Project 2

Ryan Skinner

ASEN 6037: Turbulence

Due 2015/05/08

In this project, we assess the strengths and shortcomings of Reynolds-averaged Navier-Stokes (RANS) models through derivations and comparison to DNS simulations of various flow classes.

Part 1 Derivation of RANS Models

Please see the attached handwritten work; much of this problem consists of derivations which are long enough to preclude typesetting in a reasonable amount of time.

Part 2 Testing of RANS Models: Turbulent Channel Flow

Problem 2.1

Consider a fully-developed turbulent channel flow. In such a flow, which of the components of $\overline{u}_i$, $\overline{S}_{ij}$, and $\overline{W}_{ij}$ are non-zero? What are $\partial/\partial t$ and $\overline{u}_i \partial/\partial x_i$? Comment on the validity of the equilibrium assumption used in Problem 1.8.

We will assume this turbulent channel flow to be either two- or three-dimensional, with the stream-wise direction denoted by x, and the span-wise direction(s) denoted by y.

Only $\overline{u}_x$ is non-zero, since this is the mean direction of flow. Any non-zero $\overline{u}_y$ would violate symmetry and, if near a wall, also the no-penetration boundary condition.

In the core flow away from the walls, all components of $\overline{S}_{ij}$ are zero, since the velocity profile has negligible gradients in all directions ($\partial/\partial x$ is identically zero in a fully-developed flow). In the near-wall region, $\overline{S}_{ij}$ does have non-zero components, because there is a large velocity gradient in the wall-normal direction. In this region, $\overline{S}_{xy}$ is non-zero; it involves the derivative of the stream-wise velocity with respect to the wall-normal direction. All other components are zero because span-wise mean velocities are zero.

In a similar manner, all components of $\overline{W}_{ij}$ are zero in the core flow. Again, this is because there exist no substantial velocity gradients in the core region. In the near-wall region, however, hairpin vortices peel off and create preferential vorticity. These structures create zero mean downstream and wall-normal vorticity due to symmetry of the hairpin, but non-zero vorticity in the direction of (wall-normal $\times$ stream-wise). Taking the channel to be three-dimensional, if we consider the flow direction to be into the page x, along the bottom wall (whose normal is in the y-direction, pointing into the channel) vorticity will be preferentially in the $-\hat{z}$-direction. Thus, near this wall, $\overline{W}_{xy}$ and $\overline{W}_{yx}$ will be non-zero. Similar arguments apply to the other walls.

Since the flow is fully-developed, the averages of flow quantities will have no change in time, making $\partial/\partial t = 0$. However, there will still be fluctuations in the flow for which $\partial/\partial t \neq 0$. For the convective derivative, we already know that spatial derivatives in the stream-wise direction must be zero, so $\overline{u}_x \partial/\partial x = 0$. Additionally, there are no mean velocities in the span-wise directions, so $\overline{u}_y = 0$. Thus all $\overline{u}_i \partial/\partial x_i = 0$.

In conclusion, the equilibrium assumption that postulates constant anisotropy is an acceptable one for average quantities. Though this is not true for fluctuating, instantaneous quantities, we will presumably be using the RANS equations for modelling, in which only average quantities are used.

Problem 2.2

Using the turbulent channel flow DNS data from Moser, Kim, and Mansour (1999), we calculate and plot the non-zero components of $\overline{u}_i$, $\overline{u_i' u_j'}$, k, ϵ, a_{ij}, and $\overline{S}_{ij}$ as functions of both y^+ and y/h. Results are

displayed in Figure 1 through Figure 6.

Problem 2.3

Using a_{12} and $\overline{S}_{12}$, the mean value of the eddy viscosity coefficient C_μ away from the walls ($y/h > 0.2$) is calculated to be approximately 0.086. This is the constant value that gives the closest agreement between the computational data and the closure model of

$$C_\mu = \frac{-\epsilon a_{12}}{2k\overline{S}_{12}}. \tag{1}$$

A plot of C_μ and its average value away from the wall is presented in Figure 7.

Problem 2.4

The closure from problem 1.9,

$$a_{ij} = -2\frac{\nu_T}{k}\overline{S}_{ij}, \qquad \text{where} \qquad \nu_T = C_\mu\frac{k^2}{\epsilon}, \tag{2}$$

fails to capture important physical effects near the channel walls. C_μ should drop significantly in the near-wall region, for roughly $y^+ < 75$, but it does not. This failure occurs for at least two reasons.

First, the gradient transport hypothesis that was used to derive (2) only holds (if at all) away from the walls. Near the walls, velocities and Reynolds numbers decrease to the point where viscous diffusion plays a large role in the flow dynamics. The gradient transport hypothesis assumes that Re_τ is high, and that transport within the flow is almost exclusively due to small eddies, rather than large eddies or viscous effects.

Second, we recall from Problem 1.8 that the equilibrium assumption led to the a_ij closure of (2), which yielded

$$a_{ij} = \frac{\alpha_2}{\alpha_1}\frac{k}{\epsilon}\overline{S}_{ij} = \frac{C2 - (4/3)}{(P/\epsilon) - 1 + C_1}\frac{k}{\epsilon}\overline{S}_{ij}. \tag{3}$$

Of note here is that the production term P was neglected when deriving (2). Turbulence production is important near the boundaries because of the high mean shear, and it is absent from (2).

Problem 2.5

A popular approach to improving the closure from (2) near channel walls is to limit C_μ such that

$$C_\mu = \begin{cases} 0.09 & \text{for } (Sk/\epsilon) \leq 3.4 \\ 0.31(Sk/\epsilon)^{-1} & \text{for } (Sk/\epsilon) > 3.4 \end{cases} \tag{4}$$

This has a positive effect on the model near channel walls. As one approaches the wall, the relative shear strength (Sk/ϵ) increases dramatically, and this method allows C_μ to drop continuously from its limit of 0.09 to much smaller values near the wall, inversely proportional to the relative shear strength. According to Figure 6, the strain rate really starts to pick up approaching the wall around $y^+ \approx 50$, and this is roughly where C_μ would need to start dropping to account for wall effects as seen in Figure 7.

I would consider this a decent approach to correcting our turbulence model in the near-wall region. It is a model and thus will not be perfect, but it seems to respond to the wall in a physically-justified fashion at the appropriate distance. This presumes that the activation value of $(Sk/\epsilon) = 3.4$ has been chosen in good agreement with empirical data. It is also convenient that the cut-off model for C_μ is easily computed.

