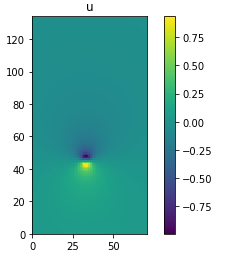
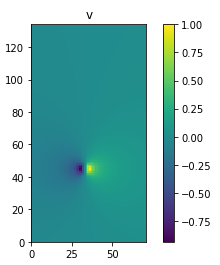
\*\*\* Please refer to the actual code file documentation to learn more about the parameters and details of how each function works.

1. **yolo\_vortex\_generator.py**

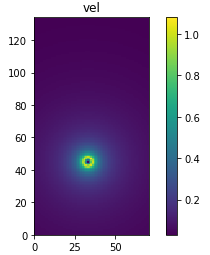
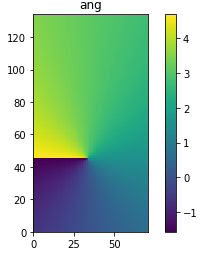
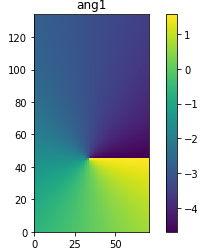
* 3 main functions:
  + **generate\_samples(num, l)** - *Randomly generate samples with image data, neural network input data and unprocessed labels.* \*\*\* Note: Contains 2 subfunctions:
    - **vortex(R1, R2, x, y, bx, by, direction)** – *Generate a 2-D velocity vector field of a circular vector field*
    - **generate\_random\_vortex(l)** - *Randomly resize the 2-D velocity vector field of a circular vortex in x and y direction and then randomly rotate it to generate a 2-D velocity vector field of a non-circular vortex rotated at an arbitrary angle.*
  + **create\_yolo\_label(image, data, label, save\_file\_num, yolo\_grid)** - *Convert labels to YOLO compatible form*
  + **plot\_samples(image, data, label, l, slice\_interval)** - *Plot a Quiver plot with arrows indicating the magnitude and direction of velocity vector and plot u and v velocity vector field.*
* User should only need to adjust parameters in “**main(l, num, save\_file\_num, yolo\_grid)**” and run to generate the required artificial data

How does “yolo\_vortex\_generator.py” generate u and v velocity vector field for a non-circular, rotated vortex?

Step 1: Use “**vortex(R1, R2, x, y, bx, by, direction)**” to generate circular vortex on non-square mesh grid. For eg: u and v velocity vector field

Step 2: Use u and v velocity field to generate velocity scalar field and angle scalar field.

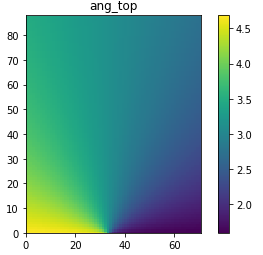
* ang and ang1 are essentially the same as they give the same u and v velocity field in the end.
* However, they have discontinuity 180 degree from each other. This property is useful later as the continuous side can be concatenated to give u and v velocity field that’s not affected by the discontinuities.

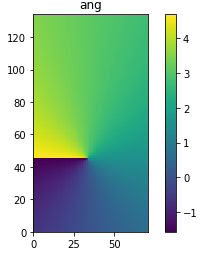
vel = np.hypot(u,v)

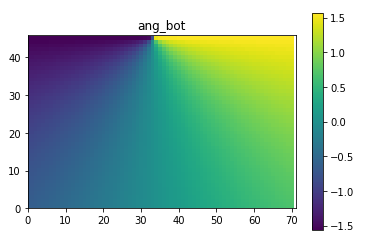
ang = np.arctan(v/u)

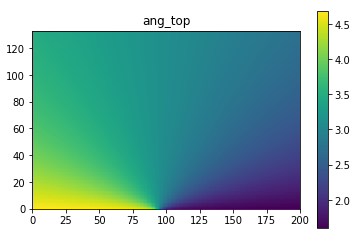
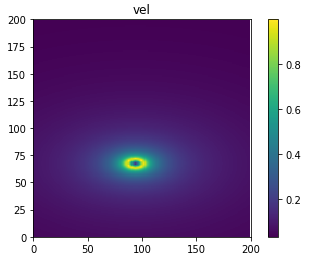
(Note: Angles are in radian)

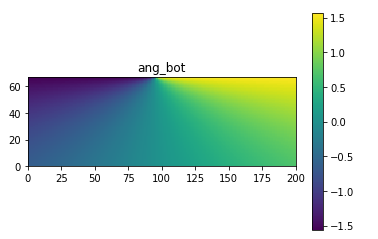
Step 3: Separate ang, ang1 to their top and bot parts before resizing to prevent interpolation error at angle discontinuities. For eg,



