МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Кваліфікаційна наукова
праця на правах рукопису

Сокол Костянтин Ігорович

УДК 004.94:697.1

ДИСЕРТАЦІЯ

МОДЕЛІ ТА МЕТОДИ КОМП'ЮТЕРНОГО МОДЕЛЮВАННЯ ПРОЦЕСІВ ТЕПЛОПЕРЕДАЧІ В ПАСИВНИХ СИСТЕМАХ ОПАЛЕННЯ

122 Комп'ютерні науки
12 Комп'ютерні науки

Подається на здобуття наукового ступеня доктора філософії
Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело
_____ К.І. Сокол

Науковий керівник
Огнєва Оксана Євгенівна
кандидат технічних наук, доцент

Хмельницький 2024

АНОТАЦІЯ

Сокол К.І. **Моделі та методи комп'ютерного моделювання процесів теплопередачі в пасивних системах опалення. – Кваліфікаційна наукова праця на правах рукопису.**

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 122 Комп'ютерні науки, Херсонський національний технічний університет, Міністерство освіти і науки України, Хмельницький, 2024.

У роботі досліджено теоретичні основи процесів теплопередачі: теплопровідності, конвекції, випромінювання, теплоємності. Для кожного з процесів, у ході роботи, створюється математично обґрунтований програмний блок. Із сукупності таких блоків утворюється система, дослідження якої є метою виконання роботи.

Житлова будівля, а також, пасивна система опалення в ній, розглядається в роботі як об'єкт моделювання та дослідження з точки зору теплових процесів, що відбуваються всередині. Представлені структурні схеми окремих складових досліджуваної системи, що складаються з блоків, кожен із яких представляє окремий процес теплопередачі.

Розглянуто методи дослідження процесів теплопередачі, що активно використовують комп'ютерні системи: метод кінцевих елементів, кінцевих різниць, кінцевих об'ємів, а також, інженерні методи, зокрема, мережу термічних опорів. Проаналізувавши переваги та недоліки всіх методів, вирішено частково інтегрувати переваги числових методів у мережі теплових опорів у новому розробленому методі для розрахунку теплових характеристик будівлі.

Продемонстровано створення нового методу для розрахунку теплових та енергетичних характеристик будівлі з пасивною системою опалення. Даний метод є поєднанням числових та інженерних методів розрахунку, що дозволяє йому мати високу швидкість та точність виконання обчислень, а також, отримати можливість дослідження системи на певному, короткому відрізку часу і побачити переваги й недоліки системи при заданих вхідних умовах. Усередині методу показана схема

оптимізації параметрів пасивної системи опалення за показниками енергетичних потреб будівлі.

Показано реалізацію комп'ютерної моделі за допомогою мови програмування «Python». Пояснено використання та порядок виклику всіх необхідних функцій.

Доказано точність створеного методу. Виконане порівняння результатів розрахунку за створеним методом і за методом, представленим у ДСТУ Б А.2.2-12 2015. У результаті, подібність за енергетичною потребою кондиціонування склала, в середньому, 86 відсотків, за енергетичною потребою опалення – 87 відсотків, за загальною енергетичною потребою – 89 відсотків, що дозволяє вважати його ефективність достатньою для використання при розрахунках теплових та енергетичних характеристик будівлі.

Продемонстровано результати роботи програми. За результатами оптимізації, товщина скління зовнішнього шару стіни Тромбе обрана максимальною абсолютно для всіх будівель, що означає, що теплова ізоляція важливіша для мінімізації енергетичних потреб, ніж прозорість і максимізація нагріву внутрішньої маси будівлі від сонячного випромінювання. Товщина повітряного прошарку обрана мінімальною для всіх будівель, що пояснюється тим, що менший об'єм повітря існує між стіною та склінням – тим більше він прогрівається під дією сонячного випромінювання. Товщина масивної стіни обрана з середини масиву для всіх будівель, крім однієї, що демонструє прагнення системи знайти баланс між витратами на кондиціонування та опалення, адже, чим тонша масивна стіна – тим менша енергетична потреба кондиціонування, чим вона товща – тим менша енергетична потреба опалення.

За результатами дослідження, отримано досить високу середню економію енергоресурсів за використання пасивної системи опалення типу стіна Тромбе, більше 1500 кВт×год на рік, в основному, за рахунок зниження енергетичних потреб кондиціонування, більше, ніж на 1000 кВт×год на рік, що відповідає 36 відсоткам. Витрати на опалення, у свою чергу, скорочуються більше, ніж на 450 кВт×год на рік, проте, у відсотковому еквіваленті це значення досягає всього 6

відсотків. Це означає, що в кліматі обраної місцевості, пасивна система опалення типу стіна Тромбе проявляє свою ефективність, в основному, не нагріваючи внутрішню масу будівлі, а попереджаючи перегрів будівлі в теплий період року.

Ключові слова: процеси теплопередачі, пасивні системи опалення, комп'ютерна система.

Список публікацій:

1. Андронова О.В., Курак В.В., Сокол К.І. Тепловий режим будівлі з пасивною системою опалення.О.В. *Вісник Херсонського національного технічного університету. Інженерні науки*. № 1(72), Ч. 1. ХНТУ. 2020. с. 9-17.
URL: https://journals.kntu.net.ua/index.php/visnyk/article/view/467/493
DOI: https://doi.org/10.35546/kntu2078-4481.2020.1.1.1

2. Сокол К.І., Огнєва О.Є. Розробка системи комп'ютерного моделювання для оптимізації параметрів пасивної системи опалення закритого типу. *Проблеми інформаційних технологій*. 2020. № 27. С. 69-77.
URL: https://journals.kntu.net.ua/index.php/pit/article/view/705
DOI: https://doi.org/10.35546/2313-0687.2020.27.69-77

3. Сокол К.І., Огнєва О.Є., Андронова О.В. Проектування комп'ютерної системи для дослідження теплового режиму будівлі з пасивною системою опалення закритого типу. *Вісник Херсонського національного технічного університету. Інформаційні технології*. № 1(80). ХНТУ. 2022. с. 61-67.
URL: https://ojs.kntu.net.ua/index.php/visnyk/article/view/236
DOI: https://doi.org/10.35546/kntu2078-4481.2022.1

4. Сокол К.І. Програмна реалізація методу розрахунку енергопотреб будівлі. *Вісник Хмельницького національного університету. Серія: технічні науки.* № 335(3.1). ХНУ. 2024. с. 209-215.
URL: https://heraldts.khmnu.edu.ua/index.php/heraldts/article/view/229
DOI: https://doi.org/10.31891/2307-5732-2024-335-3-29

Апробації результатів дисертації:

1. Сокол К.І., Огнєва О.Є. Використання SMART GRIDS у системи опалення будівель / К.І. Сокол, О.Є. Огнєва // Матеріали всеукраїнської науково-технічної

конференції «Інформаційні технології та програмування», 30.11.2020р. – Херсон:ХНТУ, 2020. – с.86-87.

2. Сокол К.І., Огнєва О.Є. Проектування комп'ютерної системи для імітації роботи та дослідження ефективності пасивної системи опалення / К.І. Сокол, О.Є. Огнєва // Матеріали II Всеукраїнської науково-практичної конференції молодих вчених, аспірантів і студентів «Молодь і наука», 24.05.2021р.- Херсон:ХНТУ, 2020.- с.12-13.

3. Sokol K. Ohnieva O. Development of the structural scheme for passive closed type passive heating system's parameters optimization / Інтелектуальні системи прийняття рішень і проблеми обчислювального інтелекту: Матеріали міжнар. наук. конф., с. Залізний Порт, 25-29 травня 2021 р. – Херсон: Видавництво ФОП Вишемирський В. С., 2021. – с.111-113.

4. Сокол К., Огнєва О. Проектування додатку для розрахунку показників моделі будівлі з пасивною системою опалення закритого типу / К. Сокол., О. Огнєва // Матеріали Всеукраїнської науково-технічна конференція «Інформаційні технології та програмування», 30.11.2021. – Херсон: ХНТУ, 2021.

5. Сокол К., Огнєва О. Концепція додатку для проектування будівель з пасивними системами опалення / К. Сокол., О. Огнєва // Матеріали міжнародної науково-практичної конференції «Синергія науки і бізнесу у повоєнному відновленні Херсонщини», 26-28.04.23. Видавництво Олді+., 2023. – с. 207-211.

6. Sokol K., Ohnieva O., Dostalek L. Determining the effectiveness of using passive closed-loop heating systems to prevent the risk of building overcooling during energy system stability issues.. International Workshop on Computational & Information Technologies for Risk-Informed Systems, December 21-22, 2023.

ABSTRACT

Sokol K.I. **Models and methods of computer simulation of heat transfer processes in passive heating systems. – Qualifying scientific work as a manuscript.**

Dissertation for the degree of Doctor of Philosophy in the specialty 122 Computer Science, Kherson National Technical University, Ministry of Education and Science of Ukraine, Khmelnytskyi, 2024.

The paper examines the theoretical foundations of heat transfer processes: thermal conductivity, convection, radiation, heat capacity. For each of the processes, during the work, a mathematically justified program block is created. A system is formed from the set of such blocks, the study of which is the goal of the work.

The residential building, as well as the passive heating system in it, is considered in the work as an object of modeling and research from the point of view of thermal processes occurring inside. Structural diagrams of individual components of the studied system consisting of blocks, each of which represents a separate heat transfer process, are presented.

The methods of research of heat transfer processes that are actively used by computer systems are considered: the method of finite elements, finite differences, finite volumes, as well as engineering methods, in particular, a network of thermal resistances. Having analyzed the advantages and disadvantages of all methods, it was decided to partially integrate the advantages of numerical methods in the network of thermal resistances in the newly developed method for calculating the thermal characteristics of the building.

The creation of a new method for calculating the thermal and energy characteristics of a building with a passive heating system is demonstrated. This method is a combination of numerical and engineering methods of calculation, which allows it to have high speed and accuracy of calculations, as well as to get the opportunity to study the system in a certain, short period of time and see the advantages and disadvantages of the system under the given input conditions. Inside the method, a scheme for optimizing the parameters of the passive heating system according to the indicators of the energy needs of the building is shown.

The implementation of the computer model using the "Python" programming language is shown. The use and order of calling all necessary functions is explained.

The accuracy of the created method has been proven. A comparison of the results of the calculation based on the created method and the method presented in DSTU B A.2.2-12 2015 was performed. As a result, the similarity for the energy demand of air conditioning was, on average, 86 percent, for the energy demand for heating - 87 percent, for the total energy demand - 89 percent, which allows us to consider its efficiency sufficient for use in calculating the thermal and energy characteristics of the building.

The results of the program are demonstrated. According to the optimization results, the glazing thickness of the outer layer of the Trombe wall is chosen to be the maximum for absolutely all buildings, which means that thermal insulation is more important for minimizing energy needs than transparency and maximizing heating of the building's internal mass from solar radiation. The thickness of the air layer is chosen to be the minimum for all buildings, which is explained by the fact that the smaller volume of air exists between the wall and the glazing - the more it will heat up under the influence of solar radiation. The thickness of the solid wall is chosen from the middle of the massif for all buildings except one, which demonstrates the system's desire to find a balance between air conditioning and heating costs, because the thinner the solid wall, the lower the energy demand for air conditioning, and the thicker it is, the lower the energy demand for heating.

According to the results of the study, a rather high average saving of energy resources was obtained for the use of a passive heating system of the Trombe wall type, more than 1500 kWh per year, mainly due to a reduction in the energy needs of air conditioning, more than 1000 kWh per year, which corresponds to 36 percent. Heating costs, in turn, are reduced by more than 450 kWh per year, however, in percentage equivalent, this value reaches only 6 percent. This means that in the climate of the selected area, the passive heating system of the Trombe wall type shows its effectiveness, mainly not by heating the internal mass of the building, but by preventing overheating of the building in the warm period of the year.

Keywords: heat transfer processes, passive heating systems, computer system.

List of publications:

1. Andronova O.V., Kurak V.V., Sokol K.I. Thermal mode of the building with a passive heating system. O.V. Bulletin of the Kherson National Technical University. Engineering sciences. No. 1(72), Part 1. KNTU. 2020. p. 9-17.

URL: https://journals.kntu.net.ua/index.php/visnyk/article/view/467/493

DOI: https://doi.org/10.35546/kntu2078-4481.2020.1.1.1

2. Sokol K.I., Ohnineva O. Development of a computer modeling system to optimize the parameters of a closed passive heating system. Problems of information technologies. 2020. No. 27. P. 69-77.

URL: https://journals.kntu.net.ua/index.php/pit/article/view/705

DOI: https://doi.org/10.35546/2313-0687.2020.27.69-77

3. Sokol K., Ohnineva O., Andronova O.V. Designing a computer system for researching the thermal regime of a building with a closed passive heating system. Bulletin of the Kherson National Technical University. Information Technology. No. 1(80). KNTU 2022. p. 61-67.

URL: https://ojs.kntu.net.ua/index.php/visnyk/article/view/236

DOI: https://doi.org/10.35546/kntu2078-4481.2022.1

4. Sokol K. Software implementation of the method of calculating the building's energy needs. Bulletin of the Khmelnytskyi National University. Series: technical sciences. No. 335(3.1). KNU 2024. p. 209-215.

URL: https://heraldts.khmnu.edu.ua/index.php/heraldts/article/view/229

DOI: https://doi.org/10.31891/2307-5732-2024-335-3-29

Approbation of the results of the dissertation:

1. Sokol K., Ohnineva O. Use of SMART GRIDS in building heating systems / K. Sokol, O. Ohnineva // Materials of the All-Ukrainian Scientific and Technical Conference "Information Technologies and Programming", November 30, 2020. – Kherson: KhNTU, 2020. – pp. 86-87.

2. Sokol K.I., Ohnineva O. Designing a computer system for simulating the operation and researching the effectiveness of a passive heating system / K. Sokol, O. Ohnineva // Materials of the 2nd All-Ukrainian scientific and practical conference of

young scientists, graduate students and students "Youth and Science", 05/24/2021 - Kherson: KNTU, 2020 - pp. 12-13.

3. Sokol K. Ohnieva O. Development of the structural scheme for passive closed type passive heating system's parameters optimization / Intelligent decision-making systems and problems of computational intelligence: Materials of the International. of science conf., Zaliznyy Port, May 25-29, 2021 - Kherson: Vyshemyrskyi V. S. Publishing House, 2021. - pp. 111-113.

4. Sokol K., Ohnineva O. Designing an application for calculating indicators of a building model with a passive closed-type heating system / K. Sokol., O. Ogneva // Materials of the All-Ukrainian Scientific and Technical Conference "Information Technologies and Programming", 11/30/2021. – Kherson: KNTU, 2021.

5. Sokol K., Ohnineva O. The concept of an application for designing buildings with passive heating systems / K. Sokol., O. Ohnineva // Materials of the international scientific and practical conference "Synergy of science and business in the post-war reconstruction of Kherson Region", 26-28.04. 23. Oldi+ Publishing House, 2023. - p. 207-211.

6. Sokol K., Ohnieva O., Dostalek L. Determining the effectiveness of using passive closed-loop heating systems to prevent the risk of building overcooling during energy system stability issues.. International Workshop on Computational & Information Technologies for Risk-Informed Systems , December 21-22, 2023.

ЗМІСТ

ВСТУП

Системи, в яких процеси приймання, накопичення та використання сонячної енергії для підтримання теплового режиму будівлі, здійснюються природним шляхом, в архітектурно-будівельних елементах будівлі, називаються пасивними системами опалення. Для дослідження та візуалізації явищ і процесів, що відбуваються всередині фізичної системи, якою є будівля з пасивною системою опалення, використовуються різноманітні сучасні технології, зокрема, методи комп'ютерного моделювання.

Актуальність дослідження. В умовах постійного зростання цін на енергоресурси, прагнення до зменшення шкідливих викидів до атмосфери, а також, поступово наростаючого дефіциту викопних видів палива, що можуть бути використані для опалення, привертають до себе увагу методи підвищення енергоефективності та енергозбереження. Одним із таких методів і є системи пасивного опалення. Використання даних систем дозволяє значно зменшувати витрати енергоресурсів на опалення та охолодження будівлі.

Зв'язок роботи з науковими програмами, планами, темами. Дисертаційна робота є складовою частиною тематичної програми досліджень кафедри програмних засобів і технологій Херсонського національного технічного університету на тему «Розробка моделей та методів комп'ютерного моделювання процесів теплопередачі в пасивних системах опалення» (державний номер реєстрації №0123U102034). Автор приймав участь у цій роботі як виконавець. Роль автора полягала в розробці моделей і методів інформаційних технологій для дослідження нових технологій транспортування енергії, впровадження енергоефективних, ресурсозберігаючих технологій, освоєння альтернативних джерел енергії.

Мета та завдання дослідження. Метою дослідження є створення комп'ютерної системи, що дозволяє визначати енергетичні потреби будівлі з пасивною системою опалення.

Основними завданнями дослідження є:

- Дослідження теоретичних основ процесів теплопередачі.
- Створення структурних схем, що представлять будівлю з пасивною системою опалення як об’єкт моделювання.
- Дослідження різних методів аналізу процесів теплопередачі та визначення доцільності їх використання для виконання заданої мети.
- Створення методу для дослідження теплових характеристик будівлі з пасивною системою опалення.
- Реалізація комп’ютерної моделі на основі запропонованого методу.
- Перевірка точності та ефективності використаного методу.
- Перевірка ефективності використання пасивних систем опалення для житлових будівель.

Об’єктом дослідження є будівля з пасивною системою опалення.

Предметом дослідження є теплові характеристики будівлі з пасивною системою опалення.

Методи дослідження. Розглянуті методи дослідження процесів теплопередачі, як числові так і інженерні. Усі методи є потужними інструментами для моделювання та числового розв'язання різних фізичних систем, зокрема, для визначення температурних характеристик різних об’єктів. Перевага числових методів полягає в тому, що вони дають можливість глибоко досліджувати процеси всередині системи та отримувати точні значення для будь-якого відрізку часу. Але, головним чином, через свою ресурсоємність, ці методи не дуже підходять для вивчення великих об'єктів, таких як будівлі. Їх використання для цієї мети вимагає застосування обчислювальних серверів і займає багато часу. Перевагою інженерних методів є висока швидкість розрахунку та відсутність необхідності використання великих обчислювальних потужностей. Головним недоліком є те, що, найчастіше, для розрахунків використовуються середньомісячні значення основних показників, що не дає заглибитися у характеристики системи на певному, короткому відрізку часу і побачити переваги й недоліки системи при певних умовах, якими можуть стати зовнішня температура, інтенсивність сонячного

випромінювання тощо. Проаналізувавши переваги та недоліки всіх методів, вирішено частково інтегрувати числові методи у мережі теплових опорів у новому розробленому методі для розрахунку теплових характеристик будівлі.

Продемонстровано створення нового методу для розрахунку теплових та енергетичних характеристик будівлі з пасивною системою опалення. Даний метод є поєднанням числових та інженерних методів розрахунку, що дозволяє йому мати високу швидкість та точність виконання обчислень, а також, отримати можливість дослідження системи на певному, короткому відрізку часу і побачити переваги й недоліки системи при певних вхідних умовах.

Наукова новизна полягає в наступному:

- Створено метод для дослідження теплових характеристик будівлі з пасивною системою опалення. На відміну від інших, даний метод є дозволяє отримати високу точність обчислень за рахунок використання елементів числових методів, і в той же момент, зберігає простоту та зрозумілість інженерних методів.
- Реалізовано комп’ютерну модель на основі запропонованого методу дослідження теплових характеристик будівлі з пасивною системою опалення. На відміну від інших, дана модель дозволяє отримати результати високої точності за короткий час, при цьому, зберігаючи можливість більш детального дослідження в будь-який момент роботи системи.

Практичне значення одержаних результатів. Результати виконання дослідження можуть бути впроваджені для виконання розрахунків теплових характеристик або підбору параметрів пасивної системи опалення для оптимізації енергетичних потреб будівлі перед створенням проекту для будівництва.

Апробація результатів дисертації відбулися на конференціях:

1. Всеукраїнська науково-технічна конференція «Інформаційні технології та програмування», 30.11.2020р., Херсон.

2. Всеукраїнська науково-практична конференція молодих вчених, аспірантів і студентів «Молодь і наука», 24.05.2021р., Херсон.

3. Міжнародна наукова конференція «Інтелектуальні системи прийняття рішень і проблеми обчислювального інтелекту», с. Залізний Порт, 25-29 травня 2021 р.

4. ІІ Всеукраїнська науково-технічна конференція «Інформаційні технології та програмування», 30 листопада 2021 р., Херсонський національний технічний університет.

5. Міжнародна науково-практична конференція «Синергія науки і бізнесу у повоєнному відновленні Херсонщини», 26-28 квітня 2023 р.

6. International Workshop on Computational & Information Technologies for Risk-Informed Systems, December 21-22, 2023.

Особистий внесок здобувача полягає в тому, що результати дослідження отримані самостійно.

Публікації

Основні результати дослідження опубліковані в 4 наукових працях, усі у наукових фахових виданнях, рекомендованих МОН України для публікації наукових досліджень:

1. Вісник Херсонського національного технічного університету. Інженерні науки. № 1(72), Ч. 1. ХНТУ. 2020. с. 9-17.

2. Проблеми інформаційних технологій. 2020. № 27. С. 69-77.

3. Вісник Херсонського національного технічного університету. Інформаційні технології. № 1(80). ХНТУ. 2022. с. 61-67.

4. Вісник Хмельницького національного університету. Серія: технічні науки. № 335(3.1). ХНУ. 2024. с. 209-215.

Структура та обсяг дисертації

Дисертація складається зі вступу, чотирьох розділів, висновків для кожного розділу, висновку, бібліографічного списку зі 60 найменувань, 2 додатків. Обсяг дисертації складає 179 сторінок, з яких 130 сторінок основного тексту, в тому числі 32 рисунки і 11 таблиць.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ ПРОЦЕСІВ ТЕПЛОПЕРЕДАЧІ

Теплопередача – це процес переносу теплової енергії з одного місця в іне за допомогою одного з механізмів: теплопровідність, конвекція або теплове випромінювання [1-4].

Процеси теплопередачі класифікують як постійні (стаціонарні) або перехідні (нестаціонарні). Для стаціонарного процесу відсутні зміни з часом у будь-якій точці середовища, температура або тепловий потік залишаються незмінними під час постійної теплопередачі через середовище в будь-якому місці, хоча обидві величини можуть змінюватися від одного місця до іншого.

Під час перехідної теплопередачі, температура, зазвичай, змінюється з часом, а також, залежить від положення об’єкту. В окремому випадку, температура середовища рівномірно змінюється з часом, при цьому залишаючись у стаціонарному положенні, такі системи називають зосередженими.

