AEROSP 525 PDF Methods for Homogeneous Isotropic Shear Turbulence

Shivani Prabala, Vishwa Mohan Twari

April 27, 2023

Abstract: In this project, we explore probability density function based approaches (PDF methods) for modeling turbulence. We focus on understanding homogeneous isotropic turbulence with uniform shear. We begin by introducing PDF methods and building on that by exploring the Generalized Langevin Model (GLM) using the Lagrangian Approach. In the Lagrangian Approach the significance of drift and diffusion terms are studied. Then, we implemented the Eulerian Approach and compare it with results found in the Lagrangian implementation in [3].

1 Introduction

The study of turbulence is generally accompanied by statistics. In a turbulent flow the quantities such as fluctuating velocities can be understood by using statistical measures such as mean and variance. This idea is nowhere more visible than in traditional RANS models where we deal with ensemble averages of the quantities of interest. Expanding on this close knit between the Turbulence and statistics Probability Density Function(PDF) based methods were proposed for studying turbulence [5]. In these PDF models, turbulence is considered to be a stochastic process and appropriate closures are derived for the evolution equation of the PDF of the velocity components. This approach assumes turbulence to be a stochastic process and all the velocity statistics can be obtained by taking the moments on the PDF. The advantage of using these models is the ability to obtain quantities such as Reynolds stresses and triple velocity correlation $\langle u_i u_j u_k \rangle$. But these methods also require modeling [1] [2] .

The literature justifies the use of Lagrangian pdf models for modeling turbulence as the behaviour of fluid partial in turbulent flow. This gives the complete description of turbulence [1]. Further the Eulerian approaches have also been described which can model the physics of turbulence. [6]. The project explores various models for modeling turbulence using PDF based models and compares the results with the available literature.

2 Methods

2.1 Langragian Method

Background: The Langevin Equation We began with an investigation of the Langevin Equation in the context of *homogeneous isotropic turbulence*, with the following parameters:

- mean velocity = 0

- turbulence intensity, $u' = [0.05, 0.2, 0.5, 0.9, 1.3, 2]$
- Lagrangian integral time scale, $T = 1$

$U^*(t)$, is a model for one component of the fluid particle velocity, $U^+(t)$. The SDE form of the Langevin equation is:

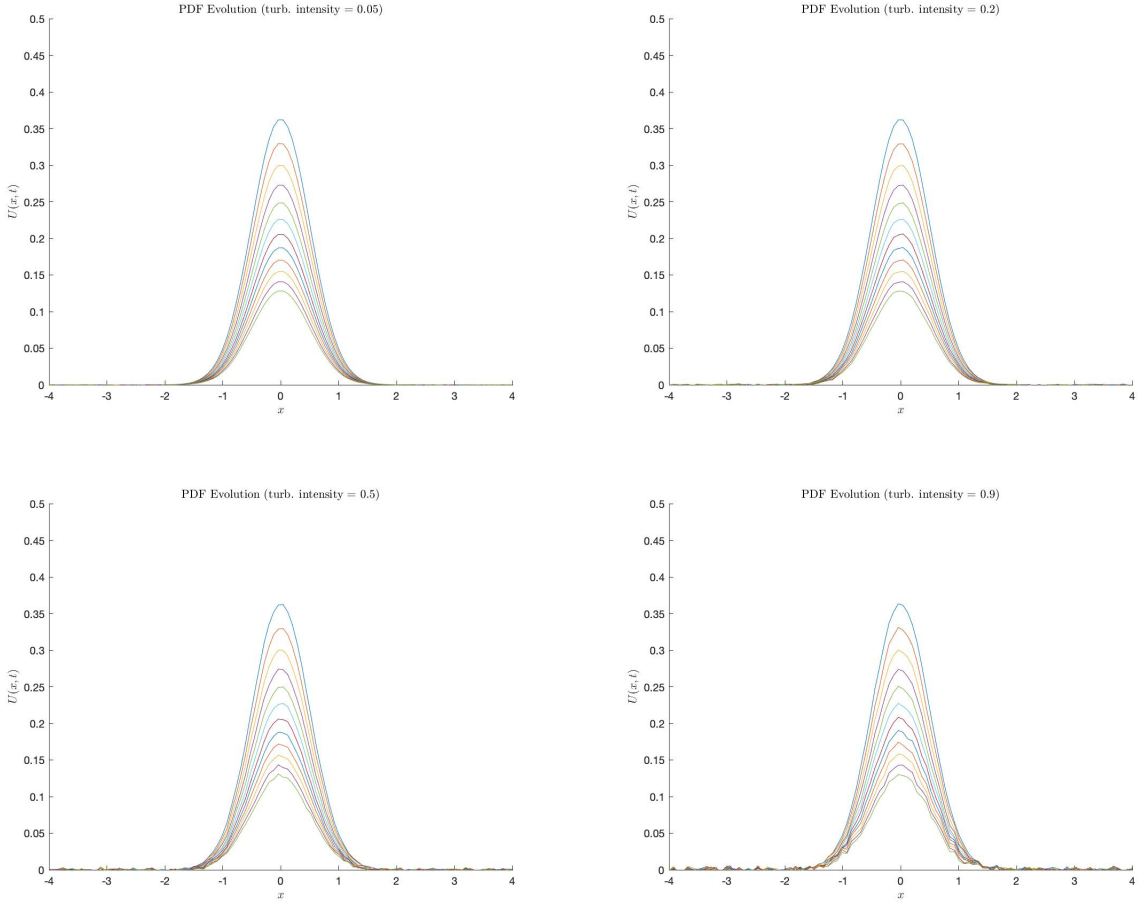
$$dU^*(t) = -U^*(t)dt/T + (2u'^2/T)^{1/2}dW(t) \tag{1}$$

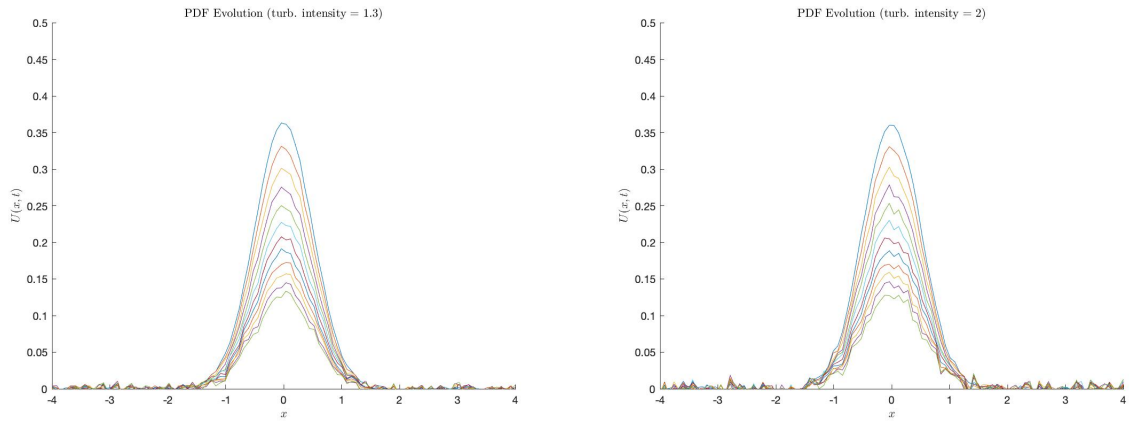
where $W(t)$ is Wiener process. To code the equation 1 we can use finite-difference approximation.

$$U^*(t + \triangle t) = U^*(t) - U^*(t)\frac{\delta t}{T} + (2u'^2\frac{\delta t}{T})^{1/2}\xi \tag{2}$$

where ξ is a standard normal Gaussian random variable ($\langle\xi\rangle = 0$, $\langle\xi^2\rangle = 1$) which is independent of the corresponding random variable on all other time steps (Markov Process). Thus, the increment in the Wiener process $dW(t)$ can be thought of as a Gaussian random variable with mean zero, and variance dt. The equation 2 is used for describing how the state evolves when subjected to deterministic and fluctuating forces.

