Flow Noise Modelling

October 12, 2016

In [1]:
```
import px4tools
import pandas
from mpl_toolkits.mplot3d import axes3d
import numpy as np
%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

In [2]:
```
data = {}
h_array = np.array([2, 4, 6, 8])
v_array = np.array([0.5, 1])
i_file = 0
files = ['./2016-10-10/11_48_20.csv',
         './2016-10-10/11_52_47.csv',
         './2016-10-10/11_56_28.csv',
         './2016-10-10/12_07_30.csv',
         './2016-10-10/12_10_43.csv',
         './2016-10-10/12_12_48.csv',
         './2016-10-10/12_15_17.csv',
         './2016-10-10/12_20_05.csv']
for v in v_array:
    for h in h_array:
        data[(v, h)] = px4tools.process_data(pandas.read_csv(
        files[i_file]))
        i_file += 1
```

In [3]:
```
def find_vy_mean_var(data, vel, cut=0.2, plot=False):
    dt = 0.1
    #vx = abs(data.LPOS_Z*data.FLOW_RawX/dt).rolling(1).mean()
    vy = abs(data.LPOS_Z*data.FLOW_RawY/dt ).rolling(1).mean() - vel
    if plot:
        figure()
        #vx.plot()
        vy.plot()

    if plot:
        figure()
```

```
            plot(data.LPOS_Y, data.LPOS_X)
            axis('equal')

        if plot:
            figure()
            vy.rolling(1).mean().plot()
            hlines(0.2, 0, vy.index.values[-1])

        vy_cut = vy[vy > cut]
        return vy_cut.mean(), sqrt(vy_cut.var())
```

In [4]:
```
stats = {}

for v in v_array:
    for h in h_array:
        mean, std = find_vy_mean_var(data[(v, h)], v)
        print('v', v, 'h', h, 'mean', round(mean, 10),
              'std', round(std, 10))
        stats[(v, h)] = {
            'mean': mean,
            'std': std
            }
```

```
v 0.5 h 2 mean 0.2974806109 std 0.0700954508
v 0.5 h 4 mean 0.4184100531 std 0.1871010859
v 0.5 h 6 mean 0.3508781026 std 0.164670356
v 0.5 h 8 mean 0.4035160134 std 0.2092591146
v 1.0 h 2 mean 0.2533118083 std 0.0343062799
v 1.0 h 4 mean 0.5521159468 std 0.2819285461
v 1.0 h 6 mean 0.6667623689 std 0.4268459759
v 1.0 h 8 mean 0.6947426203 std 0.4736582281
```

In [5]:
```
x = []
y = []
for h in h_array:
    for v in v_array:
        x.append(np.array([h, h**2, v, v*h, v*h**2]))
        y.append(stats[v, h]['std'])
x = np.array(x)
y = np.array(y)
p, res, _, _ = np.linalg.lstsq(x, y)
p, res
```

Out[5]:
```
(array([ 0.04005232, -0.00656446, -0.26265873,  0.13686658, -0.00397357]),
 array([ 0.00373951]))
```

In [6]:
```
n_v = 10
n_h = 10
```

```
y_fit = np.zeros((n_h, n_v))
v_grid = linspace(0.5, 1, n_v)
h_grid = linspace(2, 8, n_h)
for j, v in enumerate(v_grid):
    for i, h in enumerate(h_grid):
        y_fit[i, j] = \
            p[0]*h + p[1]*h**2 +  p[2]*v +  p[3]*h*v + p[4]*v*h**2
v_mesh, h_mesh = meshgrid(v_grid, h_grid)
```

In [7]:
```
fig = figure(figsize=(10,5))
ax = fig.add_subplot(111, projection='3d')
points = np.vstack([[(v, h, stats[(v, h)]['std']) for v in v_array] for h i
ax.plot3D(*points, 'r.', label='data')
X, Y, Z = axes3d.get_test_data(0.05)
ax.plot_wireframe(v_mesh, h_mesh, y_fit, rstride=1, cstride=1, label='model
ax.set_xlabel('velocity, m/s')
ax.set_ylabel('agl, m')
ax.set_zlabel('$\sigma, m$')
legend(loc='best', ncol=2)
```

Out[7]: <matplotlib.legend.Legend at 0x7ff03569d080>