Більшість випадків теплопередачі, які зустрічаються на практиці, є перехідними, але, зазвичай, досліджуються вони в деяких передбачуваних, сталих умовах, оскільки стаціонарні процеси легше аналізувати. Якщо метою аналізу теплових процесів всередині будинку є визначення необхідного розміру обігрівача, потрібно знати максимальну величину втрати тепла з будинку, яка визначається для найгірших умов протягом тривалого періоду часу, тобто під час стабільної роботи в найгірших умовах. Якщо обігрівач достатньо великий, щоб підтримувати тепло в будинку за найгірших умов, він достатньо великий для всіх умов [1-4].

1.1 Теплопровідність

Теплопровідність – це передача енергії від частинок речовини з більшим потенціалом до сусідніх, з меншим потенціалом, що відбувається у результаті взаємодії між ними. Теплопровідність може відбуватися в твердих тілах, рідинах або газах.

У газах і рідинах теплопровідність зумовлена зіткненнями та дифузією під час хаотичного руху молекул. У твердих тілах це зумовлено коливанням молекул у ґратці та переносу енергії вільними електронами.

Теплопровідність матеріалу є мірою здатності матеріалу проводити тепло. Високе значення теплопровідності вказує на те, що матеріал є хорошим теплопровідником, а низьке значення вказує на те, що матеріал є поганим теплопровідником або ізолятором. Теплопровідність матеріалу можна визначити як швидкість передачі тепла через одиницю товщини матеріалу на одиницю площі на одиницю різниці температур [1-4]:

$$Q_{cond} = \lambda S \frac{\Delta T}{\Delta x}, \text{Вт} \qquad (1.1)$$

де:

λ – коефіцієнт теплопровідності матеріалу, *Вт/(м·К)*;

Δx – товщина шару матеріалу, *м*;

S – площа теплопередачі, $м^2$;

ΔT – різниця температур поверхонь теплопередачі, *К*.

Рівняння (1.1) можна розглядати як основне рівняння теплопровідності за сталих умов.

У граничному випадку, $\Delta x \to 0$, наведене вище рівняння зводиться до диференційної форми [1-4]:

$$Q_{cond} = -\lambda S \frac{dT}{dx}, \text{Вт} \qquad (1.2)$$

де:

dT/dx – температурний градієнт.

Це називається законом теплопровідності Фур’є. Рівняння вказує на те, що швидкість розповсюдження тепла в певному напрямку пропорційна градієнту

температури в цьому ж напрямку. Тепло проводиться в напрямку зниження температури, і градієнт температури стає від'ємним, так як температура зменшується зі збільшенням відстані.

Виходячи з вище описаного, можна запропонувати блок теплопередачі теплопровідністю, що стане одним із базових для побудови майбутньої комп'ютерної системи. Він імітуватиме фізичний процес теплопровідності відповідно до рівняння (1.2). У якості вхідних даних потребуватиме площу поверхні, на якій здійснюватиметься теплопередача, відомості про матеріал, для використання коефіцієнта теплопровідності матеріалу та товщину його поверхні, значення температур зовнішньої та внутрішньої поверхонь. На виході ж такий блок видаватиме значення потужності теплопередачі теплопровідністю, що можна буде використати в подальших розрахунках для визначення необхідних характеристик системи.

1.2 Конвекція

Конвекція – це процес передачі енергії між твердою поверхнею та прилеглою рідиною або газом, які перебувають у русі, і включає комбінований ефект провідності та руху потоків. Чим швидше рух потоків – тим більше значення має конвекційний теплообмін. За відсутності будь-якого об'ємного руху потоків, передача тепла між твердою поверхнею та сусідніми потоками відбувається шляхом теплопровідності.

Конвекція називається вимушеною, якщо потоки протікають поверхнею під дією зовнішніх засобів, таких як вентилятор, насос або вітер. Навпаки, конвекція називається природною, якщо рух рідини виникає під дією сил плавучості, які викликані різницею густини внаслідок зміни температури в рідині.

Потужність конветивної теплопередачі пропорційна різниці температур і зручно виражається за допомогою закону охолодження Ньютона [1-4]:

$$Q_{conv} = \alpha(T_s - T_{out})S, \text{Вт} \quad (1.3)$$

де:

α – коефіцієнт теплопередачі конвекцією, *Вт/(м²·К)*;

T_{out} – температура зовнішнього середовища, *К*;

T_s – температура поверхні, *К*.

Коефіцієнт конвективної теплопередачі не є властивістю потоку. Це експериментально визначений параметр, значення якого залежить від усіх змінних, що впливають на конвекцію, таких як геометрія поверхні, характер руху потоку, об’ємна швидкість потоку.

Виходячи з вище описаного, можна запропонувати блок конвективної теплопередачі, що стане одним із базових для побудови майбутньої комп’ютерної системи. Він імітуватиме фізичний процес конвекції відповідно до рівняння (1.3). У якості вхідних даних потребуватиме відомості про матеріал, для використання коефіцієнта теплопередачі конвекцією, значення температур зовнішнього середовища та поверхні матеріалу. На виході ж такий блок видаватиме значення потужності конвективної теплопередачі, що можна буде використати в подальших розрахунках для визначення необхідних характеристик системи.

1.3 Теплове випромінювання.

Випромінювання – це процес передачі енергії у формі електромагнітних хвиль (або фотонів) у результаті змін електронної конфігурації атомів або молекул. На відміну від провідності та конвекції, передача тепла випромінюванням не вимагає наявності проміжного середовища.

Теплопередача випромінюванням відбувається найшвидше (зі швидкістю світла), і воно не зазнає ослаблення у вакуумі. Всі тіла при температурі вище абсолютного нуля випромінюють теплове випромінювання.

Випромінювання – це об’ємне явище, і всі тверді тіла, рідини та гази різною мірою випромінюють, поглинають або пропускають випромінювання. Однак

випромінювання зазвичай розглядається як поверхневе явище для твердих тіл, які є непрозорими для теплового випромінювання, таких як метали, деревина та каміння, оскільки випромінювання, що випромінюється внутрішніми областями такого матеріалу, ніколи не може досягти поверхні, а випромінювання, що падає на такі тіла зазвичай поглинається в межах кількох мікрон від поверхні.

Потужність випромінювання, що надходить від поверхні, визначається законом Стефана–Больцмана [1-4]:

$$Q_{rad} = \varepsilon \cdot \sigma_0 \cdot S \cdot (T_{em}^4 - T_{abs}^4), \text{Вт} \qquad (1.4)$$

де:

ε – приведена ступінь чорноти в системі двох тіл;

σ_0 – стала Стефана-Больцмана, $Вт/(м^2К^4)$;

T_{em} – температура випромінювача, $К$;

T_{abs} – температури приймача випромінювання, $К$;

S – площа поверхні теплообміну, $м^2$.

Стала Стефана–Больцмана має значення $5{,}67\times10^{-8}$ $Вт/(м^2К^4)$. Ідеалізована поверхня, яка випромінює таким чином, називається чорним тілом. Випромінювання, випромінюване всіма реальними поверхнями, менше, ніж випромінювання чорного тіла при тій же температурі. Коефіцієнт випромінювання, значення якого знаходиться в діапазоні від 0 до 1, є мірою того, наскільки поверхня наближається до чорного тіла.

Іншою важливою властивістю випромінювання поверхні є її поглинальна здатність a, яка є часткою енергії випромінювання, що падає на поверхню, яка поглинається нею. Як і коефіцієнт випромінювання, його значення знаходиться в діапазоні від 0 до 1. Чорне тіло поглинає все випромінювання, що падає на нього. Тобто чорне тіло є ідеальним поглиначем, оскільки воно є ідеальним випромінювачем [1-4].

Теплопередача випромінюванням до або від поверхні, оточеної газом, відбувається паралельно провідності (або конвекції, якщо є масовий рух газу) між

поверхнею та газом. Таким чином, загальна теплопередача визначається додаванням внесків обох механізмів теплопередачі.

Виходячи з вище описаного, можна запропонувати блок теплопередачі випромінюванням, що стане одним із базових для побудови майбутньої комп'ютерної системи. Він імітуватиме фізичний процес теплового випромінювання відповідно до рівняння (1.4). У якості вхідних даних потребуватиме площу поверхні, на якій здійснюватиметься теплопередача, приведену ступінь чорноти в системі двох тіл, температури випромінювача та приймача. На виході ж такий блок видаватиме значення потужності теплопередачі випромінюванням, що можна буде використати в подальших розрахунках для визначення необхідних характеристик системи.

1.4 Теплоємність

Теплоємність – це фізична властивість речовини, яка визначається як кількість тепла, яке необхідно надати об'єкту, щоб викликати зміну його температури на одиницю. Вона показує, скільки енергії зберігає матеріал на одиницю маси [1-4]:

$$Q_{cap} = cm(T_{i-1} - T_i), \text{Дж} \qquad (1.5)$$

де:

c – коефіцієнт теплоємності матеріалу, *Дж/(кгК)*;

m – маса, *кг*.

Виходячи з вище описаного, можна запропонувати блок теплоємності, що стане одним із базових для побудови майбутньої комп'ютерної системи та буде виконувати функцію накопичувача теплоти. Він імітуватиме фізичну властивість матеріалу до збереження певної кількості енергії відповідно до рівняння (1.5). У якості вхідних даних потребуватиме відомості про матеріал для використання коефіцієнту теплоємності та його масу. На виході ж такий блок видаватиме

значення теплоємності, що можна буде використати в подальших розрахунках для визначення необхідних характеристик системи.

1.5 Термічний опір

Термічний опір середовища залежить від його геометрії та теплових властивостей. Виходячи з рівняння (1.1) для теплопровідності [1]:

$$Q_{cond} = \frac{\Delta T}{R_{cond}}, \text{Вт} \tag{1.6}$$

$$R_{cond} = \frac{\Delta x}{\lambda S}, \text{К/Вт} \tag{1.7}$$

У рівняння теплопередачі конвекцією (1.3), термічний опір можна ввести наступним чином [1]:

$$Q_{conv} = \frac{T_s - T_{out}}{R_{conv}}, \text{Вт} \tag{1.8}$$

$$R_{conv} = \frac{1}{\alpha S}, \text{К/Вт} \tag{1.9}$$

Коли коефіцієнт теплопередачі конвекцією прямує до нескінченності, опір конвекції стає нульовим, тобто поверхня не чинить опору конвекції, а отже, не уповільнює процес теплопередачі. На практиці, це призводить до кипіння та конденсації.

У випадку, коли тверда поверхня (наприклад – стіна), оточена повітрям, теплова радіація може мати значний вплив на загальну теплопередачу. Термічний опір може бути наступним чином введений в закон Стефана-Больцмана (1.4) [1]:

$$Q_{rad} = \frac{T_{em}^{\square} - T_{abs}^{\square}}{R_{rad}}, \text{Вт} \tag{1.10}$$

$$R_{rad} = \frac{1}{h_{rad} \cdot S}, \text{К/Вт} \quad (1.11)$$

$$h_{rad} = \frac{Q_{rad}}{S\left(T_{em}^{\square} - T_{abs}^{\square}\right)} = \varepsilon \cdot \sigma (T_{em}^2 - T_{abs}^2)\left(T_{em}^{\square} - T_{abs}^{\square}\right), \text{Вт/(м}^2\text{К)} \quad (1.12)$$

Термічний опір, сам по собі, не є окремим механізмом теплопередачі, проте використання цієї концепції розповсюджене дуже широко, оскільки вона інтуїтивно проста для розуміння та зрозуміла для використання у розрахунках за нечисельними методами.

Як видно з описаного вище, використання концепції термічного опору може бути корисним для побудови комплексної системи, де теплопередача відбувається одночасно декількома способами, адже приводить усі фізичні властивості системи до одного формату, що дозволяє легко комбінувати різні механізми теплопередачі. Також, така концепція буде зручною при роботі з багатошаровими структурами, такими як стіни будівлі, де кожен окремий шар матеріалу описуватиметься новим значенням термічного.

1.6 Комбіновані режими теплопередачі

Існує три основні механізми теплопередачі, але це не значить, що всі вони будуть завжди проходити одночасно. Для твердих тіл, що є суцільними і непрозорими, теплопередача відбуватиметься лише шляхом теплопровідності, а у напівпрозорих твердих тілах – за допомогою теплопровідності та випромінювання. А от конвекція всередині твердого тіла, як і у вакуумі, є неможливою, через відсутність необхідного середовища. Проте, вже між різними твердими тілами, що знаходяться на відкритому середовищі, теплопередача конвекцією відбувається. Передача тепла у рідинах відбувається за рахунок теплопровідності або конвекції, залежно від наявності будь-якого об’ємного руху рідини. Гази практично прозорі для випромінювання, тому, у більшості випадків, газ між двома твердими поверхнями не заважає випромінюванню.

У численних застосуваннях у техніці всі три механізми теплопередачі можуть співіснувати, і взаємодія між ними має значний вплив як на загальну швидкість теплопередачі, так і на розподіл температури в складних системах [5].

Процеси теплопередачі теплопровідністю і конвекцією дуже нечасто відбуваються окремо. Це співіснування робить комбіновану теплопередачу за допомогою цих механізмів звичайним явищем у будь-якій системі. У комбінованому теплообміні провідністю та конвекцією процес провідності передає утворене тепло на поверхню компонента, а конвекція забезпечує передачу тепла в оточення. Особливо часто конвекція та теплопровідність співіснують на межі розділу рідина-тверде тіло. Для теплопровідності – це передача тепла від гарячої пластини до рідини через твердий матеріал, тоді як для конвекції – це додаткова передача тепла від поверхні внаслідок руху рідини [5-7].

Комбінована теплопровідність і конвекція теплопередачі може відбуватися або послідовно, або паралельно. Незалежно від послідовного або паралельного виникнення, комбінована провідність і конвекція теплопередачі потребують середовища для теплопередачі. У вакуумі неможлива комбінована теплопровідність і конвекція, лише випромінювання забезпечує розсіювання тепла без середовища [5-7].

У деяких ситуаціях передача тепла відбувається як шляхом внутрішньої провідності всередині матеріалу, так і зовнішнього випромінювання між його поверхнями. Провідність через частинки, які є твердими, і випромінювання між поверхнями пор часто відбуваються одночасно в пористих матеріалах, таких як ізоляційні матеріали та щільні шари. Оптимізація передачі тепла через пористі середовища має вирішальне значення в таких застосуваннях, як каталітичні реактори та ізоляція будівель, де знання цієї взаємодії має вирішальне значення. Випромінювання є ключовим компонентом передачі енергії в кількох інженерних системах, включаючи камери згоряння та печі. У середовищі-учасниках, де випромінювання поглинається та розсіюється в середовищі та впливає на загальну теплопередачу, можуть бути присутні частинки або гази [5].

Пов’язаний або комбінований теплообмін випромінюванням і теплопровідністю відбувається в нагрітих напівпрозорих середовищах, які мають спектральний діапазон часткової прозорості (діапазон частот, де значення коефіцієнта поглинання знаходиться приблизно в інтервалі від 0,01 до 100 $см^{-1}$) і де випромінювання значну частину загального потоку енергії [5].

Наявність області напівпрозорості в спектрі теплового випромінювання характерна для діелектриків і напівпровідників у конденсованих фазах і для багатоатомних газів з несиметричними молекулами.

У заповнених рідиною огородженнях, таких як опалювальні приміщення, конвекція та випромінювання можуть взаємодіяти. Провідність, конвекція та випромінювання сприяють передачі тепла в замкнутих приміщеннях, таких як електричне обладнання. Конвекція та випромінювання беруть участь у передачі тепла від поверхонь до рідини (повітря), тоді як провідність відбувається через тверді стінки. Інтегровані в ці корпуси методи теплопередачі впливають як на загальну температурну дисперсію, так і на теплові характеристики системи. Взаємодія випромінювання та конвективного теплообміну в турбулентних потоках, у тому числі в реактивних двигунах, а також промислових пальниках. Покращуючи змішування рідини, турбулентність покращує передачу тепла та може впливати на радіаційний теплообмін між гарячими газами та холоднішими середовищами [5, 8-10].

Поєднана або комбінована радіаційна і конвективна теплопередача – це окремий випадок одночасної радіаційної, конвективної і теплопровідної теплопередачі, що відбувається, коли теплопровідність є незначно малою порівняно з теплопередачою випромінюванням і конвекцією. Між граничними поверхнями та навколишнім середовищем, між поверхнями, розділеними рухомим середовищем, і всередині рухомого середовища виникає комбіноване випромінювання та конвекція. Залежно від умов у середовищі, геометричних факторів і стану поверхні можливі режими сильної і слабкої взаємодії між радіаційним і конвективним теплообміном. Для слабкої взаємодії задачі теплообміну випромінюванням і конвекцією можна вирішувати послідовно і

незалежно, тоді як для сильної взаємодії ці процеси істотно взаємозалежні. Таким чином, передача енергії одним механізмом може впливати на теплообмін іншим механізмом і навпаки [5, 8-10].

У деяких системах поверхневе випромінювання та природна конвекція, викликана виштовхуючими силами, викликаними різницею температур, можуть працювати разом. Наприклад, як у випадку з сонячними колекторами, тепло від сонця змушує повітря підніматися вгору внаслідок його зменшення щільності. При цьому, радіаційний теплообмін між нагрітими поверхнями і навколишнім середовищем впливає на процес конвективного теплообміну [5, 8-10].

Поєднання всіх трьох основних механізмів теплопередачі – це самоузгоджений процес теплообміну за механізмами теплового випромінювання, конвекції та теплопровідності. Він виникає між поверхнею, що контактує з рухомим середовищем, і між різними компонентами потоків, причому, рухоме середовище розглядається не тільки як газ або плазма, але й у конденсованому стані.

Тут самоузгоджений теплообмін означає, по суті, що кожен із зазначених вище механізмів однаково впливає на енергетичний баланс всередині та на межі розглянутої області і тим самим змінює інтенсивність енергообміну іншими механізмами.

Такий тип теплопередачі є найбільш загальним. При такому загальному випадку теплопередачі в рухомому середовищі діють усі три механізми. Однак (залежно від температури середовища, швидкості, щільності, геометрії та оптичних і фізичних властивостей) можливо домінування одного або двох механізмів.

При розв’язанні задач такого типу теплопровідності, у яких би механізмах теплопередачі не було важливого значення, корисним першим кроком є визначення впливу випромінювання на параметри рухомого середовища. Якщо цей вплив невеликий, то можна спростити задачу, розглядаючи окремо рух середовища і радіаційний теплообмін. Далі задачі вирішуються послідовно методами для рухомого середовища і радіаційного теплообміну [8-10].

1.7 Висновки до першого розділу

У першому розділі досліджено теоретичні основи складових процесу теплопередачі: теплопровідності, конвекції, випромінювання, а також, теплоємності. Виходячи з цих даних, запропоновані блоки для побудови майбутньої системи, що імітують відповідні фізичні властивості системи, а також, фізичні властивості матеріалів, результати розрахунків усередині блоків можна використовувати для визначення необхідних характеристик системи.

Розглянута концепція термічного опору, що дозволяє більш ефективно працювати з комплексними системами, в яких теплопередача відбувається декількома способами одночасно, а також, з багатошаровими структурами, такими як стіни будівлі.

РОЗДІЛ 2
ПАСИВНА СИСТЕМА ОПАЛЕННЯ ЯК ОБ'ЄКТ МОДЕЛЮВАННЯ

2.1 Комп'ютерні моделі та їх класифікація

Комп'ютерна модель – це комп'ютерна програма, ціллю виконання якої є створення абстрактної модель певної системи. Комп'ютерне моделювання стало корисною частиною математичного моделювання багатьох природних систем у фізиці, хімії та біології, економіці, психології та соціальних науках, а також у процесі розробки нових технологій, щоб отримати уявлення про функціонування цих систем. Традиційно, формальне моделювання систем здійснюється за допомогою математичної моделі, яка намагається знайти аналітичні рішення проблем, що дозволяє передбачити поведінку системи на основі набору параметрів і початкових умов. Комп'ютерне моделювання базується на суто математичних моделях [11].

Комп'ютерні моделі можна класифікувати за кількома критеріями, зокрема:

Стохастичні або детерміновані. Стохастична модель – це модель, яка включає деякі елементи випадковості або невизначеності в систему або процес. Іншими словами, стохастична модель не припускає, що результат системи або процесу повністю визначається його початковими умовами та параметрами, а радше те, що він може змінюватися відповідно до деякого розподілу ймовірностей або функції. Наприклад, стохастичну модель підкидання монети можна описати простим біноміальним розподілом, який залежить від ймовірності отримання орла або решки. Стохастична модель може бути корисною, коли система або процес є складними, динамічними та непередбачуваними, а також коли мінливість і розподіл моделі важливі.

Детермінована модель – це модель, яка передбачає, що результат системи або процесу повністю визначається його початковими умовами та параметрами. Іншими словами, детермінована модель не передбачає будь-якої випадковості чи невизначеності, і вона завжди дає однаковий результат для тих самих вхідних

даних. Детермінована модель може бути корисною, коли система або процес добре зрозумілі, передбачувані та стабільні, а також коли важливі точність і точність моделі [10, 12].

Для створення моделі в даній роботі, матиме більшу актуальність саме детермінована модель, адже для розрахунків використовуватимуться середні багаторічні дані про сонячну активність та температуру на певній місцевості, а також, параметри матеріалів, що є, загалом, незмінними. Це означає, що результати моделювання будуть визначеними заздалегідь.

Стаціонарні або динамічні. Динамічні моделі – це спрощене представлення деяких об’єктів реального світу в рівняннях або комп’ютерному коді. Вони мають на меті імітувати важливі особливості системи навчання. Для процесів теплопередачі, моделі називають динамічними, тому що вони враховують зміни умов навколишнього середовища та відповідно підлаштовують поведінку досліджуваної системи в реальному часі. Основними аспектами динамічного моделювання теплопередачі є сезонні коливання, що враховують різницю в енергетичній потребі в періоди опалення та кондиціонування будівель та добові коливання, що враховують зміни температури та інтенсивності сонячного випромінювання впродовж доби.

Таке моделювання важливе для отримання можливості переглянути стан системи опалення або процесів будівлі безпосередньо в конкретний момент часу, або ж, побачити зміни під час критичних зовнішніх умов (аномально низької або високої температури), що дає більше розуміння переваг та недоліків системи, а також, можливість для її адаптації під певні умови та оптимізації [11, 13].

Для створення моделі в даній роботі, важливо буде враховувати сезонні та добові коливання температури, тому створення динамічної моделі є більш актуальним.

Безперервні або дискретні. Безперервна та дискретна моделі – це два різних типи математичних представлень, які використовуються в системах керування для опису поведінки системи. Кожен тип моделі має свої переваги та обмеження, а

вибір між безперервною та дискретною моделями залежить від характеристик системи, якою керують, і застосованої стратегії керування.