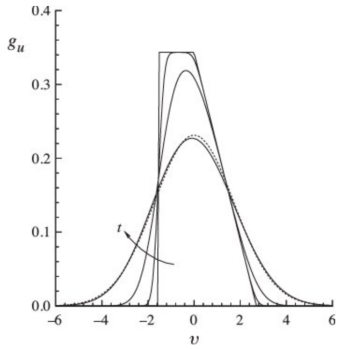
Numerical Solutions to Eqn. 2





Observations:

1. We confirm that our results show a similar diffusion profile as that depicted in Fig. 12.1 in [6].



2. Through experimentation with the turbulence intensity constant, u', we have noticed that $u' < 1$ provides a more stable PDF diffusion profile, which makes sense. The more intense the turbulence, the more likely there will be perturbations to the smoothness of the PDF, $U(x,t)$.

The Simplified Langevin Model (SLM) We implemented a version of the *simplified Langevin model* to illustrate the effect of the **drift term**. We considered the equation:

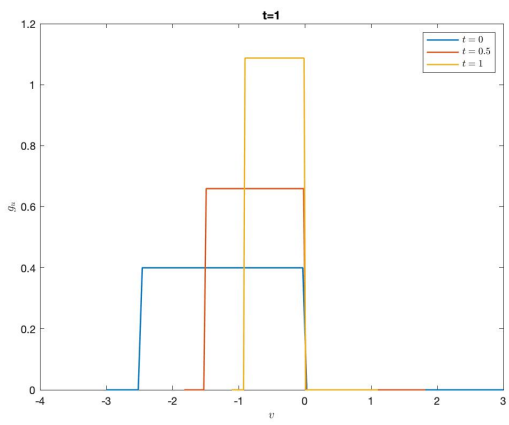
$$\frac{\partial g_u}{\partial t} = \frac{1}{T_L}\frac{\partial}{\partial v}(g_u v) \tag{3}$$

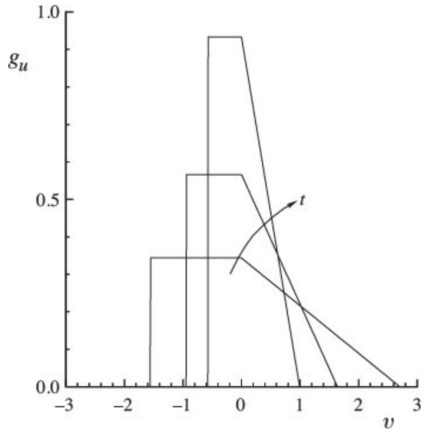
for the PDF of a zero-mean random process $u(t)$, taking T_L to be constant. The solution at time t is given in terms of the initial condition (at $t = 0$) by

$$g_u(v,t) = \frac{\sigma(0)}{\sigma(t)} g_u\Big(v\frac{\sigma(0)}{\sigma(t)}, 0\Big) \tag{4}$$

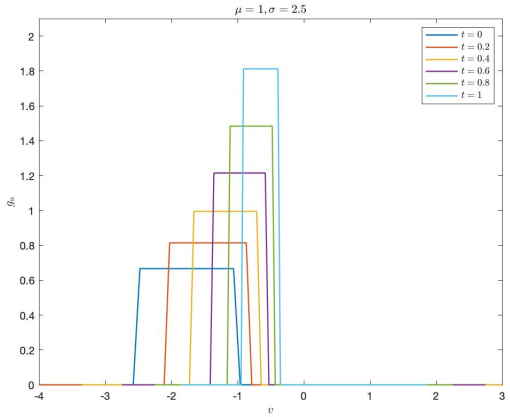
where $\sigma(t) = \sigma(0)e^{-t/T_L}$.

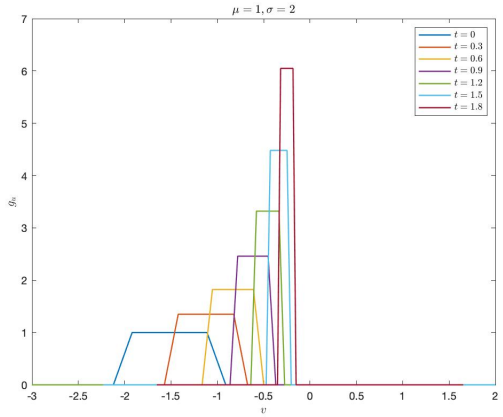
Numerical Solutions to Eqn. 2

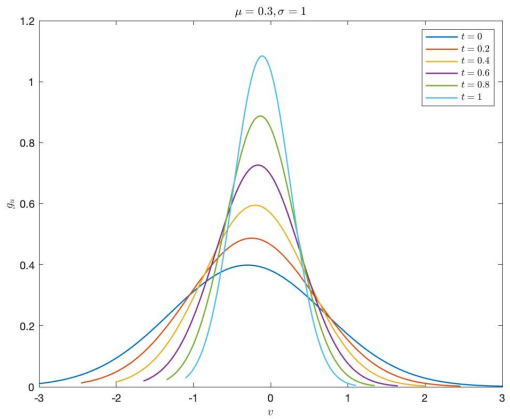


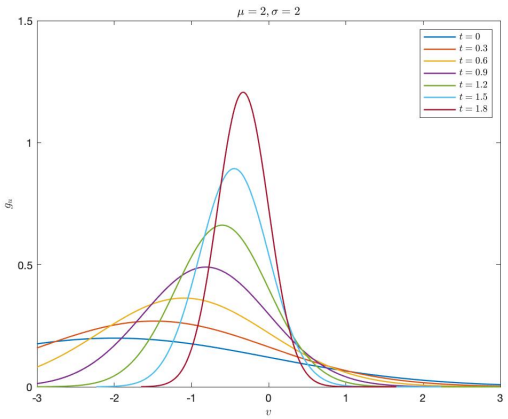


The figure on the left shows the solution to the simplified Langevin model, for a uniform PDF with mean 0, and standard deviation 2.5. We confirm out results against Figure 12.2 [6]. We plot a few more of these PDF profiles (using both uniform and normal initial PDFs):









Observations:

1. We confirm that our results show a similar drift profile as that depicted in Fig. 12.2 in [6]

2. We note that the diffusion term makes the PDF $g(\boldsymbol{v}, t)$ tend toward an isotropic joint normal, while the drift term deforms it without qualitatively affecting its shape.

3. The solution tends toward a joint normal from any initial condition, and this is the correct physical behavior since measurements in homogeneous turbulence show that the one-point PDF of velocity is joint normal.

2.2 Eulerian Approach

Upon understanding the homogeneous shear flow code provided in class, we implement the Eulerian approach to address homogeneous shear flow .

We know that an Eulerian PDF transport method focuses on a fixed spatial frame of reference. Here instead of tracing the path of each particle we solve for the PDF over the grid points. The transport equation for the one-point, one-time Eulerian PDF $f = f(\mathbf{V}; x, t)$ is shown in Eq. 5. The physics enters this Eq. 5 in the acceleration term (material derivative) $\frac{DU_i}{Dt}$. Upon substituting material derivative from Navier Stokes $-\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \nu\frac{\partial^2 u_i}{\partial x_k \partial x_k}$ and applying Reynolds averaging ($p_i = \langle P_i \rangle + p_i'$ and $u_i = \langle U_i \rangle + u_i'$) we get Eq 6.

$$\frac{\partial f}{\partial t} + V_i\frac{\partial f}{\partial x_i} = -\frac{\partial}{\partial V_i}(f\langle\frac{DU_i}{Dt}|\mathbf{V}\rangle) \tag{5}$$

$$\begin{aligned}\frac{\partial f}{\partial t} + V_i\frac{\partial f}{\partial x_i} = -\frac{\partial}{\partial V_i}(f\langle-\frac{1}{\rho}\frac{\partial(\langle P_i\rangle + p_i')}{\partial x_i} + \\ \nu\frac{\partial^2(\langle U_i\rangle + u_i')}{\partial x_k \partial x_k}|\mathbf{V}\rangle)\end{aligned} \tag{6}$$

Working with above equation one can compute $\langle U_i \rangle$ and R_{ij} by taking moments using the PDF $f(\mathbf{V}; x, t)$. Here, the $\langle P \rangle$ is obtained through solving the Poisson equation, the only unclosed terms come from the fluctuating pressure and diffusion terms. These unclosed terms are modeled using Generalized Langevin Model (GLM). Hence using this model we get Eq. 7

$$\begin{aligned}\frac{\partial f}{\partial t} + V_i\frac{\partial f}{\partial x_i} \approx -\frac{\partial f}{\partial V_i}(-\frac{1}{\rho}\frac{\partial(\langle P_i\rangle)}{\partial x_i} + \nu\frac{\partial^2(\langle U_i\rangle)}{\partial x_k \partial x_k}) \\ -\frac{\partial}{\partial V_i}[fG_{ij}(V_j - \langle U_j\rangle)] + \frac{1}{2}C_0\epsilon\frac{\partial^2 f}{\partial V_j \partial V_j}\end{aligned} \tag{7}$$

The coefficients appearing in the above equation 7 are specific to each version of GLM, an example of which is Haworth and Pope (1986).

