analytic_example_v2

November 5, 2019

[1]:
```
%matplotlib inline

import filtering
from datetime import timedelta
import numpy as np
import scipy.signal as signal
import matplotlib.pyplot as plt
from netCDF4 import Dataset
```

Construct a simple analytic flow involving a mean U_0, wave U_w and eddy ψ_e flow:
$U = U_0 + U_w \sin kx + \frac{\partial \psi_e}{\partial y}$
$V = \frac{fU_w}{kU_0} \cos kx - \frac{\partial \psi_e}{\partial x}$
where
$\psi_e = U_e L_e \exp\left(-((x - U_0 t)^2 + (y - y_0)^2)/L_e^2\right)$
The wave is propagating against the flow, and is thus trapped and steady ($kU_0 > f$). A common example of this is a lee wave.

[2]:
```
U0=0.2
Uw=0.01
Ue=0.01
f=5e-5
k=1e-3
Le=10e3;

# Periodic grids
Lx=8*2*np.pi/k
x=np.linspace(0,Lx,200,endpoint=False)
y=np.copy(x)
ym=np.mean(y);
# Time - 2 weeks
t=np.array(range(0,(2*7*24*60**2),3600))

T,Y,X=np.meshgrid(t,y,x,indexing='ij')

# estimate of the time taken to advect 1 grid cell
(x[2]-x[1])/U0
# estimate of "frequency" of eddy
```

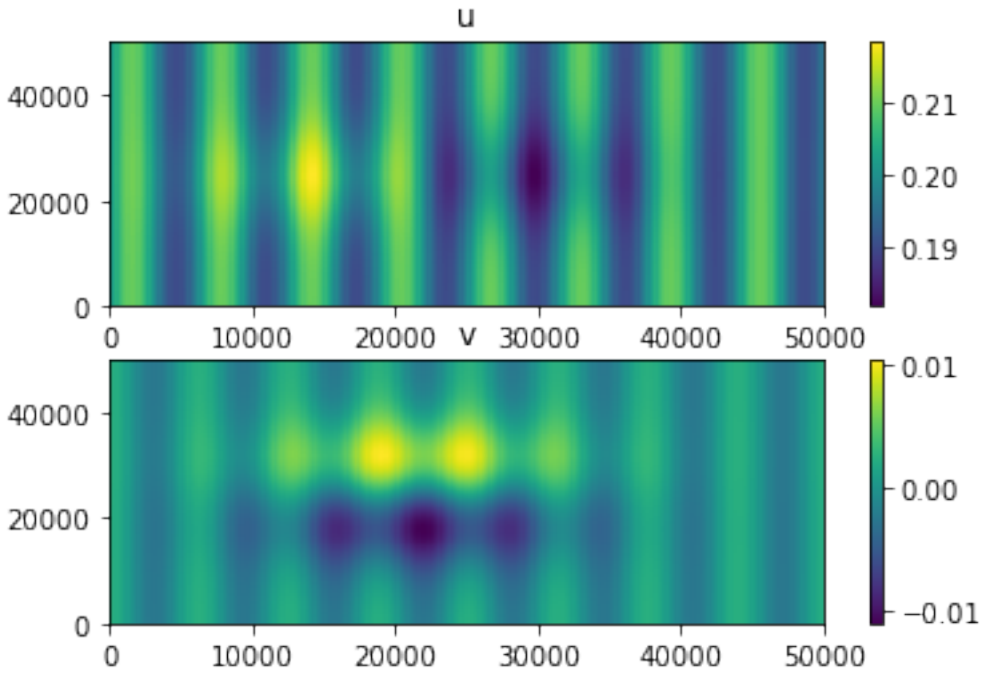
```
U0/Le
```

[2]:
```
2e-05
```

[3]:
```
# Generate the flow, accounting for periodicity
xc=(U0*t % Lx)[:,None,None]
ut1=-2*Ue/Le*(X-xc)*np.exp(-((X-xc)**2+(Y-ym)**2)/Le**2)
vt1=2*Ue/Le*(Y-ym)*np.exp(-((X-xc)**2+(Y-ym)**2)/Le**2)
ut2=-2*Ue/Le*(X-(xc+x[-1]))*np.exp(-((X-(xc+x[-1]))**2+(Y-ym)**2)/Le**2)
vt2=2*Ue/Le*(Y-ym)*np.exp(-((X-(xc+x[-1]))**2+(Y-ym)**2)/Le**2)
ut3=-2*Ue/Le*(X+(-xc+x[-1]))*np.exp(-((X+(-xc+x[-1]))**2+(Y-ym)**2)/Le**2)
vt3=2*Ue/Le*(Y-ym)*np.exp(-((X+(-xc+x[-1]))**2+(Y-ym)**2)/Le**2)
ue=(ut1+ut2+ut3)
ve=(vt1+vt2+vt3)
# total flow
U=U0+Uw*np.sin(k*X)+ue
V=f*Uw/(k*U0)*np.cos(k*X)+ve
```

