Laboratory exercise no. 2: Heat transfer simulation

Piotr Gawryś < *pgawrys2@gmail.com* >

April 23, 2018

1 Introduction

The goal of this laboratory is to implement simulation of heat transfer for three plates each made of different metal:

1. Alumina.
2. Cooper.
3. Stainless Steel.

During the course of the laboratory we will also implement initial and boundary conditions, test numerical stability and calculate spacial and temporal temperature distribution in the plate for boundary conditions of type 1 and 2.

2 Implementation

First, we will write function containing relevant physical parameters of each metal:

Listing 1: Choosing material

```
% can be Alumina, Cooper or Stainless Steel
function [K, Cw, Rho] = choose_material(material)
    if strcmp(material, 'Alumina')
        K=237;
```

```
        Cw=900;
        Rho=2700;
    elseif strcmp(material, 'Cooper')
        K=401;
        Cw=380;
        Rho=8920;
    else
        K=58;
        Cw=450;
        Rho=7860;
    end;
end
```

This will allow us to reuse it instead of repeating it every time.

The metal plate has heater attached in the middle. We assume that both heater and the plate have square shape. The side of the heater is assumed to be *0.05 m* and the side of the plate is *0.2 m*.

2.1 Equation

We will need to use equation describing our mathematical model:

Listing 2: Equation describing mathematical model

```
function T = equation(T, Nt, Nx, Ny, dx, dy, dt, K, Cw,
    Rho)
    for t = 1:Nt - 1
        for i = 2:Nx - 1
            for j = 2:Ny - 1
                if ~((i >= (0.075 / dx) &&  i <= (Nx -
                   0.075 / dx)) && (j >= (0.075 / dy)
                   &&  j <= (Nx - 0.075 / dy)))
                    T(i, j, t + 1) = T(i, j, t) ...
                        + (K * dt / (Cw * Rho * dx.^2))
                            * (T(i + 1, j, t) - 2*T(i,
                           j, t) + T(i - 1, j, t)) ...
                        + (K * dt / (Cw * Rho * dy.^2))
                            * (T(i, j + 1, t) - 2*T(i,
```

```
                                    j, t) + T(i, j - 1, t));
                    end;
                end;
            end;

            [XX, YY] = meshgrid(dx:dx:0.2, dy:dy:0.2);
            surf(XX, YY, T(:, :, t));
            title(['Simulation time = ' num2str(t * dt) ' (
                s)']);
            xlabel('x (m)');
            ylabel('y (m)');
            zlabel('Temperature (degC)');

            drawnow;
            pause(0.1);
        end;
    end
```

The reason for using *0.075* in the equation is that it is distance between the side of the plate and the heater in the straight line and before applying the equation we need to verify our current coordinates.

This function will also continuously plot simulation.

2.2 Boundary condition type 1

We consider two different boundary conditions (**type 1** and **type 2**) which also can be described with appropriate functions.

Regarding type 1 – area having contact with the heater has a constant temperature of 80°C while the edge of plate has constant temperature of 10°C. The rest is calculated using equation implemented in the previous subsection.

Listing 3: Boundary condition of type 1

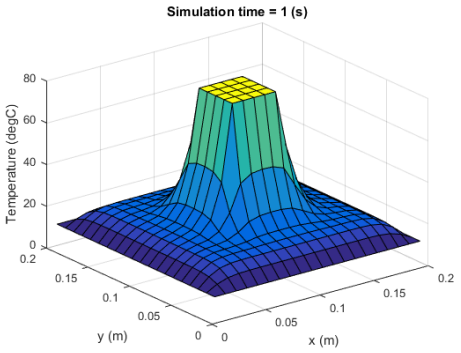
```
function T = bc_type1(Nt, Nx, Ny, dx, dy, dt, K, Cw,
    Rho)
    T = zeros(Nx ,Ny, Nt);
    T(:, : ,1) = 20;
    T(1, :, :) = 10;
    T(Nx, :, :) = 10;
```

```
    T(:, 1, :) = 10;
    T(:, Ny, :) = 10;
    T((0.075 / dx):(Nx - 0.075 / dx), (0.075 / dy):(Ny
        - 0.075 / dy), :) = 80;

    T = equation(T, Nt, Nx, Ny, dx, dy, dt, K, Cw, Rho)
        ;
end
```

