Lecture: Solving Transient Heat Equation

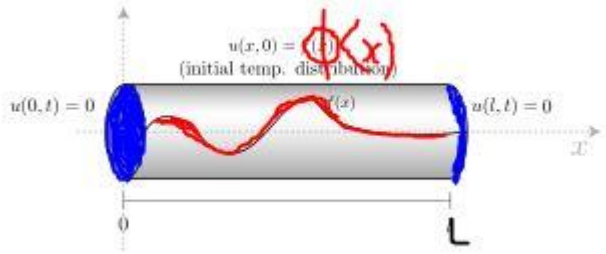
Table of Contents

********** *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

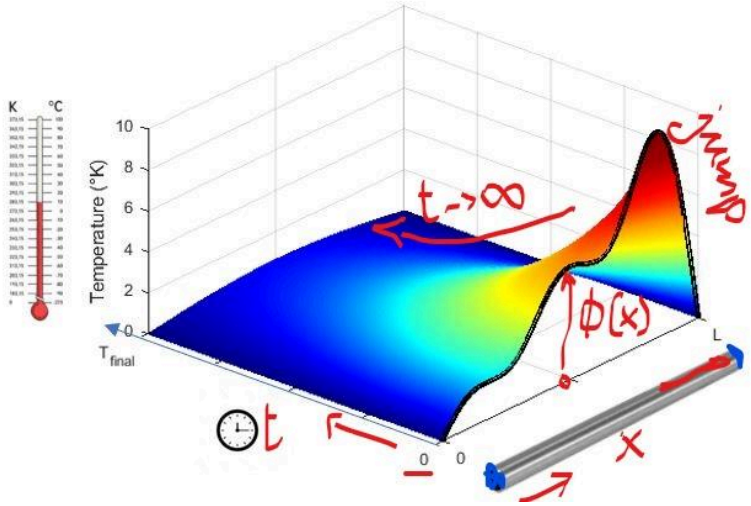


Challenges

- Understand the Heat Equation model and get deeper insight of the **heat solution structure**
- Understand different **thermal behaviours:** 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

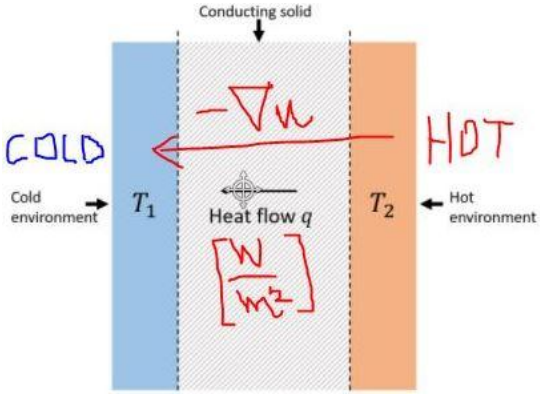
- Heat irreversibly flows from hot to cold until reaching a stationary equilibrium (zero in this case).
- **Discover the heat structure** by using analytical methods, eg separation of variables and Fourier's analysis. **Transient temperature exponentially decays to zero, the time decay depending on the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

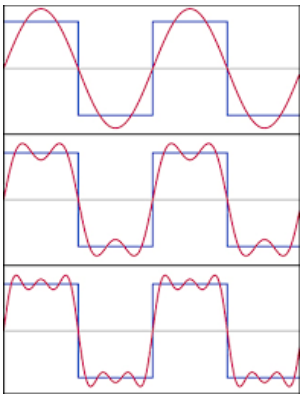


- Materials with **large diffusivity** (carbon, pyrolytic graphite, ..) can transfer heat fast without storing it, this means they don't heat up. So they can be "better" for some applications, eg semiconductors, electronics, nanomaterials, diamond nanowires, etc.