For homogeneous shear turbulence, initialized by (HIT) with $\frac{\partial^2(\langle U_i\rangle)}{\partial x_k \partial x_k}$ is zero and considering the mean pressure gradient to be zero is all directions $\frac{\partial(\langle P_i\rangle)}{\partial x_i}$ is also zero. Hence we are left with equation 8.

$$\frac{\partial f}{\partial t} + V_i\frac{\partial f}{\partial x_i} \approx -\frac{\partial}{\partial V_i}[fG_{ij}(V_j - \langle U_j\rangle)] + \frac{1}{2}C_0\epsilon\frac{\partial^2 f}{\partial V_j \partial V_j} \tag{8}$$

Realizability Working with the Langevin model we reach the above equation which represents the pdf $f(\mathbf{V}; x, t)$, for homogeneous turbulence pdf is $f(\mathbf{V}; t)$. If appropriate initial and boundary conditions are used for $f($ $f \geq 0$ and $\int\int_{-\infty}^{\infty} f d\mathbf{V} = 1$), the Langevin equation with any finite G_{ij} causes pdf to evolve such that $f \geq 0$ and $\int\int_{-\infty}^{\infty} f d\mathbf{V} = 1$ for all time. Hence f is realizable and all moments deduced from f are realizable. [4]
A comparison can between DRSM and GLM can be drawn. For this we will take the second moment of the pdf f with fluctuating velocities.
To solve the equation 8 we need the G_{ij} as a function of local mean quantities. There have been many models proposed in the literature [7], [1] and [2] to obtain the tensor G_{ij} . We focus on these models in detail in the section below.

2.3 GLM models for Tensor Gij

The equation 8 used to model the PDF based model for Homogeneous shear turbulence (HST). It remains to determine the form of the second order tensor G_{ij}. According to the available literature the agreed hypothesis is that a sufficiently general functional form of G_{ij} is

$$G_{ij} = G_{ij}(\langle u_k u_l \rangle, \frac{\partial \langle U_p \rangle}{\partial x_q}, \epsilon) \tag{9}$$

The most general functional form of G_{ij} contains more coefficients than can be determined from the available experimental data 9. Hence we have many models for G_{ij} and C_o .

$$G_{ij} = \frac{\epsilon}{k}(\alpha_1 \delta_{ij} + \alpha_2 b_{ij} + \alpha_3 b_{ij}^2) + H_{ijk} \frac{\partial \langle U_k \rangle}{\partial x_l} \tag{10}$$

$$H_{ijk} = \beta_1 \delta_{ij}\delta_{kl} + \beta_2 \delta_{ik}\delta_{jl} + \beta_3 \delta_{il}\delta_{jk} + \gamma_1 \delta_{ij} b_{kl} + \gamma_2 \delta_{ik} b_{jl} + \gamma_3 \delta_{il} b_{jk} + \gamma_4 b_{ij}\delta_{kl}\gamma_5 b_{ik}\delta_{jl} + \gamma_6 b_{il}\delta_{jk} \tag{11}$$

Using the general model proposed in equation 10 and 11, we can obtain for our pdf based models by knowing the 12 coefficients that appear. For the present project we have used four models, namely SLM, LIPM, HP1 and HP2 [1].

Table 1: Model coefficient table for Gij

Model	α_1	α_2	α_3	β_1	β_2	β_3	γ_1	γ_2	γ_3	γ_4	γ_5	γ_6
SLM	$-(\frac{1}{2} + \frac{3}{4}C_o)$	0	0	0	0	0	0	0	0	0	0	0
HP1	Eq. 12	3.7	0	-1/5	4/5	-1/5	0	3.01	-2.18	0	4.29	-3.09
HP2	Eq. 12	3.78	0	-1/5	4/5	-1/5	0	1.04	0.34	0	1.99	-0.76
LIPM	Eq. 16	3.5	-10.5	-1/5	4/5	-1/5	0	0	0	0	0.6	-0.6

The coefficient α_1 is crucial which is calculated based on the equations mentioned in the table.

$$\alpha_1 = -(\frac{1}{2} + \frac{3}{4}C_o + b_{ii}^2(\alpha_2 + \frac{1}{3}\alpha_3) + b_{ii}^3\alpha_3 + (\beta_2 + \beta_3 + \frac{1}{3}\gamma^*)\boldsymbol{I}_1 + \gamma^* I_2) \tag{12}$$

$$\gamma^* = \gamma_2 + \gamma_3 + \gamma_5 + \gamma_6 \tag{13}$$

$$\boldsymbol{I}_1 = b_{ij} S_{ij} \tag{14}$$

$$\boldsymbol{I}_2 = b_{ij}^2 S_{ij} \tag{15}$$

$$\alpha_1 = -(\frac{1}{2} + \frac{3}{4}C_o - \frac{1}{2}C_2\frac{P}{\epsilon} - 3\alpha_2 b_{ii}^2) \tag{16}$$

For homogeneous turbulence the term containing C_o causes an arbitrary initial pdf to relax to a joint normal distribution.

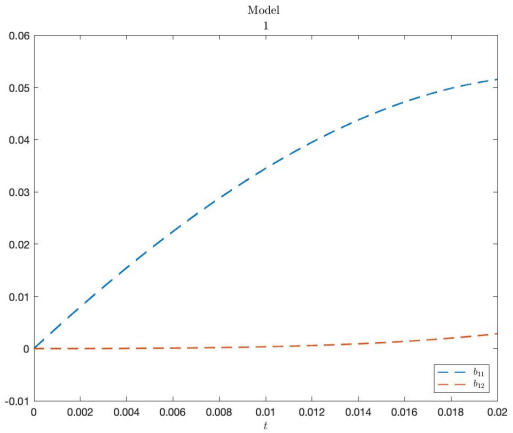
Algorithm 1 Algorithm for Eulerian PDF model

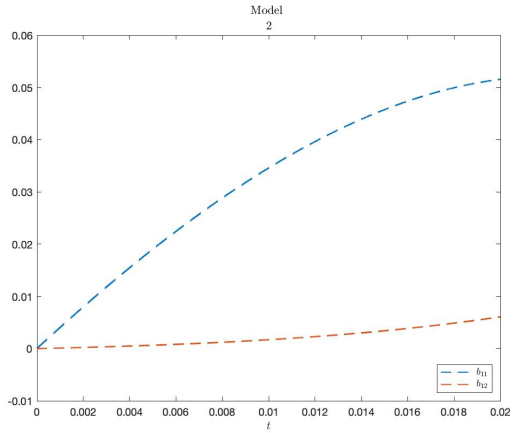
Initialize the grid and constants
for $t \to T$ **do**
  Evaluate G_{ij}
  for over the grid point(keeping account of boundary conditions) **do**
    Evaluate the derivatives
  end for
  Update step using Forward Euler or RK 2
  Evaluate C_o using production and dissipation
  Update $\mathbf{U}_{mean}$
  Get the $RIJ \to Prod \to \epsilon \to tke \to G_{ij}$ in the same order $\triangleright\ tke_{new} = tke + dt(Prod - \epsilon)$

