The use of reaction diffusion equations for the simulation of physical phenomena

Simon Abrelat

002129–0004

Math HL IA

May 2019

Words:

Abstract

This is the thing working good (Turing, 1952), here is another example (P.Gray, 1984)

Contents

1 Introduction

2 Calculus

Reaction diffusion relies on multivariable calculus, from the partial derivative to the laplacian operator that governs diffusion. Multivariable calculus is a natural extension of calculus where instead of focusing on the accumulation and change on 1 axis, that being y, it is possible to do in to any number of dimensions. This means that if your function has more than 1 input, like for 3D functions, you would have a new arsenal of tools and methods to analyse your equations. The most simple being the limit which fundamentally operates the same as in the normal case. If there is a hole in the function but otherwise is continuous, it would "fill in" that hole, and if there is a jump discontinuity the limit is defined some certain directions but the limit itself is undefined. Once again using the concept of limits we can get the integral and derivative as in normal calculus; however for this application we will be primarily focusing on partial derivatives and other related operations. Instead derivatives in in multivariate partial derivatives are used. These are computed in a similar form to derivatives but the input that is not differentiated is treated as a constant. For use in the reaction equations the partial differential equations behave in a similar way todifferential equations but rely on more inputs but are still the rate of change in a given input over

a function.

$$\frac{\partial f}{\partial x} = \partial_x f = \lim_{h \to 0} \frac{f(x+h, y) - f(x, y)}{h} \tag{1}$$

$$\frac{\partial f}{\partial y} = \partial_y f = \lim_{h \to 0} \frac{f(x, y+h) - f(z, y)}{h} \tag{2}$$

Example problem

$$f(x, y) = xy^2 + x$$

$$\partial_x f(x, y) = y^2 + 1$$

$$\partial_y f(x, y) = 2xy$$

Then there are directional derivatives which are similar to the partial derivatives, but instead of being in a component direction like along the x or y direction it would be in the direction of the vector $\vec{u}$ with components $< a, b >$. These derivates can be done in any number of dimensions (2) but for reaction diffusion formulas shown 2D cases (1) matter more.

$$D_u f = \lim_{h \to 0} \frac{f(x + ha, y + hb) - f(x, y)}{h} \tag{3}$$

$$D_u f(\vec{x}) = \lim_{h \to 0} \frac{f(\vec{x} + hu) - f(\vec{x})}{h} \tag{4}$$

The directional derivative can also be written as a dot products

$$\begin{aligned} D_u f(x,y) &= a\partial_x f + b\partial_y f \\ &=< \partial_x f, \partial_y f > * < a, b > \\ &=< \partial_x f, \partial_y f > *u \\ &= \nabla f(x,y) * u \end{aligned}$$

This ∇, or nabla, is the sign for the grad operator read "del f" This represents a vector of all of the partial derivatives that are possible in a function called the gradient. For example in $g(x,y,z)$, grad g= the partial derivative of g in respect to x, y, and z. This gradient shoulds the direction of the fastest increase of the function. So if we had a map of a mountain range the gradient of the map from a point would be the fastest way to get to the top of the nearest mountain but not necessarily the highest mountain. The gradient is often used for optimization problems since it can go "the highest" or most optimal local point in a function of any number of variables. A common abuse of notation is to set ∇ to a value (3)

$$\nabla =< \frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} > \tag{5}$$

So the notation for the gradient makes since because you would be treating the function f as a scalar and multiplying it to the ∇ vector. This abuse of notation since it would also make sense for the divergence of a vector

field. For a vector field the function F (4) would be a vector function where you could dot product of F and ∇ (5)

$$\mathbf{F} = P\hat{\imath} + Q\hat{\jmath} + R\hat{k} \tag{6}$$

$$div\mathbf{F} = \nabla * \mathbf{F} = \frac{\partial P}{\partial x} + \frac{\partial Q}{\partial y} + \frac{\partial R}{\partial z} \tag{7}$$

One can picture a vector field as being a model of fluid flow, where the vector at a point would be the velocity of the fluid. Given this analogy, the divergence if the fluid is incompressible then the amount of fluid that emanates from a point. In a vector field that would represented by all of the arrows pointing away from a point and the arrows get larger around a point then the divergence > 0 and if the arrows are smaller or point inwards than the divergence < 0. Both of these tools are useful but when applied together become what is needed for the diffusion part of reaction diffusion that being the laplacian. The laplacian is the divergence of the gradient of a function.

$$\text{Laplacian} = \nabla * (\nabla f) = \nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}\frac{\partial^2 f}{\partial z^2} = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial x_i^2} \qquad \nabla^2 = \nabla * \nabla$$