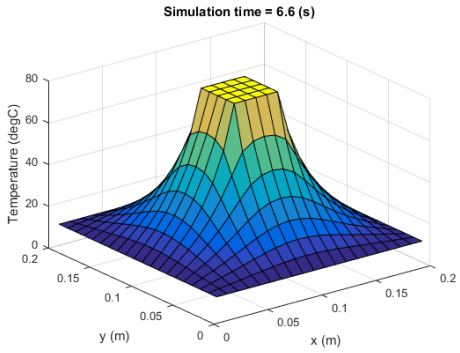
We can pull everything together and add initial parameters to run this simulation:

Listing 4: Heat transfer for boundary condition of type 1 for Alumina

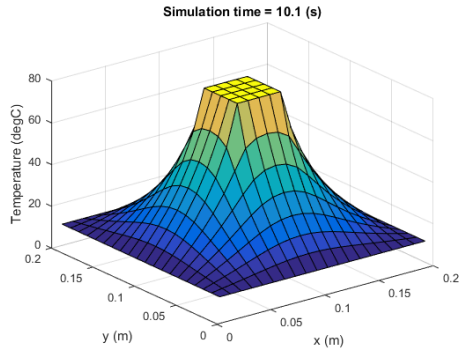
```
% preparing workspace
close all; clear all; clc;

% setting up parameters

% metal plate
A = 0.2;
% heater
B = 0.05;
% distance between computational nodes in x direction
dx = 0.01;
% distance between computational nodes in y direction
dy = 0.01;
% time step
dt = 0.1;
% number of time steps
Nt = 1000;
% number of computational nodes in x direction
Nx = A/dx;
% number of computational nodes in y direction
Ny = A/dy;

[K, Cw, Rho] = choose_material('Alumina');

T = bc_type1(Nt, Nx, Ny, dx, dy, dt, K, Cw, Rho);
```



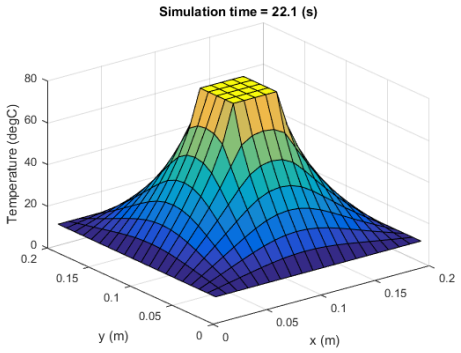
Simulation time: 1 (s)



Simulation time: 6.6 (s)



Simulation time: 10.1 (s)



Simulation time: 22.1 (s)

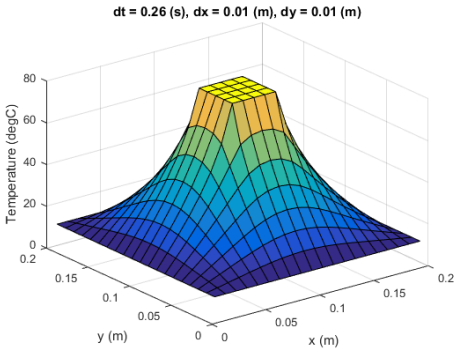
Temperature distribution in heat transfer simulation with boundary of type 1 for Alumina

The resulting plot shows that it starts with heat only around the heater but in just few seconds the area around it becomes warmer up to specific point after it remains stable regardless of time.

