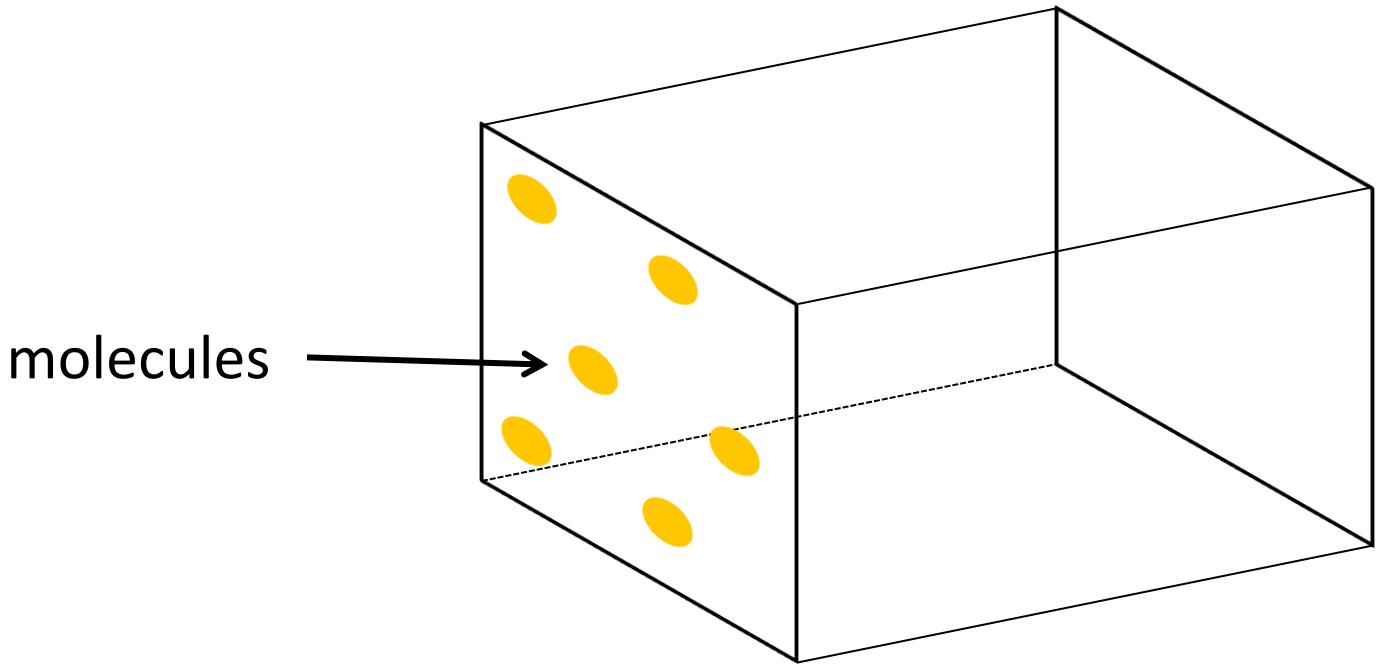
PFE 9

Palubicki

Diffusion

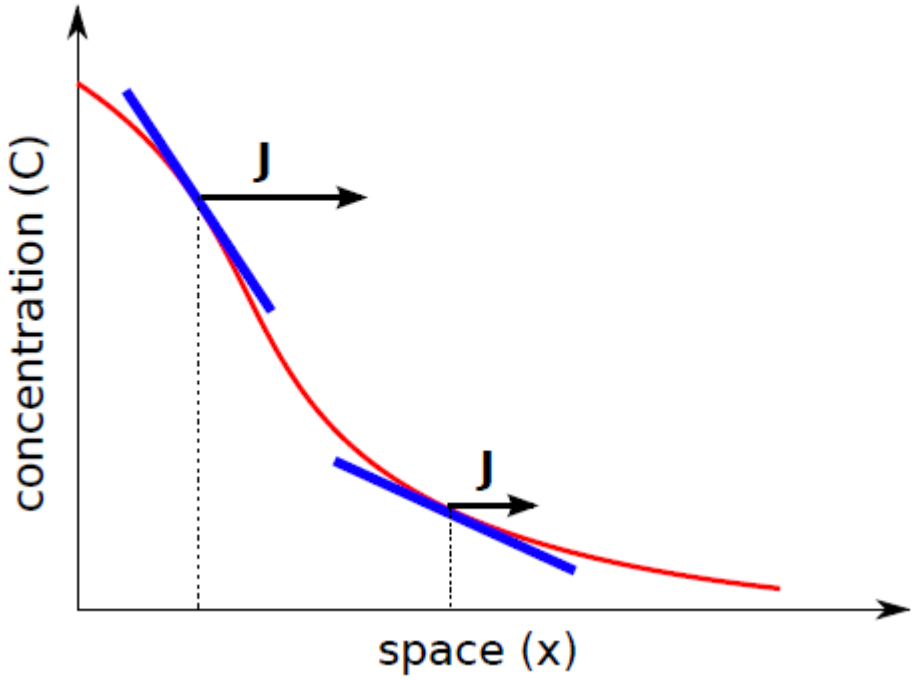


Ficks First Law

$$\dot{V} = \frac{(c_1 - c_2) \cdot A \cdot D}{x}$$

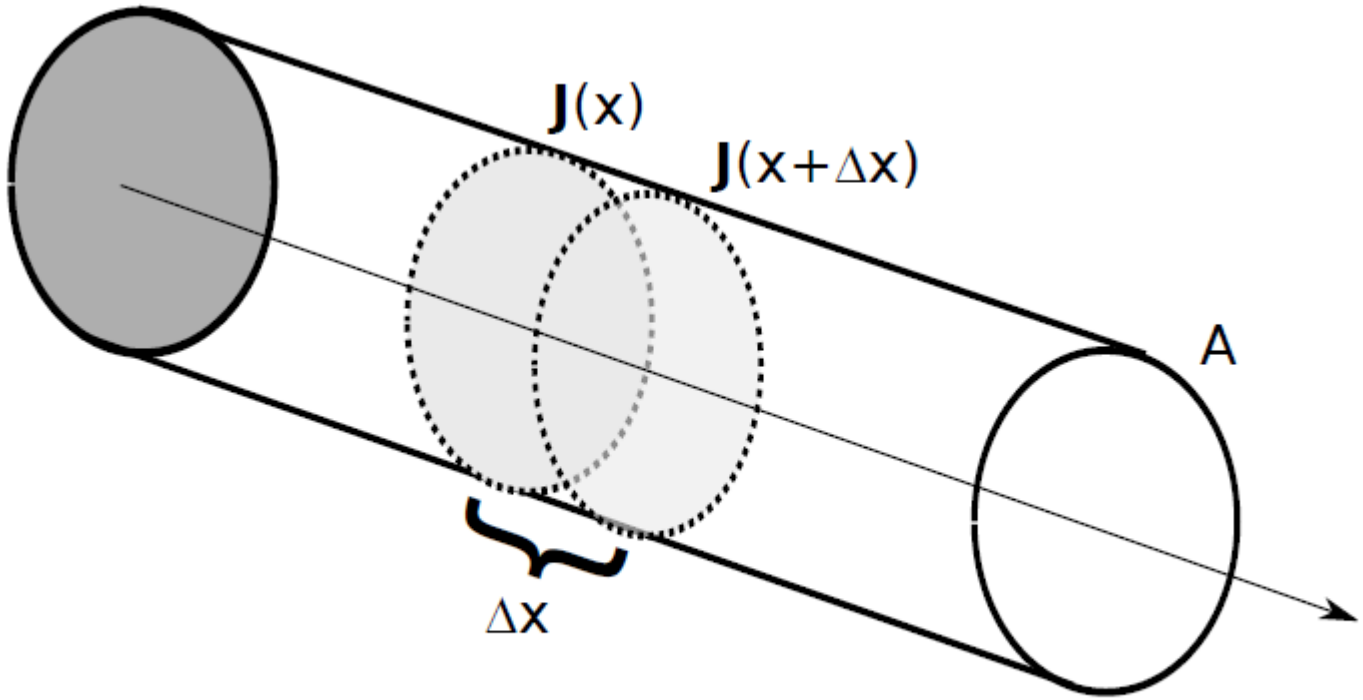
$$J = -D\nabla c$$

Flux Gradient



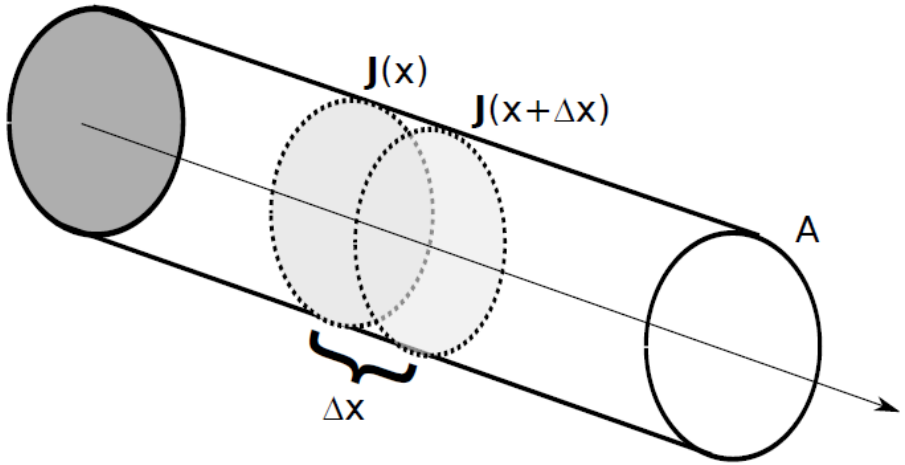
Continuity equation

- How does the number of particles change at a given position?
- For a cylinder at a position x to $x + \Delta x$:



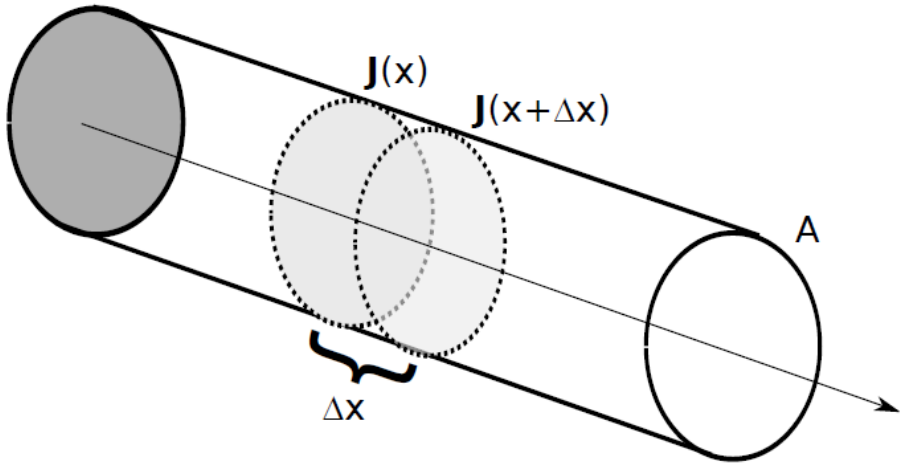
Continuity equation

- Number of particles:
  - $N = cA\Delta x$
  - Where A is the area of the cross-section
  - c, concentration of particles
  - Δx, length of the cylinder



Continuity equation

- Number of particles:
  - $N = cA\Delta x$
  - Where A is the area of the cross-section
  - c, concentration of particles
  - Δx, length of the cylinder
- Then the rate of change of the number of particles over time t is given by the fluxes at positions x and $x + \Delta x$:
- $\frac{\partial(cA\Delta x)}{\partial t} = AJ(x) - AJ(x + \Delta x) \pm \sigma A\Delta x$



Continuity equation

- $\frac{\partial(cA\Delta x)}{\partial t} = AJ(x) - AJ(x + \Delta x) \pm \sigma A\Delta x$

Continuity equation

- $\frac{\partial(cA\Delta x)}{\partial t} = AJ(x) - AJ(x+\Delta x) \pm \sigma A\Delta x$

- $\frac{\partial c(x,t)}{\partial t} = \frac{J(x)-J(x+\Delta x)}{\Delta x} \pm \sigma$

Continuity equation

- $\frac{\partial(cA\Delta x)}{\partial t} = AJ(x) - AJ(x+\Delta x) \pm \sigma A\Delta x$
- $\frac{\partial c(x,t)}{\partial t} = \frac{J(x)-J(x+\Delta x)}{\Delta x} \pm \sigma$
- When Δx is becoming close to zero:
- $\frac{\partial c}{\partial t} = -\frac{\partial J}{\partial x} \pm \sigma$

Continuity equation

- $\frac{\partial(cA\Delta x)}{\partial t} = AJ(x) - AJ(x+\Delta x) \pm \sigma A\Delta x$
- $\frac{\partial c(x,t)}{\partial t} = \frac{J(x)-J(x+\Delta x)}{\Delta x} \pm \sigma$
- When Δx is becoming close to zero:
- $\frac{\partial c}{\partial t} = -\frac{\partial J}{\partial x} \pm \sigma = \frac{\partial\left(D\frac{\partial c}{\partial x}\right)}{\partial x} \pm \sigma$

Continuity equation

- $\frac{\partial(cA\Delta x)}{\partial t} = AJ(x) - AJ(x+\Delta x) \pm \sigma A\Delta x$
- $\frac{\partial c(x,t)}{\partial t} = \frac{J(x)-J(x+\Delta x)}{\Delta x} \pm \sigma$
- When Δx is becoming close to zero:
- $\frac{\partial c}{\partial t} = -\frac{\partial J}{\partial x} \pm \sigma = \frac{\partial\left(D\frac{\partial c}{\partial x}\right)}{\partial x} \pm \sigma = D\frac{\partial^2 c}{\partial x^2} \pm \sigma$

Ficks 2nd Law of Diffusion

Laplace Operator

- The Laplace operator is a **second-order differential operator** in the **n-dimensional Euclidean space**, defined as the **divergence** $(\nabla \cdot)$ of the **gradient** (∇f)

- $\Delta f(x, y) = \nabla \cdot \nabla f$

Gradient ∇f

- The gradient of a scalar-valued differentiable function f of several variables, is a vector-valued function ∇ f : $R^n \rightarrow R^n$ whose value at a point is a tangent vector to f.

$$\nabla f = \frac{\partial f}{\partial x}\mathbf{i} + \frac{\partial f}{\partial y}\mathbf{j}$$

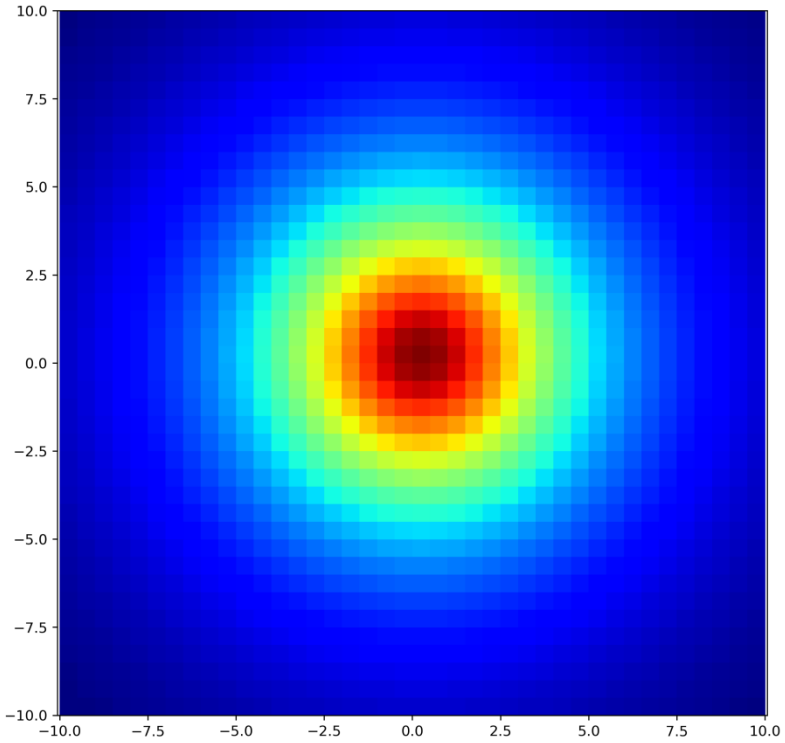
where **i**, **j** are the standard unit vectors in the directions of the *x*, *y* coordinates

