Lab 7b

EOSC 511

Christopher Rodell

Question 1.

Hand-in an answer to question 7b from the lab itself.

Problem Seven

-b) For grid C, write down the finite difference form of the shallow water equations. Shallow Water Equations

$$u(x_i, y_j, t_n), v(x_i, y_j, t_n), h(x_i, y_j, t_n)$$

Full Equations, Eqn 1

$$\frac{\partial u}{\partial t} - fv = -g\frac{\partial h}{\partial x}$$
$$\frac{\partial u}{\partial t} - fv = -g\frac{h_{(i+\frac{1}{2}),j,n} - h_{(i-\frac{1}{2}),j,n}}{\Delta x}$$
$$\frac{u_{i,j,(n+\frac{1}{2})} - u_{i,j,(n-\frac{1}{2})}}{\Delta t} - fv = -g\frac{h_{(i+\frac{1}{2}),j,n} - h_{(i-\frac{1}{2}),j,n}}{\Delta x}$$

Full Equations, Eqn 2

$$\frac{\partial v}{\partial t} - fu = -g\frac{\partial h}{\partial y}$$
$$\frac{\partial v}{\partial t} - fv = -g\frac{h_{i,(j+\frac{1}{2}),n} - h_{i,(j-\frac{1}{2}),n}}{\Delta y}$$
$$\frac{v_{i,j,(n+\frac{1}{2})} - v_{i,j,(n-\frac{1}{2})}}{\Delta t} - fu = -g\frac{h_{i,(j+\frac{1}{2}),n} - h_{i,(j-\frac{1}{2}),n}}{\Delta y}$$

Full Equations, Eqn 3

$$\frac{\partial h}{\partial t} + H\frac{\partial u}{\partial x} + H\frac{\partial v}{\partial y} = 0$$
$$\frac{dh}{dt} + H\frac{u_{(i+\frac{1}{2}),j,n} - u_{(i-\frac{1}{2}),j,n}}{\Delta x} + H\frac{v_{i,(j+\frac{1}{2}),n} - v_{i,(j-\frac{1}{2}),n}}{\Delta y} = 0$$
$$\frac{h_{i,j,(n+\frac{1}{2})} - h_{i,j,(n-\frac{1}{2})}}{\Delta t} + H\frac{u_{(i+\frac{1}{2}),j,n} - u_{(i-\frac{1}{2}),j,n}}{\Delta x} + H\frac{v_{i,(j+\frac{1}{2}),n} - v_{i,(j-\frac{1}{2}),n}}{\Delta y} = 0$$

The finite difference approximation to the full shallow water equations on the C grid:

$$\frac{u_{i,j,(n+\frac{1}{2})} - u_{i,j,(n-\frac{1}{2})}}{\Delta t} - fv = -g\frac{h_{(i+\frac{1}{2}),j,n} - h_{(i-\frac{1}{2}),j,n}}{\Delta x}$$
$$\frac{v_{i,j,(n+\frac{1}{2})} - v_{i,j,(n-\frac{1}{2})}}{\Delta t} - fu = -g\frac{h_{i,(j+\frac{1}{2}),n} - h_{i,(j-\frac{1}{2}),n}}{\Delta y}$$

$$\frac{h_{i,j,(n+\frac{1}{2})} - h_{i,j,(n-\frac{1}{2})}}{\Delta t} + H\frac{u_{(i+\frac{1}{2}),j,n} - u_{(i-\frac{1}{2}),j,n}}{\Delta x} + H\frac{v_{i,(j+\frac{1}{2}),n} - v_{i,(j-\frac{1}{2}),n}}{\Delta y} = 0$$

Question 2

You will use the interactive1.py code in numlabs/lab7 for this question. If your experience with the Coriolis force is minimal, you can chose the "small" option. See the doc string for reasonable parameters. The code solves the high water level in the center (similar to rain.py) in two-dimensions with periodic boundary conditions on a flat bottom. The depth is set in the functions find depth*. Use grid-C and edit the find depth3 function.

- a) Choose an interesting but smooth topography (remembering that the domain is periodic in both space dimensions). Implement it in find depth3 correctly given the grid-C staggering.

  (SEE interactive1_cr.py)

- b) Run your new code. Discuss any other changes you make to the code. You may want to change what and when it plots.

  I have made a new terrain that is a sin function with 2 full periods across the x-axis of the domain. The sin function is centered at H0 (1000 meters) with an amplitude of 600 meters. See 3D plot for graphical depiction.

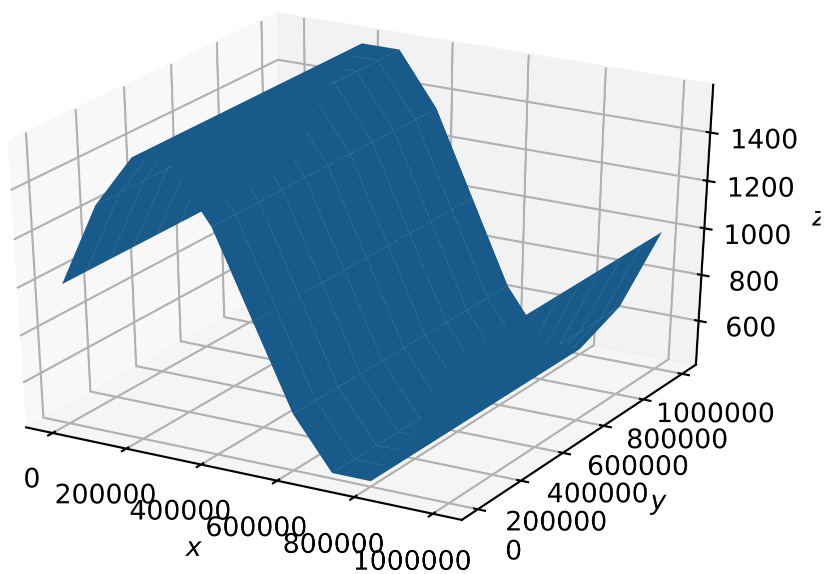
  I have also altered the plotting function to show how u, v, eta, and velocity changes through time. I would have liked to have animated this but I wasn't able to get that functioning.

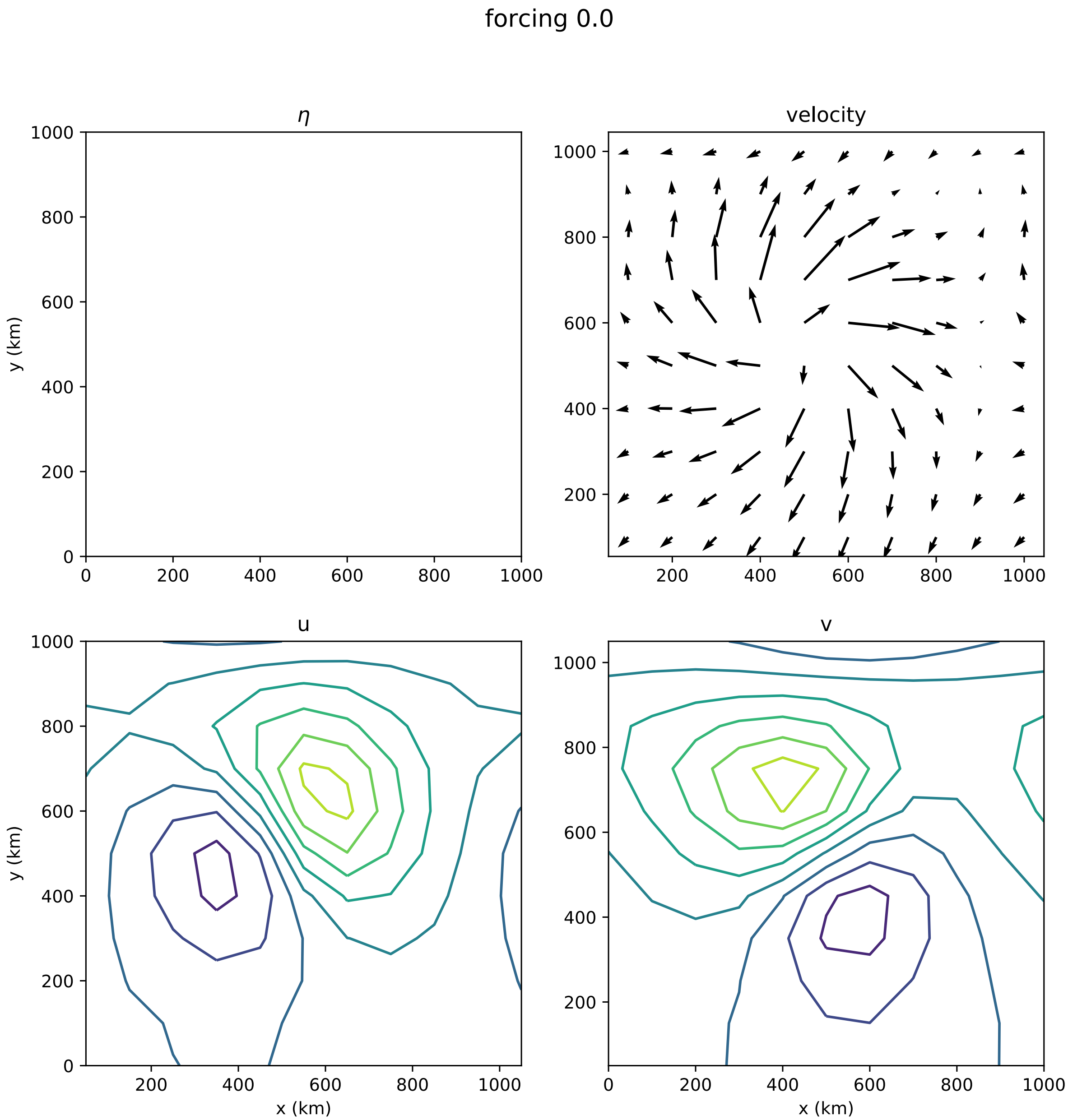
  I have blocked out (with ###) the portion of code I altered in interactive1_cr.py. The two functions altred w the find_depth3 and interactive1

