e-ISSN: 2661-6505

Tablet School Journal

Nr. 16 Vol. 1 Oct. 2023

AI gives answer about:

1D Heat equation with Python

General Information about Tablet School Journal

1. Goal.

Tablet School Journal aims to disseminate scientific knowledge at all levels and to all audiences in an understandable and verified way; with the purpose that the content described can be verified and serves as a basis for future scientific research and development.

2. Topics.

Tablet School Journal covers all the fields and subfields described by UNESCO's International Standard Nomenclature for Fields of Science and Technology focusing on the use and application of computer science, technology and data processing.

3. Editorial managers.

Chief Editor:

Marcos Gutiérrez - Tablet School

Email: marcosgutierrez@tablet-school.com

Editorial committee:

Gustavo Tapia - Universidad de Buenos Aires - Argentina

Email: gustavo.tapia1@gmail.com

Mauro de Jesús - Universidad de Buenos Aires - Argentina

Email: mauro.dejesus@fce.uba.ar

Gabriela Figueroa - Universidad de Buenos Aires - Argentina

Email: gabyyfig.gf@gmail.com

Eduardo Álvarez - Universidad de Buenos Aires - Argentina

Email: ingedualvarez@gmail.com

Eduardo Cassullo - Universidad de Buenos Aires - Argentina

Email: cassulloeduardo@gmail.com

Henrique Schneider - Nordakademie, University of Applied Sciences - Germany

Email: henrique.schneider@nordakademie.de

4. Release dates.

Tablet School Journal is published online, continuously on a quarterly basis, in February, June and October of each year.

5. Authors information.

All authors and co-authors who publish in Tablet School Journal must provide the following information:

The Institution's name to which they belong and the country in which the Institution is located.

Their status as independent researchers in the case that their research doesn't belong to any particular Institution, as well as the country in which they reside.

E-mail address of the authors and co-authors, which will be displayed in the published article so that the author and co-authors can be contacted.

6. Paper submission process.

The publication process begins with the download of the author's guide, from: https://www.tablet-school.com/tablet-school-journal/. Based on the downloaded guide, the paper must be prepared to be sent to info@tablet-school.com. After this, the double- or single-blind review process begins, which lasts approximately 30 days. In the case that the reviewers recommend corrections, the authors have 15 days to carry them out and a new shorter review process begins, with a duration of 15 days. The reviewers check each one of the corrections, and if they are according with the remarks made or if there is the properly argued rebuttal, the publication of the article is recommended and the process of publication begins in the closest and available number of the journal.

7. Publishing fees.

The publishing fees for each paper accepted papers after the reviewing process with a maximum of 4 authors (1 main author and 3 co-authors) amounts to 250 USD, payable through PayPal.

8. Originality of the research material.

All articles, essays, reviews or technical notes sent for review must be original, with their own contribution and scientific value.

Bibliographic reviews should not be based solely on paraphrasing, but on finding the research and knowledge gaps that result from the analysis of the sources being studied, and on issuing new and original solutions or procedures that provide a solution to a problem.

9. Code of ethics.

Tablet School Journal is guided by COPE Guidelines which are intended to advise editors and publishers on expected publication ethics practices.

To our readers:

Dear Esteemed Readers,

I am delighted to extend a warm welcome to you as we unveil the latest edition of the Tablet School Journal. Our mission has always been to bring you cutting-edge, relevant content in the realm of education, and this edition is no exception.

In this special issue, we explore a topic of great significance and relevance - the Heat Equation, enriched with Python codes. What sets this edition apart is the remarkable collaborative effort between human expertise and the power of artificial intelligence. The content you'll find in this issue was generated with the assistance of ChatGPT, a state-of-the-art language model developed by OpenAI.

Key Highlights of this Special Edition:

- The Basic Heat Equation.
- The Heat Equation with Internal Heat Source.
- The Heat Equation with Conduction Boundary Conditions.
- The Heat Equation with Convection Boundary Conditions.
- The Heat Equation with Radiation Boundary Conditions.