Example

```
x = y = np.linspace(-10., 10., 41)
xv, yv = np.meshgrid(x, y, indexing=’ij’)
fv = h0/(1 + (xv**2+yv**2)/(R**2)) # Some function
```

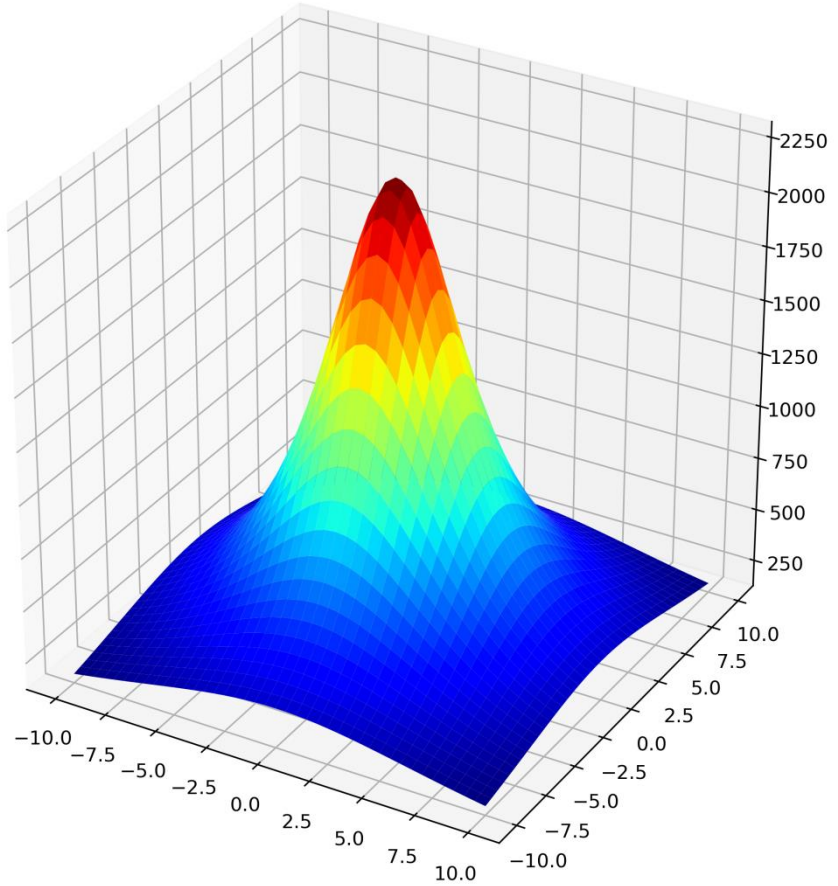
Example

```
plt.pcolormesh(x,y,fv, cmap = 'jet')
```



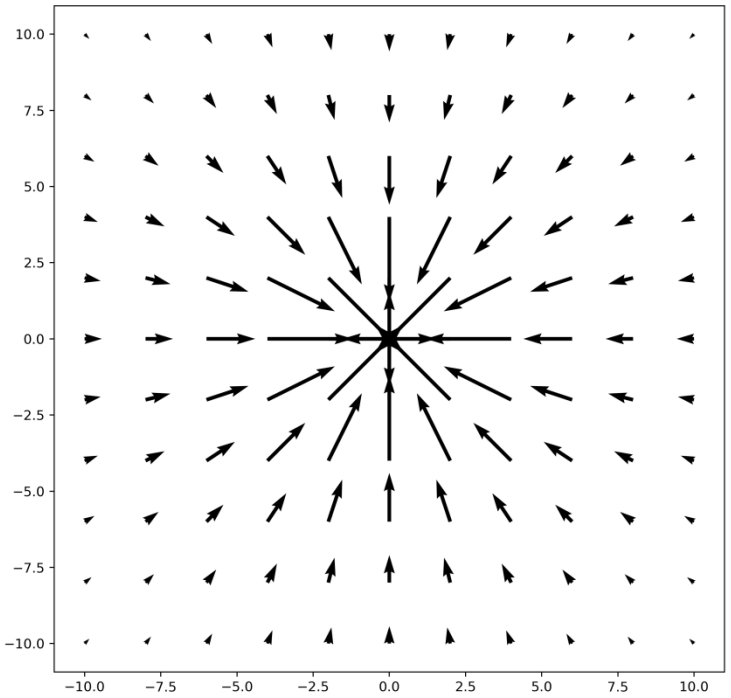
Example

```
ax.plot_surface(xv, yv, fv, cmap='jet')
```



Gradient Computation ∇f

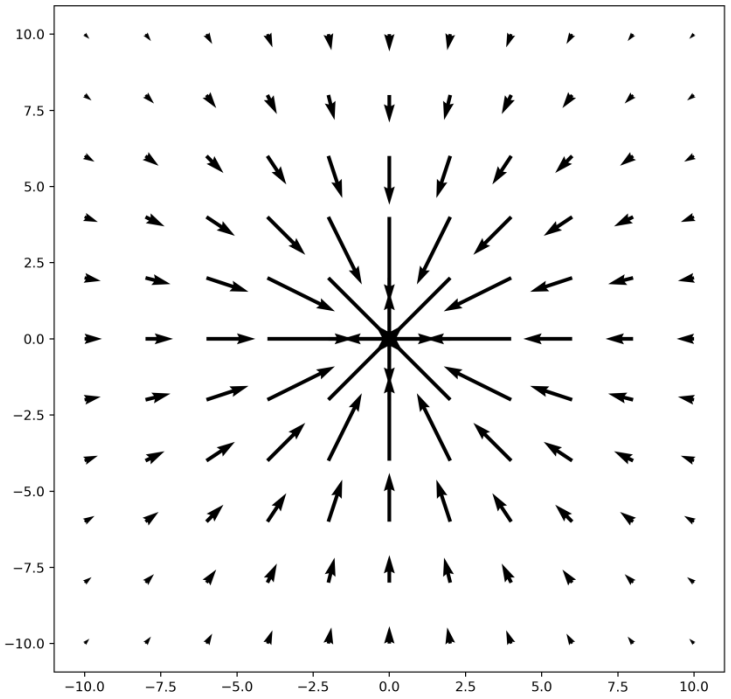
```
dhdx, dhdy = np.gradient(fv) # dh/dx, dh/dy
```



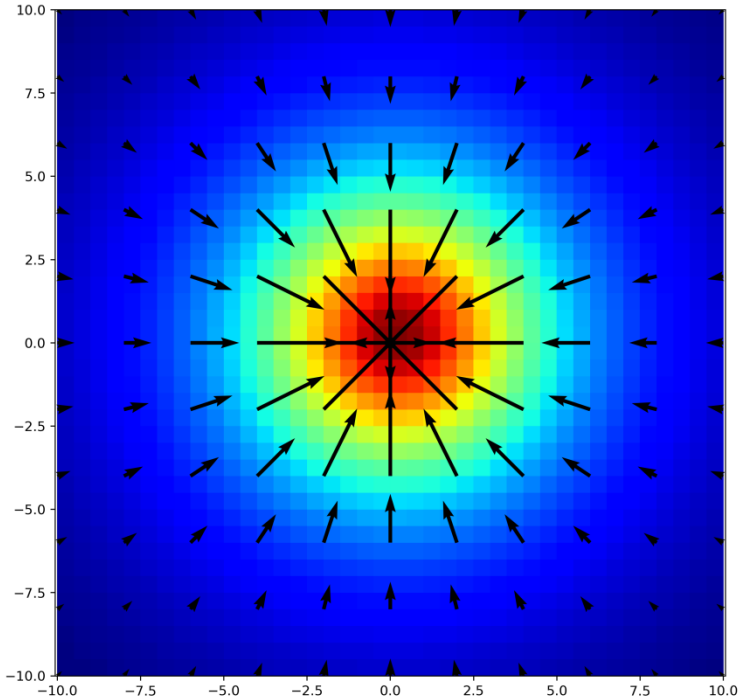
quiver

Gradient Computation ∇f

```
dhdx, dhdy = np.gradient(fv) # dh/dx, dh/d
```



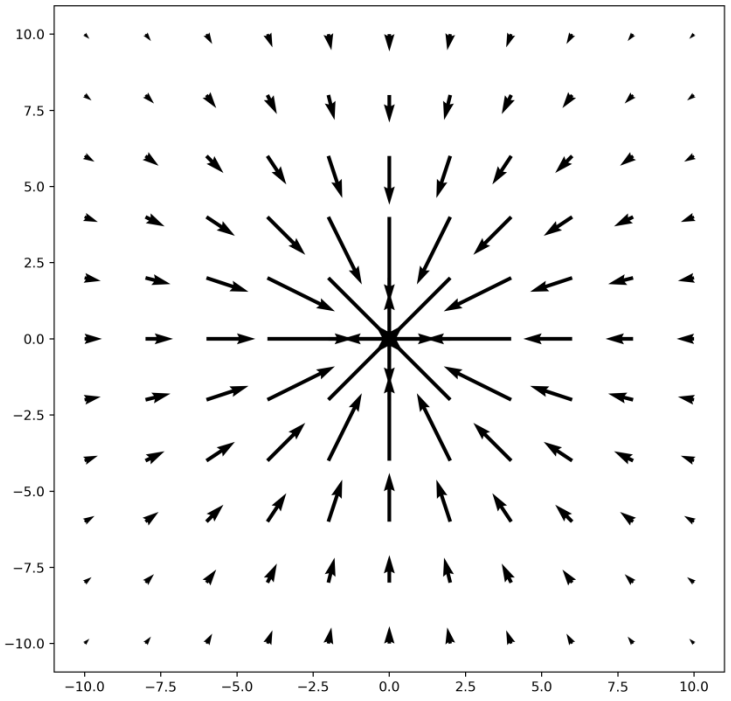
quiver



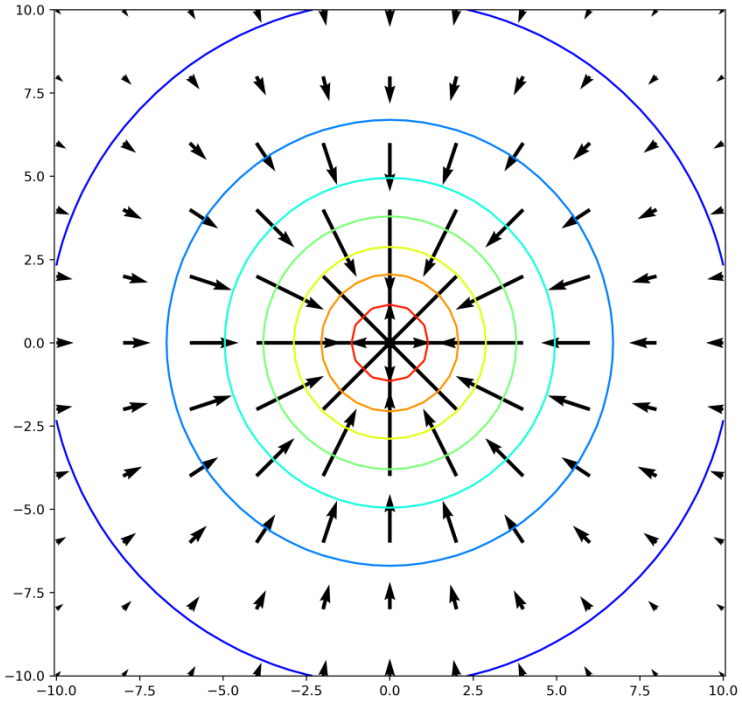
pcolormesh + quiver

Gradient Computation ∇f

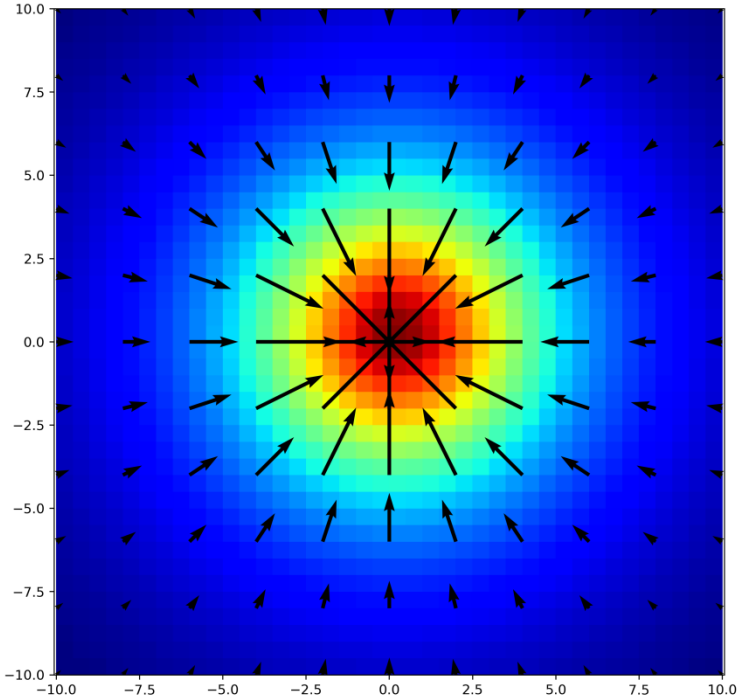
```
dhdx, dhdy = np.gradient(fv) # dh/dx, dh/d
```



quiver



contour + quiver

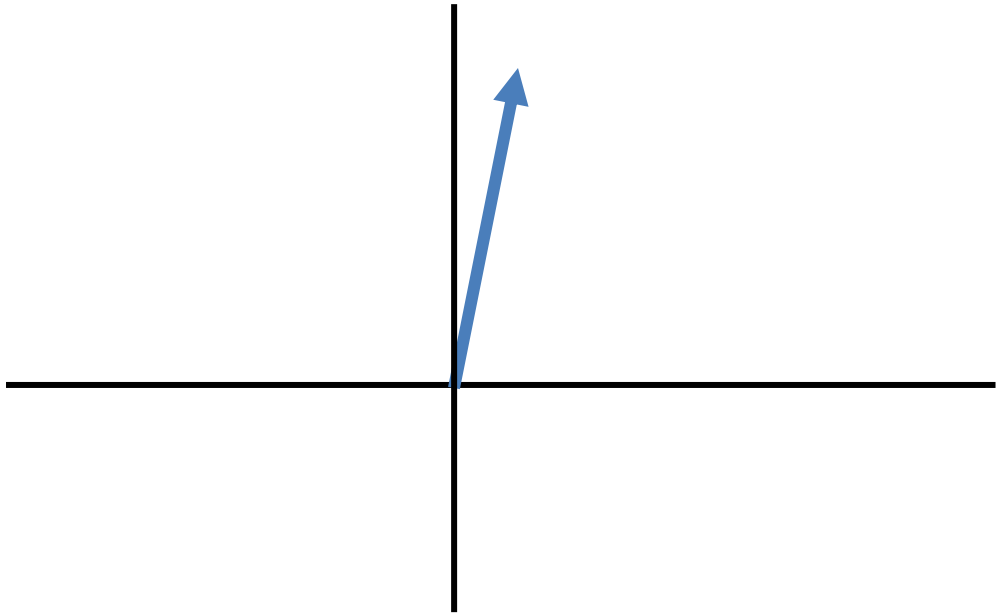


pcolormesh + quiver

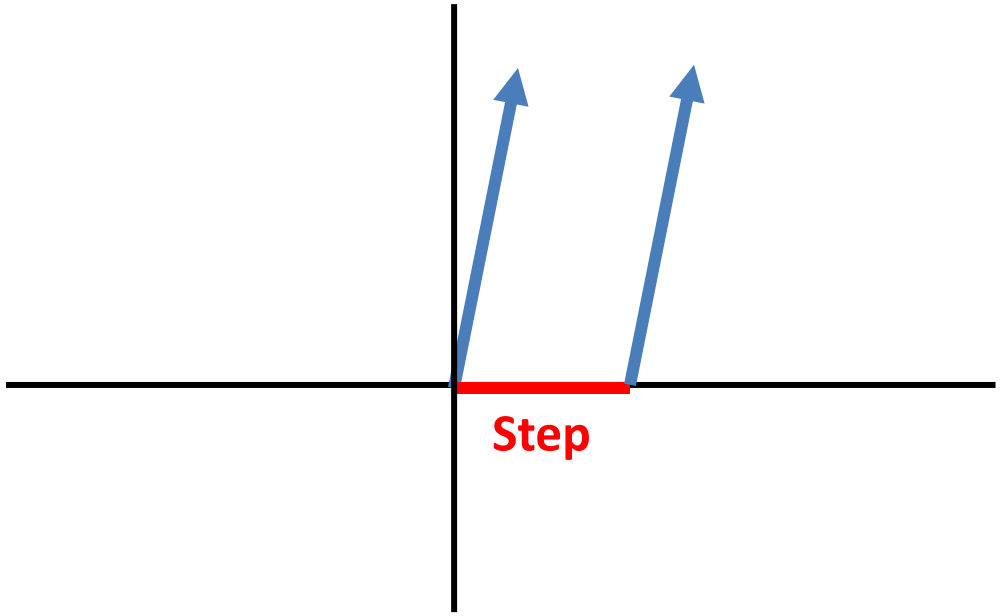
2D Divergence $\nabla \cdot$

$$\nabla \cdot (\mathbf{V}(x,y)) = \frac{\partial \, \mathbf{V}_x(x,y)}{\partial x} + \frac{\partial \, \mathbf{V}_y(x,y)}{\partial y}$$