Безперервні моделі описують системи, які працюють безперервно в часі, наприклад фізичні процеси, які керуються диференціальними рівняннями. У безперервних моделях входи та виходи системи представлені як безперервні сигнали, які постійно змінюються з часом. Ці моделі зазвичай представляють за допомогою диференціальних рівнянь, які описують поведінку системи як функцію часу.

Однією з головних переваг безперервних моделей є їх здатність з високою точністю фіксувати динаміку безперервних фізичних процесів. Безперервні моделі можна використовувати для аналізу та розробки стратегій керування для широкого діапазону фізичних процесів, включаючи механічні, електричні та хімічні системи. Крім того, безперервні моделі можуть надати уявлення про стабільність і продуктивність системи, дозволяючи інженерам з управління оптимізувати стратегії керування для відповідності бажаним характеристикам продуктивності.

Однак безперервні моделі можуть потребувати інтенсивних обчислень і вимагати передових математичних інструментів для аналізу та проектування. Крім того, дискретизація безперервних моделей може призвести до помилок у продуктивності керування, особливо якщо дискретизація виконується не ретельно.

Дискретні моделі, з іншого боку, описують системи, які працюють з дискретними кроками в часі, такі як цифрові системи керування. У дискретних моделях входи та виходи системи представлені як дискретні сигнали, які змінюються лише через певні проміжки часу. Ці моделі зазвичай представляють за допомогою різницевих рівнянь, які описують поведінку системи як функцію дискретних часових кроків.

Однією з головних переваг дискретних моделей є їх простота і легкість реалізації. Дискретні моделі можна легко реалізувати в цифрових системах керування, що робить їх ідеальними для додатків керування, які потребують високошвидкісної обробки та керування в реальному часі. Крім того, дискретні

моделі є більш ефективними для обчислення та вимагають для цього менше ресурсів, порівняно з неперервними моделями.

Однак, дискретні моделі можуть не точно фіксувати динаміку безперервних фізичних процесів, особливо якщо частота дискретизації недостатньо висока. Це може призвести до помилок у продуктивності керування та нестабільності системи керування [11, 14].

Для створення моделі в даній роботі, враховуючи, що дані про середню багаторічну температуру приводяться для типового дня місяця погодинно, виходячи з цього, логічно буде прийняти певний крок розрахунку, що буде дорівнювати одній годині та розраховувати всі характеристики саме з таким кроком, що зробить дану модель дискретною.

2.2 Визначення енергетичних потреб будівлі за допомогою комп'ютерного моделювання

Визначення енергетичних потреб будівлі є досить важливою та перспективною задачею, зокрема, через те, що це допомагає вивчати показники енергоефективності будівлі, і, виходячи з цих знань, підбирати і впроваджувати заходи для її підвищення. Енергоефективність будівлі – це ступінь, до якого споживання енергії на квадратний метр площі будівлі відповідає встановленим стандартам споживання енергії для цього конкретного типу будівлі за певних кліматичних умов.

Еталонні показники енергоспоживання будівлі є репрезентативними значеннями для поширених типів будівель, з якими можна порівняти фактичні показники будівлі. Порівняльні показники виводяться шляхом аналізу даних про різні типи будівель у певній країні. Типовим еталонним показником є середній рівень ефективності всіх будівель у даній категорії, а хороша практика представляє максимальну продуктивність квартиля. Порівняння з простими еталонними показниками річного споживання енергії на квадратний метр площі підлоги або обробленої площі підлоги (кВт×год/м^2) дозволяє оцінити стандарт

енергоефективності та визначити пріоритетні напрямки для дій. Більшість країн мають чітко визначені межі енергоефективності для нових забудівель, наприклад, в Україні, нова будівля має відповідати класу енергоефективності «С», що відповідає енерговитратам, не більшим, ніж 87 кВт×год/м^2 [15].

Інвестиції в енергоефективність будівлі можна порівняти з вартістю капітальних інвестицій, необхідних на стороні постачання енергосистеми для виробництва аналогічної пікової потужності або річного виробництва енергії. Зазвичай капітальні витрати на ефективність є нижчими, ніж порівняльні інвестиції у збільшення пропозиції, і немає додаткових операційних витрат на підвищення ефективності порівняно зі значними експлуатаційними витратами для варіантів на стороні постачання. Крім того, інвестиції в енергоефективність, як правило, мають набагато коротший час виконання, ніж інвестиції в енергопостачання, що особливо важливо враховувати в країнах, де попит на енергетичні послуги швидко зростає. Встановлюючи цілі енергоефективності для будівель, уряди розподіляють тягар і витрати на забезпечення безпеки енергопостачання з кінцевими споживачами. На експлуатації будівель високого класу енергоефективності витрачається менше коштів, це робить їх більш економічно вигідними в довгостроковій перспективі. Тобто, початкові витрати на створення такої будівлі будуть тим більшими – чим більша бажана енергоефективність, але така інвестиція окупиться з часом через менше використання енергоресурсів.

Основною перевагою заходів з підвищення енергоефективності будівель є зниження енергетичних витрат будівлі, але ,зазвичай, є й інші переваги, такі як покращений внутрішній мікроклімат будівлі, що позитивно впливає на здоров’я жителів осел, підвищена вартість нерухомості, зменшення використання викопних видів палива за рахунок підвищення енергоефективності приводить також до зменшення викидів парникових газів, що позитивно впливає на навколишнє середовище. Також, визначення енергетичних потреб допомагає при проектуванні нових будівель, зокрема, у виборах матеріалу для створення основних конструкцій або теплоізоляційного шару [15, 16].

Одним з найефективніших методів для дослідження теплоенергетичних процесів всередині будівлі та визначення енергетичних потреб є використання комп’ютерних систем. Для цього існує низка причин, значною перевагою комп’ютерних систем є те, що завдяки ним з’являється можливість створювати достатньо точні та зрозумілі моделі будівель, що можуть спиратися на велику кількість параметрів. Можна задавати різні вхідні дані, такі як матеріали, що будуть використані для створення огороджувальних конструкцій будівлі, теплоізоляційні матеріали, обирати геометрію будівлі, кліматичні умови, тип системи опалення тощо.

Комп'ютерні системи дозволяють обробляти інформацію з достатньо високою швидкістю, що дозволяє ускладнювати розрахунки, надаючи їм більшої точності, додавати вхідні дані. Також, це надає можливість оцінити більшу кількість варіантів, порівняти переваги та недоліки кожного. Доступність та простота внесення змін у комп’ютерні системи дає можливість постійно покращувати процеси розрахунку та моделювання всередині системи.

Комп’ютерні системи дають можливість легко візуалізувати отримані результати, створити графіки або діаграми, що якнайкраще впливає на сприйняття матеріалу. Також, такі системи мають можливість надавати підтримку користувачеві при прийнятті тих чи інших рішень у процесі моделювання, порівнюючи та аналізуючи різні варіанти та демонструючи саме ті характеристики, що необхідні для прийняття рішення [15, 16].

Створення комп’ютерних моделей для визначення енергетичних потреб будівлі є комплексним процесом, що складається з фізичних, математичних та технологічних аспектів, що дозволяє спеціалістам різних причетних галузей отримати кращі первинні дані та прогнози ефективності, а відповідно і ввести корективи ще на етапі проектування.

Розрахунок енергетичного балансу будівлі є одним з основних принципів для створення моделі з розрахунку енергетичних потреб. Енергетичний баланс — це найбільш повний статистичний облік енергоносіїв та їх руху в господарстві. Енергетичний баланс дозволяє користувачам бачити загальну кількість енергії,

видобутої з навколишнього середовища, проданої, перетвореної та використаної кінцевими споживачами. Це також дозволяє побачити відносний внесок кожного енергоносія (палива, продукту). Енергетичний баланс дозволяє вивчати загальний внутрішній енергетичний ринок і контролювати вплив енергетичної політики. Енергетичний баланс пропонує повне уявлення про енергетичну ситуацію в країні в компактному форматі, наприклад, про енергоспоживання всієї економіки та окремих секторів.

Концепція енергетичного балансу є системою обліку для збирання, узгодження та розуміння даних про всі енергетичні продукти, що надходять, вивозяться та використовуються в країні (території).

При побудові енергетичних балансів розраховують усю теплову енергію, що передається всередину будівлі, завдяки впливу сонячної радіації, зовнішньої температури, температури ґрунту, а також, внесок системи опалення. Усі витрати теплової енергії з поверхонь огороджувальних конструкцій будівлі. Усі додаткові внутрішні надходження теплової енергії, від людей, освітлення та обладнання, що виділяє тепло при своїй роботі. Закон збереження енергії стверджує, що повна енергія ізольованої системи постійна; енергія може бути перетворена з однієї форми в іншу, але не може бути ні створена, ні знищена. Перший закон часто формулюють, стверджуючи, що зміна внутрішньої енергії замкнутої системи дорівнює кількості тепла, що подається системі, мінус кількість роботи, виконаної системою над навколишнім середовищем. Отже, приріст енергії неможливий, і якщо він є, він є результатом або статистичних розбіжностей (дані низької точності), або неповного врахування всіх вхідних продуктів у сфері енергетичної статистики [17-22].

На енергетичний баланс, у свою чергу, найбільший вплив мають наступні показники:

1. Вплив зовнішніх умов, таких як температура навколишнього середовища та сонячне випромінювання, що головним чином впливають на зовнішні шари огороджувальних конструкцій.

2. Фізичні властивості конструкційних та теплоізоляційних матеріалів, що визначать теплові процеси всередині будівлі, такі як теплопровідність, теплоємність та емісивність, тобто, властивість поверхні конструкції, або матеріалу, випромінювати тепло.

3. Геометрія будівлі. Форма і розміри будівлі можуть значно впливати на енергетичні потреби, чим компактніша будівля – тим менше її енергетичні потреби.

Орієнтація щодо сторін світу також має значний вплив, адже найбільше сонячного випромінювання приходить на південний напрямок, відповідно – чим більша площа південних фасадів – тим більші такі надходження. Проте, найчастіше, збільшення південних фасадів відповідно впливає і на північні, що мають найменші теплові надходження від сонячного випромінювання.

Кількість скляних фасадів значно впливає на енергетичні потреби через те, що вікна, зазвичай є слабким місцем будівлі у відношенні теплових втрат через них, адже, зазвичай, стіни мають кращі теплоізоляційні властивості, проте, вікна також є і отвором для проникнення природного освітлення та сонячної радіації, що уже позитивно впливає на енергетичний баланс, крі того, вікна важливі ще й для охолодження будівель.

4. Теплова інерція, що означає, що чим більша теплоємність конструкцій та внутрішньої маси будівлі – тим більш стабільною буде температура всередині будівлі, відповідно, чим менше коливань протягом доби або певного довшого періоду часу – тим комфортніше себе почуватимуть жителі будівлі.

5. Комфорт мешканців будівлі. Деякі методи, що використовують сонячну радіацію як джерело опалення для будівлі, впливають на комфорт жителів. Наприклад, система з прямим уловлюванням, являє собою світлопрозорі конструкції великої площі, орієнтовані на південь, тобто, велика частина фасаду змінена на вікно, що може вести до незручностей, таких як засліплення сонцем, перегріви поверхонь тощо. Протилежна ситуація з пасивною системою опалення типу стіна Тромбе, там південний фасад буквально замінюється пасивною акумулюючою стіною, відповідно, може бути нестача природного освітлення, спричинена цим [17-22].

2.3 Пасивна система опалення

Розрізняють два основні види систем опалення, що використовують енергію сонця для обігріву будівель: активні і пасивні. Характерною ознакою активних систем є наявність колектора сонячної енергії, акумулятора теплоти, додаткового джерела енергії, трубопроводів, теплообмінників, насосів або вентиляторів і пристроїв для автоматичного контролю і керування. У пасивних системах процеси приймання, накопичення та використання сонячної енергії для опалення, здійснюються природним шляхом. У таких системах, роль сонячного колектора й акумулятора теплоти, зазвичай, виконують самі огороджувальні конструкції будинку, а рух теплоносія (повітря) здійснюється за рахунок природної конвекції [23-27].

Існує кілька основних концепцій пасивного та гібридного сонячного опалення, у яких є достатньо багато відмінностей і закладають основні принципи і функції пасивних систем. Це система з прямим надходженням, система з акумулюючою стіною та системи сонячних просторів.

Системи з прямим надходженням енергії являє собою світлопрозорі конструкції великої площі, орієнтовані на південь, і може покривати частину теплових навантажень будівлі. Вікно виступає у якості колектора, а непрозорі огороджувальні конструкції є уловлювачами та акумуляторами теплоти. Для запобігання перегріву приміщень у неопалювальний період використовують козирки або рухому ізоляцію для затінення світлопрозорих конструкцій. У холодному кліматі також необхідно ізолювати вікна в періоди низького надходження сонячної радіації, щоб запобігти надмірним втратам. Система з прямим надходженням енергії може забезпечити енергією південну сторону будівлі; і виникає необхідність передачі теплової енергії до приміщень, що не мають південних вікон [23-27].

Переваги: надзвичайно простий підхід: зорієнтуватися на південь, додати вікна, утеплити та налаштувати; досягнення комфорту та зниження витрат на опалення; конструкції мають низький вплив на навколишнє середовище, особливо

якщо використовуються найбільш екологічні матеріали; конструкції з прямим надходженням отримують багато денного освітлення, зменшуючи рахунки за електроенергію та створюючи приємні умови проживання та життя; додаткове сонячне скління часто дає винятковий вид на природну красу; при відповідній тепловій масі сонячні будинки комфортні та теплі, підтримуючи стабільні температури протягом року з невеликим додатковим обігрівом.

Недоліки: основний підхід приховує безліч складностей і багато ретельного налаштування; скління призводить до перегріву будинку; занадто мало теплової маси також призводить до перегріву будинку; занадто багато сонячного скління може зробити будинок дуже холодним вночі та в похмурі дні; все додаткове денне освітлення може спричинити серйозні проблеми з відблисками, якщо не дотримуватися створення сонячних зон; велика кількість сонячного скління може спричинити проблеми з конфіденційністю, якщо вікна не будуть ретельно затінені; сонячне скління може перетворитись у тепловідвід, якщо він належним чином не утеплений, або якщо утеплені жалюзі неприйнятні для того, щоб працювати щовечора.

Система з акумулюючою стіною об'єднує функції колектору та акумулятора, і також є частиною огороджувальних конструкцій будівлі. Схема такої системи, яку також називають стіною Тромбе, показана на рис. 2.1, також, саме ця система буде основним об’єктом даного дослідження. Частина південної стіни має бути одинарно або подвійно заскленою; за склом – це масивна стіна з кладочного матеріалу або резервуарів води, пофарбована чорним для поглинання сонячної радіації. Тепло переноситься в приміщення шляхом випромінювання та конвекції від акумулюючої стіни з боку кімнати, та шляхом примусової або природної конвекції кімнатного повітря через простір між склінням та стіною. Кімнатне повітря може проникати в цей простір через отвори (вентиляційні) в нижній частині стіни і повернутися в кімнату через отвори у верхній частині. Акумулююча стіна також може бути частиною даху та стелі. Необхідно використовувати рухливу ізоляцію в усіх видах клімату, крім м'якого, для контролю втрат у періоди низької інтенсивності сонячної радіації [23-27].

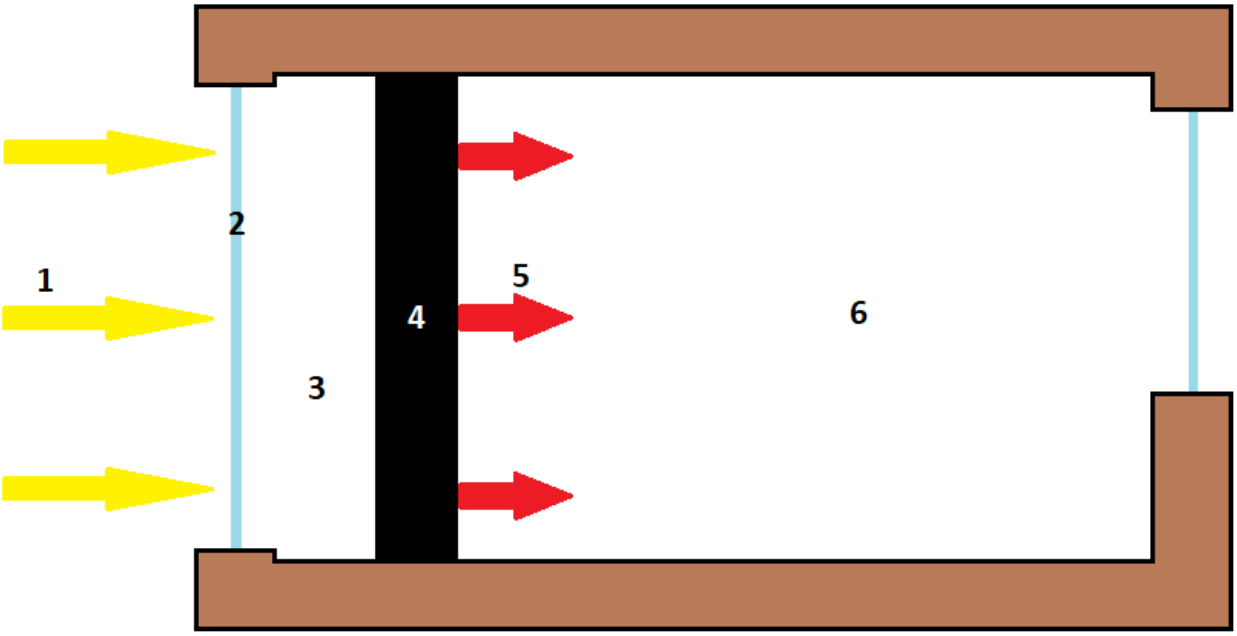

Рисунок 2.1 – Схема пасивної системи опалення закритого типу – стіни Тромбе: 1 – сонячне випромінювання; 2 – зовнішнє скління; 3 – повітряний прошарок; 4 – теплоакумулююча стіна; 5 – передача тепла від стіни до простору кімнати; 6 – простір кімнати

Переваги: стіна Тромбе – найбільш гнучка сонячна опція: вона працює як в помірному кліматі, так і в холодному; можна використовувати навіть для охолодження будівлі; зменшуються відблиски на меблі в порівнянні з конструкціями прямого посилення; ідеально підходять для домашніх офісів, телевізійних кімнат і темних спалень; найкраще працюють для нічного обігріву, хоча вони гнучкі; забезпечують теплову масу в концентрованій зоні, не займаючи житлової площі; створюють для комфортних, тихих приміщень зі стабільною температурою.

Недоліки: стіни Тромбе можуть призвести до витрат на будівництво, частково через скління, але в основному через те, що фундаменти повинні підтримувати значну додаткову вагу; можуть стати серйозними тепловідводами, втрачаючи велику кількість тепла вночі, якщо їх не утеплити; покривати їх щовечора може бути справжнім клопотом; вони зменшують денне освітлення та доступ до панорамних поглядів.

Системи з сонячними просторами використовуються і сонячні колектори і акумулювання у стінах, підлогах або галькових акумуляторах. Примусова циркуляція повітря в приміщеннях є можливістю поліпшення зберігання та використання поглиненої енергії. У холодному кліматі втрати енергії від сонячних просторів, які є парникоподібними спорудами, можуть перевищувати абсорбовану енергію, і потрібно дотримуватися обережності, щоб забезпечити отримання чистої вигоди від такої системи. Вид використання сонячного простору, для людей або для рослин, буде встановлювати обмеження на допустимі температурні коливання у сонячному просторі і впливати на енергетичний баланс у ньому [23-27].

Переваги: сонячні простори забезпечують тепло для себе та сусідніх приміщень; можуть зменшити втрати тепла від прибудованої житлової зони; забезпечують тепле повітря для циркуляції; конструкції з твердими загальними стінками зменшують відблиски та проникнення сонячного світла; забезпечують додаткову життєву площу.

Недоліки: сонячні простори, як правило, перегріваються і часто залиті сонцем; вночі вони часто занадто холодні, щоб бути житловими; сонячні простори часто не дають додаткового тепла додому поза власними стінами; тепле повітря в сонячному просторі не легко переміщається, тому, можливо, знадобляться вентилятори, щоб розповсюдити його по дому; сонячні простори часто не придатні для вирощування рослин – вони перегріваються, заливаються сонцем, а вночі холодні; якщо конструкція не буде ретельно продумана і монтаж не виконаний досвідченими професіоналами, сонячні простори можуть просто забезпечити трохи додаткового тепла і не мати зайвої житлової площі – але багато додаткових витрат на будівництво.

2.4 Структурна схема будівлі з пасивною системою опалення

Для побудови комп’ютерної системи, необхідно, для початку, розглянути структуру об’єкту дослідження, у даному випадку – будівлі з пасивною системою опалення закритого типу. Об’єкт (рис. 2.2) описується п’ятьма окремими

структурними схемами, що відповідають огороджувальним конструкціям будівлі: стіни, дах, підлога, вікна і, власне, пасивна система опалення закритого типу (стіна Тромбе). У схемах враховано теплообмін випромінюванням між огороджувальними конструкціями та оточуючим середовищем (нагрівання за рахунок надходження сонячної радіації та радіаційні тепловтрати в оточуюче середовище) та сумарну теплопередачу теплопровідністю.

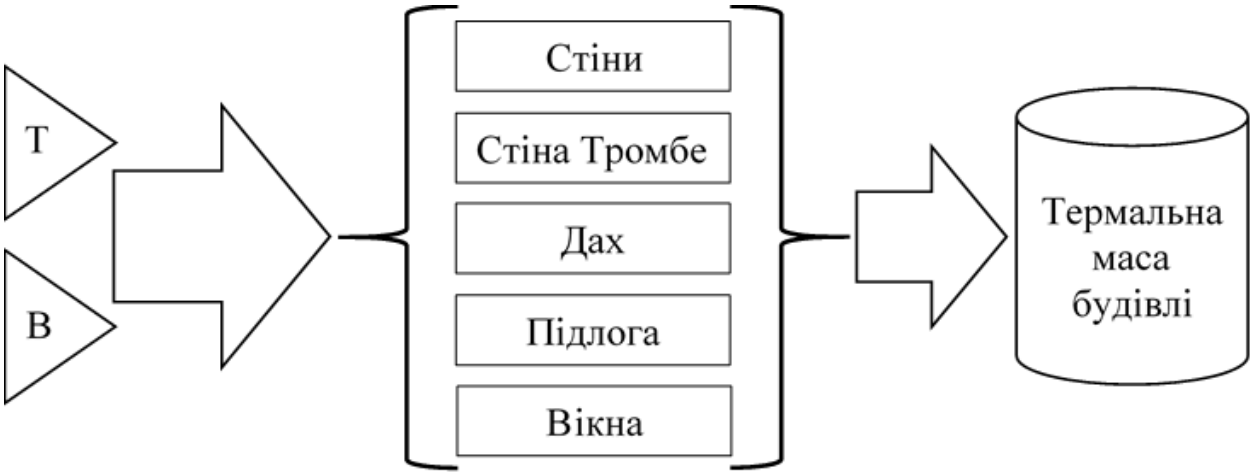

Рисунок 2.2 – Структурна схема моделі будівлі з пасивною системою опалення закритого типу

Для створення схеми використовуються блоки, що описують явища теплопередачі.

