MATHEMATICAL MODELING IN MATLAB AND PYTHON FOR THE HEAT CONDUCTION EQUATION WITH HOMOGENEOUS BOUNDARY CONDITIONS AND VARIABLE SOURCE FUNCTION

MATEMĀTISKĀ MODELĒŠANA MATLAB UN PYTHON VIDĒ SILTUMA VADĪŠANAS VIENĀDOJUMAM AR HOMOGĒNIEM ROBEŽNOSACĪJUMIEM UN MAINĪGU AVOTA FUNKCIJU

Author: **Kaspars Dortāns**, e-mail: kd20048@edu.rta.lv, phone: +371 27138117
Scientific supervisor: **Ilmārs Kangro, Mg.math., Dr.paed., associated professor,** e-mail: ilmars.kangro@rta.lv
Rēzeknes Tehnoloģiju akadēmija,
Atbrīvošanas aleja 115, Rēzekne, Latvija

Abstract. *In this paper, we consider solving the initial-boundary value problem of the second-order partial differential equation for the heat transfer equation with variable heat source function. The problem due to the approximation of the second-order derivatives by the finite differences is reduced to the initial value problem depending on one variable – time t. Some numerical results and their characteristics – figures are obtained.*

Keywords: *finite difference scheme, heat transfer equation, initial boundary value problem, MATLAB, Python.*

Introduction

Applications of one-dimensional and two-dimensional initial-boundary value problems of second-order partial differential equations are very diverse - in electromagnetism, computer modeling, physical, chemical, etc. in process simulation, as well as mathematics in the theory of ordinary and partial differential equations. The paper examines the two-dimensional initial-boundary value problem of the second-order partial differential equation for the heat conduction equation and the process of its numerical solution. Knowledge of the computer program MATLAB and the programming language Python was acquired to use their capabilities in solving the boundary problem. In the environment of these programs, programs were successfully developed, which, based on the created mathematical apparatus, find the solution of the boundary problem, and output the result in numerical form and graphically - in the form of drawings.

Materials and methods

Initial - boundary value problem

We consider the initial - boundary value problem (IBVP) for the heat transfer equation in the following form (1):

$$\begin{cases} \frac{\partial u(x,t)}{\partial t} = \frac{\partial}{\partial x}\left(k\frac{\partial u(x,t)}{\partial x}\right) + f(x), \\ u(-L,t) = 0, u(L,t) = 0, t \in \left(0, t_f\right), \\ u(x,0) = 0, x \in (-L, L), \end{cases} \tag{1}$$

$u(x,t)$ - temperature, depending on x -coordinate and time t (both are dimensionless quantities),

$k > 0$ - the constant parameter, t_f is the final time, f - given function (the source function).

It can be considered that in the interval $x \in (-L, L)$ there is, for example, a metal rod to which heat is supplied according to a certain law characterized by the source function $f(x)$.
We consider a uniform grid in the space $x_j = jh, j = \overline{0, N}, Nh = 2L.$

https://doi.org/10.17770/het2024.28.8255

We apply the finite differences of the second order of approximation [1]:

$$\frac{\partial^2 u(x_i,t_n)}{\partial x^2} \approx \frac{u(x_{i-1},t_n)-2u(x_i,t_n)+u(x_{i+1},t_n)}{h^2}$$

Applying the aforementioned approximation, the boundary value problem (1) is reduced to a one-dimensional (dependent on variable t) initial value problem. Therefore, to find the initial value problem corresponding to the boundary value problem (1) in matrix form, we first consider the following one-dimensional initial value problem (IVP) [2]:

$$\begin{cases} \dot{u}(t) = -k_1 u(t) + f(t), \\ \quad u(0) = u_0, \end{cases} \tag{2}$$

The solution of the homogeneous equation of the IVP (2) is in the form [3]:

$$u_1(t) = Ce^{-k_1 \cdot t}$$

Then, applying the constant variation method, the solution of IVP (2) is in the form [1, 3]

$$u(t) = C(t)e^{-k_1 \cdot t} \tag{3}$$

Then, inserting the expression (3) into the equation of the IVP (2), we get

$$\dot{C}e^{-k_1 \cdot t} + C(-k_1)e^{-k_1 \cdot t} + Ck_1 e^{-k_1 \cdot t} = f,$$

or

$$\dot{C}e^{-k_1 \cdot t} = f, \tag{4}$$

where kur $\dot{C} = \dot{C}(t), f = f(t)$.