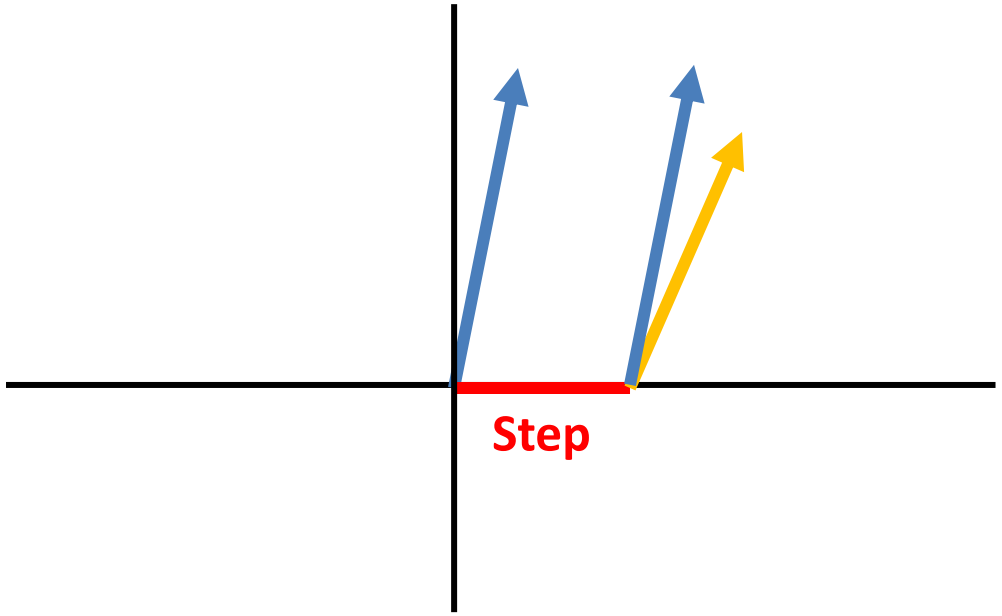
Geometric Interpretation of Divergence



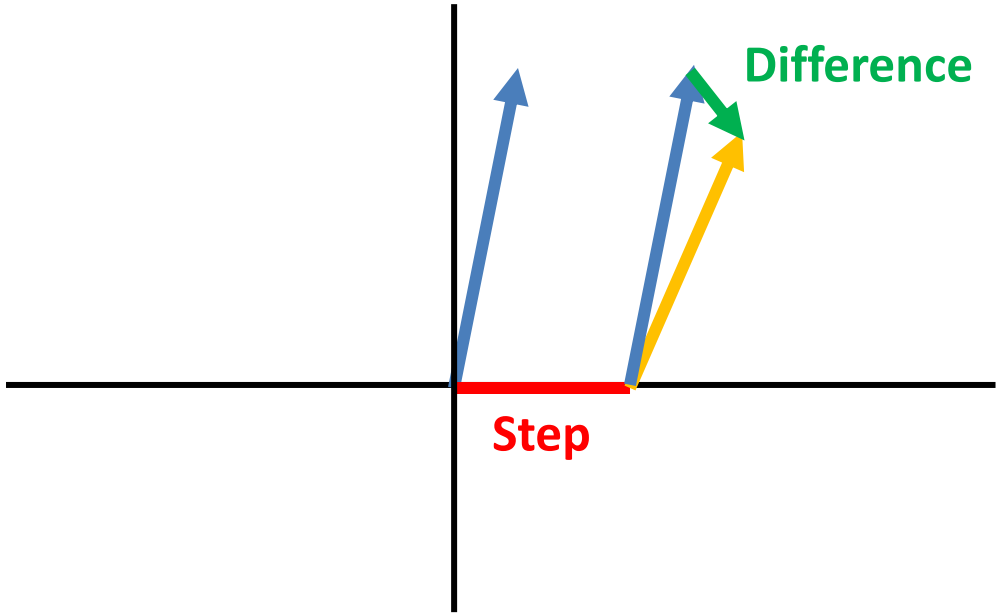
Geometric Interpretation of Divergence



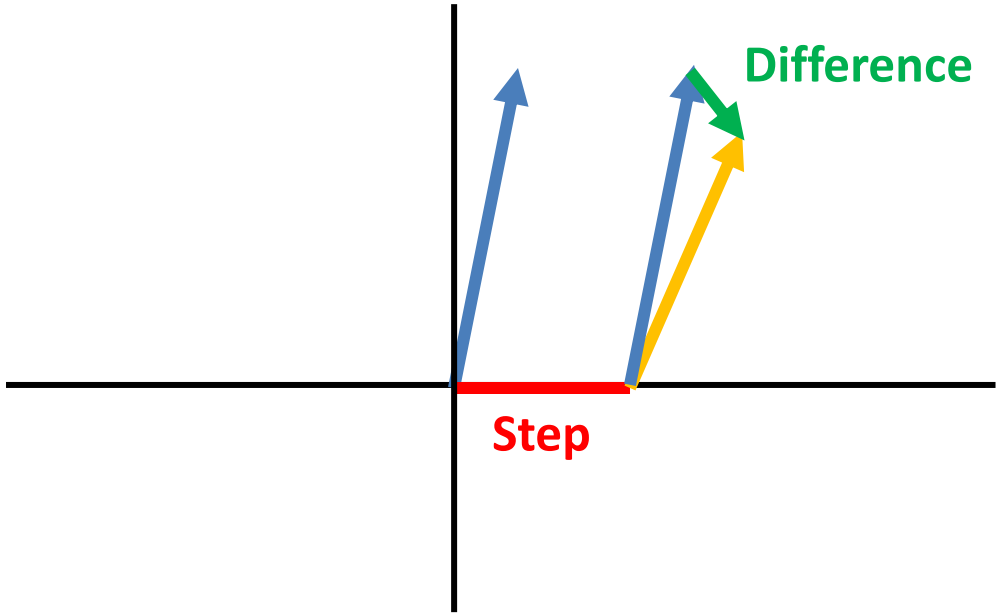
Geometric Interpretation of Divergence



Geometric Interpretation of Divergence

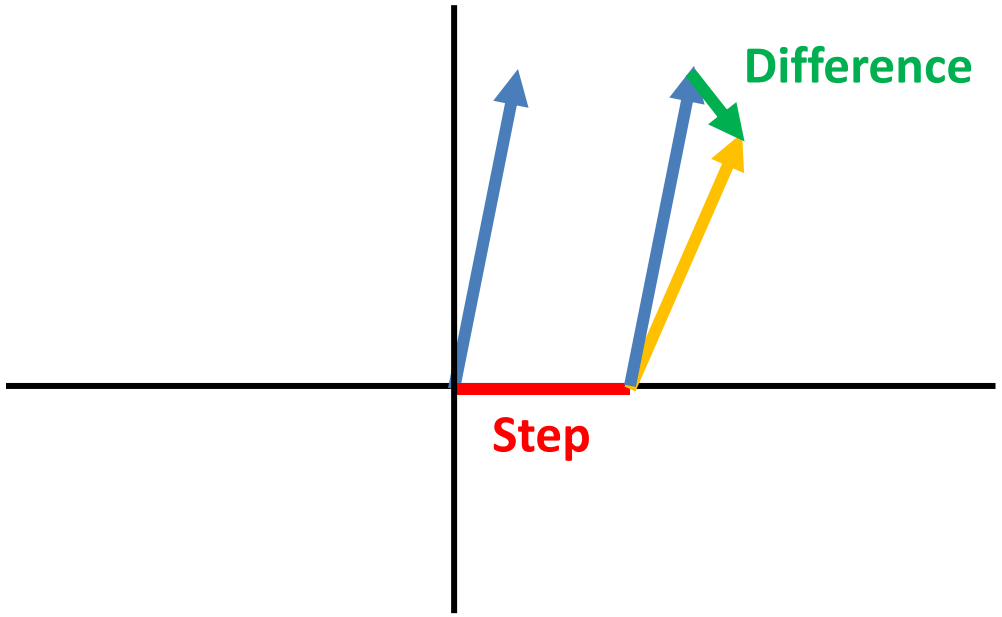


Geometric Interpretation of Divergence



$$\textbf{Step} \cdot \textbf{Difference}$$

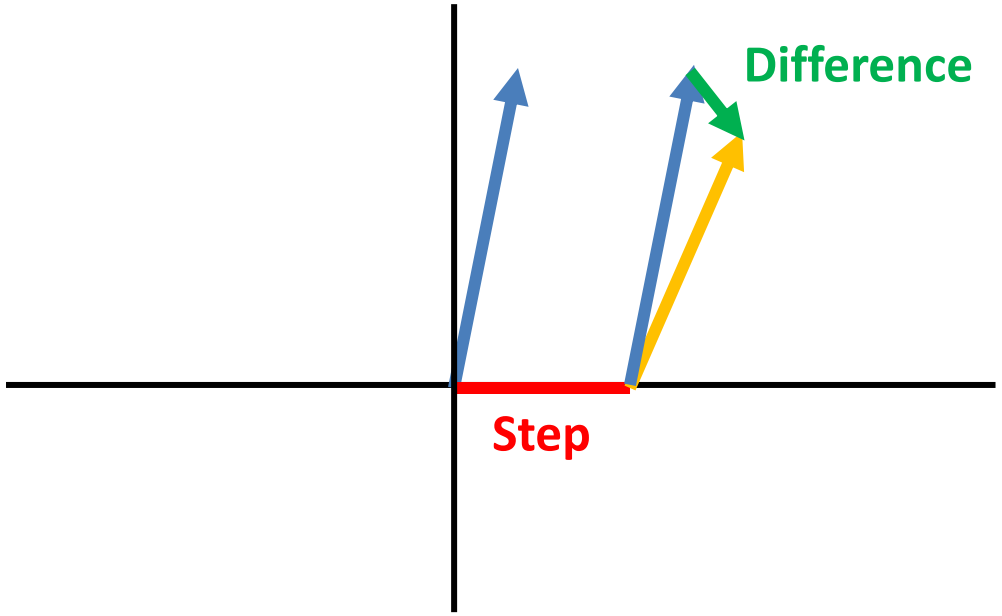
Geometric Interpretation of Divergence



Example: $(1, 0) \cdot (0.2, -0.2) = 1 \cdot 0.2 + 0 \cdot (-0.2) = 0.2$

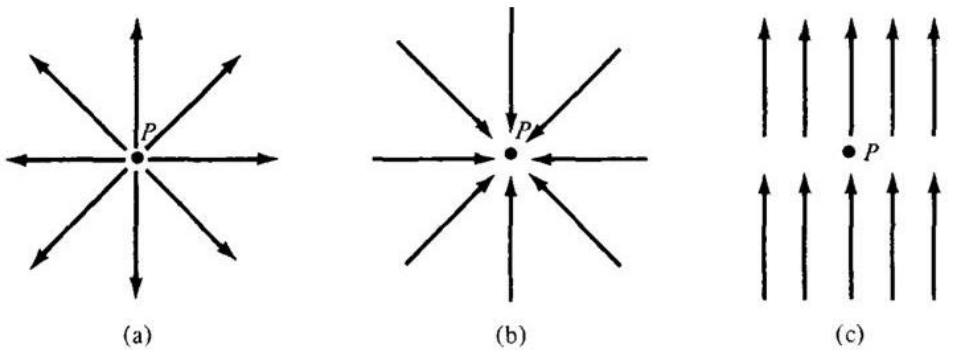
Step $\cdot$ **Difference**

Geometric Interpretation of Divergence



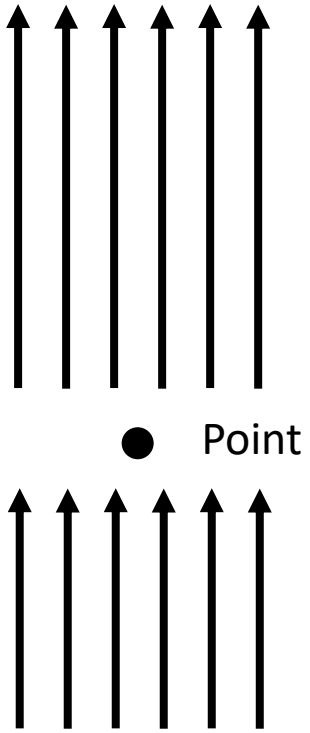
$$\frac{\partial \mathbf{V}_x(x,y)}{\partial x} + \frac{\partial \mathbf{V}_y(x,y)}{\partial y} \xrightarrow{\text{Averaged}} \textbf{Step} \cdot \textbf{Difference}$$

Divergence $\nabla \cdot$



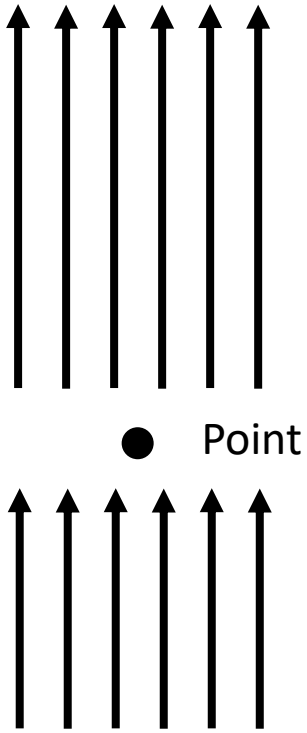
(a) Positive divergence, (b) negative divergence, (c) zero divergence.

