Hasegawa-Wakatani Equation

reference: Wakatani1984PoF, Hasegawa1987PRL, Numata2007PoP

Basic Equations

$$\partial_t \zeta + \{\phi, \zeta\} = \alpha\left(\phi - n\right) - \mu \nabla^4 \zeta$$
$$\partial_t n + \{\phi, n\} = \alpha\left(\phi - n\right) - \kappa \partial_y \phi - \mu \nabla^4 n$$
$$\zeta \equiv \nabla^2 \phi$$
$$\{a, b\} \equiv \partial_x a \partial_y b - \partial_y a \partial_x b$$

Python Packages

In [1]:
```
%config InlineBackend.figure_format = 'retina'
import numpy as np
import matplotlib.pyplot as plt
import scipy.fftpack as sf
from IPython import display
import math as mt
import matplotlib.animation as animation
plt.rcParams['font.size'] = 14
plt.rcParams['axes.linewidth'] = 1.5
plt.rcParams['animation.embed_limit']=60
plt.rcParams['animation.html'] = 'jshtml'
```

Source Code

In [2]:
```
def HW(nx,ny,lx,ly,nt,dt,kap,alph,mu,phi,n,isav):
    global KX,KY,KX2,KY2,KXD,KYD

    dx=lx/nx; dy=ly/ny

    ### define grids ###
    kx =2*np.pi/lx*np.r_[np.arange(nx/2),np.arange(-nx/2,0)]
    ky =2*np.pi/ly*np.r_[np.arange(ny/2),np.arange(-ny/2,0)]
    kxd=np.r_[np.ones(nx//3),np.zeros(nx//3+nx%3),np.ones(nx//3)]   #for de-alias
ing
    kyd=np.r_[np.ones(ny//3),np.zeros(ny//3+ny%3),np.ones(ny//3)]   #for de-alias
ing
    kx2=kx**2; ky2=ky**2
    KX ,KY =np.meshgrid(kx ,ky )
    KX2,KY2=np.meshgrid(kx2,ky2)
    KXD,KYD=np.meshgrid(kxd,kyd)

    phif=sf.fft2(phi)
    nf  =sf.fft2(n)
    zetaf=-(KX2+KY2)*phif

    phihst =np.zeros((nt//isav,nx,ny))
    nhst   =np.zeros((nt//isav,nx,ny))
    zetahst=np.zeros((nt//isav,nx,ny))
    phihst[0,:,:] =np.real(sf.ifft2(phif))
    nhst[0,:,:]   =np.real(sf.ifft2(nf))
    zetahst[0,:,:]=np.real(sf.ifft2(zetaf))

    for it in range(1,nt):

        #---double steps with integrating factor method(4th-order Runge-Kutta)---
#
        zetaf=np.exp(-mu*(KX2+KY2)**2*dt)*zetaf
        nf   =np.exp(-mu*(KX2+KY2)**2*dt)*nf

        gw1,ga1=adv(zetaf           ,nf            )
        gw2,ga2=adv(zetaf+0.5*dt*gw1,nf+0.5*dt*ga1)
        gw3,ga3=adv(zetaf+0.5*dt*gw2,nf+0.5*dt*ga2)
        gw4,ga4=adv(zetaf+    dt*gw3,nf+    dt*ga3)

        zetaf=zetaf+dt*(gw1+2*gw2+2*gw3+gw4)/6
        nf   =nf   +dt*(ga1+2*ga2+2*ga3+ga4)/6

        if(it%isav==0):
            phif=zetaf/(-(KX2+KY2)); phif[0,0]=0
            phi=np.real(sf.ifft2(phif))
            n   =np.real(sf.ifft2(nf))
            zeta=np.real(sf.ifft2(zetaf))
            phihst[it//isav,:,:]=phi
            nhst[it//isav,:,:]=n
            zetahst[it//isav,:,:]=zeta

    return locals()

def adv(zetaf,nf):
    phif=zetaf/(-(KX2+KY2)); phif[0,0]=0

    phi=np.real(sf.ifft2(phif))
    n  =np.real(sf.ifft2(nf))

    phixf = 1j*KX*phif;  phix =np.real(sf.ifft2(phixf *KXD*KYD))
    phiyf = 1j*KY*phif;  phiy =np.real(sf.ifft2(phiyf *KXD*KYD))
```

```
    zetaxf= 1j*KX*zetaf; zetax=np.real(sf.ifft2(zetaxf*KXD*KYD))
    zetayf= 1j*KY*zetaf; zetay=np.real(sf.ifft2(zetayf*KXD*KYD))
    nxf   = 1j*KX*nf;    nnx   =np.real(sf.ifft2(nxf   *KXD*KYD))
    nyf   = 1j*KY*nf;    nny   =np.real(sf.ifft2(nyf   *KXD*KYD))

    advf =-(phix*zetay-phiy*zetax)+alph*(phi-n)
    advg =-(phix*nny  -phiy*nnx)  +alph*(phi-n)-kap*np.real(sf.ifft2(phiyf))
    advff=sf.fft2(advf)
    advgf=sf.fft2(advg)

    return advff,advgf
```

