# Lecture: Solving Transient Heat Equation

**Video description of the live script is available here.**

**Table of Contents**



Switch working directory to folder containing this file.

```
editor = matlab.desktop.editor.getActive;
% Check if there is an open file
if ~isempty(editor)
    % Get the full path of the current open file
    currentFilePath = editor.Filename;

    % Extract the folder path from the full file path
    [currentFileFolder, ~, ~] = fileparts(currentFilePath);

    % Get the current working directory
    currentDir = pwd;

    % Check if the working directory is different from the file's folder
    if ~strcmp(currentDir, currentFileFolder)
        % Change the working directory to the file's folder
        cd(currentFileFolder);
        fprintf('Changed working directory to: %s\n', currentFileFolder);
    else
```

```
        fprintf('Already in the correct directory: %s\n', currentDir);
    end
else
    fprintf('No file is currently open in the MATLAB Editor.\n');
end
```

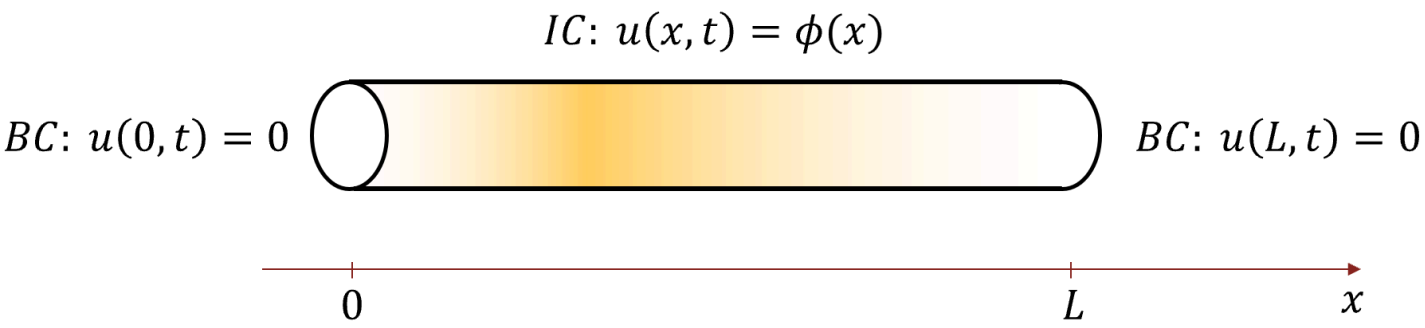
```
Already in the correct directory: C:\Users\ppanares\OneDrive - MathWorks\CSE\EventsMY\Projects\2021-Solving Heat
```

## ********** *PART I : PROBLEM* **********

## *Story*

### *Problem*

- Analyze **heat diffusion along an thin rod** and compute **absolute temperature** $u(x,t)$.
- **"Transient"** problem**:** we are given **initial temperature** and **hold boundaries at fixed 0°K.**
- Input data: **material, length** $L>0$**, initial temperature** $u(x,0)=\phi(x)$at time $t=0$
- Assumptions: 1-dim, finite positive length, isotropic material, no heat exchange with external environment

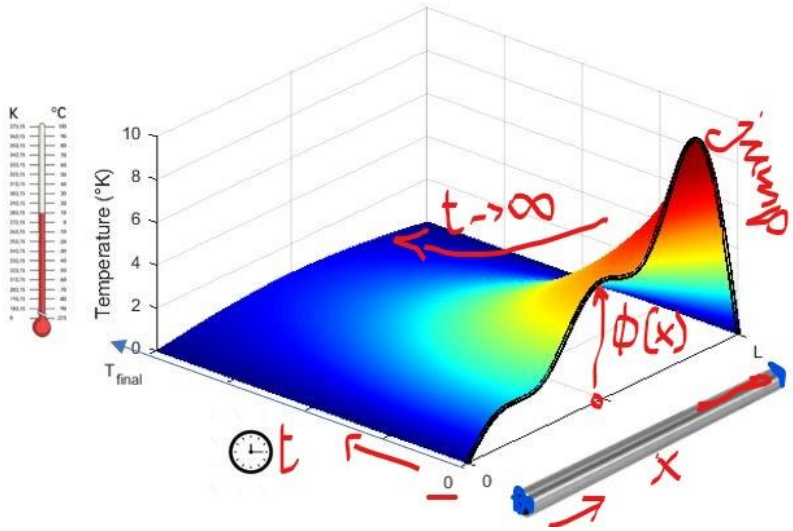



### Challenges

- Understand the Heat Equation model and get deeper insight of the **heat solution structure**
- Understand different **thermal behaviors:** materials can either **store the heat** (and heat up quickly) or **transfer the heat** (don't heat up or heat up very slowly)
- Explore the impact of **different materials**, and **different initial conditions, even with wild jumps**

### Solution

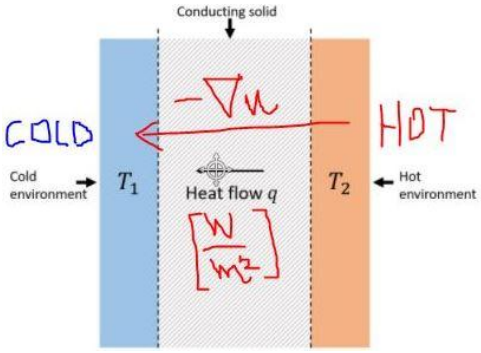
- Heat irreversibly flows from hot to cold until reaching a stationary equilibrium (zero in this case).
- **Discover the heat structure** by using analytical methods, eg separation of variables and Fourier's analysis. **Transient temperature exponentially decays to zero, the time decay depending on the material (thermal diffusivity).** Any initial "jump" is smoothed out.

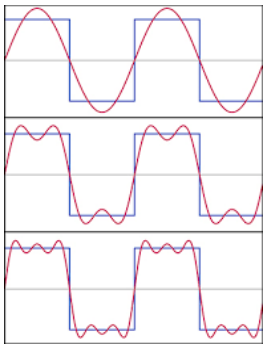



- Materials with **large diffusivity** (carbon, pyrolytic graphite, ..) can transfer heat fast without storing it, this means they don't heat up. So they can be "better" for some applications, eg semiconductors, electronics, nanomaterials, diamond nanowires, etc.

## Key facts

### The great Fourier's ideas



Joseph Fourier (1768 – 1830) demonstrated his ideas on heat diffusion in his famous treatise *Théorie analytique de la chaleur*, published in 1822:

- Fourier's Analysis**:** every function, even with jumps, can be represented as **"sum of harmonics"**. He applied this idea to represent the given initial temperature $\phi(x)$ along the rod. See details here

$$\phi(x) = \sum_{n=1}^{\infty} b_n \cdot \sin\left(\frac{n\pi}{L}x\right) \quad x \in (0, L)$$

- Fourier's Law of heat conduction: heat flows irreversibly **from higher to lower** temperature:

$$q = -k\frac{\partial u}{\partial x}(x,t) \quad \text{where } k = \textit{conductivity} \quad \left[\frac{W}{m \cdot {}^{\circ}\mathrm{K}}\right]$$

- Heat Equation follows from Fourier's Law, Laws of Thermodynamics and Conservation of Energy and can be solved by adding initial condition (IC) and boundary conditions (BC). When BC are homogeneous (eg $0°K$), the solution is transient decaying to zero as time increases.

$$\begin{cases} \frac{\partial u}{\partial t}(x,t) = \boldsymbol{D}\frac{\partial^2 u}{\partial x^2}(x,t) \\ +\text{IC}+\text{BC}(\text{homogeneous}) \end{cases} \quad \text{where } \boldsymbol{D} = \text{diffusivity } \left[\frac{m^2}{s}\right]$$

- **Meaning of Diffusivity vs conductivity:** conductivity tells only about heat transfer rate; diffusivity tells also the effect on the temperature change due to the heat transfer. Materials with **large diffusivity** can **transfer heat fast** (large $k$) **without or slowly heating up** (**have low heat capacity** $c_p$ and low density $\rho$, eg. "don't store heat", but use heat to trigger a new temp grad $\nabla u$ and make the thermal wave propagate). We'll simulate with different material diffusivity.

$$\boldsymbol{D} = \frac{k}{\boldsymbol{c_p}\cdot\rho} = \frac{\text{conducted heat}}{\text{stored heat (heating up)}}$$

- **Transient solution:** satisfies BC (0°K) **and** exponentially decays to 0 when $t \to \infty$ with **a time decay**

$$\tau = \frac{\boldsymbol{L}^2}{\boldsymbol{D}} \quad [s].$$

We'll see more details about transient structure here

**Exercise: explore thermal diffusivity of different materials**

1. Load `materials`.mat file with diffusivity values (in $\frac{mm^2}{s}$) taken from Diffusivity website
2. Create a `dictionary` to associate the material name to the corresponding diffusivity value.
3. Create a UI Control (Drop Down) to quickly choose the material.

```
load materials.mat materials  % Load the table
materials
```

`materials = 46×2 table`

| | Name | Diffusivity |
|---|---|---|
| 1 | "Pyrolytic-graphite-parallel-to-layers" | 1220 |
| 2 | "Carbon/carbon-composite-at-25-°C" | 216.5000 |
| 3 | "Helium-(300-K-1-atm)" | 190 |
| 4 | "Silver-pure-(99.9%)" | 165.6300 |
| 5 | "Hydrogen-(300-K-1-atm)" | 160 |
| 6 | "Gold" | 127 |
| 7 | "Copper-at-25-°C" | 111 |
| 8 | "Aluminium" | 97 |
| 9 | "Silicon" | 88 |
| 10 | "Al-10Si-Mn-Mg-(Silafont-36)-at-20-°C" | 74.2000 |

| | Name | Diffusivity |
|---|---|---|
| 11 | "Aluminium-6061-T6-Alloy" | 64 |
| 12 | "Molybdenum-(99.95%)-at-25-°C" | 54.3000 |
| 13 | "Al-5Mg-2Si-Mn-(Magsimal-59)-at-20-°C" | 44 |
| 14 | "Tin" | 40 |

⋮

```
% Use the table variables to build a dictionary <material name> -->
<diffusivity>
% Note: dictionary is recommended over containers.Map (since R2022b)
% DIFFUSIVITY_map = containers.Map(materials.Name, materials.Diffusivity); % old
way
DIFFUSIVITY = dictionary(materials.Name, materials.Diffusivity);

D_iron = DIFFUSIVITY("Iron")
```

```
D_iron = 23
```

```
% Pick any material and check its diffusivity (in mm^2/s)
material = "Aluminium";
D = DIFFUSIVITY(material)
```