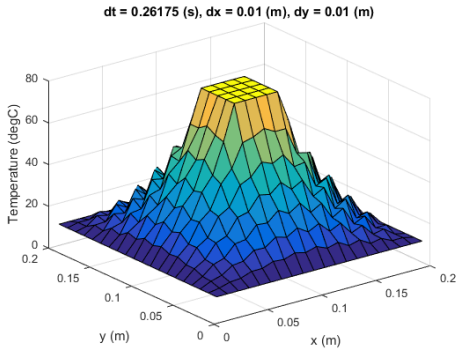
2.3 Numerical stability

Numerical stability usually means that for small disruptions in data the results will also notice small disruptions which is generally desired property as opposed to big changes.

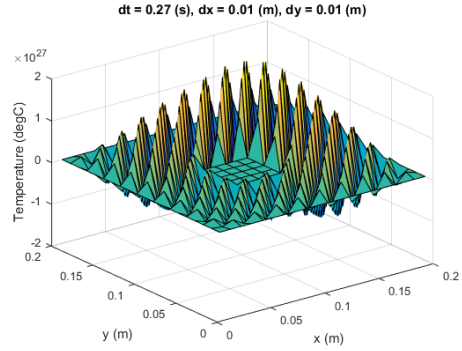
We can find out when our model becomes unstable for Alumina using different values of time steps and spatial resolutions which means changing dt, dx and dy values.



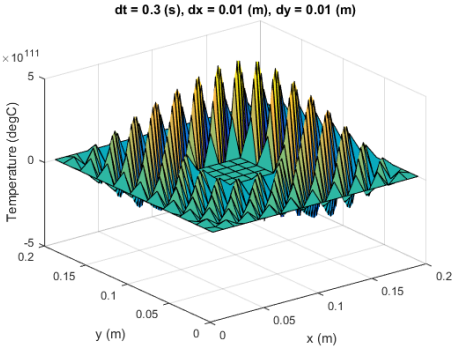
Time step: 0.26 (s)



Time step: 0.26175 (s)

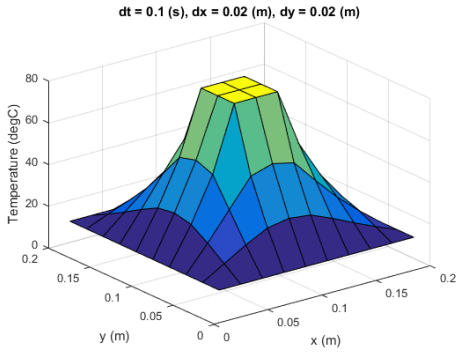


Time step: 0.27 (s)

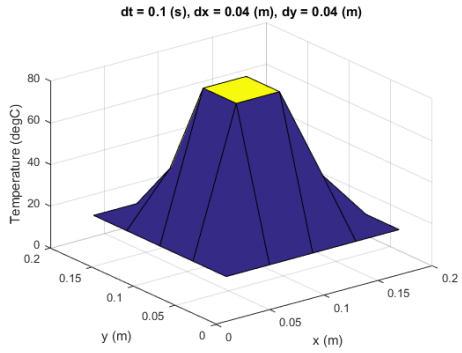


Time step: 0.3 (s)

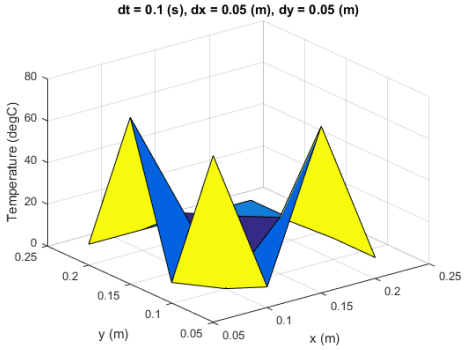
Stability tests for different time steps. We can observe that model is starting to lose stability for time steps higher than 0.26 s.



Spatial steps: 0.02 (m)



Spatial steps: 0.04 (m))



Spatial steps: 0.05 (m)

Stability tests for different spatial resolutions. The model is very quickly losing details but it is to be expected and for dx and dy = 0.04 (m) shape stays about the same. Stability loss occurs somewhere between 0.04 (m) and 0.05 (m).

