Solution: 2

According to the French Flag model, morphogens form a gradient across a field of cells, and cells determine their fate according to the local concentration of the morphogen. In this exercise, you will solve two standard 1-component models for morphogen gradients that lead to either a linear or an exponential gradient. **The missing 2 questions from exercise 1 will be uploaded as soon as possible.**

1. **Review Material from the Lecture**

   - Explain how the diffusion equation is derived!
   - According to the French Flag model, the morphogen concentration forms a gradient across a tissue, and cell fate depends on whether the local morphogen concentration is above or below a threshold concentration.
   - Without a sink, the morphogen will continuously accumulate until the domain is filled entirely; no steady state gradient emerges. A degradation term can work as a sink and results in a steady state gradient profile.
   - Why is it important to know the time to steady-state? What does this time depend on?

2. **Steady-state Gradient profiles.**

   (i) The steady state solution of

$$\frac{\partial c}{\partial t} = D \Delta c = 0$$

can be obtained by integrating the spatial term twice, i.e.

$$\begin{aligned} D\frac{\partial^2 c}{\partial x^2} &= 0 \\ \Leftrightarrow \quad \frac{\partial c}{\partial x} &= a_1 \\ \Leftrightarrow \quad c &= a_1 \cdot x + a_2 \end{aligned}$$

The integration constants a_1, a_2 need to be set to meet the boundary conditions, i.e.

$$\begin{array}{llll} \text{BC1:} \quad c(x=0,t) = c_0 & \Rightarrow & c_0 = a_2 & \\ \text{BC2:} \quad c(x=L,t) = 0 & \Rightarrow & 0 = a_1 \cdot L + a_2 = a_1 \cdot L + c_0 = 0 & \Leftrightarrow \quad a_1 = -c_0/L \end{array}$$

In summary, we have for the steady state solution

$$c + c_0/L \cdot x - c_0 = 0 \quad \Leftrightarrow \quad c(x) = \frac{c_0 \cdot (L-x)}{L}$$

(ii) The steady state solution of

$$\frac{\partial c}{\partial t} = D\Delta c - kc = 0$$

can be obtained by transforming the second order ODE into a set of two first order ODES, i.e.

$$\begin{aligned}\frac{dc}{dx} &= z \\ \frac{dz}{dx} &= \frac{k}{D}c = \frac{c}{\lambda^2}, \qquad \lambda = \sqrt{\frac{D}{k}}.\end{aligned}$$

Using the chain rule we can further write

$$\frac{dz}{dx} = \frac{dz}{dc}\frac{dc}{dx} = \frac{dz}{dc}z.$$

We then have

$$\frac{dz}{dc}z = \frac{c}{\lambda^2}$$

and separation of variables yields

$$z \cdot dz = \frac{c}{\lambda^2}dc.$$

Because of the boundary condition, both c and z go to zero as x goes to infinity, and we therefore have

$$z^2 = \frac{1}{\lambda^2}c^2.$$

Moreover, because we always have $c \geq 0$ and $z = \frac{dc}{dx} \leq 0$ (otherwise the boundary condiotions wouldn't be satisfied), we get

$$z = \frac{dc}{dx} = -\frac{1}{\lambda}c,$$

which can be solved (using separation of variables followed by integration) to yield

$$c = c_0 \cdot \exp(-x/\lambda),$$

where we used the boundary condition $c(x = 0, t) = c_0$.

3. **Time-dependent Solution**.

$$\text{PDE:} \qquad \frac{\partial c}{\partial t} = D\frac{\partial^2 c}{\partial x^2} - kc$$

Ansatz: Separation of Variables: $c\,(t, x) = T\,(t)\,X\,(x)$

$$\begin{aligned}\frac{\partial(T\,(t)\,X\,(x))}{\partial t} &= D\frac{\partial^2(T\,(t)\,X\,(x))}{\partial x^2} - k(T\,(t)\,X\,(x)) \\ X\,(x)\,\frac{\partial T\,(t)}{\partial t} &= T\,(t)\,D\frac{\partial^2 X\,(x)}{\partial x^2} - k(T\,(t)\,X\,(x)) \\ \frac{T'\,(t)}{T\,(t)} + k &= \sigma = D\frac{X''\,(x)}{X\,(x)}\end{aligned}$$