The collaboration with ChatGPT has allowed us to explore new horizons in education and research, combining human knowledge and AI capabilities to deliver content that is both informative and engaging.

I extend my heartfelt gratitude to our dedicated editorial team for their hard work in bringing this special edition to life. We hope you find this issue as enlightening and inspiring as we do.

Thank you for your continued support and readership. We remain committed to serving the academic and scientific community with quality content that advances knowledge and fosters innovation.

Enjoy this edition of the Tablet School Journal, and may it ignite your curiosity and passion for learning.

Sincerely,

Martina Memaj

Chief Editor

Tablet School Journal

The Basic Heat Equation

The heat equation is a fundamental partial differential equation that describes how heat or temperature is distributed over time within a given medium. It's a key concept in the field of heat transfer, physics, and engineering, and it's especially useful for understanding how heat is conducted through solids, liquids, and gases. The heat equation is typically expressed as:

$\partial u/\partial t = \alpha \nabla^2 u$

Where:

$\partial u/\partial t$ represents the rate of change of temperature (u) with respect to time (t).

α is the thermal diffusivity, a material property that describes how quickly heat diffuses through the medium. It's related to the material's thermal conductivity, density, and specific heat capacity.

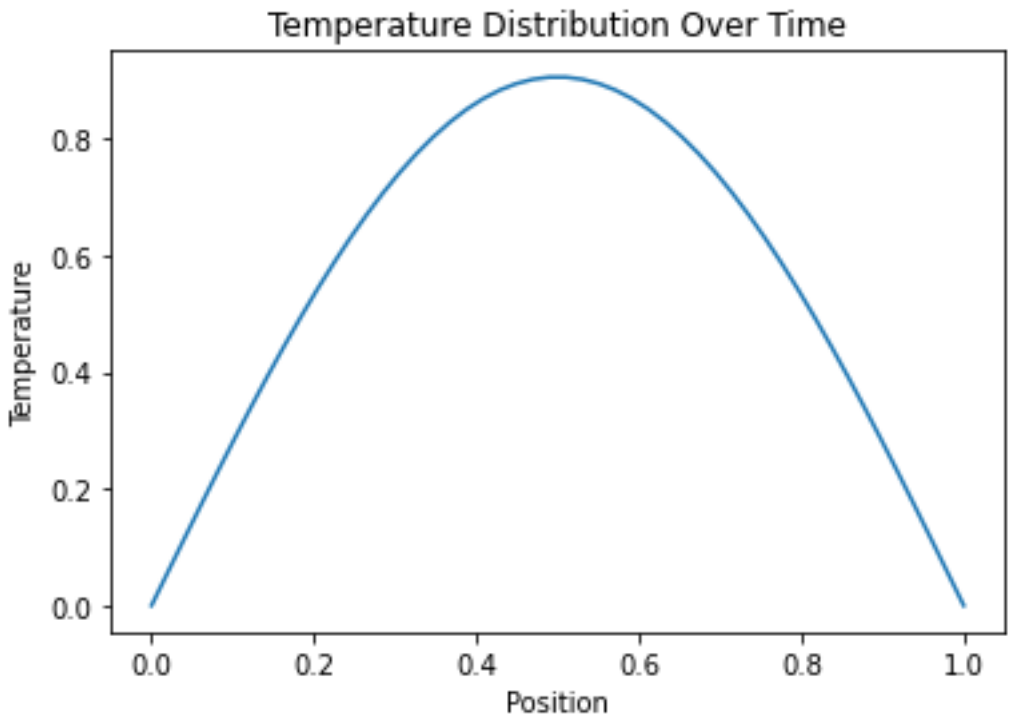
$\nabla^2 u$ (Laplacian of u) is the spatial second derivative of temperature, which quantifies the curvature of temperature distribution in space.