Example Case



Example Case

Positive Divergence



Ficks 2nd Law

- $\frac{\partial c}{\partial t} = -\frac{\partial J}{\partial x}$

Ficks 2nd Law

- $\frac{\partial c}{\partial t} = -\frac{\partial J}{\partial x} = \frac{\partial\left(D\frac{\partial c}{\partial x}\right)}{\partial x}$

Ficks 2nd Law

- $\frac{\partial c}{\partial t} = -\frac{\partial J}{\partial x} = \frac{\partial\left(D\frac{\partial c}{\partial x}\right)}{\partial x} = D\frac{\partial^2 c}{\partial x^2}$

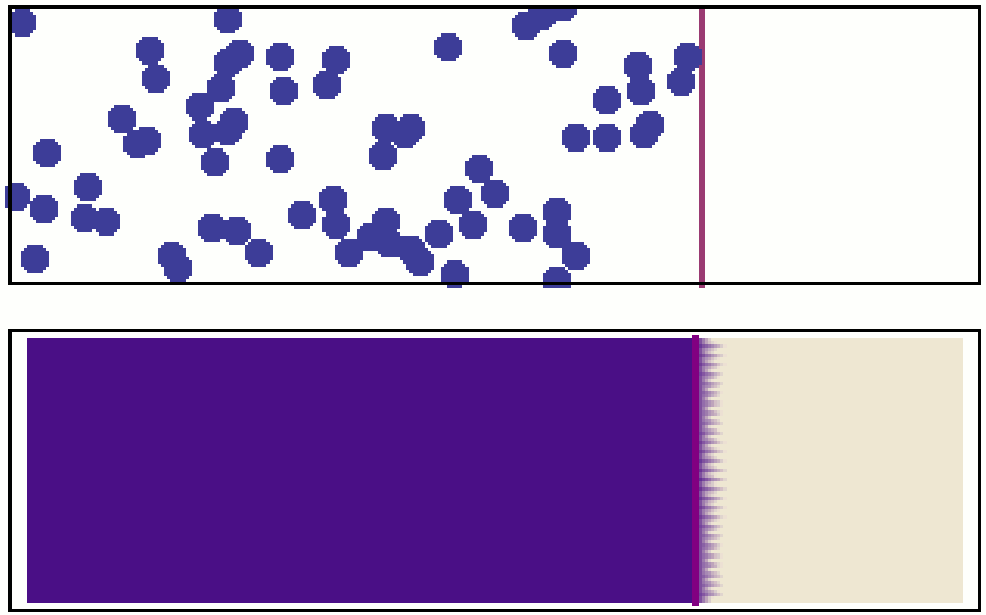
Ficks 2nd Law > 1D

- $\frac{\partial c}{\partial t} = -\frac{\partial J}{\partial x} = \frac{\partial\left(D\frac{\partial c}{\partial x}\right)}{\partial x} = D\frac{\partial^2 c}{\partial x^2} = D(\nabla \cdot \nabla c)$

Ficks 2nd Law > 1D

- $\frac{\partial c}{\partial t} = -\frac{\partial J}{\partial x} = \frac{\partial\left(D\frac{\partial c}{\partial x}\right)}{\partial x} = D\frac{\partial^2 c}{\partial x^2} = D(\nabla \cdot \nabla c) = D\Delta c$

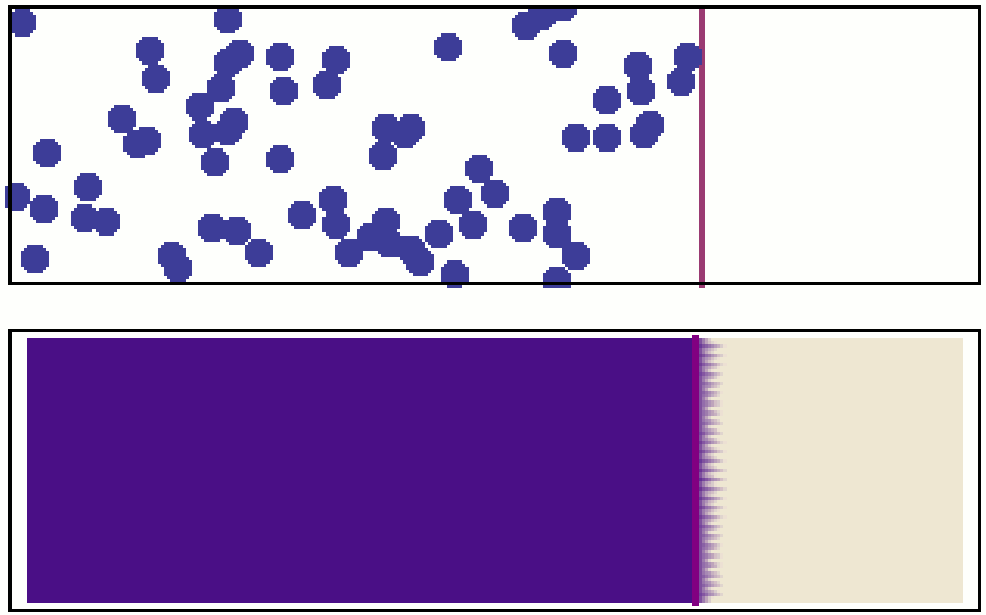
Ficks 2nd Law: $\frac{\partial c(x,t)}{\partial t} = D\Delta c$