Key facts

The great Fourier's ideas





Joseph Fourier (1768 – 1830) demonstrated his ideas on heat diffusion in his famous treatise *Théorie* \
$x \in (0, L)$

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

$$q = -k\,\frac{\partial u}{\partial x}(x,t) \quad \text{where } \boldsymbol{k} = \textit{conductivity}\ \left[\frac{W}{m\cdot{}^{\circ}\mathrm{K}}\right] \text{ and } q = \text{heat flux}\ \left[\frac{W}{m^2}\right]$$

- 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 $(\text{eg. } 0^{\circ}\mathrm{K})$, the solution is transient decaying to zero as time increases.

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

- **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** ("don't store heat", use heat to trigger new temp grad ∇u, so have low heat capacity c_p and low density ρ). We'll simulate with different material diffusivity.

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

- **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

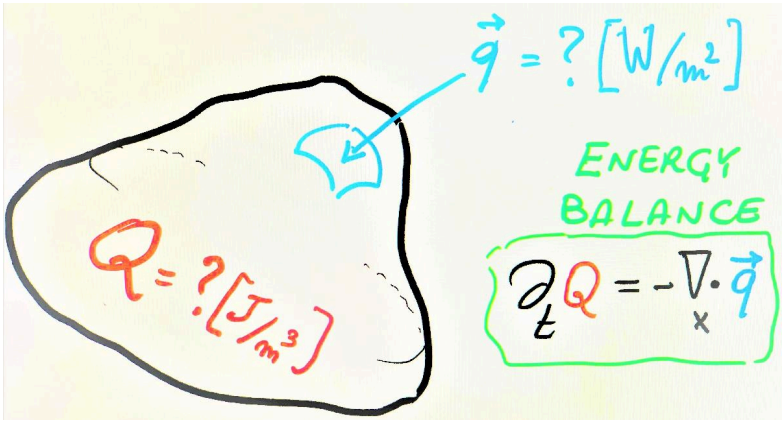
1. Load `materials`.mat file with diffusivity values (in $\frac{\text{mm}^2}{s}$) taken from Diffusivity website
2. Create a map to associate the material name to the corresponding diffusivity value.
3. Create a UI Control (Drop Down) to quickly choose the material. lo

```
load materials.mat materials
materials

DIFFUSIVITY = containers.Map(materials.Name, materials.Diffusivity);
D_Steel1C =  DIFFUSIVITY("Steel-1%-carbon")
D_Iron = DIFFUSIVITY("Iron")
D_Tin= DIFFUSIVITY("Tin")
D_Aluminium =  DIFFUSIVITY("Aluminium")
D_Gold =  DIFFUSIVITY("Gold")

material = "Carbon/carbon-composite-at-25-°C"
D = DIFFUSIVITY(material)

save materials.mat DIFFUSIVITY -append
```

Where the Heat Equation comes from

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

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



Energy Balance

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

or, equivalently, in differential form: :

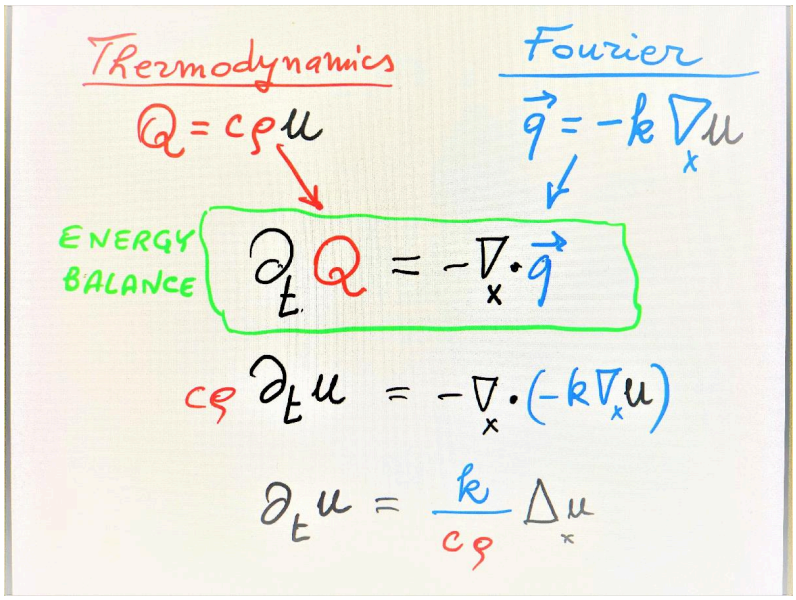
$$\partial_t Q = -\nabla_x\cdot\vec{q} \qquad \left[\frac{W}{m^3}\right]$$

