Symbol	Value	Unit
D	$2.3 \cdot 10^{-1}$	$\mu m^2/s$
Q_p	$1 \cdot 10^{-1}$	$1/s$
K	10	$\mu m/s$
u^∞	1	mol
v_x	$\frac{10}{\sqrt{2}}$	$\mu m/s$
v_y	$\frac{5}{\sqrt{2}}$	$\mu m/s$

1 Problem Statement We consider a circular slab of an infected organic fluid where pathogens (bacteria) consume oxygen, where a fluid flow is induced by motion. In this assignment, we only consider the consumption of the oxygen by the pathogens, and the diffusion of the oxygen through the tissue. Far away from the domain, the concentration of the oxygen is equal to the equilibrium in non-infected tissue and therefore, we assume the concentration to be equal to the equilibrium sufficiently far away. Since we are not able to consider an unbounded domain, we consider a circular domain with radius of 1 micrometer that is $\Omega = \{(x, y) \in \mathbb{R}^2 | x^2 + y^2 < 1\}$ with its boundary $\partial\Omega$. The dimensions are given in micrometers. In this assignment, we consider a steady-state equilibrium determined by diffusion and regeneration by the cells. The pathogens are treated as point sinks. Considering n_{cell} cells, we solve

$$-\nabla(D\nabla u) + \nabla \cdot (u\mathbf{v}) + \sum_{p=1}^{n_{cell}} Q_p u \delta(\mathbf{x} - \mathbf{x_p}) = 0, \qquad (x, y) \in \Omega, \qquad (1)$$

where D denotes the diffusion coefficient, $\mathbf{v}$ represents the velocity vector of the gel-like fluid and Q_k denotes the hormon secretion rate by cell k. Here u denotes the oxygen concentration. We use the convention $x = (x, y)$ to represent the spatial coordinates. Further, $\delta(.)$ represents the Dirac Delta Distribution, which is characterised by

$$\begin{cases} \delta(\mathbf{x}) = 0, & \text{if } \mathbf{x} \neq \mathbf{0}, \\ \int_\Omega \delta(\mathbf{x}) d\Omega = 1, & \text{where } \Omega \text{ contains the origin.} \end{cases}$$

Next to the above partial differential equation, we consider the boundary condition

$$D\frac{\partial u}{\partial n} - \mathbf{v} \cdot \mathbf{n}u + K(u - u^\infty) = 0, \qquad (x, y) \in \partial\Omega, \qquad (2)$$

Here K denotes the transfer rate coefficient of the hormon between the boundary of the domain and its surroundings. For the computations, we use the following values: We consider five cells, located at

$$\begin{cases} x_p = 0.6cos(\frac{2\pi(p-1)}{5}), \\ x_p = 0.6sin(\frac{2\pi(p-1)}{5}), \end{cases}$$

$p \in \{1, ..., 5\}$. In order to solve this problem, one needs to consider the following questions: 1. Give the weak formulation of the problem (partial differential equation + boundary condition). Hint $\int_\Omega \delta(\mathbf{x}) f(\mathbf{x}) d\Omega = f(\mathbf{0})$.
To derive the weak formulation, we multiply Equation (1) by the basis functions

ϕ and integrate over the domain.

$$\int_\Omega \phi \left[-\nabla(D\nabla u) + \nabla \cdot (u\mathbf{v}) + \sum_{p=1}^{n_{cell}} Q_p u \delta(\mathbf{x} - \mathbf{x}_p) \right] d\Omega = 0. \tag{3}$$

$$\begin{aligned}
&\int_\Omega \left[-\nabla(\phi D\nabla u) + \nabla\phi D\nabla u + \nabla \cdot (\phi u\mathbf{v}) - \nabla\phi\nabla \cdot (u\mathbf{v}) + \phi \sum_{p=1}^{n_{cell}} Q_p u \delta(\mathbf{x} - \mathbf{x}_p) \right] d\Omega = \\
&= \int_{\partial\Omega} -\phi D \frac{\partial u}{\partial n} + \phi\mathbf{v} \cdot \mathbf{n} u d\Gamma + \\
&+ \int_\Omega \left[-\nabla\phi\nabla \cdot (u\mathbf{v}) + \nabla\phi D\nabla u + \phi\nabla \cdot (u\mathbf{v}) + \phi \sum_{p=1}^{n_{cell}} Q_p u \delta(\mathbf{x} - \mathbf{x}_p) \right] d\Omega = \\
&= \int_{\partial\Omega} K(u - u^\infty) d\Gamma + \\
&+ \int_\Omega \left[-\nabla\phi\nabla \cdot (u\mathbf{v}) + \nabla\phi D\nabla u + \phi\nabla \cdot (u\mathbf{v}) + \phi \sum_{p=1}^{n_{cell}} Q_p u \delta(\mathbf{x} - \mathbf{x}_p) \right] d\Omega =
\end{aligned}$$