The solution of the temporal part reads:

$$T\left(t\right) = T(0)e^{(\sigma-k)t}$$

The solution of the spatial part reads:

$$X\left(x\right) = C_1 e^{\sqrt{\sigma/D}x} + C_2 e^{-\sqrt{\sigma/D}x}$$

If $\sigma < 0$, then the exponent is complex. The general solution has the shape:

$$c\left(t,x\right) = T\left(t\right)X\left(x\right) = T(0)e^{(\sigma-k)t}\left(C_1 e^{\sqrt{\sigma/D}x} + C_2 e^{-\sqrt{\sigma/D}x}\right)$$

We now need to take care of the boundary and initial conditions.

$$\begin{array}{rll} \text{PDE:} & \dfrac{\partial c}{\partial t} = D\dfrac{\partial^2 c}{\partial x^2} - kc & \\ \text{BC:} & c(t,0) = c_0, \qquad c(t,L) = 0 & \\ \text{IC:} & c(0,0) = c_0; \qquad c(0,x) = 0 \quad \forall \quad x > 0. & \end{array} \tag{1}$$

We have previously determined the steady-state solution:

$$\begin{aligned} c_s(x) &= c_0 \frac{\sinh\left(\frac{L-x}{\lambda}\right)}{\sinh\left(\frac{L}{\lambda}\right)} \\ \lambda &= \sqrt{\frac{D}{k}} \end{aligned}$$

We now generate a new function $u(x,t) = c(x,t) - c_s(x)$.
With $u(x,t) = c(x,t) - c_s(x)$,

$$\begin{array}{rl} \text{PDE:} & \dfrac{\partial u}{\partial t} = \dfrac{\partial(c-c_s)}{\partial t} = D\dfrac{\partial^2(c-c_s)}{\partial x^2} - k(c-c_s) \\ & \qquad = D\dfrac{\partial^2 u}{\partial x^2} - ku \\ \text{BC:} & u(t,0) = c(t,0) - c_s(0) = c_0 - c_0 = 0 \\ & u(t,L) = c(t,L) - c_s(L) = 0 \\ \text{IC:} & u(0,0) = c(0,0) - c_s(0) = 0 \\ & u(0,x) = c(0,x) - c_s(0,x) \\ & \qquad = -c_0 \dfrac{\sinh\left(\frac{L-x}{\lambda}\right)}{\sinh\left(\frac{L}{\lambda}\right)} = f(x) \quad \forall \quad x > 0. \end{array}$$

We now solve the PDE for u with homogenous boundary conditions.
We have as general solution,

$$u\left(t,x\right) = T\left(t\right)X\left(x\right) = T(0)e^{(\sigma-k)t}\left(C_1 e^{\sqrt{\sigma/D}x} + C_2 e^{-\sqrt{\sigma/D}x}\right).$$

As $c(x,t) = c_s + u(x,t)$, we require $u\left(t,x\right) \to 0$ as $t \to \infty$ and thus

$$\sigma < k.$$

Substituting $\sqrt{\sigma/D} = i\sqrt{-\sigma/D} = i\omega$, we obtain

$$X\left(x\right) \quad = \quad \cos\left(\omega x\right)\left(C_1 + C_2\right) + i\sin\left(\omega x\right))(C_1 - C_2).$$

To meet the homogenous boundary conditions as $x = 0$ and $x = L$, we require $(C_1 + C_2) = 0$ and $\omega = \frac{n\pi}{L}$, such that

$$X\left(x\right) = 2iC_1 \sin\left(\frac{n\pi x}{L}\right).$$

From $\sqrt{-\sigma/D} = \omega$, we further obtain $\sigma = -D\omega^2 = -D\frac{n^2\pi^2}{L^2}$.