Discrete model

Continuous model

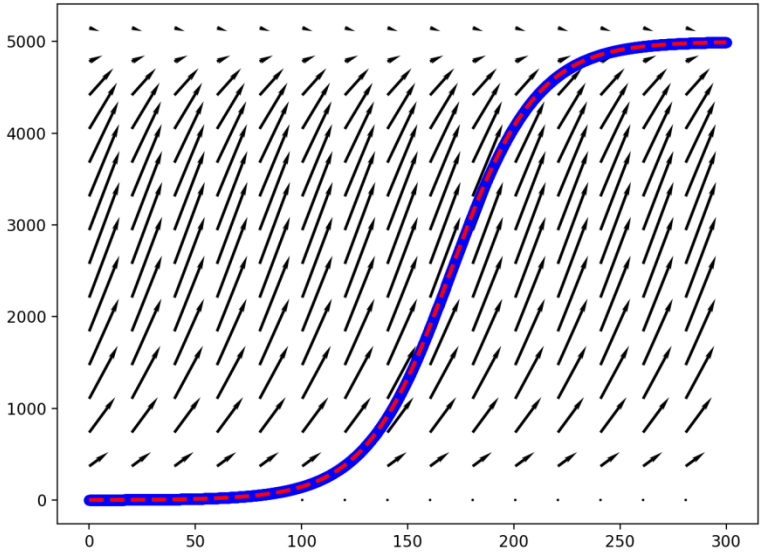
$c(x,t)$ depends on both space and time



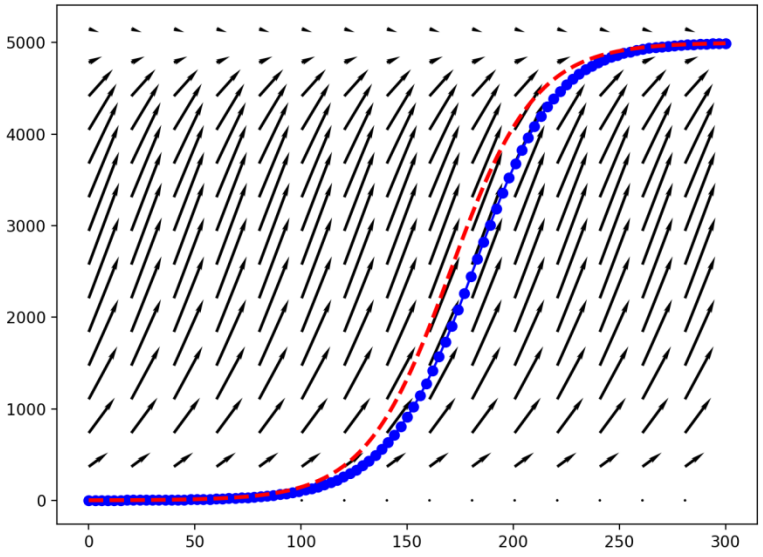
Discrete model

Continuous model

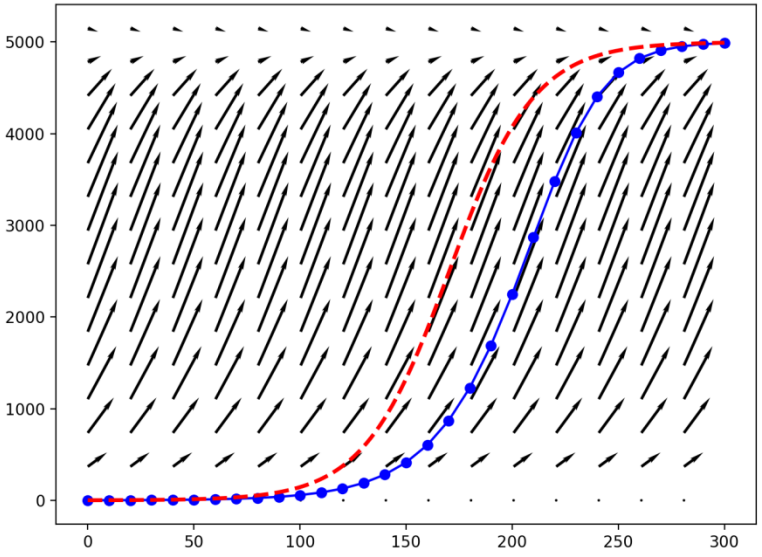
Discretization of Time



dt = 0.1



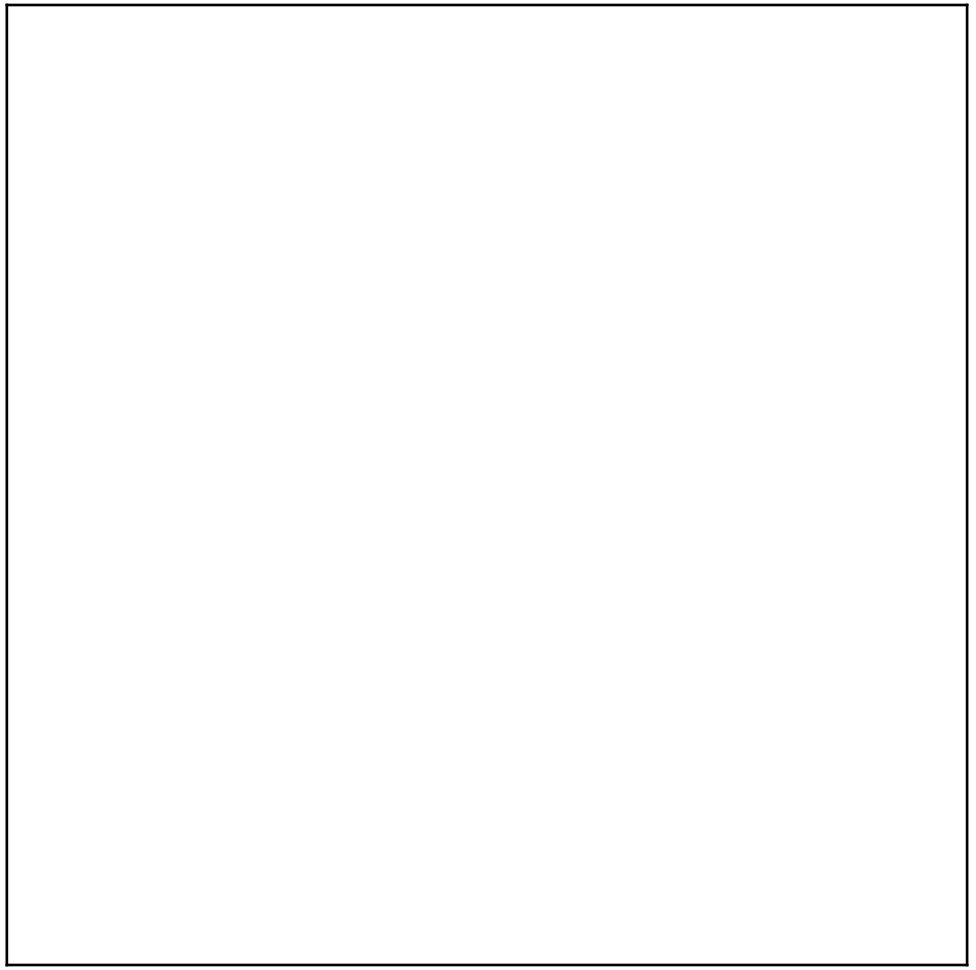
dt = 3



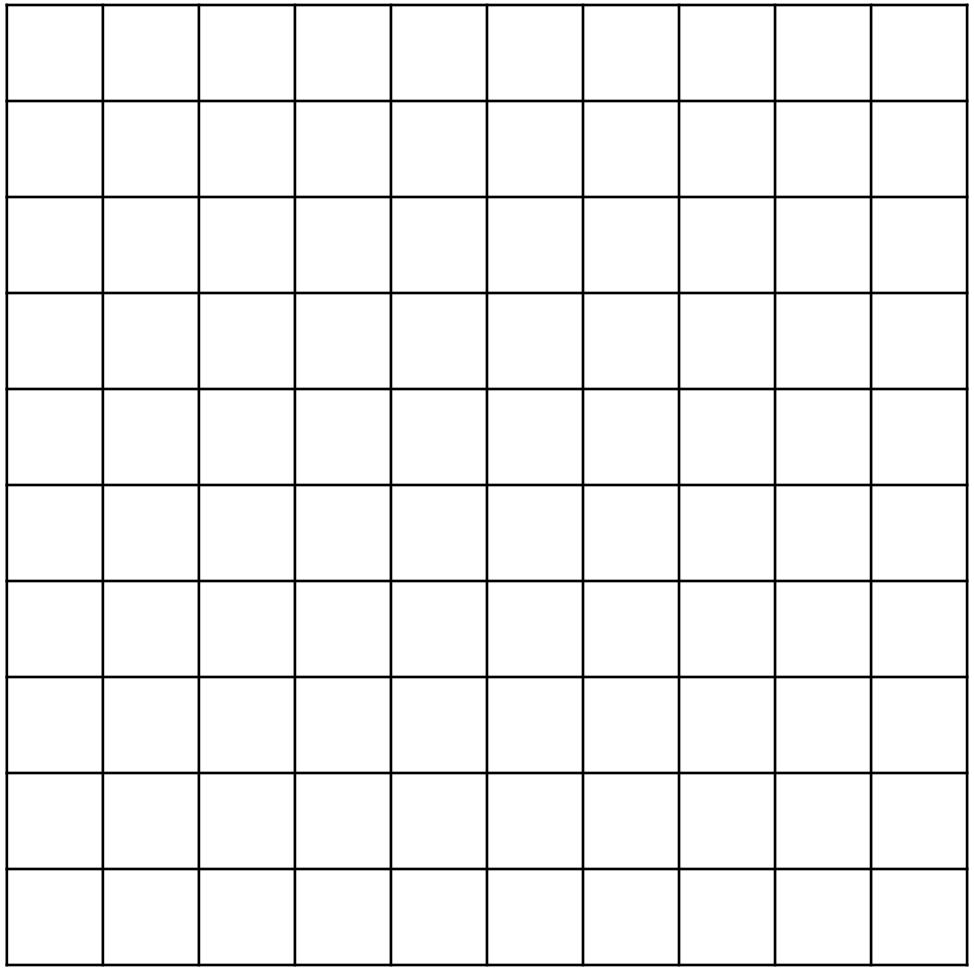
dt = 10

Follow the vector field: `x[n+1] = x[n] + dt*f(x[n], t[n])`

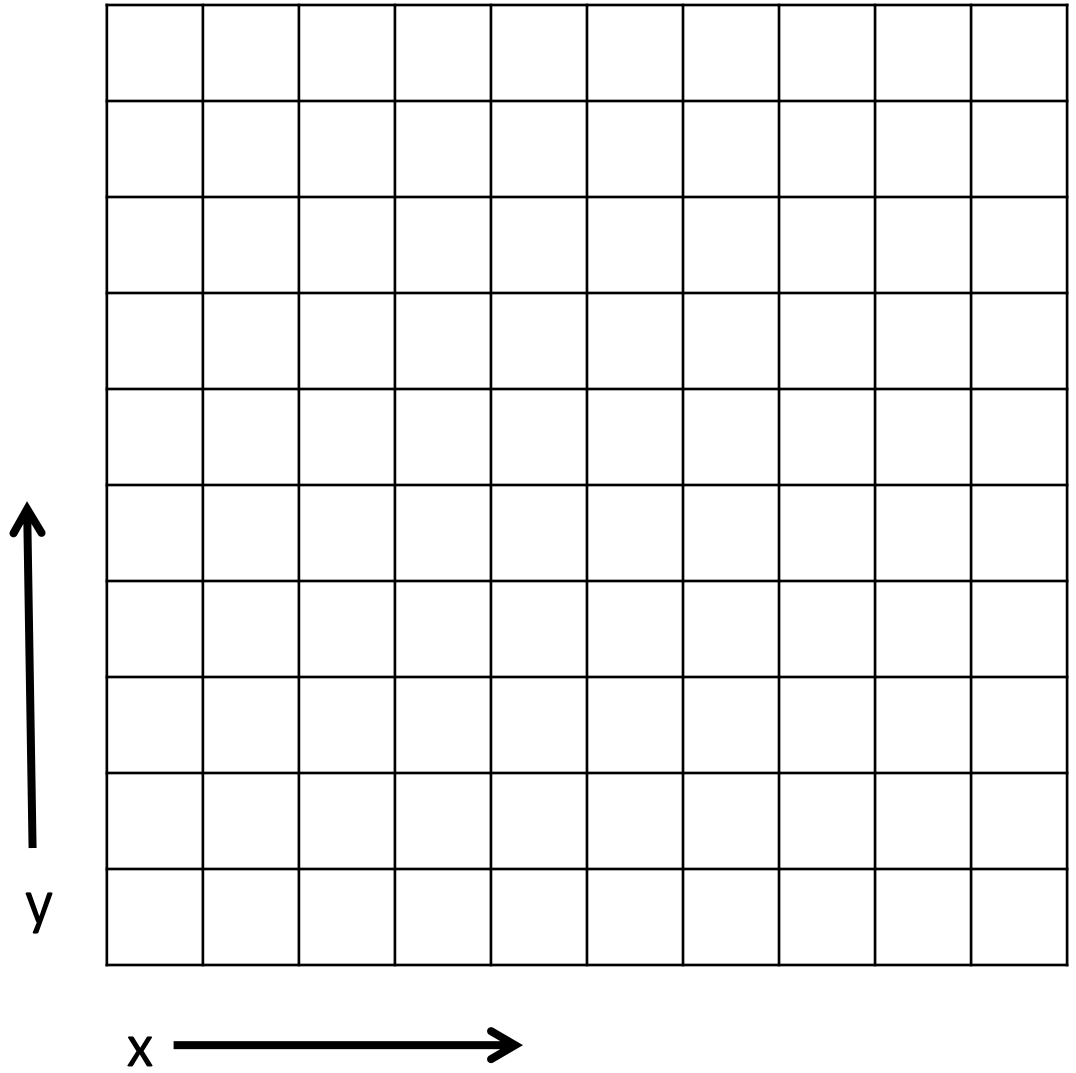
Discretization of 2D Space



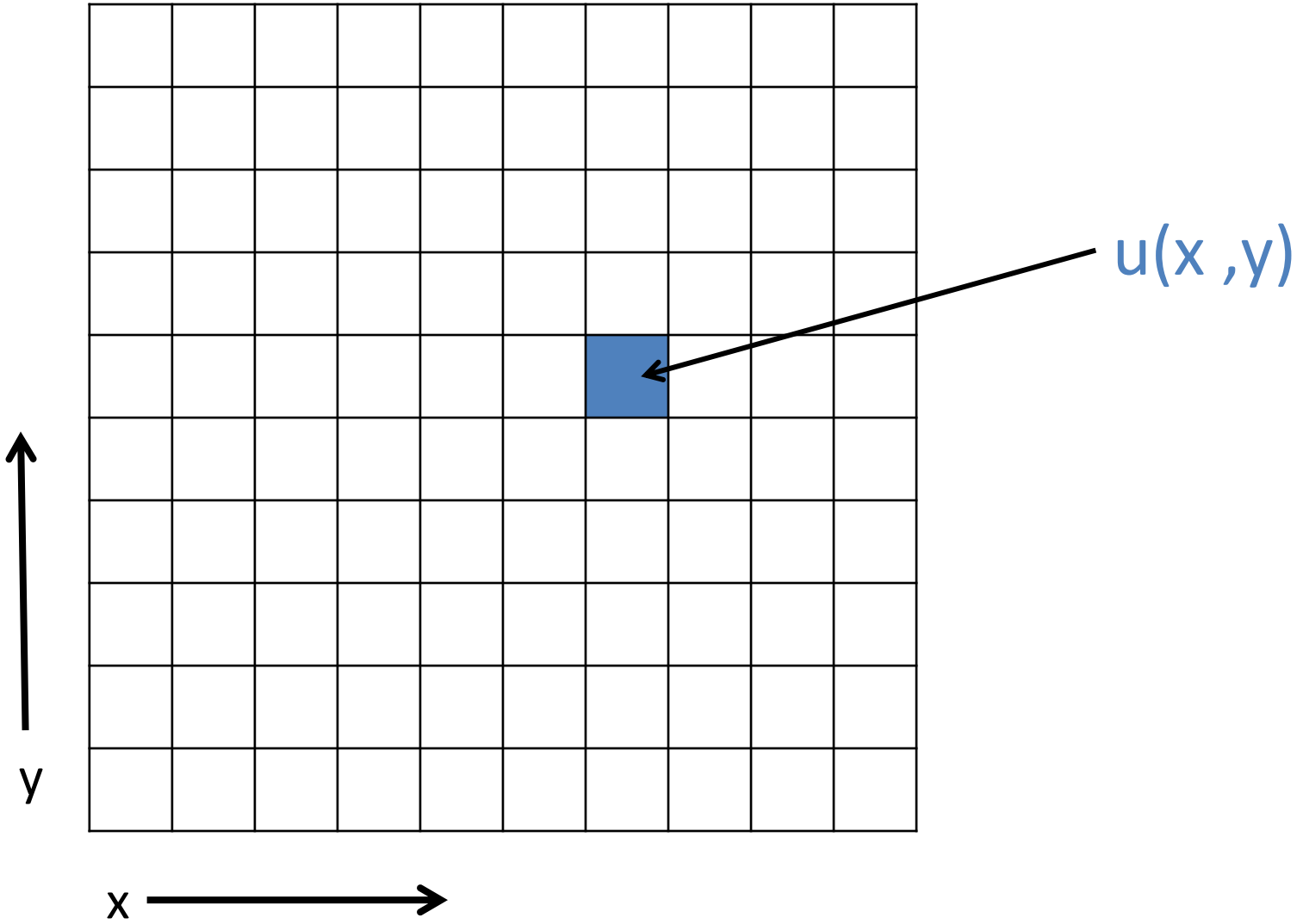
Discretization of 2D Space



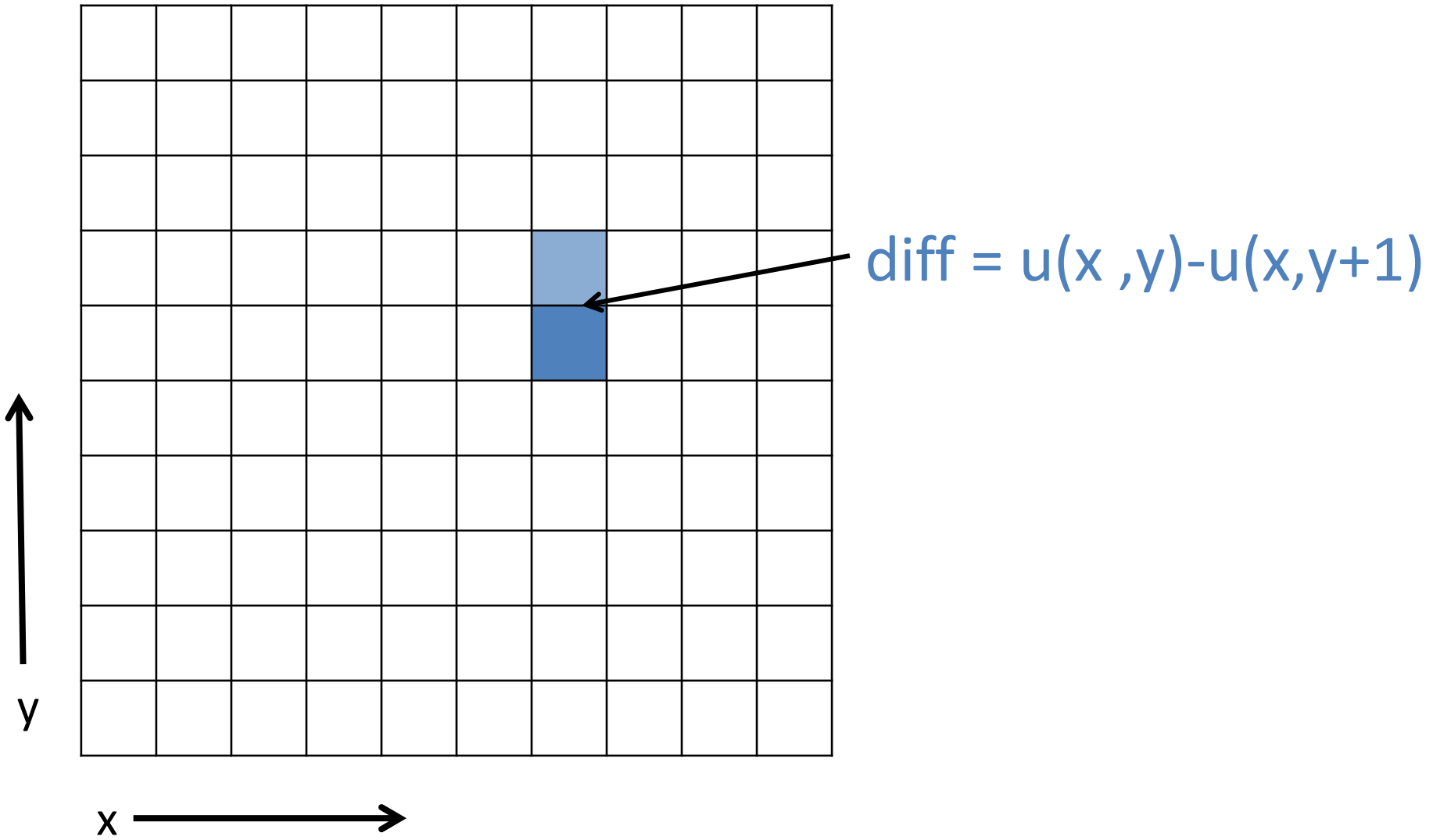
Discretization of 2D Space



