# Spectral analysis of Direct Numerical Simulation data with Matlab

Felix Dietzsch[a]

[a] *Institut für Chemieingenieurwesen und Energieverfahrenstechnik, Fuchsmühlenweg 9, Reiche Zeche, 09599 Freiberg.*

☎ +49 3731 394212, ✉ Felix.Dietzsch@vtc.tu-freiberg.de

## Contents



### 1. Clear complete workspace

If you start a new Matlab project it is always a goot idea to clear the complete workspace and command window. Since Matlab does not automatically open new figure windows each time a plot is invoked we should close all remaining figure windows. Also very important is to tell Matlab where to find the functions we will use in our project. This is done by adding the functions folder with `path('./functions',path)` to the Matlab search path.

```
display('Clear workspace ...')
path('./functions',path) % add directory the Matlab path
close all % close all figures
clear all % clear workspace
clc % clear command window
```

```
datadir='data/stuttgart/'; % data directory
flag='3D';
%set(0,'DefaultFigureWindowStyle','docked')
```

## 2. Read data files

During the evaluation of the function `ReadData` all velocity files necessary for the calculation of the spectrum and the correlation coefficients are read. In addition the import operations are enclosed in a **`tic;... ;toc`** block which measures the time needed for reading the ASCII data. Although the ASCII data format ist not the best choice in terms of speed and size, we will use it since other methodologies require additional knowledge of data processing. Just for your information a very famous and highly portable data format is hdf5. It is a software library that is available for a range of computer platforms, from laptops to massively parallel systems and implements a high-level API (Application programming interface) with C, C++, Fortran 90, and Java interfaces. Besides its hierarchical structure it is highly optimized for parallel I/O operations and can be read by nearly all data processing/visualization tools.

```
display('Read data ...')
[uvel,vvel,wvel,time_read,dim] = ReadData(datadir,flag,...
                                          'uvel',...
                                          'vvel',...
                                          'wvel');
```

```
function [uvel,vvel,wvel,time,dim] = ReadData(datadir,...
                                              flag,...
                                              u_name,...
                                              v_name,...
                                              w_name)
    tic; % enable timer
    uvel=importdata([datadir,'/',u_name]);
    vvel=importdata([datadir,'/',v_name]);
    wvel=importdata([datadir,'/',w_name]);
    time = toc; % end timer
    if strcmp(flag,'3D')
        dim=round((size(uvel,1))^(1/3));
```

```
    end
end
```

### 3. Set necessary parameters

For further computations we have to to define some parmeters of the DNS simulation.

```
display('Set parameters ...')
[u,v,w,Lx,nu]=Params(uvel,vvel,wvel,dim);
```

```
function [u,v,w,Lx,nu]=Params(uvel,vvel,wvel,dim)
    Lx=2*pi; %edge length
    nu=1.7e-5; % viscosity
    u=reshape(uvel,dim,dim,dim); % reshape to 3D
    v=reshape(vvel,dim,dim,dim);
    w=reshape(wvel,dim,dim,dim);
    clear uvel vvel wvel % save memory
end
```

### 4. Compute 3D spectrum

The core of the code is contained in the function **PowerSpec**. It computes the three dimensional energy spectrum from the given velocity fields, obtained from a direct numerical simulation. Although the theoretical analysis is relatively demanding compared to one dimensional spectra its worth investing the effort. The theory of one dimensional spectra relies on the assumption that the propagation of spectral waves ($\kappa_1$) is in the direction of the observed velocity fields or to say it differently one dimensional spectra and correlation functions are Fourier transform pairs. The theory of correlation functions will be discussed in section 7. A key drawback of this theory is that the calculated spectrum has contributions from all wavenumbers $\boldsymbol{\kappa}$, so that the magnitude of $\boldsymbol{\kappa}$ can be appreciably larger than $\kappa_1$. This phenomenon is called aliasing. In order to avoid these aliasing effects 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 $\kappa_i$. Though the directional information contained in $\kappa_i$ eliminates the aliasing problem the complexity makes a physical reasoning impossible. For homogeneous isotropic turbulence the situation can be considerably simplified. From the knowledge that the velocity field is isotropic it can be shown that the velocity spectrum tensor is fully determined by