Initial Condition

In [3]:
```
nx=256; ny=256; nt=5000; isav=25
kap=1.0
alph=0.1
mu=1e-4
dt=2e-2
lx=2*np.pi/0.15; ly=2*np.pi/0.15
dx=lx/nx; dy=ly/ny
x  =np.arange(nx)*dx
y  =np.arange(ny)*dy
X,Y=np.meshgrid(x,y)

s=2; s2=s**2
r1=(X-lx/2)**2+(Y-ly/2)**2
n  =np.exp(-r1/s2)
phi=n
```

Run

In [4]:
```
data=HW(nx,ny,lx,ly,nt,dt,kap,alph,mu,phi,n,isav)
locals().update(data)
```

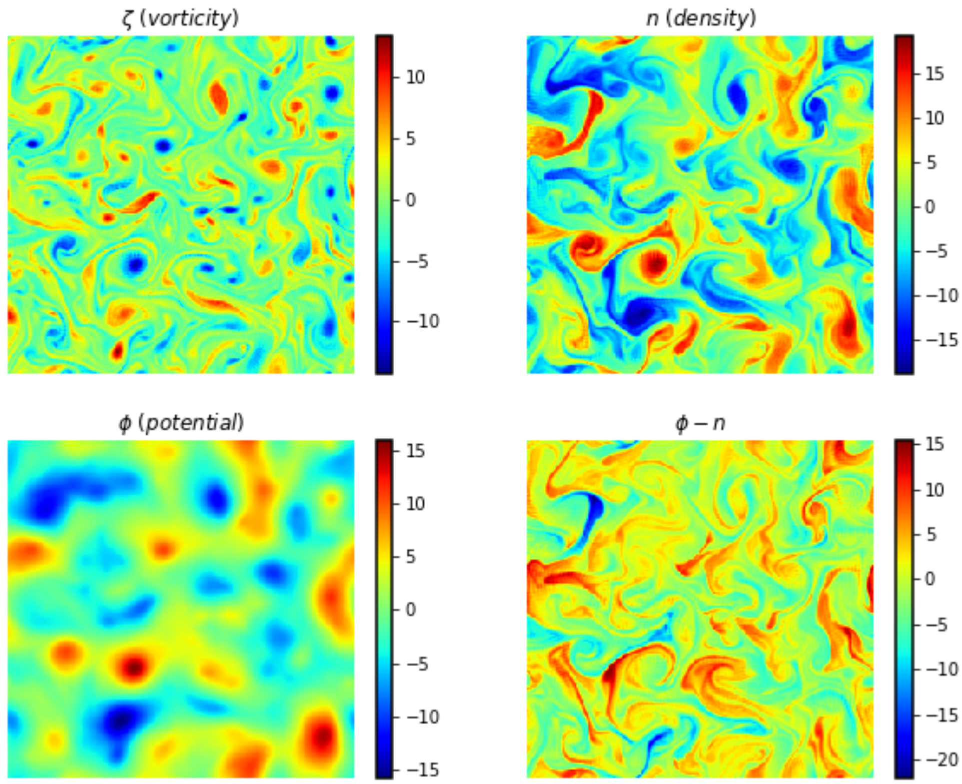
```
/home/mnakanot/anaconda3/lib/python3.7/site-packages/scipy/fftpack/basic.py:160:
FutureWarning: Using a non-tuple sequence for multidimensional indexing is depre
cated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be i
nterpreted as an array index, `arr[np.array(seq)]`, which will result either in
an error or a different result.
  z[index] = x
/home/mnakanot/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:53: R
untimeWarning: invalid value encountered in true_divide
/home/mnakanot/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:53: R
untimeWarning: divide by zero encountered in true_divide
/home/mnakanot/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:42: R
untimeWarning: divide by zero encountered in true_divide
/home/mnakanot/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:42: R
untimeWarning: invalid value encountered in true_divide
```