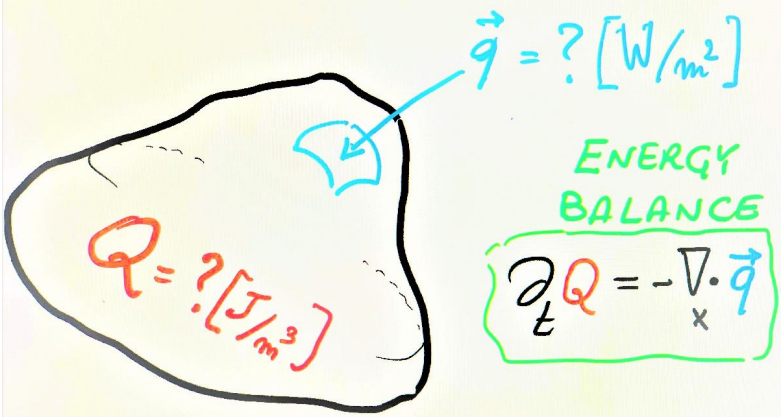
```
D = 97
```

```
save materials.mat DIFFUSIVITY -append
```

## Where the Heat Equation comes from

In 3-dim space ($x \in \mathbb{R}^3$, volume $V \subset \mathbb{R}^3$, with 2-dim boundary surface $\partial V$), the main quantities involved and corresponding units of measurement are:

- $u(x,t)$ $[°K]$ = Absolute Temperature at point $x$ at time $t$
- $Q(x,t)$ $\left[\frac{J}{m^3}\right]$ = Heat Energy per volume, so $\int_V Q(x,t)\,dx$ = **internal Heat Energy** $[J]$ in $V$
- $\vec{q}(x,t)$ $\left[\frac{W}{m^2}\right]$ = Heat Flux through surface, so $-\oint_{\partial V} \vec{q} \cdot \vec{\nu}_{out}\, d\sigma$ = **incoming Heat Flux** $[W]$ through $\partial V$



**Energy Balance**

$$\frac{\partial}{\partial t}\int_V Q(x,t)\,\mathrm{d}x = -\oint_{\partial V} \vec{q}\cdot\vec{\nu}_{\text{out}}\,\mathrm{d}\sigma \overset{\text{Gauss}}{=} -\int_V \nabla_x\cdot\vec{q}\,\mathrm{d}x \qquad [W]$$

or, equivalently, in differential form: :

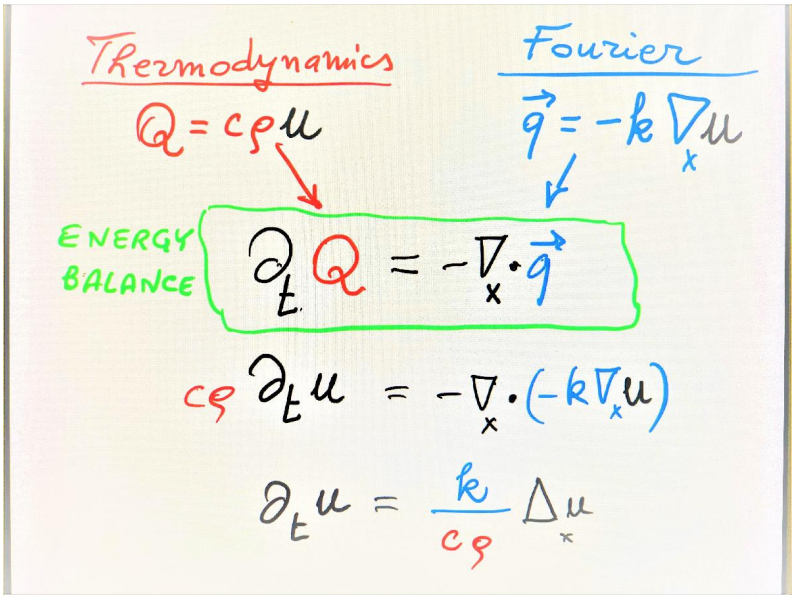
$$\partial_t Q = -\nabla_x\cdot\vec{q} \qquad \left[\frac{W}{m^3}\right]$$

Recall: $\nabla_x\cdot\vec{q} = \text{divergence}\left(\vec{q}\right) = \frac{\partial q_1}{\partial x_1}+\frac{\partial q_2}{\partial x_2}+\frac{\partial q_3}{\partial x_3}.$

**Constitutive Laws**

1. $Q = c_p \rho u$ **(Thermodynamics),** with *specific heat capacity* $c_p \left[\frac{J}{\text{kg}\cdot{}^\circ\text{K}}\right]$
2. $\vec{q} = -k\nabla u$ **(Fourier's Law),** with *conductivity* $k \left[\frac{W}{m\cdot{}^\circ\text{K}}\right]$

By combining the energy balance with the two constitutive laws above, we get the fundamental




$$\Longrightarrow \quad \frac{\partial u}{\partial t} = D\,\Delta u \qquad \textbf{Heat Equation} \text{ with diffusivity } D := \frac{k}{c_p\cdot\rho}\left[\frac{m^2}{s}\right].$$

Recall: $\nabla u = \text{grad}(u) = \left[\frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2}, \frac{\partial u}{\partial x_3}\right]$, $\Delta u = \nabla\cdot(\nabla u) = \text{Laplacian}(u) = \frac{\partial^2 u}{\partial x_1^2}+\frac{\partial^2 u}{\partial x_2^2}+\frac{\partial^2 u}{\partial x_3^2}.$

In 1-dim, we have $\nabla\cdot\vec{q} = \frac{\partial q}{\partial x}$, $\nabla u = \frac{\partial u}{\partial x}$ and $\Delta u = \frac{\partial^2 u}{\partial x^2}.$

**Exercise: derive heat equation in 1-dim case.**

Use the equations above in order (in the order RGB :-)

$$\frac{\partial u}{\partial t} \overset{\text{RED Thermodynamics}}{=} \frac{1}{c_p \rho}\frac{\partial}{\partial t}? \overset{\text{GREEN Energy Balance}}{=} \frac{1}{c_p \rho}\left(-\frac{\partial}{\partial x}?\right) \overset{\text{BLUE Fourier's law}}{=} \frac{1}{c_p \rho}\left(-\frac{\partial}{\partial x}\left(?\frac{\partial}{\partial x}?\right)\right) = \frac{?}{c_p \rho}\frac{\partial^2}{\partial x^2}? = ?\frac{\partial^2}{\partial x^2}?$$

*Solution:*

$$\frac{\partial u}{\partial t} \overset{\text{RED Thermodynamics}}{=} \frac{1}{c_p \rho}\frac{\partial}{\partial t}Q \overset{\text{GREEN Energy Balance}}{=} \frac{1}{c_p \rho}\left(-\frac{\partial}{\partial x}q\right) \overset{\text{BLUE Fourier's law}}{=} \frac{1}{c_p \rho}\left(-\frac{\partial}{\partial x}\left(-k\frac{\partial u}{\partial x}\right)\right) = \frac{k}{c_p \cdot \rho}\frac{\partial^2 u}{\partial x^2} = D\frac{\partial^2 u}{\partial x^2}$$

**Exercise: Using symbolic variables to assign units and check consistency**

Exercise Part 1: use Symbolic variables and **`symunit`** from Symbolic Math Toolbox™ to assign units

```
syms t x u(x,t) q(x,t) Q(x,t) c k rho D
whos t x u
```

```
  Name      Size            Bytes  Class     Attributes

  t         1x1                 8  sym
  u         1x1                 8  symfun
  x         1x1                 8  sym
```

```
unit = symunit; % first, load to package to use the units
t = t * unit.s % seconds
```

t = $t\,\mathrm{s}$

```
x = x * unit.m % meter
```

x = $x\,\mathrm{m}$

```
u = u * unit.K % Kelvin
```

u(x, t) = $u(x,t)\,\mathrm{K}$

```
Q = Q * unit.joule/unit.m^3 % heat energy [J/m^3]
```

Q(x, t) =

$Q(x,t)\,\frac{\mathrm{J}}{\mathrm{m}^3}$

```
q = q * unit.W/unit.m^2       % heat flux (W/m^2)
```

q(x, t) =

$q(x,t)\,\frac{\mathrm{W}}{\mathrm{m}^2}$

```
rho = rho * unit.kg/unit.m^3;  % density [kg/m^3]
c = c * unit.J/(unit.kg*unit.K); % specific heat capacity [J/(kg*K)]
```

Exercise: Complete for conductivity $k$ and diffusivity $D = \frac{k}{c_p \cdot \rho}$

```
% k = ...   % conductivity [W/(m*K)]
% D = ...   % diffusivity (use definition)
```

*Solution*

```
k = k * unit.W/(unit.m*unit.K) % conductivity [W/(m*K)]
```

```
k =
```

$$k\,\frac{\mathrm{W}}{\mathrm{K\,m}}$$

```
D = simplify(k/(c*rho))   % diffusivity [m^2/s]
```

```
D =
```

$$\frac{k}{c\rho}\,\frac{\mathrm{m}^2}{\mathrm{s}}$$

You can compute derivative using **`diff` function** and define equation using `==`. For example:

- $\frac{\partial Q}{\partial t} = -\frac{\partial q}{\partial x}$ (Energy balance)

```
EnergyConservation = diff(Q,t) == -diff(q,x);
EnergyConservation = unitConvert(EnergyConservation, unit.W)
```

```
EnergyConservation(x, t) =
```

$$\frac{\partial}{\partial t}\,Q(x,t)\,\frac{\mathrm{W}}{\mathrm{m}^3} = -\frac{\partial}{\partial x}\,q(x,t)\,\frac{\mathrm{W}}{\mathrm{m}^3}$$

Exercise Part 2: Write 1-dim equations and check units consistency of each equations:

- $Q = c_p \rho u$ (Thermodynamics)
- $q = -k\frac{\partial u}{\partial x}$ (Fourier's Law)
- $\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2}$ (Heat Equation)

```
%% Write your solution


```

*Solution*

```
Thermodynamics = Q == c*rho*u
```

```
Thermodynamics(x, t) =
```

$$Q(x,t)\,\frac{\mathrm{J}}{\mathrm{m}^3} = c\rho u(x,t)\,\frac{\mathrm{J}}{\mathrm{m}^3}$$

```
Thermodynamics_dT  = diff(Q,t) ==  c*rho*diff(u,t)
```

```
Thermodynamics_dT(x, t) =
```

$$\frac{\partial}{\partial t}\,Q(x,t)\,\frac{\mathrm{J}}{\mathrm{m}^3\,\mathrm{s}} = c\rho\frac{\partial}{\partial t}\,u(x,t)\,\frac{\mathrm{J}}{\mathrm{m}^3\,\mathrm{s}}$$

```

```

```
FourierLaw = q == -k*diff(u,x)    % are units ok?
```

```
FourierLaw(x, t) =
```

$$q(x,t)\,\frac{\mathrm{W}}{\mathrm{m}^2} = -k\,\frac{\partial}{\partial x}\,u(x,t)\,\frac{\mathrm{W}}{\mathrm{m}^2}$$

```
HeatEq = diff(u,t) == D*diff(u,x,x)
```

```
HeatEq(x, t) =
```

$$\frac{\partial}{\partial t}\,u(x,t)\,\frac{\mathrm{K}}{\mathrm{s}} = \frac{k\,\frac{\partial^2}{\partial x^2}\,u(x,t)}{c\rho}\,\frac{\mathrm{K}}{\mathrm{s}}$$

```
checkUnits(HeatEq)
```

```
ans = struct with fields:
    Consistent: 1
    Compatible: 1
```

## ********** *PART II : SOLUTION* **********

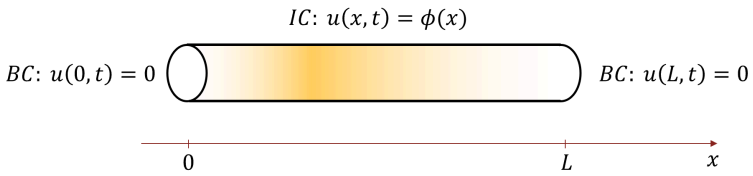
Review Table of Contents

## *Solving Transient problem for Heat Equation*

We consider 1-dim rod, without any interaction (no heat exchange) with external environment.