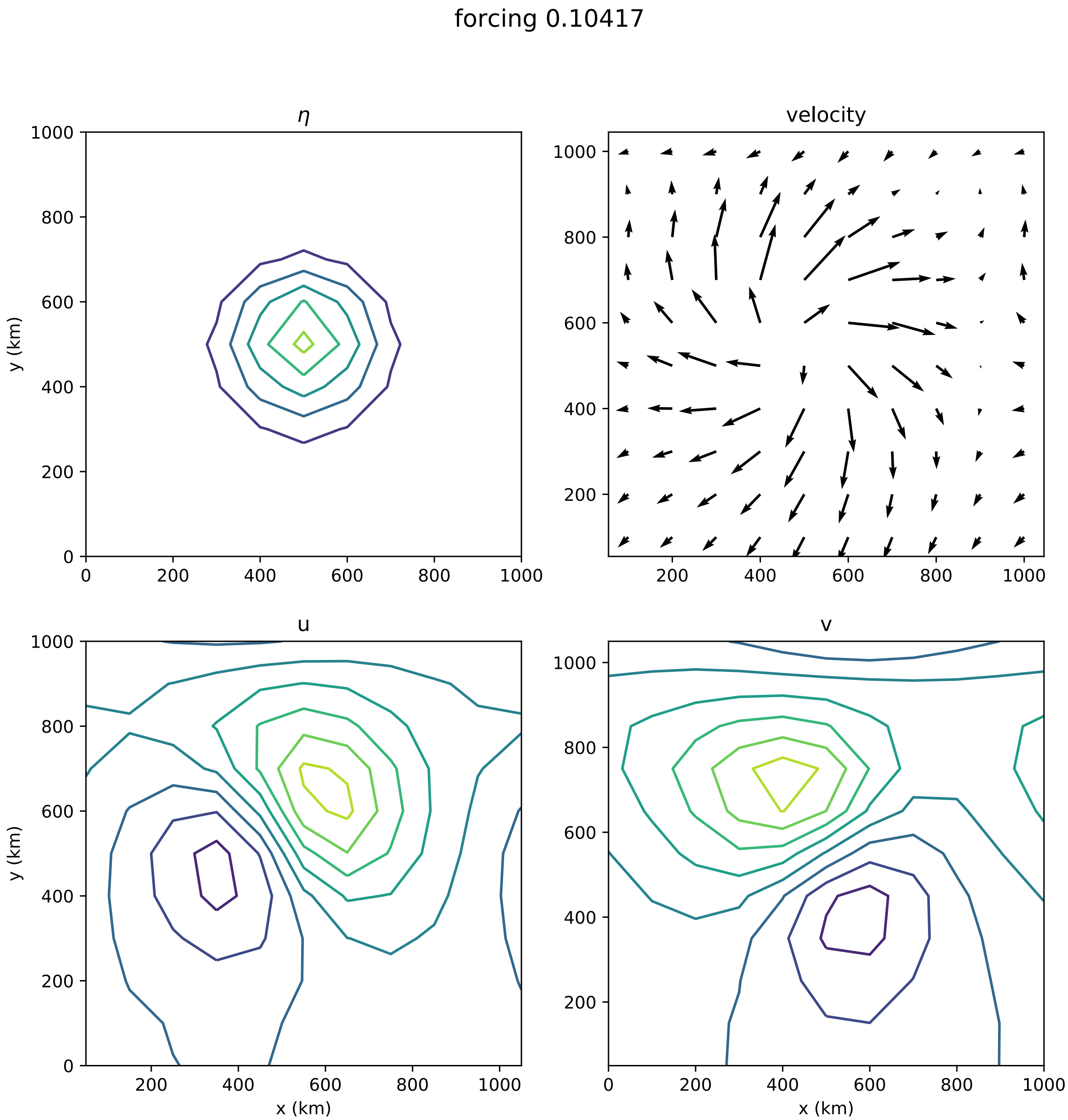
- c) Explain the differences that the bottom topography makes. The new topography has altered all u, v, eta, and velocity. For u component, you can see slower velocities where the sin wave is increasing vertically "upslope" and on the downslope of the sin wave, you can see u velocities increasing. v is nearly u but transposed 90 deg (or pi/2) in a counterclockwise direction. Eta is fun as the high point has shifted off-center and is now at the crest of the sin wave. Anf the velocity field is off-center towards the crest of the sin wave, with a bit stronger magnitudes

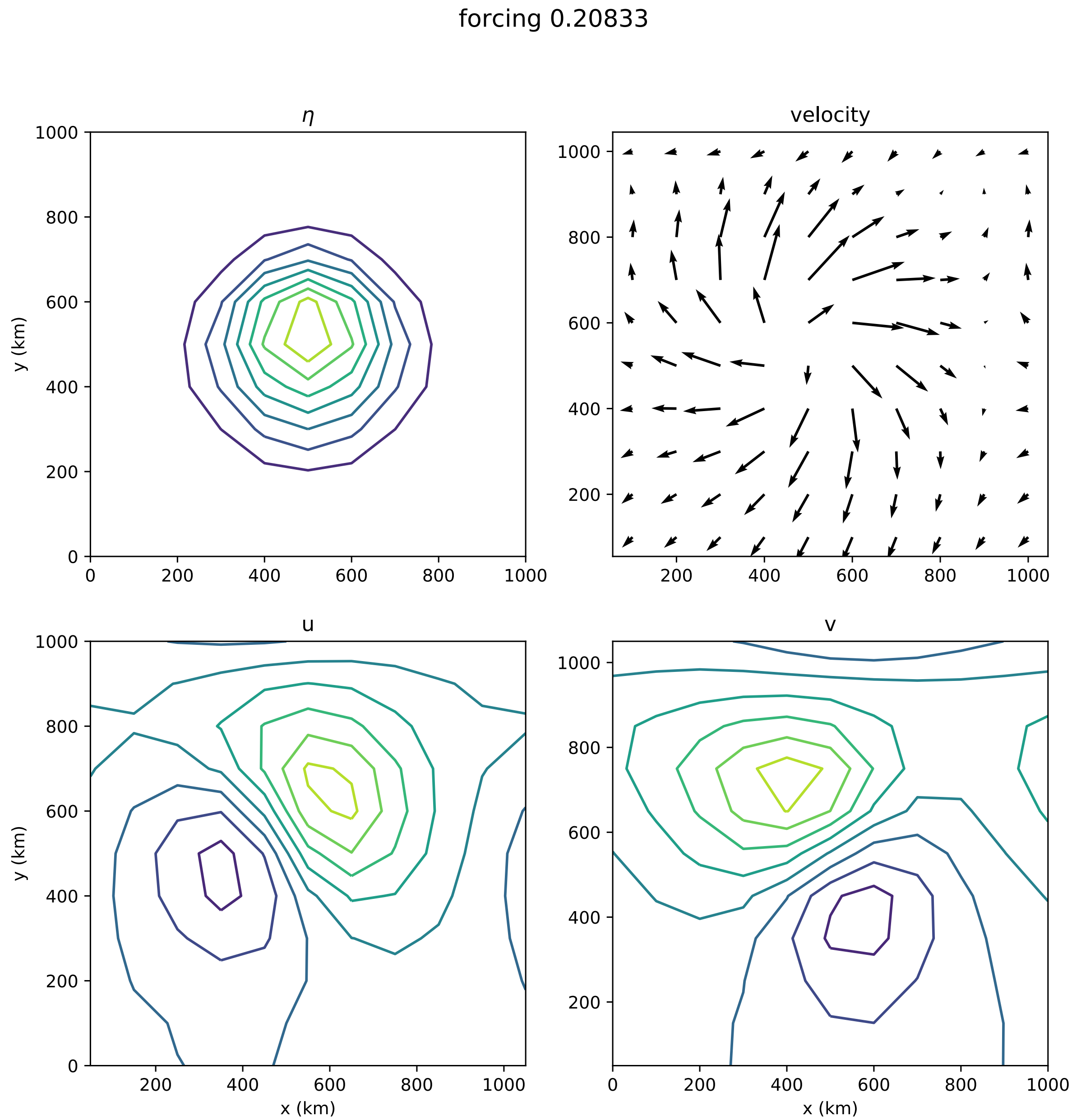
In [1]:
```
import context
from IPython.display import Image
import IPython.display as display
import matplotlib.pyplot as plt
# %matplotlib
import numpy as np

# import the 2-dimensional drop solvers
from numlabs.lab7 import interactive1_cr  ## My Modifiied verison
from numlabs.lab7 import interactive1     ## The original version

plt.close('all')
# grid A is grid 1, grid B is grid 2 and and grid C is grid 3
# ngrid is the number of grid points in x and y
# dt is the time step in seconds
# T is the time plotted is seconds to 4*3600 is 4 hours

## Make Plots with my version
interactive1_cr.interactive1(grid=3, ngrid=11, dt=150, T=4*3600)

## Make plot original version
interactive1.interactive1(grid=3, ngrid=11, dt=150, T=4*3600)

plt.show('all')
```