Problem 2.6

From a physical perspective, turbulence models have difficulty for $y^+ < 30$ for a number of reasons. First, in deriving models we often make high-Re assumptions that are invalid near the wall. This leads to a neglect of viscous effects as well as anisotropy and turbulence kinetic energy production due to shearing. It is also common to assume that ϵ_{ij} is isotropic, as noted in Problem 1.5 due to K41 theory, but this too breaks down near the walls. This becomes particularly problematic when these anisotropic turbulent structures form near the walls and are convected into the core flow.

Very close to the walls, the log law,

$$y^+ = \frac{1}{\kappa} \ln y^+ + B, \tag{5}$$

constitutes a better mathematical function that allows for more accurate prediction of a_{ij}. For *very* near-wall physics, the viscous sublayer relationship of $u^+ = y^+$ would be the most accurate.

Problem 2.7

If we write a_{ij} using its modelled form from (2), we can substitute it into the equation for the Reynolds stresses, yielding, after some simplification,

$$\overline{u'_\alpha{}^2} = \frac{2}{3} k \left(1 - C_\mu \frac{k}{\epsilon} \overline{S}_{\alpha\alpha} \right), \tag{6}$$

where no summation is implied over Greek indices by convention. However, note that $\overline{S}_{\alpha\alpha}$ involves derivatives of mean flow velocities in the same spatial direction. In this flow, mean span-wise velocities are zero, and flow statistics are invariant in the stream-wise direction. Thus $\overline{S}_{\alpha\alpha} = 0$ and we are left with the isotropic form

$$\overline{u'_i{}^2} = \frac{2}{3} k. \tag{7}$$

Referencing Figure 2, the diagonal Reynolds stresses are very much *not constant* in the DNS flow, nor are they equal to one another. The only saving grace of our model is that both it and the DNS results show diagonal Reynolds stresses summing to double the turbulence intensity k, but this occurs by definition.

The discrepancies can be explained as follows. First, we dropped all redistribution terms very early on in the derivation of our model. This makes it impossible for anisotropy to move between different components of a_{ij} as fluid parcels are subjected to convection and shear. Additionally, the gradient transport hypothesis we employed assumed that small scale eddies were solely responsible for transport phenomena. Because we dictated that ϵ_{ij} be isotropic from K41, it comes as little surprise that giving sole dynamic importance to these isotropic small scales results in an isotropic flow at all scales.

Problem 2.8

In conclusion, this closure model is simple and easy to implement, but inaccurate for wall-bounded flows, especially in the near-wall region. Even in the core flow regions, assuming a constant C_μ is not particularly accurate, as can be seen in Figure 7. This leads to inaccuracies when simulating flows with small-scale production, such as combustion or wind moving through trees. As we determined by looking at the model's treatment of diagonal Reynolds stresses, flows with any degree of anisotropy will be poorly-represented with this closure model. Having assumed equilibrium, unsteadiness in flows will also be ill-resolved. Idealized flows without boundary layers, without time-varying phenomena,

and without any appreciable degree of anisotropy will be resolved fairly well with this turbulence model.

Most real-world engineering flows fall into these categories, which makes it surprising that such a model so so widely-employed in industry. Nonetheless, the simplicity and speed with which the governing equations can be numerically solved is appealing in a production environment. Presumably, these types of models assist with rapid iteration of design. Despite it's shortcomings, this model is probably 'good enough' to reach a decent design that can be refined using more experienced humans or higher-fidelity models.

Part 3 Testing of RANS Models: Unsteady Homogeneous Flow

Problems 3.1–3.4 consist mostly of derivations which are not typeset here, but are included in the attached handwritten documents.

Problem 3.5

Assuming that $a_{12} = 0$ at $t = 0$, we numerically integrate the set of ordinary differential equations found in Problem 3.4 for the SKE and DKE models. We take $S^* \equiv Sk_0/\epsilon_0 = 3.3$, and examine the flow evolution for $(\omega/S) = \{0.01, 0.1, 0.5, 1, 10\}$. The evolution of a_{12} is plotted as a function of $S \cdot t = S^*\tau$ for each of the ω/S values and both models. We furthermore assume that $C_\mu = 0.09$ in the SKE model.

Results are shown in Figures 8 and 9. For implementation, Matlab's `ode45` numerical integration function was used, with initial conditions of $\tilde{k} = \tilde{\epsilon} = 1$ and $a_{12} = 0$ where applicable.

Problem 3.6

Figures 10, 11, and 12 present comparisons of the SKE and DKE model results to DNS data from Yu and Girimaji (2006), for $\omega/S = \{0.5, 1, 10\}$. Note that the authors present b_{12}, where $b_{12} = \frac{1}{2}a_{12}$.

The DKE model matches DNS data much more closely than the SKE model. As the ratio of shearing frequency to shearing magnitude (ω/S) increases, the accuracy of the DKE model increases, as evinced by a reduction in phase lag to near zero in the $\omega/S = 10$ case, and a reduction in magnitude to match that of the DNS data almost identically. The SKE model is decent (though still worse than DKE) for low values of ω/S, but in all comparisons to DNS, the SKE model exhibits excessive phase lag and un-physically high a_{12} amplitude.

As discussed in previous problems, the SKE model treats C_μ as constant, which fails near the walls and in any locations of high relative shear. In agreement with our observations, we would expect the SKE model to perform poorly in this flow, which undergoes periodic applied shearing.

The DKE model, on the other hand, neglects redistribution and transport terms but retains the production terms that become important in regions of high shear. Understandably, it performs better than the SKE model for the flow in question. Due to its neglect of redistribution terms (which are 'rapid' quantities involving derivatives of velocity components) though, phase lag still occurs when the frequency of shearing is high, as seen for $\omega/S = 0.5$ in Figure 10.

Problem 3.7

Our results indicate that the equilibrium approximation—which is made for the SKE model, but not for the DKE model—is inaccurate for unsteady periodically sheared turbulence. The approximation is acceptable if the flow is steady, and does not exhibit decay to isotropy. Since this assumption holds that the convective derivative of the anisotropy tensor is zero, it provides no mechanism for anisotropy generated near a shear layer or wall to decay as it is convected into the core flow. Thus, even steady

wall-bounded or sheared flows will have trouble being accurately captured when the equilibrium assumption is used. The ideal case of homogeneous isotropic turbulence in a periodic domain should be simulated accurately with this assumption, as should flow which are similar to HIT and are far from walls.

The gradient transport hypothesis was used to write $a_{ij} \sim \overline{S}_{ij}$. In flows where the mean shear rate is low, this is a fairly good assumption because most of the transport is due to small-scale turbulent motions. However, a high mean shear rate creates large scale turbulent eddies. These important transport mechanisms are not accounted for by the gradient transport hypothesis, which is why in our SKE model is highly inaccurate.