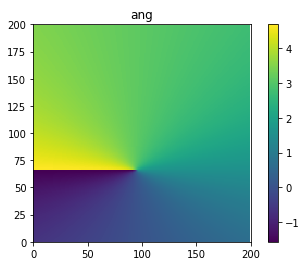
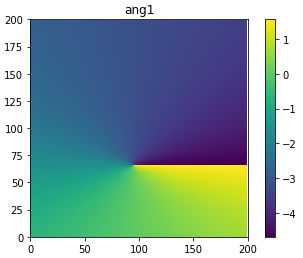




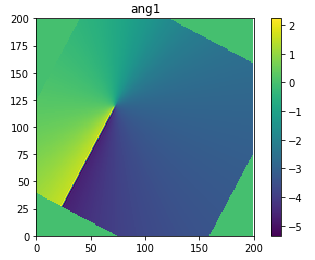
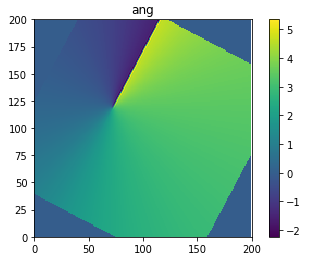
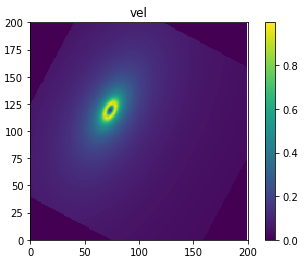
Step 4: Resize vel, ang\_top, ang\_bot, ang\_top1, ang\_bot1 so that the overall size of vel, ang, ang1 is 200x200. For eg,



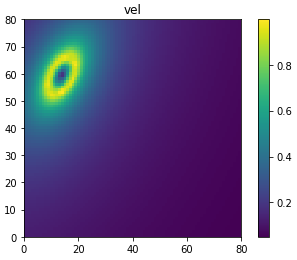
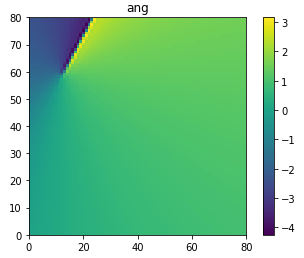
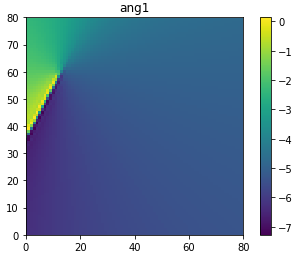
Step 5: Concatenate ang\_top and ang\_bot to form ang and do the same for ang1. For eg,

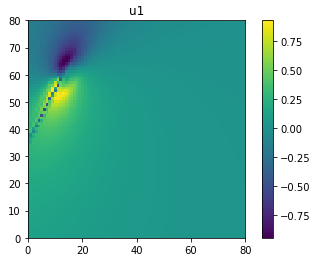


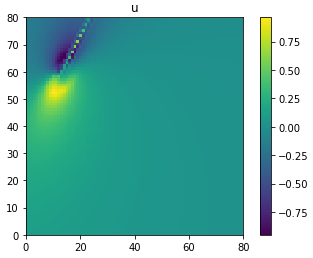
Step 6: Rotate vel, ang, ang1 by random angle using scipy.ndimage.interpolation.rotate. For eg,



Step 7: Crop a size of (80, 80) from the centre of vel, ang, ang1. For eg,



Step 8: Compute velocity vector fields (u, v, u1, v1) from vel, ang, ang1. For eg,

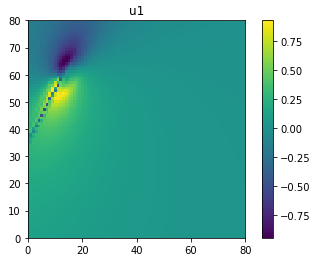
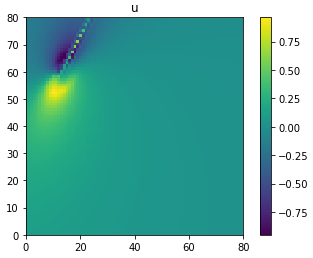
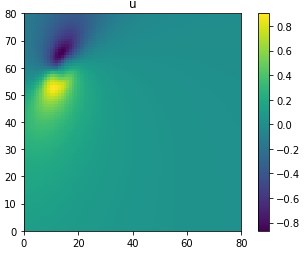
 u = vel\*np.cos(ang)

v = vel\*np.sin(ang)

u1 = vel\*np.cos(ang1)

v1 = vel\*np.sin(ang1)

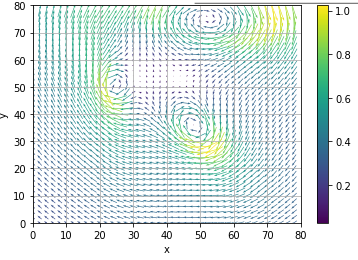
Step 9: Crop the continuous part of u, v, u1, v1 and concatenate to form u, v that are not affected by the discontinuity. For eg,