Inputs are:

- $L>0$ **length** of the rod,
- $D>0$ **diffusivity** of material and
- the initial condition $\phi(x)$




$$\begin{cases} \frac{\partial u}{\partial t}(x,t) = D\,\frac{\partial^2 u}{\partial x^2}(x,t) & x\in(0,L),\,t>0 \\ u(x,0) = \phi(x) & \forall x\in(0,L) \quad (\mathrm{IC} = \text{"Hot Initial Condition"}) \\ u(0,t) = u(L,t) = 0, & \forall t>0 \quad (\mathrm{BC} = \text{"Cold Boundary Condition"}) \end{cases}$$

The **transient problem** is characterized by **"homogeneous BC"**, eg, **hold boundaries at the same fixed temperature** $0°\mathrm{K}$. So "hot" IC will diffuse until reaching a stationary equilibrium, which is $0°\mathrm{K}$, as there is no thermal gradient between boundaries.

Note: many other "non-homogeneous" problems could be solved and a topic for other lectures.

### The structure of Heat solution

Heat Solution has the following "separated" space-time structure:

$$u(x,t) = \sum_{n=1}^{\infty} b_n \cdot \sin(\lambda_n x) \cdot e^{-tD\lambda_n^2}, \qquad \text{with } \lambda_n = \frac{n\pi}{L}$$

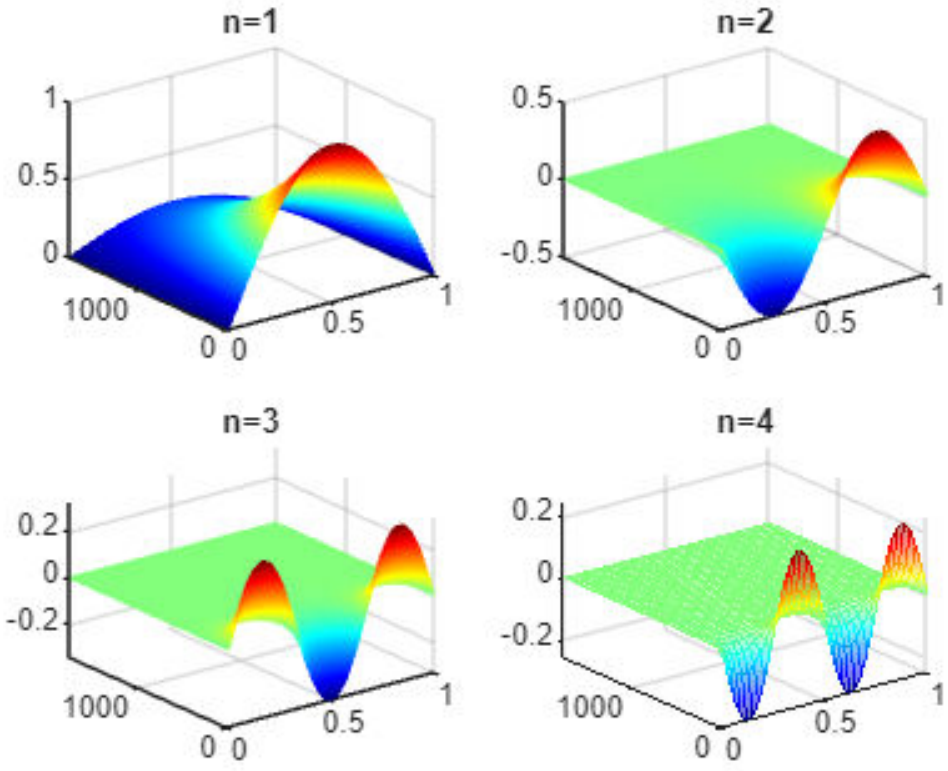
The transient heat solution enjoy some key features:

- **"SIN IN SPACE, EXP IN TIME"** (eigenvector of $\partial^2_{xx}$ and $\partial_t$ respectively)
- **time dilation is "the square" of space dilation**, and parabolic dilations $u(\alpha x, \alpha^2 t)$ are still solutions $\forall \alpha > 0$
- **homogeneous BCs are "embedded"**, eg $u(0,t) = u(L,t) = 0$ are satisfied $\forall b_n$
- **its coefficients $b_n$ are constant (depending on IC),** so the transient solution will **decay to zero as** $t \to \infty$

Go back to Key Facts

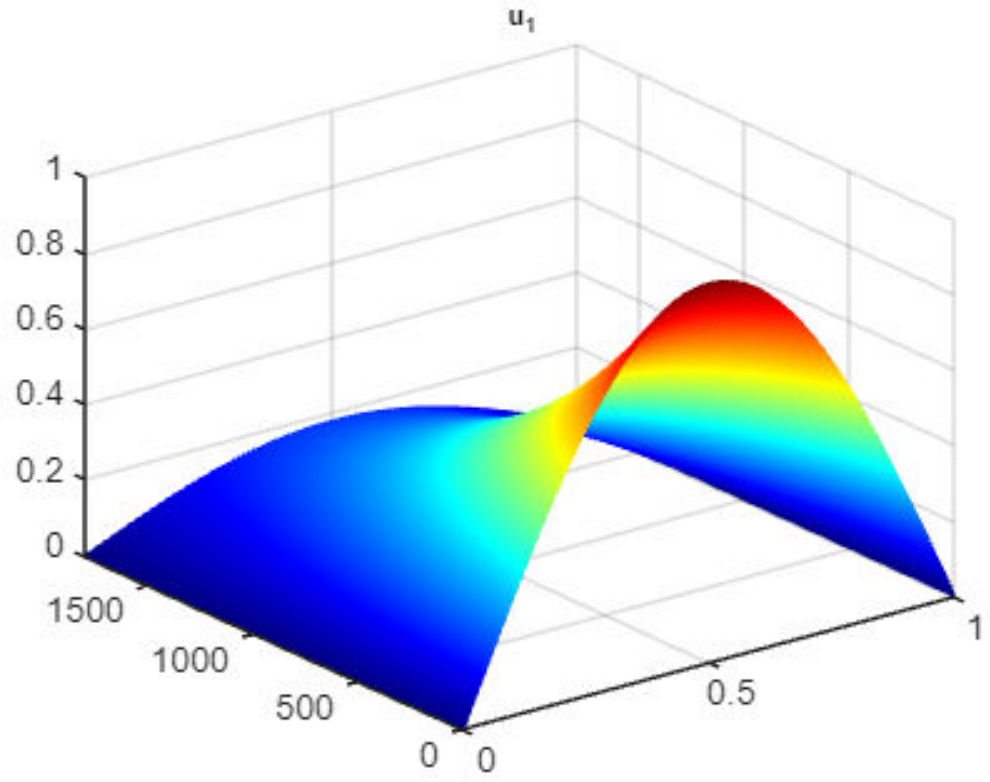
**Visualizing heat structure with animation**

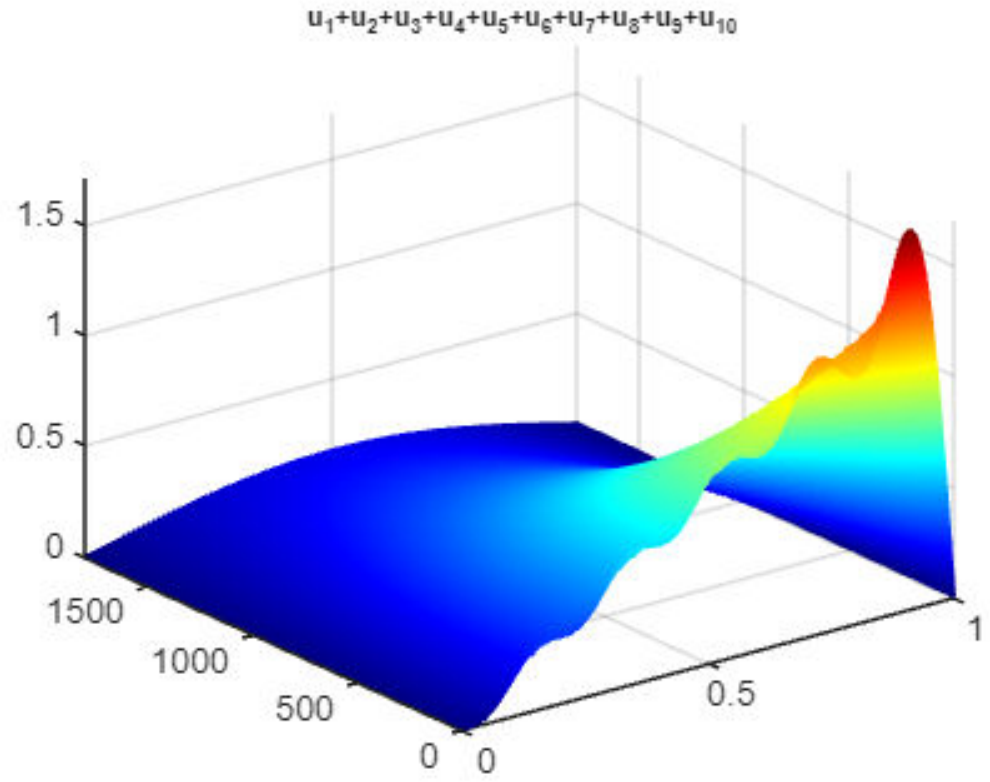
```
D = 97*1e-6; % aluminium diffusivity [m^2/s]
L = 1;       % rod length [m]
Tfinal = 1800; % final time [s] % 30 min = 1800 s

b = @(n) (-1)^(n+1)/n;
lambda = @(n) n*pi/L;
u_n =  @(x,t,n) b(n) .* sin(x*lambda(n)) .* exp(-t*D*lambda(n).^2);

close all
figure
colormap jet
for n = 1:4
    subplot(2,2,n)
    fsurf(@(x,t) u_n(x,t,n), [0 L 0 Tfinal], "EdgeColor","interp")
    hold on
    title("n="+n)
    drawnow
end
alpha(0.5)
```