According to the superposition principle, the solution that respects the boundary conditions reads:

$$u\left(t,x\right) = \sum_{n=1}^{\infty} A_n e^{-\left(D\frac{n^2\pi^2}{L^2}+k\right)t} \sin\left(\frac{n\pi x}{L}\right); \qquad A_n = 2iC_1 T(0).$$

Substituting the general solution into the IC ...

$$\begin{aligned} u\left(0,0\right) &= 0 \stackrel{!}{=} \sum_{n=1}^{\infty} A_n \sin\left(\frac{n\pi\cdot 0}{L}\right) = 0 \\ u\left(0,x\right) &= -c_0 \frac{\sinh\left(\frac{L-x}{\lambda}\right)}{\sinh\left(\frac{L}{\lambda}\right)} = f(x) \stackrel{!}{=} \sum_{n=1}^{\infty} A_n \sin\left(\frac{n\pi x}{L}\right) \quad \forall \quad x > 0 \end{aligned}$$

As seen in the lecture

$$A_n = \frac{2}{L}\int_0^L f(x)\sin\left(\frac{n\pi x}{L}\right)dx.$$

Plugging in $f(x)$ delivers

$$\begin{aligned} A_n &= \frac{2}{L}\int_0^L \left(-c_0 \frac{\sinh\left(\frac{L-x}{\lambda}\right)}{\sinh\left(\frac{L}{\lambda}\right)} \sin\left(\frac{n\pi x}{L}\right)\right) dx. \\ &= \frac{-2c_0}{L\sinh\left(\frac{L}{\lambda}\right)} \int_0^L \left(\sinh\left(\frac{L-x}{\lambda}\right)\sin\left(\frac{n\pi x}{L}\right)\right) dx. \\ &= \frac{-2c_0}{L\sinh\left(\frac{L}{\lambda}\right)} I \end{aligned}$$

We now need to determine the integral I.

$$\begin{aligned} I &= \int_0^L \left(\sinh\left(\frac{L-x}{\lambda}\right)\sin\left(\frac{n\pi x}{L}\right)\right) dx. \\ &= -\frac{L}{n\pi}\left[\sinh\left(\frac{L-x}{\lambda}\right)\cos\left(\frac{n\pi x}{L}\right)\right]_0^L \\ &\quad -\frac{L}{\lambda n\pi}\int_0^L \left(\cosh\left(\frac{L-x}{\lambda}\right)\cos\left(\frac{n\pi x}{L}\right)\right) dx. \end{aligned} \tag{2}$$

$$
\begin{aligned}
I_2 &= \int_0^L \left(\cosh\left(\frac{L-x}{\lambda}\right)\cos\left(\frac{n\pi x}{L}\right)\right) dx \\
&= \frac{L}{n\pi}\left[\cosh\left(\frac{L-x}{\lambda}\right)\sin\left(\frac{n\pi x}{L}\right)\right]_0^L \\
&\quad + \frac{L}{\lambda n\pi}\int_0^L \left(\sinh\left(\frac{L-x}{\lambda}\right)\sin\left(\frac{n\pi x}{L}\right)\right) dx. \qquad (3)
\end{aligned}
$$

$$
\begin{aligned}
I &= \int_0^L \left(\sinh\left(\frac{L-x}{\lambda}\right)\sin\left(\frac{n\pi x}{L}\right)\right) dx \\
&= \frac{L}{n\pi}\sinh\left(\frac{L}{\lambda}\right) \\
&\quad - \left(\frac{L}{\lambda n\pi}\right)^2 \int_0^L \left(\sinh\left(\frac{L-x}{\lambda}\right)\sin\left(\frac{n\pi x}{L}\right)\right) dx
\end{aligned}
$$

After rearrangements we obtain

$$
I = \frac{\frac{L}{n\pi}\sinh\left(\frac{L}{\lambda}\right)}{\left(1+\left(\frac{L}{\lambda n\pi x}\right)^2\right)}
$$

With these calculations we can determine A_n as follows,

$$
\begin{aligned}
A_n &= \frac{-2c_0}{L\sinh\left(\frac{L}{\lambda}\right)} I \\
&= \frac{-2c_0}{L\sinh\left(\frac{L}{\lambda}\right)} \frac{\frac{L}{n\pi}\sinh\left(\frac{L}{\lambda}\right)}{\left(1+\left(\frac{L}{\lambda n\pi}\right)^2\right)} \\
&= \frac{-2c_0}{n\pi\left(1+\left(\frac{L}{\lambda n\pi}\right)^2\right)} \\
&= \frac{-2c_0 n\pi}{(n\pi)^2+\left(\frac{L}{\lambda}\right)^2}
\end{aligned}
$$

