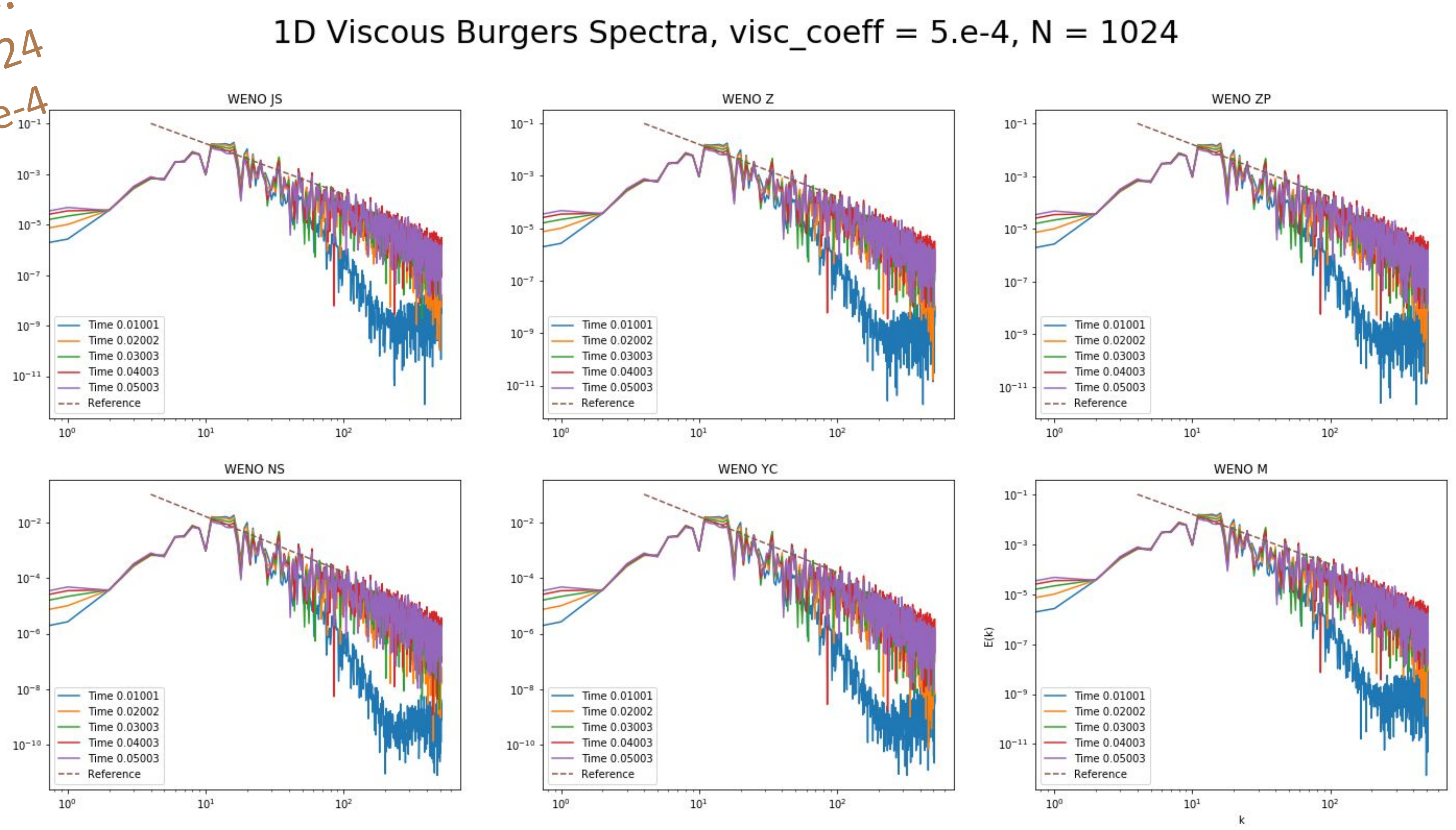
1D Viscous Burgers Spectra, visc_coeff = 1.e-4, N = 2048