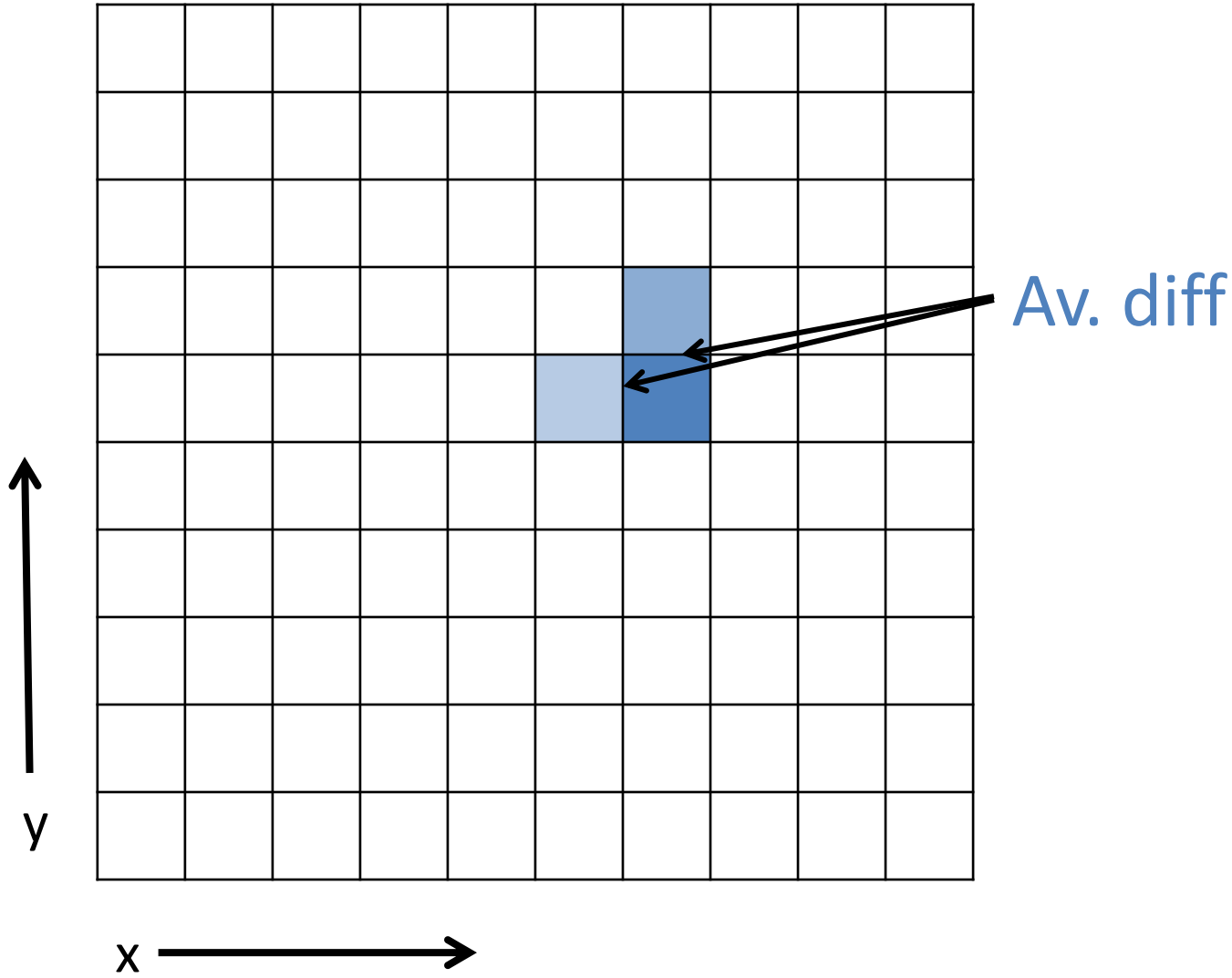
Discretization of 2D Space



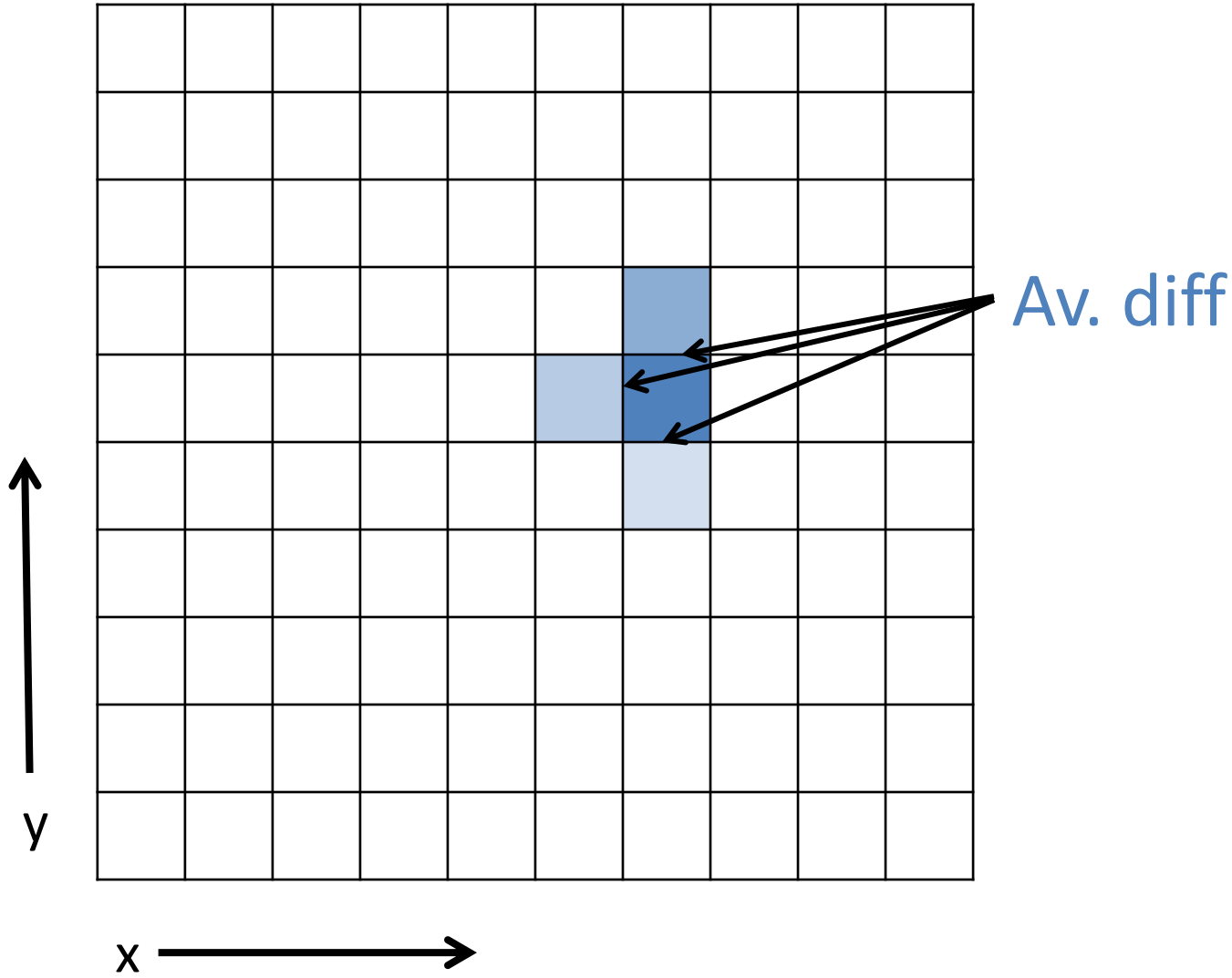
Discretization of 2D Space



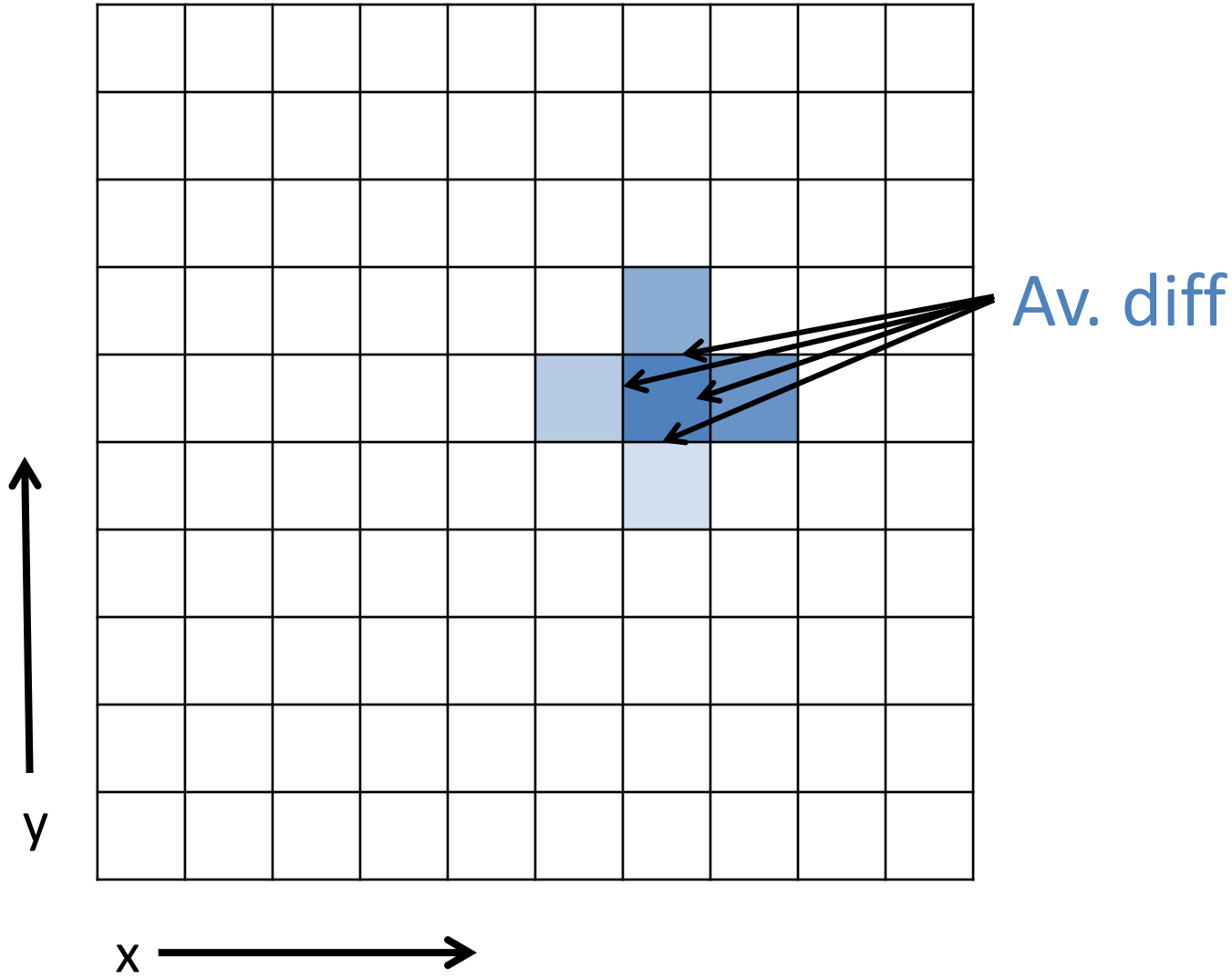
Discretization of 2D Space



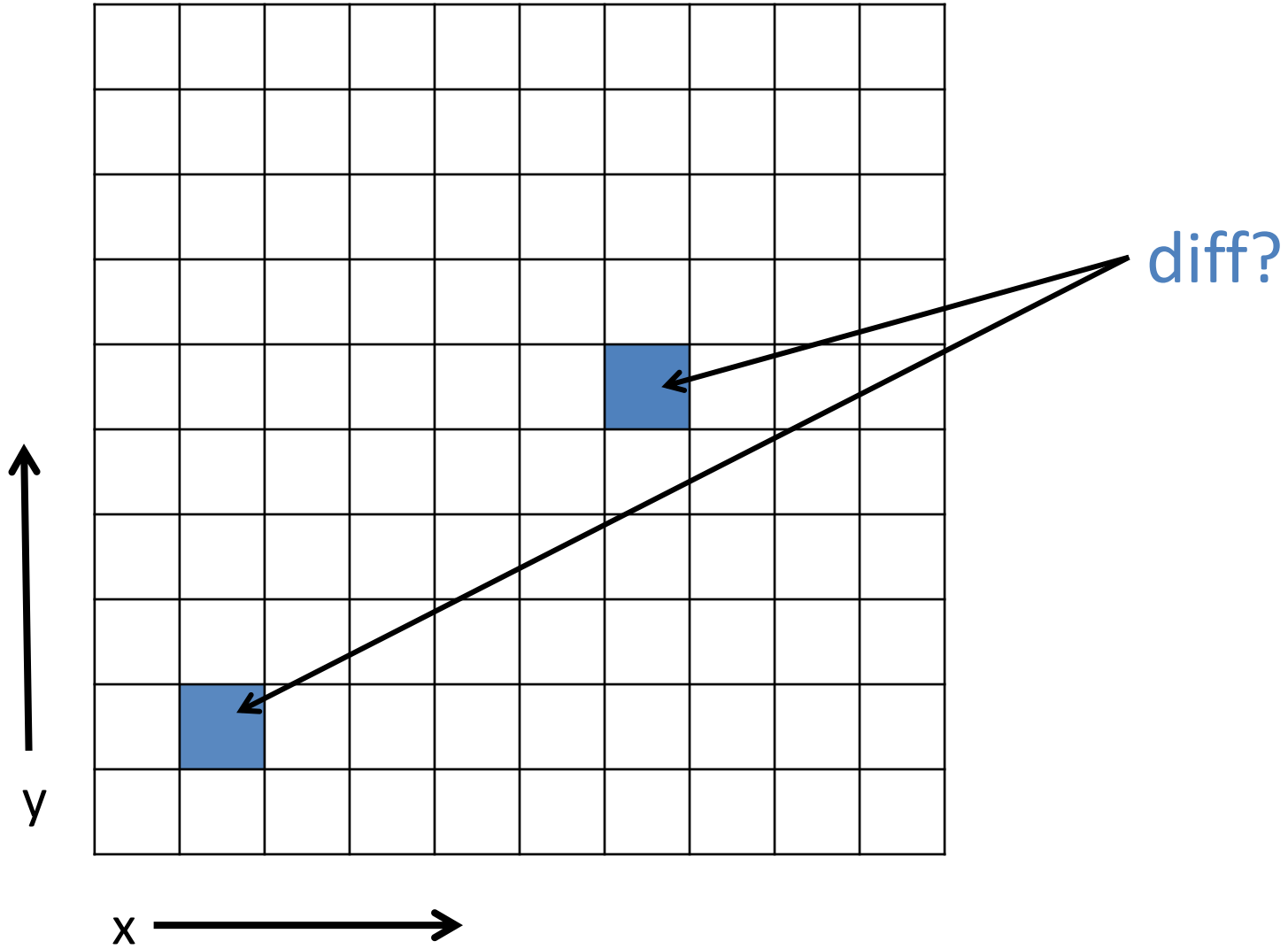
Discretization of 2D Space



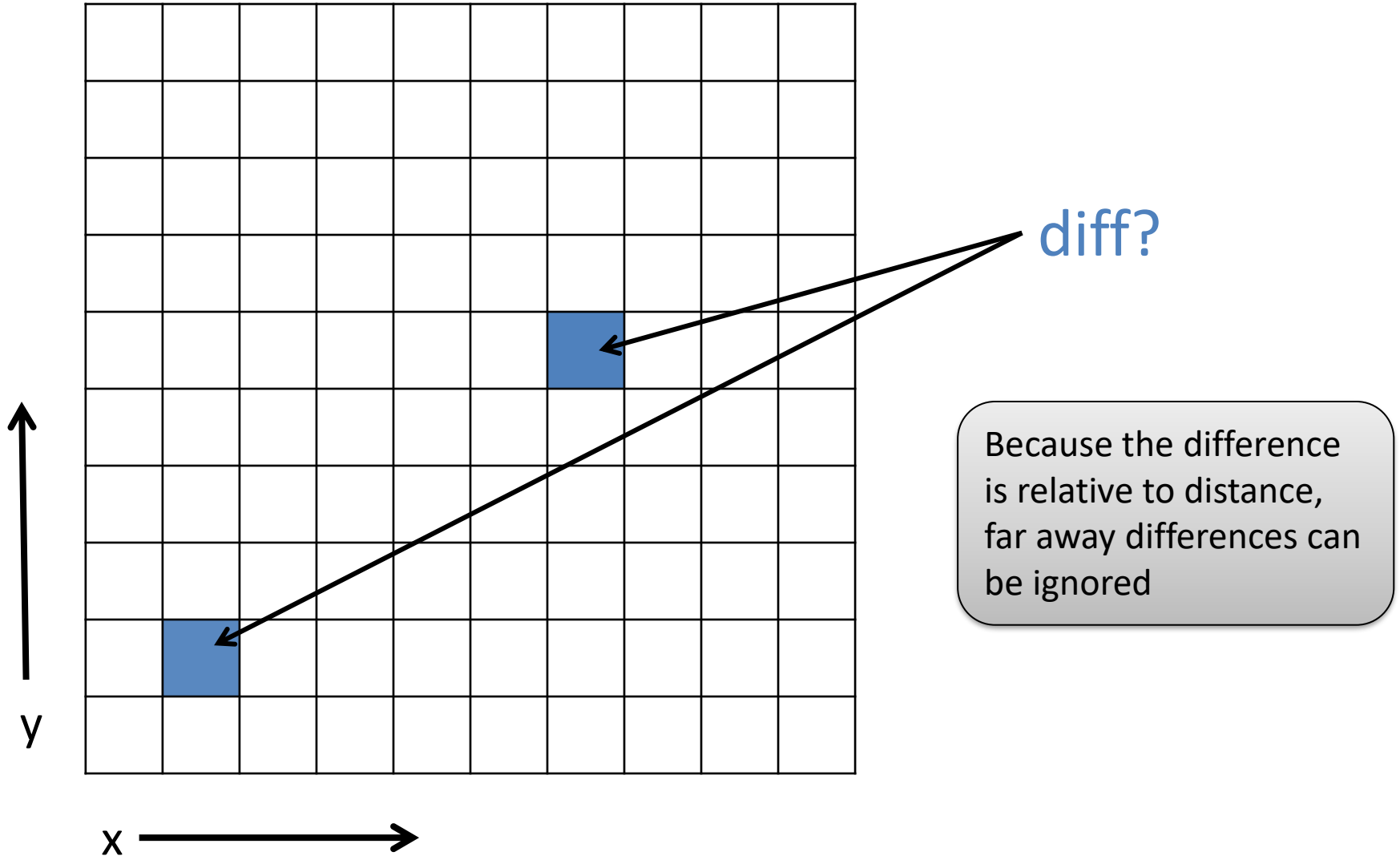
Discretization of 2D Space



Discretization of 2D Space



Discretization of 2D Space



Because the difference is relative to distance, far away differences can be ignored

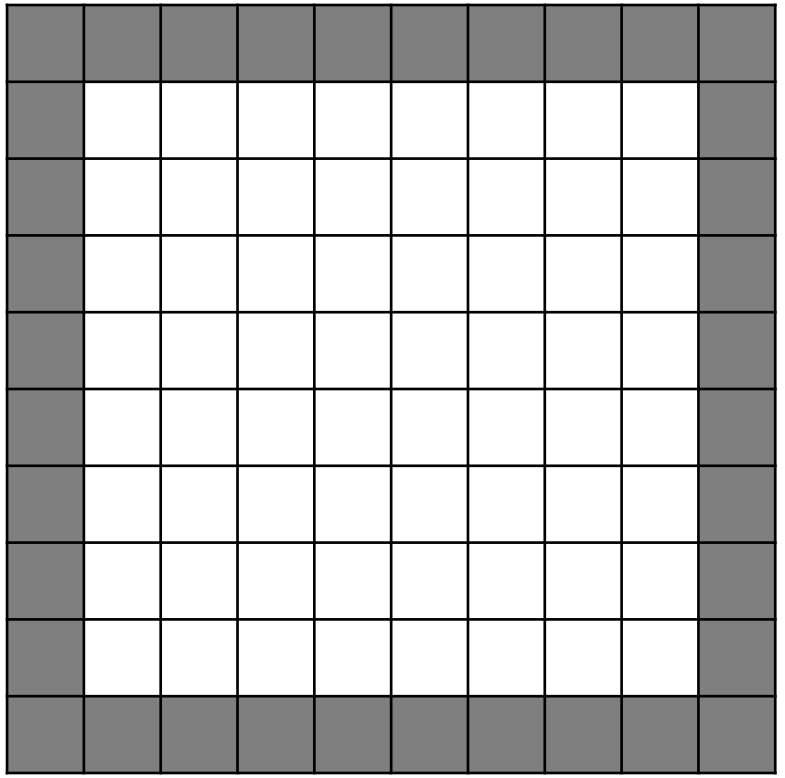
Discrete Laplace Operator

$$\Delta u(x,y) \simeq \frac{u(x+h,y)+u(x-h,y)+u(x,y+h)+u(x,y-h)-4u(x,y)}{dx^2}$$