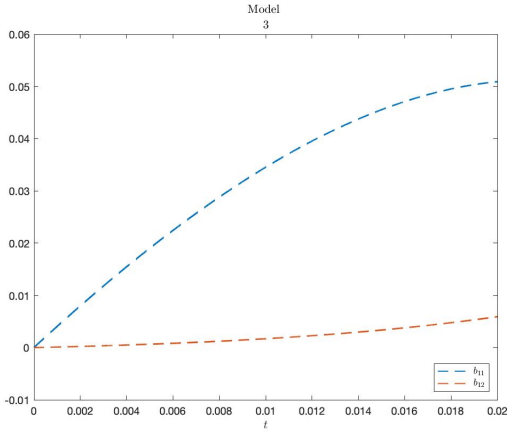
3 Results

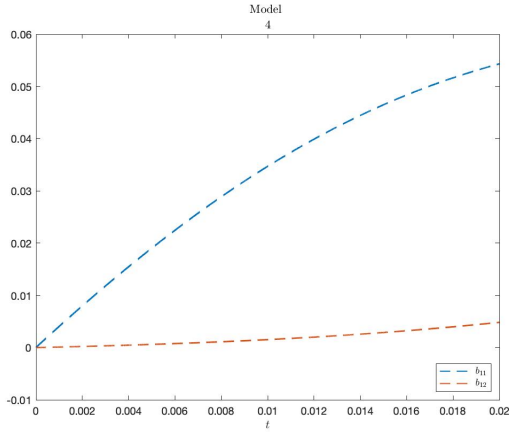
3.1 Forward Euler Time Discretization

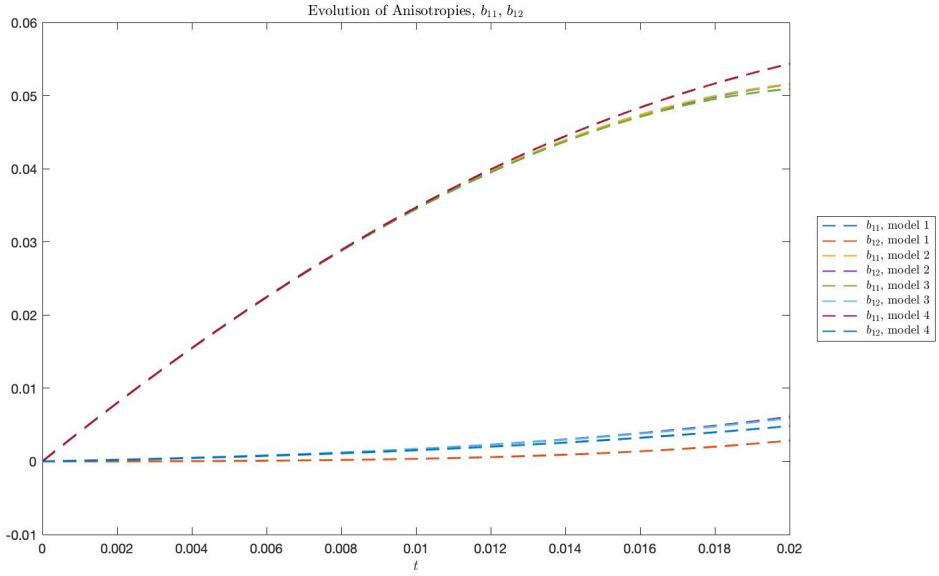
In our initial approach, we used a first order method for time-stepping. A simple forward euler time step discretization was employed where every value of the pdf $f(V1, V2, V3, t)$ at consecutive time steps, depended only on the values at the time step immediately before. Initializing our pdf with $\mu = \begin{bmatrix} 5 \times 10^{-10} \\ 5 \times 10^{-10} \\ 5 \times 10^{-10} \end{bmatrix}$ and $\Sigma = \begin{bmatrix} 0.2 & 0 & 0 \\ 0 & 0.2 & 0 \\ 0 & 0 & 0.2 \end{bmatrix}$, and a $dt = 2 \times 10^{-6}$ we had the following results:

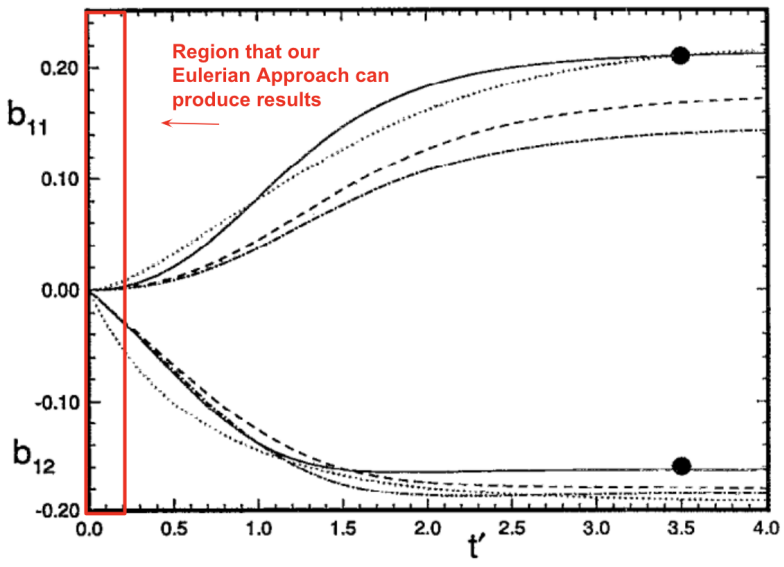





Model
3
b_{11}
b_{12}
t


Model
4
b_{11}
b_{12}
t


Evolution of Anisotropies, b_{11}, b_{12}
b_{11}, model 1
b_{12}, model 1
b_{11}, model 2
b_{12}, model 2
b_{11}, model 3
b_{12}, model 3
b_{11}, model 4
b_{12}, model 4
t


Region that our Eulerian Approach can produce results
b_{11}
b_{12}
t'

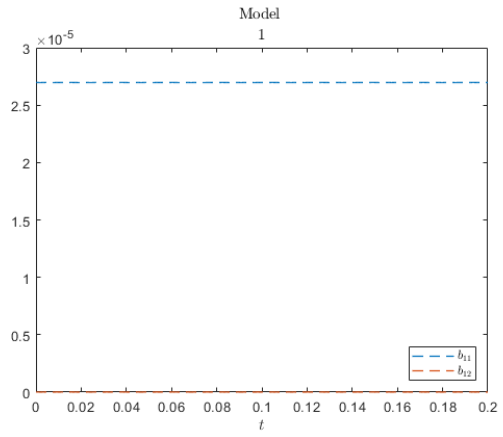
Observations: We noticed through exploration of parameters such as μ, σ, dt, and T that our results would change quite drastically with changes in the pdf initialization. We also noticed that in this scheme, we were only able to march forward for very small T, despite having fairly small dt, just up to $T = 0.02$ in most cases even with $dt = 2 \times 10^{-6}$.

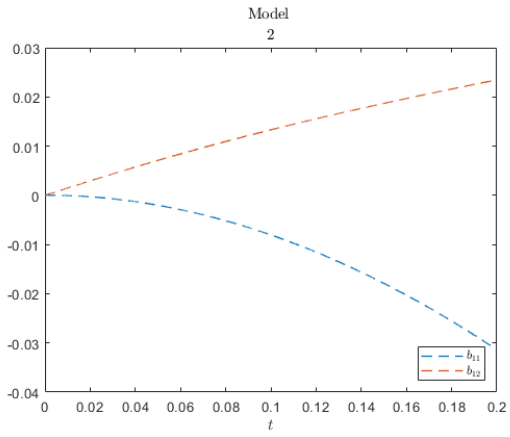
Additionally, the results from literature [3] show the theoretical result for the evolution of b_{11} is expected to increase from its initial values at $t = 0$, and that b_{12} is expected to decrease with somewhat symmetric shapes that have a steep incline or decline followed by a plateau section. Although, we were not able to resolve the solution for latter time steps to compare for this plateau phase, we observed an increasing trend in our solutions for b_{11} which was encouraging. b_{12} on the other hand matched the theretical results less accurately. We expected to find negative values or a decreasing trend which was not the case.

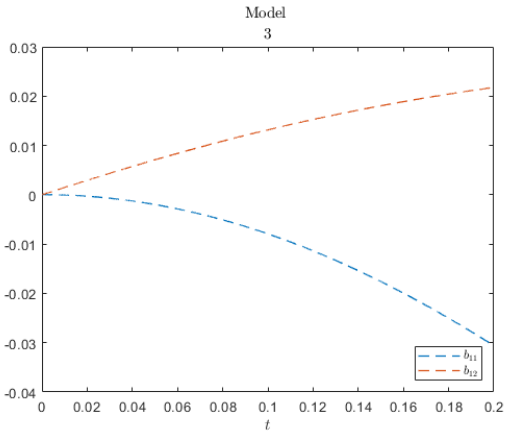
After thorough reviews of the existing algorithm we had written, we hypothesized that perhaps the degeneration of the b_{12} solution was caused by stability issues in our differencing scheme. In particular, the fact that we were using a first order time stepping method with second order spatial discretizations. We then attempted to implement a higher order time stepping discretization, which we will discuss in the next subsection.