CASE 4:
N = 512
v=5.e-4

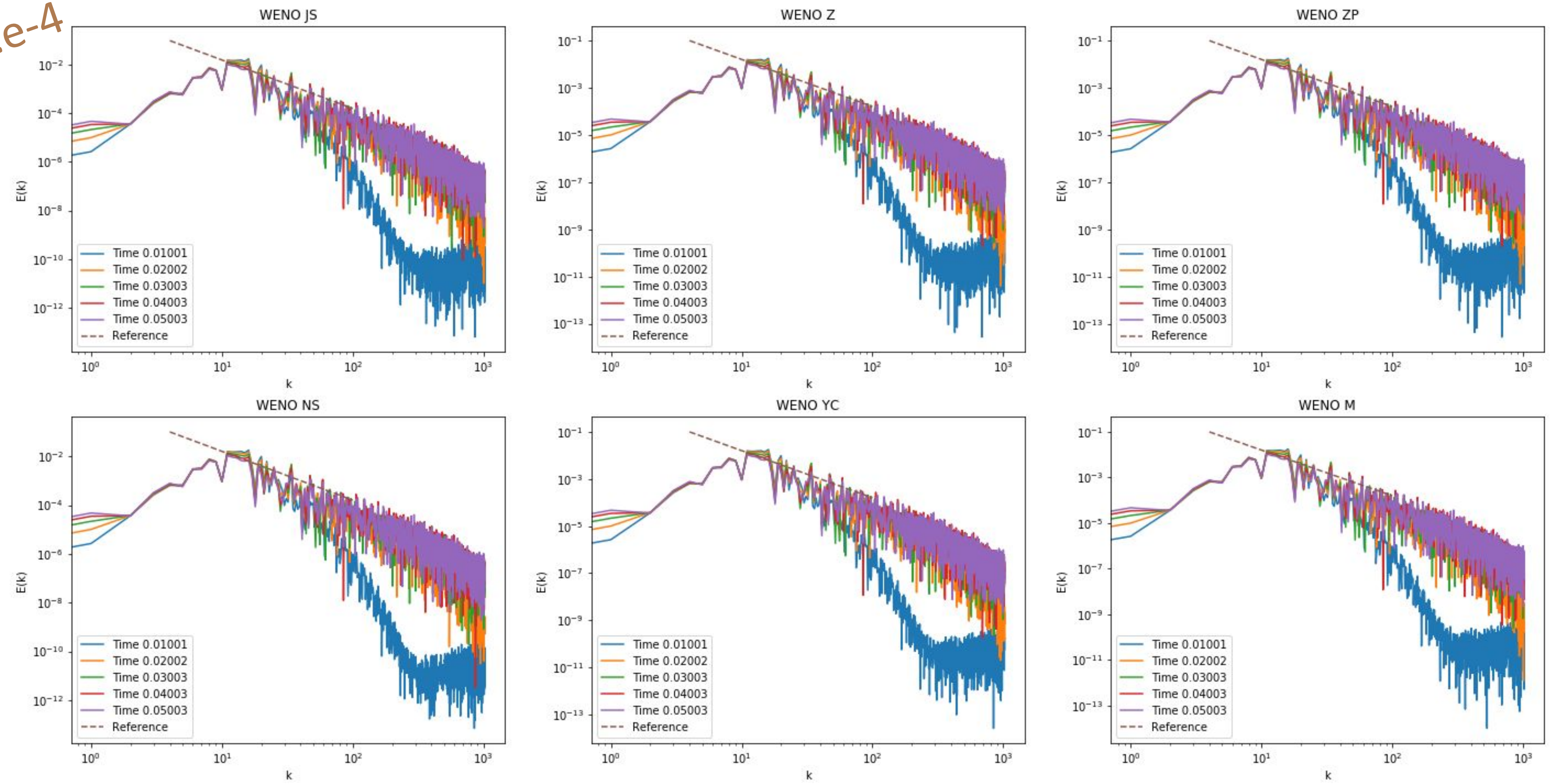


CASE 5:
N = 1024
