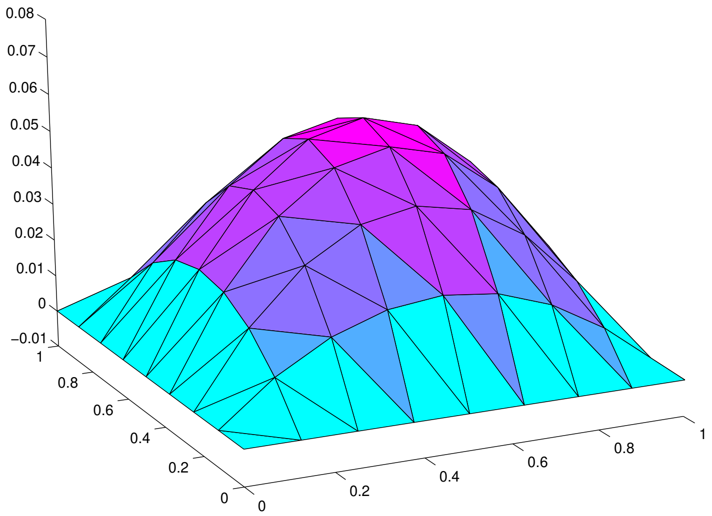
# The heat equation

Computer Session E2




## Background

Today, we will solve the heat equation,

$$
\begin{aligned}
\dot{u} - \nabla \cdot (a \nabla u) &= f \quad \text{in } \Omega \times (0, T], \\
-a \partial_n u &= \gamma(u - g_D) + g_N \quad \text{on } \Gamma \times (0, T], \\
u(\cdot, 0) &= u_0 \quad \text{in } \Omega.
\end{aligned}
\tag{1}
$$

The dG(0) formulation of the heat equation is given by

$$\begin{aligned}\int_\Omega U_n v\,dx + k\int_\Omega a\nabla U_n\cdot\nabla v\,dx + k\int_\Gamma \gamma U_n v\,ds =\\ k\int_\Omega f_n v\,dx + k\int_\Gamma(\gamma g_D - g_N)v\,ds + \int_\Omega U_{n-1}v\,dx \quad \forall v\in V_h,\end{aligned} \tag{2}$$

where $k = t_n - t_{n-1}$ denotes the size of the time step, $U_n$ denotes the value at $t = t_n$, and $U_{n-1}$ denotes the value at $t = t_{n-1}$. Note that also $\gamma$, $g_D$, and $g_N$ should be evaluated at $t = t_n$. The dG(0) method is also known as the backward Euler method.

Note that this computer session combines what you already know about time-stepping (from the C sessions), and what you know about solving stationary partial differential equations (from the D sessions and session E1).

Before today's computer session, make sure that you understand and can answer the following questions.

**Question 1** Derive the variational formulation (2) from the heat equation (1).

**Question 2** How does the corresponding variational formulation look for the cG(1) method? This method is also known as the Crank-Nicolson method.

**Question 3** Verify that

$$u(x) = x_1(1-x_1)x_2(1-x_2)\sin t \tag{3}$$

is a solution of the heat equation on the unit square $\Omega = (0,1)\times(0,1)$ with $a = 1$ and right-hand side

$$f(x,t) = x_1(1-x_1)x_2(1-x_2)\cos t + 2(x_1(1-x_1) + x_2(1-x_2))\sin t. \tag{4}$$

What are the boundary conditions?

# Preparations

Create a new directory called `e2` and the two subdirectories `problem1` and `problem2`. Then download the following files to each of the subdirectories:

- `AssembleMatrix.m`,
- `AssembleVector.m`,
- `Heat.m`,
- the files in the directory `mesh`.

These files are available on the web page of this session under *Programs and templates*.

# Problems

## Problem 1

### Preparation

Go to the directory `problem1`. Create a file called `HeatSolver.m` and write your program in this file. You also have to edit the file `Heat.m`, where you specify the variational formulation.

