Population Dynamics

Malthusian population growth model

$$\frac{dN}{dt} = bN - dN = rN$$

where b and d are the birth and death rates respectively.

Logistic growth model

$$\frac{dN}{dt} = r\left(1 - \frac{N}{K}\right)N$$

where K is the carrying capacity.

Fibonacci's rabbits

- two rabbits give birth to one pair of infant
- infant grows up in one month and mate, pair infant born next month

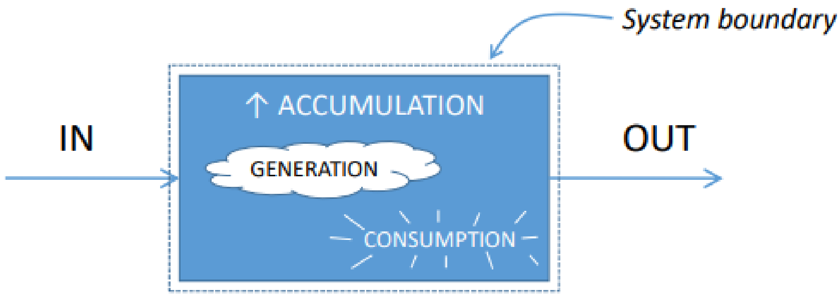
$$\begin{cases} A(N) & = A(N-1) + B(N-1) \\ B(N) & = A(N-1) \end{cases}$$

Lotka-Volterra Model

$$\begin{cases} \dfrac{dR}{dt} = \alpha R - \beta RF & R(0) = R_0 \\ \dfrac{dF}{dt} = \gamma RF - \delta F & F(0) = F_0 \end{cases}$$

Chemical Kinetics

Component balance



$$\dot{n}_{i,in} - \dot{n}_{i,out} + \sum_r \nu_{r,i}\, \dot{\xi}_r = \frac{dN_i}{dt}$$

Molar flow rate of i entering system

Molar flow rate of i leaving system

Reaction rates of i generated/consumed by all reactions in system

$\nu > 0$ *for generation,*
$\nu < 0$ *for consumption,*
$\nu = 0$ *for not involved in reaction*

From equation, obtain **stoichiometric numbers**.

- We define the *extent of reaction*, ξ:

$$\delta\xi \equiv \frac{dN_1}{\nu_1} = \frac{dN_2}{\nu_2} = \frac{dN_3}{\nu_3} = \ldots$$

$$\nu_i \xi = \Delta N_i$$

$$\nu_i \dot{\xi} = \frac{dN_i}{dt}$$

- Stirred-tank Reactor
- Catalysis

$$S + E \rightarrow SE \quad (\text{rate constant: } k_1)$$
$$SE \rightarrow S + E \quad (\text{rate constant: } k_{-1})$$
$$SE \rightarrow P + E \quad (\text{rate constant: } k_2)$$

Second-Order Differential Equations

$$\frac{d^2y}{dt^2} = f\left(t, y, \frac{dy}{dt}\right)$$

- $\{y(t_0), y'(t_0)\}$: initial value problem (IVP)
- $\{y(t_1), y(t_2)\}$: boundary value problem (**Dirichlet type**)
- $\{y'(t_1), y'(t_2)\}$: boundary value problem (**Neumann type**)

Solving second-order ODEs by computer

- The trick is to convert a second-order ODE into a system of two first-order ODEs, which we know how to solve, by *defining the first derivative as a new variable*:

$$\frac{d^2y}{dt^2} = f\left(t, y, \frac{dy}{dt}\right) \Rightarrow \begin{cases} \dfrac{dy}{dt} = \dot{y} \\ \dfrac{d\dot{y}}{dt} = f(t, y, \dot{y}) \end{cases}$$