The solution then reads:

$$
u(t,x) = -\sum_{n=1}^{\infty} \frac{2c_0 n\pi}{(n\pi)^2+\left(\frac{L}{\lambda}\right)^2} \exp\left(-\left(D\frac{n^2\pi^2}{L^2}+k\right)t\right)\sin\left(\frac{n\pi x}{L}\right).
$$

and

$$
c(t,x) = c_s(t,x) + u(t,x)
$$

We see that the solution reaches steady state exponentially fast with rate

$$
\frac{Dn^2\pi^2}{L^2} + k.
$$

4. **Numerical Solution**.

```
function Solution_exercise2()

close all
clear all

folder = '.';
%
% PDE: ut = D*uxx-k*u
% BC: u(0, t) = u0; u(L, t) = 0
% IC: u(x,0) = c0; u(x>0,0)=0

global c0 L D  k;

c0 = 1;

%%%%%%%%%%%%%%%%%%%%
% 1. Parameter set
%%%%%%%%%%%%%%%%%%%%
L = 50;
D = 0.1;
k = 1e-5;
lambda = sqrt(D/k);

x = linspace(0,L,300);
t = [0 0.5 1 1.5 20];
m = 0; % m must be 0, 1, or 2, corresponding to
   slab, cylindrical, or spherical symmetry,
   respectively.

sol = pdepe(m,@pdex1pde_MA,@pdex1ic_MA,@pdex1bc_MA,
   x,t);
% pdepe returns values of the solution on a mesh

colormap = {'k', 'b', 'g', 'r', 'c', 'm'};

fig1 = figure(1);
hold on,
xlabel('x')
ylabel('c')
set(gca, 'FontSize', 14)
title(['Parameters: L=',num2str(L), ', D=', num2str
   (D), ', k=', num2str(k)])
for i=1:length(t)
    s{2*i-1, :} = strcat('t =', num2str(t(i)));
    s{2*i, :} = strcat('t =', num2str(t(i)));
    plot(x, sol(i,:), '-', 'Color', colormap{i}, '
       LineWidth', 2)

    % Analytical Solution
    u = 0;
    for n = 1:1500
        u = u - (2*c0*n*pi)/((n*pi)^2+(L/lambda)^2)
```

```
            *exp(-(D*(n^2*pi^2)/L^2+k)*t(i))*sin(n*
            pi*x/L);
    end
    c2(i,:) = c0*sinh((L-x)/lambda)/sinh(L/lambda)
        + u;

    % t = 0, Initial condition
    if i==1
        c2(i,:) = 0;
        c2(i,1) = c0;
    end
    plot(x, c2(i,:), '.', 'Color', colormap{i}, '
        MarkerSize', 15)
end
legend(s)

saveas(fig1, strcat(folder, '
    Fig_DiffEquation_BC1_param1', '.png'));

%%%%%%%%%%%%%%%%%%%%
% 2. Parameter set
%%%%%%%%%%%%%%%%%%%%

L = 500;
D = 100;
k = 1;
lambda = sqrt(D/k);
x = linspace(0,L,300);

sol = pdepe(m,@pdex1pde_MA,@pdex1ic_MA,@pdex1bc_MA,
    x,t);
% pdepe returns values of the solution on a mesh

fig2 = figure(2);
hold on,
xlabel('x')
ylabel('c')
set(gca, 'FontSize', 14)
title(['Parameters: L=',num2str(L), ', D=', num2str
    (D), ', k=', num2str(k)])
for i=1:length(t)
    s{2*i-1, :} = strcat('t =', num2str(t(i)));
    s{2*i, :} = strcat('t =', num2str(t(i)));

    plot(x, sol(i,:), '-', 'Color', colormap{i}, '
        LineWidth', 2)

    % Analytical Solution
    u = 0;
    for n = 1:1500
        u = u - (2*c0*n*pi)/((n*pi)^2+(L/lambda)^2)
            *exp(-(D*(n^2*pi^2)/L^2+k)*t(i))*sin(n*
            pi*x/L);
```

```
    end
    c2(i,:) = c0*sinh((L-x)/lambda)/sinh(L/lambda)
        + u;
    % t = 0, Initial condition
    if i==1
        c2(i,:) = 0;
        c2(i,1) = c0;
    end
    plot(x, c2(i,:), '.', 'Color', colormap{i}, '
        MarkerSize', 15)

end

legend(s)

saveas(fig2, strcat(folder, '