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



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

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` package** to assign units

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

```
unit = symunit; % first, load to package to use the units

t = t * unit.s % seconds
x = x * unit.m % meter
u = u * unit.K % Kelvin

Q = Q * unit.joule/unit.m^3 % heat energy [J/m^3]
q = q * unit.W/unit.m^2       % heat flux (W/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)]
D = simplify(k/(c*rho))   % diffusivity [m^2/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)
```

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
```

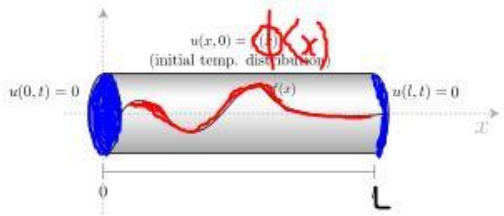
Solution

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

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

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

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

Solving Transient problem for Heat Equation

We consider 1-dim rod, without any interaction (no heat exchange) with external environment. Input: $L > 0$ **length** of the rod, $D > 0$ **diffusivity** of material and the initial condition $\phi(x)$



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

The **transient problem** is characterized by **"homogeneous BC"**, eg, **hold boundaries at the same fixed temperature** $0°\text{K}$. So "hot" IC will diffuse until reaching a stationary equilibrium, which is $0°\text{K}$, 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" structure:

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

The transient heat solution enjoy some key features:

- **"SIN IN SPACE, EXP IN TIME"**:
- **homogeneous BCs are "embedded"**, eg $u(0,t) = u(L,t) = 0$ are satified $\forall\, b_n$
- its coefficients b_n are constant (depending on IC), so it will decay to zero as $t \to \infty$.

- time dilation is "the square" of space dilation, and parabolic dilations $u(\lambda x, \lambda^2 t)$ are still solutions for any $\lambda > 0$

Go back to Key Facts

Visualizing heat structure with animation

```
D = 97*1e-6; % iron diffusivity [m^2/s]
L = 1;       % rod length [m]
Tfinal = 0.5*3600; % final time [s]

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

% Visualize each term before summing
close all
figure(1), colormap("jet")
for n = [1 2 4]
    fsurf( @(x,t) SinXExpT(x,t,n),[0 L 0 Tfinal],"EdgeColor","interp")
    hold on
    alpha(0.5)
    drawnow
end
```

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

```
N = 8;
figure, colormap jet
uStruct = @(x,t) 0;
for n = 1:N
    uStruct = @(x,t) uStruct(x,t) + SinXExpT(x,t,n);
    fsurf(uStruct, [0 L 0 Tfinal], "EdgeColor","interp")
    drawnow
end
title("Heat Structure")
```

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

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

```
myheat = buildHeatStruct
```

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 , 1)

L = 1;       % length of rod
D = 97e-6;  % some material (aluminium)
b_n = 1/n; % some initial condition

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

u3(x,t) = u0(x,t,3)
heat3(x,t) = u0(x,t,1)+ u0(x,t,2) + u0(x,t,3);


Tfinal = 300; % [s]  % about 5 minutes
figure
fsurf(u3, [0 L 0 Tfinal], "EdgeColor","interp"), colormap jet
title("myheat after replacing b_n, n, L and D")
alpha(0.5)

Tfinal = 3600; % [s]  % about 30 minutes
figure
fsurf(heat3, [0 L 0 Tfinal], "EdgeColor","interp"), colormap jet
title("myheat after replacing b_n, n, L and D")
alpha(0.5)

tau = L^2/D
```

