Uppsala university, Division of Scientific computing

Project – Design a Medical Torus

Applied Finite Element Methods 1TD056 (5.0 hp)

Murtazo Nazarov

November 5, 2021

Introduction

The overall grand project is to design a medical torus that emits a hormone through diffusion. The design variables are the shape of the torus and the initial concentration of the hormone in the torus. The design constraints are the time until a certain amount of hormone has been emitted. Additional requirements are robustness to manufacturing errors and to model errors.

In this project, we consider a medical torus of outer radius R and inner radius r that may be used in certain hormone treatments (see Fig. 1). It is inserted just under the skin on the inside of the upper arm where the conditions are such that the hormone will slowly diffuse out in the surrounding tissue.

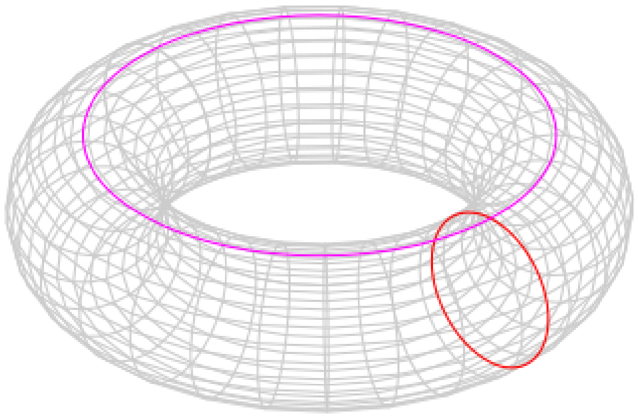

Figure 1: The geometry of a torus; R is the radius of the magenta circle, r is the radius of the red circle. *Image source:* Wikipedia.

Let $u = u(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d, t)$ mmol/mm^3 be the time-dependent concentration of hormone, where d denotes the space dimension. At $t = 0$, $u = \rho$ mmol/mm^3 inside the torus, and $u = 0$ outside. For $t > 0$, the concentration obeys

$$
\begin{aligned}
\partial_t u(\boldsymbol{x}, t) - \alpha \Delta u(\boldsymbol{x}, t) &= f(\boldsymbol{x}), & (\boldsymbol{x}, t) &\in \mathcal{B} \times (0, T], & (1)\\
u(\boldsymbol{x}, t) &= 0, & (\boldsymbol{x}, t) &\in \partial\mathcal{B} \times (0, T], & (2)\\
u(\boldsymbol{x}, 0) &= \begin{cases} \rho, & \boldsymbol{x} \in \mathcal{T}, \\ 0, & \boldsymbol{x} \in \mathcal{B} \backslash \mathcal{T}, \end{cases} & \boldsymbol{x} &\in \mathcal{B}, & (3)
\end{aligned}
$$

where α is the diffusion coefficient that is $\alpha = 0.01$ mm^2/day, $f(\boldsymbol{x})$ is a given source function, $\mathcal{B} := \{\boldsymbol{x} \in \mathbb{R}^d : \ \|\boldsymbol{x}\| \leq 1\}$ is a unit ball of radius 1 located at the origin of the space. Here the Euclidian norm of the vector of coordinates is used, i.e. $\|\boldsymbol{x}\| := \left(\boldsymbol{x}_1^2 + \ldots + \boldsymbol{x}_d^2\right)^{\frac{1}{2}}$. We denote the boundary of the ball by $\partial\mathcal{B} = \{\boldsymbol{x} \in \mathbb{R}^d : \ \|\boldsymbol{x}\| = 1\}$. The equation (2) states that the homogeneous Dirichlet boundary condition is used in the entire outer boundary.