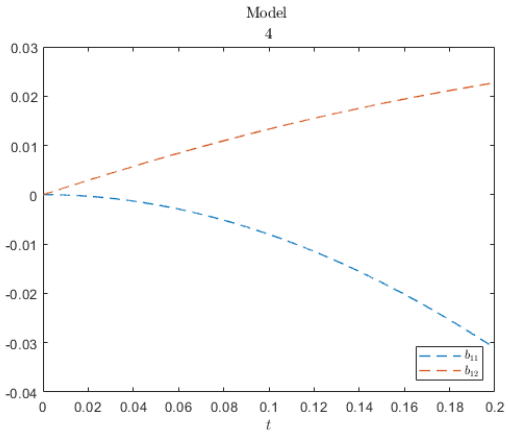
3.2 Runge Kutta 2 (Midpoint Method) Time Discretization

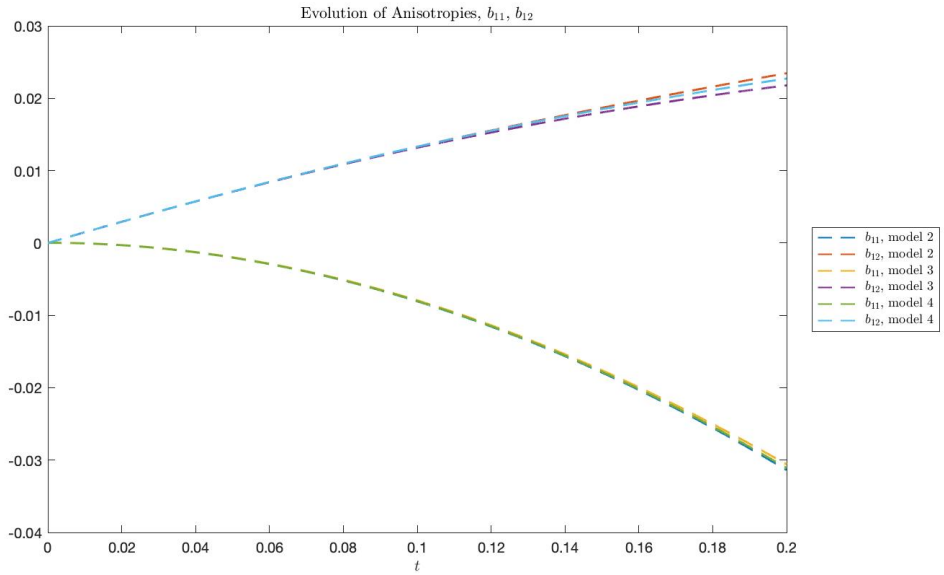
Since the initial implementation used a second order central difference inside the domain and a second order one-sided difference at the boundaries of the pdf, we hypothesized that use of forward euler time integration could be a source of instability. To solve this issue, we implemented a time integration using RK 2 (second order explicit scheme). The following results show values from the anisotropy tensor b_{ij} vs t for turbulent flow initialized as HIT with homogeneous shear being solved for using various GLM models. The initialized pdf has $\mu = \begin{bmatrix} 5 \times 10^{-10} \\ 5 \times 10^{-10} \\ 5 \times 10^{-10} \end{bmatrix}$ and $\Sigma = \begin{bmatrix} 0.2 & 0 & 0 \\ 0 & 0.2 & 0 \\ 0 & 0 & 0.2 \end{bmatrix}$, and a $dt = 2 \times 10^{-6}$ we had the following results:

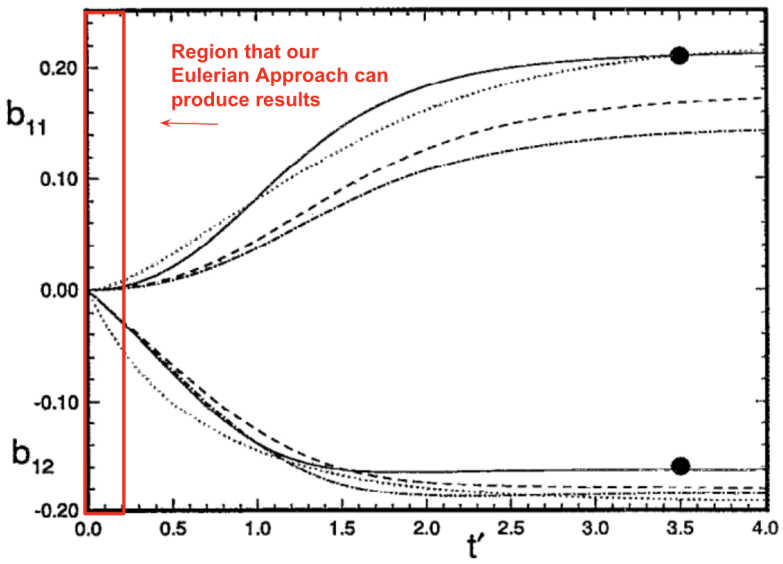





Model
3
0.03
0.02
0.01
0
-0.01
-0.02
-0.03
-0.04
0 0.02 0.04 0.06 0.08 0.1 0.12 0.14 0.16 0.18 0.2
t
b_{11}
b_{12}


Model
4
0.03
0.02
0.01
0
-0.01
-0.02
-0.03
-0.04
0 0.02 0.04 0.06 0.08 0.1 0.12 0.14 0.16 0.18 0.2
t
b_{11}
b_{12}


Evolution of Anisotropies, b_{11}, b_{12}
0.03
0.02
0.01
0
-0.01
-0.02
-0.03
-0.04
0 0.02 0.04 0.06 0.08 0.1 0.12 0.14 0.16 0.18 0.2
t
b_{11}, model 2
b_{12}, model 2
b_{11}, model 3
b_{12}, model 3
b_{11}, model 4
b_{12}, model 4


Region that our
Eulerian Approach can
produce results
b_{11}
b_{12}
0.20
0.10
0.00
-0.10
-0.20
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0
t'

*Note: In the above figure, model 1 has been excluded due to order of magnitude differences in the results.

Observations: In this RK2 time stepping implementation, we see some improvements in the outputted solutions. For one, we are able to go to a much farther T. In the Forward Euler approach we were only able to go to $T = 0.02$, however, the RK2 approach enables us to reach $T = 0.2$ with the same dt, likely due to the improved second order accuracy in time. We also see that b_{12} is decreasing which aligns more closely with the theoretical expectation for the evolution of these anisotropy values, another improvement on our initial Forward Euler implementation.

With this new approach, however, we notice an abnormal blow-up in the solution for b_{11}, and b_{12} in the first plot corresponding to model 1. This is a point for further investigation as the only notable difference in the code are the model parameters corresponding to the G_{ij} matrix. Focusing on models 2,3, and 4, we observe better accuracy with the theoretical evolution of the anisotropies using a Runge Kutta 2 second order accurate time discretizaton.

4 Limitations

1. **Inaccurate Anisotropy Evolution :** In the solutions shown above, we see that for the Forward Euler time discretization, b_{12}, while at first negative, eventually begins increasing and this does not follow theoretical results [3]. We hypothesized that this was due to our time discretization and proceeded to implement RK2. In the RK2 implemntation, however, we saw blow-up with using model 1. The reason for this is still unclear and requires further investigation and debugging.

2. **Limited Resolution of Solution in t:** The main limitations with both of our approaches are that we are only able to march forward in time for a relatively small T, 0.02 with Forward Euler, and 0.2 with Runge Kutta 2. We hypothesize that this is due to our choice of discretization and would need to do a detailed review of numerical differentiation theory to find possible issues with our current implementation in terms of stability and consistency of the schemes.

3. **The ambiguity in update** The available literature provides multiple ways to update ϵ and TKE. We did the literature review for ϵ update and found 17 and 18. Further the production ($Prod$) can be updated in two ways 19 and 20

$$\epsilon = \epsilon + dt((\frac{\epsilon^2}{k})(C_{\epsilon 1}(1.6) - C_{\epsilon 2})) \tag{17}$$

$$\epsilon = Prod/(1.6) \tag{18}$$

$$Prod = -\langle u_1 u_2 \rangle \frac{\partial U}{\partial y} \tag{19}$$

$$Prod = -(2b(i,j)S_{ij}(i,j)) \tag{20}$$

4. **Initialization of the PDF** The initialization of the pdf impacts the results drastically, hence to get the results similar to the literature we would need the HIT PDF data (μ and Σ).