2.4 Boundary condition type 2

In this case we assume a constant power of the heater equal to 100W during the first 10s of the simulation. It is switched off later.

Implementation is very similar to previous boundary but we add part responsible for turning heater off.

Listing 5: Heat transfer for boundary condition of type 2

```
function T = bc_type2(Nt, Nx, Ny, dx, dy, dt, K, Cw,
    Rho, h, P, A, B)
    T = zeros(Nx, Ny, Nt);
```

```
T(:, :, 1) = 20;

for t=1:Nt-1
    for i=2:Nx-1
        for j=2:Ny-1
            if (~((i >= (0.075 / dx) &&  i <= (Nx -
                0.075 / dx)) && (j >= (0.075 / dy)
               &&  j <= (Nx - 0.075 / dy))) || t >=
                (1 / dt))
                T(i, j, t + 1) = T(i, j, t) ...
                    + (K * dt / (Cw * Rho * dx.^2))
                        * (T(i + 1, j, t) - 2*T(i,
                        j, t) + T(i - 1, j, t)) ...
                    + (K * dt / (Cw * Rho * dy.^2))
                        * (T(i, j + 1, t) - 2*T(i,
                        j, t) + T(i, j - 1, t));
            else
                if t < ( 1 / dt)
                    T(i, j, t + 1) = T(i, j, t) ...
                        + (K * dt / (Cw * Rho * dx
                            .^2)) * (T(i + 1, j, t)
                            - 2*T(i, j, t) + T(i -
                            1, j, t)) ...
                        + (K * dt / (Cw * Rho * dy
                            .^2)) * (T(i, j + 1, t)
                            - 2*T(i, j, t) + T(i, j
                            - 1, t)) ...
                        + (P * dt) / (Cw * B.^2 * h
                            * Rho);
                end;
            end;
        end;
    end;
    T(1, 1, t + 1) = T(2, 2, t + 1);
    T(Nx, Ny, t + 1) = T(Nx - 1, Ny - 1, t + 1);
    T(1, Ny, t + 1) = T(2, Ny - 1, t + 1);
    T(Nx, 1, t + 1) = T(Nx - 1, 2, t + 1);
    T(1, :, t + 1) = T(2, :, t + 1);
```

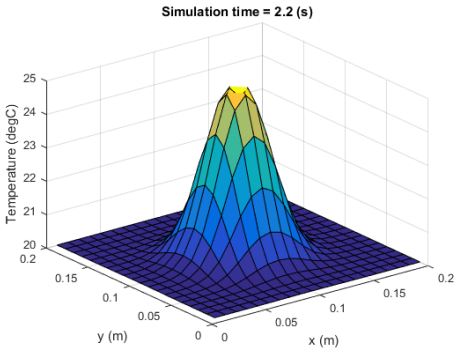
```
        T(Nx, :, t + 1) = T(Nx - 1, :, t + 1);
        T(:, 1, t + 1) = T(:, 2 , t + 1);
        T(:, Ny, t + 1) = T(:, Ny - 1, t + 1);

        [XX, YY] = meshgrid(dx:dx:A, dy:dy:A);
        surf(XX, YY, T(:, :, t));
        title(['Simulation time = ' num2str(t * dt) ' (
           s)']);
        xlabel('x (m)');
        ylabel('y (m)');
        zlabel('Temperature (degC)');
        axis([0 A 0 A 20 25]);
        drawnow;
        pause(0.1);
    end;
end
```

We can run it the same way we did last time but we need two additional parameters:

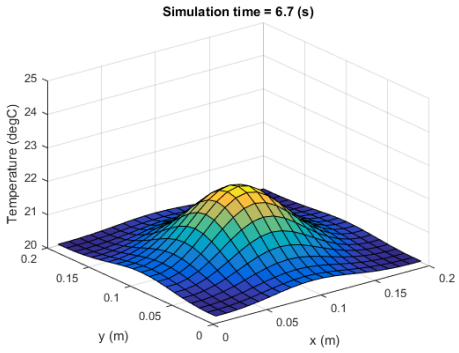
Listing 6: Additional parameters for boundary condition of type 2

```
% plate thickness
h = 0.002;
% heater power
P = 100;
```