- Блок (T) описує передачу тепла в тепловій мережі шляхом теплопровідності і описується рівнянням (1.1).
- Блок (K) описує передачу тепла в тепловій мережі за допомогою конвекції за рахунок руху потоків і описується рівнянням (1.3).
- Блок (B) описує передачу тепла в тепловій мережі шляхом теплового випромінювання між двома поверхнями і описується рівнянням (1.4).
- Блок термальної маси описує внутрішнє зберігання енергії в тепловій мережі і описується рівнянням (1.5).

Схема стіни (рис. 2.3) складається з блоків теплопередачі конвекцією та випромінюванням, до яких підходить температурний тепловий потік та потік

сонячного випромінювання відповідно. У даних блоках енергія перетворюється та надходить до блоків теплопередачі теплопровідністю. Стіна представлена двома такими блоками, що ділять її навпіл, а між ними знаходиться термальна маса матеріалу стіни. У разі необхідності та для більшої точності дослідження, кількість таких блоків може зростати. Із внутрішньої частини стіни теплові потоки передаються до термальної маси будівлі за допомогою блоків теплопередачі конвекцією та випромінюванням.

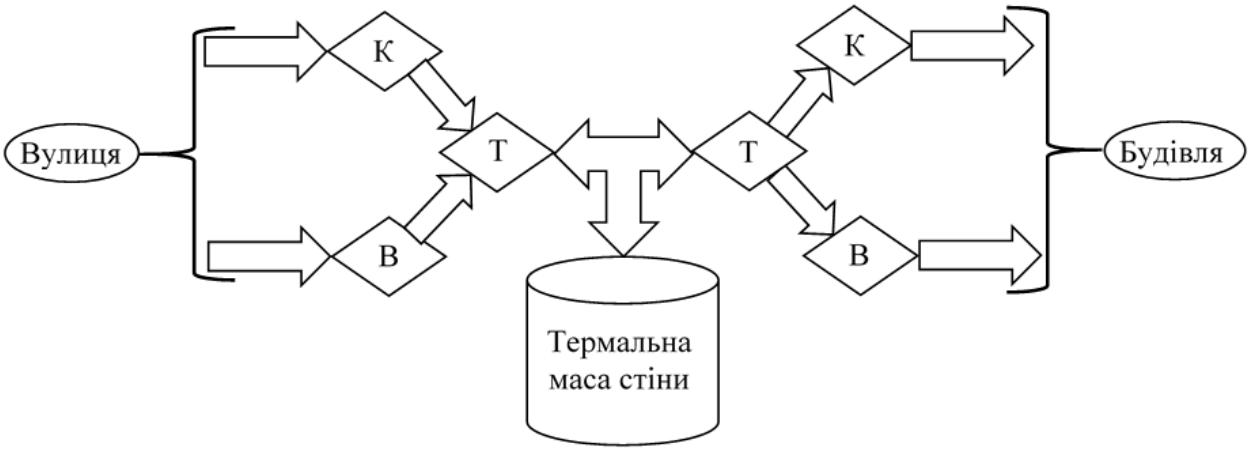

Рисунок 2.3 – Структурна схема стіни

Схема підлоги (рис. 2.4) відрізняється від схеми стіни відсутністю блоків теплопередачі конвекцією та випромінюванням зі сторони зовнішнього середовища, так як фундамент будівлі, частіше за все, розташовується на ґрунті.

Схема вікон (рис. 2.5) відрізняється від схеми стін наявністю поглиненої склінням, а також пропущеної безпосередньо на внутрішні стіни сонячної радіації, тому додається ще один блок теплопередачі випромінюванням, дія якого спрямована безпосередньо на блок теплопередачі теплопровідністю стіни, що знаходиться за вікном. Відповідно, тепло до термальної маси внутрішнього повітря будівлі передається вже від внутрішніх стін.

Схема стіни Тромбе (рис. 2.6) відрізняється від схеми вікна наявністю повітряного прошарку між склінням та стіною Тромбе, де теплопередача відбувається за допомогою всіх трьох механізмів одночасно. Також, сонячна

радіація, пропущена склінням, потрапляє безпосередньо на акумулюючу стіну, від якої вже йде розподіл теплових потоків до термальної маси внутрішнього повітря.

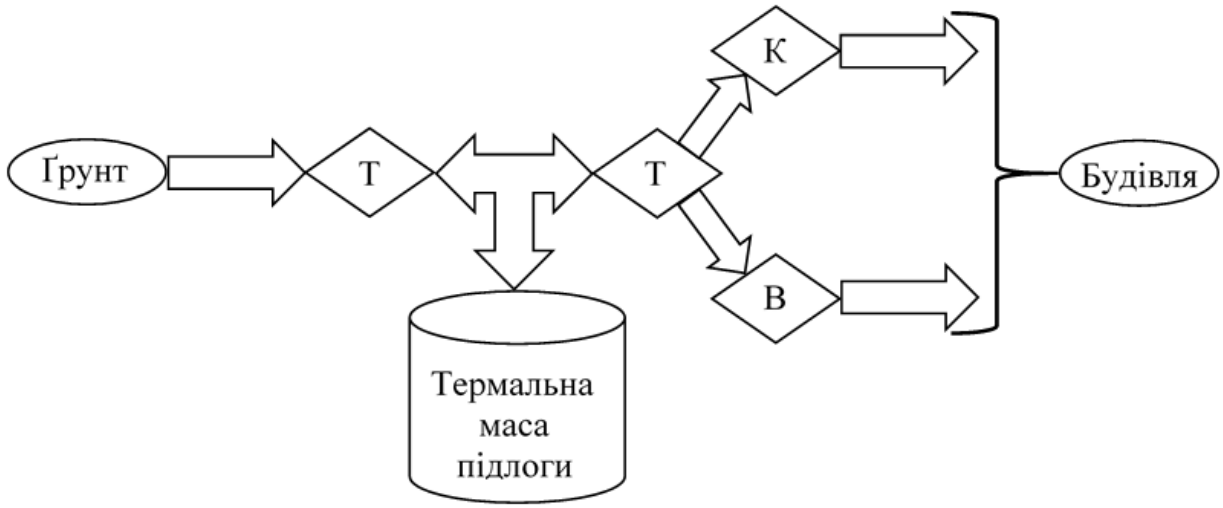

Рисунок 2.4 – Структурна схема підлоги

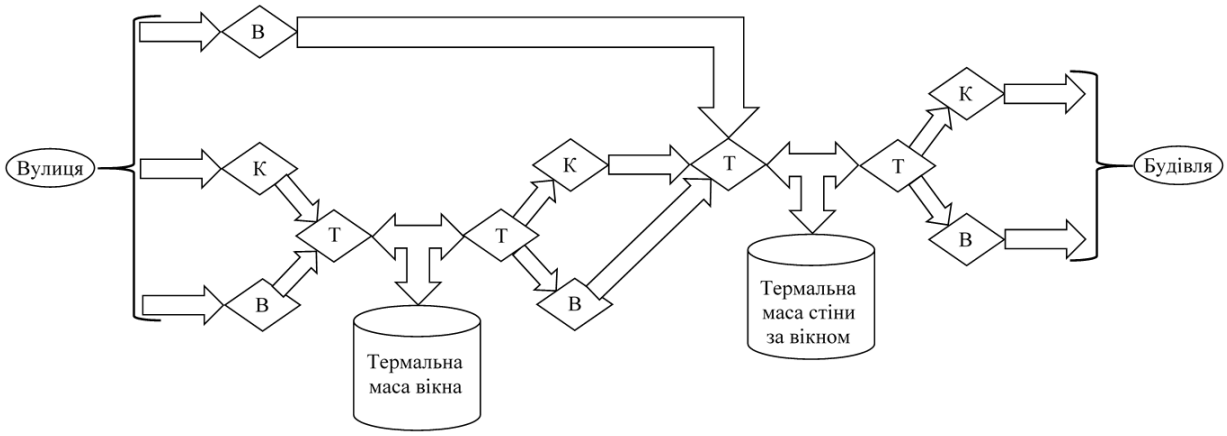

Рисунок 2.5 – Структурна схема вікон

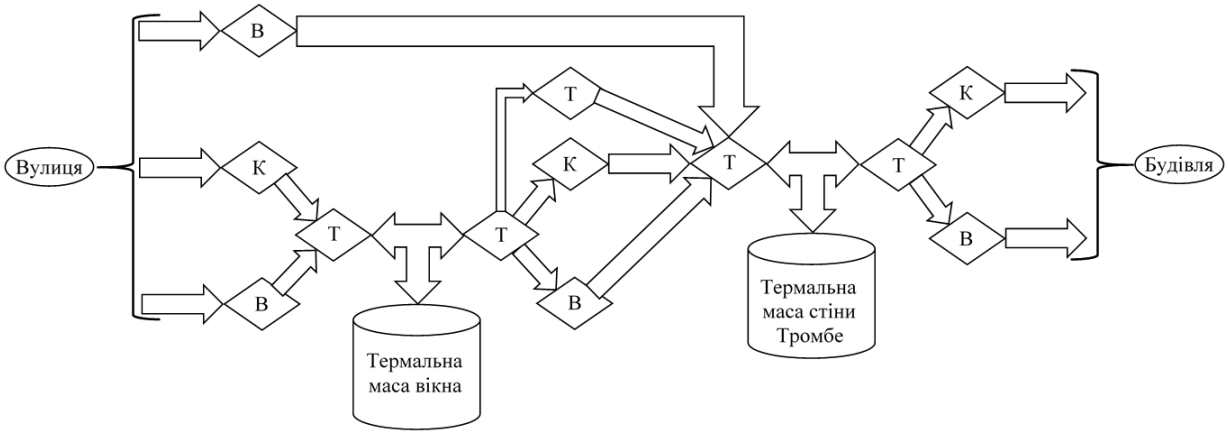

Рисунок 2.6 – Структурна схема стіни Тромбе

2.5 Висновок до другого розділу

У другому розділі розглянуто визначення комп’ютерної моделі, їх класифікацію. Визначено, що для створення моделі в даній роботі, матиме більшу актуальність детермінована, динамічна, дискретна модель. Це, у свою чергу, пов’язано з такими факторами як те, що використовуватимуться середні багаторічні дані про сонячну активність та температуру на певній місцевості, а також, параметри матеріалів, що є, загалом, незмінними. Це означає, що результати моделювання будуть визначеними заздалегідь, що зробить модель детермінованою. У моделі будуть враховані сезонні та добові коливання температури, що зробить модель динамічною. Враховується, що дані про середню багаторічну температуру приводяться для типового дня місяця погодинно, виходячи з цього, логічно буде прийняти певний крок розрахунку, що буде дорівнювати одній годині та розраховувати всі характеристики саме з таким кроком, що зробить дану модель дискретною.

Досліджена важливість визначення енергетичних потреб та їх вплив на енергетичну ефективність будівель. Також, визначена можливість використання комп’ютерних моделей для цієї цілі.

Розглянуто житлову будівлю, а також, пасивну систему опалення, як об’єкт моделювання та дослідження з точки зору теплових процесів, що відбуваються всередині. Для цього, представлені структурні схеми, що складаються з блоків, кожен із яких представляє окремий процес теплопередачі. На основі приведених схем буде побудована система, розрахунок температурних параметрів якої є метою створення програми.

РОЗДІЛ 3
МЕТОДИ ДОСЛІДЖЕННЯ ПРОЦЕСІВ ТЕПЛОПЕРЕДАЧІ

3.1 Числові методи для дослідження теплопередачі

Числовий аналіз – це розділ математики, який відповідає за розробку ефективних способів пошуку числових розв’язків складних математичних задач. Більшість математичних проблем науки та техніки дуже складні, і іноді їх неможливо вирішити безпосередньо. Тому дослідження складної математичної задачі є дуже важливим, щоб полегшити її вирішення.

Методи числового аналізу в основному використовується в галузі математики та інформатики, де створюються, аналізуються та впроваджуються алгоритми для розв’язування чисельних задач неперервної математики. Такі типи проблем, як правило, походять із реальних застосувань алгебри, геометрії та числення, а також включають змінні, які постійно змінюються. Запровадження числового аналізу протягом останніх півстоліття, зростання потужності та доступності цифрових комп’ютерів призвели до все більшого використання реалістичних математичних моделей у науці та техніці.

Числові методи – це методи, які використовуються для апроксимації математичних задач. Апроксимація необхідна через те, що задачу неможливо розв’язати процедуру, або через його складність та об’ємність [28, 29].

У процесі розв’язання задачі можна виділити декілька чітких етапів. Перший етап – формулювання. Формулюючи математичну модель фізичної ситуації, науковці повинні враховувати той факт, що вони розраховують вирішити задачу за допомогою потужностей комп’ютера. Таким чином, вони забезпечуватимуть конкретні цілі, належні вхідні дані, адекватні перевірки, а також тип і об’єм результату [2, 3, 28, 29].

Після того, як задачу сформульовано, необхідно розробити числові методи разом із попереднім аналізом похибок для вирішення проблеми. Числовий метод, за допомогою якого можна розв’язати задачу, називається алгоритмом. Вибір або

створення необхідного алгоритму залежить від початкових умов та поставлених цілей задачі.

Далі, необхідно перетворити алгоритм в набір послідовних команд, виконання яких приведе до очікуваного результату – створити комп'ютерну програму. Складність програми буде залежати від складності задачі, кількості описаних процесів, бажаної точності розрахунків.

Основні характеристики обчислювання при використанні числових методів [2, 3, 28, 29]:

- Точність: кожен числовий метод вносить похибки.
- Ефективність: кількість зусиль, необхідних як людям, так і комп'ютерам для використання методу.
- Чисельна нестабільність: ще однією проблемою чисельних методів є чисельна нестабільність. Помилки, включені в обчислення, з будь-якого джерела, збільшуються по-різному. У деяких випадках, ці помилки з'являються дуже швидко, що призводить до катастрофічних результатів.

Отже, процес вирішення певної задачі за допомогою числового методу виглядає наступним чином:

- Побудова математичної моделі.
- Побудова відповідної числової системи.
- Реалізація рішення.
- Перевірка рішення.

Дослідження теплопередачі за допомогою числових методів включає в себе процедури розв'язання набору алгебраїчних рівнянь, які наближено до диференціальних або інтегральних рівнянь, що описують теплопровідність, конвекцію та випромінювання, за допомогою комп'ютера [30-32].

Зазвичай, метою будь-якого розрахунку теплопередачі є визначення величин теплової енергії, що надходять до або відходять від якоїсь поверхні чи об'єкта. У задачах теплопровідності це вимагає визначення градієнта температури в матеріалі на його поверхні. У задачах конвекції градієнт температури в потоці, що тече

поверхнею, необхідний для отримання значення теплового потоку на цій поверхні. В обох випадках, першим кроком є визначення повного розподілу температури в цікавій області, а в конвекції також необхідно знайти розподіл швидкості. Таким чином, необхідний повний розв'язок рівняння енергії і, можливо, також рівнянь руху.

Теплове випромінювання дещо відрізняється від поверхонь, розділених потоками, які можуть або не брати участь у випромінюванні. Якщо потік прозорий і якщо відомі температури поверхонь, явища випромінювання та конвекції можуть бути розв’язані окремо. Якщо температури поверхні не вказані, але їх слід знайти як частину розчину, або якщо рідина поглинає або випромінює променисту енергію, тоді ці два явища пов’язані. У такому випадку, на додаток до диференціальних рівнянь для конвекції та теплопровідності, необхідне розв’язання алгебраїчних та інтегральних рівнянь для випромінювання [30-32].

Рівняння, що описують теплопередачу, є складними і мають деякі або всі з наступних характеристик: вони нелінійні; вони містять алгебраїчні, часткові диференціальні або інтегральні рівняння; вони складають пов'язану систему; властивості задіяних речовин зазвичай є функціями температури і можуть бути функціями тиску; область рішення, як правило, не є простим квадратом, колом або коробкою; і може (у проблемах, пов’язаних із твердінням, плавленням тощо) змінювати розмір і форму заздалегідь невідомим чином. Таким чином, аналітичні методи, що ведуть до точних рішень закритої форми, майже завжди недоступні [30-32].

Можливі два підходи. У першому випадку рівняння спрощуються – наприклад, шляхом лінеаризації, або нехтування членами, які вважаються досить малими, або шляхом припущення постійних властивостей, або за допомогою будь-якої іншої техніки, доки не буде отримано рівняння або систему рівнянь, для яких можна знайти аналітичне рішення. Можна сказати, що для наближеної задачі буде отримано точний аналітичний розв’язок. Рішення певною мірою буде помилковим, і зазвичай неможливо оцінити величину цієї помилки без звернення до зовнішньої інформації, такої як результат експерименту.

Другий підхід полягає у використанні числового методу. Для цього область безперервного рішення в більшості методів замінюється мережею або сіткою ліній і елементів. Змінні величини – температура, швидкість тощо, отримуються не в нескінченній кількості точок в області рішення, а лише в кінцевій кількості вузлів сітки або в точках у межах кінцевої кількості елементів. Диференційні рівняння замінюються набором лінійних (або, рідше, нелінійних) алгебраїчних рівнянь, які потрібно і можна розв’язувати за допомогою комп’ютера.

У цьому підході також є неточності: наприклад, якщо похідні замінити кінцевими різницями, буде отримано лише наближене значення для похідної. Проте, якщо задача поставлена правильно і метод розв’язку добре розроблений, ця помилка наближатиметься до нуля, оскільки сітка стає все точнішою. Загалом, похибка може бути оцінена, а також зменшена ціною збільшення потужності та часу роботи комп’ютера. Тож, для точної задачі буде отримано наближене рішення.

Числові методи вирішують задачі теплообміну поетапними ітераційними методами. Числові властивості, переваги, недоліки та математичні формулювання кожного чисельного методу відрізняються. Однак, загальна мета всіх чисельних методів у задачах теплообміну – отримати наближений розв’язок за найкоротший проміжок часу. Для розв’язання задач теплообміну використовуються різні числові методи, такі як метод кінцевих різниць, метод кінцевих об’ємів, метод кінцевих елементів [40-42].

Теплопередача регулюється математичними рівняннями, які можна застосовувати для вирішення проблем теплового потоку в різних застосуваннях. Залежно від задачі теплопередачі математичний вираз для теплопередачі відрізняється різними початковими та граничними умовами. Як правило, рівняння теплопередачі має форму часткових похідних. Його можна використовувати для визначення зміни часу, протягом якого тепло поширюється в просторі.

Для вирішення простих задач теплообміну, пов’язаних із простою геометрією та граничними умовами, можна використовувати аналітичні методи. Аналітичні методи дають функцію рішення, що відповідає кожній точці середовища. Аналітичні математичні методи використовуються для пошуку

розв’язків рівнянь теплопередачі. Зі збільшенням складності задачі теплопередачі кількість змінних у математичному рівнянні також збільшується. Непрактично щоразу використовувати аналітичні методи для отримання розчину теплового потоку.

Реальні ситуації теплопередачі пов’язані зі складною геометрією та складними граничними умовами або змінними властивостями. При використанні аналітичних методів для вирішення складних проблем теплообміну рішення можуть бути недоступними або взагалі не існувати. Для отримання наближеного рішення в таких випадках необхідно конвертувати рішення задачі теплообміну. При використанні аналітичного методу ітерація та збіжність стають виснажливими та важкими. Числові методи можуть допомогти в досягненні достатньо точних наближених рішень для складних рівнянь теплообміну швидшим і точнішим способом [30-32].

Числові методи використовуються для розв’язування складних задач теплообміну за участю таких механізмів, як провідність, конвекція, випромінювання або їх комбінація. Існують різні чисельні методи, такі як метод кінцевих елементів, метод кінцевих об’ємів, метод кінцевих різниць і метод граничних елементів. Усі ці методи відрізняються числовими властивостями, математичними формулюваннями, перевагами та недоліками. Загальною метою будь-якого чисельного методу, який використовується для розв’язання задач теплообміну, є досягнення наближеного розв’язку керівних рівнянь для формування результату закритої форми за мінімальний час, який неможливо отримати за допомогою аналітичних методів [30-32].

У фізично складній задачі теплообміну, якщо застосовується неправильний числовий метод, витрачається набагато більше часу, щоб підтвердити правильність процедури. Неправильний числовий метод не зможе дослідити складність явища фізичного теплообміну. Крім того, точність отриманих результатів буде сумнівною. Від якості дискретизації залежить точність числового рішення. Тому важливо вибрати правильний чисельний метод для вирішення складних задач теплообміну [30-32].

Вибір найкращого чисельного методу для розв’язання рівнянь теплопередачі може бути складним, оскільки необхідно враховувати багато факторів, таких як точність, ефективність, стабільність і міцність.

Оцінюючи чисельний метод, важливо враховувати його збіжність, точність, стабільність, ефективність і стійкість. Збіжність стосується чисельного розв’язку, що наближається до точного розв’язку зі збільшенням кількості точок або елементів дискретизації. Точність вимірює, наскільки чисельний розв’язок близький до точного розв’язку для даного розміру або порядку дискретизації. Стабільність гарантує, що чисельне рішення залишається обмеженим і послідовним, оскільки розмір дискретизації або часовий крок зменшуються. Ефективність оцінює, скільки обчислювального часу та пам’яті потрібно для певного рівня точності. Нарешті, надійність перевіряє, наскільки добре чисельний метод обробляє складні геометрії, нелінійності, розриви та сингулярності в задачі.

Вибираючи чисельний метод розв’язання задачі теплопередачі, необхідно визначити тип теплопередачі, керівні рівняння, область визначення та граничні умови, а також фізичні властивості. Після цього, необхідно порівняти доступні методи на основі їх придатності та продуктивності для певної задачі. Загалом, методи скінченних різниць прості у реалізації, але можуть мати проблеми зі складною геометрією, нелінійністю та граничними умовами. Методи кінцевих елементів є гнучкими та точними, але можуть вимагати більше обчислювальних ресурсів і складних методів сітки. Методи кінцевого об’єму є консервативними та надійними, але можуть мати проблеми з точністю та стабільністю.