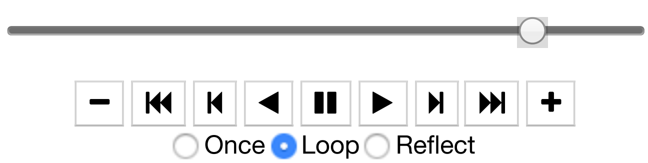
Animation

In [5]:
```
def update_anim(it):

    fig.clf()

    ax1 = fig.add_subplot(221)
    ax2 = fig.add_subplot(222)
    ax3 = fig.add_subplot(223)
    ax4 = fig.add_subplot(224)

    for ax in (ax1, ax2, ax3, ax4):
        ax.clear()

    im1=ax1.imshow(zetahst[it,:,:]            ,aspect='auto',origin='lower',cmap=
'jet');ax1.axis('off');fig.colorbar(im1, ax=ax1);ax1.set_title(r'$\zeta\ (vortici
ty)$')
    im2=ax2.imshow(nhst[it,:,:]               ,aspect='auto',origin='lower',cmap=
'jet');ax2.axis('off');fig.colorbar(im2, ax=ax2);ax2.set_title(r'$n\ (density)$')
    im3=ax3.imshow(phihst[it,:,:]             ,aspect='auto',origin='lower',cmap=
'jet');ax3.axis('off');fig.colorbar(im3, ax=ax3);ax3.set_title(r'$\phi\ (potentia
l)$')
    im4=ax4.imshow(phihst[it,:,:]-nhst[it,:,:],aspect='auto',origin='lower',cmap=
'jet');ax4.axis('off');fig.colorbar(im4, ax=ax4);ax4.set_title(r'$\phi-n$')
```

In [6]:
```
fig=plt.figure(figsize=(10,8))
anim=animation.FuncAnimation(fig,update_anim,frames=nt//isav)
plt.close()
anim
```

Out[6]:





Modified Hasegawa-Wakatani Equation

$$\partial_t \zeta + \{\phi, \zeta\} = \alpha \left(\tilde{\phi} - \tilde{n}\right) - \mu \nabla^4 \zeta$$

$$\partial_t n + \{\phi, n\} = \alpha \left(\tilde{\phi} - \tilde{n}\right) - \kappa \partial_y \phi - \mu \nabla^4 n$$