[4]:
```
ti=100;
plt.subplot(2,1,1)
plt.pcolor(x,y,U[ti,:,:])
plt.title('u')
plt.colorbar()

plt.subplot(2,1,2)
plt.pcolor(x,y,V[ti,:,:])
plt.title('v');
plt.colorbar()
```

[4]:
```
<matplotlib.colorbar.Colorbar at 0x7f2410c5ec50>
```



Now we write this test data to a netCDF before filtering

[5]:
```
ncfile_name="analytic_example_input.nc"
ncfile = Dataset(ncfile_name, 'w', format='NETCDF4')
ncfile.description = 'Synthetic data for Lagrangian Filtering test'

# dimensions
ncfile.createDimension('time', None)
ncfile.createDimension('x', np.size(x))
ncfile.createDimension('y', np.size(y))

# variables
time = ncfile.createVariable('time', 'f8', ('time',))
x1 = ncfile.createVariable('x', 'f8', ('x',))
y1 = ncfile.createVariable('y', 'f8', ('y',))
U1 = ncfile.createVariable('u', 'f8', ('time', 'y', 'x'))
V1 = ncfile.createVariable('v', 'f8', ('time', 'y', 'x'))
```

[6]:
```
U1[:] = U
V1[:] = V
time[:] = t
x1[:] = x
y1[:] = y

ncfile.close()
```

Now it's time to filter...

[7]:
```
filenames = {
        "U": ncfile_name,
    "V": ncfile_name
}
variables = {"U": "u", "V": "v"}
dimensions = {"lon": "x", "lat": "y", "time": "time"}

ff = filtering.LagrangeFilter(
        "analytic_example_out1", filenames, variables, dimensions,
        sample_variables=["U","V"], mesh="flat",
        window_size=timedelta(days=3.5).total_seconds(),
    highpass_frequency=f
)

# periodic domain
ff.make_zonally_periodic()
ff.make_meridionally_periodic()

ff.advection_dt=600
t1=timedelta(days=7).total_seconds()
ff.filter(times=[t1])
```

```
WARNING: Casting lon data to np.float32
WARNING: Casting lat data to np.float32
WARNING: Casting depth data to np.float32
INFO: Compiled SamplingParticlesample_kernel ==> /tmp/tmpqv6vife5/kernel.so
INFO: Compiled SamplingParticleAdvectionRK4sample_kernel ==>
/tmp/tmpt44vv90k/kernel.so
INFO: Compiled SamplingParticleAdvectionRK4_zonally_periodic_BCsample_kernel ==>
/tmp/tmpjn3f1cbv/kernel.so
INFO: Compiled SamplingParticleAdvectionRK4_doubly_periodic_BCsample_kernel ==>
/tmp/tmpcxwie2mb/kernel.so
WARNING: dt or runtime are zero, or endtime is equal to Particle.time. The
kernels will be executed once, without incrementing time
```