Для перевірки та підтвердження числового методу необхідно перевірити якість і надійність числового рішення різними методами. Аналітичні розв’язки можна використовувати для порівняння чисельного розв’язку з точним розв’язком для простої чи спрощеної версії задачі та обчислення похибки та коефіцієнта збіжності. Числові експерименти можна проводити, щоб змінювати розмір, порядок або схему дискретизації та спостерігати вплив на числове рішення. Еталонні задачі можна використовувати для порівняння чисельного розв’язку з результатами інших чисельних методів або експериментальними даними для

стандартної чи добре відомої задачі. Крім того, можна використовувати аналіз чутливості для зміни вхідних параметрів і вивчення їх впливу на числове рішення.

Щоб удосконалити числовий метод, необхідно визначити та усунути джерела помилок. Помилка дискретизації, наприклад, може бути зменшена шляхом збільшення розміру або порядку дискретизації, використання схем вищого порядку або адаптивних схем або застосування методів оцінки помилки та корекції. Похибку округлення можна мінімізувати, використовуючи відповідні типи даних, уникаючи віднімання майже рівних чисел або використовуючи методи контролю помилок. Крім того, помилку ітерації можна зменшити шляхом вибору відповідного критерію збіжності, використання швидшого чи більш стабільного ітераційного методу або застосування методів попереднього прискорення. Помилка алгоритму може бути усунена шляхом налагодження, тестування та документування коду, використання надійних і перевірених бібліотек програмного забезпечення або застосування методів перевірки та перевірки коду [30-32].

3.2 Рівняння теплопередачі

Рівняння, що представляє залежність теплопередачі від часу (без присутності внутрішніх джерел тепла), має кілька назв, таких як рівняння теплопровідності, рівняння дифузії та рівняння Фур'є. Але більш важливо те, що воно класифікується як лінійне, параболічне рівняння в частинних похідних. Ця класифікація використовується для деталізації набору математичних методів, які можуть бути використані для отримання точних рішень. Вигляд цього рівняння наступний [33-41]:

$$\frac{\partial u}{\partial t} - \alpha \nabla^2 u = 0 \qquad (3.1)$$

де:

u – температура в будь-якій точці простору;

t – час;

α – тепловий коефіцієнт;

∇ – оператор Лапласа.

У свою чергу [33-37]:

$$\alpha = \frac{\lambda}{\rho c} \tag{3.2}$$

де:

λ – коефіцієнт теплопровідності *Вт/(м·К)*;

ρ – густина, *кг/м*^{3};

c – питома теплоємність, *Дж/(кг·К)*.

Оператор Лапласа для тривимірного простору:

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} \tag{3.3}$$

Для випадків у меншій кількості вимірів, відсутня координата просто виводиться з рівняння (3.3).

Рівняння теплопровідності зазвичай розв’язується як початково-гранична задача, що означає, що рівняння має супроводжуватися [33-41]:

- областю розв’язку;
- набором умов, що визначають або розв’язок, або його похідні вздовж границі;
- початковими умовами, що показують розв’язок у кожній точці області на момент початку.

Приклад класичного одновимірного рівняння теплопередачі наведений нижче [33-41]:

$$\begin{aligned}&\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}; u = u(x,t);\ 0 < x < 1, t > 0;\\&u(0,t) = 0;\ u(1,t) = 0;\\&u(x,0) = f(x).\end{aligned} \tag{3.4}$$

Тут коефіцієнт теплопровідності встановлюється рівним одиниці. Часова та просторова області чітко описані, а граничні умови, складається рівно з двох точок простору для задачі, наведені у другому рядку. Початкові умови окреслені в останньому рядку. Для кусково-неперервної функції *f(x)* цю задачу легко розв'язати розділенням змінних і використанням рядів Фур'є [33-41]:

$$\begin{aligned}&u(x,t) = \sum_{n=1}^{\infty} A_n \exp(-n^2\pi^2 t)\sin(n\pi x)\\&A_n = 2\int_0^1 f(x)\sin(n\pi x)\,dx\end{aligned} \tag{3.5}$$

У багатьох випадках, можна отримати точні розв'язки рівняння теплопередачі, але лише для задач, поставлених у простих ортогональних системах координат. Для інших систем координат, необхідно застосовувати числові методи [33-41].

3.3 Типи сіток для числових методів

Сітка – це дискретизація геометричної області на малі прості фігури, такі як трикутники чи чотирикутники у двох вимірах і тетраедри чи гексаедри у трьох вимірах. Сітки знаходять застосування в багатьох сферах. Наприклад, у географії та картографії сітки дають компактне представлення даних місцевості, у комп'ютерній графіці більшість об'єктів зрештою зводяться до сіток перед рендерингом. Для чисельного розв'язання диференціальних рівнянь у частинних

похідних, що виникають у фізичному моделюванні, сітки є майже необхідними [42, 43].

По-перше, можливі входи розділяються за розмірністю на дво- чи три-розмірні. Виділяють чотири типи плоских областей, які показано на рис. 3.1. Простий багатокутник включає як границю, так і внутрішню частину. Багатокутник з отворами – це простий багатокутник без деяких внутрішніх частин, що являють собою інші прості багатокутники; його границя має більше однієї зв’язної компоненти. Множинний домен є ще більш загальним, він допускає внутрішні границі; фактично такою областю може бути будь-який плоский прямолінійний граф, у якому нескінченна грань обмежена простим циклом. Кілька доменів моделюють об’єкти, виготовлені з кількох матеріалів. Вигнуті домени допускають сторони, які є алгебраїчними кривими, такими як сплайни. Як і в перших трьох випадках, спільно відомих як полігональні домени, вигнуті домени можуть містити або не містити отвори та внутрішні межі [42].

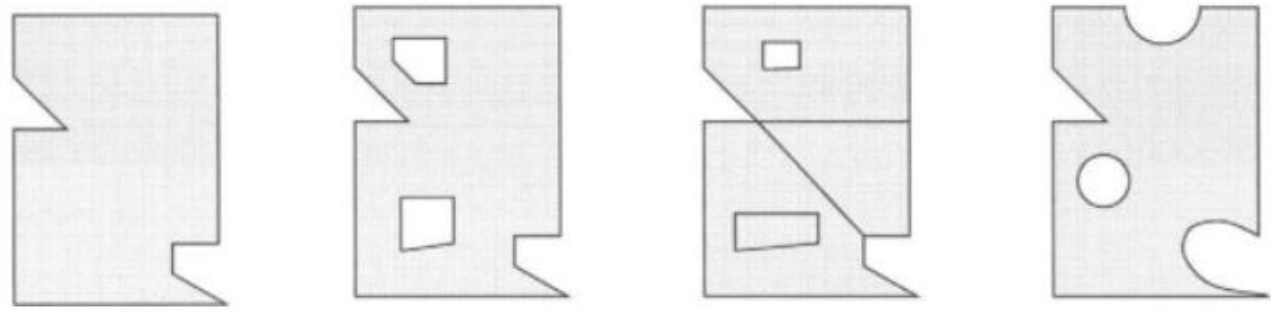

Рисунок 3.1 – Типи двовимірних вхідних даних: простий багатокутник, багатокутник з отворами, множинний домен, вигнута область [42]

Тривимірні входи мають аналогічні типи. Простий многогранник топологічно еквівалентний кулі. Загальний багатогранник може бути багатозв’язним, що означає, що він топологічно еквівалентний суцільному тору або іншому тілу вищого роду; він також може мати порожнини, що означає, що його межа може мати більше одного зв’язаного компонента. Однак припускається, що в кожній точці на межі загального багатогранника досить мала сфера охоплює один зв’язаний шматок внутрішньої сторони багатогранника та один пов’язаний шматок

його зовнішнього вигляду. Нарешті, є багатогранні домени – загальні багатогранники з внутрішніми межами і тривимірні вигнуті домени, які зазвичай мають межі, визначені сплайновими ділянками [42].

Побудова та моделювання геометрії домену виходять за межі цієї глави, тому ми просто припустимо, що домени задані в якомусь вигляді граничного представлення, без уточнення точної форми цього представлення. Обчислювальні геометри зазвичай передбачають точні, комбінаторні структури даних, такі як пов’язані списки для простих багатокутників і багатокутників з отворами, списки з подвійним зв’язком ребер або чотиригранні структури для плоских множинних областей, а також крилаті реберні структури для багатогранних областей [42].

На практиці складні домени проектуються в системах автоматизованого проектування. Ці системи використовують представлення поверхні, призначені для візуального відтворення, а потім перекладають остаточний дизайн в інший формат для введення в генератор сітки. Альтернативним підходом до перекладу формату є пряме надсилання запитів системі автоматизованого проектування за допомогою запитів на огородження, а потім створення нового представлення на основі відповідей на ці запити. Цей підхід є найбільш вигідним, коли проблема перекладання складна, як це може бути у випадку неявних поверхонь (наборів рівнів складних функцій) або формул конструктивної твердої геометрії. З будь-яким підходом, модель має бути топологічно правильною та достатньо точною, щоб уможливити створення сітки. У майбутньому очікується більша інтеграція твердотільного моделювання та сітки [42].

Основний поділ між типами сіток є на структуровані та неструктуровані (рис. 3.2). Структурована сітка – це сітка, у якій усі внутрішні вершини топологічно однакові. З точки зору теорії графів, структурована сітка є індукованим підграфом нескінченного періодичного графа, такого як сітка. Така сітка є найпростішоюдля створення. Неструктурована сітка – це сітка, у якій вершини можуть мати довільні локальні околиці. Блоково-структурована або гібридна сітка утворена кількома невеликими структурованими сітками, об’єднаними в загальний неструктурований візерунок [42, 43].

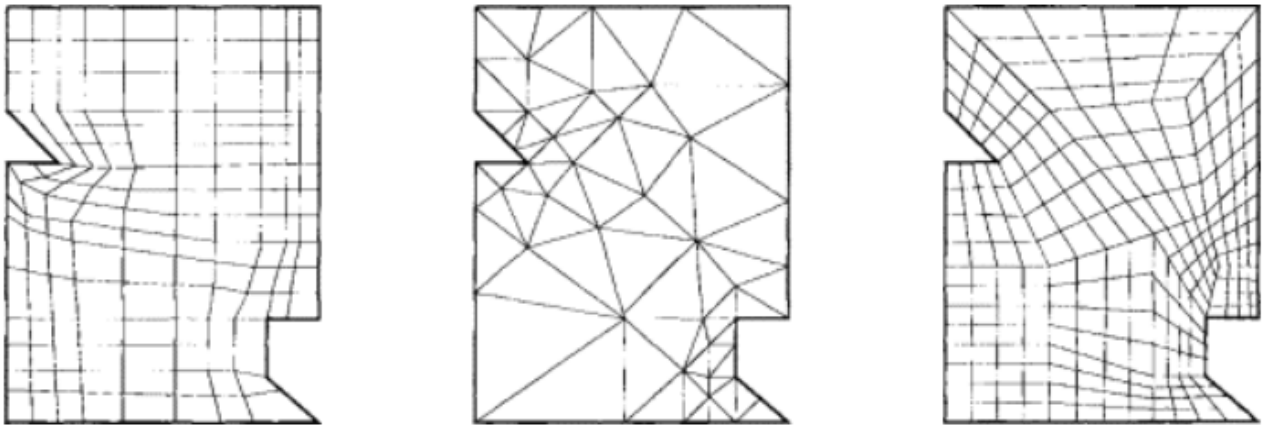

Рисунок 3.2 – Типи сіток: структурована; неструктурована; блочна [42]

Загалом, структуровані сітки пропонують простоту та легкий доступ до даних, тоді як неструктуровані сітки пропонують зручнішу адаптацію сітки (уточнення/зменшення визначень на основі початкового рішення) і краще підходять для складних доменів. Високоякісні гібридні сітки користуються перевагами обох підходів, але гібридна сітка не є повністю автоматичною [42, 43].

Поділ між структурованими та неструктурованими сітками зазвичай поширюється на форму елементів: у двовимірних структурованих сітках зазвичай використовуються чотирикутники, тоді як у неструктурованих сітках використовуються трикутники. У трьох вимірах аналогічними формами елементів є гексаедри, тобто топологічні куби, і тетраедри. Однак немає суттєвої причини для структурованих і неструктурованих сіток використовувати різні форми елементів. Насправді можна розділити елементи для перетворення між трикутниками та чотирикутниками (рис. 3.3), а також між тетраедрами та гексаедрами [42].

У двовимірній структурованій сітці точки сітки зазвичай адресуються індексами (i, j), де індекс i представляє точки, які проходять у напрямку x, тоді як індекс j представляє точки, які проходять у напрямку y. Якщо (i, j) є індексами для певної точки, то сусідні точки, безпосередньо праворуч, ліворуч, зверху і знизу визначаються збільшенням або зменшенням одного з індексів на одиницю. Призначаючи відповідні дискретні значення для Δx_i і Δy_j, координати в напрямку x

і *y* всередині фізичного простору можуть бути визначені поступово, в результаті чого прямокутна сітка охоплює всю область [42, 43].

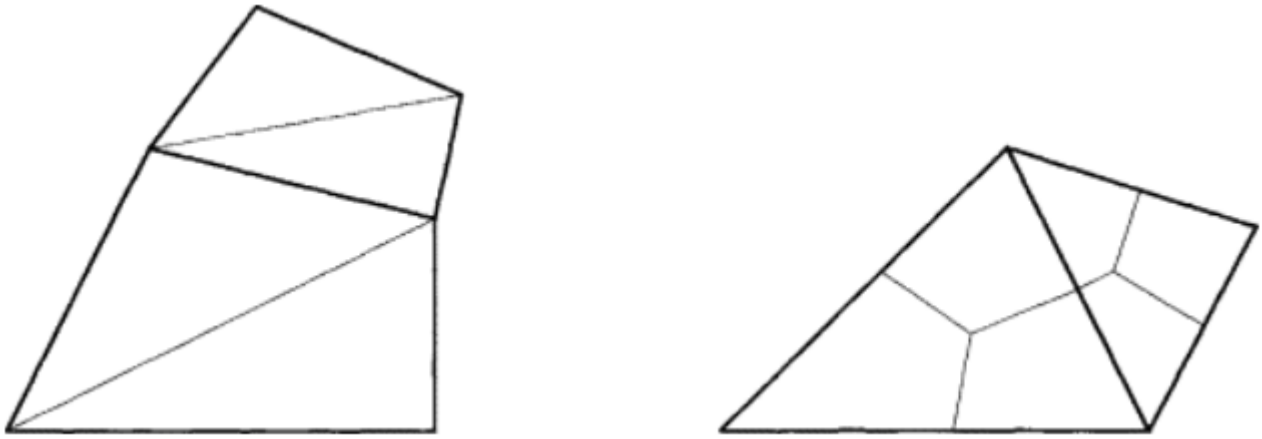

Рисунок 3.3 – Трикутники чотирикутників; Розбиття трикутників на чотирикутники [42]

Для рівномірно розподілених точок сітки відстань Δx_i або Δy_j, по суті є єдиним репрезентативним значенням у напрямку *x* або *y*. Однак, для нерівномірно розподілених точок сітки, відстань між Δx_i або Δy_j може прийняти декілька дискретних значень, отже, можливе створення сітки з абсолютно нерівним інтервалом у напрямку *x* або *y*. Наприклад, дрібна сітка в напрямку *x* може бути створена для адекватного розділення в'язкого прикордонного шару потоку поблизу геометрії стінки, тоді як рівномірно розподілена сітка зберігається в напрямку *y*, як показано на рис. 3.4. Це конкретне розташування зазвичай розглядається як розтягнута або концентрована сітка, де точки сітки можна вважати зміщеними до меж стіни.

У трьох вимірах будь-яку точку сітки в просторі можна адресувати за допомогою індексів (*i*, *j*, *k*), де введення індексу *k* представляє точки, що рухаються в напрямку *z*. Розгляд додаткового виміру тепер вимагає знання інтервалу Δz_k, на додаток до інтервалу Δx_i та інтервалу Δy_j, для побудови відповідної сітки, що покриває тривимірну геометрію [42, 43].

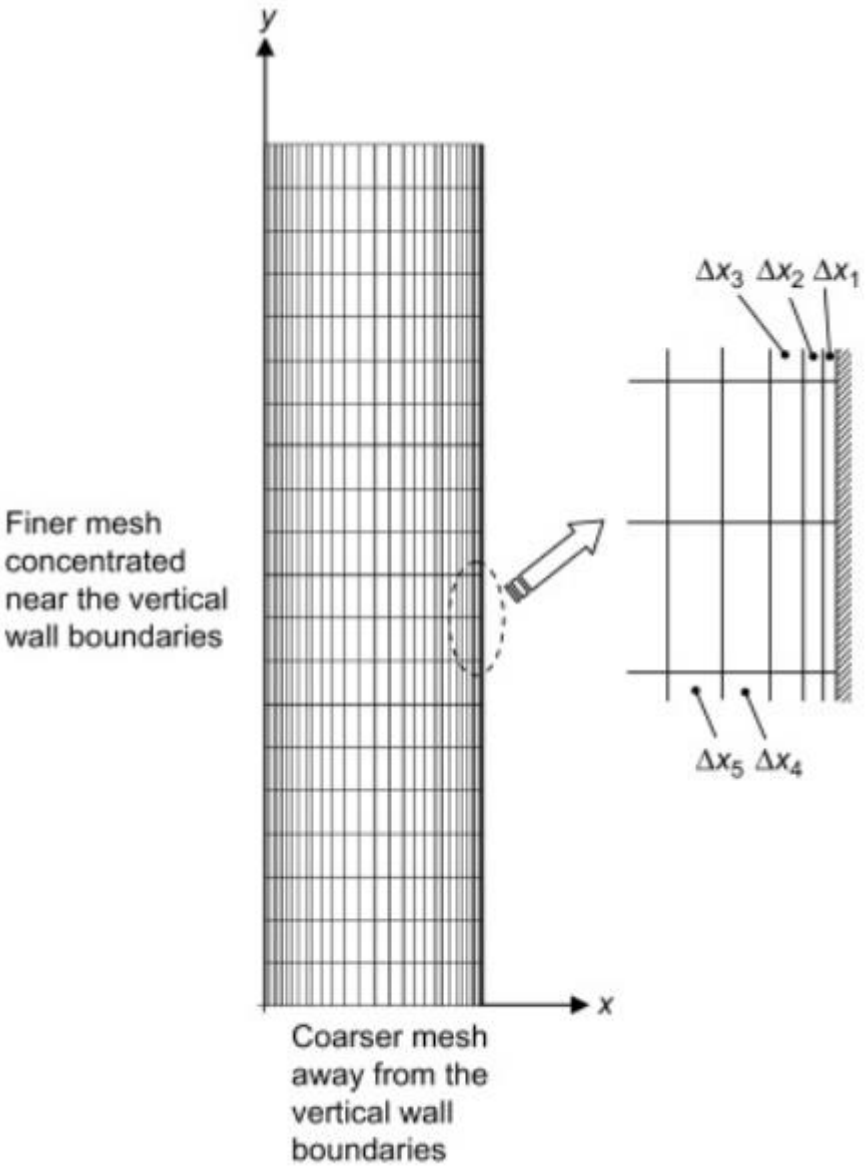

Рисунок 3.4 – Нерівномірна прямокутна сітка [43]

Приклад сітки, що використовує сходинки для відтворення геометрії перпендикулярного вигину показано на рис. 3.5. Для того, щоб застосувати ортогональну сітку, таку як структурована сітка, до геометрії, потрібно досягти компромісів, особливо на вигнутій ділянці, охарактеризувавши межі через сходи, подібні до сходів. Тим не менш, такий підхід породжує дві проблеми. По-перше, такий приблизний опис кордону є трудомістким і займає досить багато часу. По-друге, східці на межі можуть внести похибки в обчислення напружень стінки, теплових потоків, ефектів граничного шару тощо. Обробка граничних умов на ступінчастих стінках, зазвичай, вимагає тонкої декартової сітки для покриття

областей стінки, але вимога до високорегулярної структури ліній сітки може спричинити подальшу витрату потужностей комп’ютера через непотрібне вдосконалення внутрішніх областей, які становлять мінімальний інтерес. Сітчасті системи, засновані на декартових системах координат, мають серйозні обмеження в нерегулярних геометріях. Тому було б більш вигідно використовувати сітки, які більш природно справляються з кривизною та геометричною складністю [43].

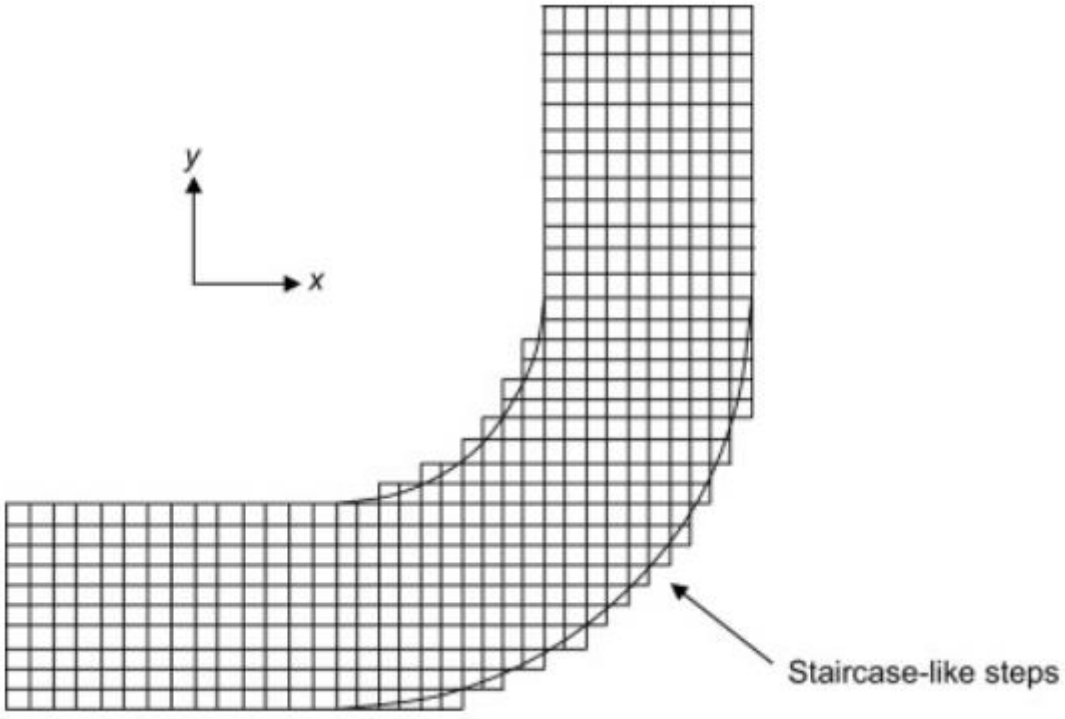

Рисунок 3.5 – Приклад сітки, що використовує сходинки для геометрії перпендикулярного вигину [43]

Коли до геометрії перпендикулярного вигину застосовано сітку, що відповідає формі тіла, можна вважати, що стінки збігаються з лініями постійного η (рис. 3.6). Розташування вздовж геометрії, від A до B або D до C, згодом відповідає конкретним значенням ξ в обчислювальній області. Відповідні точки на AB і CD, з’єднані окремою лінією η, матимуть однакове значення ξ_i, але різні значення η. У певній точці (i, j) уздовж цієї лінії η, $\xi = \xi_i$ і $\eta = \eta_j$. Відповідна точка $x=x(\xi_i, \eta_j)$ і $y=y(\xi_i, \eta_j)$ в обчислювальній області існує у фізичній області [43].