In (3) $\mathcal{T}$ denotes the torus with the major radius R and the minor radius r, $r < R$:

$$
\mathcal{T} := \begin{cases} \left(R - (\boldsymbol{x}_1^2 + \boldsymbol{x}_2^2)^{\frac{1}{2}}\right)^2 + \boldsymbol{x}_3^2 & \leq r^2, \ \text{if } \boldsymbol{x} \in \mathbb{R}^3, \\ \left|R - (\boldsymbol{x}_1^2 + \boldsymbol{x}_2^2)^{\frac{1}{2}}\right| & \leq r, \ \text{if } \boldsymbol{x} \in \mathbb{R}^2, \\ \left|R - |x|\right| & \leq r, \ \text{if } x \in \mathbb{R}^1. \end{cases} \tag{4}
$$

We are interested to compute the total amount of hormone that has been emitted at a certain time, or we refer to it as a mass loss. This functional is computed as follows:

$$
M(t) = \int_{\mathcal{B}} \left(u_0(\boldsymbol{x}) - u(\boldsymbol{x}, t)\right) \mathrm{d}x, \tag{5}
$$

where u_0 is the initial concentration. It is obvious that at $t = 0$ the mass loss $M(0) = 0$, however this value eventually increases and approaches to $\int_{\mathcal{B}} u_0(\boldsymbol{x}) \, \mathrm{d}x$, i.e. an average of the initial data u_0, as $t \to \infty$.

Task

Your task is to *investigate* the physics and the numerical modeling associated with this problem, *design* the torus so that it fulfills certain conditions and requirements, and *communicate* the design and the qualities of the design in a convincing way. **Your results should support an expert committee in reaching a decision concerning the design of a product**. You have been assigned the 'torus approach'; other teams are investigating other ideas. The committee needs solid information to select the best possible way forward. *Design:* find ρ, R, and r such that, at time $t = \{T_0, T_1, T_2\}$ days, a total dose of $M = \{M_0, M_1, M_2\}$ mmol has been delivered.

Part A

1D simplification

As one first step towards solving the full problem presented in this project, we consider a simpler 1D model,

$$-\alpha u''(x) = f(x), \quad x \in (-1,1), \tag{6}$$

$$u(-1) = u(1) = 0. \tag{7}$$

where $f(x)$ is some forcing function. The first part of the project evolves around *implementing* a FEM-solver for this 1D problem in Matlab, *investigate numerical errors* and perform adaptive mesh refinement based on an *a posteriori* error estimation.

Problem A.1. Let u_h be a finite element approximation of the solution of the problem (6) - (7) on a mesh $-1 = x_0 < x_1 < \ldots < x_N = 1$. Let $h_i = x_i - x_{i-1}$ be a mesh-size, $I_i = (x_{i-1}, x_i)$ be the i-th element and $I = \bigcup_{i=1}^n I_i$ be the mesh. Derive the following a posteriori error estimate in the energy norm:

$$\|(u-u_h)'\|^2_{L^2(I)} \leq C \sum_{i=1}^{n} \eta_i^2, \tag{8}$$

where C is a constant and $\eta_i = h_i \|f + \alpha u_h''\|_{L^2(I_i)}$.

Problem A.2. A typical algorithm for adaptive finite element approximation can be written as follows:

```
1. Given a coarse mesh with $n$ elements and a small number TOL $> 0$.
2. while $\sum_{i=1}^n \eta_i^2 >$ TOL do:
3.     compute $u_h$.
4.     compute $\eta_i^2(u_h)$ in each element $I_i$, $i = 1, 2, \ldots, n$.
5.     refine the elements with biggest contribution to the error.
6. end while
```

Now, let us set $\rho = 10$, $R = 0.5$, $r = 0.3$, $\alpha = 0.01$ and let the right hand side be defined as in (3):

$$f(x) = \begin{cases} \rho, & \text{if } \big|R - |x|\big| \leq r, \\ 0, & \text{otherwise.} \end{cases}$$

First of all, approximate u_h'' using the discrete Laplacian $\Delta_h u_h$ defined in Laboration 1. Then, implement a numerical integration, let's say the Trapezoidal quadrature rule, to compute the error indicator η_i^2. Finally, implement an adaptive 1D finite element solver that solves the problem (6) - (7) using the above adaptive algorithm. Start with a uniform grid with 12 nodes and stop the refinement when the number of nodes exceeds 10^4 or TOL $< 10^{-3}$. In one figure, in different subplots plot the solution u_h, residual $R(u_h) = f + \alpha \Delta_h u_h$, error indicator $\eta(u_h)$ and the mesh-size distribution. The mesh-size distribution can be done by plotting `plot(x(2:end),[1./diff(x)])`, where `x` is the coordinates of the final mesh.