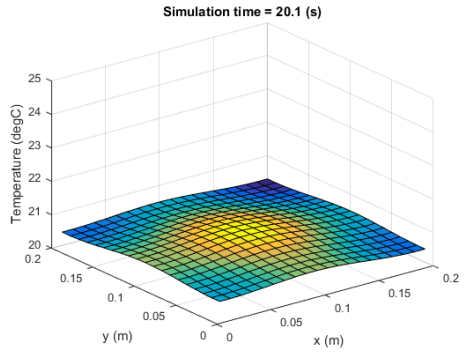
We will compare results for Alumina and Cooper.



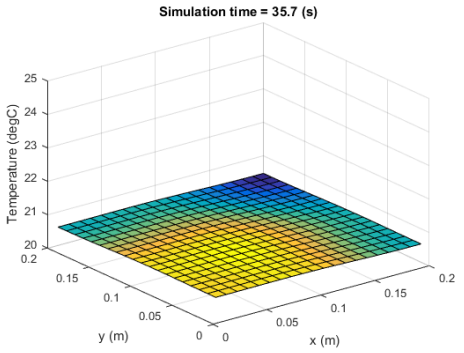
Simulation time: 2.2 (s)



Simulation time: 6.7 (s)

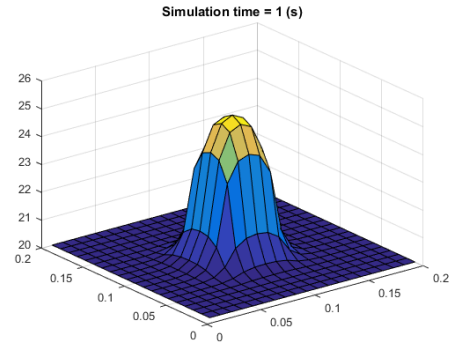


Simulation time: 20.1 (s)

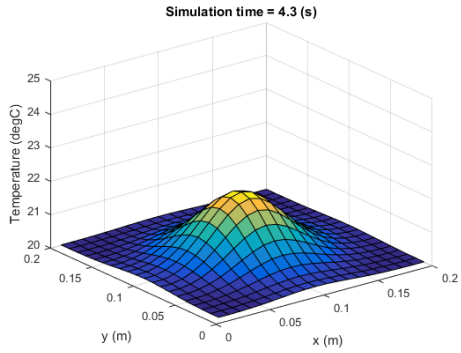


Simulation time: 35.7 (s)

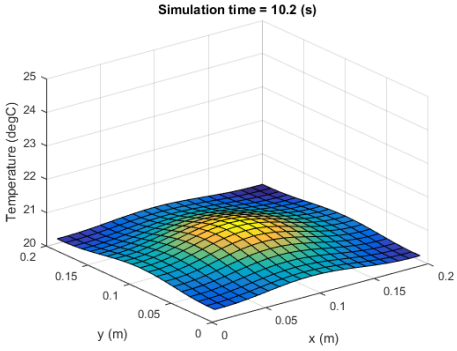
Heat transfer for boundary condition of type 2 using Alumina plate



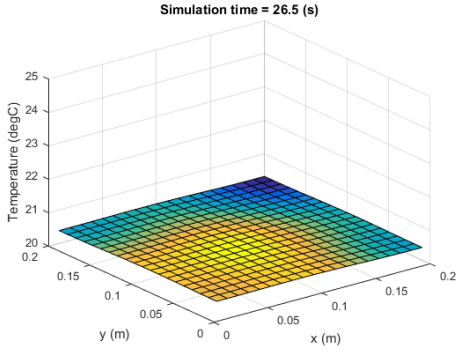
Simulation time: 1 (s)



Simulation time: 4.3 (s)



Simulation time: 10.2 (s)



Simulation time: 26.5 (s)

Heat transfer for boundary condition of type 2 using Cooper plate

In both cases it starts with rapid increase in temperature around heater area and then it dissipates (more about it in the later subsection) but the process is significantly faster for Cooper. The plate reaches similar steady state after around 26.5 seconds in comparison to 35.7 seconds in case of Alumina. This makes sense because Cooper's thermal conductivity is much bigger than Alumina's.