Part 4 Testing of RANS Models: Computational Fluid Dynamics Code

Problem 4.1

We now examine the turbulence models offered by Ansys Fluent. A brief explanation of each model follows, with physical assumptions and neglected effects and formulation type. Models which can be used for detached eddy simulation (DES) are addressed in a single category at the end. A similar approach is taken for large eddy simulation (LES) models.

Governing equations can be found in the literature, in an encyclopedia, and in the Fluent Theory Guide. They will not be typeset here, but rather discussed in qualitative detail.

1 → Spalart-Allmaras model

This is a simple one-equation model, which solves a modelled transport equation for the kinematic eddy viscosity. It performs well for wall-bounded flows, and flows which involve a combination of boundary layers and adverse pressure gradients. These are particularly common in aerospace and turbomachinery applications. It is an equilibrium model, and as such does not predict the decay of homogeneous isotropic turbulence (HIT). It is also unable to correctly predict behaviour of flow that rapidly transitions from wall-bounded to free shear, since length scales are assumed to change slowly in this model.

2 → Standard k-ϵ model

This model, referred to in previous parts of the project as SKE, is one of the simplest models. As we have seen, it requires the coupled solution of two separate transport equations for turbulence intensity k, and dissipation ϵ. The former is physically accurate, whereas the latter is derived using physical arguments such as the gradient transport hypothesis (GTH). Physical assumptions include the presence of fully-turbulent flow and negligible molecular viscosity. Constants in the model are empirically determined, and the turbulent viscosity C_μ is taken to be a constant. The validity of this assumption was the subject of Part 2. It offers reasonable accuracy for a wide variety of flows, and is considered both numerically stable and computationally efficient in many situations. It is also an equilibrium formulation and a two-equation model.

3 → Renormalization group (RNG) k-ϵ model

This model is similar to the SKE model. It performs better in flows subjected to rapid straining and swirling, as well as in flows with low-Re effects if tuned properly. These benefits are achieved through statistical arguments that result in an additional ϵ-equation term, the inclusion of swirl physics, and rewriting the effective viscosity to account for low Reynolds numbers. Of note is that the RNG k-ϵ

model is superior to the SKE model in cases with high streamline curvature. It is also an equilibrium formulation, with similar physical assumptions to the SKE model. It is a two-equation model.

4 → Realizable k-ϵ model

As with the RNG k-ϵ model, the realizable k-ϵ model modifies the SKE model; in this case, by re-formulating the turbulent viscosity ν_T and the transport equation for ϵ. In a sense, this model is more physical in that it only produces physically-realizable turbulence, unlike the previous two k-ϵ models. C_μ is also taken to be variable rather than constant, which allows improved prediction of the spreading rates of jets. In almost all practical situations, it outperforms the SKE model. However, it produces unphysical turbulent viscosities in cases where the domain contains both rotating and stationary regions of fluid. It is also an equilibrium formulation, with similar physical assumptions to the SKE model. It is a two-equation model.

5 → Standard k-ω model

The standard k-ω model is a modification of the standard k-ϵ model. Its formulation is similarly lacking in physical justification, but ϵ is replaced with an inverse turbulence time scale ω. It is still an equilibrium formulation, but offers slightly improved performance in the near-wall boundary layer region due to better handling of transport effects from the principal turbulent shear stress. This comes at an expense, however, since its accuracy is degraded away from walls. Physical assumptions not mentioned here are identical to the k-ϵ model, including use of the GTH. It is a two-equation model.

6 → Shear-stress transport (SST) k-ω model

The SST k-ω model combines the standard k-ω model with the standard k-ϵ model, to leverage improved near-wall accuracy of the former while retaining far-field accuracy of the latter. Again, physical assumptions including the GTH are the same as in both of its 'parent' models. Combination of the models is attained with a blending function, which activates one model or the other depending on which is more appropriate at a given location and time. This flow is more applicable to aeronautical flows than the standard k-ω model. It is still an equilibrium formulation. It is a two-equation model.

7 → v^2-f model

Similar to the standard k-ϵ model, the v^2-f model further seeks to resolve near-wall turbulence anisotropy and non-local pressure-strain correlations. It is intended for low-Re flows, and is valid even immediately at solid boundaries.

8 → Transition k-kl-ω model

The k-kl-ω model is a three-equation model, composed of transport equations for the following quantities: inverse turbulent time scale ω, turbulent kinetic energy k_T, and laminar kinetic energy k_L. Its primary area of application is boundary layer development, and is used to calculate the location at which the laminar-turbulent transition begins. As such, it has been specifically tailored to this need and neglects physics which do not apply near a boundary layer.

9 → Transition SST model

This model augments the SST k-ω model with additional transport equations, specifically tuned to model turbulent transition. It does an acceptable job of predicting the decay of turbulent intensity. It also requires a fairly fine mesh near the wall, with characteristic grid dimension of approximately

$y^+ \approx 1$. All approximations and physical limitations of the SST k-ω model apply here, save that this model is superior for transitional flows and requires the simultaneous solution of two more equations, one for the transition onset criterion and one for the intermittency. It is thus a four-equation model.

Appendix: Matlab Code

Listing 1: `project_02_part2.m`

```
% close all; clear all;
set(groot,'defaultTextInterpreter','none');
set(groot,'defaultLegendInterpreter','none');
set(groot,'defaultLineLineWidth',2);
set(groot,'defaultAxesXGrid','on');
set(groot,'defaultAxesYGrid','on');
set(groot,'defaultAxesFontSize',12);
set(groot,'defaultAxesPosition',[0.15,0.15,0.7,0.7]);

% Read in data.
data1 = read_MKM_data('/home/ryan/ASEN 6037 Turbulence/Project_02/data/chan590.means');
data2 = read_MKM_data('/home/ryan/ASEN 6037 Turbulence/Project_02/data/chan590.kbal');
data3 = read_MKM_data('/home/ryan/ASEN 6037 Turbulence/Project_02/data/chan590.reystress');
data = [data1;data2;data3];
data.keys

%%%
% Problem 2.1
%%%

% Plot stream-wise velocity.
figure();
hold on;
line(data('y+'),data('Umean'));
ax1 = gca();
ax2 = axes('Position',ax1.Position,'XAxisLocation','top','YAxisLocation','right');
line(data('y'),data('Umean'),'Parent',ax2);
hold off;
xlabel(ax1(1),'y+');
xlabel(ax2(1),'y/h');
ylabel(ax1(1),'mean(u)');

% Plot Reynolds stresses.
figure();
hold on;
fields = {'R_uu','R_uv','R_uw','R_vv','R_vw','R_ww'};
field_data = zeros(length(data('y')),length(fields));
for i = 1:length(fields)
    field_data(:,i) = data(fields{i});
end
line(repmat(data('y+'),1,length(fields)),field_data);
ax1 = gca();
ax2 = axes('Position',ax1.Position,'XAxisLocation','top','YAxisLocation','right');
line(repmat(data('y'),1,length(fields)),field_data);
hold off;
xlabel(ax1(1),'y+');
xlabel(ax2(1),'y/h');
ylabel(ax1(1),'Reynolds Stress');
legend(ax2,fields);
```