Understanding the analytical tricks to get heat structure

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

- **"A" solution** of Heat Equation
- **"THE" solution**, eg. solutions of Heat Equation 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's 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
ut = diff(u,t)
uxx = diff(u,x,2)
what_is_different= ut/uxx
```

Looking for a solution "in the dark" (Step 1 and Step 2)

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

$$\underbrace{\frac{\partial u}{\partial t}(x,t) = D\,\frac{\partial^2 u}{\partial x^2}(x,t) \;+\; \mathrm{BC}}_{\substack{\text{STEP 1}\\ \text{Separation of variables}}} \;+\; \underbrace{\underbrace{\mathrm{IC}}_{\text{STEP 2}}}_{\text{FOURIER's Analysis}} \qquad u(x,t) = ???$$

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

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

% Initial condition (used only in Step 2: Fourier's Analysis)
phi(x) = x;
```

Step 1: Separation of variables

Ideas: Let $u(x,t) = X(x) \cdot T(t)$. So heat equation becomes:

$$X(x)\,\frac{d}{\mathrm{d}t}T(t) = D\;T(t)\,\frac{\mathrm{d}^2}{\mathrm{d}x^2}X(x) \quad\Longrightarrow\quad \frac{\frac{d}{\mathrm{d}t}T(t)}{D\cdot T(t)} = \frac{\frac{\mathrm{d}^2}{\mathrm{d}x^2}X(x)}{X(x)} \qquad \forall\, x, \forall\, t$$

```
syms  T(t) X(x)
SepVar = subs(HeatEq, u(x,t), X(x)*T(t))
SepVar = SepVar/(D*X(x)*T(t))
```

Boundary conditions will turn into $X(0) = 0 = X(L)$

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

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

After separation, both sides of the equation depend on different variables, yet are equal. This is only possible if each side is a constant. Equate each side to an arbitrary constant C to get two differential equations.

$$\frac{\frac{d}{dt}T(t)}{D \cdot T(t)} = \frac{\frac{d^2}{dx^2}X(x)}{X(x)} = C \qquad \textbf{(They must be constant!!!)}$$

Solving ODEs with BC (eigenvalue problems)

To avoid trivial solutions, it must be $C < 0$. It is convenient to put $C = -\lambda.^2$ (as solution $X(x)$ will contain $\sqrt{-C}$)

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

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

% Solve ODEs
T(t,lambda) = dsolve(eqT); % 1-order eq for T(t), 1 constant
X(x,lambda) = dsolve(eqX, BCX_0); % 2-ord eq for X(t), 1 constant resolved using first BC X(0)=

% Replace constants
constantT = setdiff(symvar(T(t,lambda)),sym(['C',t,lambda, D]));
constantX = setdiff(symvar(X(x,lambda)),sym(['C',x,lambda]));
T(t,lambda) = subs(T(t,lambda), constantT, sym('C1'))
X(x,lambda) = subs(-X(x,lambda), constantX, sym('C2'))
```

From right boundary condition, it follows that solution must have half-period L, eg

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

```
% 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)
```

Back to solution structure

We replace λ_n, rename all constants as b_n and rebuild the product $u_n(x,t) = b_n X_n(x) \cdot T_n(t)$

```
% Replace lambda = n*pi/L and constant C2 with b_n
syms b 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 solution structure by multiplying
sinX(x,n) * expT(t,n)
uStruct(x,t,n) = collect(sinX(x,n) * expT(t,n), ["sin" "exp"])

% 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:**

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

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

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

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

Exercise: prove orthogonality of sine functions.