5 Conclusions

In this problem, we worked on developing code that could, through time-stepping, evolve a multivariate pdf in $\{V1, V2, V3\}$ space. We did so through the PDF based method known as the Eulerian Approach where we initialized a pdf corresponding to a physical system that has homogeneous isotropic turbulence with uniform shear. We then updated the PDF based on various models for the G_{ij} matrix in the GLM model at every time step. We use the aforementioned pdf, at each time step, to calculate the values, b_{11}, and b_{12} of the anisotropy tensor and tried to match this against theoretical results found using the Lagrangian Approach [3]. Initially, we used a first order, Forward Euler time discretization, and produced results that were uniform across model, but found that for b_{12} the solution was not following the expected trend [3]. We hypothesized that perhaps our solution was inaccurate due to stability issues in our discretization method. We then implemented an RK2 time stepping scheme, which gave solutions that show better behavior for b_{12}, but failed to provide consistent results across model. To improve upon these issues, one can take this project further by (1) investigating issues in the current time-space discretization scheme, (2) understanding the best update models for TKE, ϵ, and P, and (3) applying this overall method to other cases of turbulent flow.

References

[1] Daniel C Haworth and Stephen B Pope. A generalized langevin model for turbulent flows. *The Physics of fluids*, 29(2):387–405, 1986.

[2] DC Haworth and SB Pope. A pdf modeling study of self-similar turbulent free shear flows. *The Physics of fluids*, 30(4):1026–1044, 1987.

[3] S. B. Pope. On the relationship between stochastic lagrangian models of turbulence and second-moment closures. *Physics of Fluids*, 6(2):973–985, 1994.

[4] Stephen B Pope. Pdf methods for turbulent reactive flows. *Progress in energy and combustion science*, 11(2):119–192, 1985.

[5] Stephen B Pope. Lagrangian pdf methods for turbulent flows. *Annual review of fluid mechanics*, 26(1):23–63, 1994.

[6] Stephen B. Pope. *Turbulent Flows*. Cambridge University Press, 2000.

[7] J Rotta. Statistical theory of nonhomogeneous turbulence. ii. *Z. Physik*, 131, 1951.

```
%%FORWARD EULER TIME-STEPPING
%Initialize the 3-D Grid in V1, V2, V3 space:
N = 10;

v1 = linspace(-3, 3, N);
v2 = linspace(-3, 3, N);
v3 = linspace(-3, 3, N);
dv = v1(2) - v1(1);

[V1, V2, V3] = ndgrid(v1, v2, v3);
F = [V1(:) V2(:) V3(:)]; %10^3 rows, 3 columns
mu = (5 * 10^-10)*ones(1,3);
%sigma = [1, 0.5, 0.2; 0.5, 1, 0.3; 0.2, 0.3, 1];
sigma = (0.2)*eye(3);

%Initialize the probability density function 10^3 vector
f = mvnpdf(F, mu, sigma);

%The RIJ function calculates the Reynolds stresses by integrating the pdf, f
RIJ = ReStr(f,F);
eps = (RIJ(1,1)+RIJ(2,2)+RIJ(3,3))/(2*2.36);

b11 = [];
b22 = [];
b12 = [];
b33 = [];

T= 0.02;
nt= 10000;
dt=T/nt;

dUdy = 1;
Ceps1= 1.56;
Ceps2= 1.9;
CR = 1.8;
C2 = 0.6;
Co = 2.1;

V1 = F(:, 1);
V2 = F(:, 2);
V3 = F(:, 3);

U1_mean = trapz(V1 .* f);
U2_mean = trapz(V2 .* f);
U3_mean = trapz(V3 .* f);


S = 1; %magnitude of shear
model = 4;
tke = 0.5*(RIJ(1,1)+RIJ(2,2)+RIJ(3,3));
GIJ = GCalc(S,RIJ,eps,tke,model);
Prod = -RIJ(1,2)*dUdy;
%Co = 2./3.*(CR - 1 + C2 * Prod/eps);

for t = 0:dt:T

    G11 = GIJ(1, 1); G12 = GIJ(1, 2); G13 = GIJ(1, 3);
    G21 = GIJ(2, 1); G22 = GIJ(2, 2); G23 = GIJ(2, 3);
    G31 = GIJ(3, 1); G32 = GIJ(3, 2); G33 = GIJ(3, 3);

%     G11 = 1; G12 = 0.5; G13 = 1;
%     G21 = 0.5; G22 = 1; G23 = 0.5;
%     G31 = 1; G32 = 0.5; G33 = 1;


    %Calculate the derivative of f with respect to v1, v2, v3 for all grid points
    dfdv1 = [];
    for l = 1:N^3
        if mod(l, N) == 1
            dfdv1(l) = (-3*f(l) + 4*f(l+1) - f(l+2))/(2*dv);
        elseif mod(l, N) == 0
            dfdv1(l) = (3*f(l) - 4*f(l-1) + f(l-2))/(2*dv);
        else
```

```
        dfdv1(l) = (f(l+1) - f(l-1))/(2*dv);
    end
end
dfdv1 = dfdv1.';

dfdv2 = [];
for l = 1:N^3
    if l <= N
        dfdv2(l) = (-3*f(l) + 4*f(l+(N*1)) - f(l+(N*2)))/(2*dv);
    elseif l >= N^3 - N + 1
        dfdv2(l) = (3*f(l) - 4*f(l-(N*1)) + f(l-(N*2)))/(2*dv);
    else
        dfdv2(l) = (f(l+(N*1)) - f(l-(N*1)))/(2*dv);
    end
end
dfdv2 = dfdv2.';

dfdv3 = [];
for l = 1:N^3
    if l <= N^2
        dfdv3(l) = (-3*f(l) + 4*f(l+((N^2)*1)) - f(l+((N^2)*2)))/(2*dv);
    elseif l >= N^3 - (N^2) + 1
        dfdv3(l) = (3*f(l) - 4*f(l-((N^2)*1)) + f(l-((N^2)*2)))/(2*dv);
    else
        dfdv3(l) = (f(l+((N^2)*1)) - f(l-((N^2)*1)))/(2*dv);
    end
end
dfdv3 = dfdv3.';

%Calculate the second derivative of f with respect to v1, v2, v3 for all grid points;
d2fdv1 = [];
for l = 1:N^3
    if mod(l, N) == 1
        d2fdv1(l) = ((2*f(l) - 5*f(l+1) + 4*f(l+2) - f(l + 3))/(dv^3));
    elseif mod(l, N) == 0
        d2fdv1(l) = ((2*f(l) - 5*f(l-1) + 4*f(l-2) - f(l-3))/(dv^3));
    else
        d2fdv1(l) = ((f(l+1) - 2*f(1) + f(l-1))/(dv^2));
    end
end
d2fdv1 =  d2fdv1.';


d2fdv2 = [];
for l = 1:N^3
    if l <= N
        d2fdv2(l) = ((2*f(l) - 5*f(l+(N*1)) + 4*f(l+(N*2)) - f(l + (N*3)))/(dv^3));
    elseif l >= N^3 - N + 1
        d2fdv2(l) = ((2*f(l) - 5*f(l-(N*1)) + 4*f(l-(N*2)) - f(l - (N*3)))/(dv^3));
    else
        d2fdv2(l) = ((f(l+(N*1)) - 2*f(1) + f(l-(N*1)))/(dv^2));
    end
end
d2fdv2 =  d2fdv2.';

d2fdv3 = [];
for l = 1:N^3
    if l <= N^2
        d2fdv3(l) = ((2*f(l) - 5*f(l+((N^2)*1)) + 4*f(l+((N^2)*2)) - f(l + ((N^2)*3)))/(dv^3));
    elseif l >= N^3 - (N^2) + 1
        d2fdv3(l) = ((2*f(l) - 5*f(l-((N^2)*1)) + 4*f(l-((N^2)*2)) - f(l - ((N^2)*3)))/(dv^3));
    else
        d2fdv3(l) = ((f(l+((N^2)*1)) - 2*f(1) + f(l-((N^2)*1)))/(dv^2));
    end
end
d2fdv3 =  d2fdv3.';


%Update the PDF based on equation (5) from progress report 2
f = f - dt* ((G11*(f + (V1 - U1_mean).* dfdv1)) + (G12*((V2 - U2_mean) .* dfdv1)) + (G13*((V3 - U3_mean) .* dfdv1)) + ...
    + (G21*((V1 - U1_mean) .* dfdv2)) + (G22*(f + (V2 - U2_mean).* dfdv2)) + (G23*((V3 - U3_mean).* dfdv2)) + ...
    + (G31*((V1 - U1_mean).* dfdv3)) + (G32*((V2 - U2_mean).* dfdv3)) + (G33*(f + (V3 - U3_mean).* dfdv3)) + ...
    - (3/2)*Co*eps*(d2fdv1 + d2fdv2 +  d2fdv3));
```

```
    %Update all values
    U1_mean = trapz(V1 .* f);
    U2_mean = trapz(V2 .* f);
    U3_mean = trapz(V3 .* f);

    RIJ = ReStr(f,F);
    Prod = -RIJ(1,2)* dUdy;

    eps = eps + dt*((eps^2/tke)*(Ceps1*(Prod/eps) - Ceps2));
%     eps = Prod/1.6;
    tke = tke + dt*(Prod - eps);

    GIJ = GCalc(S,RIJ,eps,tke,model);

    %R11_current = trapz((V3 - U3_mean).^2 .* f);
    b11_current = RIJ(1, 1)/(2*tke) - 1/3;
    b11 = [b11, b11_current];

    b22_current = RIJ(2, 2)/(2*tke) - 1/3;
    b22 = [b22, b22_current];

    b33_current = RIJ(3, 3)/(2*tke) - 1/3;
    b33 = [b33, b33_current];

    %R12_current = trapz((V2 - U2_mean).* (V1 - U1_mean).* f);
    b12_current = RIJ(1, 2)/(2*tke);
    b12 = [b12, b12_current];

end

plot(0:dt:T, b11 , '--', 'LineWidth',1);
hold on;
plot(0:dt:T, b12, '--', 'LineWidth',1);
% hold on;
% plot(0:dt:T, b22, '--', 'LineWidth',1);
% hold on;
% plot(0:dt:T, b33, '--', 'LineWidth',1);

xlabel('$t$', 'interpreter', 'latex');
lgd = legend('$b_{11}$','$b_{12}$', 'interpreter', 'latex', 'location', 'southeast');
title('Model ',model, 'interpreter', 'latex');
ylim([-0.01 0.06])
```