Animation to understand the heat structure as a "sum":

```
N = 10;
hf =figure;
colormap jet
str = "";
uStruct = @(x,t) 0;
for n = 1:N
    uStruct = @(x,t) uStruct(x,t) +  u_n(x,t,n);
    fsurf(gca, uStruct, [0 L 0 Tfinal], "EdgeColor","interp")
    if n ==1, str="u_1";else str = str+"+u_{"+n+"}"; end
    title(str, FontSize=8)
    drawnow
end
```




**Using Symbolic variables to create a general heat structure**

Let's use symbolic variables and symbolic functions, also to define the heat structure:

```
syms t x n b_n L D  lambda(n) u_n(x,t,n)
assume(n, "integer")
```

```
assume([L D], "positive")
lambda(n) = n*pi/L;
u_n(x,t,n) =  b_n .* sin(x*lambda(n)) .* exp(-t*D*lambda(n).^2) ;

whos t x lambda u_n
```

```
  Name        Size            Bytes  Class     Attributes

  lambda      1x1                 8  symfun
  t           1x1                 8  sym
  u_n         1x1                 8  symfun
  x           1x1                 8  sym
```

We'd better **refactor the previous code with a local function** to quickly build symbolic function object for the heat structure:

Scroll down to insert a local function

```
myheat   = buildHeatStruct
```

```
myheat(x, t, n) =
```

$$b_n e^{-\frac{D n^2 t \pi^2}{L^2}} \sin\left(\frac{\pi n x}{L}\right)$$

### Using SUBS to embed expressions and values

You can then easily substitute expression $b_n = \frac{1}{n}$ or fix numerical values $n=3, L=1, D=97\cdot 10^{-6}$ and then visualize:

```
syms L
subs(myheat, L , 5)
```

```
ans(x, t, n) =
```

$$b_n \sin\left(\frac{\pi n x}{5}\right) e^{-\frac{D n^2 t \pi^2}{25}}$$

```
L = 1; % length of the rod
D = 97e-6; % diffusivity of alluminium (m^2/s)
b_n = 1/n; % some initial condition

u0(x,t,n) = subs(myheat, sym(["L" "D" "b_n"]), [L D b_n])
```

```
u0(x, t, n) =
```

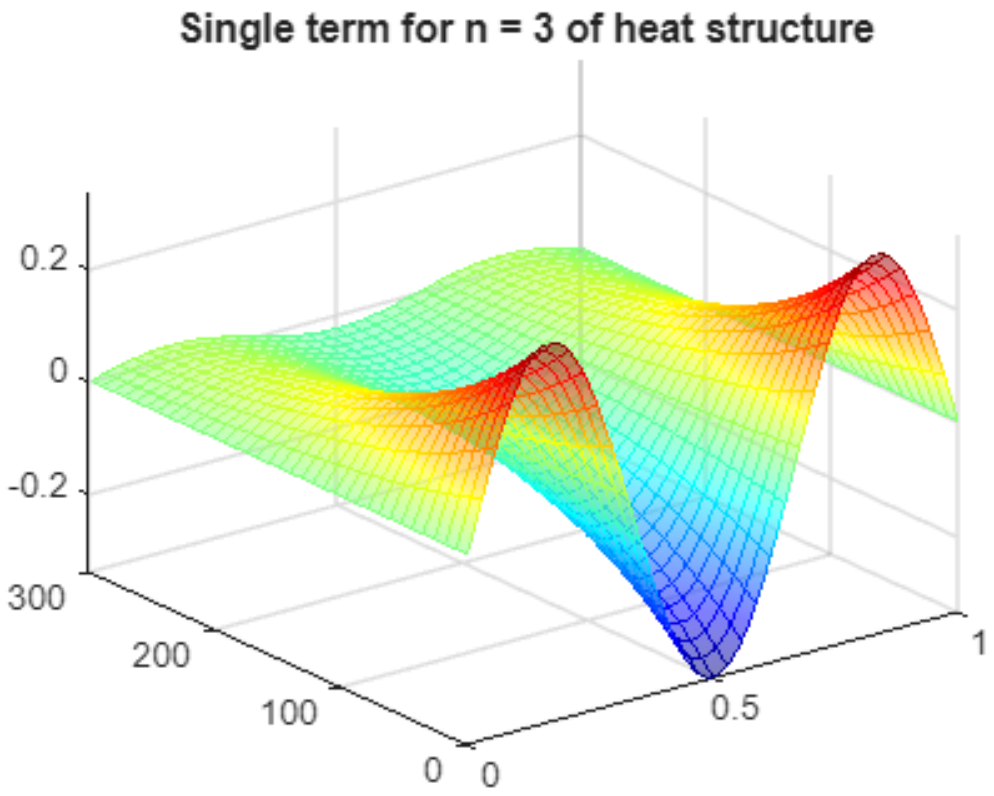
$$\frac{\sin(\pi n x)\, e^{-\frac{3578668350299925 n^2 t}{3689348814741910}}}{n}$$

```
u3(x,t) = u0(x,t, 3) % for example, set n = 3
```

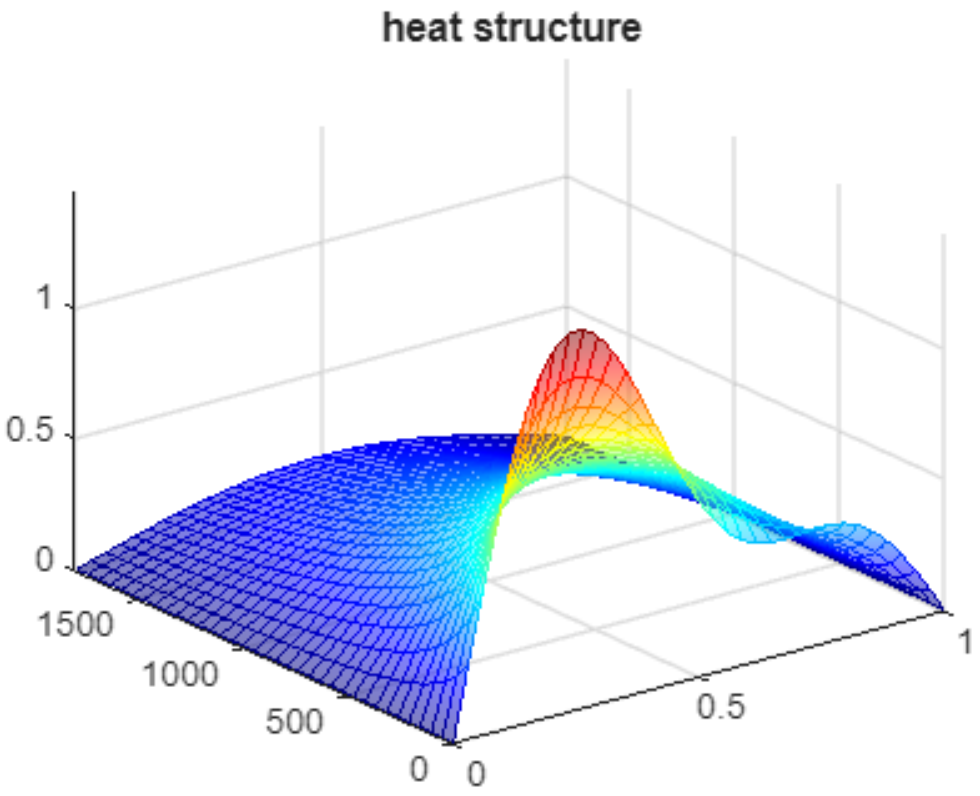
```
u3(x, t) =
```

$$\frac{e^{-\frac{3220801515269687 t}{3689348814741910323}} \sin(3\pi x)}{3}$$

```
Tfinal = 300; % [s]  % about 5 minutes
figure
fsurf(u3, [0 L 0 Tfinal], "EdgeColor","interp"), colormap jet
title("Single term for n = 3 of heat structure")
alpha(0.5)
```




```
heat3(x,t) = u0(x,t, 1) + u0(x,t, 2) + u0(x,t, 3);
Tfinal =1800; % [s]  % about 30 minutes
figure
fsurf(heat3, [0 L 0 Tfinal], "EdgeColor","interp"), colormap jet
title("heat structure")
alpha(0.5)
```




## Understanding the analytical tricks to get the "why" of heat structure

We demonstrate that heat structure $\sum_{n=1}^{\infty} b_n \cdot \sin(\lambda_n x) \cdot e^{-tD\lambda_n^2}$ is actually

- **"A" solution** of Heat Equation
- **"THE" solution**, eg. solutions must have that structure necessarily.

Live Script can help

- **UNDERSTAND where the structure of heat solution come from.**
- **RUN AND VERIFY** analytical methods (separation of variables, ODE, Fourier Analysis,..)

### Exercise: VERIFY that the heat structure satisfies Heat Equation

Reload the heat structure (using local function built before) and then compute derivatives

```
syms t x D
u = buildHeatStruct
```

```
u(x, t, n) =
```

$$b_n \mathrm{e}^{-\frac{D n^2 t \pi^2}{L^2}} \sin\left(\frac{\pi n x}{L}\right)$$

```
ut = diff(u,t)
```

```
ut(x, t, n) =
```

$$-\frac{D b_n n^2 \pi^2 \mathrm{e}^{-\frac{D n^2 t \pi^2}{L^2}} \sin\left(\frac{\pi n x}{L}\right)}{L^2}$$

```
uxx = diff(u, x, 2)
```

uxx(x, t, n) =

$$-\frac{b_n n^2 \pi^2 e^{-\frac{D n^2 t \pi^2}{L^2}} \sin\left(\frac{\pi n x}{L}\right)}{L^2}$$

```
what_is_different = ut/uxx
```

what_is_different(x, t, n) = D

### Look for a solution "in the dark"

Forget any heat structure. How can we find a solution for Heat Equation problem?