To prove $e_n \perp e_m$, when $m \neq n$. We have to prove the inner product defined by $\langle e_n, e_m \rangle \overset{\text{def}}{:=} \int_0^L e_m e_n$ is null. So we have to compute the integral of the product $e_n \cdot e_m$. It will result $\int_0^L e_m e_n = \frac{L}{2}\delta_{\text{nm}} = 0$ when $m \neq n$.

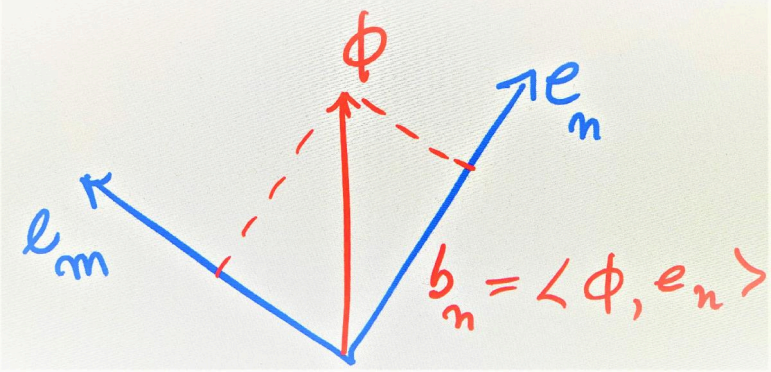
```
syms x m n L
assume([m n], ["integer" "positive"]), assume(L, "positive"), assumeAlso(not(L==pi))
lambda(n) = n*pi/L;
e(n) = sin(lambda(n)*x)
check_orthogonality = int( e(n)*e(m), 0, L)
```

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

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

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

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



Compute Fourier's coefficients for initial condition

```
syms x n b(n)
assume(n, ["integer", "positive"])
% Initial condition
L = 1;
phi(x) = L-x;
% phi(x) = L-x;
phi(x) = piecewise(x<L/2, 0, x>L/2, 1); % square
phi(x) = piecewise(x<L/2, x, x>L/2, 1);
phi(x) = piecewise(x<L/2, x, x>L/2, x-0.5); % saw tooth

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

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")
    %fplot(symsum(b(n)*sin(n*pi/L *x),n, 1,N), [0 L])
    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)
assume(n, "integer")
L = 1;
phi(x) = piecewise(x<L/2, x, x>L/2, L/2); % initial condition

D = 23e-6; % iron
N = 40;

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

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

Visualize animation of heat surface

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

```
figure
colormap jet
Nmax = 30;
% Define numerical grid for space and time
xResolution = 500;
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:Nmax));

TEMP = zeros(size(XX));
for k = 1:Nmax
    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

Refactoring for code reusability

Prepare some reusable local function for quick computation and good visualization:

```
material = "Iron";
L = 1;
syms x
phi(x) =  x;
Tfinal = 2;
DHMS = "hours";
N = 5;

%% buildHeatSol (local function):
% The first 4 input are mandatory.
% To visualize the heat surface, pass also 5th and 6th
% To see animation of heat surface, specify also 7th input (eg. 1)

[uN, D, tau] = buildHeatSol(material, L, phi, N);  % only computation
[uN, D, tau] = buildHeatSol(material, L, phi, N, Tfinal, DHMS); % visualize for some duration
```

Using interactive controls to change parameters (HAVE FUN)

Change parameters, then click RUN button. Have Fun!!!

```
%%%%%%%%%%%%%%%% YOUR INPUT PARAMETERS %%%%%%%%%%
%  Choose the material
material =  "Pyrolytic-graphite-parallel-to-layers";
% Choose the material length
L = 1; % meter
% Choose the time duration (only for visualization)
Tfinal = 3; DHMS = "minutes";
% Define initial condition u(x,0) = phi(x)
syms x
phi(x) =  piecewise(x<L/2, 5, x>L/2, 10); %
```

```
%e = 0.5;
%phi(x) =  piecewise(x>=L/2-e&x<=L/2+e, 10, x<=L/2-e|x>=L/2+e, 0);

% Choose the Fourier approximation degree
N = 40;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Build heat solution
[~, D, tau] = buildHeatSol(material, L, phi, N);
disp("Material: " + material)
```

