March 3, 2020
Huan Q. Bui

I was going to send you an email with this document and the new images attached, but it (as usual) got too long. So I decided to put everything into this document instead. There are 05 items in this document.

1. A FT of ϕ, or $\hat{\phi}$, that works is the following:

$$\hat{\phi}(x,y) = \frac{1}{3}\left(3 - \frac{i}{2}\sin^2(x/2) - \sin^4(x/2) - \frac{i}{2}\sin^4(y/2) - \sin^8(y/2) - \frac{i}{16}(\sin x \cos y - \sin x)\right) \tag{1}$$

Taylor expand this around $(0,0)$ gives

$$\begin{aligned}&\left(1 - \frac{iy^4}{96} + \frac{iy^6}{576} + \mathcal{O}\left(y^7\right)\right) + x\left(\frac{iy^2}{96} - \frac{iy^4}{1152} + \frac{iy^6}{34560} + \mathcal{O}\left(y^7\right)\right) - \frac{ix^2}{24}\\ &+x^3\left(-\frac{iy^2}{576} + \frac{iy^4}{6912} - \frac{iy^6}{207360} + \mathcal{O}\left(y^7\right)\right) - \left(\frac{1}{48} - \frac{i}{288}\right)x^4 + \mathcal{O}\left(x^5\right).\end{aligned} \tag{2}$$

I have checked that $\hat{\phi}(0,0) = 1$. With this, Taylor-expand

$$\log\left(\frac{\hat{\phi}((x,y)+(0,0))}{\hat{\phi}(0,0)}\right) \tag{3}$$

gives

$$\left(-\frac{iy^4}{96} + \mathcal{O}\left(y^5\right)\right) + x\left(\frac{iy^2}{96} - \frac{iy^4}{1152} + \mathcal{O}\left(y^5\right)\right) + x^2\left(-\frac{i}{24} + \frac{y^4}{2048} + \mathcal{O}\left(y^5\right)\right) + \mathcal{O}\left(x^3\right). \tag{4}$$

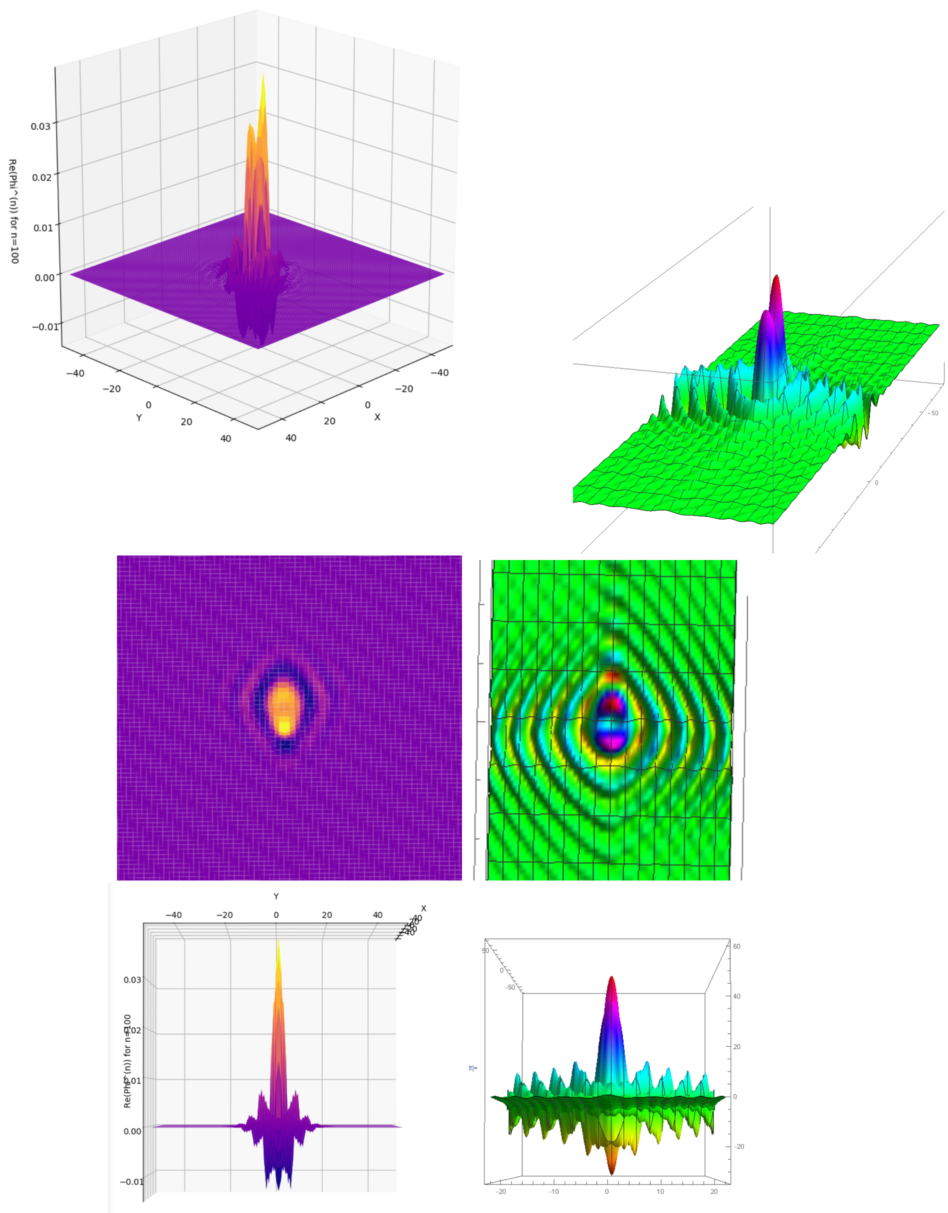
Once cos and sin have been replaced by $e^{i\cdots}$ this turns into