    Fig_DiffEquation_BC1_param2', '.png'));

%%%%%%%%%%%%%%%%%%%%
% 3. Parameter set
%%%%%%%%%%%%%%%%%%%%

L = 50;
D = 0.1;
k = 0;
lambda = sqrt(D/k);
x = linspace(0,L,300);

sol = pdepe(m,@pdex1pde_MA,@pdex1ic_MA,@pdex1bc_MA,
    x,t);
% pdepe returns values of the solution on a mesh

fig3 = figure(3);
hold on,
xlabel('x')
ylabel('c')
set(gca, 'FontSize', 14)
title(['Parameters: L=',num2str(L), ', D=', num2str
    (D), ', k=', num2str(k)])
for i=1:length(t)
    s{2*i-1, :} = strcat('t =', num2str(t(i)));
    s{2*i, :} = strcat('t =', num2str(t(i)));

    plot(x, sol(i,:), '-', 'Color', colormap{i}, '
        LineWidth', 2)

    % Analytical Solution for k=0
    u = 0;
    for n = 1:1500
        u = u - (2*c0*n*pi)/((n*pi)^2)*exp(-(D*(n
            ^2*pi^2)/L^2+k)*t(i))*sin(n*pi*x/L);
    end
    c2(i,:) = c0*(L-x)/L + u;
```

```
    % t = 0, Initial condition
    if i==1
        c2(i,:) = 0;
        c2(i,1) = c0;
    end
    plot(x, c2(i,:), '.', 'Color', colormap{i}, '
       MarkerSize', 15)
end

legend(s)

saveas(fig3, strcat(folder, '
   Fig_DiffEquation_BC1_param3', '.png'));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 1. Parameter set with flux boundary condition
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

L = 50;
D = 0.1;
k = 1e-5;
lambda = sqrt(D/k);
x = linspace(0,L,300);

sol = pdepe(m,@pdex1pde_MA,@pdex1ic_MA,@pdex1bc2_MA
   ,x,t);
% pdepe returns values of the solution on a mesh

fig4 = figure(4);
hold on,
xlabel('x')
ylabel('c')
set(gca, 'FontSize', 14)
title(['Flux BC, Parameters: L=',num2str(L), ', D='
   , num2str(D), ', k=', num2str(k)])
for i=1:length(t)
    s{2*i-1, :} = strcat('t =', num2str(t(i)));
    s{2*i, :} = strcat('t =', num2str(t(i)));

    plot(x, sol(i,:), '-', 'Color', colormap{i}, '
       LineWidth', 2)

    % Analytical Solution
    u = 0;
    for n = 1:1500
        u = u - (2*c0*n*pi)/((n*pi)^2+(L/lambda)^2)
           *exp(-(D*(n^2*pi^2)/L^2+k)*t(i))*sin(n*
           pi*x/L);
    end
    c2(i,:) = c0*sinh((L-x)/lambda)/sinh(L/lambda)
       + u;
    % t = 0, Initial condition
```

```
    if i==1
        c2(i,:) = 0;
        c2(i,1) = c0;
    end
    plot(x, c2(i,:), '.', 'Color', colormap{i}, '
       MarkerSize', 15)
end

legend(s)

saveas(fig4, strcat(folder, '
   Fig_DiffEquation_BC2_param1', '.png'));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 2. Parameter set with flux boundary condition
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

L = 50;
D = 0.1;
k = 0;
lambda = sqrt(D/k);
x = linspace(0,L,300);

sol = pdepe(m,@pdex1pde_MA,@pdex1ic_MA,@pdex1bc2_MA
   ,x,t);
% pdepe returns values of the solution on a mesh

fig5 = figure(5);
hold on,
xlabel('x')
ylabel('c')
set(gca, 'FontSize', 14)
title(['Flux BC, Parameters: L=',num2str(L), ', D='
   , num2str(D), ', k=', num2str(k)])
for i=1:length(t)
    s{2*i-1, :} = strcat('t =', num2str(t(i)));
    s{2*i, :} = strcat('t =', num2str(t(i)));