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

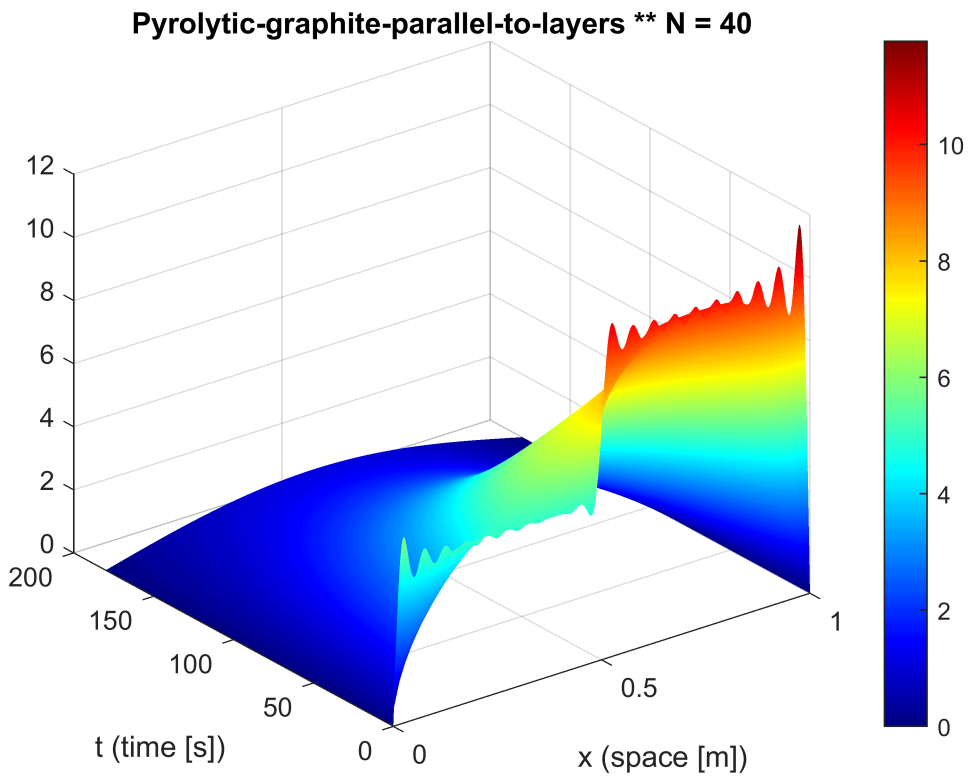
```
disp("Diffusivity: D = " + D*1e6 + " mm^2/s")
```

```
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 3m 25s
```

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



Thanks

THE END

********** *LOCAL FUNCTIONS* **********

buildHeatStruct

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

buildHeatSol

```
function [heatSolN, D, tau] = buildHeatSol(material, L, phi, N, Tfinal, DHMS, animationflag)
syms x t n b(n)

% Compute diffusivity of material
if isstring(material) || ischar(material)
    load materials.mat DIFFUSIVITY
    D = DIFFUSIVITY(material)*1e-6;
elseif isnumeric(material)
    D = material;
    material = "Some material with diffusivity D = " + D;
end
tau = L^2/D;

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

%% Visualize the heat surface if you pass further input Tfinal and its unit HMS
if nargin >= 6
    % Some visualization is required, so I need to compute a final time duration
    Tfinal_s = seconds(eval(DHMS+"("+Tfinal+")"));
end

if nargin == 6
    % Visualize just the final heat surface using symbolic
    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 using numerical approach
    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
```

Thanks

THE END