Hint: In Matlab you can implement the error indicator as follows: let's say you define a vector `eta2` to save η_i^2:

```
eta2 = zeros(N,1);
for i = 1:N
  h = x(i+1)-x(i);
  eta2(i) = ‘‘put your numerical integration’’;
end
```

Here `x` is a vector of node coordinates and `N` is the number of elements.

There are different possibilities for selecting the elements to be refined given the element error indicator η_i. For instance, a popular method is to refine element i if

$$\eta_i > \lambda \max_{i=1,2,\ldots,N} \eta_i, \tag{9}$$

where $0 \leq \lambda \leq 1$ is a parameter which must be chosen by the user. Note that $\lambda = 0$ gives a uniform refinement, while $\lambda = 1$ gives no refinement at all.

The refinement procedure consists of the insertion of new nodes at the center of the elements chosen for refinement. In other words, if we are refining $[x_i, x_{i+1}]$, then we replace it by $[x_i, (x_i + x_{i+1})/2] \cup [(x_i + x_{i+1})/2, x_{i+1}]$. This is easily implemented by adding the coordinate of the midpoint of all the elements to be refined to the vector of nodes `x`, and then to sort it. We list the code:

```
lambda = 0.9;
for i = 1:length(eta2)
  if eta2(i) > lambda*max(eta2)
    x = [x (x(i+1)+x(i))/2];
  end
end
x = sort(x);
```

The stopping criterion determines when the adaptive algorithm should stop. It can, for instance, take the form of a maximum bound on the number of nodes, the memory usage, the time of the computation, the total size of the residual, or a combination of these.

Part B

2D convergence analysis

Now, let us consider problem (1)-(3) in two space dimensions. The goal of this part is to learn the Matlab *implementation* of piece-wise linear finite element approximation in two space dimensions and study the *convergence analysis* of a finite element method.

The computational domain, i.e. the disc $\mathcal{B}$ in this case, can be triangulated in Matlab as follows: define the geometry using `circleg`, a build-in Matlab function, and triangulate it with respect to the maximum mesh-size $h_{\max}$:

```
geometry = @circleg;
hmax = 1/5;
[p,e,t] = initmesh(geometry,'hmax',hmax);
```

where `[p,e,t]` are point, edge and element data given by the triangulation in Matlab.

Provided that you have written functions to assemble the stiffness matrix and load vector, the homogeneous Dirichlet boundary condition is applied *strongly* (i.e. after assembling the linear system $\boldsymbol{A}\boldsymbol{\xi} = \boldsymbol{b}$) as follows:

```
I = eye(length(p));          % construct the identity matrix
A = AssembleA(...);          % stiffness matrix
b = Assembleb(...);          % load vector
A(e(1,:),:) = I(e(1,:),:);   % replace the rows corresponding
                             % to the boundary nodes by
                             % corresponding rows of I
b(e(1,:)) = 0;               % put the boundary value into the RHS
```

Furthermore, the energy norm of the error

$$\|u - u_h\|_E = \Big(\int_{\mathcal{B}} (\nabla u - \nabla u_h) \cdot (\nabla u - \nabla u_h) \, \mathrm{d}\boldsymbol{x}\Big)^{\frac{1}{2}}$$

can be computed in Matlab as `EnE=sqrt(err'*A*err);`, where `A` is the stiffness matrix and `err=u-u_h` is the error.

Problem B.1. Let us consider the following two dimensional stationary problem, which is obtained by setting $\partial_t u = 0$ and $\alpha = 1$:

$$-\Delta u(\boldsymbol{x}) = f(\boldsymbol{x}), \qquad \boldsymbol{x} \in \mathcal{B}, \tag{10}$$

$$u(\boldsymbol{x}) = u_{\text{exact}}(\boldsymbol{x}), \quad \boldsymbol{x} \in \partial\mathcal{B}. \tag{11}$$

The right hand side is set as $f(\boldsymbol{x}) = 8\pi^2 \sin(2\pi \boldsymbol{x}_1) \sin(2\pi \boldsymbol{x}_2)$, which gives the exact solution $u_{\text{exact}}(\boldsymbol{x}) = \sin(2\pi \boldsymbol{x}_1) \sin(2\pi \boldsymbol{x}_2)$.

