```
matplotlib.rc("image",cmap="gray")

import urllib2
from urllib2 import urlopen
from StringIO import StringIO
from PIL import Image
def fig(url):
    data = urlopen(url).read()
    axis("off"); return imshow(Image.open(StringIO(data)))
from pylab import *
from scipy.ndimage import filters,measurements
import pygame
from pygame import surfarray
matplotlib.rc("image",cmap="gray")
matplotlib.rc("image",interpolation="nearest")
pygame.init()
OR = logical_or
AND = logical_and
def unif(lo,hi,size=1):
    if size==1: return rand()*(hi-lo)+lo
    if type(size)==int: return rand(size)*(hi-lo)+lo
    return rand(*size)*(hi-lo)+lo
def animate(images,n=20,k=1):
    for i,image in enumerate(images):
        if i>=n: break
        if i%k!=0: continue
        b = array(255*clip(image,0,1),'B')
```

Basic Reaction Diffusion Systems

General reaction-diffusion equation:

$$\frac{\partial q}{\partial t} = D\nabla^2 q + R(q)$$

This is like the simple diffusion equation, but with two differences:

- q may be a vector, so it is multiple signals that diffuse
- there is an additional term $R(q)$ that represents reactions

The matrix D is diagonal and gives the diffusion coefficient for each species.

The fact that it is diagonal means that the different substances don't change into each other.

One-Components Systems

Simple diffusion $R(q) = 0$:

$$\frac{\partial q}{\partial t} = D\frac{\partial^2 q}{\partial x^2}$$

We already derived this:

- arises from Gaussian random walk
- the change at x over time is the stuff that diffuses out to the left and right minus the stuff that diffuses in from the left and right

```
q = zeros(1000)
q[400:600] = 1.0
result = []
for i in range(20000):
    if i%20==0: result.append(q.copy())
    q += 0.2*laplacian(q)
subplot(121); imshow(array(result))
subplot(122); ylim((-.1,1.1)); plot(result[0]); plot(result[500]);
    plot(result[-1])
```

Fisher's equation:

$$\frac{\partial q}{\partial t} = D\frac{\partial^2 q}{\partial x^2} + q(1-q)$$

Motivation:

- animal populations
- diffusion term: spatial "diffusion" / migration of individuals
- logistic term: growth of advantageous allele over original wild type

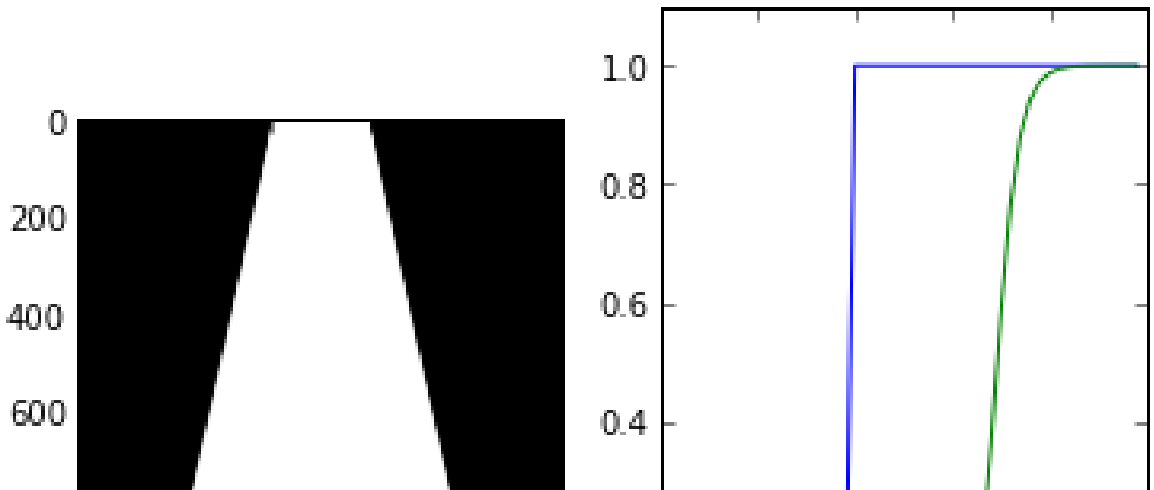
Behavior:

- traveling wave solutions

1D Fisher equation - traveling wavefront

```
q = zeros(1000)
q[400:600] = 1.0
result = []
for i in range(20000):
    if i%20==0: result.append(q.copy())
    q += 0.01*(laplacian(q)+q*(1-q))
result = array(result); print amin(result),amax(result)
subplot(121); imshow(array(result))
subplot(122); ylim((-.1,1.1)); diffusionplot(result[0][380:430]);
    plot(result[-1][150:200])
```

```
0.0 1.0
```



2D Fisher equation - traveling wavefront

```
q = zeros((w,h))
q[w//2:w//2+10,h//2:h//2+10] = 1N = 100
for trial in range(10000):
    u = 0.5*(1+0.1*randn(N))
    v = 0.1*(1+0.1*randn(N))
    a = exprand(0.001,0.1)
    b = exprand(0.001,0.1)
    r = exprand(0.001,0.1)
    for t in range(10000):
        u += 0.02*laplacian(u)
        v += 0.04*laplacian(v)
        u -= r*u*v**2
        v += r*u*v**2
        u += a*(1-u)
        v -= b*v
    delta = amax(u)-amin(u)
    if delta>0.9:
        print a,b,r,delta
        break
q[w//2+30:w//2+40,h//2:h//2+10] = 1
def rd():
    global q
    for i in range(50000):
        yield q
```

Turing Systems

Two components:

Let's consider two components now. This turns the reaction-diffusion equation into:

$$\begin{pmatrix} \partial_t u \\ \\ \partial_t v \end{pmatrix} = \begin{pmatrix} D_u & 0 \\ \\ 0 & D_v \end{pmatrix} \begin{pmatrix} \partial_{xx} u \\ \\ \partial_{xx} v \end{pmatrix} + \begin{pmatrix} F(u,v) \\ \\ G(u,v) \end{pmatrix}$$

Turing systems:

- two morphogens
- first morphogen is autocatalytic and diffuses slower
- second morphogen is inhibitory and diffuses faster

Similar to the three component system we discussed before, but local growth, slow and fast diffusion are mixed up.

Reaction Kinetics:

Chemical reactions:

$$R \xrightarrow{k_1} 2R$$

$$F + R \xrightarrow{k_2} 2F$$

$$F \xrightarrow{k_3} D$$

Differential equations:

$$[\dot{R}] = k_1[R] - k_2[F][R]$$

$$[\dot{F}] = k_2[F][R] - k_3 F$$

Gray-Scott Systems:

Autocatalytic reaction: $U + 2V \rightarrow 3V$

Corresponding rates: $[\dot{U}] = -[U][V]^2$

Replenishment: $[\dot{U}] = f\ (1 - [U])$

Destruction: $[\dot{V}] = -(f + k)\ [V]$

Equations:

$$\dot{u} = r_u \nabla^2 u - uv^2 + f(1 - u)$$

$$\dot{v} = r_v \nabla^2 v + uv^2 - (f + k)v$$

(Examples: later)

Brusselator:

$$A \rightarrow X$$

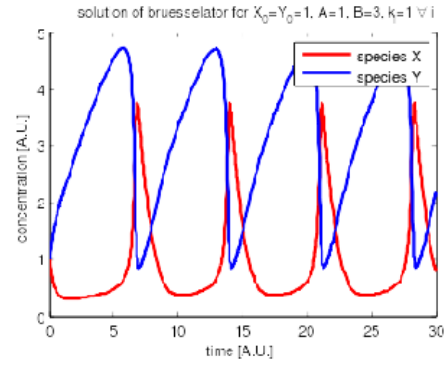
$$2X + Y \rightarrow 3X$$

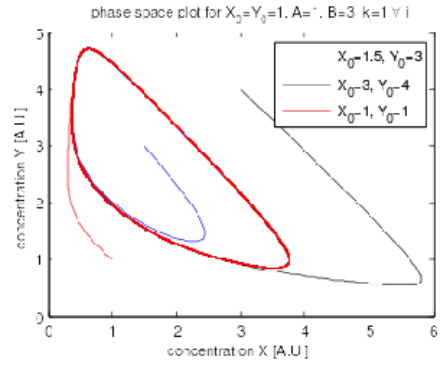
$$B + X \rightarrow Y + D$$

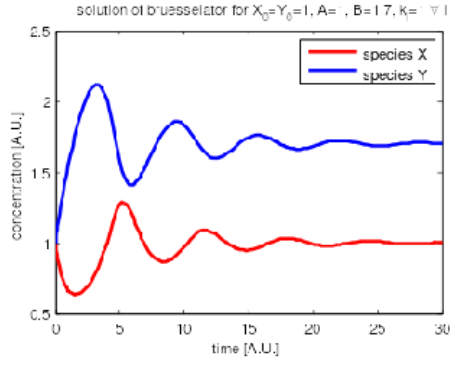
$$X \rightarrow E$$

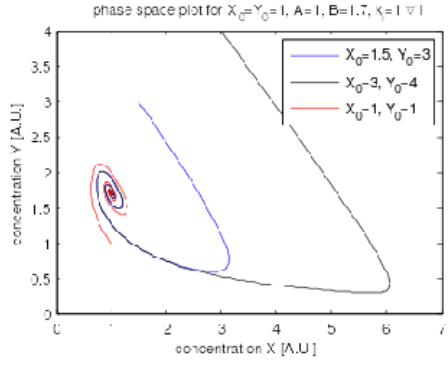
(Note that concentrations of A , B , C , and D determine rate constants.)

http://iupr1.cs.uni-kl.de/ tmb/simso-videos/brusselator.mp4


species X
species Y
concentration [A.U.]
time [A.U.]


X0=1.5, Y0=3
X0=3, Y0=4
X0=1, Y0=1
concentration Y [A.U.]
concentration X [A.U.]


species X
species Y
concentration [A.U.]
time [A.U.]

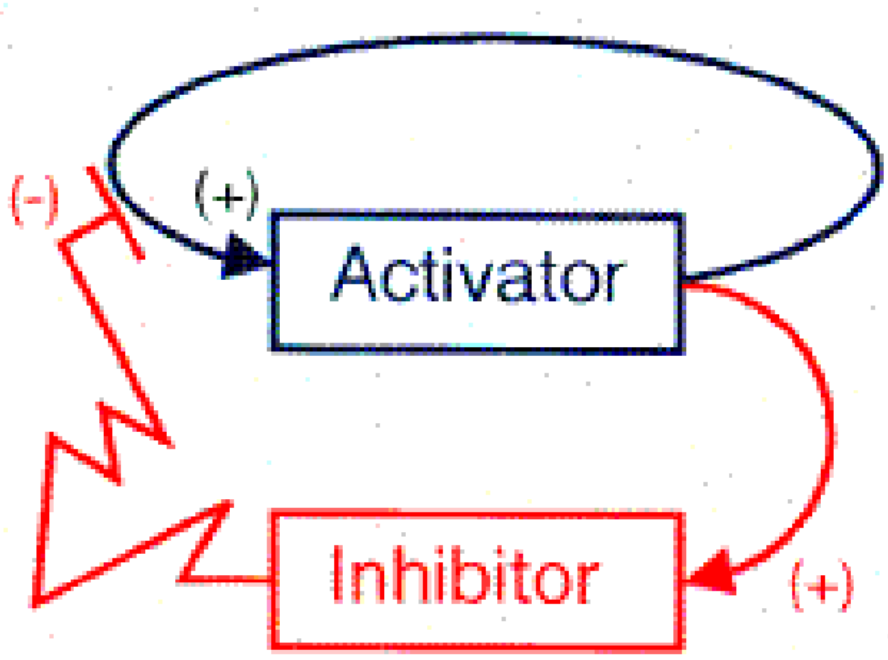

X0=1.5, Y0=3
X0=3, Y0=4
X0=1, Y0=1
concentration Y [A.U.]
concentration X [A.U.]

Gierer-Meinhardt Model:

$$\dot{a} = \alpha a^2 - \mu_a a + D_a \nabla^2 a + \rho_a$$
$$\dot{h} = \beta a^2 - \mu_h h + D_h \nabla^2 h + \rho_h$$

http://iupr1.cs.uni-kl.de/ tmb/simso-videos/gierer-meinhard-spots.mp4

http://iupr1.cs.uni-kl.de/ tmb/simso-videos/gierer-meinhardt-labyrinth.mp4


(-)
(+)
Activator
Inhibitor
(+)

FitzHugh-Nagumo Equation with Diffusion:

We already discussed this as a model of the action potential.

$$\dot{u} = D_u \nabla^2 u + f(u) - sv$$

$$\tau \dot{v} = D_v \nabla^2 v + u - v$$

With:

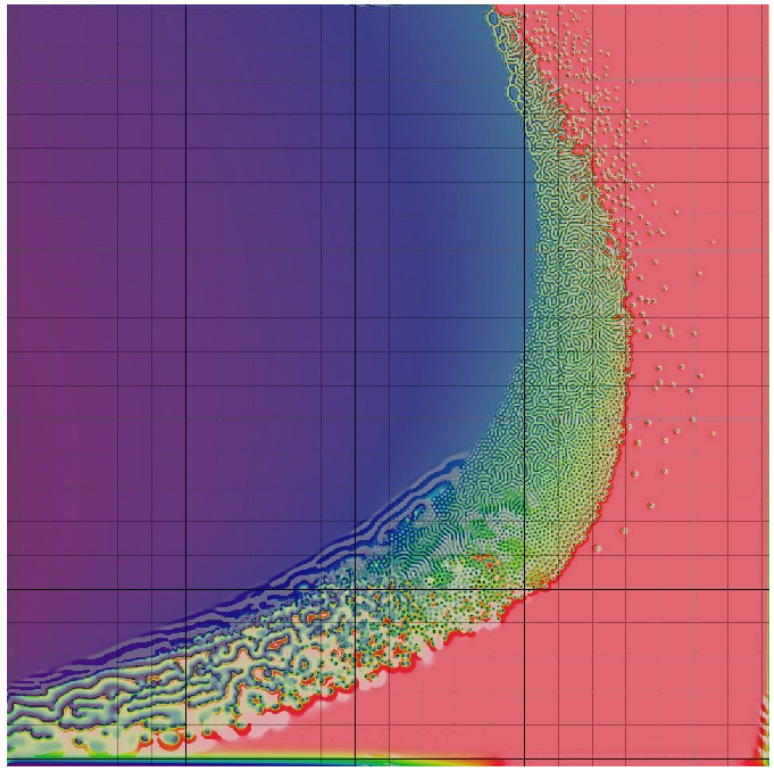
$$f(u) = u^3 - \kappa$$

Here, the "activator" is the sodium channels, and the "inhibitor" is the potassium channels.

http://iupr1.cs.uni-kl.de/ tmb/simso-videos/fitzhugh-tip-splitting.mp4

Exploring Gray Scott

Finding parameters that work can be tricky. A simple method that often works is to run the equations on a variable parameter space.

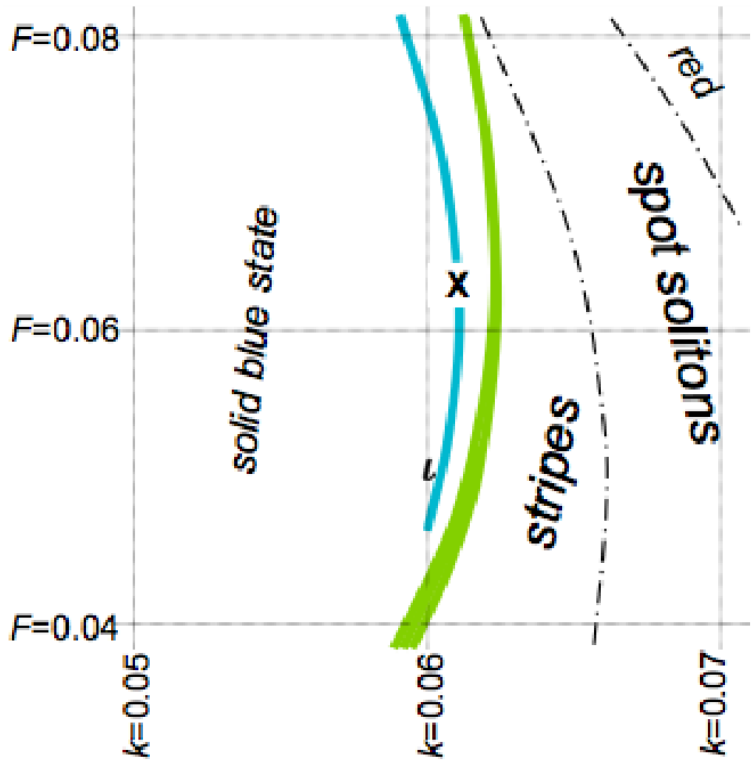


You can explore this space here:

http://mrob.com/pub/comp/xmorphia/formula

http://iupr1.cs.uni-kl.de/ tmb/simso-videos/Gray-Scott-F300-k630-fr47.mp4

http://iupr1.cs.uni-kl.de/ tmb/simso-videos/Gray-Scott-F380-k630-fr64.mp4


F=0.08
F=0.06
F=0.04
k=0.05
k=0.06
k=0.07
solid blue state
stripes
spot solitons
red

U-Skate World:

http://iupr1.cs.uni-kl.de/ tmb/simso-videos/uskate-world-complexity.mp4