$u(x,t) = ???$ to satisfy $\underbrace{\frac{\partial u}{\partial t}(x,t) = D\frac{\partial^2 u}{\partial x^2}(x,t) + \text{BC}}_{\text{STEP 1 Separation of variables (Heat Structure with BC embedded)}}$ + IC STEP 2 FOURIER Analys

```
syms   x t u(x,t) L D n b_n
assume([L D], "positive")
% Heat Equation
HeatEq = diff(u,t) == D *diff(u,x,2)
```

HeatEq(x, t) =

$$\frac{\partial}{\partial t} u(x,t) = D\frac{\partial^2}{\partial x^2} u(x,t)$$

```
% Boundary conditions (used in Step 1: Separation of variables)
BC_0 = u(0,t) == 0
```

BC_0 = $u(0,t) = 0$

```
BC_L = u(L,t) == 0
```

BC_L = $u(L,t) = 0$

```
% Initial condition (used in Step 2: Fourier's Analysis)
phi(x) = x; % or any other function of x
```

### Step 1: Separation of variables

$$\underbrace{\frac{\partial u}{\partial t}(x,t) = D\frac{\partial^2 u}{\partial x^2}(x,t) + \text{BC: } u(0,t) = u(L,t) = 0}_{\text{STEP 1 Separation of variables}}$$

SEPARATION of variables means $u(x,t) = X(x) \cdot T(t)$. So

$$X(x)\frac{d}{dt}T(t) = D\,T(t)\frac{d^2}{dx^2}X(x) + \text{BC: } X(0) = 0 = X(L)$$

```
% Separate variables in the Heat Equation
syms  T(t) X(x)
SepVar = subs(HeatEq, u(x,t), X(x)*T(t))
```

```
SepVar(x, t) =
```

$$X(x)\frac{\partial}{\partial t}T(t) = D T(t)\frac{\partial^2}{\partial x^2}X(x)$$

```
% Separate variables in BC
BCX_0 = subs(BC_0, u(0,t), X(0)*T(t))/T(t)
```

```
BCX_0 =
```

$$X(0) = 0$$

```
BCX_L = subs(BC_L, u(L,t), X(L)*T(t))/T(t)
```

```
BCX_L =
```

$$X(L) = 0$$

**SEPARATION** equation means: $t$ on the left side, $x$ on the right side

$$\frac{\frac{d}{dt}T(t)}{D\cdot T(t)} = \frac{\frac{d^2}{dx^2}X(x)}{X(x)} = C = -\lambda^2 \qquad \forall x, \forall t$$

```
% Separate the equation further
SepVar = SepVar/(D*X(x)*T(t))
```

```
SepVar(x, t) =
```

$$\frac{\frac{\partial}{\partial t}T(t)}{D T(t)} = \frac{\frac{\partial^2}{\partial x^2}X(x)}{X(x)}$$

For the 2 sides to be equal $\forall x, \forall t$ , they MUST BE constant $C$. To avoid trivial solutions, it must be $C < 0$, so it is convenient to **put** $C = -\lambda^2$.

So we can **separate into two ODEs** (eigenvalue problems):

1. 1st order **ODE for time:** $\frac{\dot{T}(t)}{D\cdot T(t)} = -\lambda^2 \implies T(t) = C_1 e^{-t D\lambda^2} \qquad \forall \lambda > 0$
2. 2nd order **ODE for space:** $\frac{\ddot{X}(x)}{X(x)} = -\lambda^2 \; + \underbrace{X(0) = 0}_{\text{left BC}} \implies X(x) = C_2 \sin(\lambda x)$

```
% Separate ODEs introducing constant -lambda^2
var = children(SepVar);
syms lambda
assume(lambda, "positive")
eqT = var{1} == -lambda^2
```

```
eqT =
```

$$\frac{\frac{\partial}{\partial t}T(t)}{D T(t)} = -\lambda^2$$

```
eqX = var{2} == -lambda^2
```

```
eqX =
```

$$\frac{\frac{\partial^2}{\partial x^2}X(x)}{X(x)} = -\lambda^2$$

```
% Solve ODEs
T(t,lambda) = dsolve(eqT)        % 1-order eq for T(t)
```

T(t, lambda) = $C_1 e^{-D\lambda^2 t}$

```
X(x,lambda) = dsolve(eqX, BCX_0); % 2-ord eq for X(t), using X(0)= 0
% Replace constant in X
constantX = setdiff(symvar(X(x,lambda)),sym(['C',x,lambda]));
X(x,lambda) = subs(-X(x,lambda), constantX, sym('C2'))
```

X(x, lambda) = $C_2 \sin(\lambda x)$

**Can we take any** $\lambda$? Well.. no, because **right boundary condition** at $x = L$ will force ...

$$\underbrace{X(L) = 0}_{\text{right BC}} \implies \sin(\lambda L) = 0 \implies \lambda_n = \frac{n\pi}{L} \; \forall n \in Z$$

```
% Solve right boundary condition wrt lambda: X(L,lambda) = 0, lambda = ??
[lambdaL,parameters, conditions] = solve(X(L,lambda) == 0, lambda,
"ReturnConditions",true);
% Add an arbitrary multiplier (replace k with n)
n = sym("n", ["integer", "positive"]);
lambdaL = subs(lambdaL, parameters ,n)
```

lambdaL =

$$\frac{\pi n}{L}$$

**Back to solution structure** $u_n(x,t) = b_n X_n(x) \cdot T_n(t)$

```
% Replace lambda = n*pi/L and rename constant C2 as b_n
syms b_n uStruct(x,t,n)
expT(t,n) = subs(T(t,lambda), [lambda sym('C1')], [lambdaL sym('1')]);
sinX(x,n) = subs(X(x,lambda), [lambda, sym('C2')],[lambdaL, b_n]);

% Back to product solution
uStruct(x,t,n) = collect(sinX(x,n) * expT(t,n), ["sin" "exp"])
```

uStruct(x, t, n) =

$$b_n e^{-\frac{D n^2 t \pi^2}{L^2}} \sin\left(\frac{\pi n x}{L}\right)$$

```
% From symbolic to function handle (optional)
% SinXExpTfun = matlabFunction(uStruct)
```

***Step 2: Fourier Analysis***

The heat structure satisfies BCs for $\forall b_n$, but heat solution must **satisfy the initial condition too:**

$$\text{at } t=0: \quad \phi(x) \overset{\text{IC}}{=} u(x,t=0) = \sum_{n=1}^{\infty} b_n \cdot \sin(\lambda_n x) \cdot e^0, \quad \text{for which } b_n??$$

Do you remember the Key facts? What is **FOURIER ANALYSIS**? **The main idea is <u>ORTHOGONALITY!!!!</u>**

Just multiply each member by $e_m = \sin(\lambda_m x)$ and take integral of product over $[0, L]$.

$$\int_0^L \phi(x) \cdot \sin(\lambda_m x)\,dx = \sum_{n=1}^{\infty} b_n \int_0^L \sin(\lambda_n x) \cdot \sin(\lambda_m x)\,dx = \overset{\text{DO YOU SEE ORTHOGONALITY}}{\dots}$$

**Exercise: prove orthogonality of sine functions,** ie. $e_n \perp e_m$ . We prove $\langle e_n, e_m \rangle = \int_0^L e_n e_n = \frac{L}{2}\delta_{nm} = 0 \text{ when } m \neq n$

```
syms x m n L
assume([m n], ["integer" "positive"]), assume(L, "positive"),
assumeAlso(not(L==pi))
lambda(n) = n*pi/L;
e(n) = sin(lambda(n)*x)
```

```
e(n) =
```

$$\sin\left(\frac{\pi n x}{L}\right)$$

```
check_orthogonality = int( e(n)*e(m), 0, L)
```

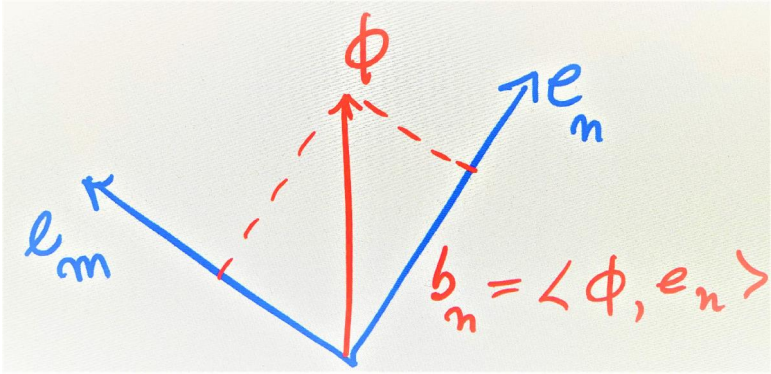
```
check_orthogonality =
```

$$\begin{cases} \frac{L}{2} & \text{if } m = n \\ 0 & \text{if } m \neq n \end{cases}$$

$$\overset{\text{NOW SEE ORTHOGONALITY}}{\dots} = \sum_{n=1}^{\infty} b_n \frac{L}{2}\delta_{nm} = \frac{L}{2} b_m$$

So **<u>FOURIER solution is:</u>** given initial condition $\phi(x)$, take heat structure with

$$b_n = \frac{2}{L}\int_0^L \phi(x) \cdot \sin(\lambda_n x)\,dx$$




Note: if we "normalize" $e_n := \sqrt{\frac{2}{L}}\sin(\lambda_n x)$ and define **inner product** $\langle u, v \rangle = \int_0^L u(x)v(x)dx$, we could say $\{e_n\}_{n=1,2,\dots}$ is an orthonormal and dense set in the Hilbert space $L^2([0,L])$, so $\forall \phi \in L^2([0,L])$ $\phi = \sum_{n=1}^{\infty} b_n e_n$, with

$$b_n = \langle \phi, e_n \rangle \quad \textbf{<u>(projection</u> of } \phi \textbf{ over } e_n)$$