    plot(x, sol(i,:), '-', 'Color', colormap{i}, '
       LineWidth', 2)

    % Analytical Solution for k=0
    u = 0;
    for n = 1:1500
        u = u - (2*c0*n*pi)/((n*pi)^2)*exp(-(D*(n
           ^2*pi^2)/L^2+k)*t(i))*sin(n*pi*x/L);
    end
    c2(i,:) = c0*(L-x)/L + u;

    % t = 0, Initial condition
    if i==1
        c2(i,:) = 0;
        c2(i,1) = c0;
```

```
    end
    plot(x, c2(i,:), '.', 'Color', colormap{i}, '
       MarkerSize', 15)
end

legend(s)

saveas(fig5, strcat(folder, '
   Fig_DiffEquation_BC2_param1', '.png'));

end

function [c,f,s] = pdex1pde_MA(x,t,u,DuDx)
%PDEX1PDE  Evaluate the differential equations
   components for the PDEX1 problem.
%
%   See also PDEPE, PDEX1.

%   Lawrence F. Shampine and Jacek Kierzenka
%   Copyright 1984-2014 The MathWorks, Inc.

global D  k;

c = 1;
f = D*DuDx;
s = -k*u;

% There are npde unknown solution components that
   satisfy a system of npde equations of the form
%
%      c(x,t,u,Du/Dx) * Du/Dt = x^(-m) * D(x^m * f(x
   ,t,u,Du/Dx))/Dx + s(x,t,u,Du/Dx)
%
%      Here f(x,t,u,Du/Dx) is a flux and s(x,t,u,Du/
   Dx) is a source term.
end

function u0 = pdex1ic_MA(x)
%PDEX1IC  Evaluate the initial conditions for the
   problem coded in PDEX1.
%
%   See also PDEPE, PDEX1.

%   Lawrence F. Shampine and Jacek Kierzenka
%   Copyright 1984-2014 The MathWorks, Inc.

global c0;

if x == 0
    u0 = c0;
else
    u0 = 0;
end
```

```
end

function [pl,ql,pr,qr] = pdex1bc_MA(xl,ul,xr,ur,t)
%PDEX1BC  Evaluate the boundary conditions for the
   problem coded in PDEX1.
%
%   See also PDEPE, PDEX1.

%   Lawrence F. Shampine and Jacek Kierzenka
%   Copyright 1984-2014 The MathWorks, Inc.

global c0;

pl = ul-c0;
ql = 0;

pr = ur;
qr = 0;

% The solution components are to satisfy boundary
%     conditions at x=a and x=b for all t of the
   form
%
%          p(x,t,u) + q(x,t) * f(x,t,u,Du/Dx) = 0
%
%     q(x,t) is a diagonal matrix. The diagonal
   elements of q must be either
%     identically zero or never zero. Note that the
    boundary conditions are
%     expressed in terms of the flux rather than Du
   /Dx. Also, of the two
%     coefficients, only p can depend on u.
end


function [pl,ql,pr,qr] = pdex1bc2_MA(xl,ul,xr,ur,t)
%PDEX1BC  Evaluate the boundary conditions for the
   problem coded in PDEX1.
%
%   See also PDEPE, PDEX1.

%   Lawrence F. Shampine and Jacek Kierzenka
%   Copyright 1984-2014 The MathWorks, Inc.

global D;

pl = 1;
ql = 1/D;

pr = 0;
qr = 1/D;
```

```
% The solution components are to satisfy boundary
%      conditions at x=a and x=b for all t of the
   form
%
%          p(x,t,u) + q(x,t) * f(x,t,u,Du/Dx) = 0
%
%      q(x,t) is a diagonal matrix. The diagonal
   elements of q must be either
%      identically zero or never zero. Note that the
    boundary conditions are
%      expressed in terms of the flux rather than Du
   /Dx. Also, of the two
%      coefficients, only p can depend on u.
end
```