The heat equation implies that the change in temperature at any point is proportional to the Laplacian of temperature and time. In simple terms, it states that the rate of change of temperature at a point is influenced by the temperature differences in its neighborhood and the thermal properties of the material.

Now, let's write a basic Python code to solve the one-dimensional heat equation using the finite difference method. This code demonstrates how to approximate the temperature distribution over time within a one-dimensional rod. We'll use explicit time-stepping, which is a straightforward but less stable numerical method.

```
import numpy as np

import matplotlib.pyplot as plt


# Parameters

L = 1.0  # Length of the rod

T = 1.0  # Total simulation time

Nx = 50  # Number of spatial points

Nt = 500  # Number of time steps

alpha = 0.01  # Thermal diffusivity
```

```
# Spatial and temporal discretization
dx = L / (Nx - 1)
dt = T / Nt

# Initial conditions
x = np.linspace(0, L, Nx)
u = np.sin(np.pi * x)  # Example initial temperature distribution

# Simulation loop
for n in range(Nt):
    u_new = u.copy()
    for i in range(1, Nx - 1):
        u_new[i] = u[i] + alpha * (u[i + 1] - 2 * u[i] + u[i - 1]) * dt / (dx**2)
    u = u_new

# Plot the temperature distribution
plt.plot(x, u)
plt.xlabel('Position')
plt.ylabel('Temperature')
plt.title('Temperature Distribution Over Time')
plt.show()
```



This code simulates the heat conduction in a rod with an initial sinusoidal temperature distribution and plots the temperature evolution over time. Please note that for more complex problems, you may need to use more advanced numerical methods and libraries like NumPy and SciPy to solve the heat equation.

The Heat Equation with Internal Heat Source

The heat equation with an internal heat source is an extension of the standard heat equation that takes into account the generation of heat within the medium itself. This equation is particularly useful in scenarios where there is a heat source (or sink) within the material, such as chemical reactions, electrical resistance, or other processes that produce or absorb heat.

The heat equation with an internal heat source is given as follows:

$\partial t/\partial u = \alpha \nabla^2 u + S(x,t)$

Where:

$\partial t/\partial u$ represents the rate of change of temperature (u) with respect to time (t).

α is the thermal diffusivity of the material, as explained in the previous response.

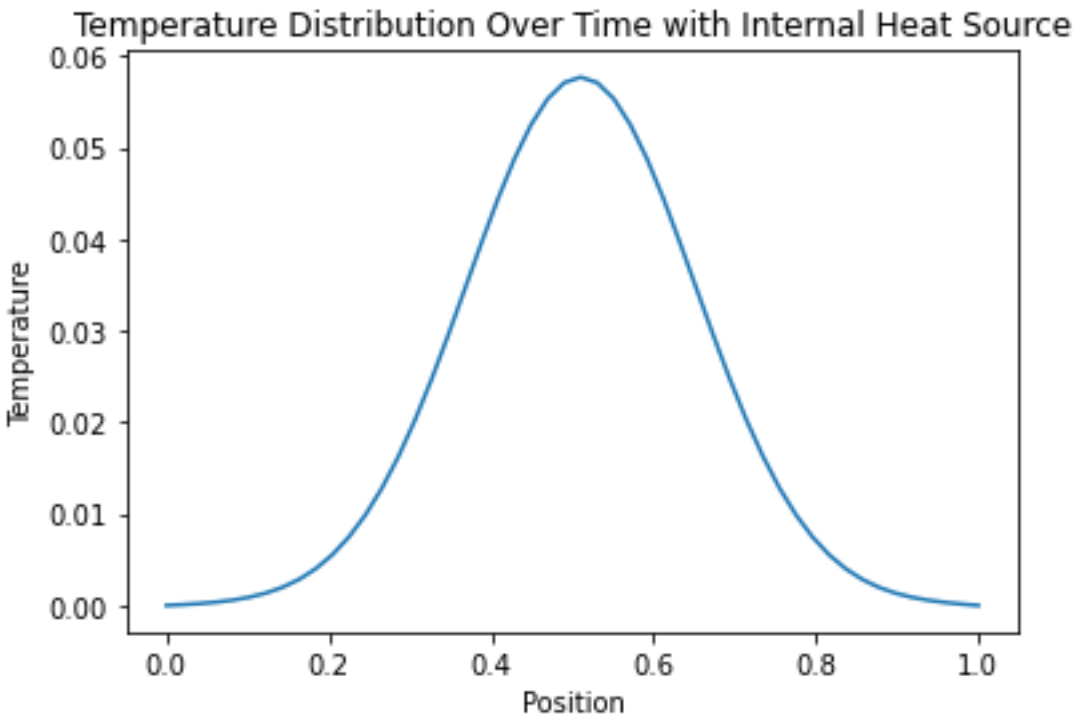
$\nabla^2 u$ is the Laplacian of temperature, which describes how temperature changes with respect to space.