2.5 The comparison of final temperature increment for 2-nd type of boundary condition with theoretical heat balance calculation.

We can compare *steady state* which can be observed at the last sub figures in the previous subsection to theoretical heat balance. To calculate steady

state we can either check heat changes between time steps or do it empirically. Either way it occurs after about 35.7 seconds for Alumina.

If we saved our current state, we can simply calculate the value:

Listing 7: Heat balance for simulation

```
% we take maximum from T and subtract starting
    temperature
max(max(T(:, :, t))) - 20
```

In our case it results in around 0.59°C.

Theoretical heat balance can be calculated from the equation:

$$\Delta T = \frac{P \cdot t}{c_w \cdot A^2 \cdot h \cdot \rho} \tag{1}$$

Which is around 0.514°K for Alumina with A = 0.2 and h = 0.002.

Results are not quite the same but pretty similar.

2.6 Heat dissipation

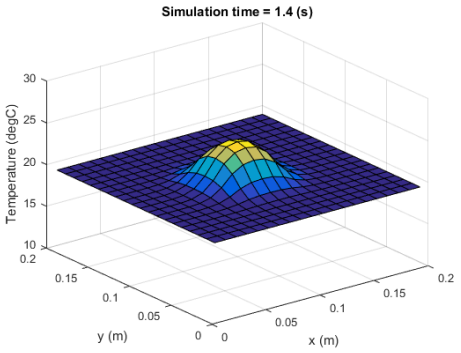
It occurs when an object that is hotter than other objects is placed in an environment where the heat is transferred to colder objects and the environment itself. To put it simply – it refers to heat loss from a hotter body to colder one. Implementation can be found below:

Listing 8: Heat dissipation for Alumina

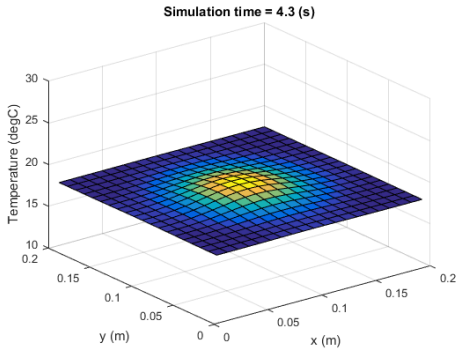
```
% preparing workspace
close all; clear all; clc;

% setting up parameters

% metal plate
A = 0.2;
% heater
B = 0.05;
% distance between computational nodes in x direction
dx = 0.01;
% distance between computational nodes in y direction
dy = 0.01;
```

```
% time step
dt = 0.1;
% number of time steps
Nt = 1000;
% number of computational nodes in x direction
Nx = A/dx;
% number of computational nodes in y direction
Ny = A/dy;

h = 0.002;
P = 100;
% temp outside
Tout = 10;

[K, Cw, Rho] = choose_material('Alumina');

T = zeros(Nx, Ny, Nt);
T(:, :, 1) = 20;

for t = 1:Nt - 1
    for i = 2:Nx - 1
        for j = 2:Ny - 1
            Tdiff = -5.678 * 10^-8 * T(i, j, t) * T(i,
               j, t) * T(i, j, t) * T(i, j, t) * (T(i,
               j, t) - Tout);
            if (~((i >= (0.075 / dx) &&  i <= (Nx -
               0.075 / dx)) && (j >= (0.075 / dy) &&  j
                <= (Nx - 0.075 / dy))) || t >= (1/dt))
                T(i, j, t + 1) = T(i, j, t) ...
                    + (K * dt / (Cw * Rho * dx.^2)) * (
                       T(i + 1, j, t) - 2*T(i, j, t) +
                       T(i - 1, j, t)) ...
                    + (K * dt / (Cw * Rho * dy.^2)) * (
                       T(i, j + 1, t) - 2*T(i, j, t) +
                       T(i, j - 1, t)) + Tdiff;
            else
                if t < ( 1 / dt)
                    T(i, j, t + 1) = T(i, j, t) ...
```