But using Equation (1) (bc) the first term above is zero, then we have:

$$\int_0^1 \left[D \frac{d\phi}{dx} \frac{du}{dx} + \lambda\phi u \right] dx = \int_0^1 \phi f(x) dx \tag{4}$$

2. Give the Galerkin equations (the system of linear equations). 3. Give the element matrix and element vector for the internal elements. Distinguish between cases where the point sink lies inside or outside the considered element.

1. Assignment 1 Derive the *weak* formulation.

2. Assignment 2 Write the Galerkin formulation of the weak form as derived in the previous assignment for a general number of elements given by n (hence $x_n = 1$). Give the Galerkin equations, that is, the linear system in terms of

   $$S\bar{u} = \bar{f}, \tag{5}$$

   all expressed in the basis-functions, $f(x)$, λ and D.
   For the Galerkin formulation we approximate u with the basis functions ϕ_j as:

   $$u(x) \sim \sum_{j=1}^{n} a_j \phi_j(x).$$

   Approximating u and substituting $\phi = \phi_i$ in Equation (4) we have:

   $$\sum_{j=1}^{n} a_j \int_0^1 \left[D \frac{d\phi_i}{dx} \frac{d\phi_j}{dx} + \lambda\phi_i\phi_j \right] dx = \int_0^1 \phi_i f(x) dx \tag{6}$$

Then

$$S_{ij}^{e_k} = a_j \int_{e_k} \left[D \frac{d\phi_i}{dx} \frac{d\phi_j}{dx} + \lambda \phi_i \phi_j \right] dx, \qquad S_{ij} = \sum_{k=1}^{nel} S_{ij},$$

$$f_i^{e_k} = \int_{e_k} \phi_i f(x) dx, \qquad f_i = \sum_{k=1}^{nel} f_i^{e_k}.$$

3. Assignment 3 Write a matlab routine, called GenerateMesh.m that generates an equidistant distribution of meshpoints over the interval [0, 1], where $x_1 = 0$ and $x_n = 1$ and $h = \frac{1}{n-1}$. You may use x = linspace(0,1,n).

Algorithm 1
function [x]=GenerateMesh(n) x = linspace(0,1,n); end

4. Assignment 4 Write a routine, called GenerateTopology.m, that generates a two dimensional array, called elmat, which contains the indices of the ver- tices of each element, that is

$$elmat(i,1) = i. \qquad for i \in 1, ..., n-1.$$
$$elmat(i,2) = i+1$$

Next we compute the element matrix S_{elem}. In this case, the matrix is the same for each element, that is, if we consider element e_i.

Algorithm 1

function [elmat] = GenerateTopology(n)
for i = 1 : n-1;
elmat(i, 1) = i;
elmat(i, 2) = i + 1;
end
end

5. Assignment 5 Compute the element matrix, S_{elem} over a generic line element e_k.
For each element we have the following marix:

$$S^{e_k} = \begin{bmatrix} \int_{e_k} D\phi'_{k-1}\phi'_{k-1} + \lambda\phi_{k-1}\phi_{k-1}dx & \int_{e_k} D\phi'_{k-1}\phi'_k + \lambda\phi_{k-1}\phi_k dx \\ \int_{e_k} D\phi'_k\phi'_{k-1} + \lambda\phi_k\phi_{k-1}dx & \int_{e_k} D\phi'_k\phi'_k + \lambda\phi_k\phi_k dx \end{bmatrix}$$

Algorithm 3

```
function [Selem]=GenerateElementMatrix(D,l,n)
    h = 1/(n-1);
    Selem(1,1) = D/h + l*h/3;
    Selem(1,2) = -D/h + l*h/6;
    Selem(2,1) = -D/h + l*h/6;
    Selem(2,2) = D/h + l*h/3;
end
```