Write down a Galerkin finite element method for this problem using piecewise linear basis functions in space, then implement it in Matlab. Compute the energy norm in a sequence of meshes obtained by setting: $h_{\max} = \{\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}\}$. Then, find *the convergence rate* of the Galerkin approximation in the energy norm numerically, let us denote it by γ. Plot $h_{\max}$ versus the energy norm of the error and $h_{\max}$ versus $h_{\max}^{\gamma}$ in the same figure using the `loglog`-plot command in Matlab. Explain how you did to determine the value of γ and report the value of it.

Plot the solutions that are obtained using the coarsest and the finest meshes.

Problem B.2. Formulate a Galerkin finite element method using continuous piecewise linear approximation of (1)–(3). Discretize the time derivative using the Crank-Nicolson method and write down the corresponding linear algebra matrices and vectors.

Problem B.3. Lets say the source term is zero, i.e. $f(\boldsymbol{x}) = 0$. Now, implement the variational problem you derived in Problem B.2 in Matlab. Implement a numerical integration in 2D to compute the mass loss (5). For example, for computing the integral of a general function $g(\boldsymbol{x})$, one can use the Trapezoidal rule in an element K:

$$\int_K g(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \approx \frac{|K|}{3} \sum_{i=1}^{3} g(N_i),$$

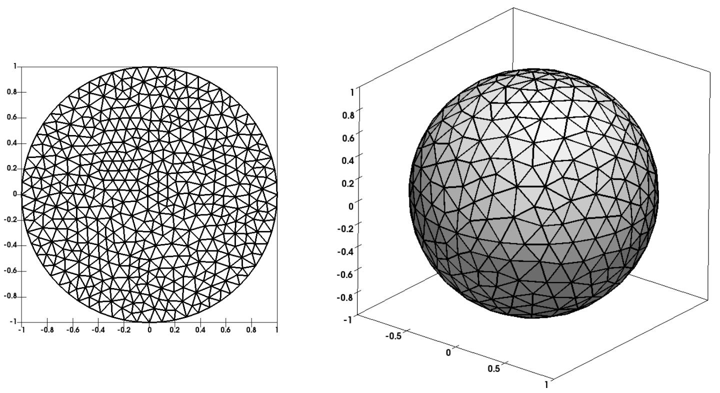

Figure 2: 2D and 3D mesh of a unit ball $\mathcal{B}$

where N_i, $i = 1, 2, 3$, is the nodal points of the cell K.

Set the control parameters to $\rho = 10, R = 0.5, r = 0.3, T = 30$ and run your code with two different mesh resolutions: $h_{\max} = \frac{1}{5}$ and $h_{\max} = \frac{1}{20}$. Plot the amount of emitted hormone versus time in one figure for both simulations. Also, plot your solution at the initial time and the final time for the fine mesh.

Problem B.4. Now, let us consider the heat equation with a reaction term:

$$\partial_t u(\boldsymbol{x}, t) - \alpha \Delta u(\boldsymbol{x}, t) = \beta(1 - \gamma u)u, \qquad (\boldsymbol{x}, t) \in \mathcal{B} \times (0, T], \tag{12}$$

where β and γ are reaction coefficients yet to be defined. It is an interesting model since the concentration will not diffuse out completely in the body, instead, it is continuously added to the system via the reaction term. Equation (12) is usually referred to as the Fisher's equation in the literature and is widely used in the population dynamics and biology. Set $\beta = 1$ and $\gamma = 0.2$ and repeat the task in Problem B.3. What do you observe? How about $\beta = 0.2$ and $\gamma = 0.5$? Motivate your answer.

The right hand side of (12), $S(u) = \beta(1 - \gamma u)u$ is nonlinear. In your implementation, you can treat this term explicitly, *i.e.* compute $S(u)$ using the solution from previous time step . Also, it is useful to take a linear interpolant of $S(u)$ in the implementation. That is $S(u) \approx \pi_h S = \sum_{j=1}^{\mathcal{N}} S_j \varphi_j$, where $\mathcal{N}$ is the total number of nodes.