$$\zeta \equiv \nabla^2 \phi$$

$$\{a, b\} \equiv \partial_x a \partial_y b - \partial_y a \partial_x b$$

$$\tilde{f} \equiv f - \langle f \rangle$$

$$\langle f \rangle \equiv \frac{1}{L_y} \int f dy$$

Source Code

In [7]:
```
def MHW(nx,ny,lx,ly,nt,dt,kap,alph,mu,phi,n,isav):
    global KX,KY,KX2,KY2,KXD,KYD

    dx=lx/nx; dy=ly/ny

    ### define grids ###
    kx =2*np.pi/lx*np.r_[np.arange(nx/2),np.arange(-nx/2,0)]
    ky =2*np.pi/ly*np.r_[np.arange(ny/2),np.arange(-ny/2,0)]
    kxd=np.r_[np.ones(nx//3),np.zeros(nx//3+nx%3),np.ones(nx//3)]   #for de-alias
ing
    kyd=np.r_[np.ones(ny//3),np.zeros(ny//3+ny%3),np.ones(ny//3)]   #for de-alias
ing
    kx2=kx**2; ky2=ky**2
    KX ,KY =np.meshgrid(kx ,ky )
    KX2,KY2=np.meshgrid(kx2,ky2)
    KXD,KYD=np.meshgrid(kxd,kyd)

    phif=sf.fft2(phi)
    nf  =sf.fft2(n)
    #zetaf=sf.fft2(np.random.randn(nx,ny)*2)
    zetaf=-(KX2+KY2)*phif

    phihst =np.zeros((nt//isav,nx,ny))
    nhst   =np.zeros((nt//isav,nx,ny))
    zetahst=np.zeros((nt//isav,nx,ny))
    phihst[0,:,:] =np.real(sf.ifft2(phif))
    nhst[0,:,:]   =np.real(sf.ifft2(nf))
    zetahst[0,:,:]=np.real(sf.ifft2(zetaf))

    for it in range(1,nt):

        #---double steps with integrating factor method(4th-order Runge-Kutta)---
#
        zetaf=np.exp(-mu*(KX2+KY2)**2*dt)*zetaf
        nf   =np.exp(-mu*(KX2+KY2)**2*dt)*nf

        gw1,ga1=adv(zetaf           ,nf            )
        gw2,ga2=adv(zetaf+0.5*dt*gw1,nf+0.5*dt*ga1)
        gw3,ga3=adv(zetaf+0.5*dt*gw2,nf+0.5*dt*ga2)
        gw4,ga4=adv(zetaf+    dt*gw3,nf+    dt*ga3)

        zetaf=zetaf+dt*(gw1+2*gw2+2*gw3+gw4)/6
        nf   =nf   +dt*(ga1+2*ga2+2*ga3+ga4)/6

        if(it%isav==0):
            phif=zetaf/(-(KX2+KY2)); phif[0,0]=0
            phi=np.real(sf.ifft2(phif))
            n   =np.real(sf.ifft2(nf))
            zeta=np.real(sf.ifft2(zetaf))
            phihst[it//isav,:,:]=phi
            nhst[it//isav,:,:]=n
            zetahst[it//isav,:,:]=zeta

    return locals()

def adv(zetaf,nf):
    phif=zetaf/(-(KX2+KY2)); phif[0,0]=0

    phi=np.real(sf.ifft2(phif))
    n  =np.real(sf.ifft2(nf))

    phiz=np.sum(phi*dy,axis=0)/ly
```

```
    nz  =np.sum(n  *dy,axis=0)/ly

    phixf = 1j*KX*phif;  phix =np.real(sf.ifft2(phixf *KXD*KYD))
    phiyf = 1j*KY*phif;  phiy =np.real(sf.ifft2(phiyf *KXD*KYD))
    zetaxf= 1j*KX*zetaf; zetax=np.real(sf.ifft2(zetaxf*KXD*KYD))
    zetayf= 1j*KY*zetaf; zetay=np.real(sf.ifft2(zetayf*KXD*KYD))
    nxf   = 1j*KX*nf;    nnx   =np.real(sf.ifft2(nxf   *KXD*KYD))
    nyf   = 1j*KY*nf;    nny   =np.real(sf.ifft2(nyf   *KXD*KYD))

    advf =-(phix*zetay-phiy*zetax)+alph*((phi-phiz)-(n-nz))
    advg =-(phix*nny  -phiy*nnx)  +alph*((phi-phiz)-(n-nz))-kap*np.real(sf.ifft2(
phiyf))
    advff=sf.fft2(advf)
    advgf=sf.fft2(advg)

    return advff,advgf
```