ν=5.e-4

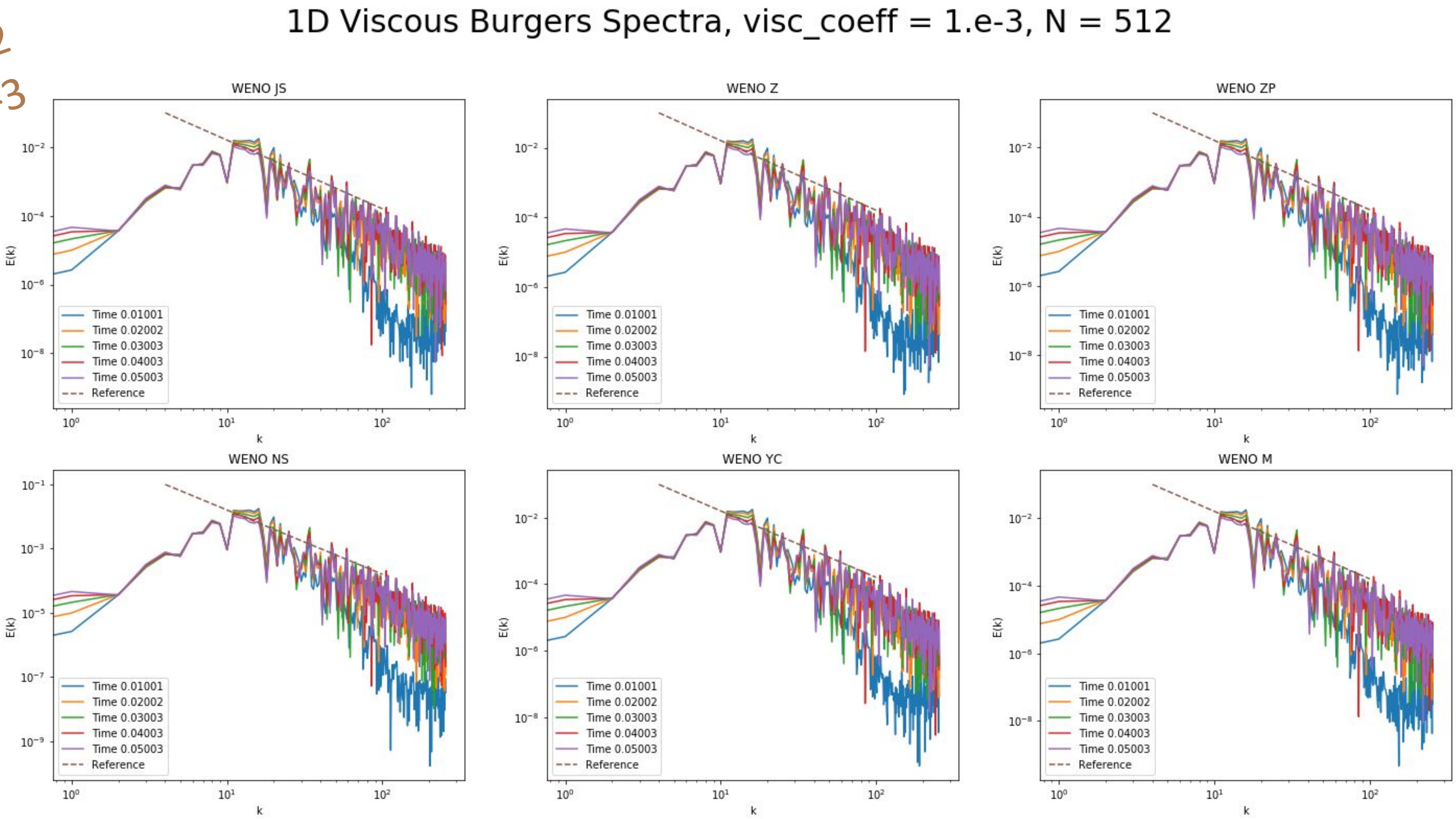


CASE 6:
N = 2048
v=5.e-4