Initial values:

$$y(0) = y_0$$

$$\dot{y}(0) = \left.\frac{dy}{dt}\right|_{t=0} = v_0$$

- Package into vector form:

$$\frac{d}{dt}\vec{Y} = \vec{F}(t, \vec{Y}) \quad where \quad \vec{Y} = \begin{bmatrix} y \\ \dot{y} \end{bmatrix} \;;\; \vec{F} = \begin{bmatrix} \dot{y} \\ f(t, y, \dot{y}) \end{bmatrix}$$

Initial value:

$$\vec{Y}(0) = \begin{bmatrix} y_0 \\ v_0 \end{bmatrix}$$

Shooting Method

- Simulate 2nd-order ODE by computer
- Compare the value of the function at $y(t_2)$ and the given initial condition
- Using **method of bisection** find the missing initial value

Finite Difference Methods

For a linear second order ODE

$$\frac{d^2y}{dt} + p(t)\left(\frac{dy}{dt}\right) + q(t)y(t) = r(t)$$

Approximate derivatives by *finite differences*:

$$\frac{dy}{dt} \approx \frac{y_{i+1} - y_{i-1}}{2h} \qquad \frac{d^2y}{dt^2} \approx \frac{\frac{y_{i+1}-y_i}{h} - \frac{y_i - y_{i-1}}{h}}{h} = \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2}$$

Substituting these finite differences back into our *ODE* and rearranging the y_i terms, we have

$$\left(\frac{1}{h^2} + \frac{p(t_i)}{2h}\right) y_{i+1} + \left(-\frac{2}{h^2} + q(t_i)\right) y_i + \left(\frac{1}{h^2} - \frac{p(t_i)}{2h}\right) y_{i-1} = r(t_i)$$

The two types of boundary conditions give us 2 extra equations. Lastly, we transform our $(n+1)$ equations into a linear system of equations:

$$\left(\frac{1}{h^2}+\frac{p}{2h}\right)y_2+\left(-\frac{2}{h^2}+q\right)y_1+\left(\frac{1}{h^2}-\frac{p}{2h}\right)y_0=r$$

$$\left(\frac{1}{h^2}+\frac{p}{2h}\right)y_3+\left(-\frac{2}{h^2}+q\right)y_2+\left(\frac{1}{h^2}-\frac{p}{2h}\right)y_1=r$$

$$\vdots$$

$$\left(\frac{1}{h^2}+\frac{p}{2h}\right)y_n+\left(-\frac{2}{h^2}+q\right)y_{n-1}+\left(\frac{1}{h^2}-\frac{p}{2h}\right)y_{n-2}=r$$

$$\Rightarrow \begin{bmatrix} 1 & 0 & 0 & & 0 & 0 & 0 \\ C & B & A & \cdots & 0 & 0 & 0 \\ 0 & C & B & & 0 & 0 & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & 0 & 0 & & B & A & 0 \\ 0 & 0 & 0 & \cdots & C & B & A \\ 0 & 0 & 0 & & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-2} \\ y_{n-1} \\ y_n \end{bmatrix} = \begin{bmatrix} \alpha \\ r \\ r \\ \vdots \\ r \\ \alpha \end{bmatrix}$$

We can solve this linear system for the y_i's!

Transport Phenomena and Partial Differential Equations

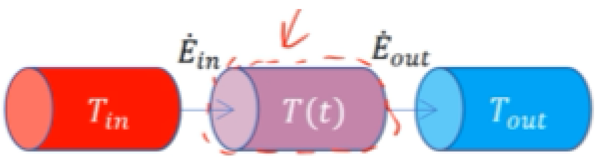
- *Partial differential equations* (PDEs) are equations involving partial derivatives.
  - By definition, this means our dependent variable is a multivariable function.
- Types of linear PDEs (A, B, C, D, E, F, G can depend on t and x, but not on M):

$$A\frac{\partial^2 M}{\partial t^2}+B\frac{\partial^2 M}{\partial t \partial x}+C\frac{\partial^2 M}{\partial x^2}+D\frac{\partial M}{\partial t}+E\frac{\partial M}{\partial x}+FM+G=0$$