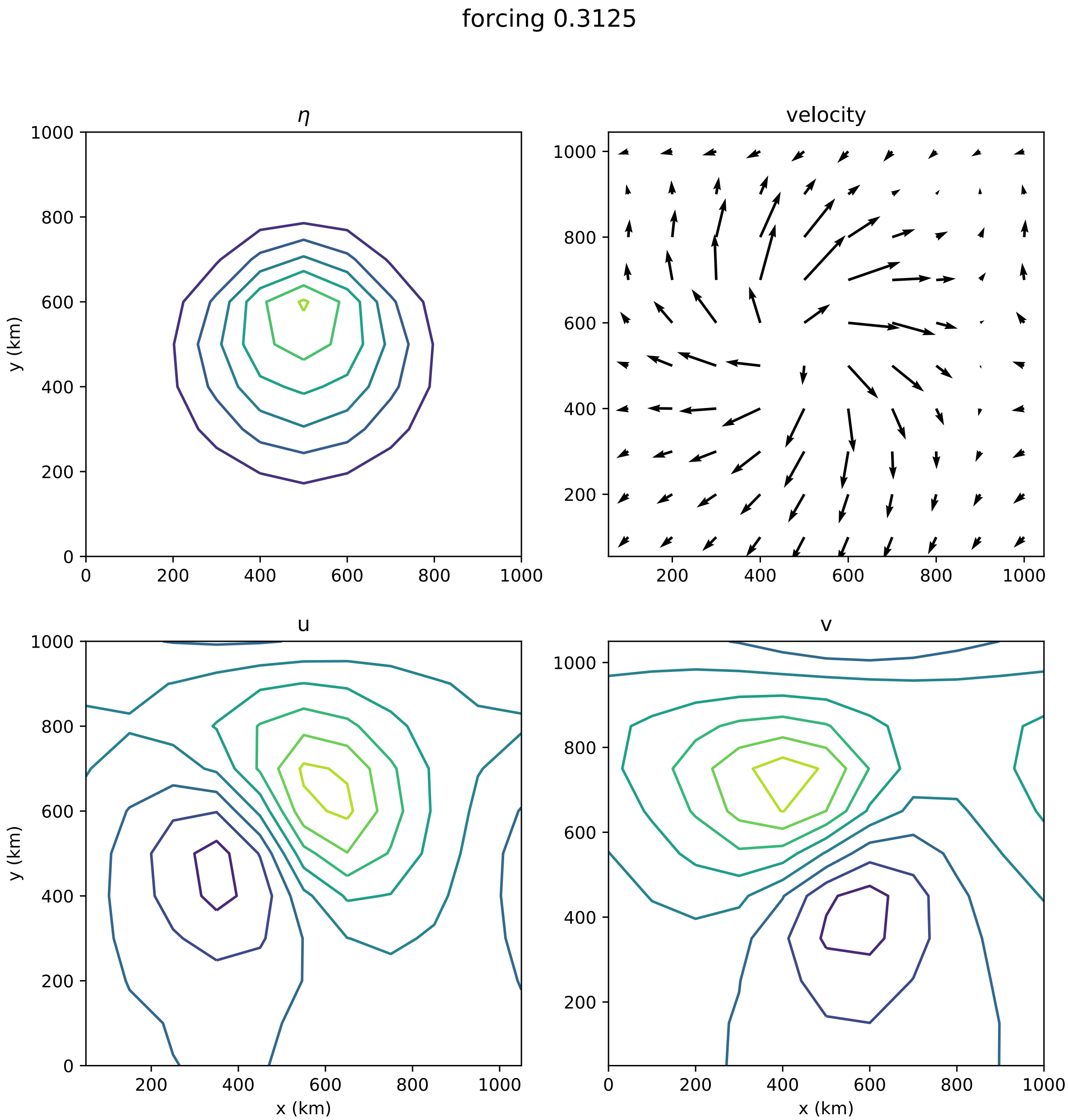
```
*****************************
context imported. Front of path:
/Users/rodell/repos/numeric_students
back of path: /Users/rodell/.ipython
*****************************

through /Users/rodell/repos/numeric_students/numeric_notebooks/lab7/con
text.py
Sin wave Heights 1600 to 400
dx/dt 666.667

sqrt(g*H0) 100.000

Flat Bottom
```