$$\Phi_{ij}(\boldsymbol{\kappa}) = A(\kappa)\delta_{ij} + B(\kappa)\kappa_i\kappa_j, \tag{1}$$

where $A(\kappa)$ and $B(\kappa)$ are arbitrary scalar functions. Since we assume incompressible fluids (mathematically expressed by $\nabla \cdot u = 0$ or $\kappa_i u_i = 0$ the following condition holds

$$\kappa_i \, \Phi_{ij}(\boldsymbol{\kappa}) = 0. \tag{2}$$

It can be shown that this yields a relation between $A$ and $B$ by means of

$$B(|\boldsymbol{\kappa}|) = -\frac{A(|\boldsymbol{\kappa}|)}{(|\boldsymbol{\kappa}|)^2} \tag{3}$$

In the end this gives a relation between the three dimensional energy spectrum function $E(|\boldsymbol{\kappa}|)$ and the velocity spectrum tensor $\Phi_{ij}$.

$$\Phi_{ij} = \frac{E(|\boldsymbol{\kappa}|)}{4\pi\,(|\boldsymbol{\kappa}|)^2}\left(\delta_{ij} - \frac{\kappa_i\kappa_j}{(|\boldsymbol{\kappa}|)^2}\right) \tag{4}$$

The question is now how the remaining variable ($A$ or $B$) can be determined. Regarding the turbulent kinetic energy we know that

$$k = \int_{-\infty}^{\infty} E(|\boldsymbol{\kappa}|)\,\mathrm{d}k = \sum_{\boldsymbol{\kappa}} E(\boldsymbol{\kappa}) = \sum_{\boldsymbol{\kappa}} \frac{1}{2}\left\langle u^*(\boldsymbol{\kappa})\, u(\boldsymbol{\kappa})\right\rangle = \iiint_{-\infty}^{\infty} \frac{1}{2}\Phi_{ii}(\boldsymbol{\kappa})\,\mathrm{d}\boldsymbol{\kappa}. \tag{5}$$

Comparing the second and last expression we get

$$E(|\boldsymbol{\kappa}|) = \oiint \frac{1}{2}\,\Phi_{ii}(\boldsymbol{\kappa})\,\mathrm{d}S(\kappa). \tag{6}$$

This integral can be solved analytically by utilizing again the assumption of isotropy. For these kind of flows the energy spectrum function can be regarded as the sum of kinetic energy (in wave number space) on different