  - First-order ($A = B = C = 0$): e.g. advection
  - Parabolic ($B^2 - 4AC = 0$): e.g. one-dimensional heat conduction / diffusion
  - Elliptic ($B^2 - 4AC < 0$): e.g. steady-state two-dimensional heat conduction / diffusion
  - Hyperbolic ($B^2 - 4AC > 0$): e.g. wave equation

Heat Conduction $T(t, x)$

Lumped model (Finite Volume)



1. Balance Equation

$$\Delta E = mC\Delta T \Longrightarrow \frac{dE}{dt} = mC\frac{dT}{dt} = q_{in}A - q_{out}A$$

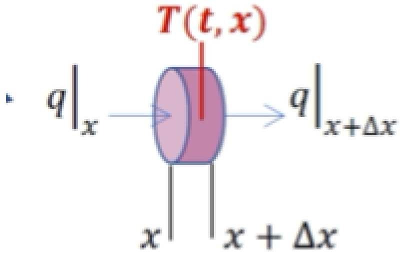
   q: heat flux

2. Constitutive Equation

$$q_{in} = \frac{k}{L}(T_{in} - T)$$

   Also consider that T_{in} and T_{out} varies as time.

Continuous Model



1. Balance Equation (continuous **energy conservation**)

$$\frac{dE}{dt} = A(q(x) - q(x+\Delta x)) \xrightarrow[m=\rho A \Delta x]{E=mC\Delta T} \frac{\partial T}{\partial t} = -\frac{1}{\rho C}\left(\frac{\partial q}{\partial x}\right)$$

2. Constitutive Equation (**Fourier's Law**)

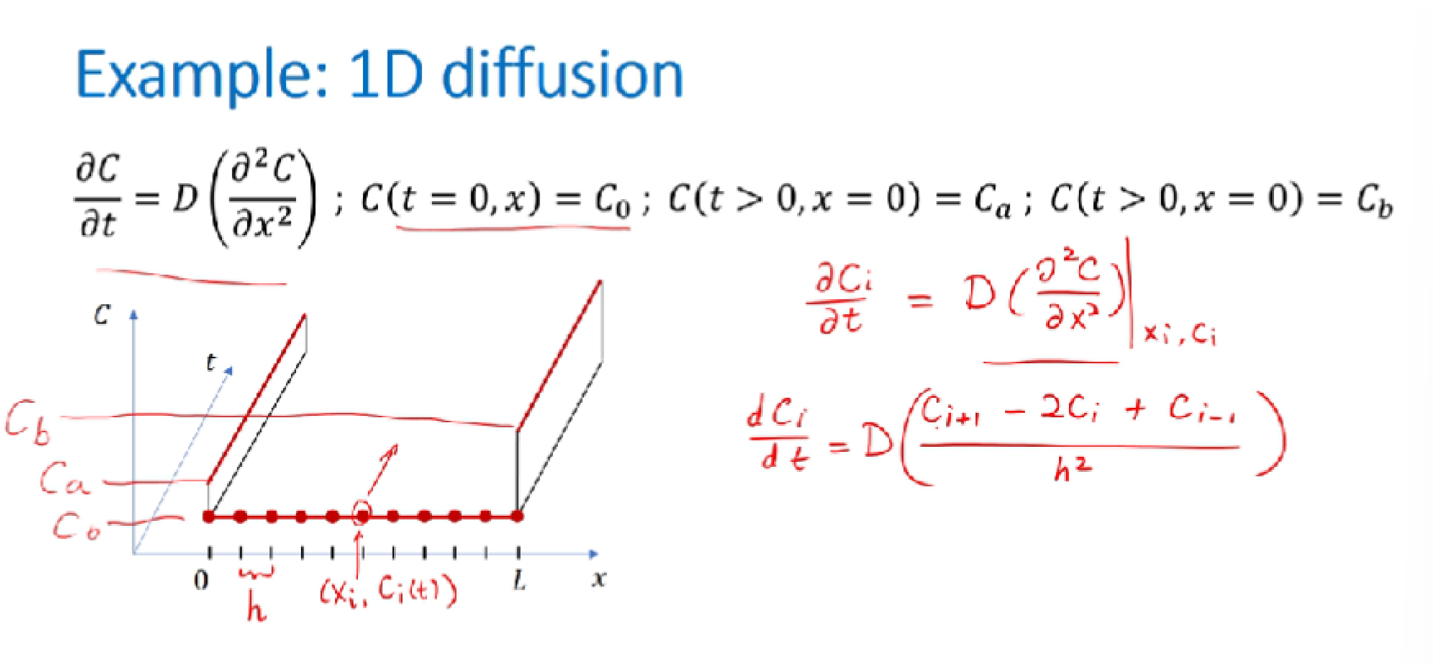
$$q_x = -k\frac{\partial T}{\partial x}$$