$$\begin{aligned}&\frac{1}{192}e^{iy-ix} - \frac{1}{192}e^{ix+iy} + \frac{1}{192}e^{-ix-iy} - \frac{1}{192}e^{ix-iy} + \left(\frac{3}{32} + \frac{i}{24}\right)e^{ix}\\ &+\left(\frac{7}{96} + \frac{i}{24}\right)e^{-ix} - \frac{1}{48}e^{-2ix} - \frac{1}{48}e^{2ix} + \left(\frac{7}{96} + \frac{i}{24}\right)e^{-iy} - \frac{1}{768}e^{-4iy} + \frac{1}{96}e^{-3iy}\\ &-\left(\frac{7}{192} + \frac{i}{96}\right)e^{-2iy} + \left(\frac{7}{96} + \frac{i}{24}\right)e^{iy} - \left(\frac{7}{192} + \frac{i}{96}\right)e^{2iy} + \frac{1}{96}e^{3iy} - \frac{1}{768}e^{4iy} + \left(\frac{301}{384} - \frac{7i}{48}\right)\end{aligned} \tag{5}$$

from which we can read off the values of ϕ on $\mathbb{Z}^2$.

2. I rescaled the x, y in the previous $H(x,y)$ with $t = 100$ and got good agreement! (t here appears in H_p^t and t^E in the paper). I wish I had thought about the possibility that the stretchingcould be so extreme that the peaks appear rotated earlier...

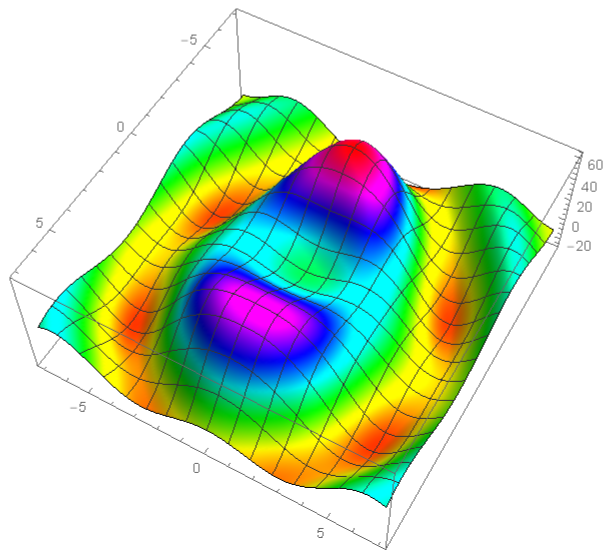
Here are the images. Purple-ly plots are convolution powers with $n = 100$. Green-ish plots are the approximated attractor $H_P^t(x,y)$ with $t = 100$. (I don't know what the exact correspondence between t and n is for now, but I think setting them equal is an o.k. starting point.)