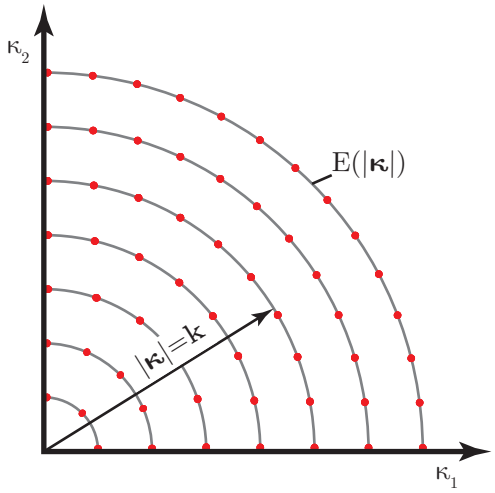


**Fig. 1:** Illustration of the two dimensional shell integration

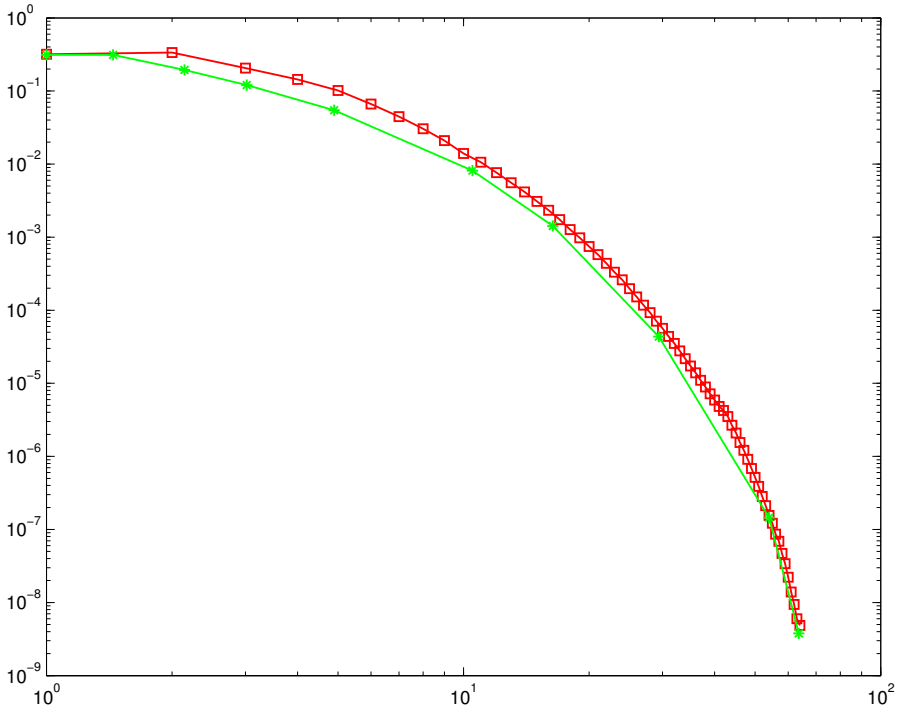


**Fig. 2:** Computed spectrum

energy levels. Each of these energy levels is denoted by a spherical shell in wave number space. Since the surface of a sphere is completely determined by its radius the surface integral can be solved analytically. The idea of this integration is illustrated in Fig. 2. As a result of this one gets

$$E(|\boldsymbol{\kappa}|) = \oiint \frac{1}{2}\,\Phi_{ii}(\boldsymbol{\kappa})\,\mathrm{d}S(\kappa) = 4\pi(|\boldsymbol{\kappa}|)^2\,\Phi_{ii}(|\boldsymbol{\kappa}|). \tag{7}$$

Introducing this relation to equations (1) and (3) one arrives at an expression for the variable $B$.

$$B = -\frac{E(|\boldsymbol{\kappa}|)}{4\pi(|\boldsymbol{\kappa}|)^2} \tag{8}$$

Together with the approximation of the integral of $\Phi$ (equation (5))

$$\iiint_{-\infty}^{\infty} \frac{1}{2}\Phi_{ii}(\boldsymbol{\kappa})\,\mathrm{d}\boldsymbol{\kappa} \approx \frac{1}{2}\sum_{\boldsymbol{\kappa}} \Phi_{ii}(\boldsymbol{\kappa})\,\Delta\kappa_x\,\Delta\kappa_y\,\Delta\kappa_z, \tag{9}$$

where $\Delta\kappa$ refers to the step size in wave number space, the final expression of the three dimensional discrete energy spectrum can be derived.

$$E(|\boldsymbol{\kappa}|) = 2\pi(|\boldsymbol{\kappa}|)^2\frac{\langle u^*(\boldsymbol{\kappa})\,u(\boldsymbol{\kappa})\rangle}{(\Delta\kappa)^3} \tag{10}$$