Published with MATLAB® R2021a

```
%%RUNGE KUTTA 2 TIME-STEPPING
%Initialize the 3-D Grid in V1, V2, V3 space:
for m =1:4
N = 10;

v1 = linspace(-3, 3, N);
v2 = linspace(-3, 3, N);
v3 = linspace(-3, 3, N);
dv = v1(2) - v1(1);

[V1, V2, V3] = ndgrid(v1, v2, v3);
F = [V1(:) V2(:) V3(:)]; %10^3 rows, 3 columns
mu = (5e-10)*ones(1,3);
%sigma = [1, 0.5, 0.2; 0.5, 1, 0.3; 0.2, 0.3, 1];
sigma = 0.2*eye(3);

%Initialize the probability density function 10^3 vector
f = mvnpdf(F, mu, sigma);

%The RIJ function calculates the Reynolds stresses by integrating the pdf, f
RIJ = ReStr(f,F);
eps = (RIJ(1,1)+RIJ(2,2)+RIJ(3,3))/(2*2.36);

b11 = [];
b22 = [];
b12 = [];
b33 = [];

T= 0.2;
nt= 10000;
dt=T/nt;

dUdy = 1;
Ceps1= 1.56;
Ceps2= 1.9;
CR = 1.8;
C2 = 0.6;
Co = 2.1;

V1 = F(:, 1);
V2 = F(:, 2);
V3 = F(:, 3);

U1_mean = trapz(V1 .* f);
U2_mean = trapz(V2 .* f);
U3_mean = trapz(V3 .* f);


S = 1; %magnitude of shear
model = m;
tke = 0.5*(RIJ(1,1)+RIJ(2,2)+RIJ(3,3));
GIJ = GCalc(S,RIJ,eps,tke,model);
Prod = -RIJ(1,2)*dUdy;
Co = 2./3.*(CR - 1 + C2 * Prod/eps);

for t = 0:dt:T

    G11 = GIJ(1, 1); G12 = GIJ(1, 2); G13 = GIJ(1, 3);
    G21 = GIJ(2, 1); G22 = GIJ(2, 2); G23 = GIJ(2, 3);
    G31 = GIJ(3, 1); G32 = GIJ(3, 2); G33 = GIJ(3, 3);

%     G11 = 1; G12 = 0.5; G13 = 1;
%     G21 = 0.5; G22 = 1; G23 = 0.5;
%     G31 = 1; G32 = 0.5; G33 = 1;


    %Calculate the derivative of f with respect to v1, v2, v3 for all grid points
    dfdv1 = [];
    for l = 1:N^3
        if mod(l, N) == 1
            dfdv1(l) = (-3*f(l) + 4*f(l+1) - f(l+2))/(2*dv);
        elseif mod(l, N) == 0
            dfdv1(l) = (3*f(l) - 4*f(l-1) + f(l-2))/(2*dv);
        else
            dfdv1(l) = (f(l+1) - f(l-1))/(2*dv);
        end
    end
```

```
dfdv1 = dfdv1.';

dfdv2 = [];
for l = 1:N^3
    if l <= N
        dfdv2(l) = (-3*f(l) + 4*f(l+(N*1)) - f(l+(N*2)))/(2*dv);
    elseif l >= N^3 - N + 1
        dfdv2(l) = (3*f(l) - 4*f(l-(N*1)) + f(l-(N*2)))/(2*dv);
    else
        dfdv2(l) = (f(l+(N*1)) - f(l-(N*1)))/(2*dv);
    end
end
dfdv2 = dfdv2.';

dfdv3 = [];
for l = 1:N^3
    if l <= N^2
        dfdv3(l) = (-3*f(l) + 4*f(l+((N^2)*1)) - f(l+((N^2)*2)))/(2*dv);
    elseif l >= N^3 - (N^2) + 1
        dfdv3(l) = (3*f(l) - 4*f(l-((N^2)*1)) + f(l-((N^2)*2)))/(2*dv);
    else
        dfdv3(l) = (f(l+((N^2)*1)) - f(l-((N^2)*1)))/(2*dv);
    end
end
dfdv3 = dfdv3.';

%Calculate the second derivative of f with respect to v1, v2, v3 for all grid points;
d2fdv1 = [];
for l = 1:N^3
    if mod(l, N) == 1
        d2fdv1(l) = ((2*f(l) - 5*f(l+1) + 4*f(l+2) - f(l + 3))/(dv^3));
    elseif mod(l, N) == 0
        d2fdv1(l) = ((2*f(l) - 5*f(l-1) + 4*f(l-2) - f(l-3))/(dv^3));
    else
        d2fdv1(l) = ((f(l+1) - 2*f(1) + f(l-1))/(dv^2));
    end
end
d2fdv1 =  d2fdv1.';


d2fdv2 = [];
for l = 1:N^3
    if l <= N
        d2fdv2(l) = ((2*f(l) - 5*f(l+(N*1)) + 4*f(l+(N*2)) - f(l + (N*3)))/(dv^3));
    elseif l >= N^3 - N + 1
        d2fdv2(l) = ((2*f(l) - 5*f(l-(N*1)) + 4*f(l-(N*2)) - f(l - (N*3)))/(dv^3));
    else
        d2fdv2(l) = ((f(l+(N*1)) - 2*f(1) + f(l-(N*1)))/(dv^2));
    end
end
d2fdv2 =  d2fdv2.';

d2fdv3 = [];
for l = 1:N^3
    if l <= N^2
        d2fdv3(l) = ((2*f(l) - 5*f(l+((N^2)*1)) + 4*f(l+((N^2)*2)) - f(l + ((N^2)*3)))/(dv^3));
    elseif l >= N^3 - (N^2) + 1
        d2fdv3(l) = ((2*f(l) - 5*f(l-((N^2)*1)) + 4*f(l-((N^2)*2)) - f(l - ((N^2)*3)))/(dv^3));
    else
        d2fdv3(l) = ((f(l+((N^2)*1)) - 2*f(1) + f(l-((N^2)*1)))/(dv^2));
    end
end
d2fdv3 =  d2fdv3.';

Co = 2./3.*(CR - 1 + C2 * Prod/eps);

%Update the PDF based on equation (5) from progress report 2
f1 = f - (dt*0.5)* ((G11*(f + (V1 - U1_mean).* dfdv1)) + (G12*((V2 - U2_mean) .* dfdv1)) + (G13*((V3 - U3_mean) .* dfdv1)) + ...
    + (G21*((V1 - U1_mean) .* dfdv2)) + (G22*(f + (V2 - U2_mean).* dfdv2)) + (G23*((V3 - U3_mean).* dfdv2)) + ...
    + (G31*((V1 - U1_mean).* dfdv3)) + (G32*((V2 - U2_mean).* dfdv3)) + (G33*(f + (V3 - U3_mean).* dfdv3)) + ...
    - (3/2)*Co*eps*(d2fdv1 + d2fdv2 +  d2fdv3));
f = f1 - dt* ((G11*(f1 + (V1 - U1_mean).* dfdv1)) + (G12*((V2 - U2_mean) .* dfdv1)) + (G13*((V3 - U3_mean) .* dfdv1)) + ...
    + (G21*((V1 - U1_mean) .* dfdv2)) + (G22*(f1 + (V2 - U2_mean).* dfdv2)) + (G23*((V3 - U3_mean).* dfdv2)) + ...
    + (G31*((V1 - U1_mean).* dfdv3)) + (G32*((V2 - U2_mean).* dfdv3)) + (G33*(f1 + (V3 - U3_mean).* dfdv3)) + ...
    - (3/2)*Co*eps*(d2fdv1 + d2fdv2 +  d2fdv3));


%Update all values
U1_mean = trapz(V1 .* f);
```

```
    U2_mean = trapz(V2 .* f);
    U3_mean = trapz(V3 .* f);

    RIJ = ReStr(f,F);
    Prod = -RIJ(1,2)* dUdy;

    %eps = eps + dt*((eps^2/tke)*(Ceps1*(1.6) - Ceps2));
    eps = Prod/1.6;
    tke = tke + dt*(Prod - eps);

    GIJ = GCalc(S,RIJ,eps,tke,model);

    %R11_current = trapz((V3 - U3_mean).^2 .* f);
    b11_current = RIJ(1, 1)/(2*tke) - 1/3;
    b11 = [b11, b11_current];

    b22_current = RIJ(2, 2)/(2*tke) - 1/3;
    b22 = [b22, b22_current];

    b33_current = RIJ(3, 3)/(2*tke) - 1/3;
    b33 = [b33, b33_current];

    %R12_current = trapz((V2 - U2_mean).* (V1 - U1_mean).* f);
    b12_current = RIJ(1, 2)/(2*tke);
    b12 = [b12, b12_current];

end
figure
plot(0:dt:T, b11 , '--', 'LineWidth',1);
hold on;
plot(0:dt:T, b12, '--', 'LineWidth',1);
% hold on;
% plot(0:dt:T, b22, '--', 'LineWidth',1);
% hold on;
% plot(0:dt:T, b33, '--', 'LineWidth',1);

xlabel('$t$', 'interpreter', 'latex');
lgd = legend('$b_{11}$','$b_{12}$', 'interpreter', 'latex', 'location', 'southeast');
title('Model ',model, 'interpreter', 'latex');
end
```