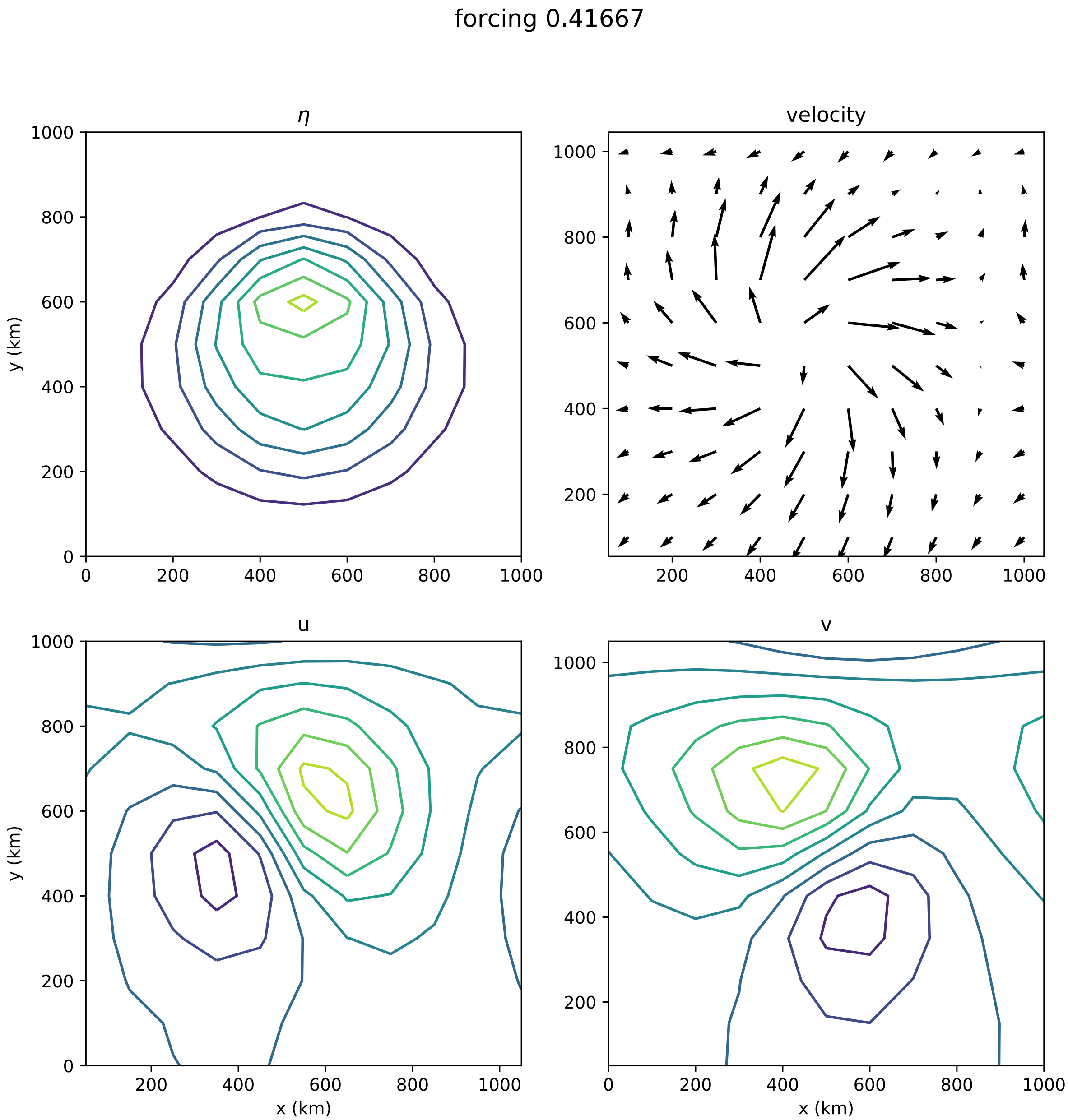


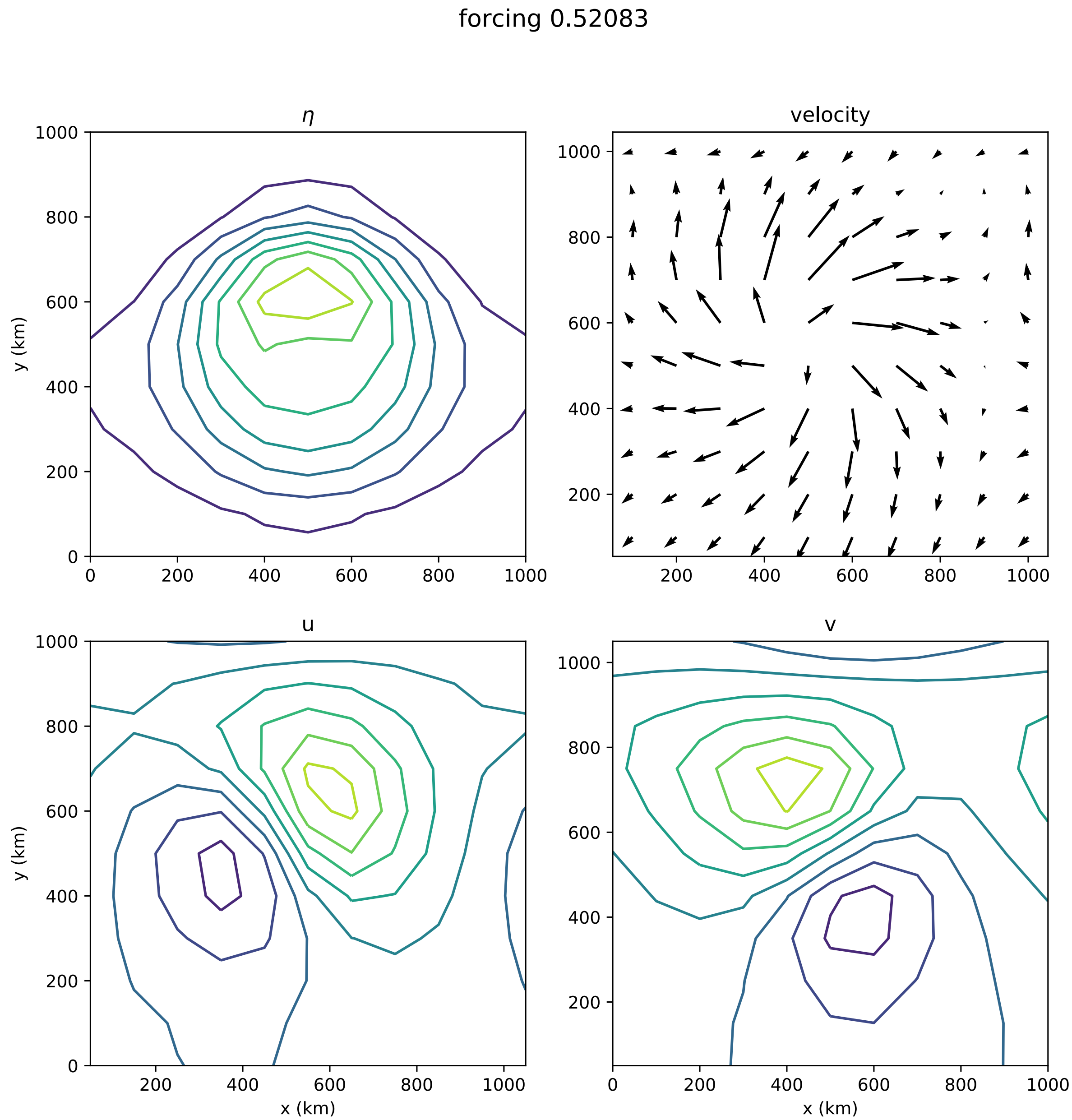

forcing 0.0
η
velocity
u
v
y (km)
x (km)
0
200
400
600
800
1000

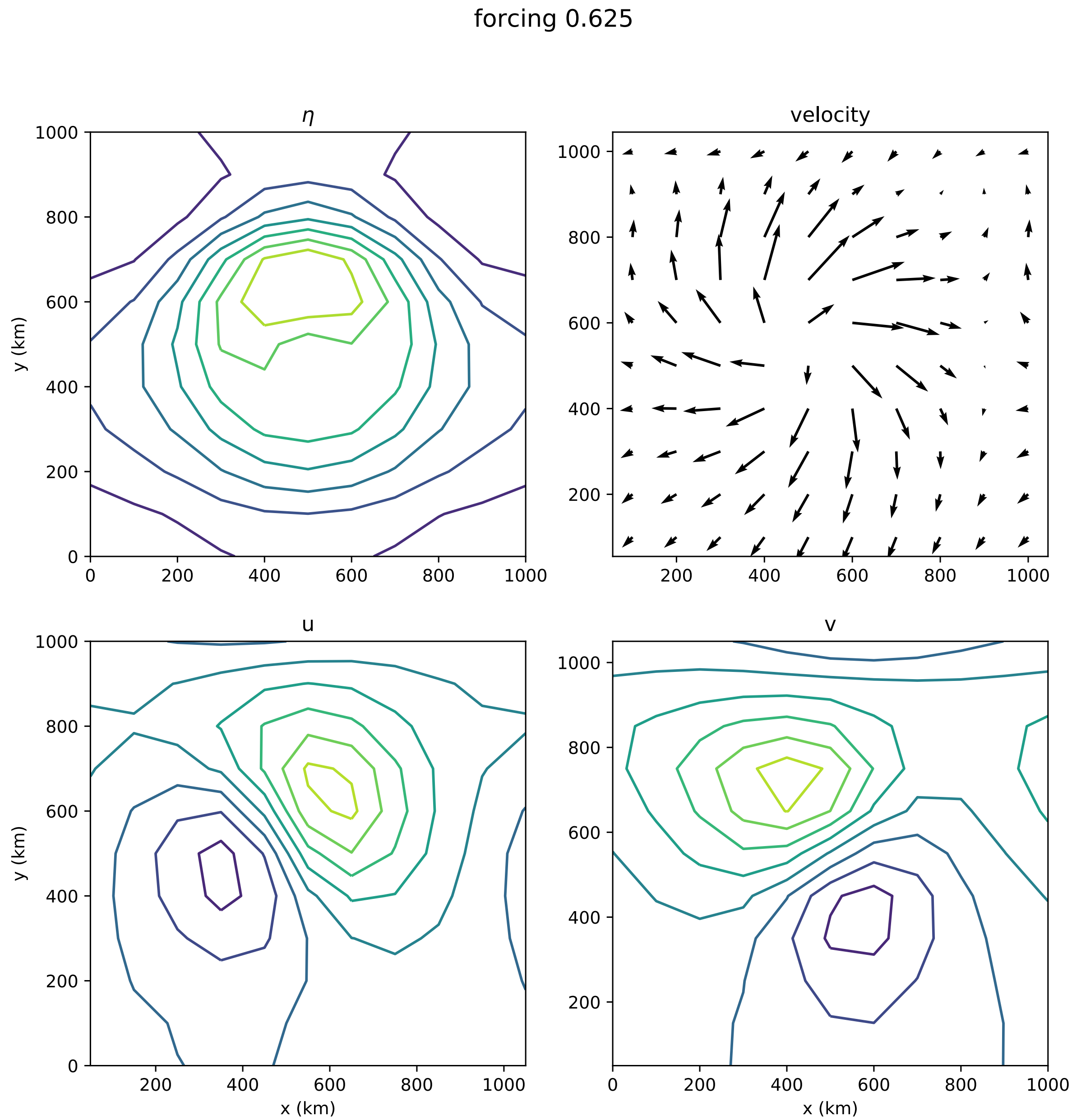

forcing 0.10417
η
velocity
u
v
y (km)
x (km)
0
200
400
600
800
1000