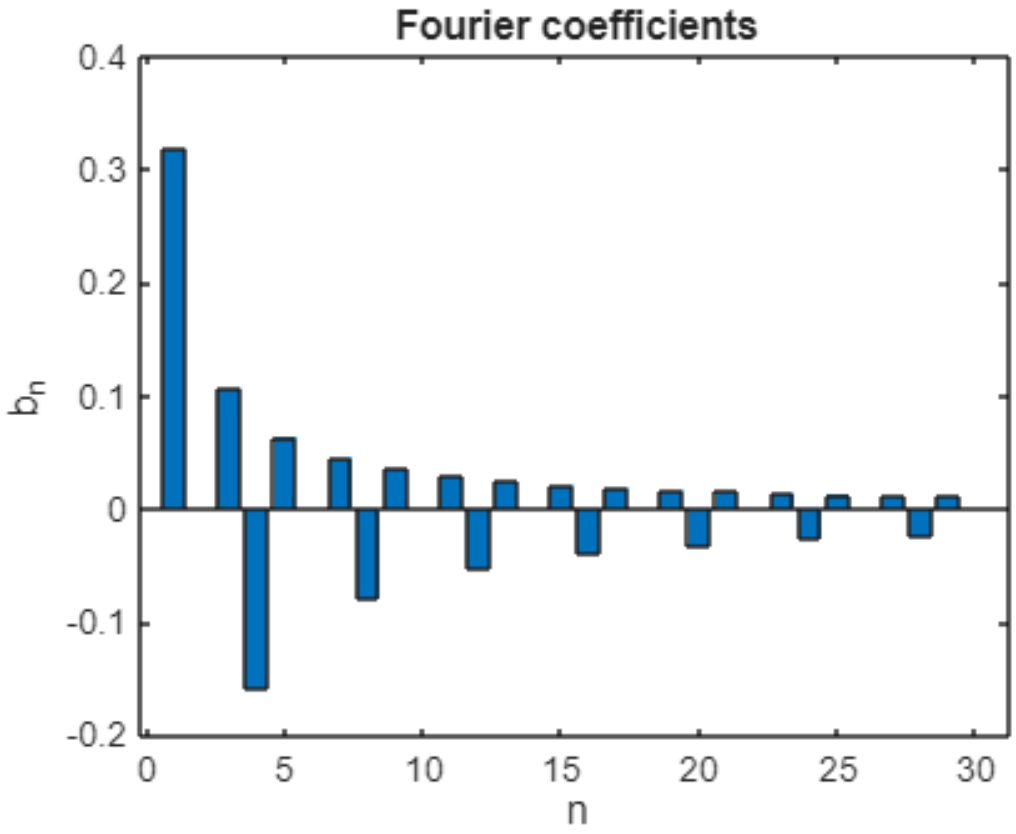
### Computing Fourier's coefficients for IC

```
syms x n b(n)
assume(n, ["integer", "positive"])
L = 1; % length of the rod [meter]

% Choose the Initial condition phi(x)
% phi(x) = L - x;
%% Examples:
% phi(x) = L-x;
phi(x) = piecewise(x<0.5, x, x>0.5, x-0.5);

% Compute Fourier coefficients of phi(x)
b(n) = simplify(2/L* int(phi(x) * sin(n*pi/L * x), 0, L))
```

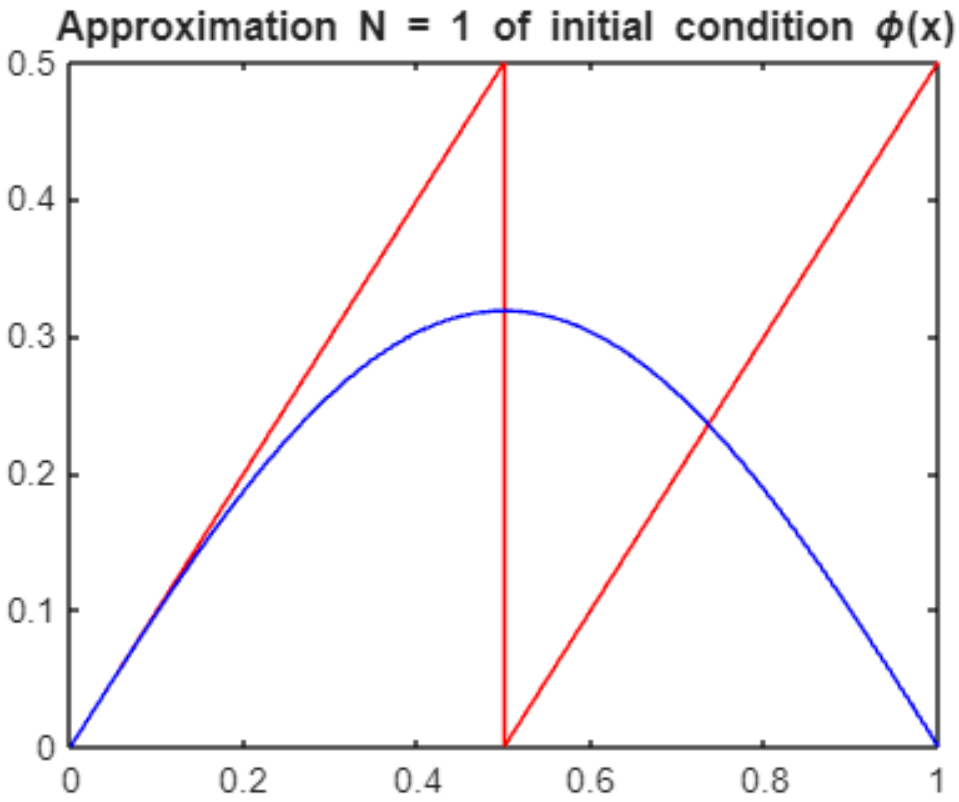
```
b(n) =
```

$$-\frac{(-1)^{n/2}\,((-1)^{n/2}+1)^2}{2n\pi}$$

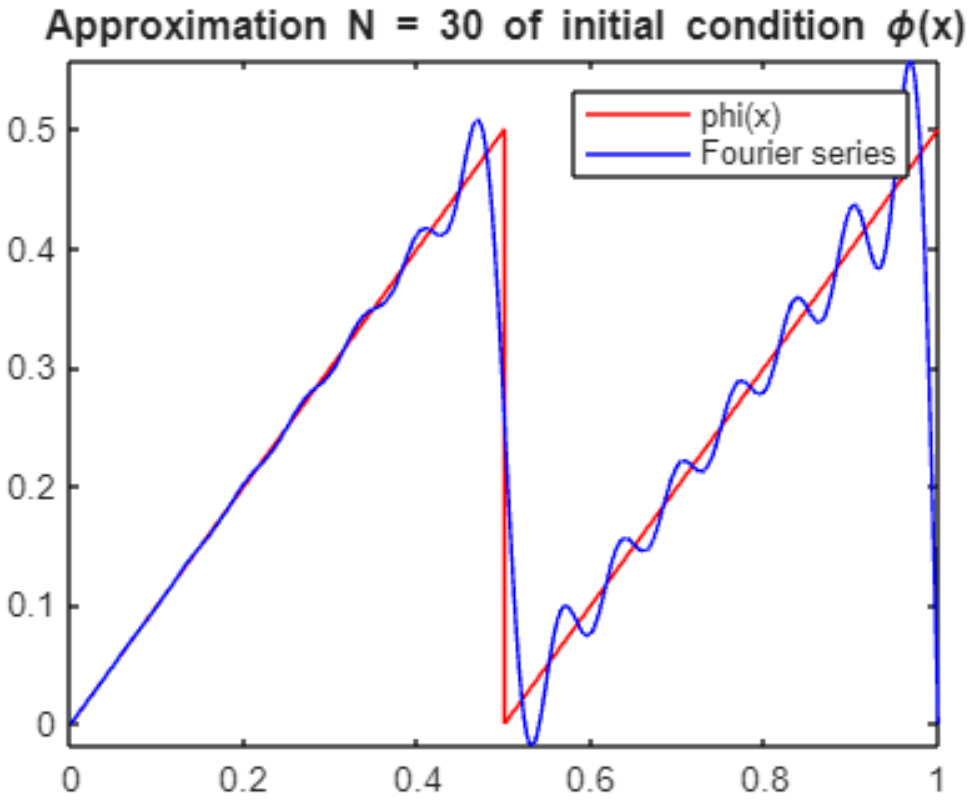
```
Nmax = 30;
% Visualize the coefficients
figure
bar(double(subs(b, n , 1:Nmax)))

xlabel("n")
ylabel("b_n")
title("Fourier coefficients")
```




```
% Visualize approximation of initial condition
figure
for N = 1:Nmax
    fplot(phi, [0 L], "r")
    hold on

    IC = symsum(b(n) * sin(n*pi/L*x), n, 1, N);
    fplot(IC, [0 L], "b")

    title("Approximation N = " + N + " of initial condition \phi(x)")
    drawnow
    hold off
end
```




```
legend("phi(x)", "Fourier series")
```




**Embed Fourier coefficients into Heat structure to get solution**

Suppose to fix $L, D, N$ and $\phi(x)$:

```
syms x t n b(n) phi(x)
assume(n, "integer")
L = 3; % length
phi(x) =  piecewise(x<L/2, x, x> L/2, L/2);  % IC
D = 23e-6;  % material (ex iron)
N = 30; % N terms for approximation

% A:Compute Fourier coefficients (as symbolic function) using phi(x) and L
b(n) = simplify(2/L*int( phi(x)* sin(n*pi/L*x) , 0, L))
```

```
b(n) =
```

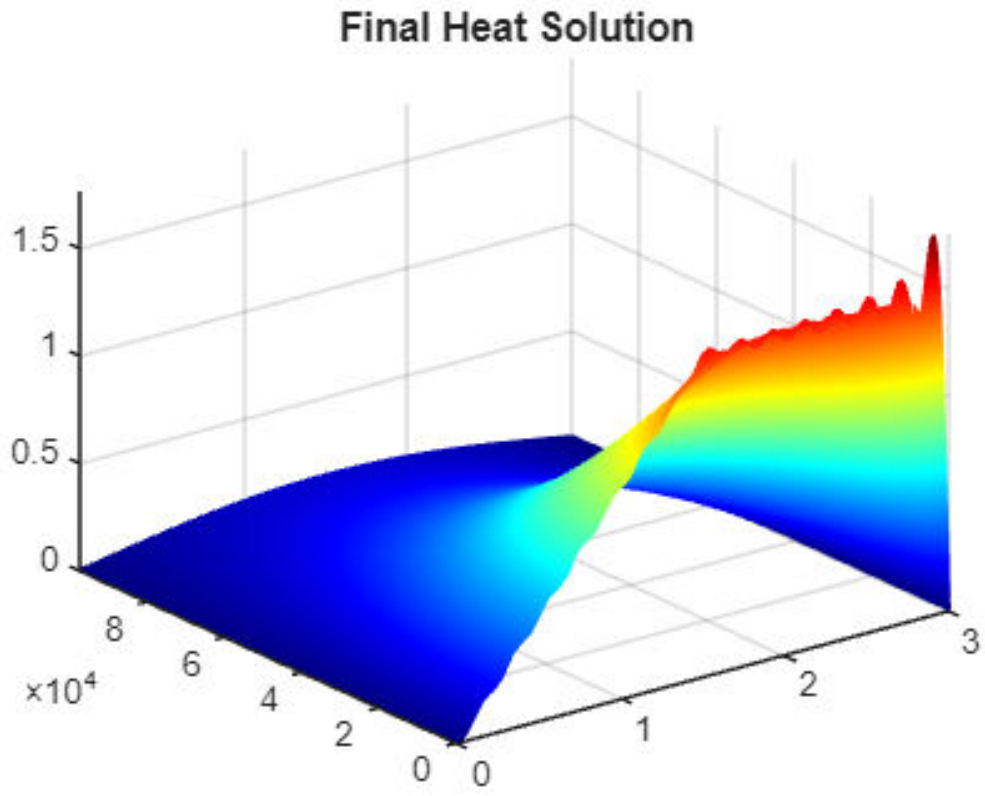
$$\frac{6n\pi\sin\left(\frac{\pi n}{2}\right)^2+6\sin\left(\frac{\pi n}{2}\right)-3\pi n}{n^2\pi^2}$$

```
% B: Build heat structure
heatStruct = buildHeatStruct
```

```
heatStruct(x, t, n) =
```