Initial Condition

In [8]:
```
nx=256; ny=256; nt=5000; isav=25
kap=1.0
alph=1.0
mu=1e-4
dt=2e-2
lx=2*np.pi/0.15; ly=2*np.pi/0.15
dx=lx/nx; dy=ly/ny
x  =np.arange(nx)*dx
y  =np.arange(ny)*dy
X,Y=np.meshgrid(x,y)

s=2; s2=s**2
r1=(X-lx/2)**2+(Y-ly/2)**2
n  =np.exp(-r1/s2)
phi=n
```

Run

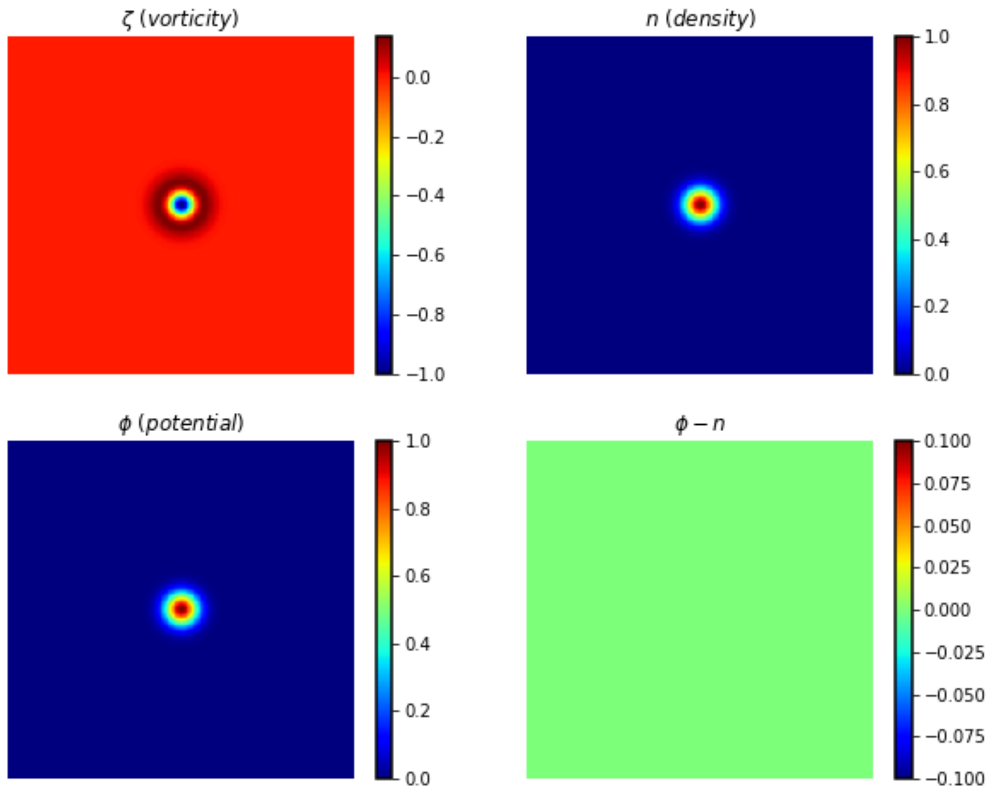
In [9]:
```
data=MHW(nx,ny,lx,ly,nt,dt,kap,alph,mu,phi,n,isav)
locals().update(data)
```

```
/home/mnakanot/anaconda3/lib/python3.7/site-packages/scipy/fftpack/basic.py:160:
FutureWarning: Using a non-tuple sequence for multidimensional indexing is depre
cated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be i
nterpreted as an array index, `arr[np.array(seq)]`, which will result either in
an error or a different result.
  z[index] = x
/home/mnakanot/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:54: R
untimeWarning: invalid value encountered in true_divide
/home/mnakanot/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:54: R
untimeWarning: divide by zero encountered in true_divide
/home/mnakanot/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:43: R
untimeWarning: divide by zero encountered in true_divide
/home/mnakanot/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:43: R
untimeWarning: invalid value encountered in true_divide
```

Animation

In [10]:
```
fig=plt.figure(figsize=(10,8))
anim=animation.FuncAnimation(fig,update_anim,frames=nt//isav)
plt.close()
anim
```

Out[10]:



Once Loop Reflect