3. I'm running the calculations for $H_P^t(x, y)$ again with $t = 100$. The output looks good so far. I'm integrating in "batches" and will aggregate the data afterwards. This should give us a lot of data for future stretching/contracting/scaling, etc. Below is the first batch (around the origin). The peaks have the correct orientation, and are very much like the convolution powers we've been generating!

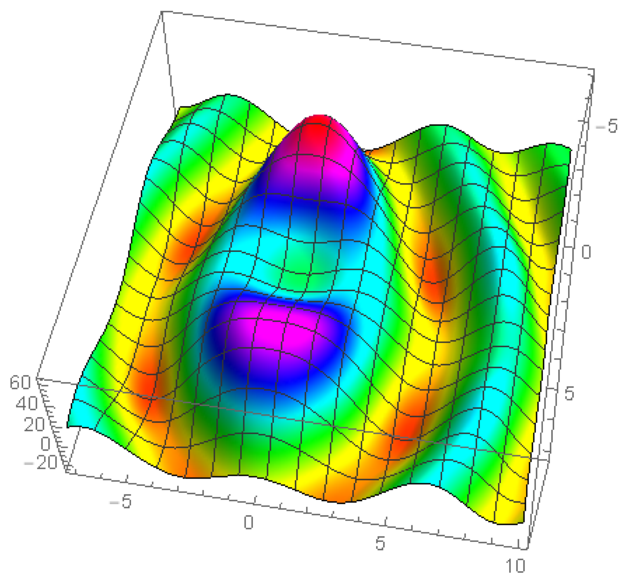
```
In[53]:= ListPlot3D[data, ColorFunction → Hue, PlotRange → Full]
```

Out[53]=



Here's the second batch (a strip adjacent to the center region), appended to the first batch.

Out[99]=



I'll keep doing this a few more times until we get a good enough range.

Did I mention these look very much like the convolution powers we've been generating?

4. Here's the Python code I use to calculate the convolution powers

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal
from matplotlib import cm, colors
from mpl_toolkits.mplot3d.axes3d import Axes3D
import operator
import time
from numpy import unravel_index


def fast_convolve(n_times, support_bound, drift):
    Phi = np.zeros(shape=(9,9),dtype=np.complex_)

    Phi[ 0+9//2][ 0+9//2]  = complex(301/384,-7/48)
    Phi[ 0+9//2][-1+9//2]  = complex(7/96,1/24)
    Phi[ 0+9//2][ 1+9//2]  = complex(3/32,1/24)
    Phi[ 0+9//2][ 2+9//2]  = -1/48
    Phi[ 0+9//2][-2+9//2]  = -1/48
    Phi[ 1+9//2][ 0+9//2]  = complex(7/96,1/24)
    Phi[-1+9//2][ 0+9//2]  = complex(7/96,1/24)
    Phi[ 2+9//2][ 0+9//2]  = -complex(7/192,1/96)
    Phi[-2+9//2][ 0+9//2]  = -complex(7/192,1/96)
    Phi[ 3+9//2][ 0+9//2]  = 1/96
    Phi[-3+9//2][ 0+9//2]  = 1/96
    Phi[ 4+9//2][ 0+9//2]  = -1/768
    Phi[-4+9//2][ 0+9//2]  = -1/768
    Phi[-1+9//2][-1+9//2]  =  1/192
    Phi[ 1+9//2][-1+9//2]  =  1/192
    Phi[ 1+9//2][ 1+9//2]  = -1/192
    Phi[-1+9//2][ 1+9//2]  = -1/192

    conv_power = np.copy(Phi)
    offset = np.array([0,0])