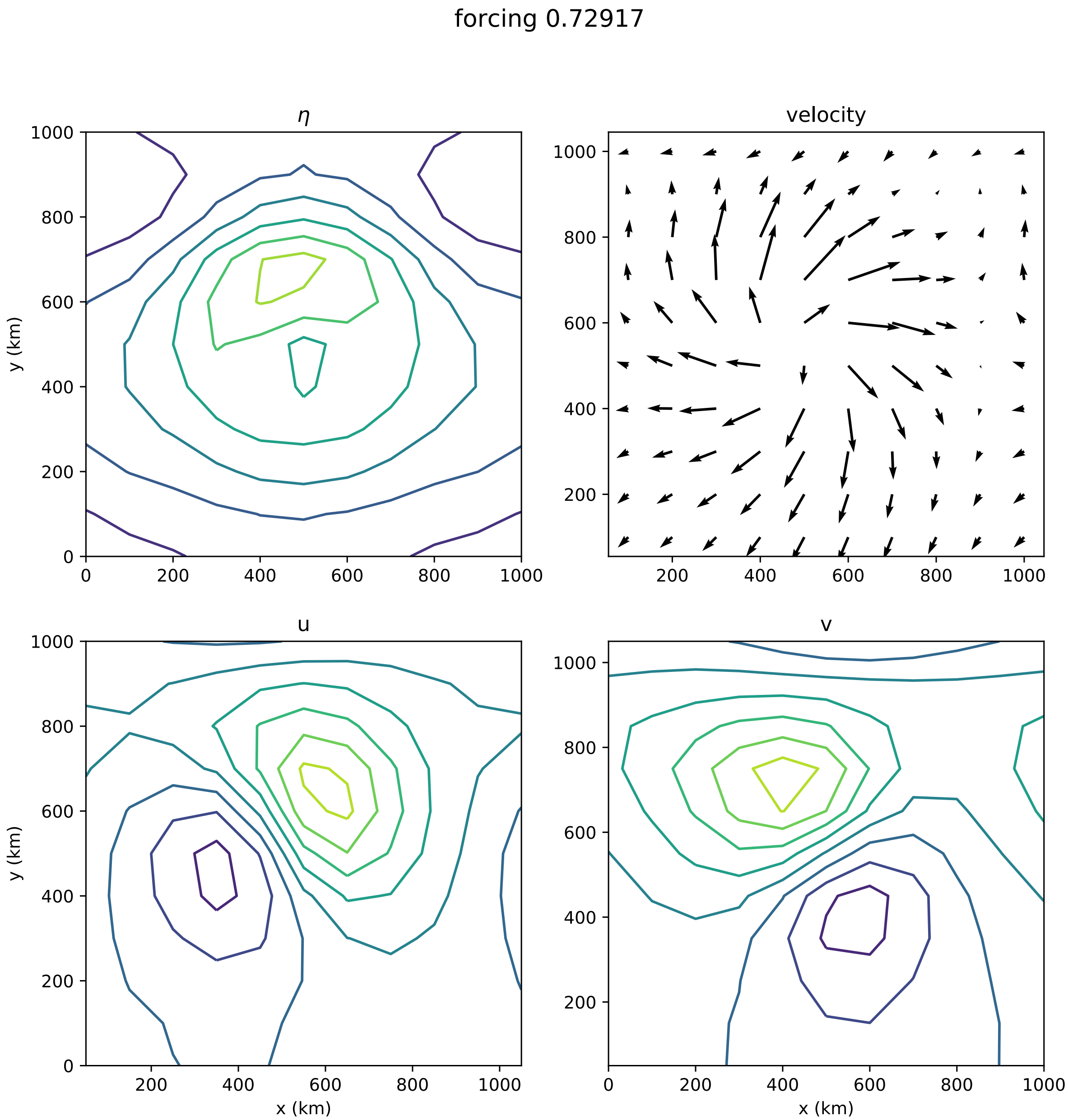

forcing 0.20833
η
velocity
u
v
y (km)
x (km)
0
200
400
600
800
1000

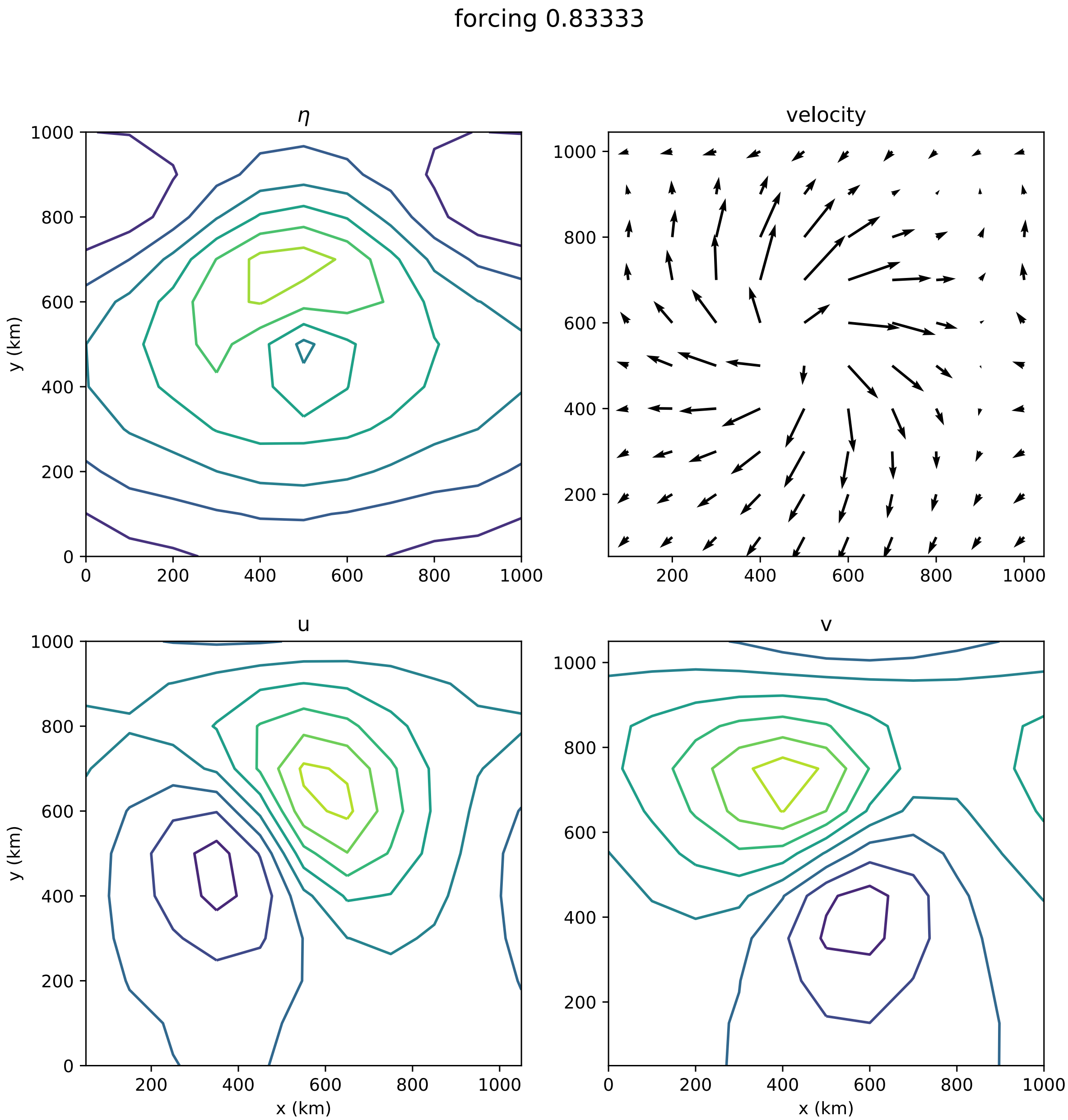

forcing 0.3125
η
velocity
u
v
y (km)
x (km)
0
200
400
600
800
1000

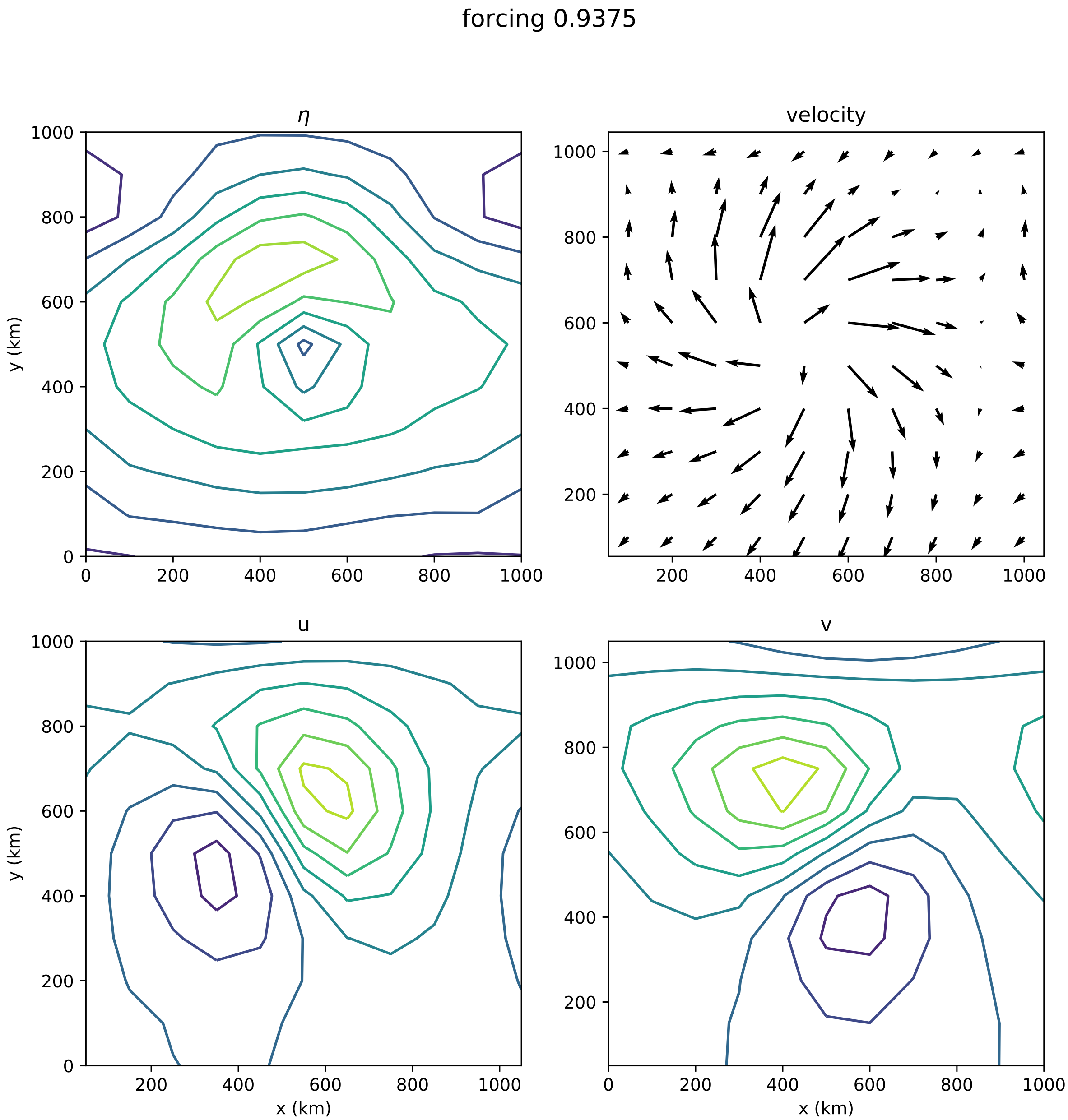

forcing 0.41667
η
velocity
u
v