Published with MATLAB® R2021a

```
function [RIJ] = ReStr(f,F)
%this function takes the velocities:
% F : N^3 x 3
% f : the pdf NxNxN
% the mean velocity calculation in respective directions
Vm = trapz(F.*f);
Vm1 = Vm(1);
Vm2 = Vm(2);
Vm3 = Vm(3);


% Reynold stresses by using moments of pdf 'f'
RIJ = zeros(3,3);

% diagonal elements
RIJ(1,1) = trapz(((F(:,1)- Vm1).^2 ).*f);
RIJ(2,2) = trapz(((F(:,2)- Vm2).*(F(:,2)- Vm2) ).*f);
RIJ(3,3) = trapz(((F(:,3)- Vm3).*(F(:,3)- Vm3) ).*f);

% upper diagonal elements
RIJ(1,2) = trapz(((F(:,1)- Vm1).*(F(:,2)- Vm2) ).*f);
RIJ(1,3) = trapz(((F(:,1)- Vm1).*(F(:,3)- Vm3) ).*f);
RIJ(2,3) = trapz(((F(:,2)- Vm2).*(F(:,3)- Vm3) ).*f);

% lower diagonal elements

RIJ(2,1) = RIJ(1,2);
RIJ(3,1) = RIJ(1,3);
RIJ(3,2) = RIJ(2,3);




end
```

Published with MATLAB® R2021a

```
function [Gij] = GCalc(S,RIJ,eps,TKE,model)
%This function returns Gij (3x3) tensor
% S = the magnitude of shear
% RIJ = reynolds stress matrix
% eps = dissipation
% TKE = turbulent kinetic energy
% model : 1-> SLM (RA), 2-> HP1, 3-> HP2, 4-> LIPM
%initializing drift factor
Gij = zeros(3,3);

% evaluating the anisotropy tensor
b = RIJ/(2*TKE)-eye(3).*(1/3);

%Strain rate defination
Sij = [0,0.5*S,0; 0.5*S,0,0;0,0,0];
% Hijkl calculation
% defining constants based on model
if (model == 1)
    C0 = 2.1;
    alpha1 = -(0.5+ 0.75*C0);
    alpha2 = 0;alpha3 = 0;
    beta1 =0; beta2 = 0;beta3 = 0;
    gamma1 =0; gamma2=0; gamma3 =0; gamma4 = 0; gamma5 =0; gamma6=0;
elseif(model == 2)
    C0 = 2.1;
    alpha2 = 3.7 ;alpha3 = 0;
    beta1 =-1/5; beta2 = 4/5;beta3 = -1/5;
    gamma1 =0; gamma2=3.01; gamma3 =-2.18; gamma4 = 0; gamma5 =4.29; gamma6=-3.09;
elseif(model == 3)
    C0 = 2.1;
    alpha2 = 3.78 ;alpha3 = 0;
    beta1 =-1/5; beta2 = 4/5;beta3 = -1/5;
    gamma1 =0; gamma2=1.04; gamma3 =0.34; gamma4 = 0; gamma5 =1.99; gamma6=-0.76;

elseif(model == 4)
    C0 = 2.1;
    alpha2 = 3.5 ;alpha3 = -10.5;
    beta1 =-1/5; beta2 = 4/5;beta3 = -1/5;
    gamma1 =0; gamma2=0; gamma3 =0; gamma4 = 0; gamma5 =0.6; gamma6=-0.6;
end


% Calculating the matrix Gij using the GLM coefficients

k =1;l=2;
for i=1:3
    for j =1:3
        if (model == 2 || model == 3)
            gamstar = gamma2 + gamma3 +gamma5 + gamma6;
            Prod = eps*(-2*(b(i,j)*Sij(i,j)));
            I1 = - 0.5*(Prod/eps);
            I2 = (b(i,j)^2)*Sij(i,j);
            alpha1 = -(0.5 + 0.75*C0 + (b(1, 1) + b(2, 2) + b(3,3))^2 *(alpha2+ 0.33*alpha3) + (b(1, 1) + b(2, 2) + b(3,3))^3 * alpha3 +(beta1 + beta2
        end
        if (model == 4)
            Prod = eps*(-2*(b(i,j)*Sij(i,j)));
            C2 =0.6;
            alpha1 = -(0.5 + 0.75*C0 ) + 0.5*C2*(Prod/eps) + 3*alpha2*((b(1, 1) + b(2, 2) + b(3,3))^3);
        end

    term1 = beta1*krdel(i,j)*krdel(k,l);
    term2 = beta2*krdel(i,k)*krdel(j,l);
    term3 = beta3*krdel(i,l)*krdel(j,k);
    term4 = gamma1*krdel(i,j)*b(k,l);
    term5 = gamma2*krdel(i,k)*b(j,l);
    term6 = gamma3*krdel(i,l)*b(j,k);
    term7 = gamma4*b(i,j)*krdel(k,l);
    term8 = gamma5*b(i,k)*krdel(j,l);
    term9 = gamma6*b(i,l)*krdel(j,k);
    H = term1 + term2 + term3 + term4 + term5 + term6 + term7 + term8 + term9;

    term0 =(eps/TKE)*(alpha1*krdel(i,j) + alpha2*b(i,j) + alpha3*(b(i,j)^2));
    Gij(i,j) = term0 + H*S;
    end
end

end
```

Published with MATLAB® R2021a

```
function [num] = krdel(a,b)
%UNTITLED3 Summary of this function goes here
%   Detailed explanation goes here
if (a==b)
    num =1;
else
    num = 0;
end


end
```

Published with MATLAB® R2021a