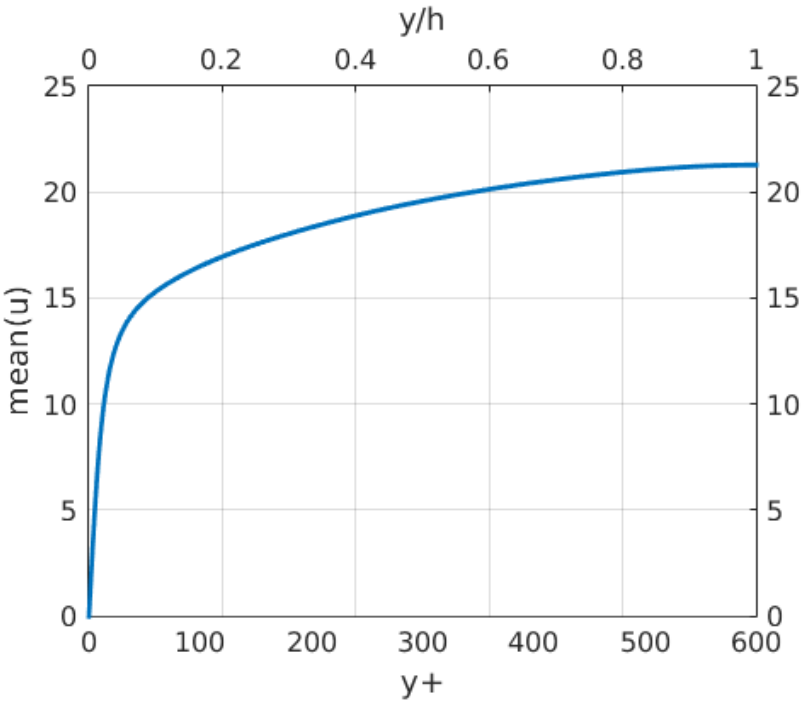

Figure 1: Mean velocity in the stream-wise direction.

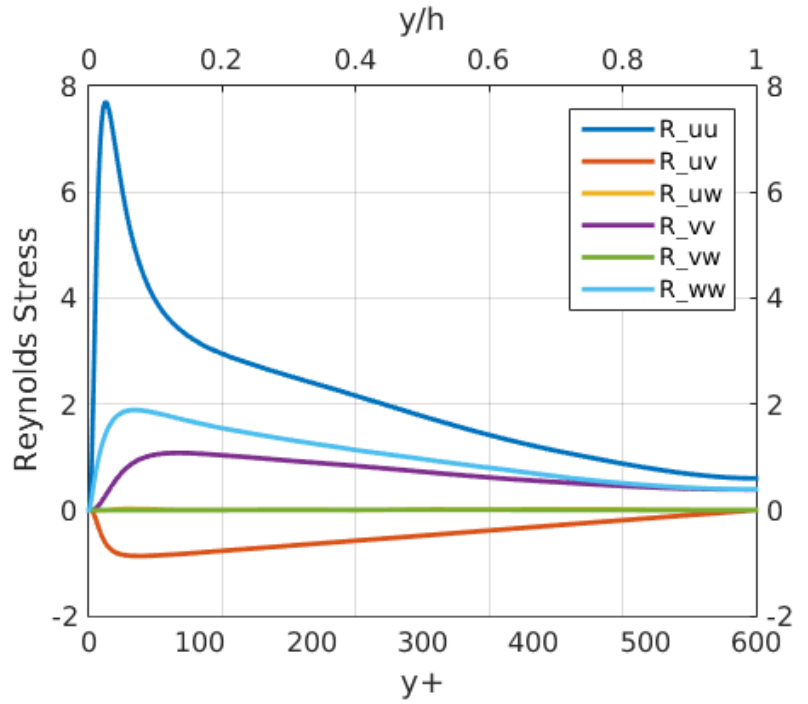

Figure 2: Components of the Reynolds stress tensor.

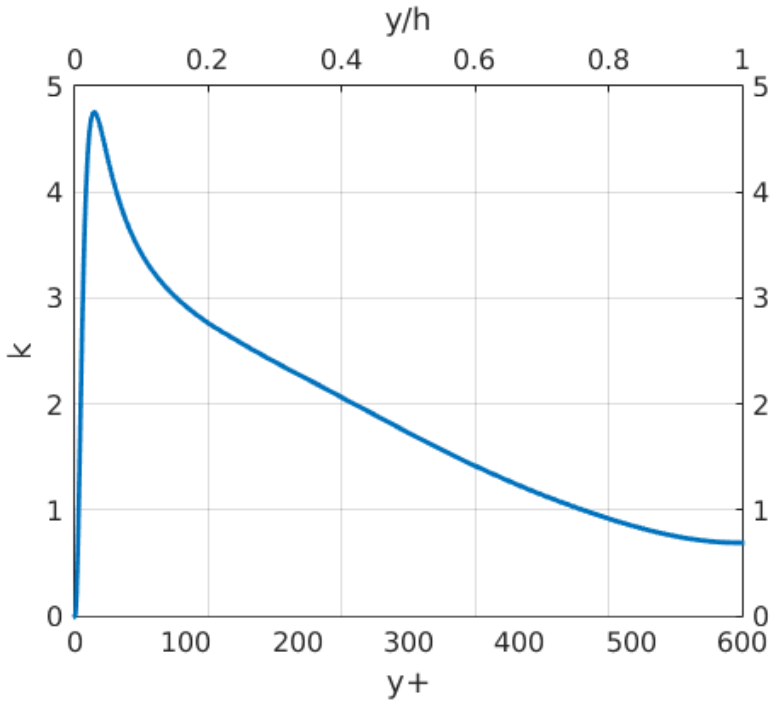

Figure 3: Turbulence intensity.