Integrating both sides of equation (4) over the variable t gives

$$C(t) = \frac{f}{k_1} e^{k_1 \cdot t} + C_1. \tag{5}$$

Inserting the $C(t)$ expression (5) into the expression (3) gives

$$u(t) = \frac{f}{k_1} + C_1 e^{-k_1 \cdot t}. \tag{6}$$

To calculate the constant C_1, the initial condition of the problem (2) $u(0) = u_0$ is applied to the expression (6):

$$u_0 = \frac{f}{k_1} + C_1,$$

and we obtain

$$C_1 = u_0 - \frac{f}{k_1}.$$

Then the function $C(t)$ (5) has the form:

$$C(t) = \frac{f}{k_1} e^{k_1 \cdot t} + \left(u_0 - \frac{f}{k_1}\right). \tag{7}$$

Inserting the obtained expression of the function $C(t)$ into expression (3), the solution of the one-dimensional initial value problem (2) is obtained in the following form:

$$u(t) = u_0 e^{-k_1 \cdot t} + \frac{f}{k_1}(1 - e^{-k_1 \cdot t}) \tag{8}$$

Based on the obtained one-dimensional initial value problem solution (8), the initial value problem corresponding to the boundary value problem (1) in vector form is [2, 4]:

$$\begin{cases} \dot{U}(t) + AU(t) = F(t), \\ \quad U(0) = U_0, \end{cases} \tag{9}$$

The solution of (9) is in the form:

$$U(t) = U_0 e^{-At} + (E - e^{-At})A^{-1}F, \tag{10}$$

where A is a 3-diagonal matrix of order $N-1$ with multiplier k, A^{-1} is the inverse matrix of matrix A, E is the unit matrix of order *N-1*, $U(t), \dot{U}(t), U_0, F(t)$ are the column-vectors of $N-1$ order with elements $u_j(t) \approx u(x_j, t)$.

The initial value problem (9) was solved using MATLAB [5] and Python.

MATLAB

MATLAB is short for Matrix Laboratory. It is a high-level programming language and interactive environment primarily designed for numerical computing. It's widely used in academia, engineering, and industry for a variety of applications including robotics, signal processing, image processing, control systems, machine learning, and more. [6]

At its core, MATLAB is suited for computation, data analysis, and data visualization. MATLAB offers extensive built-in functions and tools for numerical computation and data visualization.

MATLAB is interactive, it provides GUI controls and a command-line interface. Users can execute commands or scripts and see their results immediately, which allows for faster development, testing, and experimentation.

Additionally, MATLAB supports the creation of graphical user interfaces (GUIs) for building user-friendly applications [7]. Users can convert scripts into simple applications. Applications can be built interactively and programmatically.

It is possible to use MATLAB with other programming languages as it provides two-way integration with programming languages like C/C++, Java, and Python, thus enabling users to incorporate their MATLAB algorithms into other software systems. Or utilize existing code in MATLAB environment.

MATLAB also serves as a platform for machine learning (ML) research and application development. It is capable of ML tasks such as classification, regression, clustering, dimensionality reduction, and more.

Python

Python is interpreted object-oriented dynamically typed programming language [8]. It is used to build a wide range of applications, from web development and scripting to scientific computing and data analysis. One of its main principles is clear and concise syntax, Python promotes code readability and maintainability. Python has the standard library which provides a set of modules and functions for diverse tasks. Python's popularity within the scientific community is increased by libraries such as NumPy, SciPy, and Matplotlib, which offer powerful tools for numerical computing and visualization.

There can be found frameworks that are built using Python such as Django and Flask for web development, PyTorch and TensorFlow for machine learning, and pandas for data manipulation. A wide range of frameworks and tool packages allows for Python to be a versatile tool for many applications.

Python is interpreted, which means that Python code is executed line-by-line, without the need for compilation into machine code beforehand. This can speed up development and make experimentation easier as developers can execute code snippets and receive immediate feedback. On the other hand, the interpreted nature of Python decreases its performance, making it less performant than compiled languages.