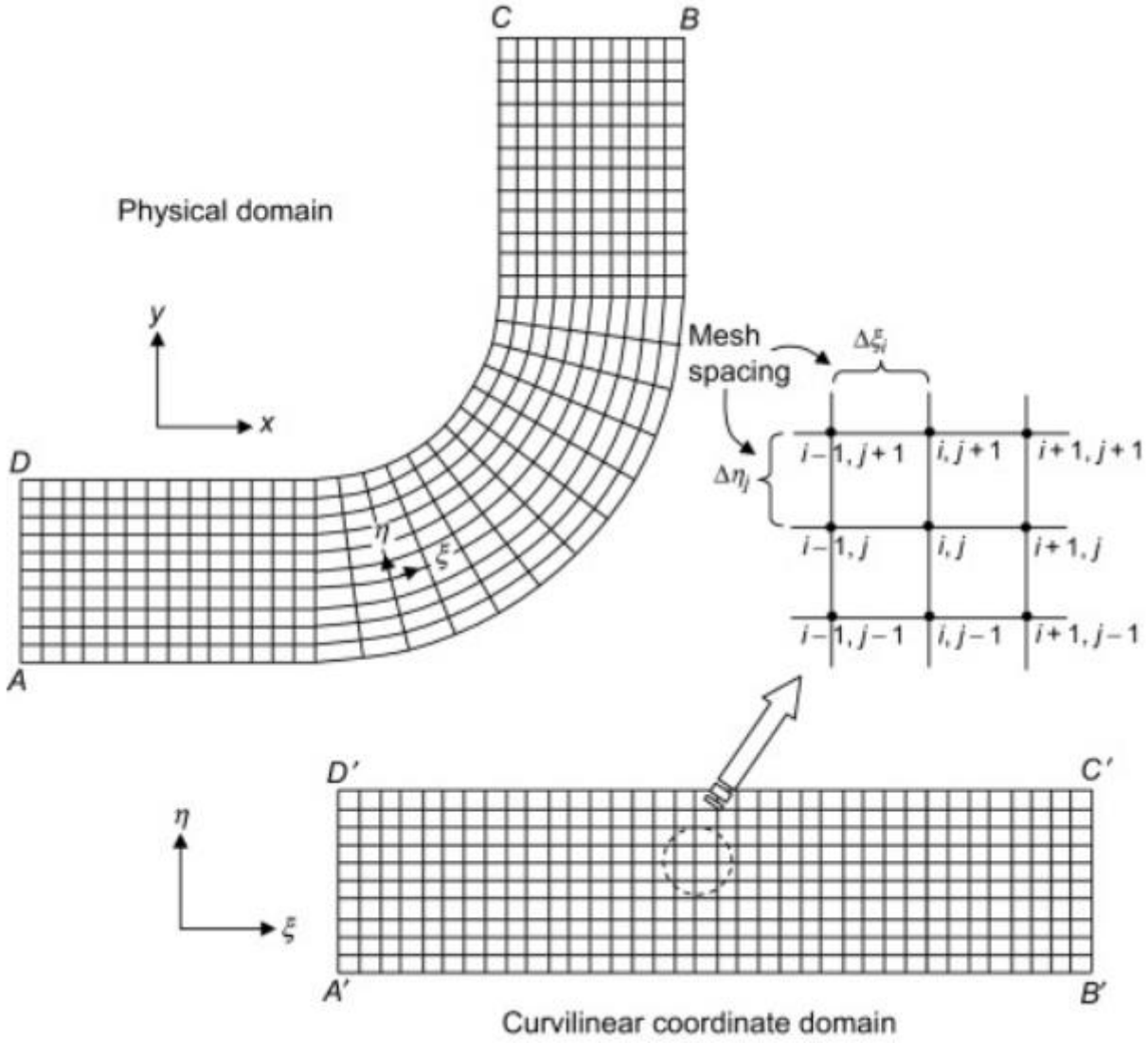

Рисунок 3.6 – Приклад підігнаної до тіла або криволінійної сітки для геометрії перпендикулярного вигину та відповідної обчислювальної геометрії [43]

На рис. 3.6 трансформація геометрії перпендикулярного вигину має бути визначена таким чином, щоб існувала однозначна відповідність між прямокутною сіткою в обчислювальній області та криволінійною сіткою у фізичній області. Алгебраїчні форми керівних рівнянь для задач потоку виконуються в обчислювальній області, яка має рівномірний інтервал $\Delta\xi$ і рівномірний інтервал $\Delta\eta$. Потім, обчислена інформація безпосередньо надходить у фізичну область через однозначну відповідність точок сітки. Через необхідність розв’язувати рівняння в обчислювальній області керівні рівняння повинні бути виражені в термінах

криволінійних координат, а не в декартових координатах, що означає, що вони повинні бути перетворені з (x, y) в (ξ, η) як нові незалежні змінні [43].

Конструкція сітки внутрішньої області геометрії перпендикулярного вигину, зазвичай, може бути досягнута за допомогою двох підходів. З одного боку, декартові координати можуть бути алгебраїчно визначені через інтерполяцію граничних значень. Ця методологія не вимагає ітераційної процедури, і вона є досить простою з обчислювальної точки зору. З іншого боку, система диференціальних рівнянь у частинних похідних із відповідними декартовими координатами може бути розв'язана числовими методами з набором граничних значень як граничних умов, щоб отримати дуже гладку сітку у фізичній області.

На додаток до використання сітки, пристосованої до тіла, можна створити неструктуровану сітку для заповнення внутрішньої області геометрії перпендикулярного вигину (рис. 3.7). На зображеному рисунку немає регулярності в розташуванні комірок у накладній сітці. Тут клітини є повністю неструктурованими, і немає координатних ліній, які відповідають криволінійним напрямкам ξ і η, як, наприклад, у сітці, пристосованій до тіла. Таким чином, забезпечується максимальна гнучкість у зіставленні комірок, особливо з сильно вигнутими межами, і необхідні комірки можна цілеспрямовано вставляти для визначення областей потоку, де вони найбільш важливі, наприклад, у областях із високими градієнтами. Трикутник і тетраедральна сітка є, безумовно, найпоширенішими формами створення неструктурованої сітки [43].

Як показано на рис. 3.7, початковий набір граничних вузлів геометрії можна ефективно тріангуляти відповідно до критерію тріангуляції Делоне. Тут найважливішою властивістю тріангуляції Делоне є те, що вона має властивість порожнього описаного кола. За визначенням, описане коло трикутника - це єдиний трикутник, який проходить через три його вершини. Тому тріангуляцію Делоне набору вершин можна розглядати як тріангуляцію (зазвичай, але не завжди, унікальну), у якій кожен трикутник має порожнє описане коло, тобто коло не охоплює жодної вершини тріангуляції. Можна показати, що описане коло кожного трикутника Делоне сітки, згенерованої для геометрії перпендикулярного вигину, є

порожнім. Усі алгоритми для обчислення тріангуляцій Делоне покладаються на швидкі операції для визначення того, коли точка сітки знаходиться в межах описаного кола трикутника, і ефективну структуру даних для зберігання трикутників і ребер. Найпростіший спосіб обчислення тріангуляції Делоне – це багаторазове додавання однієї вершини за раз, а потім повторна тріангуляція зачеплених частин. Коли додається вершина, виконується пошук усіх описаних кіл трикутника, які містять цю вершину. Потім видаляються трикутники, описані кола яких містять щойно вставлену точку. Потім уся нова тріангуляція формується шляхом приєднання нової точки до всіх граничних вершин порожнини, утвореної попереднім видаленням пересічних трикутників. Методи тріангуляції Делоне, засновані на вставці точок, природно поширюються на три виміри, розглядаючи описану сферу (описану сферу), пов’язану з тетраедром [43].

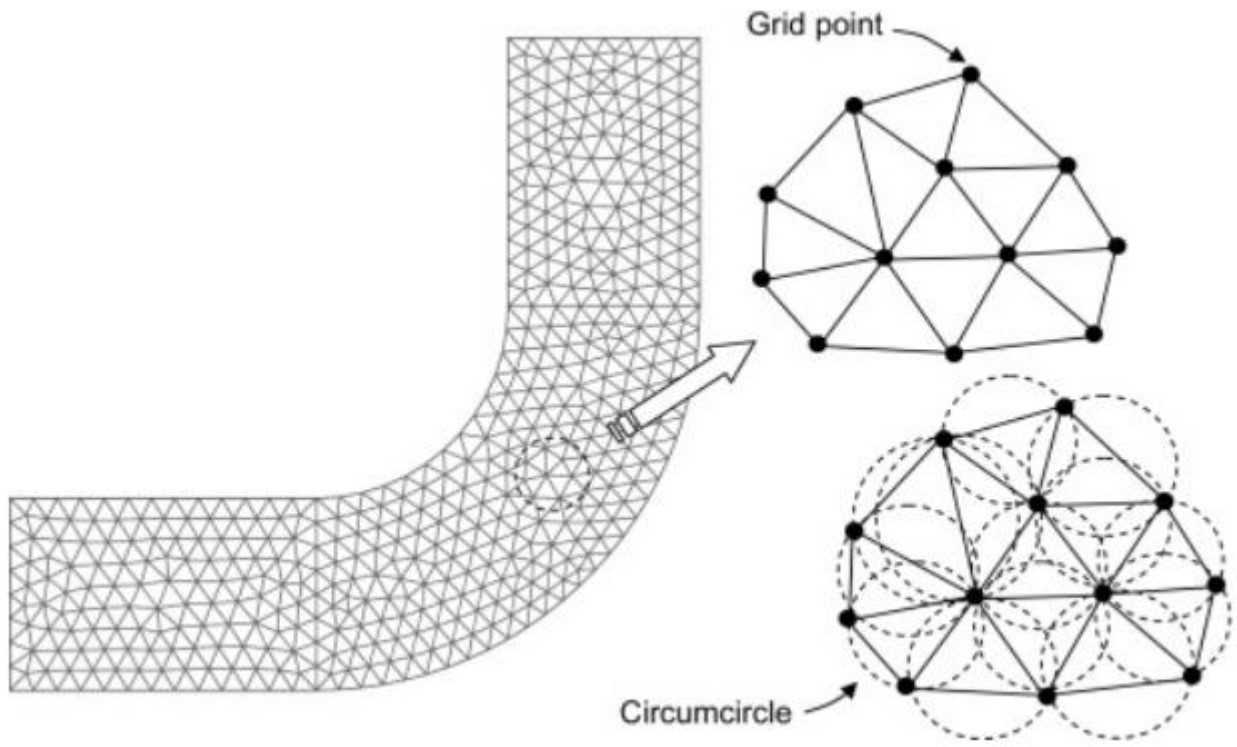

Рисунок 3.7 – Приклад трикутної сітки для геометрії перпендикулярного згину [43]

Іншим способом виконати такі обчислення, є використання методу просування фронту, його основна ідея полягає у створенні неструктурованої сітки

шляхом додавання окремих елементів по одному до існуючого фронту згенерованих елементів. Після створення граничних вузлів ці ребра формують початковий фронт, який потрібно просунути в поле. Трикутні комірки утворюються на кожному сегменті лінії, і, у свою чергу, ці клітини створюють додаткові сегменти лінії на передній частині. Таким чином, передня частина утворює стек, а краї безперервно додаються або видаляються зі стосу. Процес завершується, коли стек порожній, тобто коли всі фронти зливаються один з одним і домен повністю покритий. Для генерації тривимірної сітки поверхнева сітка спочатку будується шляхом створення двовимірної трикутної сітки на поверхневих границях області. Ця сітка утворює початковий фронт, який потім просувається у фізичний простір шляхом розміщення нових точок попереду фронту та формування тетраедричних елементів. Необхідна перевірка перетину тепер включає трикутні передні грані, а не ребра, як у двовимірному випадку.

В якості альтернативи, метод квадродерева/октодерева передбачає генерацію неструктурованої сітки через рекурсивне поділ фізичного простору до заданої (просторово змінної) роздільної здатності. Вершини отриманої структури квадродерева або октанта використовуються як точки сітки, а квадранти або октанти дерева поділяються на трикутні або тетраедральні елементи у двох або трьох вимірах відповідно. Слід, однак, згадати, що комірки дерева квадродерева/октодерева, які перетинають граничні поверхні та вершини на кордонах, повинні якимось чином бути зміщені або обгорнуті, щоб збігатися з межею. Цей метод є відносно простим і недорогим і створює якісну сітку у внутрішніх областях домену. Одним із недоліків методу є те, що він має тенденцію генерувати нерегулярний розподіл клітин поблизу кордонів [43].

Інша неструктурована сіткова система, яка викликає значний інтерес у розрахунковій динаміці потоків, це використання сітки, що містить багатогранні комірки. Багатогранну сітку можна створити шляхом об'єднання чотиригранних комірок у багатогранні. Враховуючи тетраедральну сітку, яку було створено для геометрії перпендикулярного вигину на рис. 3.7, багатогранну сітку, як показано на рис. 3.8, можна створити за допомогою агломерації клітин, що призводить до

значного зменшення загальної кількості клітин. Що ще важливіше, агломерація клітин має здатність покращувати вихідну сітку шляхом перетворення окремих областей із сильно перекошеними тетраедричними осередками на багатогранні, тим самим покращуючи якість сітки. Використання багатогранної сітки також призводить до більш швидкої збіжності числового рішення. Очевидною потенційною перевагою застосування багатогранної сітки є те, що вона дозволяє застосовувати гнучкість неструктурованої сітки до складної геометрії без обчислювальних витрат, пов’язаних із великою тетраедричною сіткою. Застосування багатогранної сітки набуває значного поширення в спільноті обчислювальної гідродинаміки. Поки було показано, що поліедральна сітка має значні переваги перед тетраедричною сіткою щодо досягнутої точності та ефективності чисельних обчислень [43].

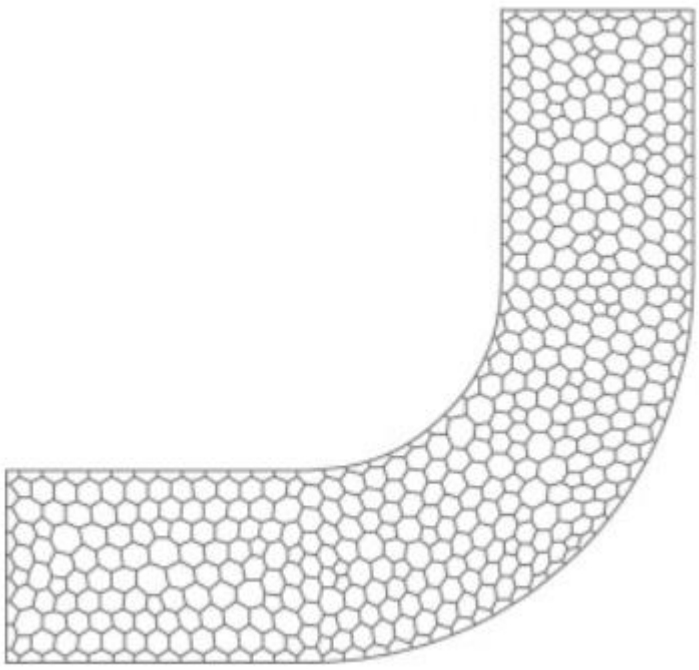

Рисунок 3.8 – Сітка, що складається з багатогранних комірок з геометрією перпендикулярного вигину [43]

3.4 Метод кінцевих різниць

У всіх числових розв’язках, неперервне диференціальне рівняння в частинних похідних замінено дискретним наближенням. Це означає, що числовий розв’язок відомий лише в кінцевій кількості точок у фізичній області. Кількість цих точок може вибрати користувач. Загалом, збільшення кількості точок збільшує не лише роздільну здатність, але й точність числового рішення [36, 37, 39-46].

Результатом дискретної апроксимації є набір алгебраїчних рівнянь, які обчислюються для значень дискретних невідомих. Сітка — це набір місць, де обчислюється дискретні розв’язки. Ці точки називаються вузлами, і якщо провести лінії між суміжними вузлами в області, отримане зображення буде нагадувати сітку. Двома ключовими параметрами сітки є Δx, локальна відстань між сусідніми точками в просторі, і Δt, локальна відстань між сусідніми часовими кроками [36, 37, 39-41, 44-48].

Основна ідея методу кінцевих різниць полягає в заміні безперервних похідних так званими різницевими формулами, які включають лише дискретні значення, пов’язані з положеннями на сітці.

Метод кінцевих різниць отримує наближений розв’язок для u(x, t) при кінцевому наборі x і t. Дискретні значення x розподілені в діапазоні $0 \leq x \leq L$, тому [36, 37, 39-41, 44-48]:

$$x_i = (i-1)\Delta x; i = 1,2, \dots N \tag{3.6}$$

де:

N – загальна кількість просторових вузлів.

Δx — розмір просторового кроку.

$$\Delta x = \frac{L}{N-1} \tag{3.7}$$

Аналогічно, дискретні значення t рівномірно розподілені в $0 \le t \le t_{max}$ [36, 37, 39-41, 44-48]:

$$t_m = (m-1)\Delta t; m = 1,2,\dots M \tag{3.8}$$

де M — кількість часових кроків;

Δt — розмір часового кроку.

$$\Delta t = \frac{t_{max}}{M-1} \tag{3.9}$$

Область вирішення зображена на рис. 3.9. Суцільні квадрати - відомі початкові значення. Пусті квадрати – відомі граничні значення. Кружечки вказують на положення внутрішніх точок, де обчислюється апроксимація кінцевої різниці [36, 37].

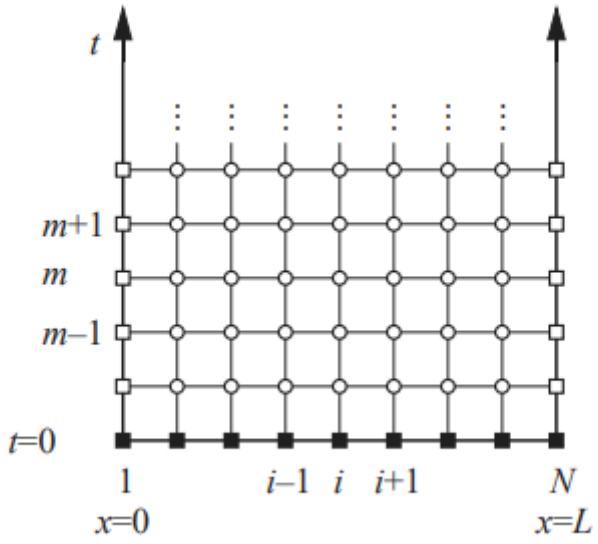

Рисунок 3.9 – Сітка на напівнескінченній смузі, що використовується для вирішення одновимірного теплового рівняння [37]

Метод кінцевих різниць передбачає використання дискретних наближень, таких як [36, 37, 39-41, 44-48]:

$$\frac{\partial u}{\partial x} \approx \frac{u_{i+1} - u_i}{\Delta x} \tag{3.10}$$

У цьому рівнянні, величини в правій частині визначені на кінцево-різницевій сітці. Наближення керівного диференційного рівняння отримують шляхом заміни всіх безперервних похідних дискретними формулами, такими як у рівнянні.

Розглянемо розкладання *u(x)* у ряд Тейлора навколо точки x_i [36, 37, 39-41, 44-48]:

$$u(x_i + \delta x) = u(x_i) + \delta x \frac{\partial u}{\partial x}\Big|_{x_i} + \frac{\delta x^2}{2}\frac{\partial^2 u}{\partial x^2}\Big|_{x_i} + \frac{\delta x^3}{3!}\frac{\partial^3 u}{\partial x^3}\Big|_{x_i} + \cdots \tag{3.11}$$

де:

δx – зміна x відносно x_i.

Нехай $\delta x = \Delta x$, тобто, для значення u у місці розташування лінії сітки x_{i+1}:

$$u(x_i + \Delta x) = u(x_i) + \Delta x \frac{\partial u}{\partial x}\Big|_{x_i} + \frac{\Delta x^2}{2}\frac{\partial^2 u}{\partial x^2}\Big|_{x_i} + \frac{\Delta x^3}{3!}\frac{\partial^3 u}{\partial x^3}\Big|_{x_i} + \cdots \tag{3.12}$$

Для $(\partial u/\partial x)_{xi}$:

$$\frac{\partial u}{\partial x}\Big|_{x_i} = \frac{u(x_i + \Delta x) - u(x_i)}{\Delta x} - \frac{\Delta x}{2}\frac{\partial^2 u}{\partial x^2}\Big|_{x_i} - \frac{\Delta x^2}{3!}\frac{\partial^3 u}{\partial x^3}\Big|_{x_i} + \cdots \tag{3.13}$$

Змінюючи наближене рішення на точне, тобто $u_i \approx u(x_i)$ і $u_{i+1} \approx u(x_i + \Delta x)$, отримується [36, 37, 39-41, 44-48]:

$$\frac{\partial u}{\partial x}\Big|_{x_i} \approx \frac{u_{i+1} - u_i}{\Delta x} - \frac{\Delta x}{2}\frac{\partial^2 u}{\partial x^2}\Big|_{x_i} - \frac{\Delta x^2}{3!}\frac{\partial^3 u}{\partial x^3}\Big|_{x_i} + \cdots \tag{3.14}$$

Теорема про середнє значення може бути використана для заміни похідних вищого порядку [36, 37, 39-41, 44-48]:

$$\frac{\Delta x^2}{2}\frac{\partial^2 u}{\partial x^2}\bigg|_{x_i} + \frac{\Delta x^3}{3!}\frac{\partial^3 u}{\partial x^3}\bigg|_{x_i} + \cdots = \frac{\Delta x^2}{2}\frac{\partial^2 u}{\partial x^2}\bigg|_{\xi} \tag{3.15}$$

де $x_i \leq \xi \leq x_{i+1}$

Звідси:

$$\begin{aligned} \frac{\partial u}{\partial x}\bigg|_{x_i} &\approx \frac{u_{i+1} - u_i}{\Delta x} + \frac{\Delta x^2}{2}\frac{\partial^2 u}{\partial x^2}\bigg|_{\xi} \\ \frac{\partial u}{\partial x}\bigg|_{x_i} - \frac{u_{i+1} - u_i}{\Delta x} &\approx \frac{\Delta x^2}{2}\frac{\partial^2 u}{\partial x^2}\bigg|_{\xi} \end{aligned} \tag{3.16}$$

Права частина рівняння (3.16) називається похибкою скорочення апроксимації кінцевої різниці. Це помилка, яка є результатом скорочення в рівнянні (3.15).