y (km)
x (km)
0
200
400
600
800
1000

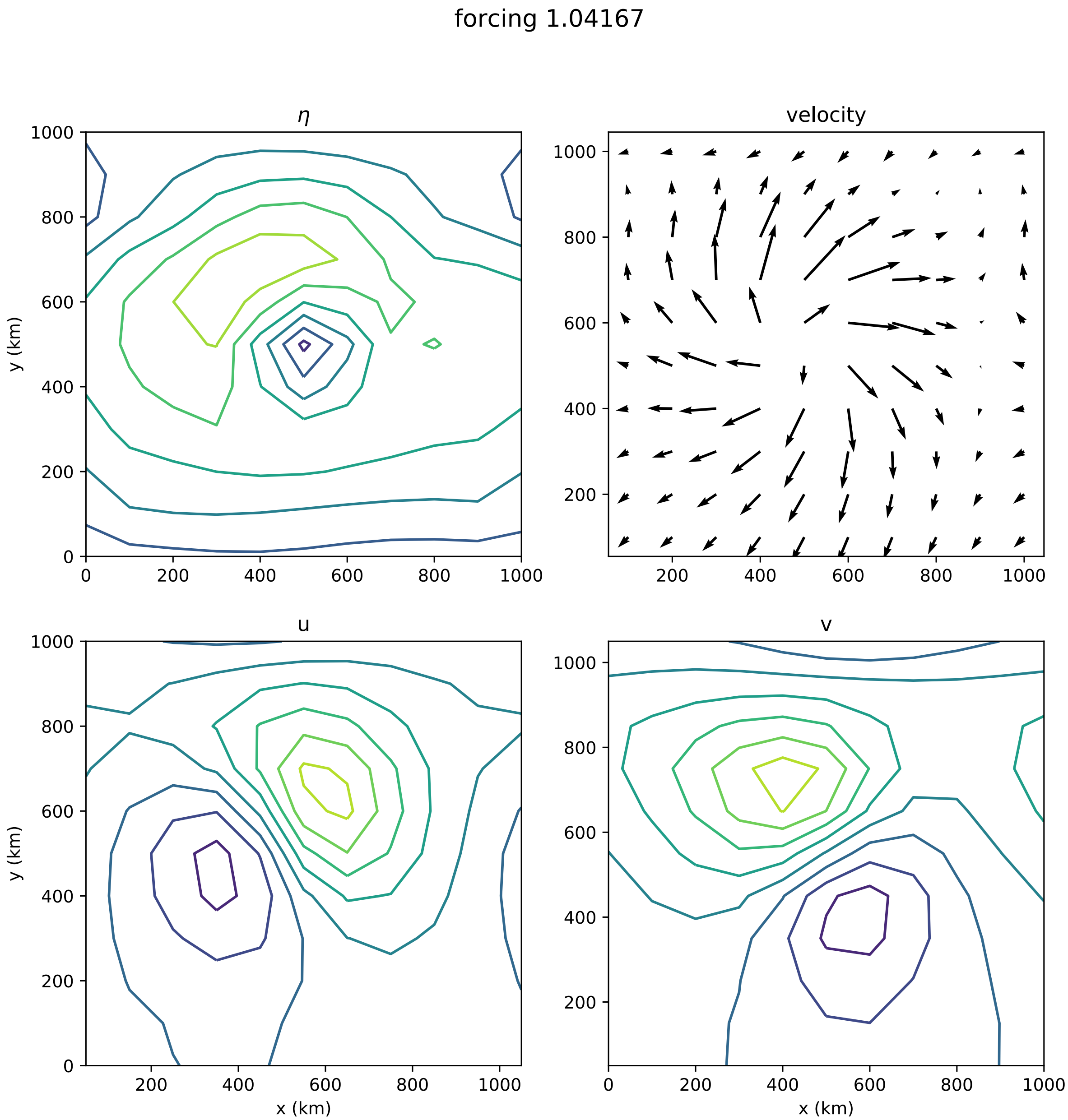

forcing 0.52083
η
velocity
u
v
y (km)
x (km)
0
200
400
600
800
1000

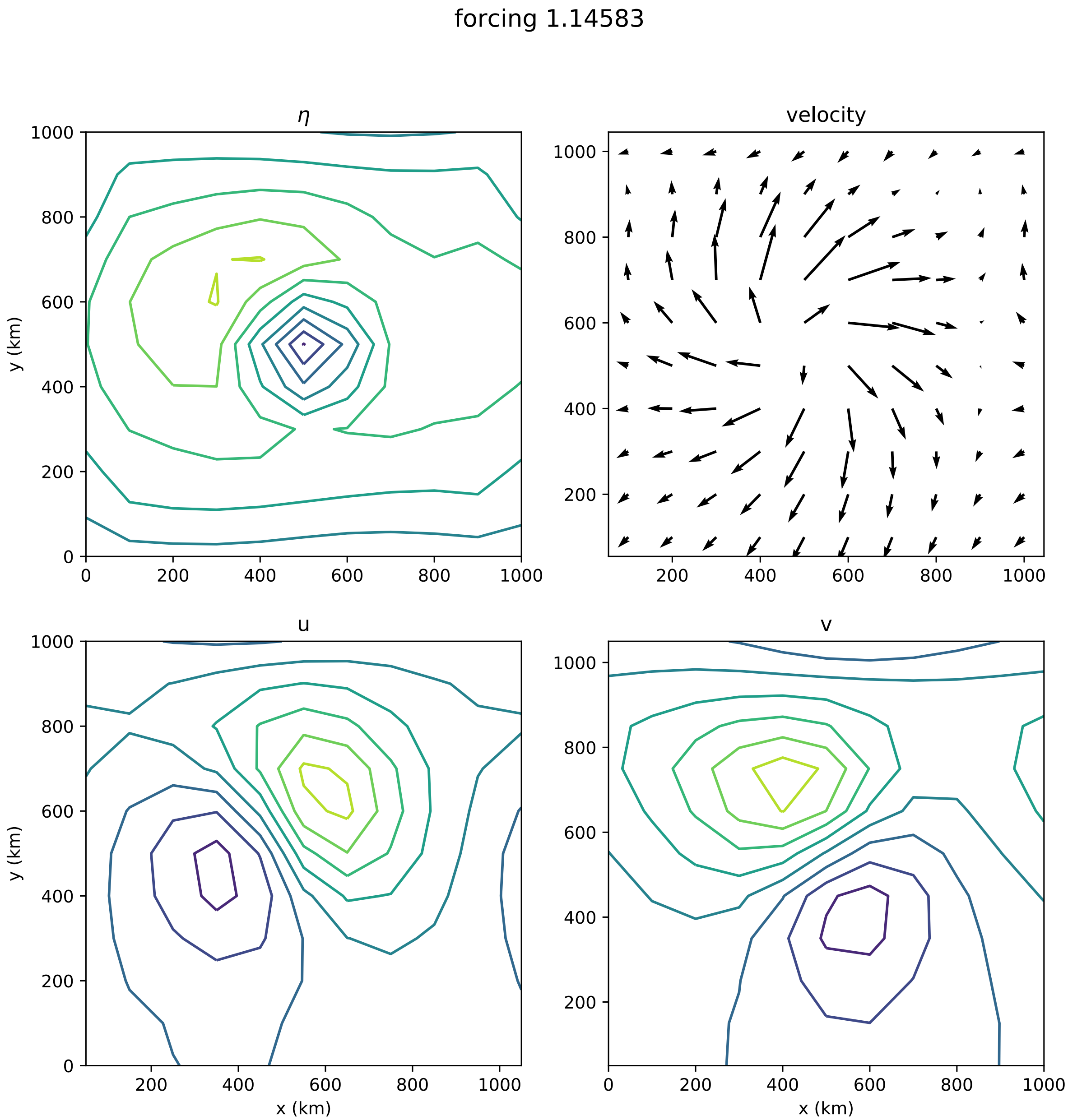

forcing 0.625
η
velocity
u
v
y (km)
x (km)
1000
800
600
400
200
0
200
400
600
800
1000