S(x,t) is the internal heat source term, representing the heat generation or absorption within the material at a given position (x) and time (t).

Now, let's write a simple Python code to solve the one-dimensional heat equation with an internal heat source using the finite difference method. In this code, we will simulate heat conduction in a rod with a heat source at its center.

```
import numpy as np

import matplotlib.pyplot as plt


# Parameters

L = 1.0  # Length of the rod

T = 1.0  # Total simulation time

Nx = 50  # Number of spatial points

Nt = 500  # Number of time steps

alpha = 0.01  # Thermal diffusivity

source_position = 0.5  # Position of the internal heat source

source_strength = 1.0  # Strength of the heat source


# Spatial and temporal discretization

dx = L / (Nx - 1)
```

```
dt = T / Nt

# Initial conditions
x = np.linspace(0, L, Nx)
u = np.zeros(Nx)
u[int(Nx / 2)] = 1.0  # Initial temperature distribution with a heat source at the center

# Simulation loop
for n in range(Nt):
    u_new = u.copy()
    for i in range(1, Nx - 1):
        source = 0
        if abs(x[i] - source_position) < dx / 2:
            source = source_strength
        u_new[i] = u[i] + alpha * (u[i + 1] - 2 * u[i] + u[i - 1]) * dt / (dx**2) + source * dt
    u = u_new

# Plot the temperature distribution
plt.plot(x, u)
plt.xlabel('Position')
plt.ylabel('Temperature')
plt.title('Temperature Distribution Over Time with Internal Heat Source')
plt.show()
```



In this code, we add an internal heat source at the center of the rod (position source_position) with a specified strength (source_strength). This source term is incorporated into the finite difference method to model heat generation at that location, resulting in a temperature distribution that changes over time.

The Heat Equation with Conduction Boundary Conditions

The heat equation with conduction boundary conditions extends the basic heat equation to include constraints or conditions at the boundaries of the medium. Conduction boundary conditions specify how heat is conducted or transferred at the edges of a region. There are different types of boundary conditions, including Dirichlet, Neumann, and Robin boundary conditions, each of which describes different scenarios at the boundary. Let's consider the Dirichlet boundary condition, which specifies the temperature at the boundary points.

The one-dimensional heat equation with Dirichlet boundary conditions is expressed as:

$\partial t/\partial u = \alpha \, \partial x^2/\partial^2 u$ for $0<x<L$

With boundary conditions:

$u(0,t)=T_0$ and $u(L,t)=T_L$

Where:

$\partial t/\partial u$ is the rate of change of temperature (u) with respect to time (t).

α is the thermal diffusivity.

$\partial x^2/\partial^2 u$ is the spatial second derivative of temperature, representing the curvature of the temperature distribution with respect to space.

$u(0,t)$ is the temperature at the left boundary, set to T_0.