According to Holland and Bell's Theorem, we have:

$$\int_{e_k} \phi_{k-1}\phi_{k-1}dx = \frac{||x_k - x_{k-1}||}{3}$$
$$\int_{e_k} \phi_k\phi_{k-1}dx = \frac{||x_k - x_{k-1}||}{6}$$
$$\int_{e_k} \phi'_{k-1}\phi'_{k-1}dx = \frac{1}{||x_k - x_{k-1}||}$$
$$\int_{e_k} \phi'_{k-1}\phi'_{k}dx = -\frac{1}{||x_k - x_{k-1}||}$$

Then we have

$$S^{e_k} = \begin{bmatrix} D\frac{1}{||x_k - x_{k-1}||} + \lambda\frac{||x_k - x_{k-1}||}{3} & -D\frac{1}{||x_k - x_{k-1}||} + \lambda\frac{||x_k - x_{k-1}||}{6} \\ -D\frac{1}{||x_k - x_{k-1}||} + \lambda\frac{||x_k - x_{k-1}||}{6} & D\frac{1}{||x_k - x_{k-1}||} + \lambda\frac{||x_k - x_{k-1}||}{3} \end{bmatrix}$$

But $||x_k - x_{k-1}|| = h = \frac{1}{n-1}$ then

$$S^{e_k} = \begin{bmatrix} D\frac{1}{h} + \lambda\frac{h}{3} & -D\frac{1}{h} + \lambda\frac{h}{6} \\ -D\frac{1}{h} + \lambda\frac{h}{6} & D\frac{1}{h} + \lambda\frac{h}{3} \end{bmatrix}$$

6. Assignment 6 Write a matlab routine, called GenerateElementMatrix.m, in which S_{elem} (2 x 2-matrix) is generated. Subsequently, we are going to sum the connections of the vertices in each element matrix, over all the elements. The result is an n-by-n matrix, called S.

7. Assignment 7 Write a matlab routine, called AssembleMatrix.m, that performs this summation, such that S is first initialized as a zero n-by-n matrix and subsequently:

$$S(elmat(i,j), elmat(i,k)) = S(elmat(i,j), elmat(i,k)) + Selem(j,k), \quad (7)$$

for $j, k \in \{1, 2\}$ over all elements $i \in \{1, ..., n-1\}$. Note that GenerateElementMatrix.m needs to be called for each element. Now, you developed a routine for the assembly of the large matrix S from the element matrices Selem for each element. This procedure is common for the construction of the large discretization matrices needed in Finite Element methods. The procedure, using the array elmat looks a bit overdone and complicated.

Algorithm 4

```
function [S]=AssembleMatrix(elmat,Selem,n)
S = zeros(n,n);
for i = 1 : n-1
    for j = 1 : 2
        for k = 1 : 2
            S(elmat(i, j), elmat(i, k)) = S(elmat(i, j), elmat(i, k))+Selem(j, k);
        end
    end
end
end
```

However, this approach facilitates the application to multi dimensional problems. The next step is to generate a large right-hand side vector using the same procedure. First, we need the element vector.

8. Assignment 8 Compute the element vector over a generic line-element. For this purpose, we proceed as follows

9. Assignment 9 Implementation of the right-hand vector:

   a Write a matlab routine, called GenerateElementVector.m, that gives the vector felem (column vector of length 2). in which felem (1) and felem (2) respectively provide information about node i and node i + 1, which are the vertices of element e_i . This is needed for all elements. Use $f(x) = 1$ here.
   According to Newton-Cotes' Theorem, we have:

$$\int_{e_k} \phi_{k-1} f(x)dx = \frac{x_k - x_{k-1}}{2} f(x_k)$$

$$f_{e_k} = \begin{bmatrix} \int_{e_k} \phi_{k-1} f(x)dx \\ \\ \int_{e_k} \phi_k f(x)dx \end{bmatrix}$$

   Then, for each element we have

$$f_{e_k} = \frac{h}{2} \begin{bmatrix} f(x_{k-1}) \\ \\ f(x_k) \end{bmatrix} = \frac{h}{2} \begin{bmatrix} 1 \\ \\ 1 \end{bmatrix}$$

   b Write a matlab routine, called AssembleVector.m, that performs the following summation after setting $f = zeros(n, 1)$:

$$f(elmat(i, j)) = f(elmat(i, j)) + felem(j), \tag{8}$$

   for $j \in \{1, 2\}$ over all elements $i \in \{1, ..., n - 1\}$.