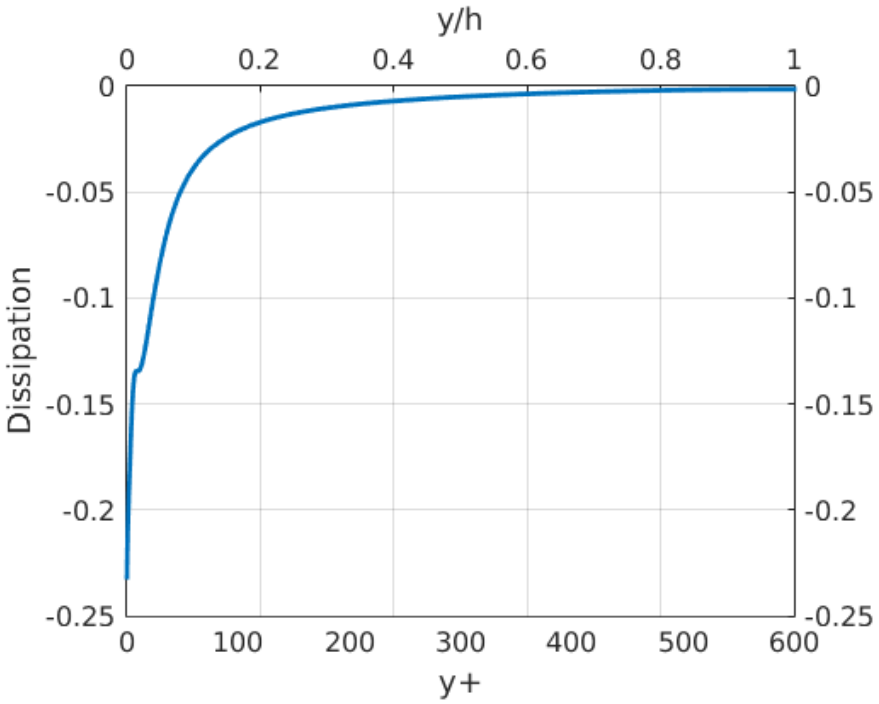

Figure 4: Turbulent kinetic energy dissipation.

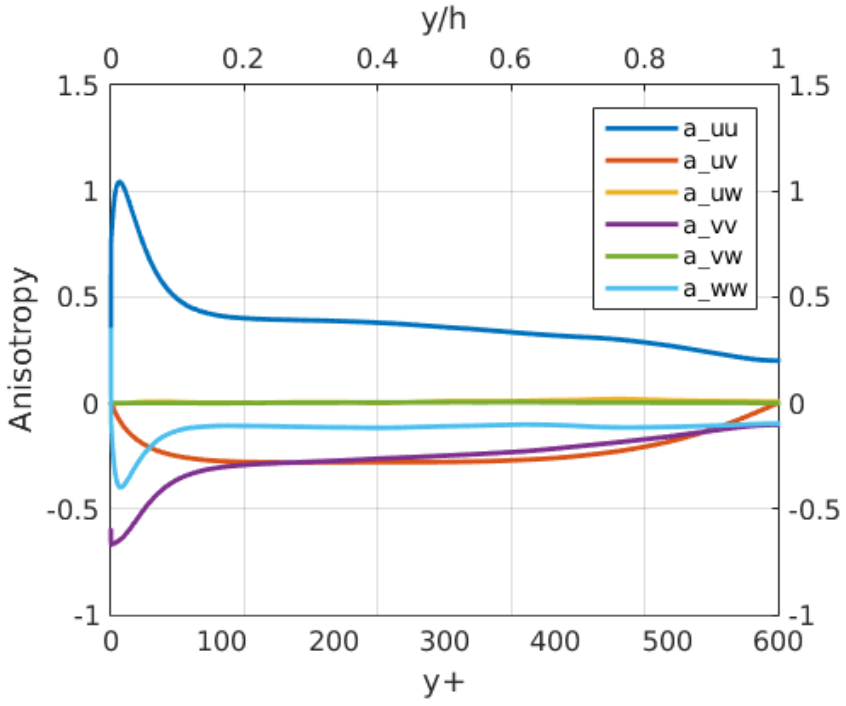

Figure 5: Components of the anisotropy tensor.

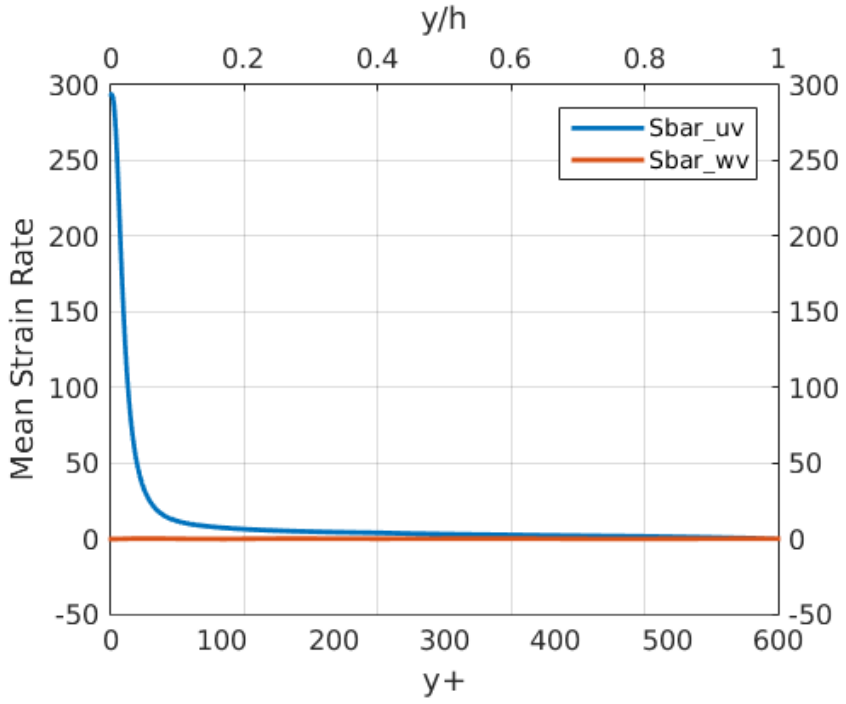

Figure 6: Components of the mean strain rate tensor; all components not shown are zero. Additionally, $\overline{S}_{wv}$ should be zero in an ideal flow because there should be no mean span-wise velocities.