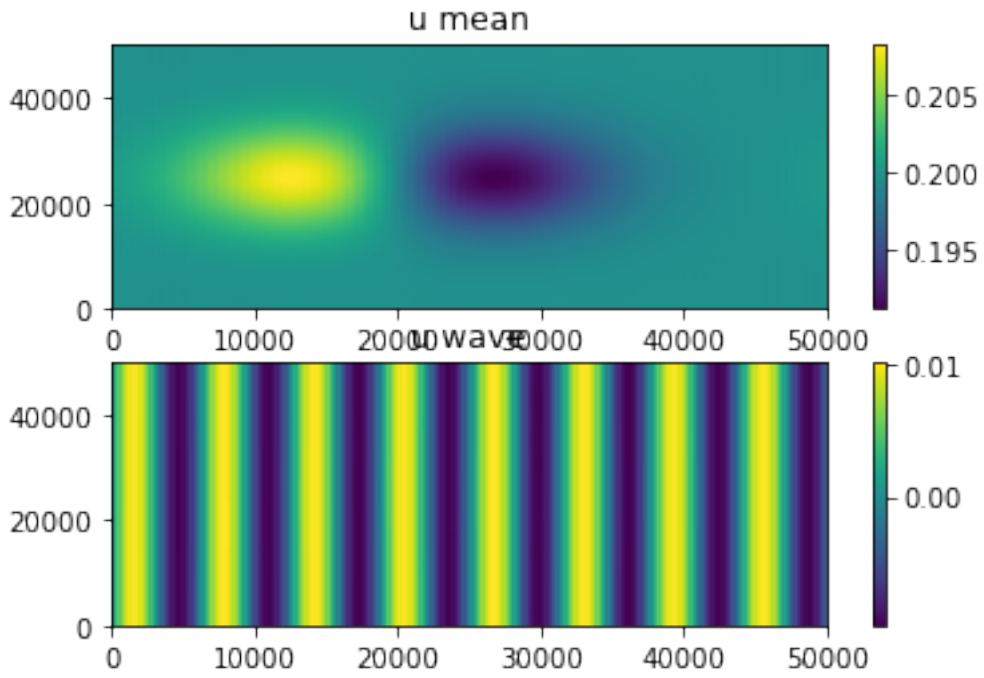
Read data back in and plot the mean and wave parts

[8]:
```
ncfile_out_name="analytic_example_out1.nc"
ncfile = Dataset(ncfile_out_name, 'r', format='NETCDF4')
wave_U=ncfile.variables['var_U'][0,:,:]
wave_V=ncfile.variables['var_V'][0,:,:]
ncfile.close()
```

[9]:
```
plt.subplot(2,1,1)
plt.pcolor(x,y,U[7*24-1,:,:]-wave_U[:])
plt.title('u mean')
plt.colorbar()

plt.subplot(2,1,2)
```

```
plt.pcolor(x,y,wave_U[:])
plt.title('u wave');
plt.colorbar()
```

[9]: <matplotlib.colorbar.Colorbar at 0x7f23b777ca50>



We would like to compare the Lagrgangian filter to an Eulerian filter. First step is to make a zero velocity field.

[10]:
```
ncfile_name0="analytic_example_zeros.nc"
ncfile = Dataset(ncfile_name0, 'w', format='NETCDF4')
ncfile.description = 'Synthetic data for Lagrangian Filtering test'

# dimensions
ncfile.createDimension('time', None)
ncfile.createDimension('x', np.size(x))
ncfile.createDimension('y', np.size(y))

# variables
time = ncfile.createVariable('time', 'f8', ('time',))
x1 = ncfile.createVariable('x', 'f8', ('x',))
y1 = ncfile.createVariable('y', 'f8', ('y',))
U1 = ncfile.createVariable('u', 'f8', ('time', 'y', 'x'))
V1 = ncfile.createVariable('v', 'f8', ('time', 'y', 'x'))

U1[:] = U*0
```

```
V1[:] = V*0
time[:] = t
x1[:] = x
y1[:] = y

ncfile.close()
```

Now filter with the same settings. The wave and mean flows are incorrectly identified in this case.

