V.7.1 Anisotropic errors (optional)

You plotted this error estimate also for the channel flow, see Section U.8. You can use the first and second-order derivatives that you computed in Section V.6. The Δx and Δy needed in Eq. U.6 can be computed as

```
dx=diff(xp2d);
dx=repmat(dx,[1 1 nk-1]);
dx(ni,:)=dx(ni-1,:);

dy=diff(yp2d,1,2);
dy(:,nj)=dy(:,nj-1);
dy=repmat(dy,[1 1 nk-1]);
```

(Δz is constant). Only two sets of velocity fields are loaded, `u1_pans_iddes.mat`, `...w2_pans_iddes.mat`. When everything works, use more velocity fields (possibly all eight) in order to get better statistics.

V.7.2 Two-point correlations

In Item 5 in Section V.7 you should compute two-point correlation. To do that, you will use data from another simulation of the hump flow presented in [171, 174]. Download the

- Pythons file `pl_twocorr_computed_fk.py` or
- the Matlab/Octave file `pl_twocorr_computed_fk.m`.

They load a new grid with 305×109 grid points. The 3D grid has 32 cells in the spanwise direction ($z_{max} = 0.2$). `pl_twocorr_computed_fk` will also load time histories of the spanwise velocity $\bar{v}_3$ at $x = 0.65, 0.8, 1.1$ and $x = 1.3$ (files `w_time_z65`, `w_time_z80`, `w_time_z110` and `w_time_z130`). These files include time histories at cell center $j = 10, 30, 50, 70, 90$ in the wall-normal direction at 16 positions in the x_3 direction. The time series are re-arranged into 3D arrays `w_y_z_t_65(j,k,t)`, `w_y_z_t_80(j,k,t)`, `w_y_z_t_110(j,k,t)` and `w_y_z_t_130(j,k,t)` where index j, k, t denote wall-normal direction, spanwise direction and time, respectively.

Now, use the time series to compute two-point correlations using the formulas given in Section 10.1. Equation 10.3 gives the two-point correlation of v_1' at two points separated in the x_1 direction. In your case, you will compute the two-point correlation of v_3' at two points separated in the x_3 direction, i.e. $B_{33}^{norm}(x_3^A, \hat{x}_3)$. Plot it at all x_1 positions ($x_1 = 0.65, 0.8, 1.1$ and 1.3) and at a couple of x_2 locations. Then compute the integral lengthscale, see Eq. 10.6. A good approximation of the integral lengthscale is usually the point in the two-point correlation where $B_{33}^{norm} \simeq 0.2$ (i.e. $\mathcal{L} \simeq \hat{x}_3$ where $B_{33}^{norm} \simeq 0.2$). Check if it is good approximation.

In [128, 129] it is recommended that in a good LES (or in the LES region of a DES/hybrid LES-RANS), the integral lengthscale should cover 8-16 cells, depending on how accurate the user wants her/his LES/DES to be.

Here are some hints on how to compute the two-point correlation. Start by computing it for one y location (=0 in Python and =2 in Matlab/Octave) and one z separation, $\hat{z} = 2\Delta z$). In Python

```
B33=0
```

```
k=2
for n in range(1,N): # time average
  B33=B33+w[0,0,n]*w[0,0+k,n]/N
```

and in Matlab/Octave

```
B33=0;
k=2;
for n=1:N % time average
  B33=B33+w(2,1,n)*w(2,1+k,n)/N;
end
```

where $k = 2$ (because the separation is $\hat{z} = 2\Delta z$) and N is the number of time steps. This gives $B_{33}(2\Delta z)$. Next, do it for $\hat{z} = 3\Delta z$, i.e. (in Python)

```
B33=0
k=3
for n in range(1,N): # time average
  B33=B33+w[0,0,n]*w[0,0+k,n]/N
```

This gives $B_{33}(3\Delta z)$. Now, do it for all $k = 0, 1, \ldots 15$. And then normalize by w_{rms}^2 (or, easier, simply by `B33[0]` (Python) or `B33(1)` (Matlab/Octave)). Check that it is correct by plotting $B_{33}(\hat{z})$ versus $\hat{z}$. It should look someting like the two-point correlation in Fig. 10.1. Finally, compute the two-point correlation for all five y locations and four x locations and compute the integral length scale.