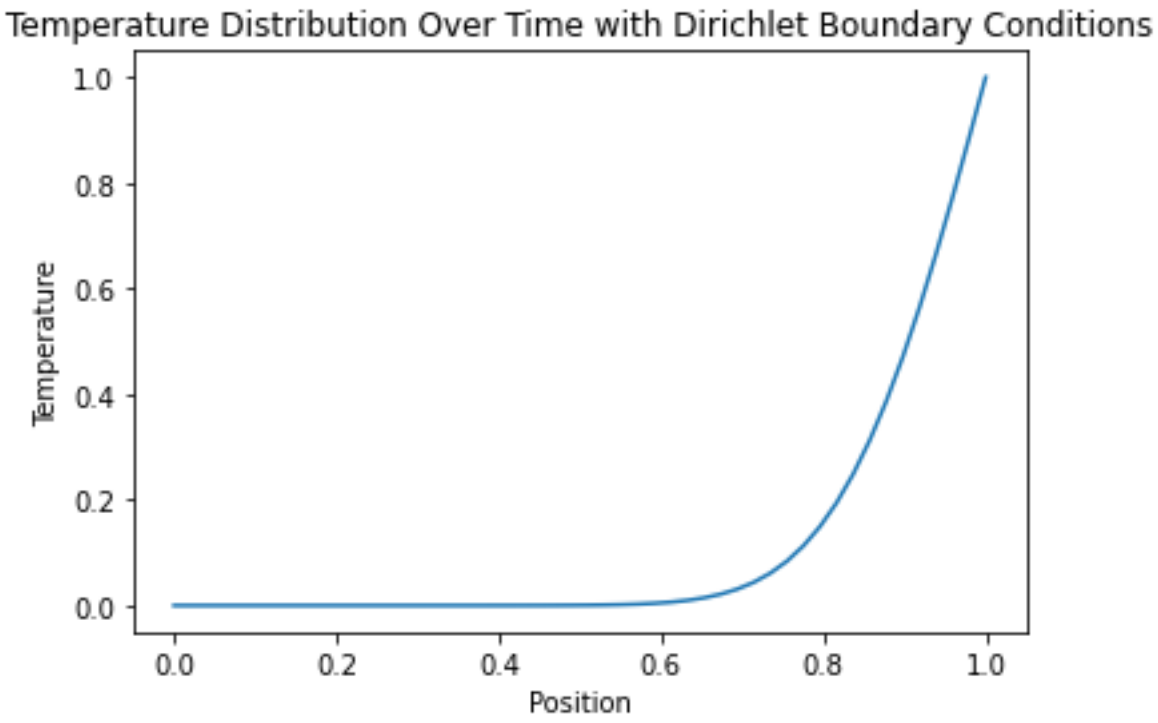
$u(L,t)$ is the temperature at the right boundary, set to T_L.

$0<x<L$ represents the spatial domain of interest.

Now, let's write a Python code to solve the one-dimensional heat equation with Dirichlet boundary conditions using the finite difference method. This code simulates the heat conduction in a rod with fixed temperatures at the boundaries.

```
import numpy as np

import matplotlib.pyplot as plt


# Parameters

L = 1.0  # Length of the rod

T = 1.0  # Total simulation time

Nx = 50  # Number of spatial points

Nt = 500  # Number of time steps

alpha = 0.01  # Thermal diffusivity
```

```
T0 = 0.0  # Temperature at the left boundary
TL = 1.0  # Temperature at the right boundary

# Spatial and temporal discretization
dx = L / (Nx - 1)
dt = T / Nt

# Initial conditions
x = np.linspace(0, L, Nx)
u = np.zeros(Nx)
u[0] = T0
u[-1] = TL

# Simulation loop
for n in range(Nt):
    u_new = u.copy()
    for i in range(1, Nx - 1):
        u_new[i] = u[i] + alpha * (u[i + 1] - 2 * u[i] + u[i - 1]) * dt / (dx**2)
    u = u_new

# Plot the temperature distribution
plt.plot(x, u)
plt.xlabel('Position')
plt.ylabel('Temperature')
plt.title('Temperature Distribution Over Time with Dirichlet Boundary Conditions')
plt.show()
```



In this code, we apply Dirichlet boundary conditions by fixing the temperature at the left and right boundaries. The temperature distribution within the rod changes over time according to the heat equation.