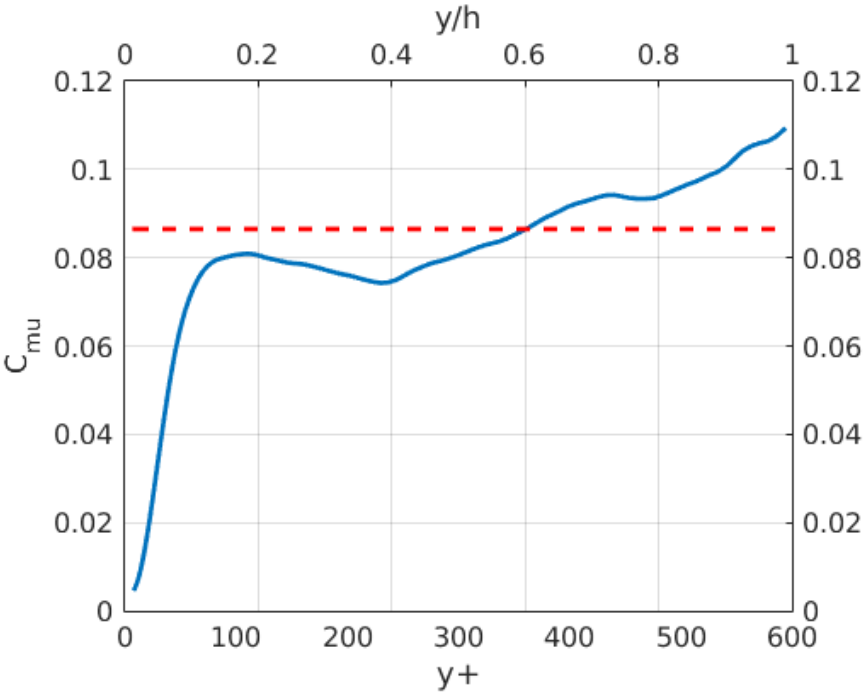

Figure 7: Eddy viscosity coefficient C_μ as a function of distance from wall, calculated using a_{12} and $\overline{S}_{12}$. The average value is 0.086 for $y/h > 0.2$.

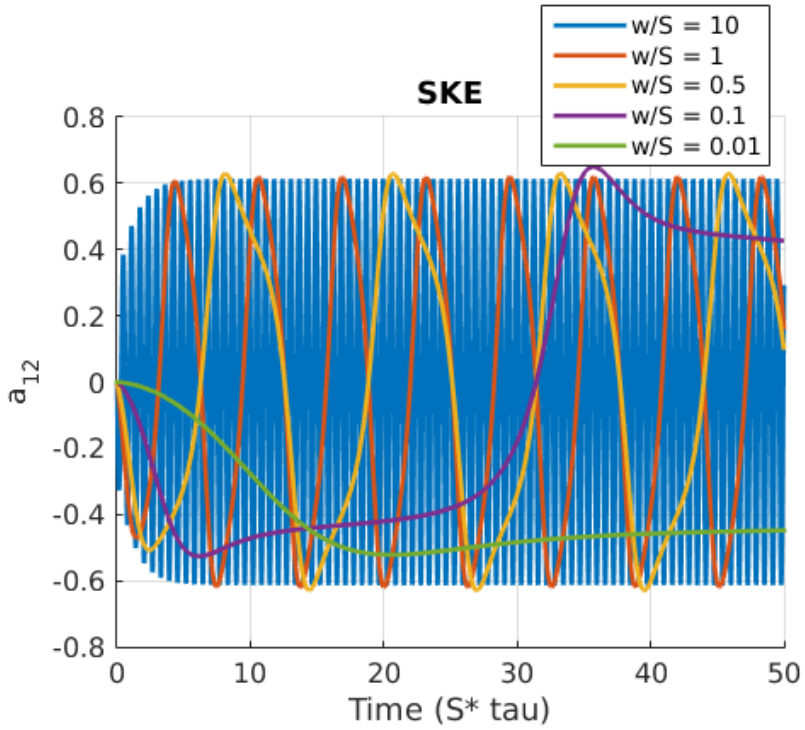

Figure 8: Evolution of a_{12} as a function of time $(S^*\tau)$ using the SKE model for various values of ω/S.

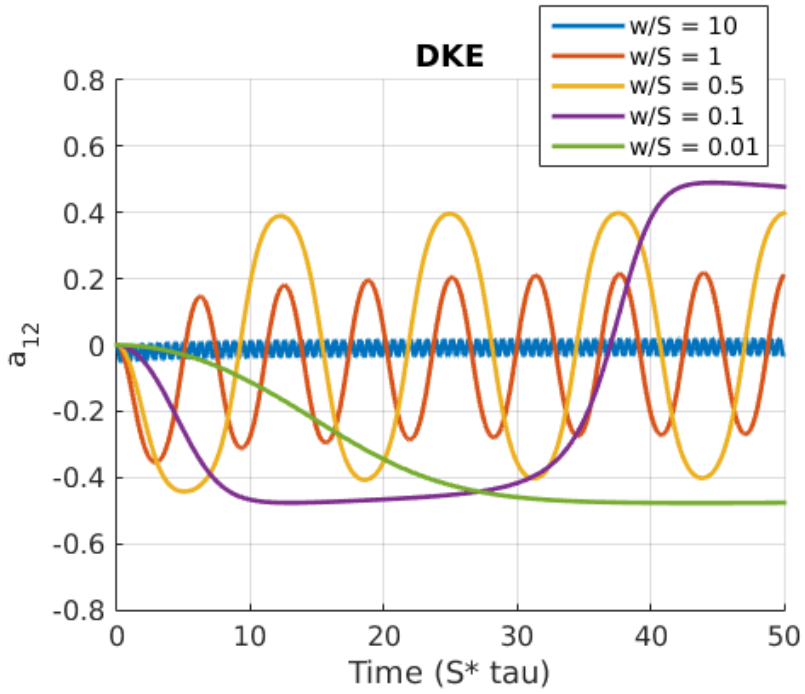

Figure 9: Evolution of a_{12} as a function of time $(S^*\tau)$ using the DKE model for various values of ω/S.

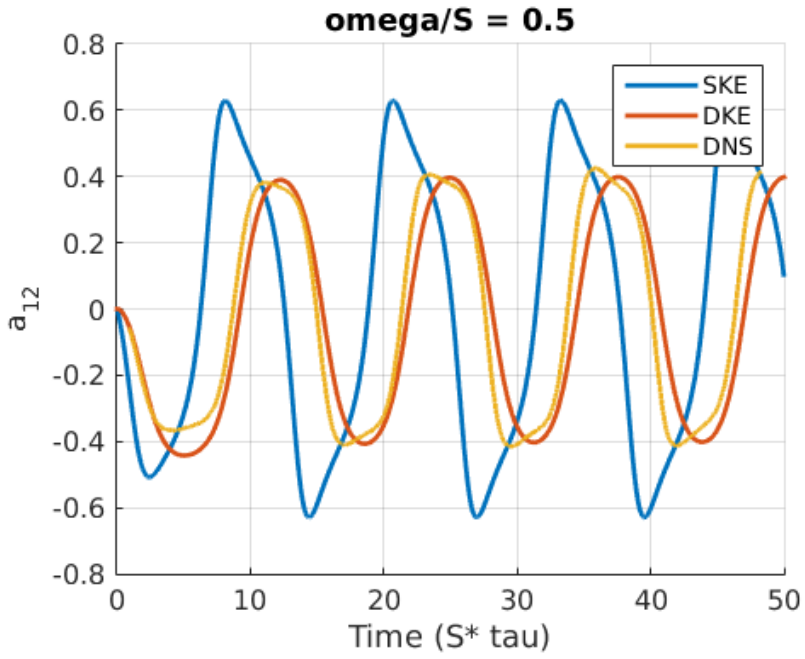

Figure 10: Comparison of SKE and DKE models to DNS data of Yu and Girimaji (2006) for $\omega/S = 0.5$.