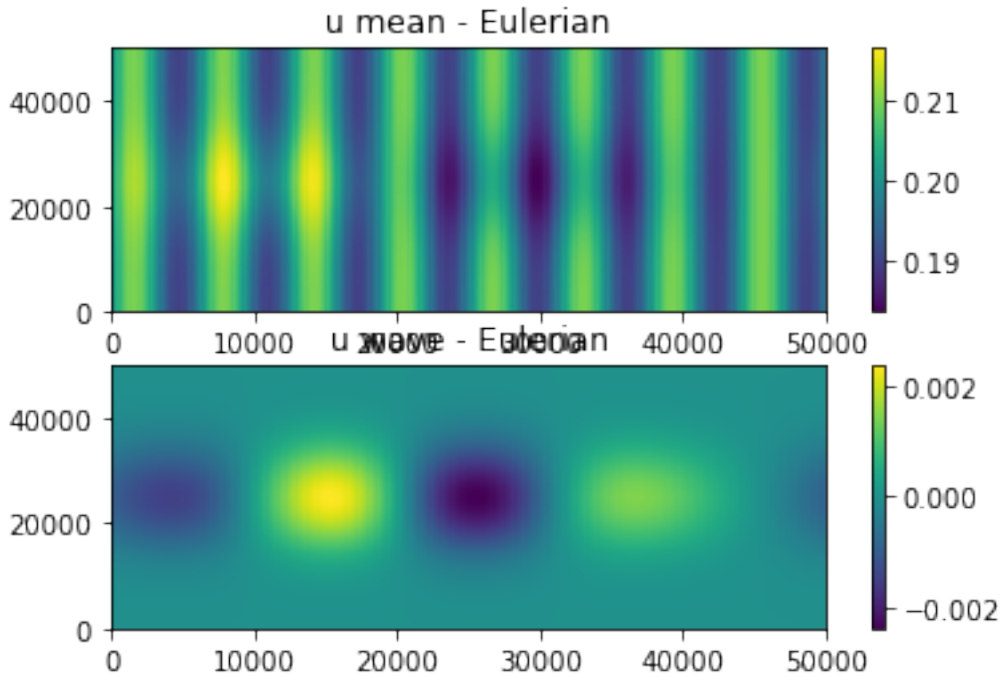
[11]:
```
filenames = {
    "U": ncfile_name0,
    "V": ncfile_name0,
        "U2": ncfile_name,
    "V2": ncfile_name
}
variables = {"U": "u", "V": "v","U2": "u", "V2": "v"}
dimensions = {"lon": "x", "lat": "y", "time": "time"}

ff = filtering.LagrangeFilter(
        "analytic_example_zeros1", filenames, variables, dimensions,
        sample_variables=["U2","V2"], mesh="flat",
        window_size=timedelta(days=3.5).total_seconds(),
    highpass_frequency=f
)

# periodic domain
ff.make_zonally_periodic()
ff.make_meridionally_periodic()

ff.advection_dt=600
t1=timedelta(days=7).total_seconds()
ff.filter(times=[t1])
```

```
INFO: Compiled SamplingParticlesample_kernel ==> /tmp/tmpxdnls9f2/kernel.so
INFO: Compiled SamplingParticleAdvectionRK4sample_kernel ==>
/tmp/tmpyi9b8w_0/kernel.so
INFO: Compiled SamplingParticleAdvectionRK4_zonally_periodic_BCsample_kernel ==>
/tmp/tmp8xf5pqe4/kernel.so
INFO: Compiled SamplingParticleAdvectionRK4_doubly_periodic_BCsample_kernel ==>
/tmp/tmpe6sr_wey/kernel.so
```

[12]:
```
ncfile_out_name="analytic_example_zeros1.nc"
ncfile = Dataset(ncfile_out_name, 'r', format='NETCDF4')
waveE_U=ncfile.variables['var_U2'][0,:,:]
waveE_V=ncfile.variables['var_V2'][0,:,:]
ncfile.close()
```

[13]:
```
plt.subplot(2,1,1)
plt.pcolor(x,y,U[7*24-1,:,:]-waveE_U[:])
plt.title('u mean - Eulerian')
plt.colorbar()

plt.subplot(2,1,2)
plt.pcolor(x,y,waveE_U[:])
plt.title('u wave - Eulerian');
plt.colorbar()
```