The calling sequence for the computation of the energy spectrum reads

```
display('Compute spectrum...')
[spectrum,k,bin_counter,time_spec,R,dx] = PowerSpec(u,...
                                                    v,...
                                                    w,...
                                                   Lx,dim);
```

```
function [spectrum,k,bin_counter,time,r,dx] = PowerSpec(u,
    v,w,...
                                                    L,dim)
     tic;
     uu_fft=fftn(u);
     vv_fft=fftn(v);
     ww_fft=fftn(w);
```

```
7       if mod(dim,2)==0
8           muu = abs(uu_fft)/length(u)^3;
9           mvv = abs(vv_fft)/length(v)^3;
10          mww = abs(ww_fft)/length(w)^3;
11
12          muu = muu.^2;
13          mvv = mvv.^2;
14          mww = mww.^2;
15
16          k_end = (dim)/2;
17      else
18          rx=[0:1:dim-1] - (dim-1)/2;
19          ry=[0:1:dim-1] - (dim-1)/2;
20          rz=[0:1:dim-1] - (dim-1)/2;
21
22          R_x=circshift(rx',[(dim+1)/2 1]);
23          R_y=circshift(ry',[(dim+1)/2 1]);
24          R_z=circshift(rz',[(dim+1)/2 1]);
25
26          k_end = (dim-1)/2;
27      end
28
29      rx=[0:1:dim-1] - (dim)/2+1;
30      ry=[0:1:dim-1] - (dim)/2+1;
31      rz=[0:1:dim-1] - (dim)/2+1;
32
33      R_x=circshift(rx',[(dim)/2+1 1]);
34      R_y=circshift(ry',[(dim)/2+1 1]);
35      R_z=circshift(rz',[(dim)/2+1 1]);
36
37      [X,Y,Z]= meshgrid(R_x,R_y,R_z);
38      r=(sqrt(X.^2+Y.^2+Z.^2));
39
40      dx=2*pi/L;
41      k=[1:k_end].*dx;
42
43      spectrum=zeros(size(k,2),1);
44      bin_counter=zeros(size(k,2),1);
45      for N=2:k_end-1
46          picker = (r(:,:,:)*dx <= (k(N+1) + k(N))/2) & ...
47                   (r(:,:,:)*dx > (k(N) + k(N-1))/2);
48          spectrum(N) = sum(muu(picker))+...
49                        sum(mvv(picker))+...
50                        sum(mww(picker));
51          bin_counter(N) = size(find(picker==1),1);
```

```
52      end
53      % compute first value of spectrum
54      picker = (r(:,:,:)*dx <= (k(2) + k(1))/2);
55      spectrum(1) = sum(muu(picker))+...
56                    sum(mvv(picker))+...
57                    sum(mww(picker));
58      bin_counter(1) = size(find(picker==1),1);
59      % compute last value of spectrum
60      picker = (r(:,:,:)*dx > (k(end) + k(end-1))/2 & ...
61                r(:,:,:)*dx <= k(end));
62      spectrum(end) = sum(muu(picker))+...
63                      sum(mvv(picker))+...
64                      sum(mww(picker));
65      bin_counter(end) = size(find(picker==1),1);
66      %compute final spectrum
67      spectrum = spectrum*2*pi.*k'.^2./(bin_counter.*dx.^3);
68      time=toc;
69
70      y = [k; spectrum'];
71      fid = fopen('spectrum.dat', 'w');
72      fprintf(fid, '%10.2E %10.2E\n', y);
73      fclose(fid);
74  end
```

## 5. Compute dissipation and turbulent kinetic energy

The function **SpecProp** calculates the kinetic energy both from the velocities and the previously computed spectrum. The latter one is calculated by using relation (5) and incorporating equation (13)

$$k \approx \sum_{\boldsymbol{\kappa}} \frac{E(|\boldsymbol{\kappa}|)}{4\pi\,(|\boldsymbol{\kappa}|)^2}\,\Delta\kappa_x\,\Delta\kappa_y\,\Delta\kappa_z \tag{11}$$

Knowing that the underlying flow field is isotropic this triple sum may be reduced to a single one by

$$k \approx \sum_{\boldsymbol{\kappa}} \frac{E(|\boldsymbol{\kappa}|)}{4\pi\,(|\boldsymbol{\kappa}|)^2}\,\Delta\kappa_x\,\Delta\kappa_y\,\Delta\kappa_z = \sum_{|\boldsymbol{\kappa}|} E(|\boldsymbol{\kappa}|)\cdot\Delta|\boldsymbol{\kappa}| \tag{12}$$