Оскільки, ξ та $u(x, t)$ невідомі, $\partial^2 u/\partial x^2$ неможливо обчислити. Незважаючи на те, що точну величину похибки скорочення визначити неможливо, для вираження залежності помилки відсікання від інтервалу сітки може бути використаний параметр сітки Δx, що обирається дослідником. Помилка скорочення записується наступним чином [36, 37, 39-41, 44-48]:

$$\frac{\Delta x^2}{2}\frac{\partial^2 u}{\partial x^2}\bigg|_{\xi} = O(\Delta x^2) \tag{3.17}$$

Цей вираз показує, наскільки швидко цей вираз наближається до нуля, коли Δx зменшується.

З урахуванням цього, (3.16) можна записати як:

$$\left.\frac{\partial u}{\partial x}\right|_{x_i} = \frac{u_{i+1} - u_i}{\Delta x} + O(\Delta x) \tag{3.18}$$

Це рівняння називається формулою прямої диференціації для $(\partial u/\partial x)_{xi}$. Апроксимація прямої диференціації має похибку скорочення, яка становить $O(\Delta x)$. Розмір похибки може контролювати дослідник, оскільки вона залежить від розміру сітки Δx. Частина похибки скорочення, яку неможливо контролювати, становить $|\partial\varphi/\partial x|_{\xi}$.

Альтернативну формулу кінцевої різниці першого порядку можна отримати, якщо ряд Тейлора, як у рівнянні (3.11), записати як $\delta x = -\Delta x$. Використовуючи дискретні змінні сітки замість усіх невідомих, отримується [36, 37, 39-41, 44-48]:

$$u_{i-1} = u_i - \Delta x \left.\frac{\partial u}{\partial x}\right|_{x_i} + \frac{\Delta x^2}{2}\left.\frac{\partial^2 u}{\partial x^2}\right|_{x_i} - \frac{\Delta x^3}{3!}\left.\frac{\partial^3 u}{\partial x^3}\right|_{x_i} + \cdots \tag{3.19}$$

Для $(\partial u/\partial x)_{xi}$:

$$\left.\frac{\partial u}{\partial x}\right|_{x_i} = \frac{u_{i+1} - u_i}{\Delta x} + \frac{\Delta x}{2}\left.\frac{\partial^2 u}{\partial x^2}\right|_{x_i} - \frac{\Delta x^2}{3!}\left.\frac{\partial^3 u}{\partial x^3}\right|_{x_i} + \cdots \tag{3.20}$$

Або, використовуючи O:

$$\left.\frac{\partial u}{\partial x}\right|_{x_i} = \frac{u_i - u_{i-1}}{\Delta x} + O(\Delta x) \tag{3.21}$$

Це формула зворотної диференціації, оскільки вона включає значення u при x_i і x_{i-1}.

Ряд Тейлора для φ_{i+1} та φ_{i-1} [36, 37, 39-41, 44-48]:

$$u_{i+1} = u_i + \Delta x \frac{\partial u}{\partial x}\bigg|_{x_i} + \frac{\Delta x^2}{2}\frac{\partial^2 u}{\partial x^2}\bigg|_{x_i} + \frac{\Delta x^3}{3!}\frac{\partial^3 u}{\partial x^3}\bigg|_{x_i} + \cdots \quad (3.22)$$

$$u_{i-1} = u_i - \Delta x \frac{\partial u}{\partial x}\bigg|_{x_i} + \frac{\Delta x^2}{2}\frac{\partial^2 u}{\partial x^2}\bigg|_{x_i} - \frac{\Delta x^3}{3!}\frac{\partial^3 u}{\partial x^3}\bigg|_{x_i} + \cdots \quad (3.23)$$

При відніманні рівняння (3.23) від рівняння (3.22) отримується:

$$u_{i+1} - u_{i-1} = 2\Delta x \frac{\partial u}{\partial x}\bigg|_{x_i} + 2\frac{\Delta x^3}{3!}\frac{\partial^3 u}{\partial x^3}\bigg|_{x_i} + \cdots \quad (3.24)$$

Для $(\partial u/\partial x)_{xi}$:

$$\begin{aligned} \frac{\partial u}{\partial x}\bigg|_{x_i} &= \frac{u_{i+1} - u_{i-1}}{2\Delta x} - \frac{\Delta x^2}{3!}\frac{\partial^3 u}{\partial x^3}\bigg|_{x_i} + \cdots \\ \frac{\partial u}{\partial x}\bigg|_{x_i} &= \frac{u_{i+1} - u_{i-1}}{2\Delta x} + O(\Delta x) \end{aligned} \quad (3.25)$$

Це рівняння апроксимації центральної диференціації.

Скінченно-різницеві наближення для похідних вищого порядку можна отримати за допомогою додаткових маніпуляцій із розкладанням у ряд Тейлора щодо $u(x_i)$. Із додавання рівняння (3.22) до (3.23), отримується [36, 37, 39-41, 44-48]:

$$u_{i+1} + u_{i-1} = 2u_i + \Delta x^2 \frac{\partial^2 u}{\partial x^2}\bigg|_{x_i} + \frac{2\Delta x^4}{4!}\frac{\partial^4 u}{\partial x^4}\bigg|_{x_i} + \cdots \quad (3.26)$$

Для $(\partial^2 u/\partial x^2)_{xi}$:

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{x_i} = \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2} + \left.\frac{\Delta x^2}{12}\frac{\partial^4 u}{\partial x^4}\right|_{x_i} + \cdots$$
$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{x_i} = \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2} + O(\Delta x^2) \quad (3.27)$$

Це рівняння апроксимації центральної диференціації другого порядку.

Існують три основні схеми розв'язку для методу кінцевих різниць: пряма схема Ейлера, з прямим диференціюванням по часу і центральним по простору (зліва на рис. 3.10); зворотна схема Ейлера, зі зворотнім диференціюванням по часу і центральним по простору (центральна на рис. 3.10); схема Кренка-Ніколсона (праворуч на рис. 3.10) [36, 37, 39-41, 44-48].

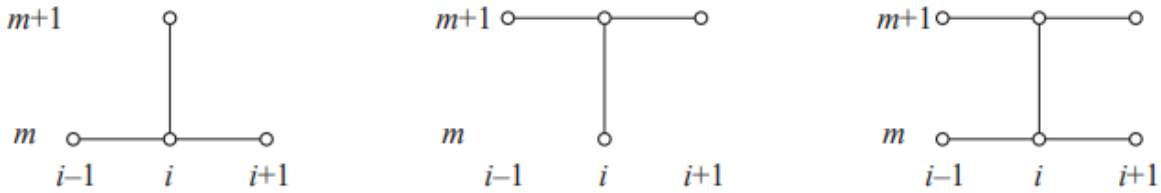

Рисунок 3.10 – Схеми розв'язку для методу кінцевих різниць [37]

Для того, щоб застосувати чисельний метод та отримати однозначний розв'язок рівняння, необхідно задати початкові та граничні умови. Початковою умовою буде температурне поле, встановлене на полі розв'язку задачі, а граничними – відповідно температури границь цього поля.

Апроксимація рівняння (3.4) за допомогою прямої диференціації, отримується [36, 37, 39-41, 44-48]:

$$\left.\frac{\partial u}{\partial x}\right|_{x_i, t_{m+1}} = \frac{u_i^{m+1} - u_i^m}{\Delta t} + O(\Delta t) \quad (3.28)$$

Використовуючи апроксимацію центральної диференціації для $(\partial^2 u/\partial x^2)_{xi}$ у момент часу m [36, 37, 39-46]:

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{x_i} = \frac{u_{i-1}^m - 2u_i^m + u_{i+1}^m}{\Delta x^2} + O(\Delta x^2) \tag{3.29}$$

При підстановці (3.28) у ліву частину рівняння (3.4), а (3.29) у праву, отримується:

$$\frac{u_i^{m+1} - u_i^m}{\Delta t} = \alpha \frac{u_{i-1}^m - 2u_i^m + u_{i+1}^m}{\Delta x^2} + O(\Delta t) + O(\Delta x^2) \tag{3.30}$$

Часові та просторові помилки мають різний порядок. Розв'язок (3.30) для u_i^{m+1} за умови скорочення помилки має наступний вигляд:

$$u_i^{m+1} = u_i^m + \frac{\alpha \Delta t}{\Delta x^2}(u_{i+1}^m - 2u_i^m + u_{i-1}^m) \tag{3.31}$$

Рівняння (3.31) є рівнянням з прямою за часом та центральною за простором апроксимацією.

Цю схему не складно реалізувати, оскільки значення u_{i+1}^m можна оновлювати незалежно одне від одного. Усе рішення міститься в двох циклах: зовнішньому циклі, що проходить усі часові кроки, і внутрішньому циклі, що проходить усі внутрішні вузли.

Усі розв'язки рівняння (3.4) з урахуванням початкових і граничних умов у рівнянні є обмеженими, спадаючими функціями. Іншими словами, величина розв'язку спадає від початкової умови до постійної. Така схема може давати нестабільні рішення, які коливаються та зростають, якщо Δt має занадто велике значення. Стабільність схеми прямого Ейлера виконується у випадку [36, 37, 39-41, 44-48]:

$$\frac{\alpha \Delta t}{\Delta x^2} < \frac{1}{2} \tag{3.32}$$

Якщо для виведення рівняння (3.31) використати апроксимацію зворотної диференціації, то [36, 37, 39-41, 44-48]:

$$\left.\frac{\partial u}{\partial x}\right|_{x_i,t_{m+1}} = \frac{u_i^m - u_i^{m-1}}{\Delta t} + O(\Delta t) \tag{3.33}$$

При підстановці (3.33) у ліву частину рівняння (3.4), а (3.29) у праву, отримується:

$$\frac{u_i^m - u_i^{m-1}}{\Delta t} = \alpha \frac{u_{i+1}^m - 2u_i^m + u_{i-1}^m}{\Delta x^2} + O(\Delta t) + O(\Delta x^2) \tag{3.34}$$

Помилки скорочення в цьому наближенні мають такий же порядок величини, як і в рівнянні (3.30), але, на відміну від нього, це рівняння не може бути змінено, щоб отримати просту алгебраїчну формулу для обчислення u_i^m через його сусідів u_{i+1}^m, u_{i-1}^m та u_i^{m-1}. Таким чином, (3.34) є одним з рівнянь у системі для значень *u* у внутрішніх вузлах просторової сітки (*i = 2, 3,...,N − 1*).

Інакше, (3.34) можна переписати без додавання похибок скорочення:

$$-\frac{\alpha}{\Delta x^2} u_{i-1}^m + \left(\frac{1}{\Delta t} + \frac{2\alpha}{\Delta x^2}\right) u_i^m - \frac{\alpha}{\Delta x^2} u_{i+1}^m = \frac{1}{\Delta t} u_i^{m-1} \tag{3.35}$$

Систему рівнянь можна представити в матричній формі у вигляді [36, 37, 39-46]:

$$\begin{bmatrix} b_1 & c_1 & 0 & & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & & 0 & 0 & 0 \\ 0 & a_3 & b_3 & & c_3 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & & \ddots & 0 \\ 0 & 0 & 0 & a_{n-1} & b_{n-1} & & c_{n-1} \\ 0 & 0 & 0 & 0 & a_n & & b_n \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{n-1} \\ u_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix} \tag{3.36}$$

де коефіцієнти внутрішніх вузлів:

$$\begin{aligned} a_i &= -\frac{\alpha}{\Delta x^2} \\ b_i &= \frac{1}{\Delta t} + \frac{2\alpha}{\Delta x^2} \\ c_i &= -\frac{\alpha}{\Delta x^2} \\ d_i &= \frac{1}{\Delta t} u_i^{m-1} \\ i &= 2,3,\dots,N-1 \end{aligned} \tag{3.37}$$

Реалізація даної схеми вимагає вирішення системи рівнянь на кожному часовому кроці. На додаток до цього, обчислювальне зусилля на часовий крок для даної схеми більше, ніж обчислювальне зусилля на часовий крок прямої схеми. Перевагою даної схеми перед прямою схемою Ейлера є її стабільність.

Пряма і зворотна схеми Ейлера мають часову помилку скорочення $O(\Delta t)$. Коли важливі рішення з точністю до часу, схема Кренка-Ніколсона має значні переваги. Ця схема має часову помилку скорочення, яка становить $O(\Delta t^2)$. Схема Кренка-Ніколсона є неявною, як і зворотна схема Ейлера, тому має чудову стабільність.

Ліва частина рівняння (3.4) апроксимується зворотною диференціацією, яка використовується в зворотній схемі Ейлера, права частина рівняння апроксимується центральною диференціацією, обчисленою на поточному та попередньому часових кроках [36, 37, 39-41, 44-48]:

$$\frac{u_i^m - u_i^{m-1}}{\Delta t} = \frac{\alpha}{2}\left(\frac{u_{i+1}^m - 2u_i^m + u_{i-1}^m}{\Delta x^2} + \frac{u_{i+1}^{m-1} - 2u_i^{m-1} + u_{i-1}^{m-1}}{\Delta x^2}\right) \tag{3.38}$$

Це рівняння використовується для прогнозування значень *u* в момент часу *m*, тому всі значення *u* в момент часу *m-1* вважаються відомими. Переносячи значення

u в момент часу *m* ліворуч, а значення *u* в момент часу *m-1* – праворуч, отримується [48, 49, 51-58]:

$$-\frac{\alpha}{2\Delta x^2}u_{i-1}^m+\left(\frac{1}{\Delta t}+\frac{\alpha}{\Delta x^2}\right)u_i^m-\frac{\alpha}{2\Delta x^2}u_{i+1}^m = \frac{\alpha}{2\Delta x^2}u_{i-1}^{m-1}+\left(\frac{1}{\Delta t}+\frac{\alpha}{\Delta x^2}\right)u_i^{m-1}+\frac{\alpha}{2\Delta x^2}u_{i+1}^{m-1} \quad (3.39)$$

Схема Кренка-Ніколсона є неявною, і, як наслідок, на кожному кроці часу необхідно розв'язувати систему рівнянь, тотожну до (3.36), з коефіцієнтами [36, 37, 39-41, 44-48]:

$$\begin{aligned} a_i &= -\frac{\alpha}{2\Delta x^2} \\ b_i &= \frac{1}{\Delta t}+\frac{\alpha}{\Delta x^2} \\ c_i &= -\frac{\alpha}{2\Delta x^2} \\ d_i &= \frac{1}{\Delta t}u_i^{m-1}-a_iu_{i-1}^{m-1}+(a_i+c_i)u_i^{m-1}-c_iu_{i+1}^{m-1} \\ i &= 2,3,\dots,N-1 \end{aligned} \quad (3.40)$$

Алгоритмічно, зворотна схема Ейлераі схема Кренка-Ніколсона дуже схожі. Головною відмінністю є те, що схема Кренка-Ніколсона має похибку скорочення $O(\Delta t^2)+O(\Delta x^2)$, тобто часова похибку значно менша, ніж у зворотній схемі Ейлера [36, 37, 39-41, 44-48].

3.5 Метод кінцевих об'ємів

Метод кінцевих об'ємів побудований на основі теореми дивергенції інтегрального числення. Для аналізу рівняння теплопередачі (3.1), простір розділяється на набір кубічних об'ємів. Довжини сторін для кожного об'єму є

еквівалентними $\Delta x = \Delta y = \Delta z$. Типова гексаедральна декартова комірка кінцевого об'єму – це куб у координатах *(i, j, k)*. Першим кроком у розв'язку рівнянь за допомогою методу кінцевих об'ємів є інтегрування (3.1) за об'ємом комірки [34, 35, 49-53]:

$$\int_{V_{i,j,k}} \frac{\partial u}{\partial t} dV = \int_{V_{i,j,k}} \nabla \nabla u dV \tag{3.41}$$

Оскільки об'єми комірок є фіксованими, вони не залежать від часу, часткову похідну за часом можна вивести з інтеграла в лівій частині рівняння. Оскільки просторову змінну було інтегровано з *u* ліворуч, похідна за часом перетворюється з часткової похідної на звичайну. У правій частині рівняння використовується теорема про розбіжність, інтеграл об'єму перетворюється на замкнутий інтеграл по межі комірки [34, 35, 49-53]:

$$\frac{d}{dt} \int_{V_{i,j,k}} u dV = \oint_{\partial V_{i,j,k}} \nabla u d\vec{a} \tag{3.42}$$

де $d\vec{a}$ – векторний елемент площі поверхні.

У цьому рівнянні $\partial V_{i,j,k}$ позначає граничну поверхню для об'єму комірки. Інтеграл у лівій частині рівняння є середнім *u*, обчисленим для об'єму комірки, тому середнє значення *u* представляється як $\bar{u}$, інтеграл об'єму приймає вигляд $\bar{u}$, помноженого на об'єм комірки, тобто [34, 35, 49-53]:

$$V_{i,j,k} \frac{d\bar{u}_{i,j,k}}{dt} = \oint_{\partial V_{i,j,k}} \nabla u d\vec{a} \tag{3.43}$$

Для роботи з одновимірним рівнянням теплопередачі (3.4) за допомогою цього метода, необхідно проінтегрувати його в просторі [34, 35, 47-51]:

$$\int_{x_i}^{x_{i+1}} \frac{\partial u}{\partial t} dx = \int_{x_i}^{x_{i+1}} \frac{\partial^2 u}{\partial x^2} dx \tag{3.44}$$

У лівій частині рівняння можна змінити місцями порядок інтеграції та диференціації, а праворуч відбувається антидиференціація [34, 35, 49-53]:

$$\frac{d}{dt} \int_{x_i}^{x_{i+1}} u dx = \frac{\partial u}{\partial x}\Big|_{x_i}^{x_{i+1}} \tag{3.45}$$

Інтеграл ліворуч створює одновимірний об'єм *u*, тоді як правий вираз обчислює середнє значення *u* в кінцевих точках клітинки кінцевого об'єму. Розглядаючи інтеграл ліворуч як середнє значення *u* для комірки [34, 35, 49-53]:

$$\Delta x \frac{\partial \bar{u}_i}{\partial t} = \frac{\partial \bar{u}_{i+1}}{\partial x} - \frac{\partial \bar{u}_i}{\partial x} \tag{3.46}$$

Використовуючи зворотну диференціацію по простору, отримується [34, 35, 49-53]:

$$\Delta x \frac{\partial \bar{u}_i}{\partial t} = \frac{\bar{u}_{i+1} - \bar{u}_i}{\Delta x} - \frac{\bar{u}_i - \bar{u}_{i-1}}{\Delta x} \tag{3.47}$$

Оцінюючи цей вираз на рівні часу *m* і застосовуючи пряму диференціацію до похідної за часом, отримується [34, 35, 49-53]:

$$\Delta x \frac{\bar{u}_i^{m+1} - \bar{u}_i^m}{\Delta t} = \frac{\bar{u}_{i+1}^m - \bar{u}_i^m}{\Delta x} - \frac{\bar{u}_i^m - \bar{u}_{i-1}^m}{\Delta x} \tag{3.48}$$

Після спрощення отримується рівняння кінцевої різниці об'ємів [34, 35, 49-53]:

$$\frac{\bar{u}_i^{m+1} - \bar{u}_i^m}{\Delta t} = \alpha \frac{\bar{u}_{i+1}^m - 2\bar{u}_i^m + \bar{u}_{i-1}^m}{\Delta x^2} \tag{3.49}$$

Це рівняння виглядає дуже схожим на (3.30), за винятком усереднення.

3.6 Метод кінцевих елементів

У розв’язках методу кінцевих елементів, область задачі розділяється на скінченну кількість елементів і отримуються наближені рішення поліноміального типу над кожним елементом. Завдання методу полягає в тому, щоб знайти лінійні наближені рішення над кожним елементом, що вимагає розрахунку невідомих значень у вузлах сітки. Для цього використовуються лінійні алгебраїчні рівняння.

Вираз (2.4) є сильною формою диференційного рівняння. Метод кінцевих елементів – це чисельний метод зваженого залишкового типу, що використовує слабку форму рівнянь для отримання розв’язків. Існують різні способи, за допомогою яких можна отримати слабку форму диференційного рівняння. Для задач теплопередачі, найкращим є метод зважених залишків.

Залишок диференційного рівняння отримується шляхом збору всіх членів на одній стороні рівняння. Залишок рівняння (2.4) [38, 54-58]:

$$R(x) = u\frac{dT}{dx} - \alpha\frac{d^2t}{dx^2} - f \tag{3.50}$$

За визначенням, точний розв’язок диференційного рівняння дорівнюватиме нульовій нев’язці у всіх точках задачі. Однак, нев'язка, в загальному випадку, не звернеться в нуль, якщо в неї підставити наближений розв'язок. Основним принципом методів зважених залишків є мінімізація залишку у зваженому інтегральному значенні, таким чином [38, 54-58]:

$$\int_{\Omega} w(x)R(x)dx = 0 \tag{3.51}$$

де w – обрані вагові функції.

Підставляючи (3.50) у (3.51), отримується [38, 54-58]:

$$\int_{\Omega} \left(wu\frac{dT}{dx} - w\alpha\frac{d^2t}{dx^2} - wf \right) dx = 0 \tag{3.52}$$

Це рівняння відоме як зважений залишковий вираз диференційного рівняння. Ідея полягає в тому, щоб обрати стільки різних вагових функцій, скільки необхідно для отримання необхідних лінійних алгебраїчних рівнянь. У теорії, коли кількість рівнянь прямує до нескінченності, інтеграл рівняння (3.52) прийме значення нулю для нескінченної кількості різних вибраних вагових функцій, що може бути вірним, лише якщо сама нев'язка дорівнює нулю в області задачі, тобто, коли кількість рівнянь прямує до нескінченності, наближене рішення наближається до точного рішення.

Для того, щоб мати можливість працювати з неперервними наближеними розв'язками, необхідно знизити вимоги до диференціювання невідомого у зваженому залишковому виразі. Це виконується шляхом застосування інтегрування по частинах до другого члена рівняння (3.52) [38, 54-58]:

$$\int_{\Omega} -w\alpha\frac{d^2t}{dx^2}dx = \int_{\Omega} \alpha\frac{dw}{dx}\frac{dT}{dx}dx - \int_{\Gamma} w\alpha\frac{dT}{dx}n_x d\Gamma \tag{3.53}$$

Як побічний продукт інтегрування частинами, виходить останній член рівняння, який називається граничним інтегралом. Він оцінюється на границях *(Γ)* області задачі *(Ω)*, де n_{ix} є *x* компонентом одиничної зовнішньої нормалі границі. Із наведеного вище рівняння, видно, що інтегрування за частинами знижує порядок

диференціювання невідомого з другого до першого і підвищує порядок диференціювання вагової функції до першого.