Results and discussion

Solving initial-boundary value problems in MATLAB and Python

Task description. A homogenous thin rod with a given starting temperature. For all x values in the interval (-L, L) with start time t=0 temperature changes by formula U(x, 0) = f(x). Time changes in interval (0, t_f).

Find out the temperature distribution for all x coordinates in given x and time intervals.

The solution function was implemented with 4 parameters: chosen x coordinate count; k constant; f0 heat source force; and t time points at which to compute temperature distribution.

In solution 2 heat source distributions were chosen – evenly distributed and exponentially distributed heat sources.

Equation (9) was implemented and used to calculate the temperature for each given x coordinate at each time moment.

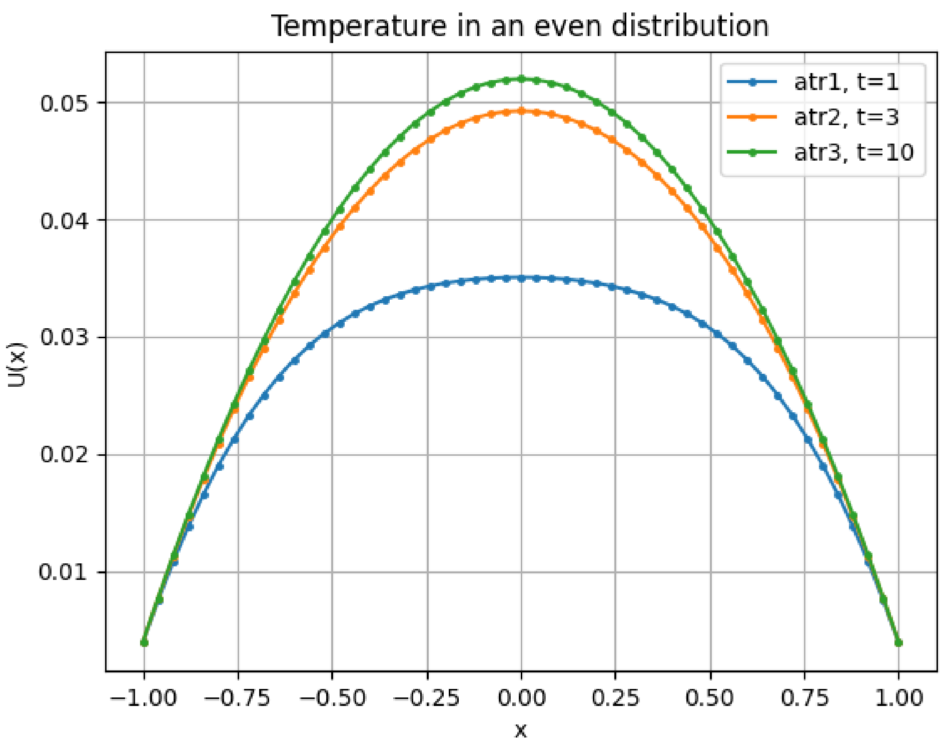


Fig.1. **Temperature distribution dependence on x with fixed temperature**