Finally, we get

$$\frac{\partial T}{\partial t} = \frac{k}{\rho C}\frac{\partial^2 T}{\partial x^2}$$

- Fick's Law: $J_x = -D\left(\frac{\partial C}{\partial x}\right)$

Method of Lines



becomes **a system of coupled equations**

$$\frac{dC_i}{dt} = \left(\frac{D}{h^2}\right)C_{i+1} + \left(\frac{-2D}{h^2}\right)C_i + \left(\frac{D}{h^2}\right)C_{i-1}$$

which we can model as a **matrix** or **shifted vectors**

Matrix

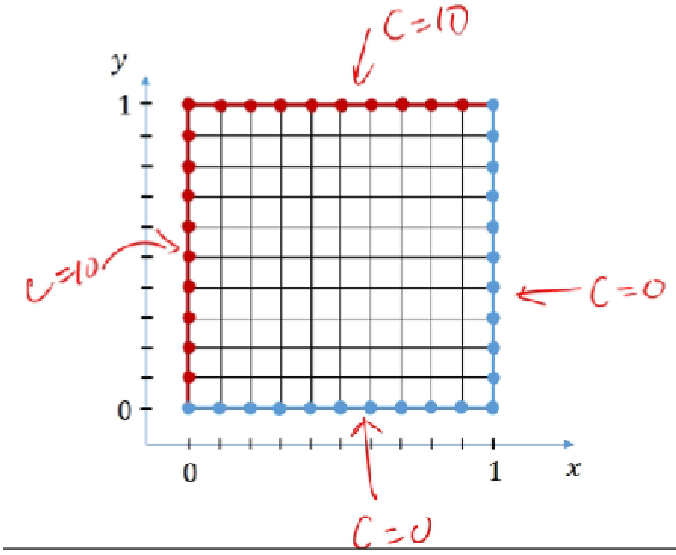
$$\frac{d}{dt}\begin{bmatrix} C_0 \\ C_1 \\ \vdots \\ \vdots \\ C_{10} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \cdots & \cdots & 0 \\ \frac{D}{h^2} & -\frac{2D}{h^2} & \frac{D}{h^2} & 0 & \cdots & 0 \\ 0 & \frac{D}{h^2} & -\frac{2D}{h^2} & \frac{D}{h^2} & \cdots & 0 \\ & 0 & \ddots & \ddots & \ddots & 0 \\ & & & \frac{D}{h^2} & -\frac{2D}{h^2} & \frac{D}{h^2} \\ 0 & 0 & 0 & \cdots & \cdots & 0 \end{bmatrix} \begin{bmatrix} C_0 \\ C_1 \\ \vdots \\ \vdots \\ C_{10} \end{bmatrix}$$