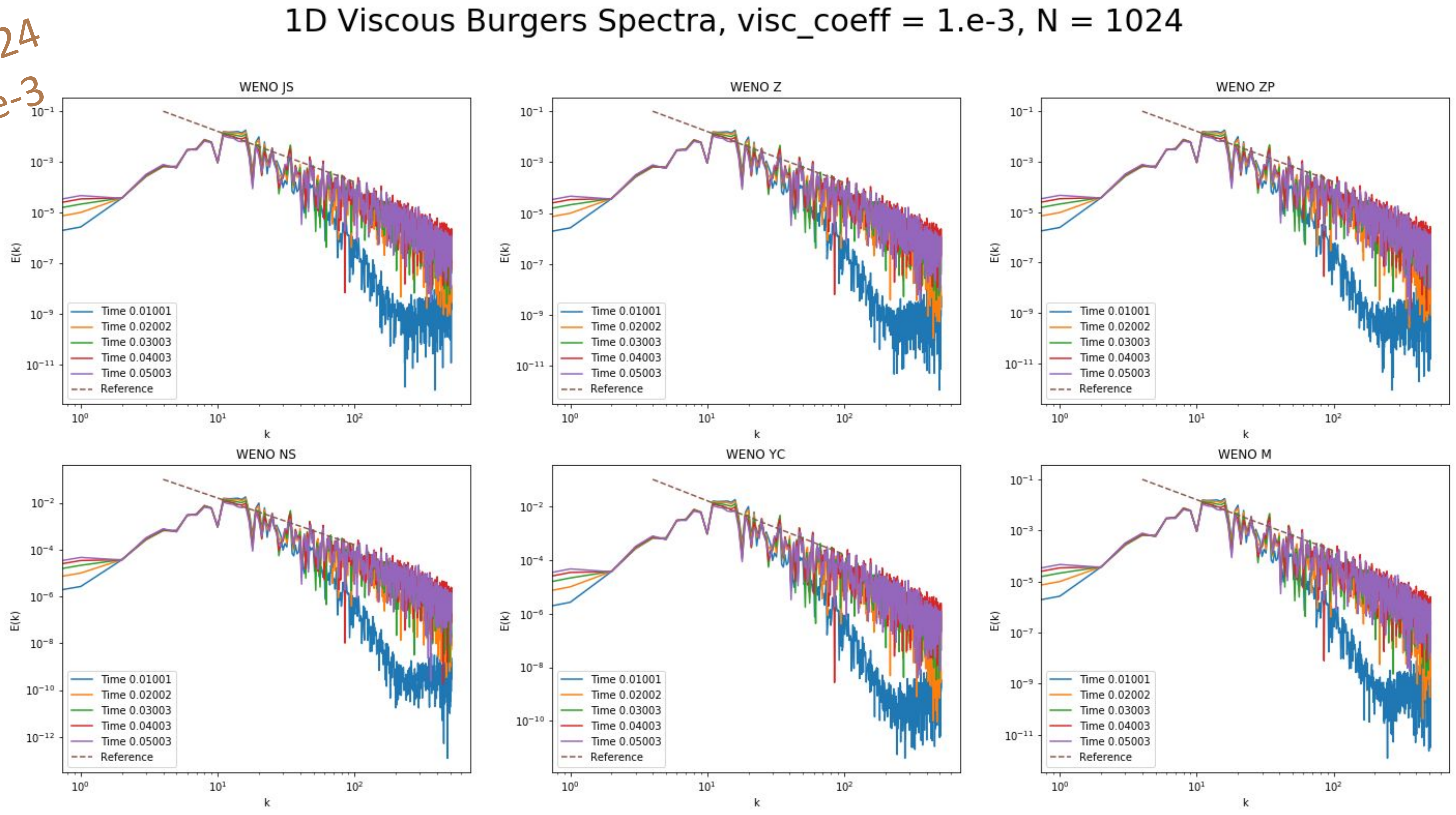
1D Viscous Burgers Spectra, visc_coeff = 5.e-4, N = 2048