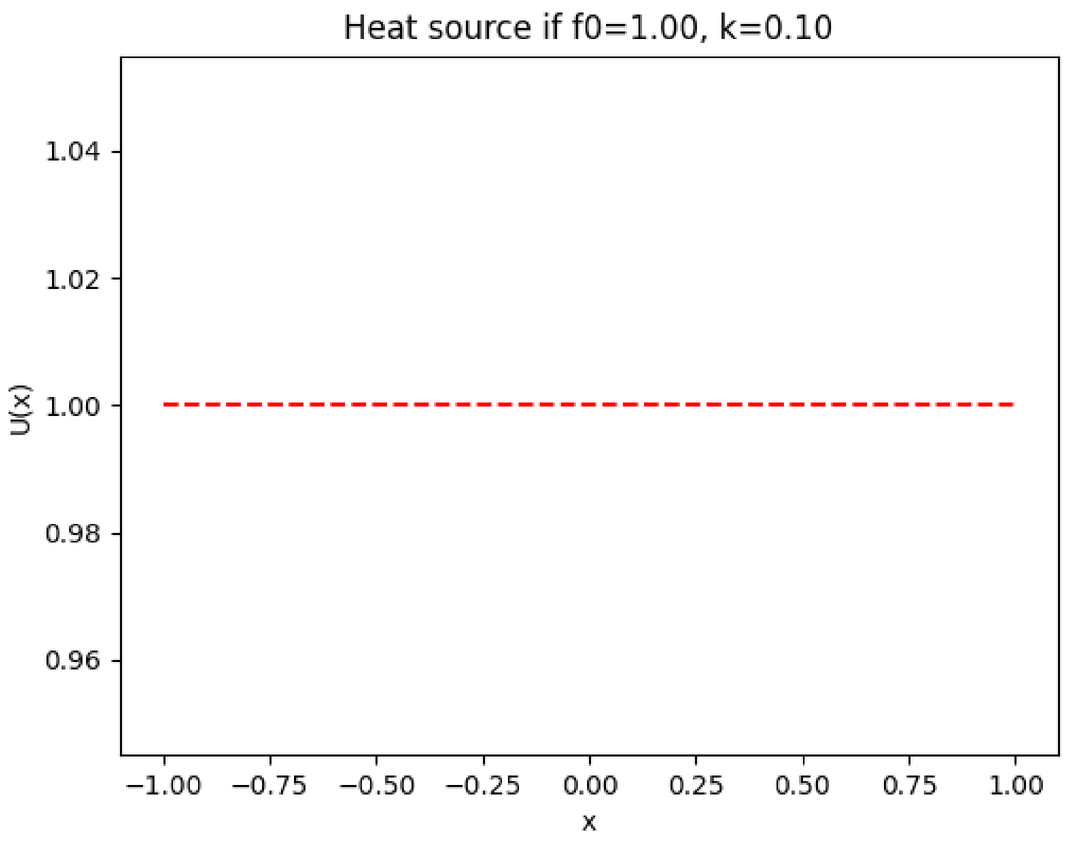


Fig.2. **Even heat source dependence on x**

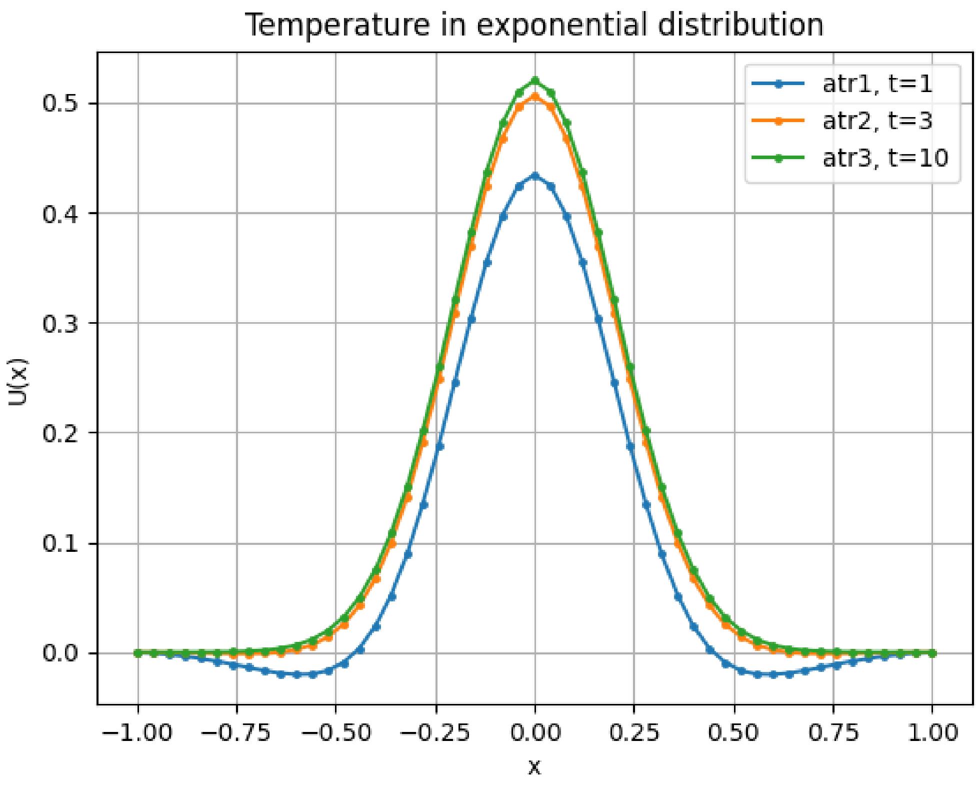


Fig.3. **Temperature distribution dependence on x with fixed temperature**

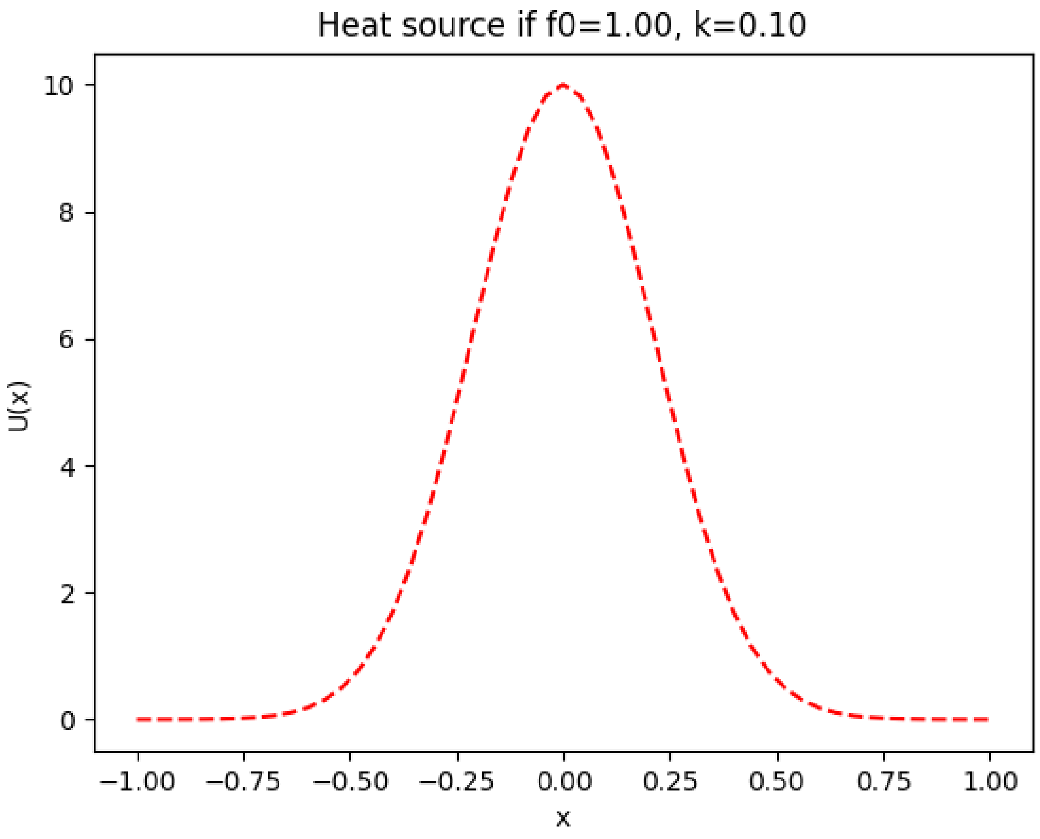
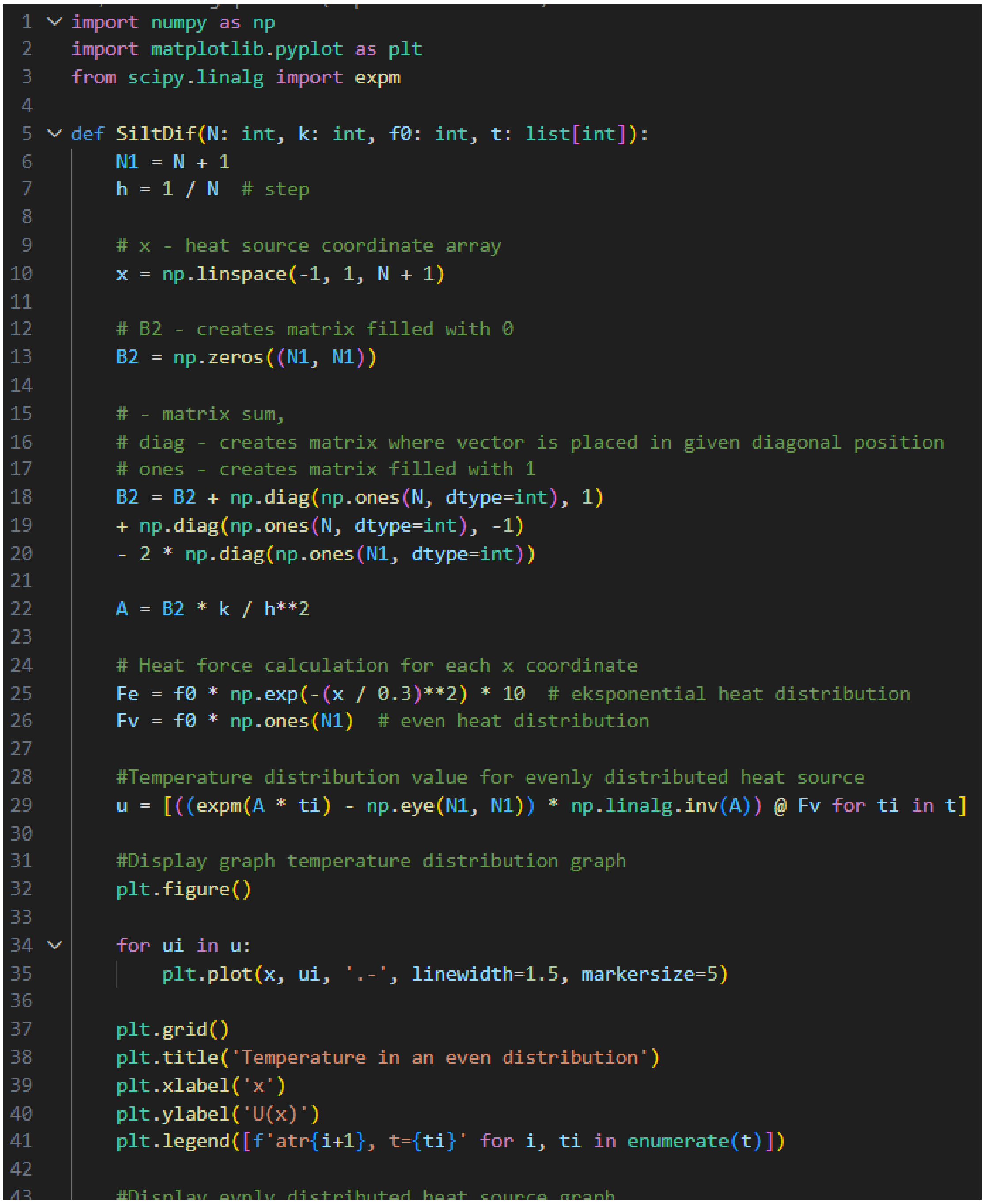


