Coding Example: Reaction-Diffusion

This lesson covers another case study called "Reaction-Diffusion" based on the Gary Scott model.

WE'LL COVER THE FOLLOWING

Problem Description

Reaction and diffusion of chemical species can produce a variety of patterns, reminiscent of those often seen in nature. The Gray-Scott equations model such a reaction. For more information on this chemical system see the article Complex Patterns in a Simple System (John E. Pearson, Science, Volume 261, 1993).

Let's consider two chemical species `U` and `V` with **respective concentrations** `u` and `v` and **diffusion rates** `Du` and `Dv`.

`V` is converted into `P` with a rate of conversion `k`. `f` represents the rate of the process that feeds `U` and drains `U`, `V` and `P`. This can be written as:

Chemical reaction	Equations
$U + 2V \rightarrow 3V$	$\dot{u} = Du\nabla^2 u - uv^2 + f(1-u)$
$V \rightarrow P$	$\dot{v} = Dv\nabla^2 v + uv^2 - (f+k)v$

Based on the Game of Life example, we will try to implement such reaction-

diffusion system. Here is a set of interesting parameters to test:

Name	Du	Dv	f	k
Bacteria 1	0.16	0.08	0.035	0.065
Bacteria 2	0.14	0.06	0.035	0.065
Coral	0.16	0.08	0.060	0.062
Fingerprint	0.19	0.05	0.060	0.062
Spirals	0.10	0.10	0.018	0.050
Spirals Dense	0.12	0.08	0.020	0.050
Spirals Fast	0.10	0.16	0.020	0.050
Unstable	0.16	0.08	0.020	0.055
Worms 1	0.16	0.08	0.050	0.065
Worms 2	0.16	0.08	0.054	0.063
Zebrafish	0.16	0.08	0.035	0.060

Complete Solution

Given below, I have implemented a reaction-diffusion system, have a look! It's initially running on "Bacteria 1", you can comment-uncomment a different specie and then run the code!

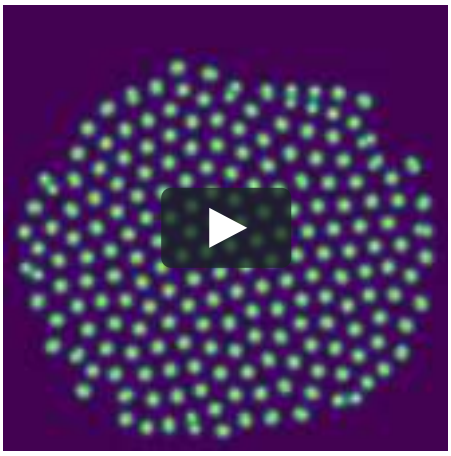
```
# -----------------------------------------------------------------------------
# From Numpy to Python
# Copyright (2017) Nicolas P. Rougier - BSD license
# More information at https://github.com/rougier/numpy-book
# -----------------------------------------------------------------------------
```

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation



# Parameters from http://www.aliensaint.com/uo/java/rd/
# ----------------------------------------------------
n = 256
Du, Dv, F, k = 0.16, 0.08, 0.035, 0.065  # Bacteria 1
# Du, Dv, F, k = 0.14, 0.06, 0.035, 0.065  # Bacteria 2
# Du, Dv, F, k = 0.16, 0.08, 0.060, 0.062  # Coral
# Du, Dv, F, k = 0.19, 0.05, 0.060, 0.062  # Fingerprint
# Du, Dv, F, k = 0.10, 0.10, 0.018, 0.050  # Spirals
# Du, Dv, F, k = 0.12, 0.08, 0.020, 0.050  # Spirals Dense
# Du, Dv, F, k = 0.10, 0.16, 0.020, 0.050  # Spirals Fast
# Du, Dv, F, k = 0.16, 0.08, 0.020, 0.055  # Unstable
# Du, Dv, F, k = 0.16, 0.08, 0.050, 0.065  # Worms 1
# Du, Dv, F, k = 0.16, 0.08, 0.054, 0.063  # Worms 2
# Du, Dv, F, k = 0.16, 0.08, 0.035, 0.060  # Zebrafish


Z = np.zeros((n+2, n+2), [('U', np.double),
                          ('V', np.double)])
U, V = Z['U'], Z['V']
u, v = U[1:-1, 1:-1], V[1:-1, 1:-1]


r = 20
u[...] = 1.0
U[n//2-r:n//2+r, n//2-r:n//2+r] = 0.50
V[n//2-r:n//2+r, n//2-r:n//2+r] = 0.25
u += 0.05*np.random.uniform(-1, +1, (n, n))
v += 0.05*np.random.uniform(-1, +1, (n, n))


def update(frame):
    global U, V, u, v, im

    for i in range(10):
        Lu = (                  U[0:-2, 1:-1] +
              U[1:-1, 0:-2] - 4*U[1:-1, 1:-1] + U[1:-1, 2:] +
                                U[2:  , 1:-1])
        Lv = (                  V[0:-2, 1:-1] +
              V[1:-1, 0:-2] - 4*V[1:-1, 1:-1] + V[1:-1, 2:] +
                                V[2:  , 1:-1])
        uvv = u*v*v
        # Based on the equations for chemical reactions mentioned in the above table
        u += (Du*Lu - uvv + F*(1-u))
        v += (Dv*Lv + uvv - (F+k)*v)

    im.set_data(V)
    im.set_clim(vmin=V.min(), vmax=V.max())

fig = plt.figure(figsize=(4, 4))
fig.add_axes([0.0, 0.0, 1.0, 1.0], frameon=False)
im = plt.imshow(V, interpolation='bicubic', cmap=plt.cm.viridis)
plt.xticks([]), plt.yticks([])

Writer = animation.writers['ffmpeg']
writer = Writer(fps=15, metadata=dict(artist='Me'), bitrate=1800)

anim = animation.FuncAnimation(fig, update, interval=10, frames=100)
```

```
anim.save('output/output.mp4', writer=writer)

# animation.save('gray-scott-1.mp4', fps=40, dpi=80, bitrate=-1, codec="libx264",
#                extra_args=['-pix_fmt', 'yuv420p'],
#                metadata={'artist':'Nicolas P. Rougier'})
plt.show()
```

Output:

The figure below shows some animations of the model for a specific set of parameters. The output of above code would look exactly like this if you run it and download the file that it generates.



Further Readings

- John Conway new solitaire game “life” Martin Gardner, Scientific American 223, 1970.
- Gray Scott Model of Reaction Diffusion, Abelson, Adams, Coore, Hanson, Nagpal, Sussman, 1997.
- Reaction-Diffusion by the Gray-Scott Model, Robert P. Munafo, 1996.

Let’s look at temporal vectorization in the next lesson.