    i=0
    if drift:
        while i < n_times:
            i += 1
            init_vec = unravel_index(np.absolute(conv_power).argmax(), np.absolute(conv_power).shape)
            conv_power = signal.convolve2d(Phi, conv_power, 'full')
            after_vec = unravel_index(np.absolute(conv_power).argmax(), np.absolute(conv_power).shape)
            offset += np.subtract(init_vec , after_vec)

            dim_f = np.shape(conv_power)

            if dim_f[0] > support_bound or dim_f[0] > support_bound:
                conv_power = crop(conv_power, support_bound)
    else:
        while i < n_times:
            i += 1
            conv_power = signal.convolve2d(Phi, conv_power, 'full')
            dim_f = np.shape(conv_power)

            if dim_f[0] > support_bound or dim_f[0] > support_bound:
                conv_power = cropND(conv_power, support_bound)
return conv_power

def cropND(img, sup_bd):
    if sup_bd < np.shape(img)[0] and sup_bd < np.shape(img)[1]:
    dim = np.shape(img)
    return img[(dim[0]//2)-sup_bd//2:(dim[0]//2)+sup_bd//2,
        (dim[1]//2)-sup_bd//2:(dim[1]//2)+sup_bd//2]

def crop(img, sup_bd):
    center = unravel_index(np.absolute(img).argmax(), np.absolute(img).shape)
    return img[center[0]-sup_bd//2:center[0]+sup_bd//2,
        center[1]-sup_bd//2:center[1]+sup_bd//2]

if __name__ == '__main__':

while True:

    n_times = int(input('Convolve how many times? '))
    support_bound = int(input('NxN suppport bound, N = '))
    drift_ans = str(input('Expect asymetric drift? [y/n]: '))
    print('Calculating...')
    start = time.time()

    if drift_ans == 'y':
```

```
        drift = True
    elif drift_ans == 'n':
        drift = False
    else:
        print('WARNING: Write "y" for YES and "n" for NO.')
        print('-----------------------------------------')
        print('\n')
        continue

    data = np.real(fast_convolve(n_times, support_bound, drift))
    dim = np.shape(data)
    x = range((-dim[0]//2)+1,(dim[0]//2)+1)
    y = range((-dim[1]//2)+1,(dim[1]//2)+1)

    hf = plt.figure()
    ha = hf.add_subplot(projection='3d')
    ha.set_xlim(-np.shape(data)[0]//2, np.shape(data)[0]//2)
    ha.set_ylim(-np.shape(data)[0]//2, np.shape(data)[0]//2)

    drift = False # I'm setting this for now for testing
    if drift:
        ha.set_xlabel('\n \n X \n \n DRIFTING CONVOLUTION POWERS!')
        ha.set_ylabel('\n \n Y \n \n DRIFTING CONVOLUTION POWERS!')
        ha.set_zlabel(' \n \n Re(Phi^(n)) for n='+str(n_times))
    else:
        ha.set_xlabel('X')
        ha.set_ylabel('Y')
        ha.set_zlabel(' \n \n Re(Phi^(n)) for n='+str(n_times))

    X, Y = np.meshgrid(x, y)
    surf = ha.plot_surface(X, Y, data , rstride=1, cstride=1, cmap='plasma', edgecolor='none', linewidth=0.2)

    end = time.time()
    print('Time elapsed (s): ', end - start)

    plt.show()
    print('-------------------------------------')
```

5. Here's the Mathematica code that I use to approximate and plot the attractor:

```
H[i_, j_] := NIntegrate[Cos[(-i*x/(100^(1/2)) - j*y/(100^(1/4)) - y^4/96 + y^2*x/96 - x^2/24)],
 {x, -11, 11}, {y, -11, 11}, PrecisionGoal -> 4,
Method -> "OscillatorySelection"]
data = Flatten[
Table[{i, j, H[i, j]}, {i, -7, 7, 0.1}, {j, 7, 10, 0.1}], 1];
Export["ConvolutionPowers/data5.csv", data, "CSV"]


ListPlot3D[Import["ConvolutionPowers/data5.csv"], ColorFunction -> Hue, PlotRange -> Full]
```

The output looks something like