Fig.4. **Exponential heat source dependence on x**

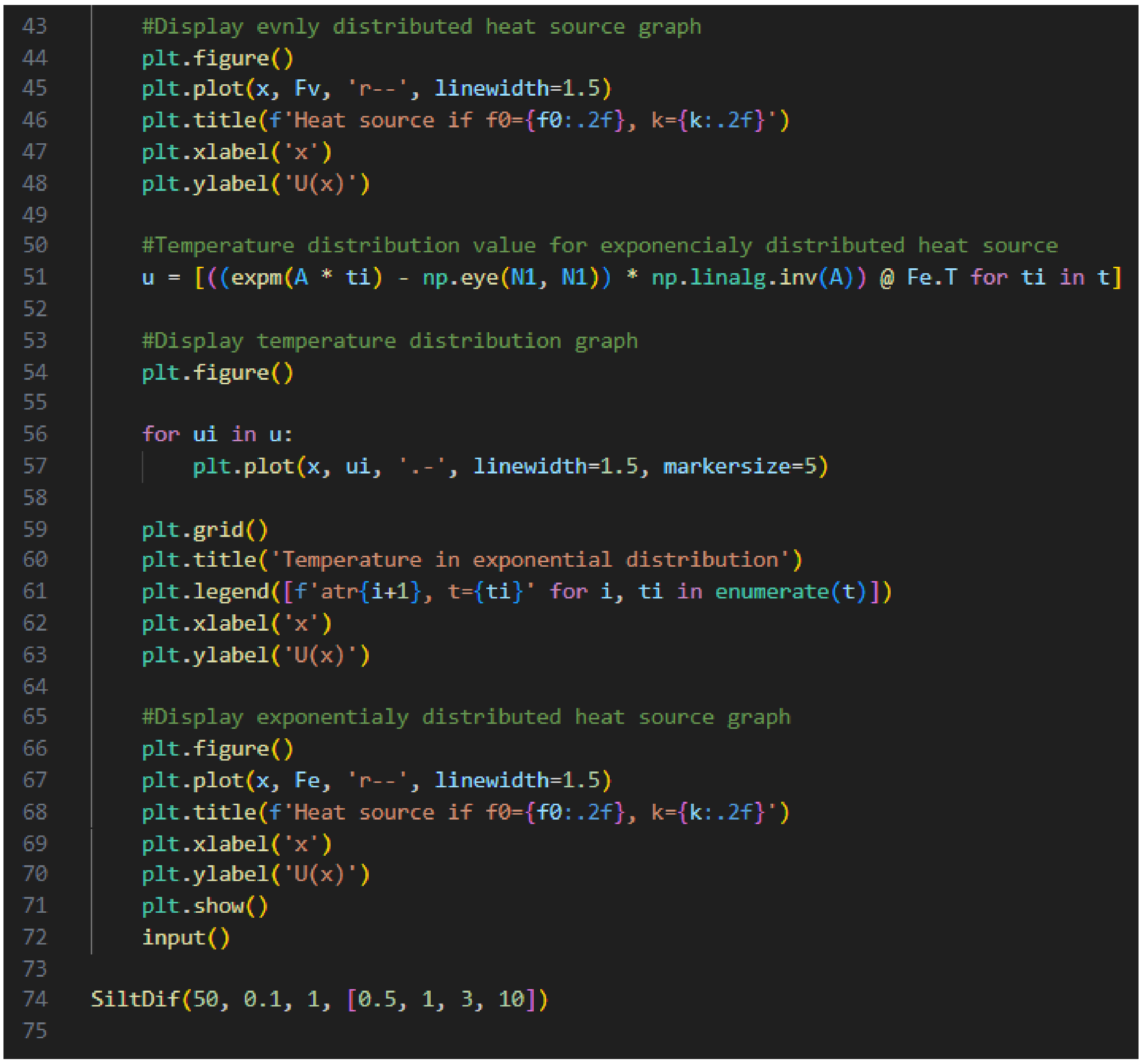
Python solution source code

Figures 5. and 6. are displayed Python solution source code.



```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.linalg import expm

def SiltDif(N: int, k: int, f0: int, t: list[int]):
    N1 = N + 1
    h = 1 / N  # step

    # x - heat source coordinate array
    x = np.linspace(-1, 1, N + 1)

    # B2 - creates matrix filled with 0
    B2 = np.zeros((N1, N1))

    # - matrix sum,
    # diag - creates matrix where vector is placed in given diagonal position
    # ones - creates matrix filled with 1
    B2 = B2 + np.diag(np.ones(N, dtype=int), 1)
    + np.diag(np.ones(N, dtype=int), -1)
    - 2 * np.diag(np.ones(N1, dtype=int))

    A = B2 * k / h**2

    # Heat force calculation for each x coordinate
    Fe = f0 * np.exp(-(x / 0.3)**2) * 10  # eksponential heat distribution
    Fv = f0 * np.ones(N1)  # even heat distribution

    #Temperature distribution value for evenly distributed heat source
    u = [((expm(A * ti) - np.eye(N1, N1)) * np.linalg.inv(A)) @ Fv for ti in t]

    #Display graph temperature distribution graph
    plt.figure()

    for ui in u:
        plt.plot(x, ui, '.-', linewidth=1.5, markersize=5)

    plt.grid()
    plt.title('Temperature in an even distribution')
    plt.xlabel('x')
    plt.ylabel('U(x)')
    plt.legend([f'atr{i+1}, t={ti}' for i, ti in enumerate(t)])

    #Display evnly distributed heat source graph
```