The Heat Equation with Convection Boundary Conditions

The heat equation with convection boundary conditions extends the basic heat equation to include conditions at the boundaries where heat is transferred not only through conduction but also through convection, which involves the movement of the heat-carrying fluid. Convection boundary conditions are relevant when you have a fluid (liquid or gas) flowing over the boundaries, which affects the heat transfer. One common convection boundary condition is the Robin boundary condition.

The one-dimensional heat equation with Robin (convection) boundary conditions is expressed as:

$\partial t/\partial u = \alpha\, \partial x^2 \partial^2 u - h(u - T_\infty)$ for $0<x<L$

With boundary conditions:

$u(0,t)=T_0$

$\partial x/\partial u(L,t) = -hu(L,t)$

Where:

$\partial t/\partial u$ is the rate of change of temperature (u) with respect to time (t).

α is the thermal diffusivity.

$\partial x^2/\partial^2 u$ is the spatial second derivative of temperature, representing the curvature of the temperature distribution with respect to space.

h is the convective heat transfer coefficient, describing the rate of heat transfer from the surface to the fluid.

u(0,t) is the temperature at the left boundary, set to T_0.

$\partial x/\partial u(L,t)$ represents the spatial derivative of temperature at the right boundary, which is proportional to the rate of heat transfer from the solid to the fluid.

u(L,t) is the temperature at the right boundary.

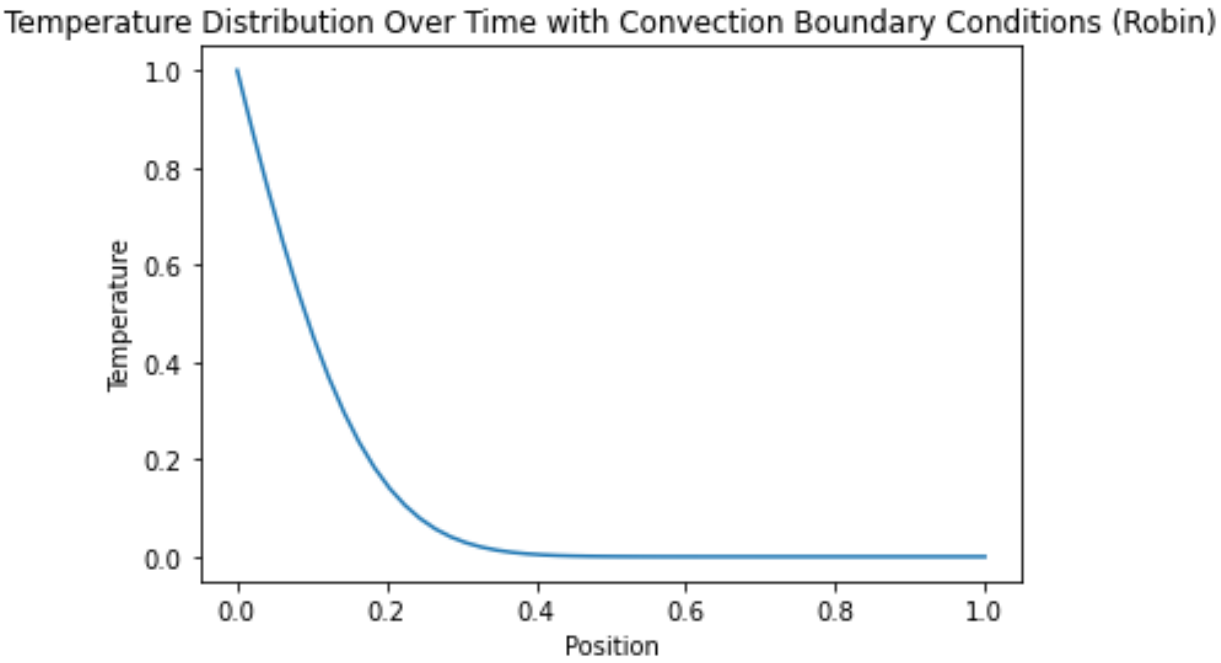
T_∞ is the ambient (fluid) temperature.