A second integral, approximated in the same manner, gives the value of the Dissipation

$$\varepsilon = \iiint_{-\infty}^{\infty} 2(|\boldsymbol{\kappa}|)^2 \nu \, \frac{1}{2} \Phi_{ii}(\boldsymbol{\kappa}) \, \mathrm{d}\boldsymbol{\kappa} \approx \frac{1}{2} \sum_{\boldsymbol{\kappa}} 2(|\boldsymbol{\kappa}|)^2 \nu \, \Phi_{ii}(\boldsymbol{\kappa}) \, \Delta\kappa_x \, \Delta\kappa_y \, \Delta\kappa_z \tag{13}$$

$$\approx \sum_{|\boldsymbol{\kappa}|} 2(|\boldsymbol{\kappa}|)^2 \nu \, E(|\boldsymbol{\kappa}|) \, \Delta(|\boldsymbol{\kappa}|) \tag{14}$$

where $\nu$ refers to the kinematic viscosity. The calling sequence reads

```
display('Compute kinetic energy...')
[Dissipation,kin_Sp,kin_Ph,kin_E,up] = SpecProp(spectrum,k,...
                                                nu,u,v,w,...
                                                dim,dx);
```

```
function [Dissipation,kin_Sp,kin_Ph,kin_E,up] = ...
                                    SpecProp(E,k,nu,...
                                             u,v,w,...
                                             dim,dx)
%     kin_Sp = trapz(k,E);
    kin_Sp = sum(E)*dx;
    Dissipation = sum(2*nu.*k.^2.*E')*dx;
%     Dissipation = trapz(k,2*nu.*k.^2.*E');
    up = sqrt(1/3/dim^3*sum(sum(sum(u.^2+v.^2+w.^2))));
    kin_Ph = sum(sum(sum(0.5*(u.^2+v.^2+w.^2))))/length(u)
        ^3;
    kin_E=sum(sum(sum(0.5*(fftn(u).*conj(fftn(u))+...
                           fftn(v).*conj(fftn(v))+...
                           fftn(w).*conj(fftn(w))))...
                /size(u,1)^6));
end
```

## 6. Kolmogorov properties

According to the Kolomogorov hypotheses the length scale $\eta$, characteristic velocity $u_\eta$ and the characteristic time scale $\tau$ of the smallest swirls in the

flow are computed within the function **KolmoScale**. From a dimensionality analysis Kolmogorov derived

$$\eta = \left(\frac{\nu^3}{\varepsilon}\right)^{1/4}, \tag{15}$$

$$u_\eta = (\varepsilon\,\nu)^{1/4}, \tag{16}$$

$$\tau_\eta = \left(\frac{\nu}{\varepsilon}\right)^{1/2}. \tag{17}$$

For further reading concerning his theory it is refered to Pope [1], Hinze [2] and Tennekes and Lumley [3].

```
display('Compute Kolmogorov scales...')
[eta,u_eta,tau]=KolmoScale(nu,Dissipation);
```

```
function [eta,u_eta,tau]=KolmoScale(nu,Dissipation)
    eta = (nu^3/Dissipation)^(1/4);
    u_eta = (nu*Dissipation)^(1/4);
    tau = (nu/Dissipation)^(1/2);
end
```

## 7. Compute correlations

From a general perspective correlation functions are a measure of how much two physical quantities are connected. So how is this helpful for the analysis of turbulent flows? For seemingly chaotic and random processes it would be beneficial if we had a measure of how the velocity at point $A$ is influenced by the velocity at point $B$. A maybe more intuitive quantity that can be calculated from the correlation functions is the integral length scale which gives a measure of the largest eddies in the flow. In fluid dynamics one generally differentiates between two forms of correlation functions, the longitudinal and the transversal or lateral correlation function. The difference between both forms is illustrated in figure 3. In general the correlation between two components of an isotropic homogeneous velocity field is expressed by

$$R_{ij} = \langle u_i(\mathbf{x}+r)\,u_j(\mathbf{x})\rangle\,. \tag{18}$$


**(a)** Longitudinal correlation **(b)** Transversal correlation

**Fig. 3:** Illustration of different correlation functions

The correlation coefficients are computed by normalizing $R_{ij}$ with the square root of the product of the two variances $\sigma_i^2$ and $\sigma_j^2$.