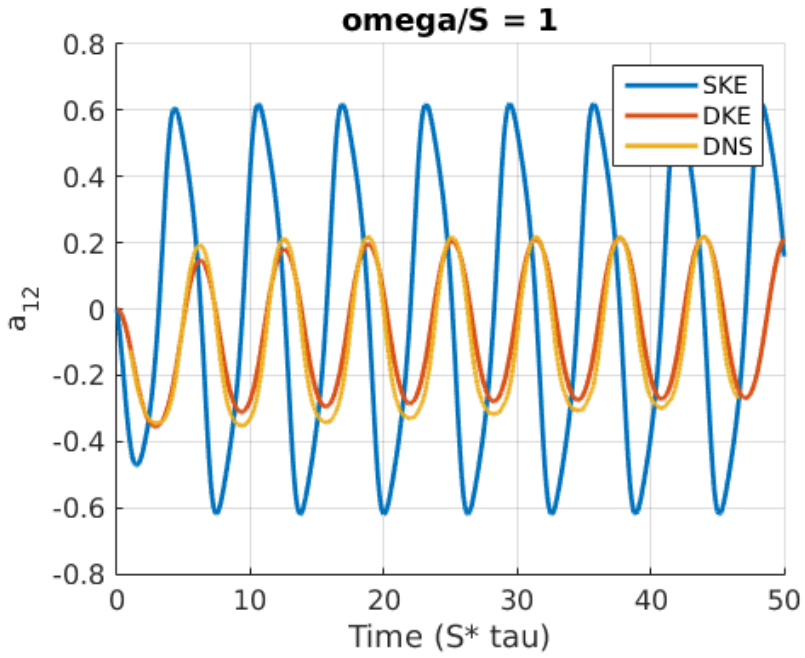

Figure 11: Comparison of SKE and DKE models to DNS data of Yu and Girimaji (2006) for $\omega/S = 1$.

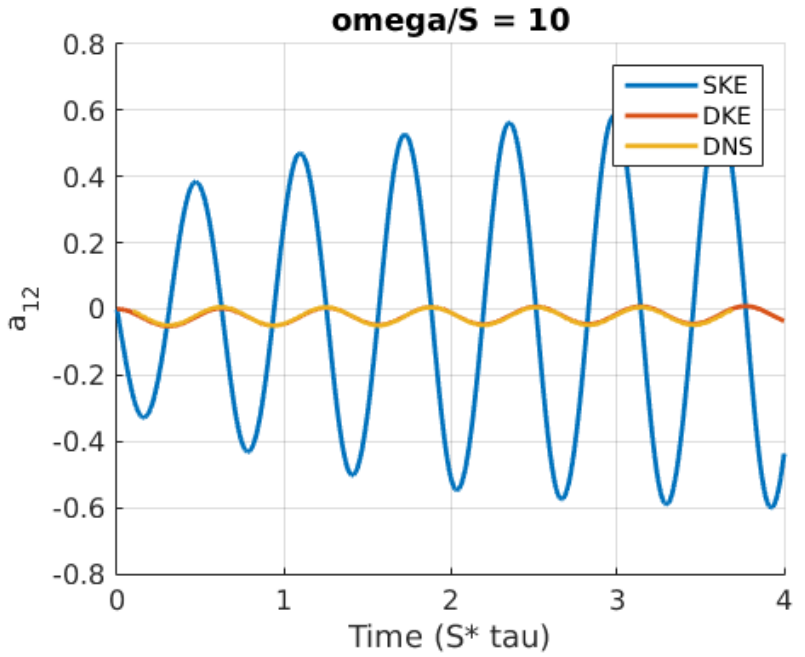

Figure 12: Comparison of SKE and DKE models to DNS data of Yu and Girimaji (2006) for $\omega/S = 10$.

```
% Plot turbulence intensity.
figure();
hold on;
k = zeros(length(data('y+')),1);
for field = {'R_uu','R_vv','R_ww'}
    temp = data(field{1});
    k = k + temp / 2;
end
line(data('y+'),k);
ax1 = gca();
ax2 = axes('Position',ax1.Position,'XAxisLocation','top','YAxisLocation','right');
line(data('y'),k,'Parent',ax2);
hold off;
xlabel(ax1(1),'y+');
xlabel(ax2(1),'y/h');
ylabel(ax1(1),'k');

% Plot dissipation.
figure();
hold on;
line(data('y+'),data('dissip'));
ax1 = gca();
ax2 = axes('Position',ax1.Position,'XAxisLocation','top','YAxisLocation','right');
line(data('y'),data('dissip'),'Parent',ax2);
hold off;
xlabel(ax1(1),'y+');
xlabel(ax2(1),'y/h');
ylabel(ax1(1),'Dissipation');

% Plot anisotropy.
figure();
hold on;
fields = {'R_uu','R_uv','R_uw','R_vv','R_vw','R_ww'};
aniso = zeros(length(data('y')),length(fields));
for i = 1:length(fields)
    aniso(:,i) = data(fields{i}) ./ k;
    if any(strcmp(fields{i},{'R_uu','R_vv','R_ww'}))
        aniso(:,i) = aniso(:,i) - 2/3;
    end
end
fields = {'a_uu','a_uv','a_uw','a_vv','a_vw','a_ww'};
line(repmat(data('y+'),1,length(fields)),aniso);
ax1 = gca();
ax2 = axes('Position',ax1.Position,'XAxisLocation','top','YAxisLocation','right');
line(repmat(data('y'),1,length(fields)),aniso);
hold off;
xlabel(ax1(1),'y+');
xlabel(ax2(1),'y/h');
ylabel(ax1(1),'Anisotropy');
legend(ax2,fields);

% Plot mean strain rate.
figure();
hold on;
fields = {'dUmean/dy','dWmean/dy'};
strain = zeros(length(data('y')),length(fields));
for i = 1:length(fields)
    strain(:,i) = data(fields{i}) / 2;
end
fields = {'Sbar_uv','Sbar_wv'};
line(repmat(data('y+'),1,length(fields)),strain);
ax1 = gca();
ax2 = axes('Position',ax1.Position,'XAxisLocation','top','YAxisLocation','right');
line(repmat(data('y'),1,length(fields)),strain);
hold off;
xlabel(ax1(1),'y+');
xlabel(ax2(1),'y/h');
ylabel(ax1(1),'Mean Strain Rate');
```

```
legend(ax2,fields);

%%%
% Problem 2.3
%%%

% Plot Cmu, and find its average away from the walls.
Cmu = (590 .* data('dissip') .* aniso(:,2)) ./ (2 .* k .* strain(:,1));
cutoff_index = 15;
Cmu = Cmu(cutoff_index:end-1);
y = data('y');
yplus = data('y+');
y = y(cutoff_index:end-1);
yplus = yplus(cutoff_index:end-1);
figure();
hold on;
line(yplus,Cmu);
ax1 = gca();
ax2 = axes('Position',ax1.Position,'XAxisLocation','top','YAxisLocation','right');
line(y,Cmu,'Parent',ax2);
line([min(y),max(y)],ones(2,1)*mean(Cmu(40:end)),'Parent',ax2,'Color','r','LineStyle','--');
fprintf('Problem 2.3: average calculated from y = %5f to be %5f\n',y(40),mean(Cmu(40:end)));
hold off;
xlabel(ax1(1),'y+');
xlabel(ax2(1),'y/h');
ylabel(ax1(1),'C_{mu}');
```