Discrete Space represented by Array

- Define array of size of discrete grid + 2 (for padding)

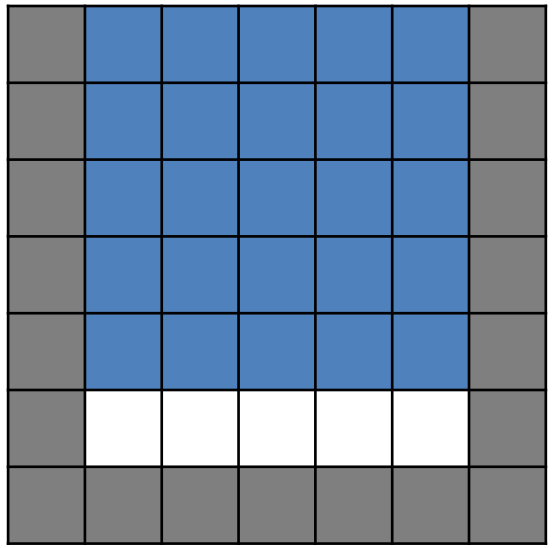
```
U = np.random.rand(size, size)
```



Discrete Laplacian in Python

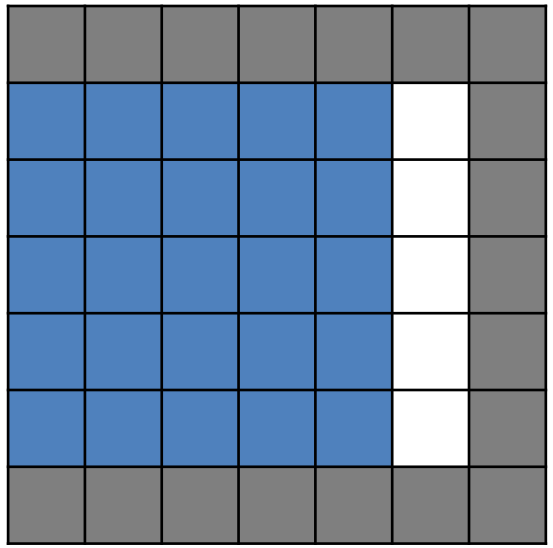
```
# dx length of grid cell side

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
```

Discrete Laplacian in Python



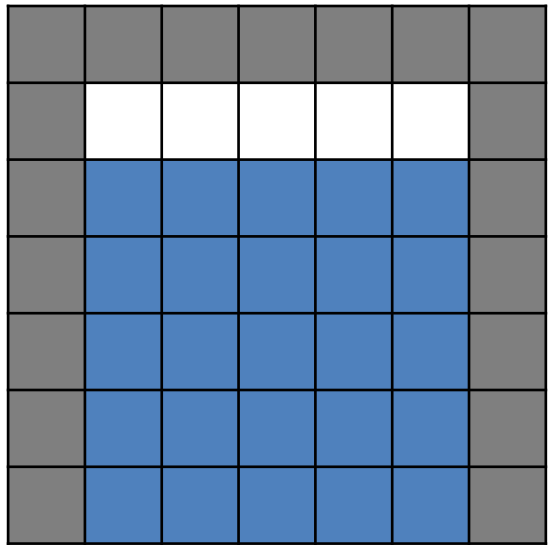
```
# dx length of grid cell side

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
```

Discrete Laplacian in Python



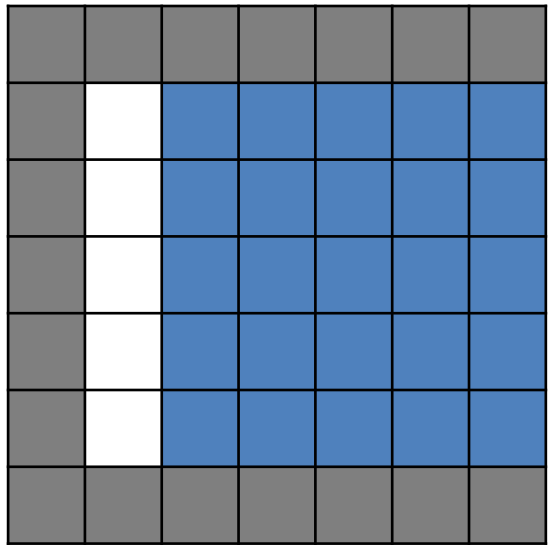
```
# dx length of grid cell side

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
```

Discrete Laplacian in Python



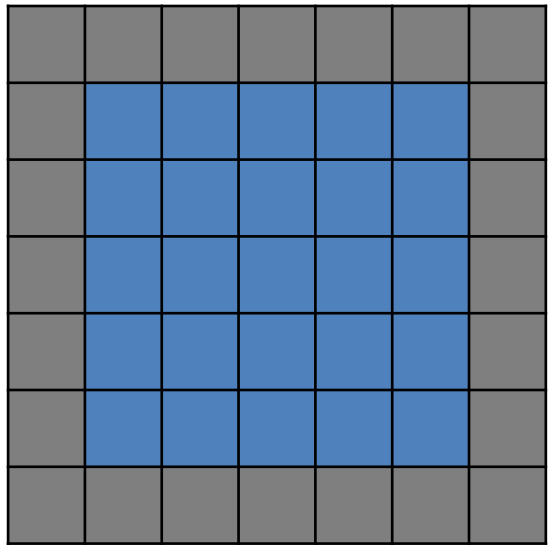
```
# dx length of grid cell side

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
```

Discrete Laplacian in Python



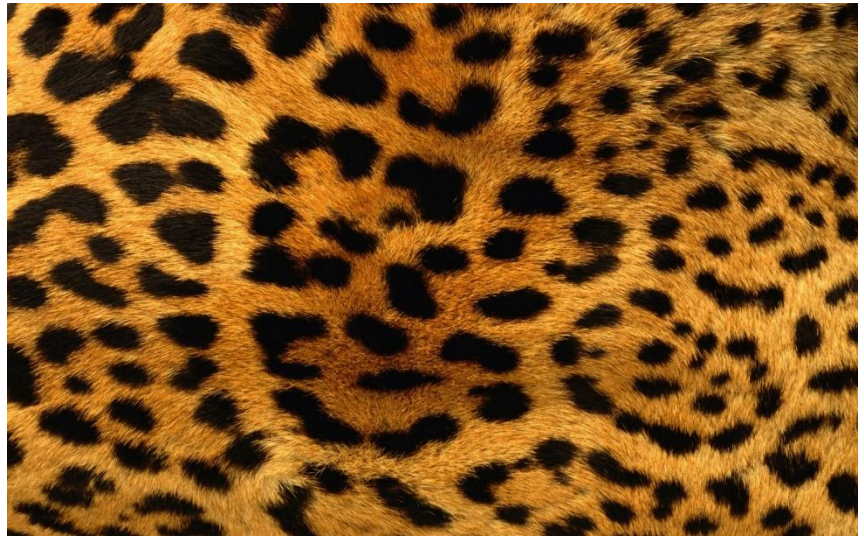
```
# dx length of grid cell side

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
```

Discrete Laplacian in Python



```
# dx length of grid cell side

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
```

Modeling of pigmentation patterns with DE

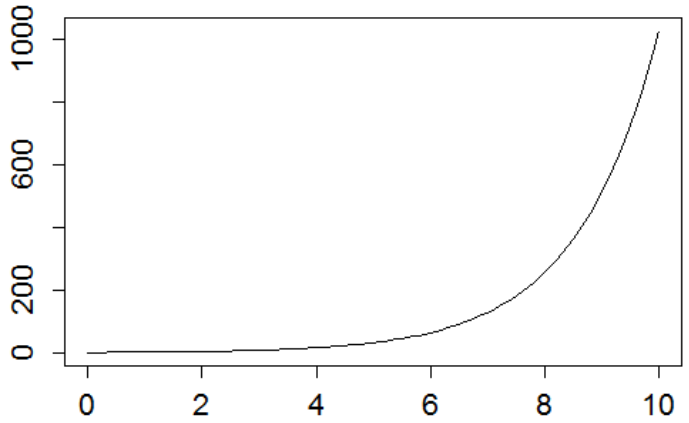


Reaction-Diffusion Equations

$$\frac{\partial c}{\partial t} = \underbrace{\mathrm{D}\frac{\partial^2 c}{\partial x^2}}_{\text{Diffusion}} + \underbrace{f(c)}_{\text{Reaction}}$$

Reaction in 0D Space

- E.g. Population growth model
- $\frac{dc}{dt} = Kc$
- $\mathrm{c}(t) = C_0 e^{Kt}, C_0 = c(0)$



Activator-Inhibitor model

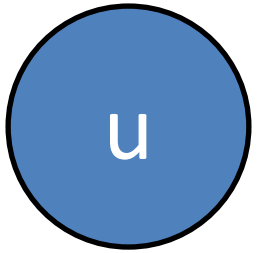
- $\frac{\partial u}{\partial t} = u^3 - v - k + D_u \frac{\partial^2 u}{\partial x^2}$

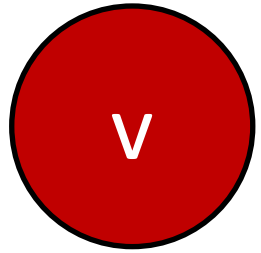
- $\frac{\partial v}{\partial t} = a(u - v + D_v \frac{\partial^2 v}{\partial x^2})$

Activator-Inhibitor model

- $\frac{\partial u}{\partial t} = u^3 - v - k + D_u \frac{\partial^2 u}{\partial x^2}$

- $\frac{\partial v}{\partial t} = a(u - v + D_v \frac{\partial^2 v}{\partial x^2})$

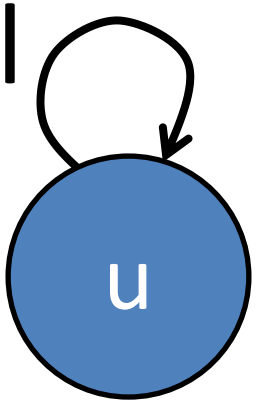


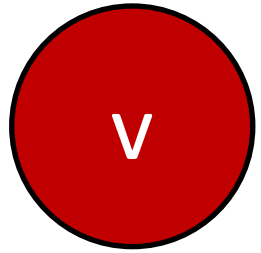


Activator-Inhibitor model

- $\frac{\partial u}{\partial t} = u^3 - v - k + D_u \frac{\partial^2 u}{\partial x^2}$

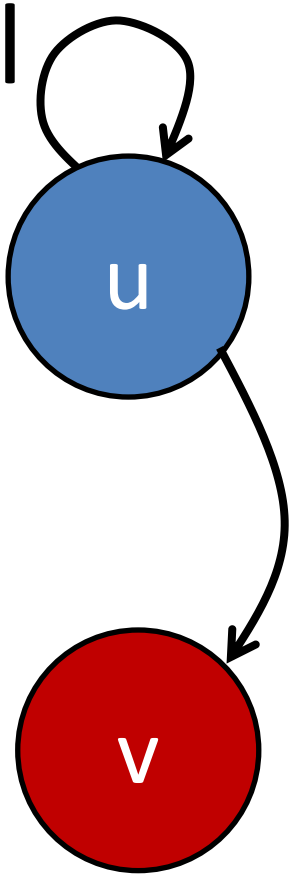
- $\frac{\partial v}{\partial t} = a(u - v + D_v \frac{\partial^2 v}{\partial x^2})$





Activator-Inhibitor model

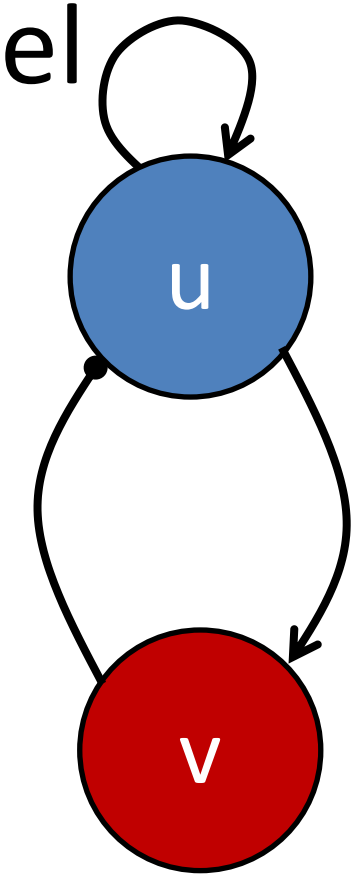
- $\frac{\partial u}{\partial t} = u^3 - v - k + D_u \frac{\partial^2 u}{\partial x^2}$
- $\frac{\partial v}{\partial t} = a(u - v + D_v \frac{\partial^2 v}{\partial x^2})$



Activator-Inhibitor model

- $\frac{\partial u}{\partial t} = u^3 - v - k + D_u \frac{\partial^2 u}{\partial x^2}$

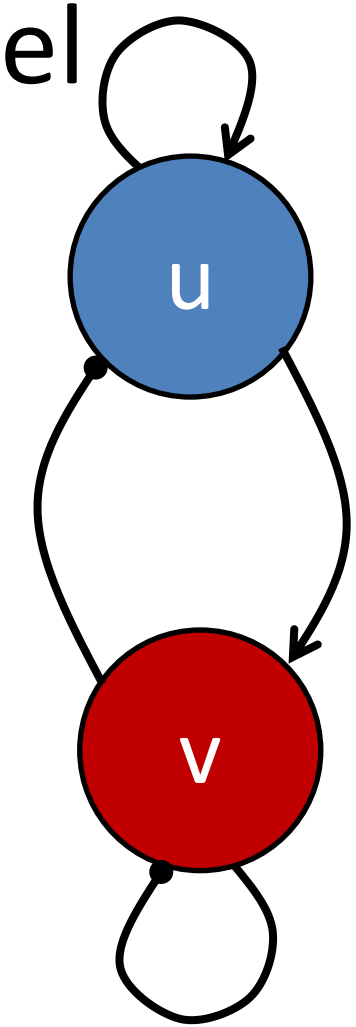
- $\frac{\partial v}{\partial t} = a(u - v + D_v \frac{\partial^2 v}{\partial x^2})$



Activator-Inhibitor model

- $\frac{\partial u}{\partial t} = u^3 - v - k + D_u \frac{\partial^2 u}{\partial x^2}$

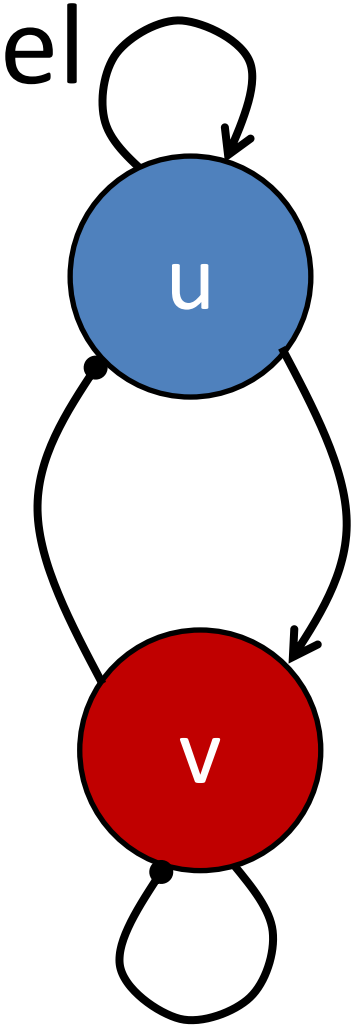
- $\frac{\partial v}{\partial t} = a(u - v + D_v \frac{\partial^2 v}{\partial x^2})$



Activator-Inhibitor model

- $\frac{\partial u}{\partial t} = u^3 - v - k + D_u \frac{\partial^2 u}{\partial x^2}$

- $\frac{\partial v}{\partial t} = a(u - v + D_v \frac{\partial^2 v}{\partial x^2})$