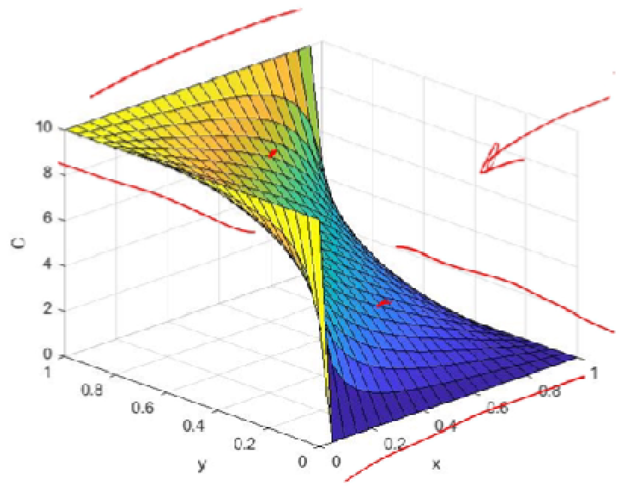
Shifted vectors

$$\frac{d}{dt}\begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_9 \end{bmatrix} = \frac{D}{h^2}\begin{bmatrix} C_2 \\ C_3 \\ \vdots \\ C_{10} \end{bmatrix} - \frac{2D}{h^2}\begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_9 \end{bmatrix} + \frac{D}{h^2}\begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ \vdots \\ C_8 \end{bmatrix}$$

- Steady-state: $\dfrac{\partial C}{\partial t} = 0$
- 2D Steady-state Diffusion: $\dfrac{\partial C}{\partial t} = D\left(\dfrac{\partial^2 C}{\partial x^2} + \dfrac{\partial^2 C}{\partial y}\right)$, fix boundaries

  discretize x and y coordinates, and use finite differences to get a system of linear equations





Basic MATLAB Syntax

```
# Matrix manipulation
A(:, 2)
A(2, [2, 3])
A.'            # transpose
zeros(4, 3)    # zero matrix
eye(3)         # identity matrix
ones(4, 3)     # ones matrix

x = 1:1:10            # (first element):(step size):(last element)
linspace(1,10,100)   # (first):(last):(total elements)
```

```
# For loops:

for i=1:1:top
...
end


# If statements:
if n ==0
...
elseif n == 1
...
else
...
end
```

Some useful functions in Matlab

Graph Plotting

- MATLAB simply plots a series of data points (x_i, y_i) and (optionally) joins them together.
- Syntax: plot(x, y, *style*)
  - x is the vector (row or column) of x-coordinates of the data points: $x = [x_1, x_2, x_3 \ldots]$
  - y is the vector (row or column) of y-coordinates of the data points: $y = [y_1, y_2, y_3 \ldots]$
  - *style* is a string:
    - 'b-' means blue solid line
    - 'g:' means green dotted line
    - 'rx' means red crosses (no line)
    - 'k-x' means black solid line, with data points marked with crosses

figure(1)	shows a figure
hold on, hold off	whether to overlay on the old graph
xlabel('x'), ylabel('1/x')	
title('reciprocal')	
legend('x')	
grid on, grid off	
plot(x1,y1,style1,x2,y2,style2)	
plot3(x,y,z,style)	
>> x=1:1:10;y=1:1:10; >> [XX YY]=meshgrid(x,y) >> zz = log(1+XX .^ 2+YY.^2) >> surf(XX,YY,zz)	plotting a surface

Root Finding

```
r = fzero(@function, initial_guess)

# initial_guess can be
# x_0: initial guess
# [x_0, x_1]: root range
```

Definite Integration

```
I = integral(@function, limit_lower, limit_upper)
```

Solving ODEs

```
function [] = doLotka(a, b, c, e)
```

```
    [T, Y] = ode45(@lotka, [0, 365], [100; 10]);

    plot(T, Y(:,1), 'r-', T, Y(:,2), 'g-');

    function D = lotka(t, X)
        R = X(1);
        F = X(2);
        dRdt = a .* R - b .* R .* F;
        dFdt = e .* b .* R .* F - c .* F;
        D = [dRdt; dFdt];
    end
end
```