forcing 0.72917
η
velocity
u
v
y (km)
x (km)
0
200
400
600
800
1000

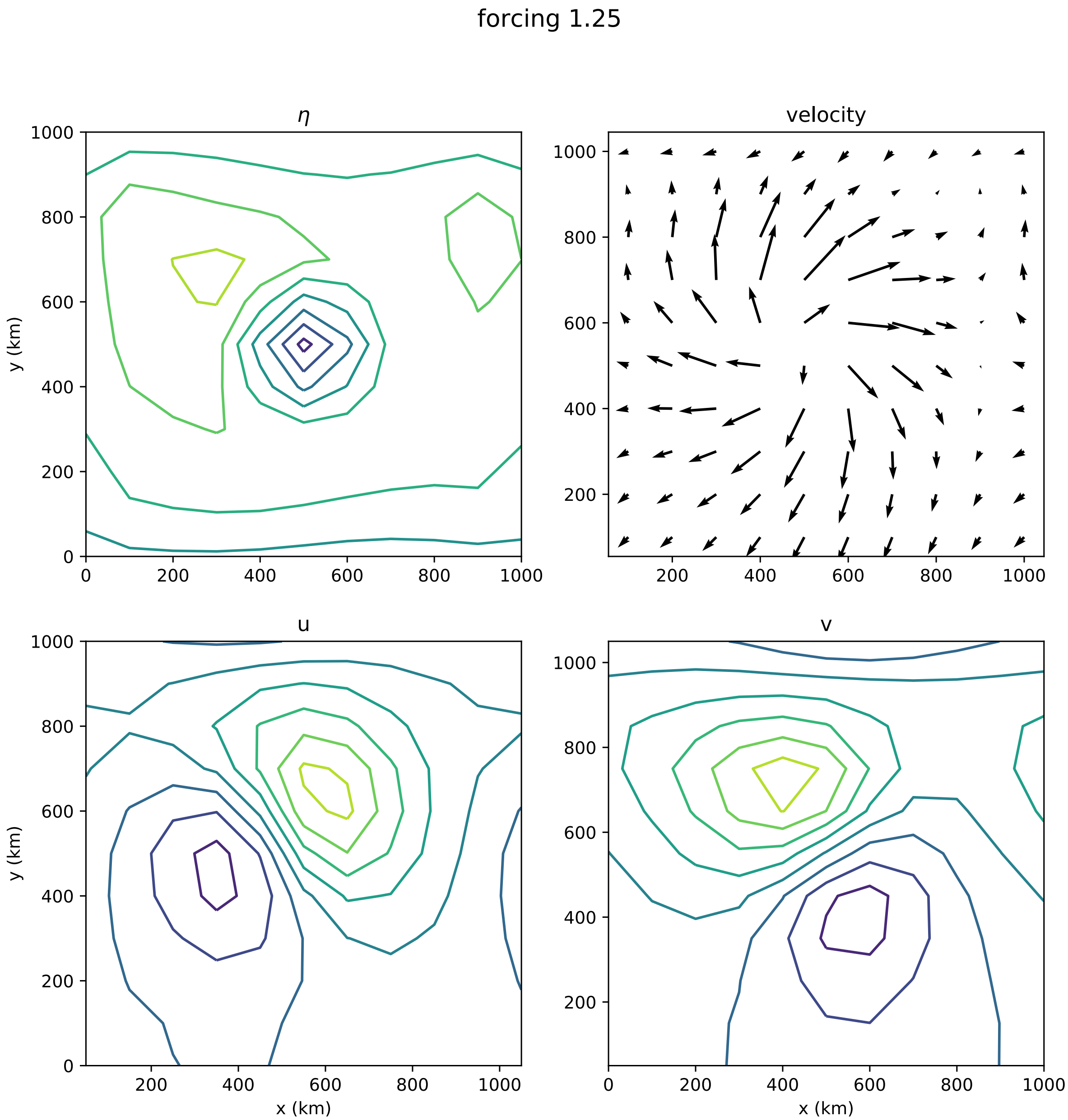

forcing 0.83333
η
velocity
u
v
y (km)
x (km)
1000
800
600
400
200
0


forcing 0.9375
η
velocity
u
v
y (km)
x (km)
0
200
400
600
800
1000

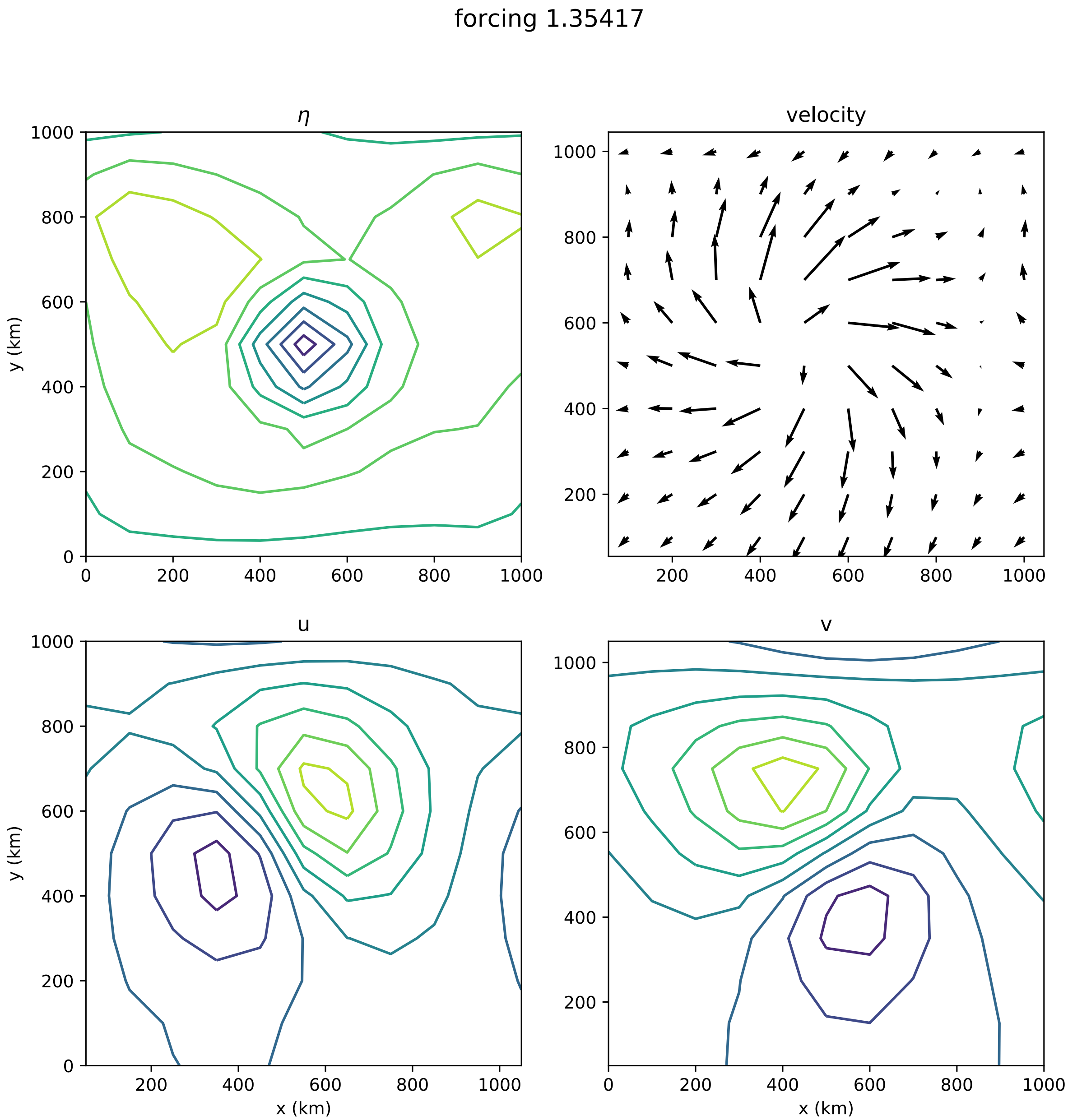

forcing 1.04167
η
velocity
u
v
y (km)
x (km)
0
200
400
600
800
1000

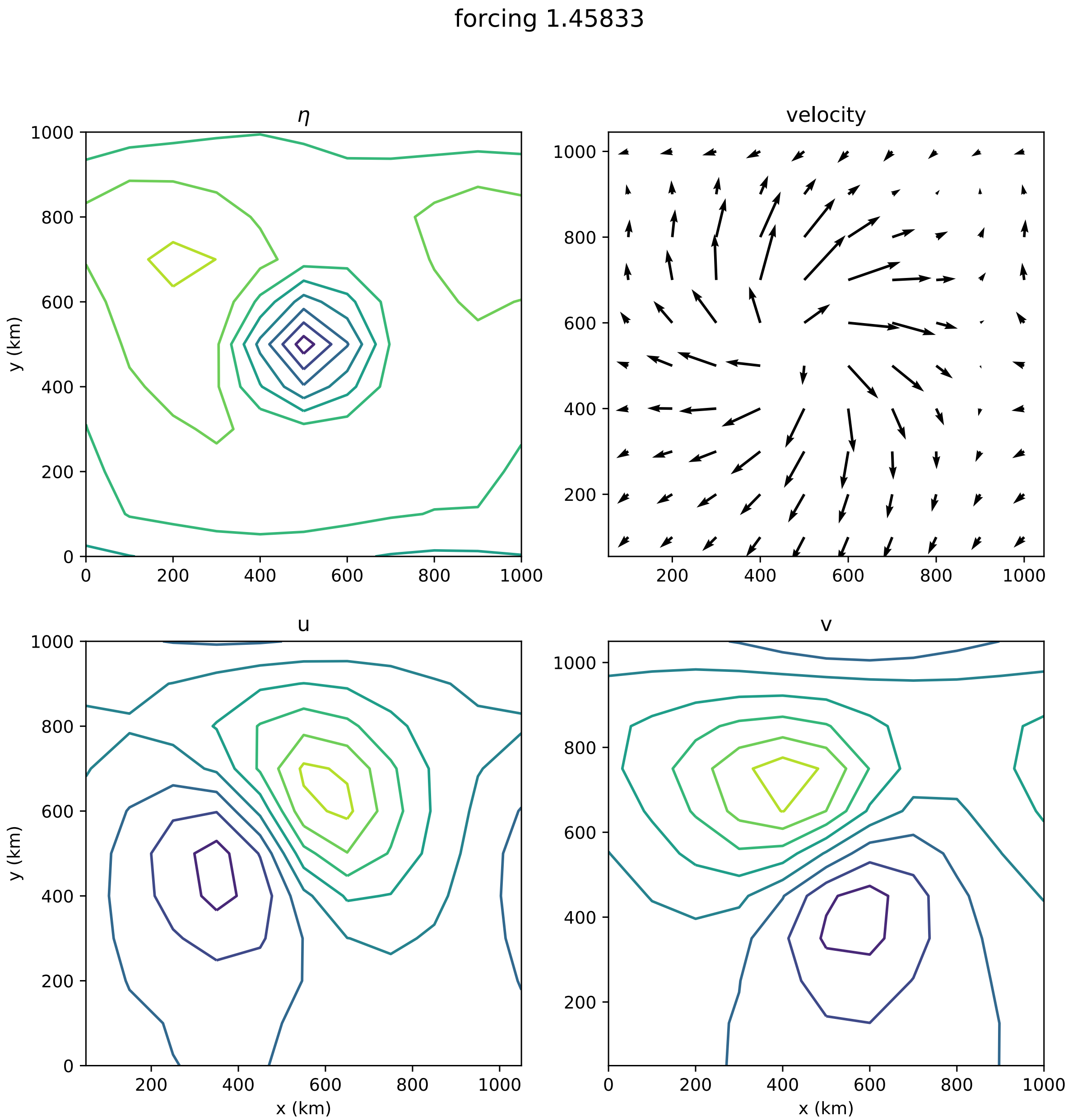


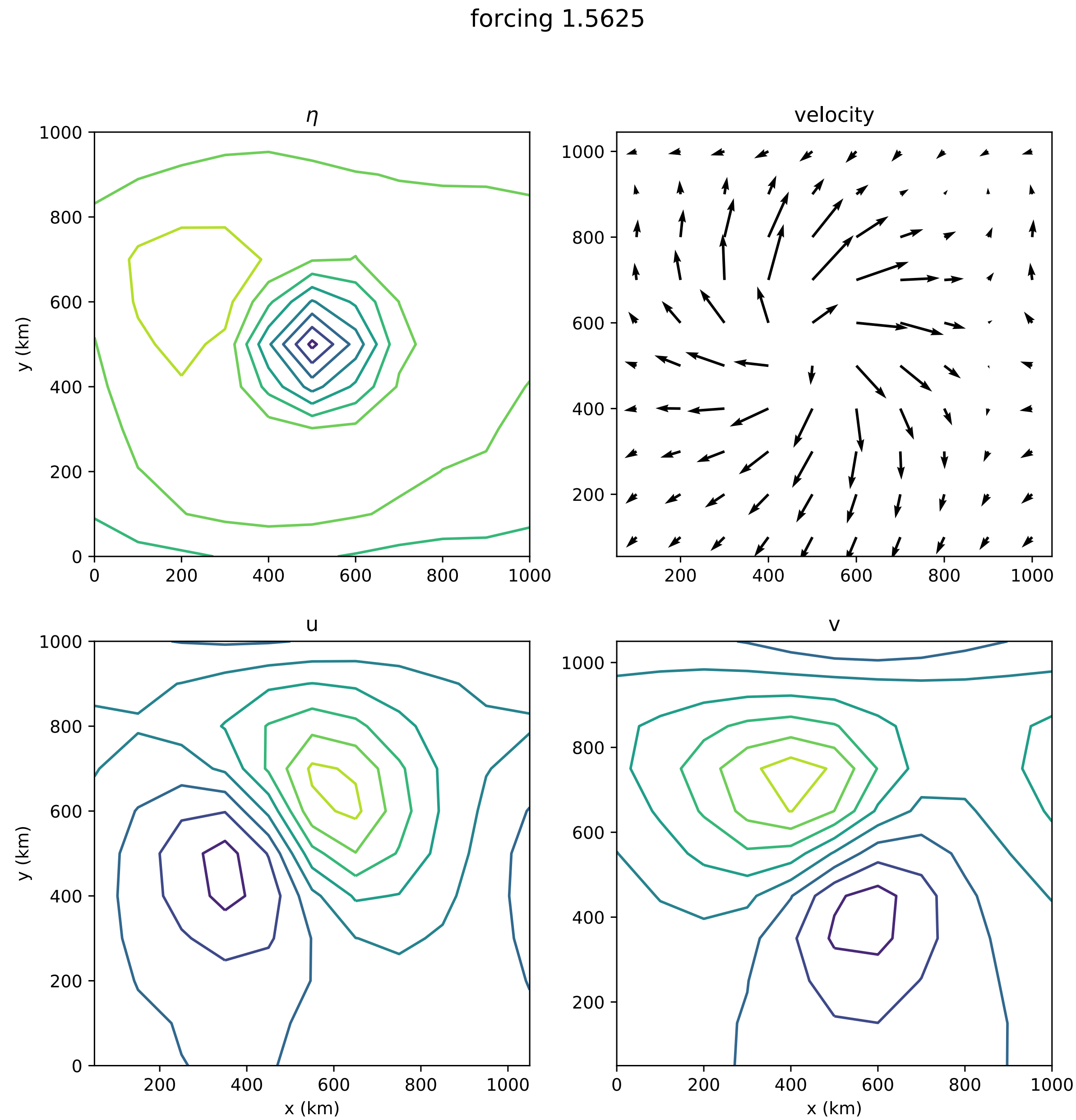