Listing 2: `project_02_part3.m`

```
clear all;
set(groot,'defaultTextInterpreter','none');
set(groot,'defaultLegendInterpreter','none');
set(groot,'defaultLineLineWidth',2);
set(groot,'defaultAxesXGrid','on');
set(groot,'defaultAxesYGrid','on');
set(groot,'defaultAxesFontSize',12);
set(groot,'defaultAxesPosition',[0.15,0.15,0.7,0.7]);

global i A

% Load data from Yu and Girimaji.
dns_data = {};
dns_data{1} = csvread('../data/YuGirimaji_Fig25.txt',1,0);
dns_data{2} = csvread('../data/YuGirimaji_Fig15.txt',1,0);
dns_data{3} = csvread('../data/YuGirimaji_Fig14.txt',1,0);

Cmu = 0.09;
Sstar = 3.3;
options = odeset('RelTol',1e-7);
Sstartau_range = [0,50];
A = [10,1,0.5,0.1,0.01];

figure();
hold on;
for i = 1:length(A)
    % SKE Model
    init_cond = [1,1];
    [tvec,yvec] = ode45(@ode_a12_SKE,Sstartau_range,init_cond,options);
    a12 = -Cmu .* yvec(:,1) ./ yvec(:,2) .* Sstar .* sin(tvec .* A(i));
    plot(tvec,a12,'DisplayName',['w/S = ',num2str(A(i))]);
end
% Annotation
hold off;
xlabel('Time (S* tau)');
ylabel('a_{12}');
title('SKE');
legend(gca,'show','location','northeast');
% Set axis range.
xlim([0,50]);
```

```
ylim([-0.8,0.8]);

figure();
hold on;
for i = 1:length(A)
    % DKE Model
    init_cond = [1,1,0];
    [tvec,yvec] = ode45(@ode_a12_DKE,Sstartau_range,init_cond,options);
    plot(tvec,yvec(:,3),'DisplayName',['w/S = ',num2str(A(i))]);
end
% Annotation
hold off;
xlabel('Time (S* tau)');
ylabel('a_{12}');
title('DKE');
legend(gca,'show','location','northeast');
% Set axis range.
xlim([0,50]);
ylim([-0.8,0.8]);

for i = 1:3
    figure();
    hold on;
    % SKE Model
    init_cond = [1,1];
    [tvec,yvec] = ode45(@ode_a12_SKE,Sstartau_range,init_cond,options);
    a12 = -Cmu .* yvec(:,1) ./ yvec(:,2) .* Sstar .* sin(tvec .* A(i));
    plot(tvec,a12,'DisplayName','SKE');
    % DKE Model
    init_cond = [1,1,0];
    [tvec,yvec] = ode45(@ode_a12_DKE,Sstartau_range,init_cond,options);
    plot(tvec,yvec(:,3),'DisplayName','DKE');
    % DNS Results
    if i<4
        plot(dns_data{i}(:,1),2*dns_data{i}(:,2),'DisplayName','DNS');
    end
    % Annotation
    hold off;
    xlabel('Time (S* tau)');
    ylabel('a_{12}');
    title(['omega/S = ',num2str(A(i))]);
    legend(gca,'show','location','northeast');
    % Set axis range.
    if i==1
        xlim([0,4]);
    else
        xlim([0,50]);
    end
    ylim([-0.8,0.8]);
end
```

Listing 3: `read_MKM_data.m`

```
function [ data ] = read_MKM_data( filename )

% Read in headers.
fid = fopen(filename);
max_scans = 400;
strings = textscan(fid,'%s',max_scans);
fclose(fid);
headers = {};
header_line = 24;
i = 1;
line = 1;
while (line < header_line)
    if strcmp(strings{1}{i},'#')
        line = line + 1;
    end
    i = i + 1;
```

```
end
i = i - 1;
while ~strcmp(strings{1}{i+1},'#')
    headers{length(headers)+1} = strings{1}{i+1};
    i = i+1;
end

% Read in data.
raw_data = dlmread(filename,'\t',header_line+1,0);

% Match headers with data.
for i = 1:length(headers)
    values{i} = raw_data(:,i);
end
data = containers.Map(headers,values);

end
```

Listing 4: `ode_a12_SKE.m`

```
function [ dy ] = ode_a12_SKE(Sstartau,y)

global i A

Cmu = 0.09;
Sstar = 3.3;
Ce1 = 1.44;
Ce2 = 1.92;

k = y(1);
e = y(2);

a12 = -Cmu * (k / e) * Sstar * sin(A(i) * Sstartau);
dy = zeros(2,1);
dy(1) = -k * a12 * Sstar * sin(A(i) * Sstartau) - e;
dy(2) = -Ce1 * e * a12 * Sstar * sin(A(i) * Sstartau) - Ce2 * e^2 / k;

end
```

Listing 5: `ode_a12_DKE.m`

```
function [ dy ] = ode_a12_DKE(Sstartau,y)

global i A

Sstar = 3.3;
Ce1 = 1.44;
Ce2 = 1.92;
C1 = 1.5;
C2 = 0.8;

k = y(1);
e = y(2);
a12 = y(3);

alpha1 = -(k / e) * a12 * Sstar * sin(A(i) * Sstartau) - 1 + C1;
alpha2 = C2 - (4/3);
dy = zeros(3,1);
dy(1) = -k * a12 * sin(A(i) * Sstartau) - e / Sstar;
dy(2) = -Ce1 * e * a12 * sin(A(i) * Sstartau) - Ce2 * e^2 / (k * Sstar);
dy(3) = -alpha1 * (e / k) * a12 / Sstar + 0.5*alpha2 * sin(A(i) * Sstartau);

end
```