$$b_n e^{-\frac{D n^2 t \pi^2}{L^2}} \sin\left(\frac{\pi n x}{L}\right)$$

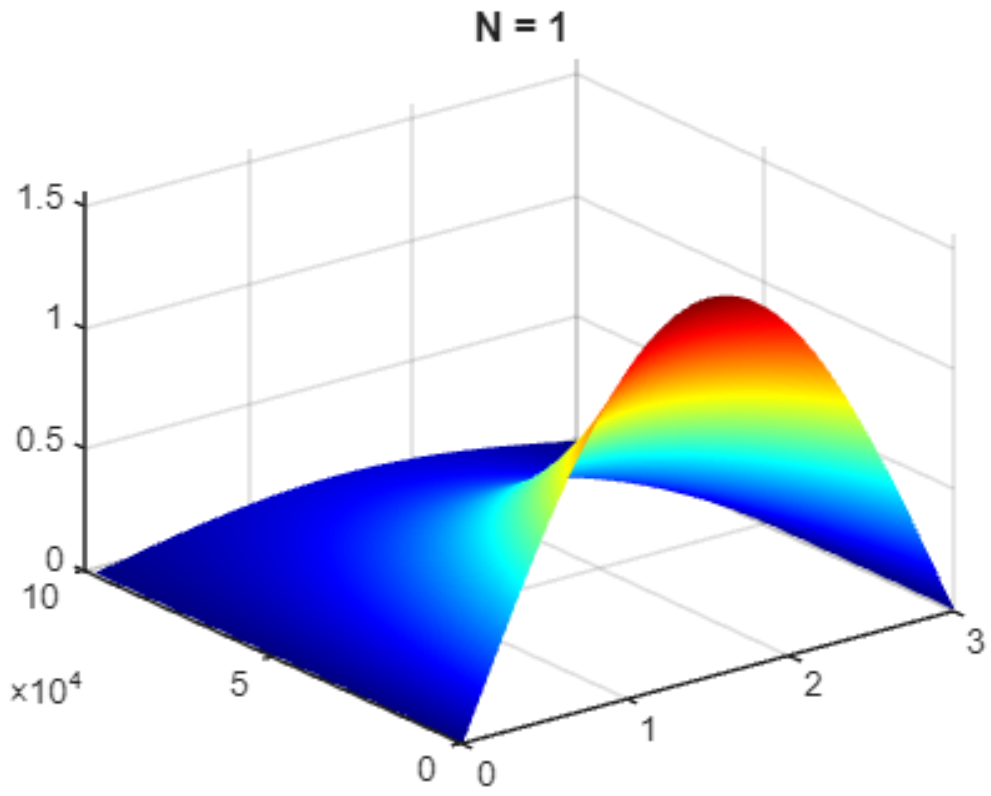
```
% C: Embedding b(n), L and D in the heat structure
heatStructEmb(x,t,n) = subs(heatStruct, sym(["b_n" "L" "D"]), [b L D]);

% D: Build solution by summing up N terms of the series
heatSolN = simplify(symsum(heatStructEmb, n, 1, N));

% Visualize
figure
colormap jet
tau = L^2/D; % [s]
fsurf(heatSolN,[0 L 0 tau/4], "EdgeColor","interp")
title("Final Heat Solution")
```

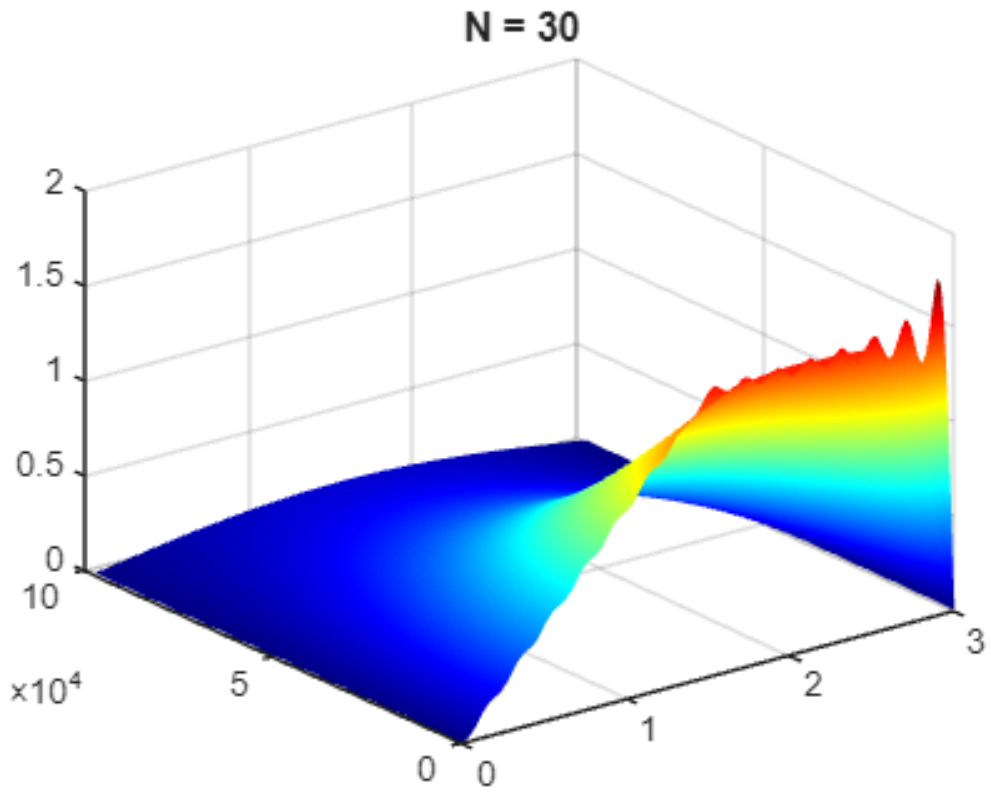



**Visualize animation of heat surface**

To speed up animation, we use numerical grids to discretize space and time and convert symbolic Fourier coefficients into numerical values.

```
figure
colormap jet
N = 30;

% Define numerical grid for space and time
xResolution = 1000;
tResolution = 100;
xnum = linspace(0,L, xResolution);
tnum = linspace(0, tau/4, tResolution);
[XX, TT] = meshgrid(xnum, tnum);

% Convert Fourier coefficients to numerical values
bNum = double(subs(b, n, 1:N));

TEMP = zeros(size(XX));
for k = 1:N
    lambda_k = k*pi/L;
    % Define TEMPerature matrix and surf at each loop
    TEMP = TEMP +  bNum(k) * sin(lambda_k*XX) .* exp(-TT*D*(lambda_k)^2);
    surf(XX, TT, TEMP )
    title ("N = "+ k)
    shading interp
    drawnow
end
```




## Simulation

### Prepare code for reusability

Prepare some reusable local function for quick computation and good visualization (go back to code)

```
material = "Iron";
```

```
L = 1;
syms x
phi(x) =  2*x;
N = 5;

Tfinal = 2;
DHMS = "hours";

%% LOCAL FUNCTION REQUIREMENTS:
% The first 2 input arguments are mandatory to compute D and tau
% The first 4 input arguments are mandatory to compute heat solution
% 5th and 6th are mandatory to visualize: Tfinal+DHMS
(Days,Hours,Minutes,Seconds)
% 7th input (is mandatory to see animation (any placeholder: 1,true, ...)

%% Testing code (and debug) before using it for final simulation
[~, D, tau] = buildHeatSol(material, L) % only computation
```