forcing 1.14583
η
velocity
u
v
y (km)
x (km)

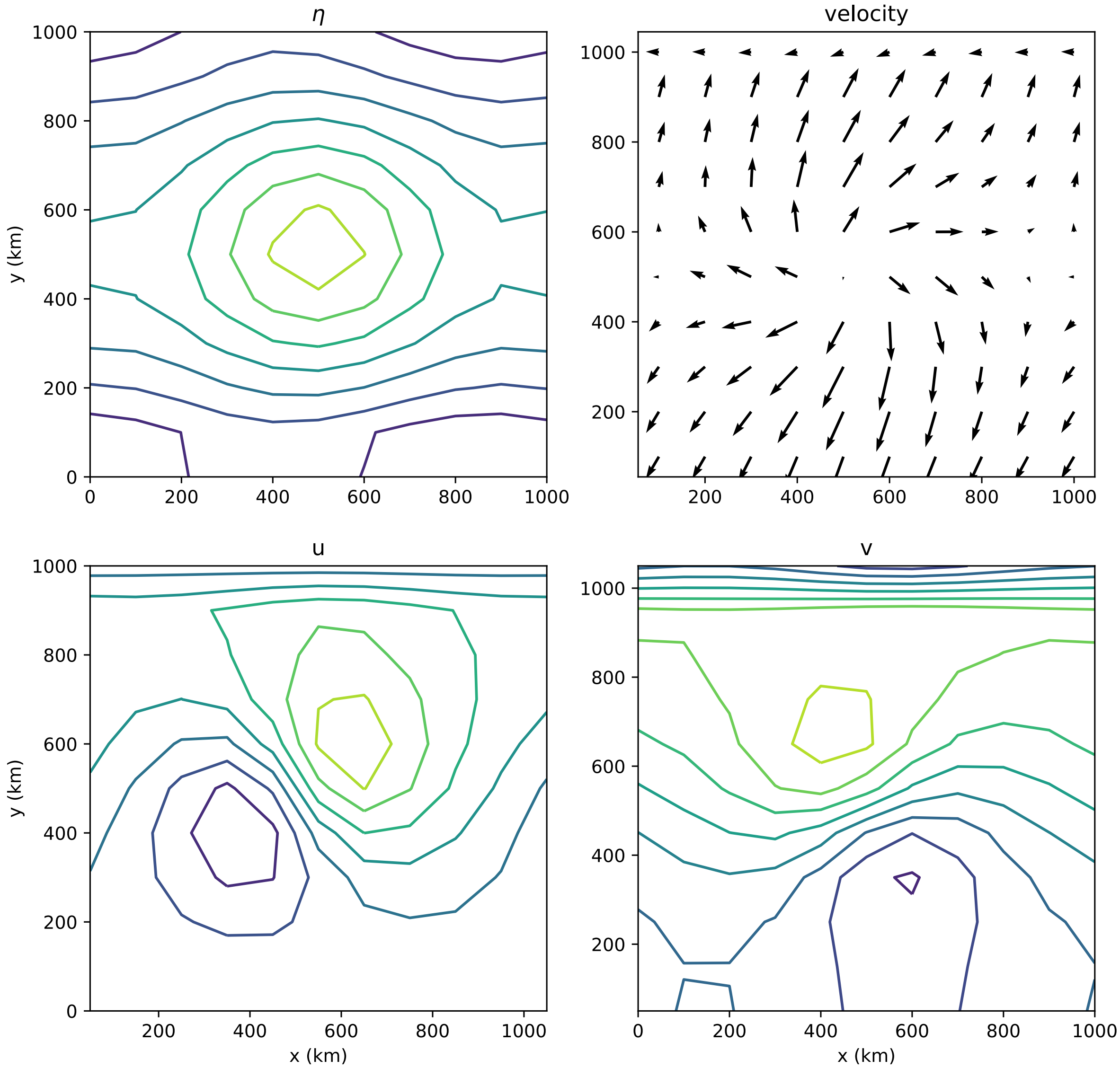


forcing 1.25
η
velocity
u
v
y (km)
x (km)
0
200
400
600
800
1000


forcing 1.35417
η
velocity
u
v
y (km)
x (km)
0
200
400
600
800
1000


forcing 1.45833
η
velocity
u
v
y (km)
x (km)
0
200
400
600
800
1000


forcing 1.5625
η
velocity
u
v
y (km)
x (km)
0
200
400
600
800
1000


η
velocity
u
v
y (km)
y (km)
x (km)
x (km)
0
200
400
600
800
1000