Constant Synthesis /Degradation

Python Implementation

```
deltaU = laplacian(U)
deltaV = laplacian(V)

Uc = U[1:-1, 1:-1]
Vc = V[1:-1, 1:-1]

U[1:-1, 1:-1], V[1:-1, 1:-1] = \
    Uc + dt * (Du * deltaU + Uc - Uc**3 - Vc + k),\
    Vc + dt * (Dv * deltaV + Uc - Vc) * a
```

Python Implementation

Loop over number of simulation steps

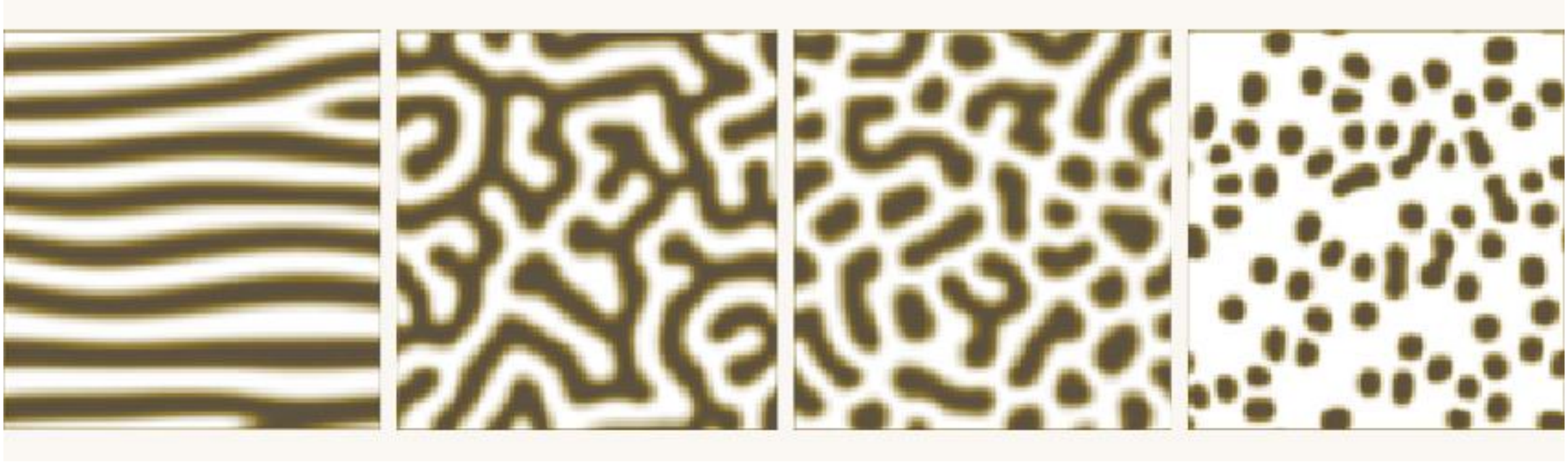
```
deltaU = laplacian(U)
deltaV = laplacian(V)

Uc = U[1:-1, 1:-1]
Vc = V[1:-1, 1:-1]

U[1:-1, 1:-1], V[1:-1, 1:-1] = \
    Uc + dt * (Du * deltaU + Uc - Uc**3 - Vc + k),\
    Vc + dt * (Dv * deltaV + Uc - Vc) * a
```

Fill Padding Values

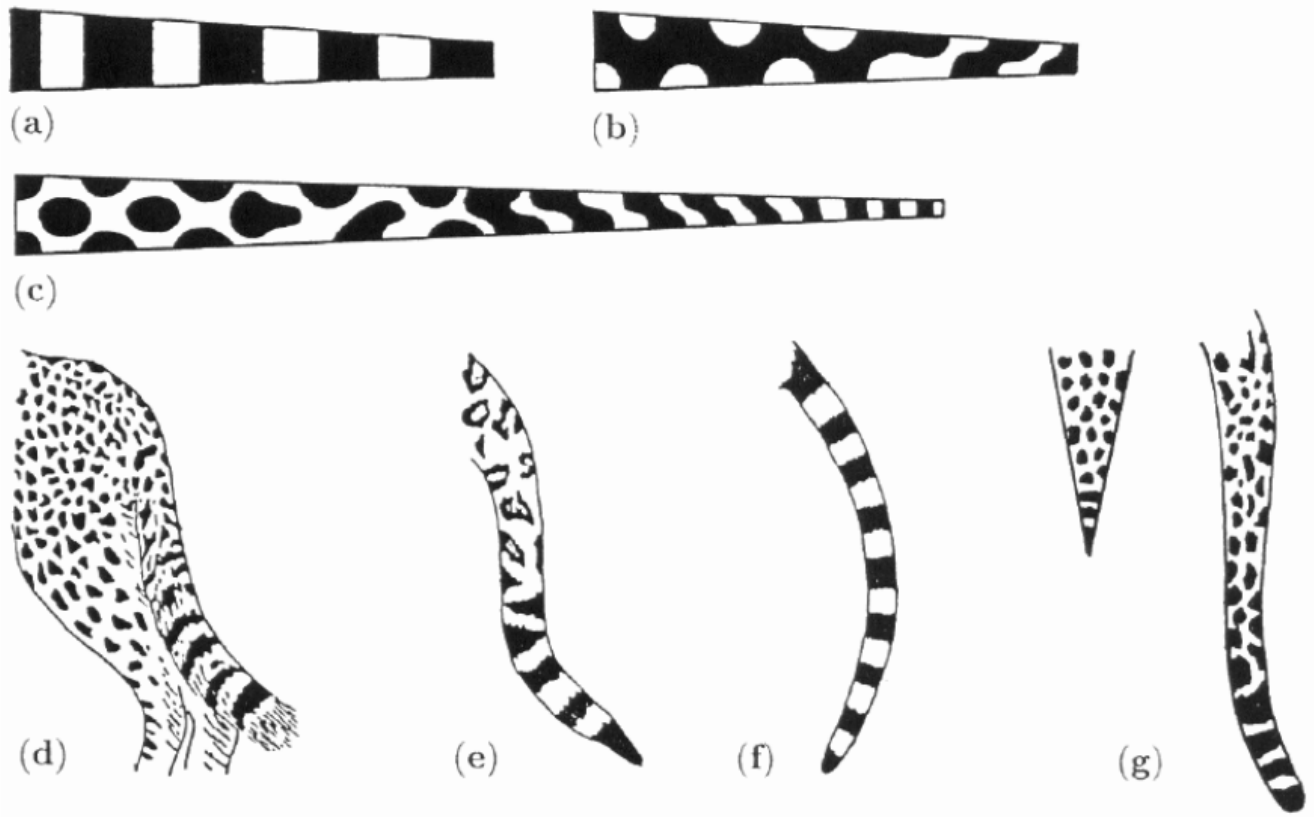
Loop over number of simulation steps

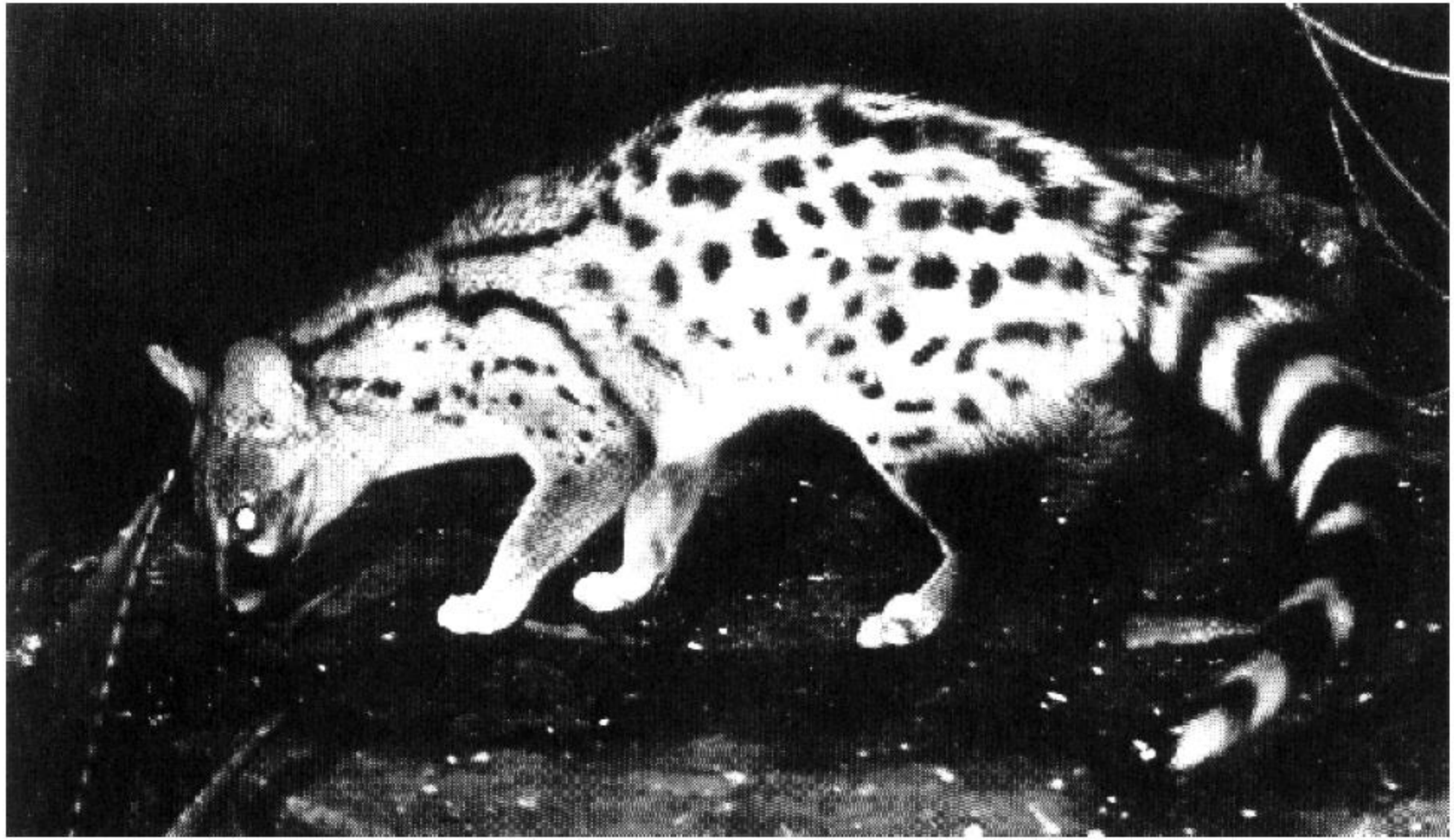
```
for Z in (U, V):
        Z[0, :] = Z[1, :]
        Z[-1, :] = Z[-2, :]
        Z[:, 0] = Z[:, 1]
        Z[:, -1] = Z[:, -2]
```

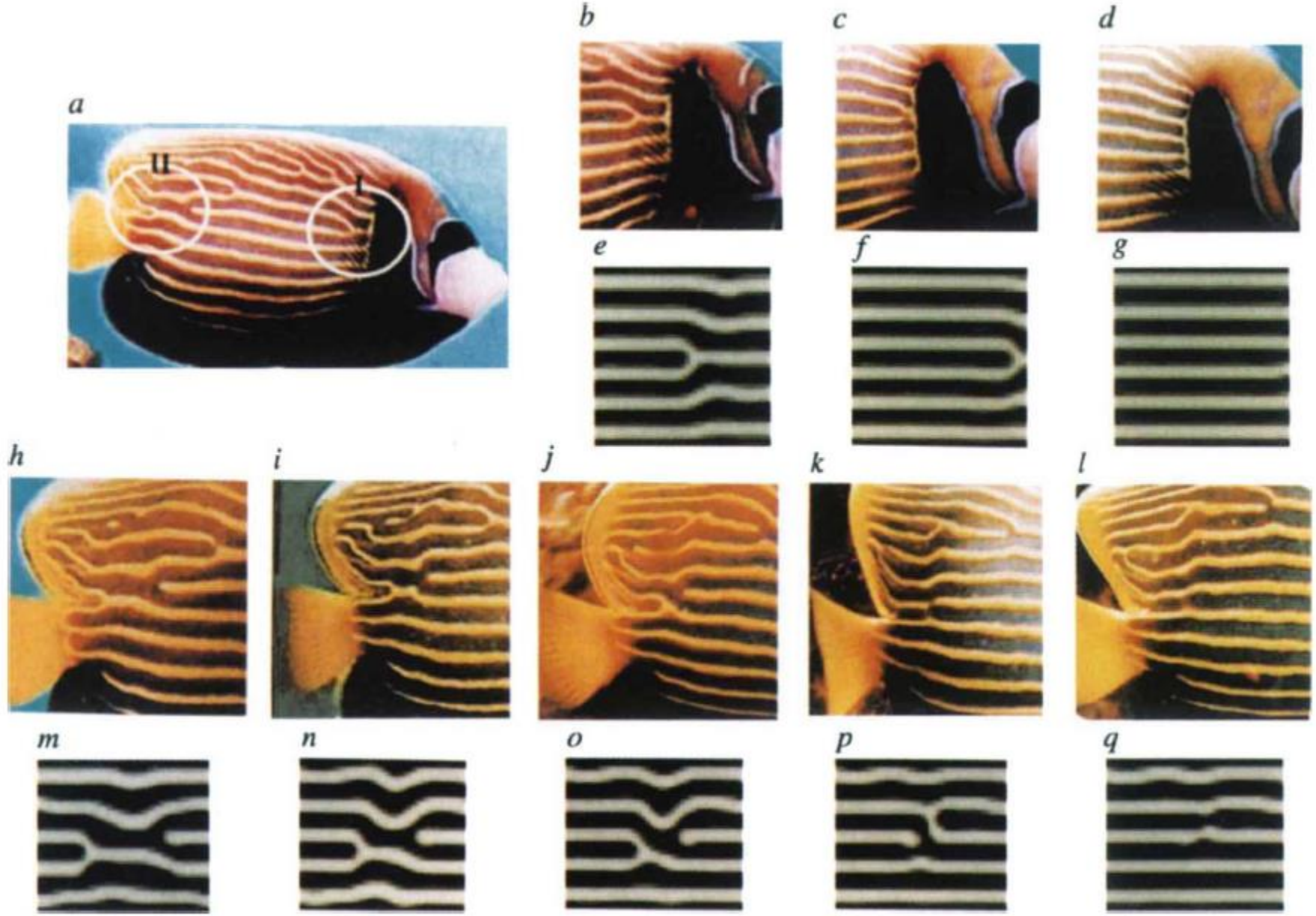
Parameter Space Exploration



Animal tails







Kondo S. et al, Nature 1995

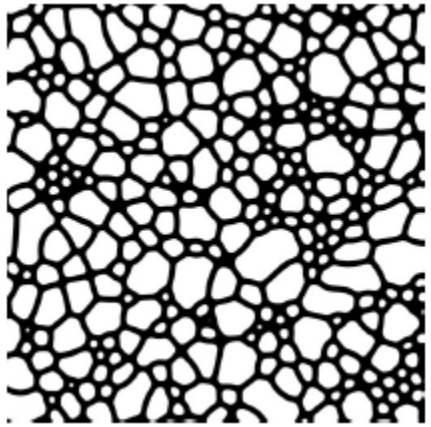
Reaction Diffusion in 3D

Gray-Scott model

- $\frac{\partial u}{\partial t} = -uv^2 + f(1-u) + D_u \frac{\partial^2 u}{\partial x^2}$

- $\frac{\partial v}{\partial t} = uv^2 - (k+f)v + D_v \frac{\partial^2 v}{\partial x^2}$






Exercise

- Implement the activator inhibitor model in python. Use a uniform grid size of 100, dx = 0.02, dt = 0.001, T = 15, a = 6, Dv = 5e-3. Visualize using matplotlib all a selection of patterns of stripes and spots. Visualize in an image plot a few consecutive time steps for a selected pattern. Parameters to change are k and Du, try ranges from -0.5 to 0.15 for k and 3e-5 to 1.4e-4 for Du. Report the settings for parameter values that give you spot, stripe, gap and noise patterns.