$$r_{ij} = \frac{\langle u_i(\mathbf{x}+\mathbf{r})\, u_j(\mathbf{x})\rangle}{\sqrt{\sigma_i^2\, \sigma_j^2}} \tag{19}$$

As illustrated in figure 3 the longitudinal and lateral correlation depend on the direction of $\mathbf{r}$, i.e.

$$f(r) = \frac{\langle u_1(\mathbf{x}+r\mathbf{e}_1)\, u_1(\mathbf{x})\rangle}{\sigma_1} \tag{20}$$

$$g(r) = \frac{\langle u_2(\mathbf{x}+r\mathbf{e}_1)\, u_2(\mathbf{x})\rangle}{\sigma_2} \tag{21}$$

From several theoretical analysis it is well known that correlations can be efficiently computed by means of multiplying the Fourier transform of a quantity with its complex conjugate. Using the FFT approach this gives an enormeous speed advantage.

$$R_{ij} = \mathfrak{F}^{-1}\left\{\mathfrak{F}\left\{u_i\right\}^* \cdot \mathfrak{F}\left\{u_j\right\}\right\} \tag{22}$$

```
display('Compute Correlations...')
[R11,R22,r,R1,R2,R3]=Correlation(u,v,w,Lx,dim);
```

The content of `Correlation` reads

```
1 function [R11,R22,r,R1,R2,R3]=Correlation(u,v,w,Lx,dim)
2     scaling = 1;
```

```
%     NFFT = 2.^nextpow2(size(u)); %power of 2 fitting
    length of u
%     u_fft=fftn(u,NFFT)./scaling;
%     NFFT = 2.^nextpow2(size(v));
%     v_fft=fftn(v,NFFT)./scaling;
%     NFFT = 2.^nextpow2(size(w));
%     w_fft=fftn(w,NFFT)./scaling;

    u_fft=fftn(u)./scaling;
    v_fft=fftn(v)./scaling;
    w_fft=fftn(w)./scaling;

    Rij_x=(u_fft.*conj(u_fft)); % compute velo.
        correlation tensor
    Rij_y=(v_fft.*conj(v_fft));
    Rij_z=(w_fft.*conj(w_fft));

% %     % x-component
% %     NFFT = 2.^nextpow2(size(u_fft));
% %     R1=ifftn(Rij_x,NFFT)/std2(u)^2/dim^3;
% %
% %     % y-component
% %     NFFT = 2.^nextpow2(size(v_fft));
% %     R2=ifftn(Rij_y,NFFT)/std2(v)^2./dim^3;
% %     % z-component
% %     NFFT = 2.^nextpow2(size(w_fft));
% %     R3=ifftn(Rij_z,NFFT)/std2(w)^2./dim^3;
        % x-component
    R1=ifftn(Rij_x)/std2(u)^2/dim^3;
    R2=ifftn(Rij_y)/std2(v)^2./dim^3;
    R3=ifftn(Rij_z)/std2(w)^2./dim^3;

    NFFT=size(u_fft,1);
    R11 = (reshape(R3(1,1,:),NFFT(1),1)+R2(1,:,1)'+R1
        (:,1,1))/3;
    R11 = R11(1:size(u_fft)/2+1);
    %
    R1_22 = (R1(1,:,1)+R3(1,:,1))/2;
    R2_22 = (R2(:,1,1)+R3(:,1,1))/2;
    R3_22 = (reshape(R1(1,1,:),size(u_fft,1),1)+...
             reshape(R2(1,1,:),size(u_fft,1),1))/2;

    R22 = (R1_22'+R2_22+R3_22)/3;
    R22 = R22(1:size(u_fft)/2+1);

```

```
45      r = linspace(0,Lx/2,size(u_fft,1)/2+1)/(Lx/2);
46  end
```

## References


[1] S. B. Pope, Turbulent Flows, Cambridge University Press, 1 edition, 2000.

[2] J. O. Hinze, Turbulence, Mcgraw-Hill College, 2 edition, 1975.

[3] H. Tennekes, J. L. Lumley, A first course in turbulence, The MIT Press, 1972.