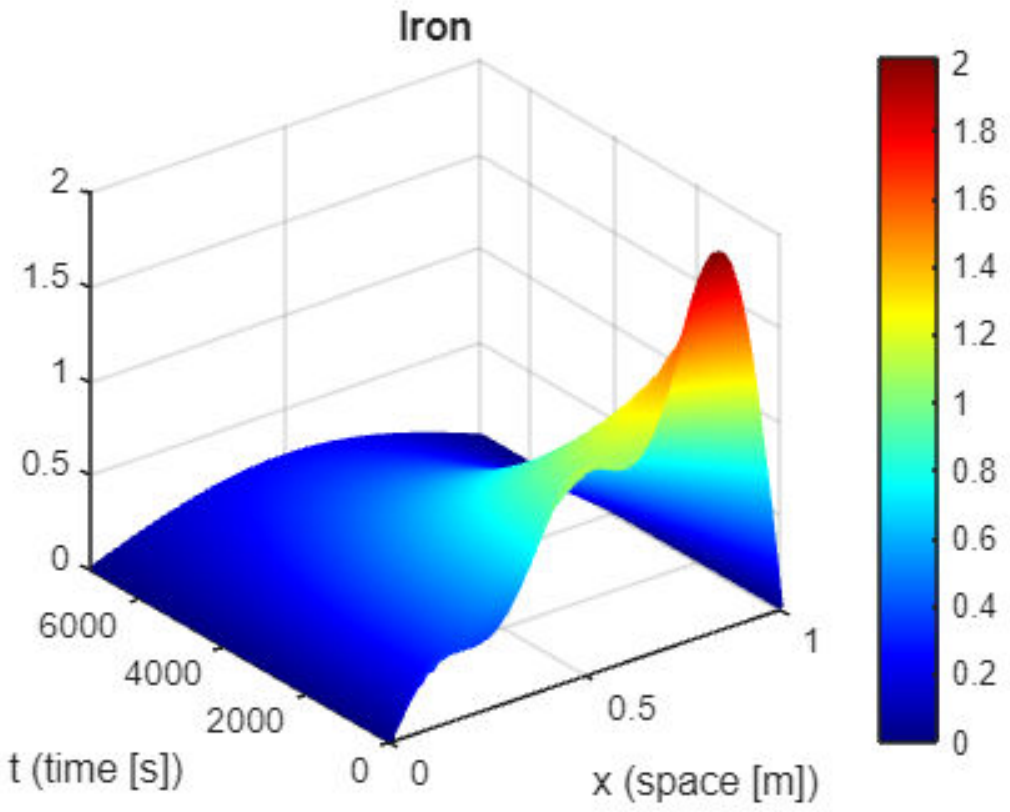
```
D = 2.3000e-05
tau = 4.3478e+04
```

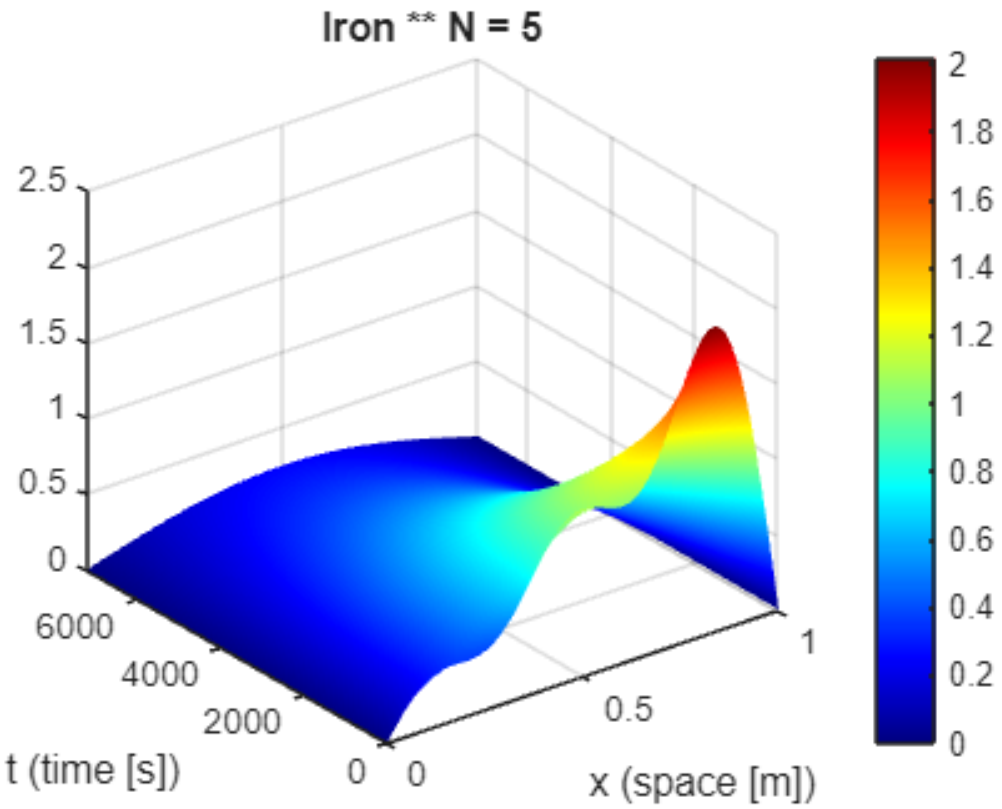
```
uN  = buildHeatSol(material, L, phi, N); % soluti
[~, D, tau, b] = buildHeatSol(material, L, phi, N, Tfinal, DHMS); % Viz. surface
b
```

```
b(n) =
```

$$-\frac{4\,(-1)^n}{n\pi}$$

```
[~, D, tau] = buildHeatSol(material, L, phi, N, Tfinal,DHMS, 1); % Viz. animation
```






**Using interactive controls to change parameters (HAVE FUN)**

Change parameters, then click RUN button. Have fun!

```
%%%%%%%%%%%% YOUR INPUT PARAMETERS %%%%%%%%%%%%
% Choose the material (diffusivity [mm^2/s] or [m^2/s])
material = "Pyrolytic-graphite-parallel-to-layers";
% Choose the material length [m]
L = 0.5;
% Choose initial condition [°K], eg phi(x) =  piecewise(x<L/2, 5, x>L/2, 10);
syms x , phi(x) =piecewise(x<L/2, 5, x>L/2, 10);
% Choose the Fourier approximation degree (sum of N terms)
N = 30;
% Choose the time duration (only for visualization)
Tfinal = 50; DHMS = "seconds";
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Build heat solution
[~, D, tau] = buildHeatSol(material, L, phi, N);
disp("Material: " + material)
```

```
Material: Pyrolytic-graphite-parallel-to-layers
```

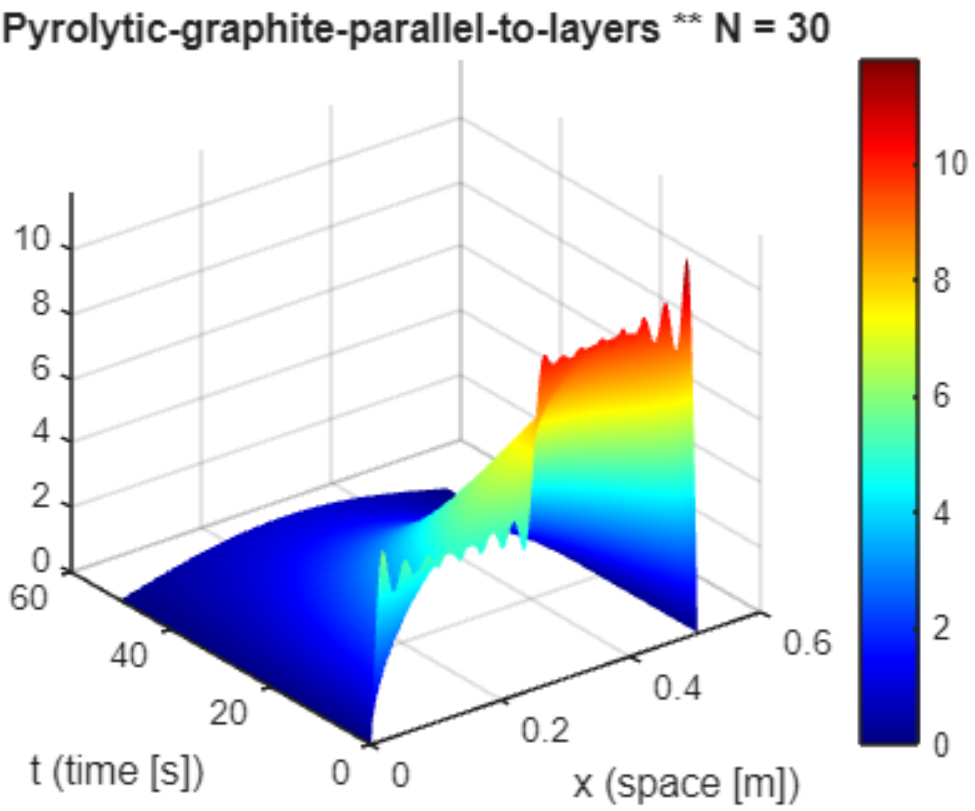
```
disp("Diffusivity: D = " + D*1e6 + " mm^2/s")
```

```
Diffusivity: D = 1220 mm^2/s
```

```
disp("Recommended time duration (tau/4): " +
string(calendarDuration(0,0,0,0,0,round(tau/4))))
```

```
Recommended time duration (tau/4): 0h 0m 51s
```

```
heatSolN  = buildHeatSol(material, L, phi, N, Tfinal, DHMS, 1);
```




**Live Script can be exported into a PDF, HTML, Word or LaTeX:**

```
% Uncomment the lines below to convert and open the PDF version of this live
script
%export("HeatEquationDemo.mlx")
%open HeatEquationDemo.pdf
```

*Thanks*

*THE END*

## ********** *LOCAL FUNCTIONS* **********

### buildHeatStruct

Go back to Structure section, Go back to Code Line 236

```
function u_n  = buildHeatStruct
syms t x n b_n L D u_n(x,t,n) lambda(n)
assume(n, "integer")
assume([L D], "positive")
lambda(n) = n*pi/L;
u_n(x,t,n) =  b_n .* sin(x*lambda(n)) .* exp(-t*D*lambda(n).^2) ;
end % end buildHeatStruct
```

## buildHeatSol

Go back

```
function [heatSolN, D, tau, b] = buildHeatSol(material, L, phi, N, Tfinal, DHMS,
animationflag)
% The first 2 input arguments are mandatory to compute D and tau
% The first 4 input arguments are mandatory to compute heat solution
% 5th and 6th are mandatory to visualize: Tfinal+DHMS
(Days,Hours,Minutes,Seconds)
% 7th input (is mandatory to see animation (any placeholder: 1,true, ...)

%%%%%%%%% Compute diffusivity of material %%%%%%%%%
if isstring(material) || ischar(material)
    load materials.mat DIFFUSIVITY   % load the dictionary (values in mm^2/s)
    D = DIFFUSIVITY(material) * 1e-6;  % convert from mm^2/s to m^2/s
elseif isnumeric(material)
    D = material;
    material = "Material with diffusivity D = " + D;
end
tau = L^2/D;

%%%%%%%%% CONVERSION OF TIME DURATION IN SECONDS %%%%%%%%%
%   5th input:  Tfinal (double, positive)
%   6th input:  DHMS   (string)  (= time unit, eg Days-Hours-Minutes-Seconds)
%   Ex. if Tfinal = 30, DHMS = "minutes", then Tfinal_s = seconds(minutes(30)) =
1800 [s]
%   Ex. if Tfinal =  2, DHMS = "hours", then Tfinal_s = seconds(hours(2)) = 7200
[s]
if nargin >= 6
    Tfinal_s = seconds(eval(DHMS + "(" + Tfinal + ")")); % [s]
end

%%%%%%%%% TODO Build heat solution here %%%%%%%%%
if nargin >= 4
%
syms x t n b(n)
assume(n, ["positive", "integer"])
% A:Compute Fourier coefficients (as symbolic function) using phi(x) and L
b(n) = simplify(2/L*int( phi(x)* sin(n*pi/L*x) , 0, L));

% B: Build heat structure
heatStruct = buildHeatStruct;

% C: Embedding b(n), L and D in the heat structure
heatStructEmb(x,t,n) = subs(heatStruct, sym(["b_n" "L" "D"]), [b L D]);

% D: Build solution by summing up N terms of the series
heatSolN = simplify(symsum(heatStructEmb, n, 1, N));
else
    heatSolN = [];
end
```

```
%%%%%%%%  FLEXIBLE VISUALIZATION %%%%%%%%
if nargin == 6
    % Visualize just the final heat surface
    figure
    fsurf(heatSolN, [0 L 0 Tfinal_s], "EdgeColor","interp");
    colormap jet
    colorbar
    title(material)
    xlabel("x (space [m])")
    ylabel("t (time [s])")

elseif nargin == 7
    % Do ANIMATION
    bNum = double(subs(b, n, 1:N));
    xResolution = max(500, 2.5*N/2);
    tResolution = 100;
    xnum = linspace(0,L,xResolution);
    tnum = linspace(0, Tfinal_s, tResolution);
    [XX, TT] = meshgrid(xnum,tnum);

    colormap jet
    TEMP = zeros(size(XX));
    for k = 1:N
        lambda_k = k*pi/L;
        TEMP = TEMP +  bNum(k) * sin(lambda_k*XX) .* exp(-TT*D*(lambda_k)^2);
        surf(XX,TT,TEMP)
        title(material + " ** N = " +  k)
        xlabel("x (space [m])")
        ylabel("t (time [s])")
        shading interp
        colorbar
        drawnow
    end % loop k

end % if nargin

end % function buildHeatSol
```

Go back