### Problem

Solve the heat equation on the unit square (`square.m`) with homogeneous Dirichlet boundary conditions, $a = 1$, initial condition $u_0 = 0$, $T = \pi/2$, and the right-hand side given by

$$f(x,t) = x_1(1-x_1)x_2(1-x_2)\cos t + 2(x_1(1-x_1) + x_2(1-x_2))\sin t, \quad (5)$$

using a time step of size $k = T/10$.

Compare your computed solution $U(x,t)$ with the exact solution $u(x,t)$. How large is the error in the maximum norm at the final time? The maximum norm of the error is given by

$$\|e(T)\|_\infty = \|U(T) - u(T)\|_\infty = \max_{x\in\Omega} |U(x,T) - u(x,T)|. \quad (6)$$

How large is the error when you refine the mesh (`square_refined.m`)? How large is the error when you divide the time step with a factor 2?

To help you solve the problem, we will guide you through the implementation, step by step. Don't look at these instructions if you want to solve the problem in your own way. (You could of course glance occasionally at the instructions.)

1. We start by implementing the time-stepping loop. Write a loop that takes 10 time steps and updates a variable called `time`, starting at `time = 0`. Note that you should name the time variable `time` instead of `t`, since `t` is used for the mesh.

2. Next, we add two vectors `U0` and `U1`, representing the solution at time $t = t_{n-1}$ and time $t = t_n$ respectively. Initialize both vectors to zero (of the correct size), and update the vector `U0` to be equal to `U1` at the beginning of every time step. Remember to load the mesh at the beginning of the program.

3. We now need to assemble the matrix `A` for the left-hand side and the vector `b` for the right-hand side of the variational formulation. The matrix is not time-dependent, so it is enough if we assemble it one time, before the time-stepping begins. Since the vector depends on time ($f$ is time-dependent), we need to assemble it in each step of the loop.

   Note that the old value $U_{n-1}$ (given by `U0`) is present in the right-hand side of the variational formulation (2). We thus need to supply this value to the assembler. This is done by using the extra argument `W` of the program `AssembleVector.m`. In your program `Heat.m`, you then obtain the value of $U_{n-1}$ in the variable `w`.

4. Now solve the linear system to compute the new value of `U1` in each step of the loop. Compute the exact solution at final time and compute the error.

*Check your answer:* You should obtain the following errors in the maximum norm for the two different meshes and the different values of $k$:

| | $k = T/10$ | $k = T/20$ |
|---|---|---|
| `grid.m` | 0.000998 | 0.000876 |
| `grid_refined.m` | 0.000445 | 0.000318 |

## Problem 2

### Preparation

Go to the directory `problem2` and copy the two files `HeatSolver.m` and `Heat.m` from the directory `problem1`.

### Problem

Experiment with different boundary conditions and different source terms $f(x,t)$. Plot the solution in each time step to see the time evolution of the solution. Either use the command `pause` each time you plot the solution (otherwise you will not see the solution) or use the commands `movie` and `getframe` to create a movie of your solution.

# Hints

## Problem 1

1. The time-stepping loop could look something like

```
T = pi/2;
k = T/10;
time = 0;

for n = 1:10

  time = time + k;
```

```
end
```

2. Add `U0 = zeros(size(p,2),1)` and `U0 = zeros(size(p,2),1)` at the beginning of the program, and then add `U0 = U1` in every step of the loop.

3. The variational formulation that you need to put in the file `Heat.m` is

   ```
   u*v*dx + k*(du'*dv*dx + g(x,d,t)*u*v*ds)
   ```

   for the left-hand side and

   ```
   k*(f(x,d,t)*v*dx + (g(x,d,t)*gd(x,d,t) - gn(x,d,t))*v*ds)
   + w*v*dx
   ```

   for the right-hand side.

### Problem 2

Use the commands `help movie` or `help getframe` to find out how to create a movie.

## Solutions

Make sure that you really try to solve each problem before looking at the solutions. Have you really tried to solve the problem or should you try again before looking at the solution?

The solutions are available on the web page of this session under *Solutions to problems*.

## About

This Computer Session is part of the Body and Soul educational program. More information can be found at `http://www.phi.chalmers.se/bodysoul/`

This Computer session is written by Anders Logg.