# CASE 7:
N = 512
ν=1.e-3

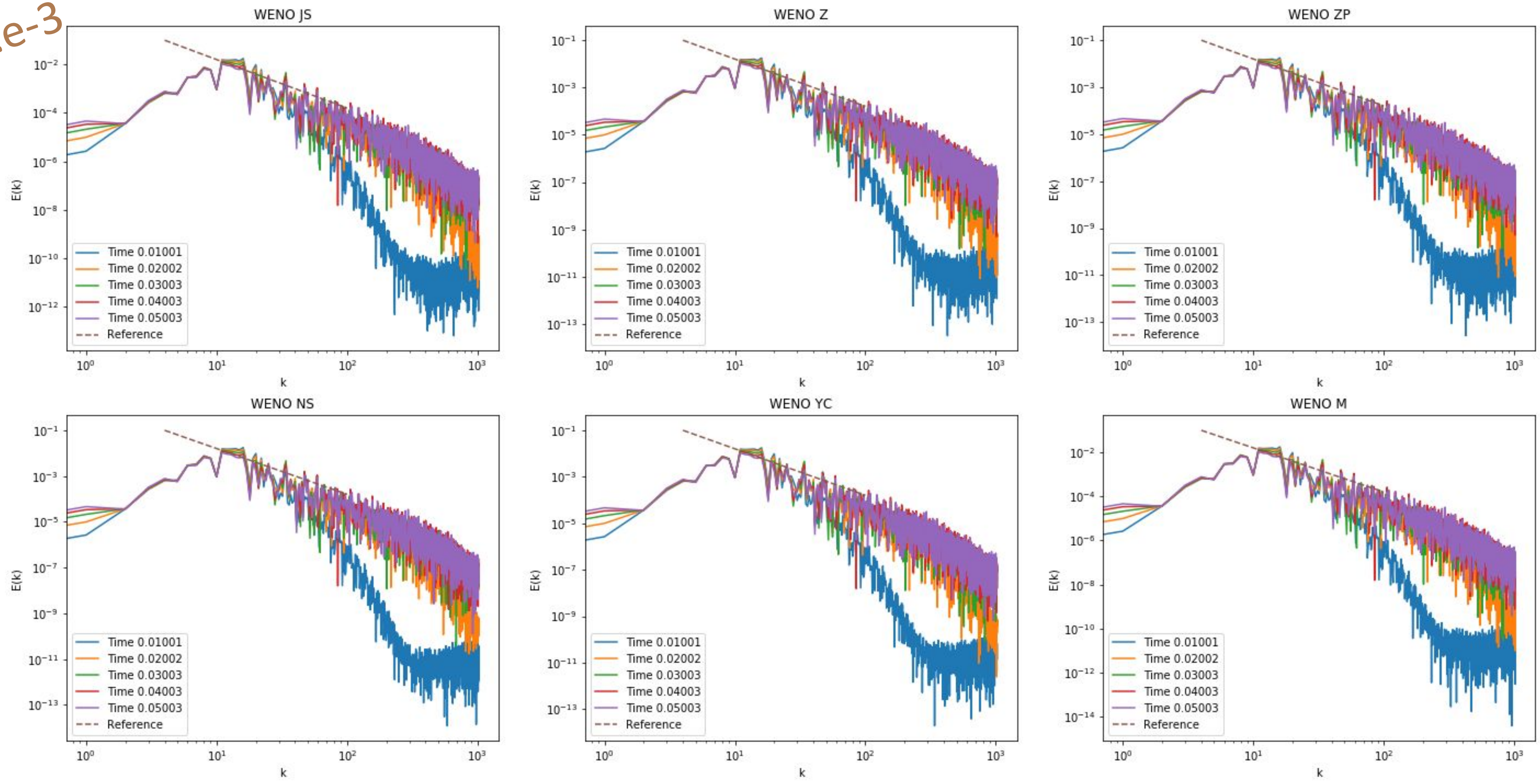


CASE 8:
N = 1024
v=1.e-3



CASE 9:
N = 2048
v=1.e-3

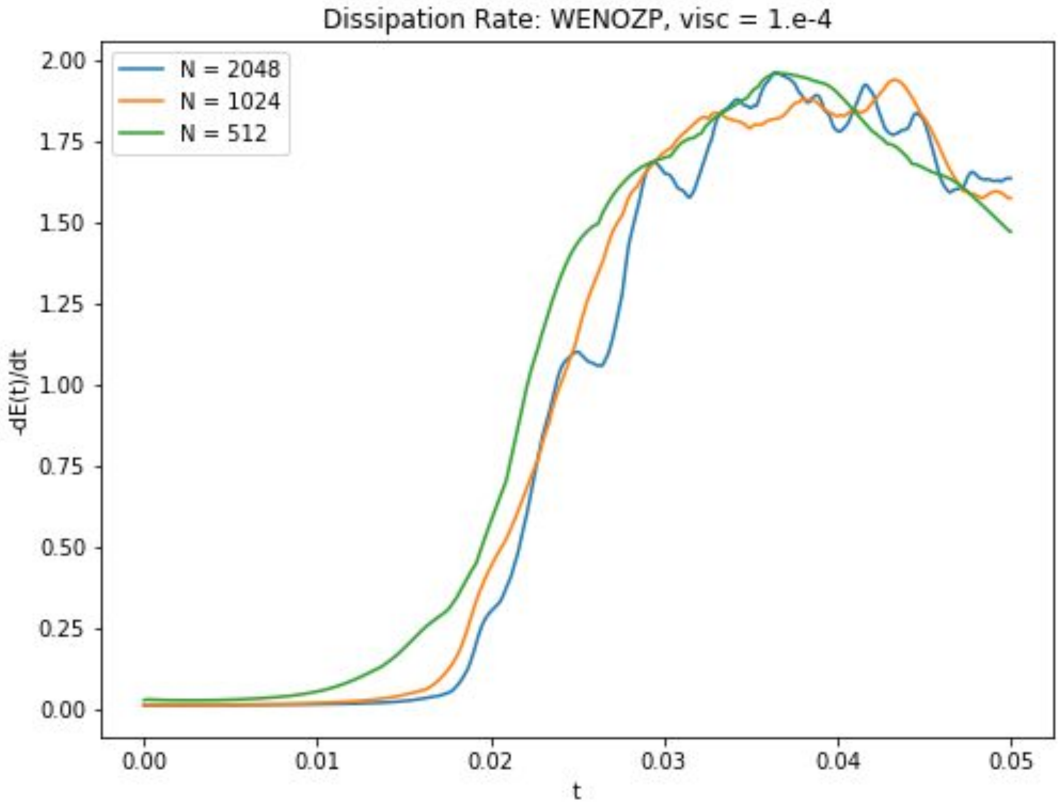
1D Viscous Burgers Spectra, visc_coeff = 1.e-3, N = 2048



Total Dissipation Rates

$$D(t) = -\frac{\partial E(t)}{\partial t},$$

$$where\ E(t) = \int_{-k_m}^{k_m} E(k)\,\mathrm{d}k$$



Conclusions & Further Work

What we've done…

- Applied ILES model with WENO to 1D Burger's Equation
- Simulation energy cascade followed the k^{-2} relation
- **Compared to spectral results, the WENO schemes performed well for ILES**

Looking ahead to the future…

- want to examine effect of ensembling
- want to examine other LES models