10. Assignment 10 Run the assembly routines to get the matrix S and vector f for $n = 100$.

Algorithm 5

```
function [felem]=GenerateElementVector(n)
h = 1 / (n-1);
for i=1:n-1
felem(1,i) = (h/2)*fn(x(i));
felem(2,i) = (h/2)*fn(x(i+1));
end
end
```

Algorithm 6

```
[f]=AssembleVector(felem,elmat,n)
f=zeros(n,1);
for i=1:n-1
    for j=1:2
        f(elmat(i,j)) = f(elmat(i,j)) + felem(j,i);
    end
end
end
```

Algorithm 7

```
clear all
[x] = GenerateMesh(n);
[elmat] = GenerateTopology(n);
[Selem] = GenerateElementMatrix(D,lambda,n);
[S] = AssembleMatrix(elmat,Selem,n);
[felem] = GenerateElementVector(n,fn,x(1:2));
[f] = AssembleVector(felem,elmat,n);
 u = S\f;
plot(x,u)
```

Algorithm 7b

```
clear all
n = 100;
D = 1;
lambda = 1;
fn = @(x) 1;
[x] = GenerateMesh(n);
[elmat] = GenerateTopology(n);
[Selem] = GenerateElementMatrix(D,lambda,n);
[S] = AssembleMatrix(elmat,Selem,n);
[felem] = GenerateElementVector(n,fn,x);
[f] = AssembleVector(felem,elmat,n);
 u = S\f;
plot(x,u)
```

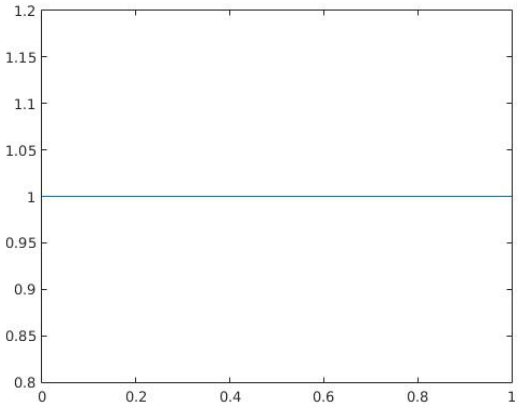

Figure 1: Result $f(x) = 1$.

11. Assignment 11 Write the main program that gives the finite-element solution. Call the main program femsolve1d.m.

12. Assignment 12 Compute the Finite Element solution u for $f(x) = 1$, $D = 1$, $\lambda = 1$ and $n = 100$ by using $u = S\backslash f$ in matlab. Plot the solution. Is this what you would expect?

13. Assignment 13 Choose $f(x) = sin(20x)$, do some experiments with several values of n (n = 10, 20, 40, 80 160). Plot the solutions for the various numbers of gridnodes in one plot. Explain what you see.

Algorithm 7c

```
clear all
close all
t = 0;
for n = [10 20 40 80 160]
    n = 100;
    D = 1;
    lambda = 1;
    fn = @(x) sin(20*x);
    [x] = GenerateMesh(n);
    [elmat] = GenerateTopology(n);
    [Selem] = GenerateElementMatrix(D,lambda,n);
    [S] = AssembleMatrix(elmat,Selem,n);
    [felem] = GenerateElementVector(n,fn,x);
    [f] = AssembleVector(felem,elmat,n);
     u = S\f;
    plot(x,u)
    hold on
    t = t + 1;
    name{t}=num2str(n);
end
legend(['n=' name{1}],['n=' name{2}],['n=' name{3}],['n=' name{4}],['n=' name{5}])
```

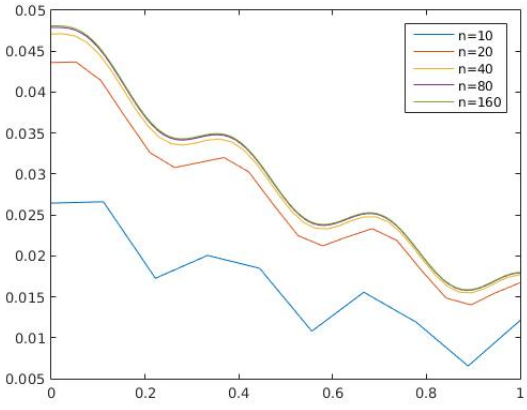

Figure 2: Results $f(x) = sin(20x)$.