Part C

FEniCS implementation

This part of the project is considered to be the final and main part, where we consider the full 3D model of the problem (1)–(3) with zero source term $f(\boldsymbol{x}) = 0$. We will solve the problem using the open source finite element solver FEniCS. FEniCS project is a collection of open-source finite element software for solving partial differential equations in any space dimensions and polynomial degrees.

Laboration 3 can be a good starting point to implement a time-dependent heat equation in FEniCS. Below some useful tools of FEniCS are discussed that could be useful in the implementation. The structure of your solver could look like the following:

```python
from dolfin import *

# Create mesh and define function space
mesh = Mesh("circle.xml")
Q = FunctionSpace(mesh, "CG", 1)

# Define parameters:
T = ...
h = mesh.hmin()
dt = h
alpha = 0.01

# Create subdomain for Dirichlet boundary
class DirichletBoundary(SubDomain):
    def inside(self, x, on_boundary):
        return on_boundary

# Set up boundary condition
g = Constant(0.0)
bc = DirichletBC(Q, g, DirichletBoundary())

# Define initial condition
indata = Expression("...")
u0 = Function(Q)
u0 = interpolate(indata, Q)

# Create bilinear and linear forms
a = ...
L = ...

# Set an output file
...

# Set initial condition
...

# Time-stepping
while t < T:

   # assign u0
   u0.assign(u)
   ...
   ...
```

Problem C.1. Set $\rho = 10, R = 0.5, r = 0.2, T = 20$ and implement a FEniCS solver to solve the problem (1)–(3) in a given 2D mesh `circle.xml` and 3D mesh `sphere.xml`, see Figure 2. Plot the solution at initial time $t = 0$ and the final time $t = T$ and motivate your plots.

Problem C.2. In your FEniCS implementation, write a code that computes the mass loss $M(t)$ at every time level. Then save the mass loss together with it's corresponding time level into a file. Plot the mass loss as a function of time for three different sets of control parameters in one figure:

- $\rho = 10, R = 0.5, r = 0.2, T = 50,$

- $\rho = 20, R = 0.5, r = 0.2, T = 50$,
- $\rho = 40, R = 0.5, r = 0.2, T = 50$.

Discuss your findings in terms of mass loss: for example, which parameter set gives the fastest mass loss and why?

Hint: The integral (5) or in general any functionals in FEniCS can be implemented as follows:

```
# Copy initial data
u_initial = Function(Q)
u_initial = interpolate(indata, Q)

# Define an integral functional
M = (u_initial - u) * dx

# compute the functional
mass = assemble(M)
```

Now, you just need to call this assembly inside your time loop and save it to a file.

Problem C.3. Finally, we reached to the point of designing the medical torus. We have the following constraints: at time levels $t = \{5, 7, 30\}$ days the total doses $M = \{10, 15, 30\}$ mmol should have been delivered correspondingly. Let us set $T = 50$ days and find the optimal parameters ρ, R, r such that the above doses are delivered at the given days. Some additional constrains, such as $R < 1$ and $r < R$ should be guaranteed.

Hint: To find the optimal parameters you can consider to minimize the following functional:

$$F(\rho, R, r) = \sum_{i=0}^{2} \big(M(T_i) - M_i\big)^2,$$

where the mass loss M is in fact a function of (ρ, R, r). You are welcome to use some existing functionality of the Python libraries to minimize $F(\rho, R, r)$. For example, you can use the `minimize` function from the `scipy` library:

```
import scipy.optimize as optimize
from scipy.optimize import minimize

def func(data):
    # solve the heat equation and compute the mass loss M(t)
    # get M(5), M(10), M(30)
    # compute the minimization functional f(rho, R, r)
    F = (M(5) - 10)**2 + ...

    return F

# set initial guess (rho, R, r)
data = [20, 0.5, 0.1]
res = minimize(func, data, method='nelder-mead', options={'xtol':
   1e-3, 'disp': True})
print res
```