Now, let's write a Python code to solve the one-dimensional heat equation with Robin (convection) boundary conditions using the finite difference method. This code simulates heat conduction in a rod with convection boundary conditions at the right boundary.

```
import numpy as np

import matplotlib.pyplot as plt


# Parameters

L = 1.0  # Length of the rod
```

```
T = 1.0  # Total simulation time
Nx = 50  # Number of spatial points
Nt = 500  # Number of time steps
alpha = 0.01  # Thermal diffusivity
T0 = 1.0  # Temperature at the left boundary
h = 0.1  # Convective heat transfer coefficient
T_infinity = 0.0  # Ambient (fluid) temperature

# Spatial and temporal discretization
dx = L / (Nx - 1)
dt = T / Nt

# Initial conditions
x = np.linspace(0, L, Nx)
u = np.zeros(Nx)
u[0] = T0

# Simulation loop
for n in range(Nt):
    u_new = u.copy()
    for i in range(1, Nx - 1):
        u_new[i] = u[i] + alpha * (u[i + 1] - 2 * u[i] + u[i - 1]) * dt / (dx**2) - h * (u[i] - T_infinity) * dt
    u_new[-1] = u[-1] - h * (u[-1] - T_infinity) * dx * dt  # Robin boundary condition
    u = u_new

# Plot the temperature distribution
plt.plot(x, u)
plt.xlabel('Position')
plt.ylabel('Temperature')
```

```
plt.title('Temperature Distribution Over Time with Convection Boundary Conditions (Robin)')
plt.show()
```



In this code, we apply Robin (convection) boundary conditions at the right boundary, taking into account the heat transfer to the fluid, which affects the temperature distribution within the rod over time.

The Heat Equation with Radiation Boundary Conditions

The heat equation with radiation boundary conditions extends the basic heat equation to account for heat transfer by radiation at the boundaries. Radiation boundary conditions are relevant when a medium exchanges heat with its surroundings through electromagnetic radiation, such as infrared radiation. In one dimension, the boundary conditions are often modeled using Stefan-Boltzmann's law, which relates the radiative heat transfer rate to the temperature difference between the object and its surroundings.

The one-dimensional heat equation with radiation boundary conditions is expressed as:

$\partial t/\partial u = \alpha\, \partial x^2 \partial^2 u - \sigma(u - T_\infty)^4$ for $0<x<L$

With boundary conditions:

$\partial x/\partial u(0,t)=0$

$\sigma(u(L,t)-T_\infty)^4=\epsilon\sigma(T_\infty{}^4-u(L,t)^4)$

Where:

$\partial t/\partial u$ is the rate of change of temperature (u) with respect to time (t).

α is the thermal diffusivity.

$\partial x^2/\partial^2 u$ is the spatial second derivative of temperature.

σ is the Stefan-Boltzmann constant.

u(0,t) represents the temperature at the left boundary, and the condition $\partial x/\partial u(0,t)=0$ ensures no heat flow across the left boundary.

u(L,t) is the temperature at the right boundary.

T_∞ is the ambient temperature.