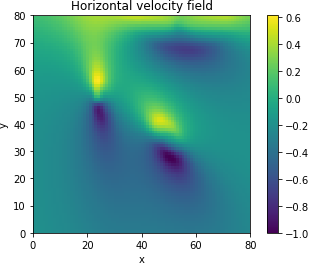
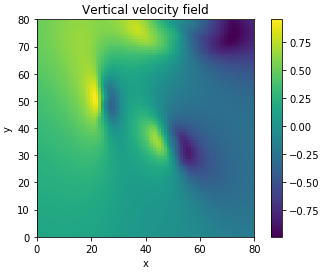
1. **yolo\_deep.py**

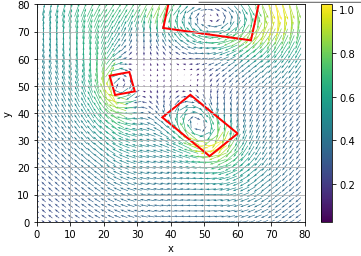
* 2 main functions:
  + **cnn\_model(x, l, label\_depth, keep\_prob)** - *Build the convolution neural network structure.*\*\*\* Note: Contains 2 subfunctions:
    - conv\_layer(input, size\_in, size\_out, fs, name="conv") - *Apply convolution filter, batch normalisation and activation function onto the input*
    - logits\_layer(input, size\_in, size\_out, fs, name="conv") - *Apply convolution filter onto the input.*
  + **train\_predict\_nn(image\_input, label, mode, save\_status, save\_model\_num, restore\_model\_num, epoch, batch\_size, drop\_out\_keep\_prob)** - *Run training or vortex detection and prediction.*
* User should only need to adjust parameters in “**main(mode, save\_status, save\_model\_num, restore\_model\_num, save\_file\_num, epoch, batch\_size, drop\_out\_keep\_prob)**” and run to train the model or use it to generate prediction.

How does yolo\_deep.py works?

 2D velocity vector field

The 2D velocity vector field can be separated into their x and y components





Bounding Boxes can then be plotted on the 2D vector field

The conv net then outputs a list consists of the centre location, width and height of the vortices.

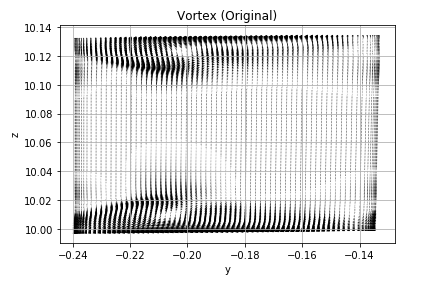
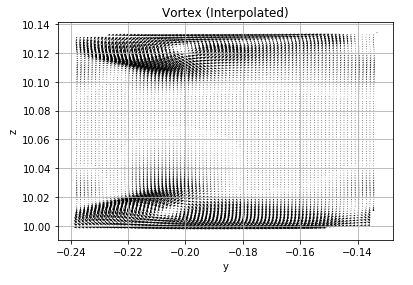
Two 80x80 velocity scalar field matrices are fed into a conv net

Convolutional Neural Network

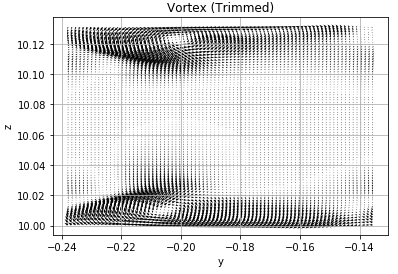
1. **CFD\_preprocess\_plot\_data.py**

* 5 main functions:
  + **load\_CFD\_data(angle, i)** - *Load data of i th slice of the blade CFD simulation data at specific blade angle.*
  + **interpolate(ly, lz, y, z, vsec\_t, vsec\_r)** - *Interpolate data onto an even, regular mesh grid*
  + **reshape\_remove\_nan(ly, lz, yi, zi, vsec\_t1, vsec\_r1, angle, i)** - *Generate data arrays of shape (lz, ly) after applying interpolation and removing Nan values.* \*\*\* Note: Contains 1 subfunction:
    - **remove\_nan(y, z, v, u)** - *Remove Nan values as a result of error from interpolating data that are near the edge or corner*
  + **save\_data(yi, zi, vsec\_t1, vsec\_r1, angle, i)** - *Save image and data file in .npy form to be loaded and fed into the neural network*
  + **plot\_CFD\_data(yi, zi, vsec\_t1, vsec\_r1, y, z, vsec\_t, vsec\_r)** - *Plot all before and after arrays and quiver plots*
* User should only need to adjust parameters in “**main(angle = 40, i = 45, ly = 80, lz = 80)**” and run to pre-process CFD data.

How does CFD\_preprocess\_plot\_data.py works?



Apply interpolate function

**interpolate(ly, lz, y, z, vsec\_t, vsec\_r)**

Apply remove\_nan function

Nan values at the edges need to be trimmed off using remove\_nan function