Small post processor

1. Contents

- Clear complete workspace
- Read data files
- Set neccessary parameters
- Compute 3D spectrum
- Compute dissipation and turbulent kinetic energy
- Verify computation of kinetic Energy
- Kolmogrov properties
- Compute model spectra
- Compute correlations

2. Clear complete workspace

For Matlab projects the best practise is to clear the complete workspace and the command window also closing all figures might be helpful. In the subsequent you may also define some neccessary flags distinguishing bewteen different data sets.

```
path('./functions',path) % add functions directory the Matlab path
ClearWs();
```

The content of `ClearWs` reads

```
function ClearWs()
    close all
    clear all
    clc

    datadir='data'; % specify the directory containg the data
    flag='3D'; % specify the subdirectory of data
end
```

3. Read data files

Read in the data files and measure the time for reading. The output of the tic/toc block is in seconds. What you should get from the tic/toc block is that most of the time is spend during data I/O. The actual computation needs only ??? compared to the time of the I/O operations. One of the most important part of all postprocessors is to read the files that actually contain your data. Although in terms of speed and size the ASCII data format ist not the prefered choice, for this purpose we will use this format to make ease the understanding. For your information one very famous and highly protable data format is hdf5. Besides its hierarchical structure it is highly optimized for parallel I/O operations which are essential for DNS simulations.

```
[uvel,vvel,wvel,time_read] = ReadData(datadir,flag);
```

The content of `ReadData` reads

```
function [uvel,vvel,wvel,time] = ReadData(datadir,flag)
    tic; % enable timer
    uvel=importdata([datadir,'/',flag,'/uvel']);
    vvel=importdata([datadir,'/',flag,'/vvel']);
    wvel=importdata([datadir,'/',flag,'/wvel']);
    time = toc; % end timer
end
```

4. Set neccessary parameters

latex

- Number of grid points in on direction n_{px}
- Physical length of one direction L_x
- Physical grid spacing $\delta\, x$
- Kinematic viscosity ν

/latex

```
dim=256; % number of points in one dimension
Lx=5e-3; % domain size
Ly=Lx;
Lz=Lx;
dx=Lx/dim; % grid spacing
dy=dx;
dz=dx;
```

```
nu=1.7e-5; % viscosity
u=reshape(uvel,dim,dim,dim); % reshape arrays to have them in 3D
v=reshape(vvel,dim,dim,dim);
w=reshape(wvel,dim,dim,dim);
clear uvel vvel wvel
```

5. Compute 3D spectrum

In order to avoid aliasing effects usually connected with a one dimensional spectrum it is also possible to produce correlations that involve all possible directions. The three dimensional Fourier transformation of such a correlation produces a spectrum that not only depends on a single wavenumber but on the wavenumber vector κ_i. Though the directional information contained in κ_i eliminates the aliasing problem the complexity makes a physical reasoning impossible. For homogeneous isotropic turbulence the situation can be simplified by integrating the three dimensional spectrum over spherical shells.

$$E(\kappa) = \oiint E(\boldsymbol{\kappa}) \mathrm{d}S(\kappa) = \oiint \frac{1}{2} Phi_{ii}(\boldsymbol{\kappa}) \mathrm{d}S(\kappa) \tag{1}$$

Since the surface of a sphereis completly determined by its radius the surface integral can be solved analytically.

$$\oiint ( ) \mathrm{d}S(\kappa) = 4\pi\kappa^2 \cdot ( ) \tag{2}$$

This leads to

$$E(|\kappa|) = \frac{1}{2} \Phi_{ii}(|\boldsymbol{\kappa}|) \tag{3}$$

```
[spectrum,k,mu,mv,mw] = power_spec(u,v,w,Lx);
```

6. Compute dissipation and turbulent kinetic energy

```
[Dissipation,kinetic_E] = spec_prop(spectrum,k,nu);
```

7. Verify computation of kinetic Energy

```
up = sqrt(1/3*(u.^2+v.^2+w.^2));
kinetic_Up = sum(sum(sum(3/2*up.^2)))/size(up,1)^3;
```

8. Kolmogrov properties

```
eta = (nu^3/Dissipation)^(1/4);
u_eta = (nu*Dissipation)^(1/4);
tau = (nu/Dissipation)^(1/2);
```

9. Compute model spectra

Von Karman-Pao Spektren

```
close all
kd = k(end);
ke = pi/Lx/2;
A = 1.5;
up = mean2(up);

VKP1 = A*up^5/Dissipation.*(k./ke).^4./(1+(k./ke).^2).^(17/6).*exp(-3/2*A.*(k./kd).^(4/3));

kd = 1./eta;
VKP2 = 1.5*(k./kd).^(-5/3)./(Dissipation*nu^5)^(-1/4).*exp(-1.5*1.5.*(k./kd).^(4/3));

% Kolmogorov Spektrum
Kolmo=1.5*Dissipation^(2/3)*(k.^(-5/3));

% Plot spectra
h=loglog(k,Kolmo,k,VKP1,k,VKP2,k,spectrum);
set(h,'LineWidth',2);

h=legend('Kolmogorov','VKP1','VKP2','Computed');
set(h,'Location','SouthWest')
```

10. Compute correlations

Computing a correlation can be a tedious work (requireing tremendeous effort) especially if you have large data sets. From theory it is well known that the multiplication of the transform of a data set and its complex conjugate are an accurate representation of the correlation function. Using the FFT approach this gives an enormeous speed advantage. Since we already computed the veloity correlation tensor we may use this result in order to compute the correlation tensor.

$$R_{ij} = \frac{cov(U_i, U_j)}{\sqrt{\sigma_i^2\,\sigma_j^2}} = \frac{(u_i' - \mu_i)\,(u_j - \mu_j)}{\sqrt{\sigma_i^2\,\sigma_j^2}} \tag{4}$$

```
[R11,R22,r,R1,R2,R3]=correlation(u,v,w,Lx);
```