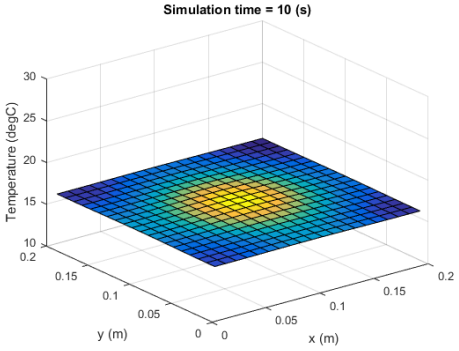
```
                        + (K * dt / (Cw * Rho * dx.^2)) *
                            (T(i + 1, j, t) - 2*T(i, j, t
                            ) + T(i - 1, j, t)) ...
                        + (K * dt / (Cw * Rho * dy.^2)) *
                            (T(i, j + 1, t) - 2*T(i, j, t
                            ) + T(i, j - 1, t)) ...
                        + Tdiff + (P * dt) / (Cw * B.^2 *
                            h * Rho);
                end;
            end;
        end;
    end;
  T(1, 1, t + 1) = T(2, 2, t + 1);
  T(Nx, Ny, t + 1) = T(Nx - 1, Ny - 1, t + 1);
  T(1, Ny, t + 1) = T(2, Ny - 1, t + 1);
  T(Nx, 1, t + 1) = T(Nx - 1, 2, t + 1);
  T(1, :, t + 1) = T(2, :, t + 1);
  T(Nx, :, t + 1) = T(Nx - 1, :, t + 1);
  T(:, 1, t + 1) = T(:, 2 , t + 1);
  T(:, Ny, t + 1) = T(:, Ny - 1, t + 1);
end;

[XX, YY] = meshgrid(dx:dx:A, dy:dy:A);

for tt=1:t+1
  surf(XX, YY, T(:, :, tt));
  title(['Simulation time = ' num2str(tt * dt) ' (s)'])
      ;
  xlabel('x (m)');
  ylabel('y (m)');
  zlabel('Temperature (degC)');
  axis([0 A 0 A Tout 30]);
  drawnow;
  pause(0.1);
end;
```



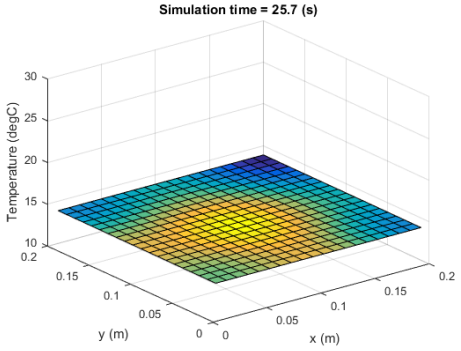
Simulation time: 1.4 (s)



Simulation time: 4.3 (s)



Simulation time: 10 (s)



Simulation time: 25.7 (s)

Heat dissipation for Alumina

The heater starts surrounded by cold objects transferring its heat with time. We can see that overall level of heat in the environment (including heater) is decreasing which complies to the second law of thermodynamics.

3 Conclusion

During this laboratory we have successfully implemented simulation of heat transfer for several metals using two different boundary conditions.

We have visualized temporal evolution of spatial temperature distribution, tested numerical stability, compared some of the results and even introduced heat dissipation to our model.

We also concluded that when implementing heat transfer simulation this

way it is necessary to choose correct time and spatial steps to avoid losing numerical stability.