Fig.5. Python **solution source code**



```python
    #Display evnly distributed heat source graph
    plt.figure()
    plt.plot(x, Fv, 'r--', linewidth=1.5)
    plt.title(f'Heat source if f0={f0:.2f}, k={k:.2f}')
    plt.xlabel('x')
    plt.ylabel('U(x)')

    #Temperature distribution value for exponencialy distributed heat source
    u = [((expm(A * ti) - np.eye(N1, N1)) * np.linalg.inv(A)) @ Fe.T for ti in t]

    #Display temperature distribution graph
    plt.figure()

    for ui in u:
        plt.plot(x, ui, '.-', linewidth=1.5, markersize=5)

    plt.grid()
    plt.title('Temperature in exponential distribution')
    plt.legend([f'atr{i+1}, t={ti}' for i, ti in enumerate(t)])
    plt.xlabel('x')
    plt.ylabel('U(x)')

    #Display exponentialy distributed heat source graph
    plt.figure()
    plt.plot(x, Fe, 'r--', linewidth=1.5)
    plt.title(f'Heat source if f0={f0:.2f}, k={k:.2f}')
    plt.xlabel('x')
    plt.ylabel('U(x)')
    plt.show()
    input()

SiltDif(50, 0.1, 1, [0.5, 1, 3, 10])
```

Fig.6. **Python solution source code**

Conclusions

1. The main MATLAB and Python operators for solving the initial boundary value problem of the heat conduction equation were learned.
2. The solution of the considered initial-boundary value problem, taken as a whole, was performed numerically, that is, by replacing the second-order partial derivatives of the variable x with finite differences. In contrast, the initial value problem, corresponding to the second variable - t (temperature), was solved using the analytical solution found for the initial value problem solution.
3. Both MATLAB and Python were used to construct valid solutions to a given problem.
4. MATLAB in itself provides a code editor, a built-in debugger with interactive controls, and provides many mathematical computational methods and data visualization methods already built in.
5. On the other hand Python, while more freely available, does not have as many mathematical calculation and data visualization functions built-in, because it is a programming language. Nevertheless, many packages containing necessary tools for mathematical computations and data visualization can be easily found.
6. In the didactic aspect, it should be concluded that the solution of the boundary problem of the second-order partial differential equation considered in the work expands the range of

knowledge about the issues of the theory of differential equations in general, and provides practical skills for solving specific tasks/problems, using one of the most powerful and complete computer-mathematical tools used in the teaching process and scientific research systems MATLAB and the high-level programming language Python.

Summary

The initial boundary value problem of the second-order partial differential equation for the heat transfer equation with variable heat source function is considered in its theoretical and practical realization (performance).

At first, a two-dimensional initial-boundary value problem of the second-order partial differential equation for the heat conduction equation was reduced to a one-dimensional initial value problem depending on one variable – time t.

The second problem under question was obtaining the analytical solution to the above-mentioned initial value problem.

Initially, the analytical solution of the one-dimensional problem was obtained, and then, based on the theoretical principles, by generalizing the obtained solution, the analytical solution was obtained in the form of vectors. Next, an initial boundary value problem was solved numerically using MATLAB and Python - obtained results in numerical form and graphically - in the form of drawings, depicting the temperature distribution depending on the considered interval point at different values of time t.

Bibliography

1. Kalis, H. *Skaitliskās metodes.*- Rīga, 2008. -185 lpp.
2. H. Kalis I. Kangro and A. Gedroics, “Numerical methods of solving some nonlinear heat transfer problems”, “Int. Journ. of Pure and Applied Mathematics”, vol. 57, No. 4, 2009, pp. 467-484.
3. Braun, M. Differential Equations and Their Applications: An Introduction to Applied Mathematics. -2nd Edition. - N.Y.: John Wiley and Sons, 1978. 518 p.
4. H. Kalis and A. Buikis, “Method of lines and finite difference schemes with the exact spectrum for solution the hyperbolic heat conduction equation,” Mathematical Modelling and Analysis”, vol. 16, No 2, 2011, pp. 220-232
5. Kalis, H., Kangro, I. (2010). *Datorprogrammas MATLAB lietošana matemātikas mācību procesā. Mācību līdzeklis.* Rēzekne: RA Izdevniecība, 2010, 264 lpp.
6. About MATLAB. [Viewed 23.04.2024] Available: https://www.mathworks.com/products/matlab.html
7. About MATLAB GUI build tools [Viewed 23.04.2024] Available: https://www.mathworks.com/discovery/matlab-gui.html
8. Python guide [Viewed 23.04.2024] Available: https://wiki.python.org/moin/BeginnersGuide