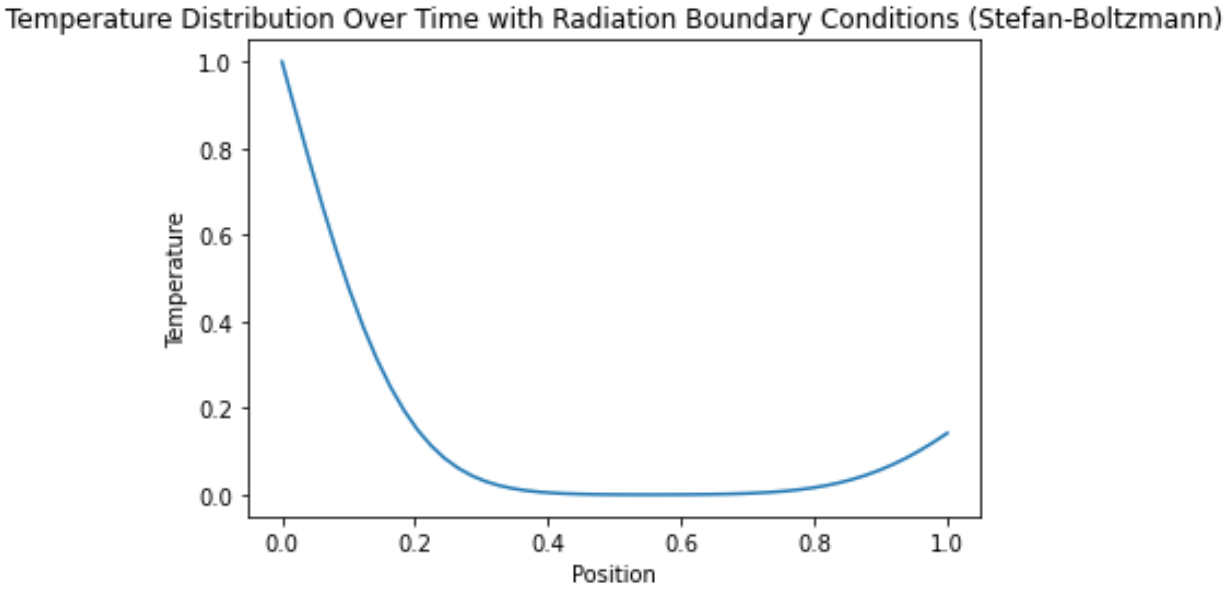
ϵ is the emissivity, representing the ability of the material to emit radiation.

Now, let's write a Python code to solve the one-dimensional heat equation with radiation boundary conditions using the finite difference method. This code simulates heat conduction in a rod with radiation boundary conditions at the right boundary.

```
import numpy as np

import matplotlib.pyplot as plt


# Parameters

L = 1.0  # Length of the rod

T = 1.0  # Total simulation time

Nx = 50  # Number of spatial points
```

```
Nt = 500  # Number of time steps
alpha = 0.01  # Thermal diffusivity
T0 = 1.0  # Temperature at the left boundary
T_infinity = 2.0  # Ambient temperature
epsilon = 0.9  # Emissivity
sigma = 5.67e-8  # Stefan-Boltzmann constant

# Spatial and temporal discretization
dx = L / (Nx - 1)
dt = T / Nt

# Initial conditions
x = np.linspace(0, L, Nx)
u = np.zeros(Nx)
u[0] = T0

# Simulation loop
for n in range(Nt):
    u_new = u.copy()
    for i in range(1, Nx - 1):
        u_new[i] = u[i] + alpha * (u[i + 1] - 2 * u[i] + u[i - 1]) * dt / (dx**2)
    u_new[-1] = ((sigma * epsilon * T_infinity**4 + u[-1]**4) / (sigma * epsilon + 1))**(1/4)  # Radiation boundary condition
    u = u_new

# Plot the temperature distribution
plt.plot(x, u)
plt.xlabel('Position')
plt.ylabel('Temperature')
```

```
plt.title('Temperature Distribution Over Time with Radiation Boundary Conditions (Stefan-Boltzmann)')

plt.show()
```



In this code, we apply the radiation boundary condition at the right boundary based on Stefan-Boltzmann's law, allowing for radiative heat transfer between the rod and its surroundings. This condition adjusts the temperature at the boundary to account for radiation heat exchange.

Contenido

e-ISSN: 2661-6505