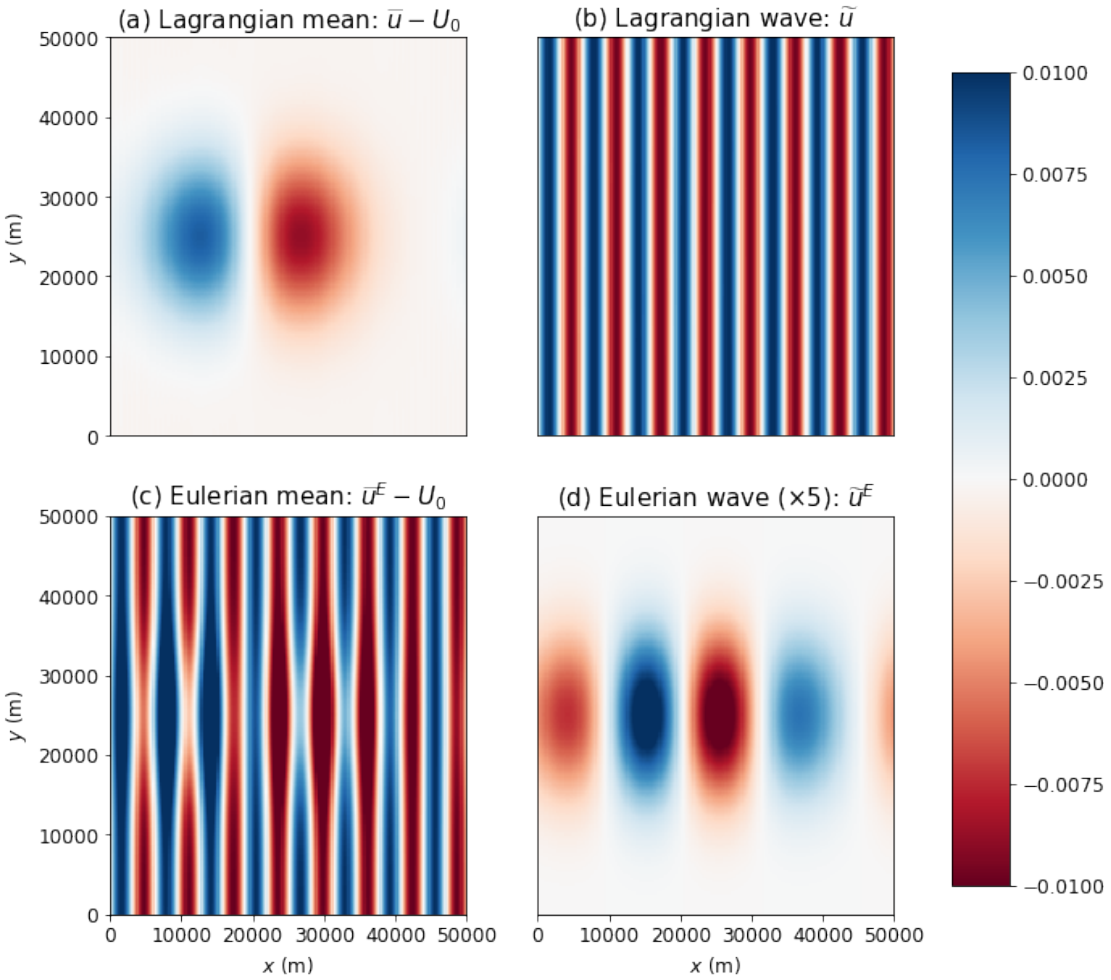
[13]:
```
<matplotlib.colorbar.Colorbar at 0x7f23b4593cd0>
```



Final plot for this example:

[14]:
```
fig1=plt.figure(figsize=(10,10))
plt.subplot(2,2,1)
plt.pcolor(x,y,U[7*24-1,:,:]-wave_U[:]-U0,vmin=-0.01,vmax=0.01,cmap='RdBu')
plt.title(r'(a) Lagrangian mean: $\widebar{u}-U_0$',fontsize=15)
plt.xticks([])
plt.tick_params(labelsize=12)
plt.ylabel(r'$y$ (m)',fontsize=12)

plt.subplot(2,2,2)
plt.pcolor(x,y,wave_U[:],vmin=-0.01,vmax=0.01,cmap='RdBu')
plt.title(r'(b) Lagrangian wave: $\widetilde{u}$',fontsize=15)
plt.xticks([])
```

```
plt.yticks([])
plt.tick_params(labelsize=12)

plt.subplot(2,2,3)
plt.pcolor(x,y,U[7*24-1,:,:]-waveE_U[:]-U0,vmin=-0.01,vmax=0.01,cmap='RdBu')
plt.title(r'(c) Eulerian mean: $\widebar{u}^E-U_0$',fontsize=15)
plt.tick_params(labelsize=12)
plt.ylabel(r'$y$ (m)',fontsize=12)
plt.xlabel(r'$x$ (m)',fontsize=12)

plt.subplot(2,2,4)
im=plt.pcolor(x,y,5*waveE_U[:],vmin=-0.01,vmax=0.01,cmap='RdBu')
plt.title('u wave');
plt.title(r'(d) Eulerian wave ($\times$5): $\widetilde{u}^E$',fontsize=15);
plt.yticks([])
plt.tick_params(labelsize=12)
plt.xlabel(r'$x$ (m)',fontsize=12)

plt.subplots_adjust(right=0.8)
cbar_ax = fig1.add_axes([0.85, 0.15, 0.05, 0.7])
plt.colorbar(im, cax=cbar_ax)
plt.tick_params(labelsize=12)
```


(a) Lagrangian mean: $\overline{u} - U_0$
(b) Lagrangian wave: $\widetilde{u}$
(c) Eulerian mean: $\overline{u}^E - U_0$
(d) Eulerian wave (×5): $\widetilde{u}^E$
y (m)
x (m)
0
10000
20000
30000
40000
50000
0.0100
0.0075
0.0050
0.0025
0.0000
−0.0025
−0.0050
−0.0075
−0.0100