This is called the Laplace operator or the Laplacian of a field because of its relation to Laplace's Equation which has similarities to the heat transfer equation for diffusion. The reason that when put in PDEs they exhibit this diffusion or spreading properly is due to their relation to the second derivative in normal calculus. They identify local maxima and minima but in normal calculus where the second derivative is 0 in both cases the laplacian is highly

positive in minima and highly negative in maxima. In the case of PDEs, that would mean that the values would "move away" from maxima and "fill in" minima which exactly describes osmosis and other diffusion systems.

$$\text{Laplace's Equation: } \nabla^2\mathbf{F} = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2} = 0$$

Equations that satisfies Laplace's Equation are called harmonic functions where at all points the value of the laplacian = 0

3 Diffusion

4 Reaction

5 Reaction Diffusion

6 References

References

P.Gray, S. (1984). Autocatalytic reactions in the isothermal, continuous stirred tank reactor: Oscillations and instabilities in the system a + 2b to 3b; b to c. *Chemical Engineering Science*, 39(6):1087–1097.

Turing, A. (1952). The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London*, 237(641):37–72.

7 Appedix

Listing 1: Diffusion

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

# Simulation constants
alpha = 0.00001  # Laplician multiplier
n = 2000  # number of iterations
s = 100  # size of the matrix
dx = 2/s  # Simulation speed/accuracy

# Matrix initialization
M = np.empty((s, s))
M.fill(0)
x, y = M.shape
# Hotspot Seeding
M[20:30, 45:50] = 90
M[70:80, 45:50] = 90
np.set_printoptions(precision=1)


# Implementation of the decretized laplacian operator
def laplacian(Z):
    Ztop = Z[0:-2, 1:-1]
    Zleft = Z[1:-1, 0:-2]
    Zbottom = Z[2:, 1:-1]
    Zright = Z[1:-1, 2:]
    Zcenter = Z[1:-1, 1:-1]
    return (Ztop + Zleft + Zbottom + Zright - (4 * Zcenter)) / dx**2


```

```
# Iterations through PDE
for _ in range(n):
    Mn = M[1:-1, 1:-1]
    M[1:-1, 1:-1] = Mn + np.multiply(laplacian(M), alpha)

# Graph Data
plt.imshow(M, cmap=plt.cm.coolwarm,
           interpolation='bilinear', extent=[-1, 1, -1, 1])
plt.show()
```

Listing 2: Lotka-Volterra

```
import matplotlib.pyplot as plt

# Model params
alpha = 2/3
beta = 4/3
gamma = 1
delta = 1

# Simulation constants: descrete time and space
dt = 0.001  # the change in time
n = 15000  # number of iterations

# Seed the space
U = 1  # Population of Pray
V = 0.1  # Population of Preditors
# output Arrays
Uy = []
Vy = []
X = []

# Simulate this ODE with Euler's method
for i in range(n):
```

```
    # Gets copies to prevent overwritting
    Uc = U
    Vc = V
    # Generate next time step
    U += (alpha*Uc - beta*Uc*Vc)*dt
    V += (gamma*Uc*Vc - delta*Vc)*dt
    # Record outputs
    Uy.append(U)
    Vy.append(V)
    X.append(i*dt)

# Graph Data
plt.title("Lotka-Volterra Model")
plt.plot(X, Uy, 'b', label="pray")
plt.plot(X, Vy, 'r', label="preditor")
plt.legend()
plt.show()
```

Listing 3: SIR

```
import matplotlib.pyplot as plt

# Model params
b = 1/2  # days of contact to spread
k = 1/4  # days to recover

# Simulation params
dt = 0.1  # change in time
n = 1400  # number of iterations

# Seeding
s = 1  # suseptible population
i = 1.27E-4  # Infected
r = 0  # Recovered
```

```
# To graph and record values
Sy = []
Iy = []
Ry = []
X = []

# Simulate ODE with Euler's method
for x in range(n):
    # Edge case
    if i <= 0:
        print("Done")
        break
    # copy to prevent overwritting
    Sc = s
    Ic = i
    Rc = r
    # Generate next timestep
    s += (-b * Sc * Ic) * dt
    i += ((b * Sc * Ic) - (k * Ic)) * dt
    r += (k * Ic) * dt
    # Record values
    Sy.append(s)
    Iy.append(i)
    Ry.append(r)
    X.append(x*dt)

# Graph Data
plt.title("SIR model")
plt.plot(X, Sy, 'g', label="Suseptible")
plt.plot(X, Iy, 'r', label="Infected")
plt.plot(X, Ry, 'b', label="Recovered")
plt.legend()
plt.show()
```