Далі, підставяючи рівняння (3.53) у рівняння (3.52), отримується [38, 54-58]:

$$\int_{\Omega}^{\square}\left(wu\frac{dT}{dx}+\alpha\frac{dw}{dx}\frac{dT}{dx}\right)dx=\int_{\Omega}^{\square}wfdx+\int_{\Gamma}^{\square}w\alpha\frac{dT}{dx}n_x d\Gamma \qquad (3.54)$$

Члени в лівій частині наведеного рівняння тепер включають лише похідні першого порядку. Це називається слабкою формою через нижчі вимоги до диференціювання порівняно з вихідним зваженим залишком. Підводячи підсумок, слабка форма дозволяє працювати з неперервними наближеними рішеннями.

3.7 Мережа термічного опору

Використання мережі термічного опору базується на понятті про термічний опір, описаному в 1.5, і не є числовим методом вирішення. Цей метод часто використовується в інженерних розрахунках, є досить зрозумілим і швидким для обчислення. Його суть полягає в розгляданні будівлі або будь-яких огороджувальних конструкцій як елементу опору теплопередачі.

Для твердої поверхні, оточеної повітрям, до теплопередачі випромінюванням додається ще теплопередача конвекцією. Термічний опір конвекції та випромінювання паралельні один одному (рис. 3.11). Сумарне значення теплопередачі, у такому випадку, визначається алгебраїчною сумою величин механізмів, що беруть у ній участь. Для простоти та зручності, це часто виконується шляхом визначення комбінованого коефіцієнта теплопередачі, який включає ефекти як конвекції, так і випромінювання [1-4]:

$$Q_{sum}=Q_{conv}+Q_{rad}, \text{Вт} \qquad (3.55)$$

$$Q_{sum}=h_{comb}(T_s-T_{out})S, \text{Вт} \qquad (3.56)$$

$$h_{comb} = \alpha + h_{rad}, \text{Вт}/(\text{м}^2\text{К}) \quad (3.57)$$

У випадку, якщо тверда поверхня оточена повітрям з обох сторін, утворюється мережа термічного опору (рис. 3.12). Якщо припустити, що $T_{out2}<T_{out1}$, зміна температури буде такою, як на рис. 3.12. Температура лінійно змінюється при проходженні стіни та асимптотично наближається до T_{out2} та T_{out1} у навколишньому середовищі, при віддаленні від стіни. У такій мережі, загальна величина теплопередачі розраховується за формулою [1-4]:

$$Q_{sum} = \frac{T_{out1} - T_{out2}}{R_{conv1} + R_{cond} + R_{conv2}} = \frac{T_{out1} - T_{out2}}{R_{sum}}, \text{Вт} \quad (3.58)$$

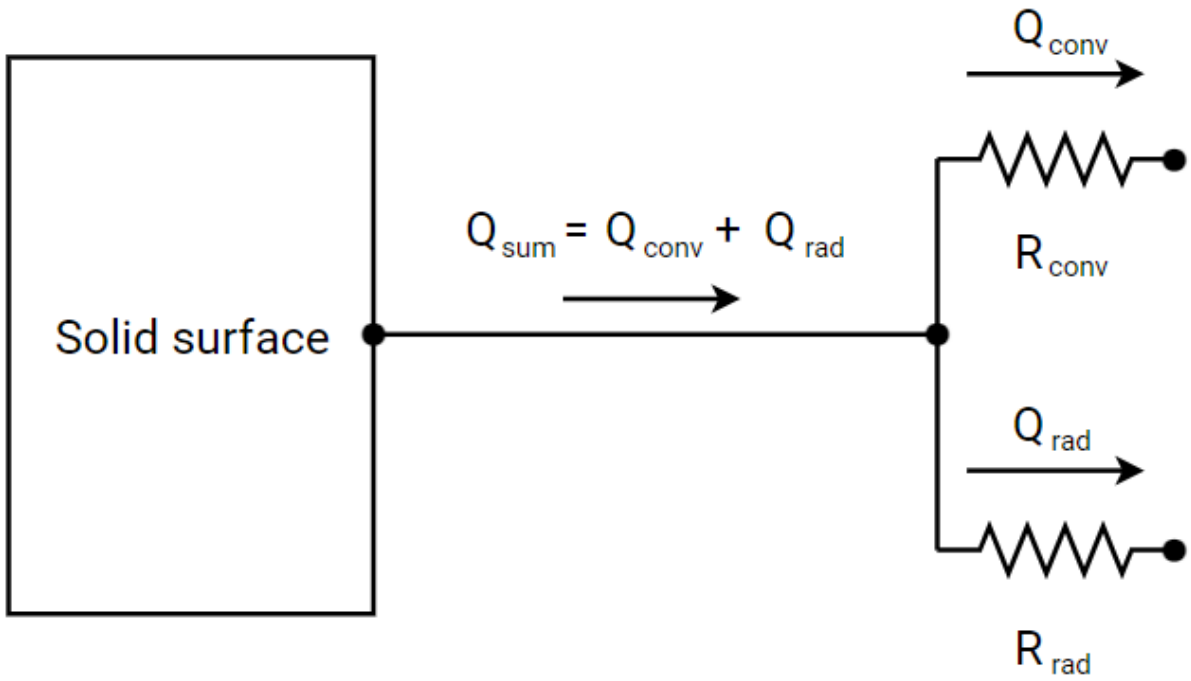

Рисунок 3.11 – Мережа термічного опору для межі тверда поверхня – повітря

Якщо ж, наприклад, стіна буде складатися з двох шарів або більше – можна створити відповідну мережу термічних опорів, просто додавши ще один R_{cond}. Використання мережі термічних опорів є досить потужним інструментом у

вирішенні проблем теплопередачі для систем, у яких відсутня генерації тепла, якими і є пасивні системи опалення.

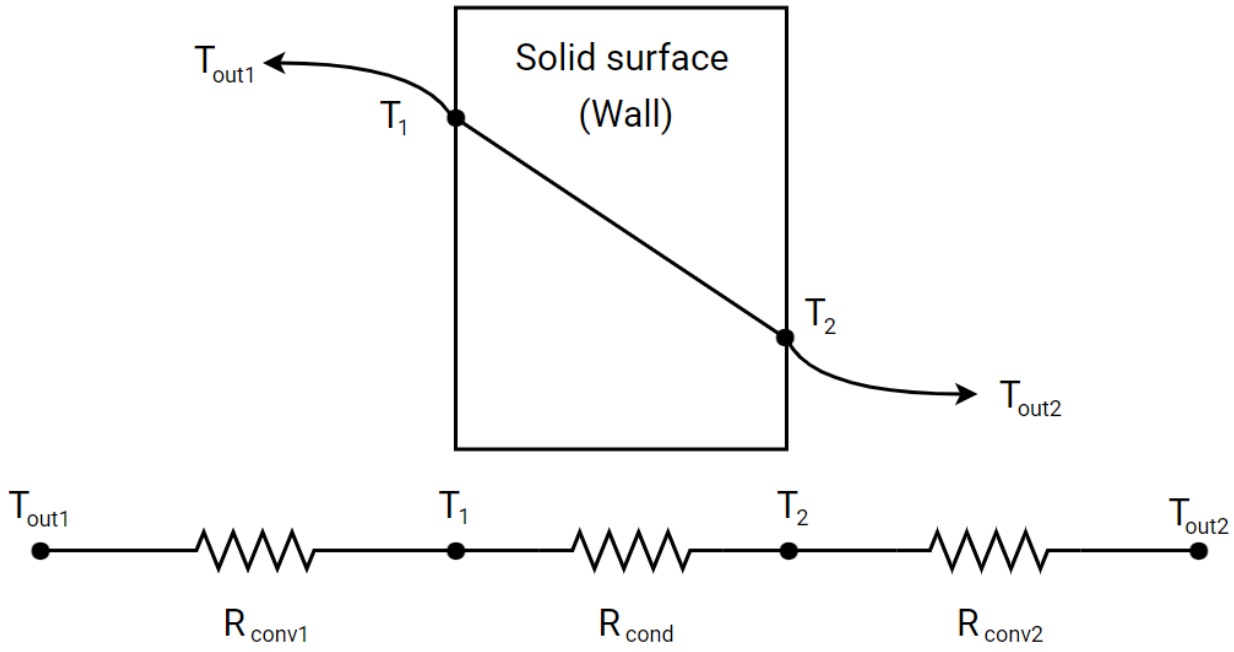

Рисунок 3.12 – Мережа термічного опору для стіни, оточеної повітрям з обох сторін

3.8 Створення методу розрахунку

Для дослідження теплових процесів всередині будівлі, особливо пасивних систем опалення, і розрахунку споживання енергії необхідні точні методи розрахунку. Проаналізувавши переваги та недоліки всіх методів, описаних вище, вирішено частково інтегрувати числові методи у мережі теплових опорів у новому розробленому методі для розрахунку теплових характеристик будівлі.

Ззовні, на поверхню огороджувальної конструкції діє пряме сонячне випромінювання і температура зовнішнього повітря (конвекція). Інтенсивність сонячного випромінювання, що надходить на зовнішні огороджувальні конструкції будівлі, розраховується за формулою [23, 59]:

$$I_t = I_b \cdot R_b + I_d \cdot \frac{1+\cos(\beta)}{2} + (I_b + I_d) \cdot \frac{1-\cos(\beta)}{2} \cdot a, \text{ МДж}/\text{м}^2, \quad (3.59)$$

де:

I_b і I_d – годинні значення прямої і розсіяної складових сонячного випромінювання на горизонтальну поверхню, (*МДж/м²*);

a – альбедо поверхні;

β – кут нахилу приймаючої поверхні до;

R_b – коефіцієнт перерахунку прямої сонячної радіації від горизонтальної поверхні до похилої.

Коефіцієнт перерахунку прямої сонячної радіації від горизонтальної поверхні до похилої розраховується як [23, 59]:

$$R_b = \frac{cos(\theta)}{cos(\delta) \cdot cos(\varphi) \cdot cos(\omega) + sin(\delta) \cdot sin(\varphi)}, \quad (3.60)$$

де:

θ – кут між напрямком потоку випромінювання на поверхню та нормаллю до неї;

δ – кут нахилу;

φ – кут широти;

ω – часовий кут.

У свою чергу [23, 59]:

$$cos(\theta) = (A - B)sin(\delta) + [C \cdot sin(\omega) + (D + E)cos(\omega)]cos(\delta), \quad (3.61)$$

де:

$A = cos(\beta) \cdot sin(\varphi)$;

$B = sin(\beta) \cdot cos(\varphi) \cdot cos(\alpha_{\text{п}})$;

$C = sin(\beta) \cdot sin(\alpha_{\text{п}})$;

$D = cos(\beta) \cdot cos(\varphi)$;

$E = sin(\beta) \cdot sin(\varphi) \cdot cos(\alpha_{\text{п}})$;

α_s – азимут поверхні.

Інтенсивність прямого сонячного випромінювання [23, 59]:

$$I_b = I_t - I_d, {}^{\text{МДж}}/_{\text{м}^2}, \quad (3.62)$$

де:

I_t – годинне значення сумарного сонячного випромінювання на горизонтальну поверхню, (*МДж/м²*).

Сумарна інтенсивність сонячного випромінювання [23, 59]:

$$I_t = r_t \cdot H_t, {}^{\text{МДж}}/_{\text{м}^2}, \quad (3.63)$$

де:

r_t – відношення годинної сумарної інтенсивності випромінювання до добової сумарної інтенсивності випромінювання;

H_t – сумарна добова інтенсивність випромінювання, (*MJ/m²*).

Інтенсивність розсіяного сонячного випромінювання [23, 59]:

$$I_d = r_d \cdot d, {}^{\text{МДж}}/_{\text{м}^2}, \quad (3.64)$$

де:

r_d – відношення годинної розсіяної інтенсивності випромінювання до добової розсіяної інтенсивності випромінювання;

H_d – інтенсивність добового розсіяного випромінювання, (*МДж/м²*).

У свою чергу [23, 59]:

$$r_t = \frac{\pi}{24}\left(a + b \cdot cos(\omega)\right)\frac{cos(\omega) - cos(\omega_s)}{sin(\omega_s) - \frac{\pi \cdot \omega_s}{180}cos(\omega_s)}, \quad (3.65)$$

де:

$a = 0{,}409 + 0{,}5016 \cdot sin(\omega_s - 60)$;

$b = 0{,}6609 - 0{,}4767 \cdot sin(\omega_s - 60)$;

ω_s – годинний кут заходу Сонця для горизонтальної поверхні.

У свою чергу [23, 59]:

$$r_d = \frac{\pi}{24}\frac{cos(\omega) - cos(\omega_s)}{sin(\omega_s) - \frac{\pi \cdot \omega_s}{180}cos(\omega_s)} \quad (3.66)$$

$$\omega_s = arccos\left(-tg(\varphi) \cdot tg(\delta)\right) \quad (3.67)$$

$$H_d = \frac{H_d}{H} \cdot H_t, {}^{MJ}/_{m^2}, \quad (3.68)$$

де:

H_d/H – відношення інтенсивність добового розсіяного випромінювання до сумарної інтенсивності випромінювання.

Відношення інтенсивність добового розсіяного випромінювання до сумарної інтенсивності випромінювання визначається як [23, 59]:

$$\frac{H_d}{H} = 1{,}311 - 3{,}022 \cdot K_t + 3{,}427 \cdot K_t^2 - 1{,}821 \cdot K_t^3 \quad (3.69)$$

де:

K_t – індекс хмарності.

Індекс хмарності розраховується як [23, 59]:

$$K_t = \frac{H_t}{H_0} \quad (3.70)$$

де:

H_0 – позаатмосферна інтенсивність випромінювання, (*МДж/м²)*.

Позаатмосферна інтенсивність випромінювання розраховується як [23, 59]:

$$H_0 = \frac{24}{\pi} \cdot G_0 \left[\left(1 + 0{,}33 \cdot cos\left(\frac{360 \cdot d}{365} \right) \right) \left(cos(\delta) \cdot cos(\varphi) \cdot sin(\omega_s) + \frac{2\pi \cdot \omega_s}{360} sin(\delta) \cdot sin(\varphi) \right) \right], {MJ}/{m^2}, \quad (3.71)$$

де:

G_0 – сонячна стала, 1367 (*Вт/м²*);

d – порядковий номер дня у році.

Перед початком розрахунків визначаються матеріали та геометрія будівлі (надаючи всі необхідні коефіцієнти для розрахунку компонентів теплопередачі). Потім розраховується годинне значення сонячної радіації, що досягає кожної поверхні будівлі, формуючи дві основні бази даних (зовнішня температура – щогодини протягом року та інтенсивність сонячної радіації на квадратний метр поверхні за певних кліматичних умов, також щогодини протягом року). Для підвищення точності розрахунків зовнішні огороджувальні конструкції поділяють на шари, рівні по товщині.

Значення інтенсивності сонячного випромінювання було отримано з (3.59), але в розрахунках також буде використовуватися формула (1.4), яка показує взаємодію двох поверхонь, які беруть участь в теплообміні.

Для реалізації методу розрахунку зручно буде розглянути схеми, аналогічні таким, що представлені у розділі (3.6), але представлені у вигляді мереж, де буде показаний напрямок переносу теплових потоків, виділених з рівнянь, що описують основні механізми теплопередачі. Суть методу полягає у розрахунку температури крок за кроком у просторовій координаті, де один крок відповідає одному шару

конструкції, та часовій координаті, де один крок дорівнює одній годині. Температура визначається з рівнянь механізмів теплопередачі, що приймають участь у переносі тепла для певного шару.

Схема теплопередачі для суцільної стіни (рис. 3.13), показує, що ззовні на стіну впливають конвекція (Q_{conv}) та випромінювання (Q_{rad}), у внутрішніх шарах енергія переноситься за рахунок теплопровідності (Q_{cond}), а теплова інерція конструкції враховується за рахунок теплоємності (Q_{cap}). Від стіни, до внутрішньої термальної маси будівлі, перенос енергії відбувається за рахунок конвекції та випромінювання, і знову, накопичення тепла враховується за рахунок теплоємності. Схема теплопередачі для даху або горища не буде відрізнятися від стіни.

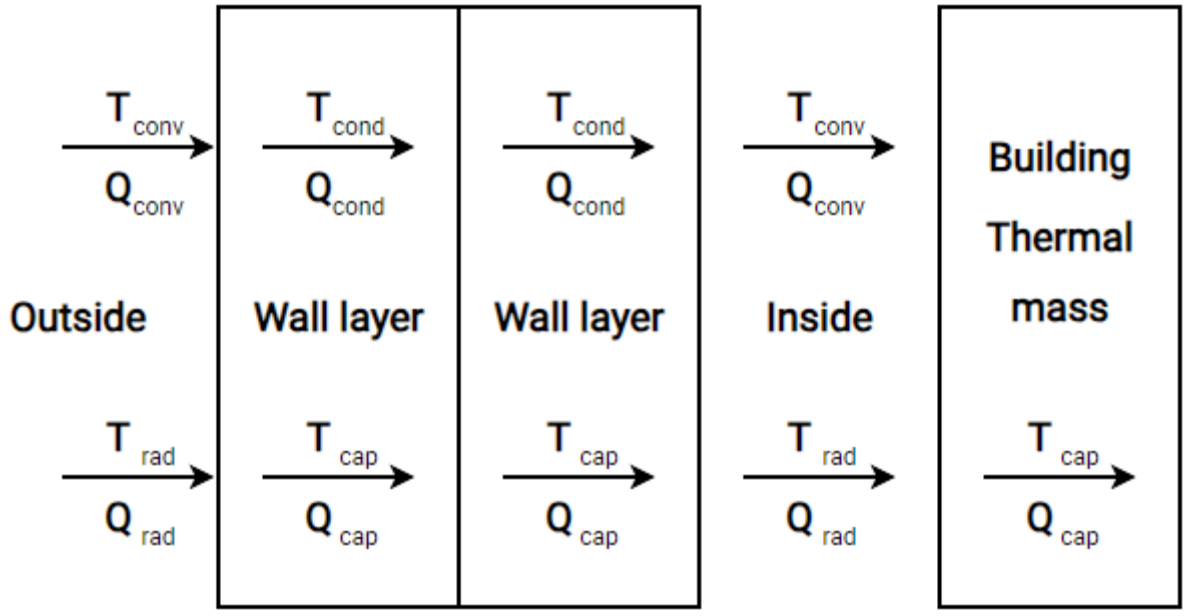

Рисунок 3.13 – Схема теплопередачі для суцільної стіни

Схема підлоги (рис. 3.14) відрізняється тим, що для неї відсутня конвекція та випромінювання ззовні, адже фундамент будівлі, найчастіше, немає контакту з середовищем, де можливе виникнення теплових потоків, а тільки із землею, тому теплопередача до зовнішнього шару підлоги відбувається за рахунок теплопровідності.

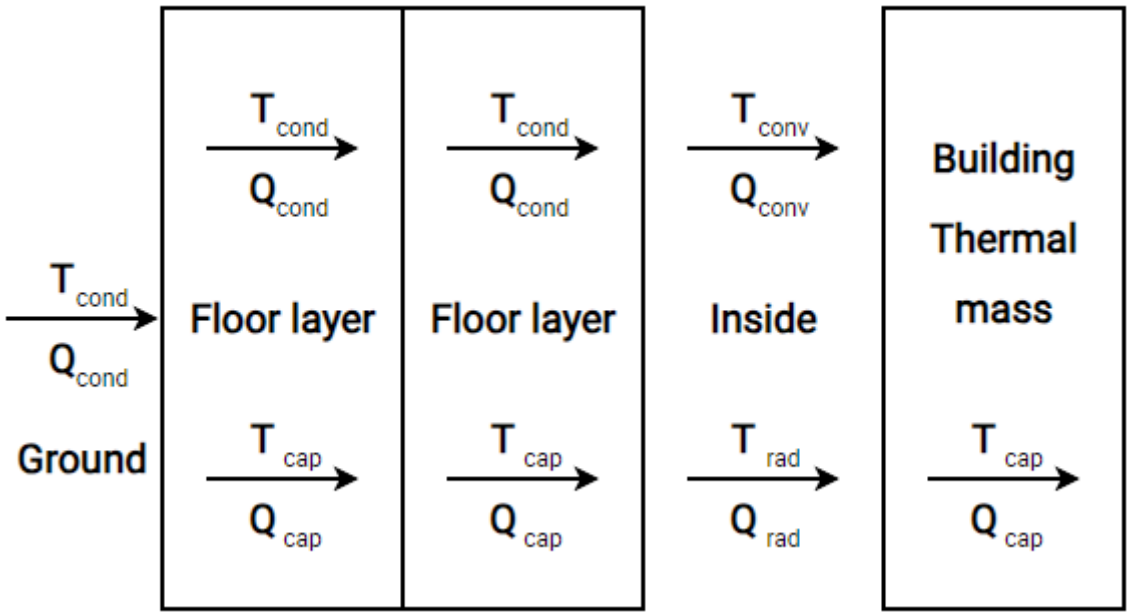

Рисунок 3.14 – Схема теплопередачі для підлоги

Температурні компоненти конвекції та випромінювання приведені в формулах (3.72) та (3.73):

$$T_{s,c,n} = T_{out,n} - \frac{Q_{conv,n,out}}{\alpha_{out} \cdot S}\ ,K, \tag{3.72}$$

де:

$T_{out,n}$ – випромінювальна температура оточуючого середовища в годину n, K.

$Q_{conv,n,out}$ – величина конвекції в годину n, $Вт$;

α – коефіцієнт теплопередачі конвекцією, $Вт/(м^2 \cdot К)$;

S – площа поверхні теплопередачі, $м^2$.

$$T_{rad,n}^4 = \frac{Q_{rad,n}}{\varepsilon \cdot \sigma_0 \cdot S} + T_{s,c,n}^4\ ,K, \tag{3.73}$$

де:

$T_{rad,n}$ – температура випромінювача у годину n;

$Q_{radv,n}$ – величина сонячного випромінювання в годину n, $Вт$;

σ_0 – стала Стефана-Больцмана, $Вт/(м^2 К^4)$.

Невідомою залишається температура в нульову годину, на якій базуються подальші обчислення, формула для її розрахунку враховує враховує випромінювання від стіни назовні в нічний час:

$$T_{s,c,0} = T_{out,0} \cdot \left(T_{out,0}^4 \cdot (\sigma_0 - \varepsilon\sigma_0)\right)^{\frac{1}{4}}, K, \tag{3.74}$$

де:

$T_{out,0}$ – температура оточуючого середовища в нульову годину, *K*.

Для розрахунку температури зовнішнього шару, необхідно скласти вищезазначені температурні компоненти, але, очевидно, що вплив кожної з величин в певний період часу має різний вплив на сумарну величину, тому вирішено використовувати вагові коефіцієнти, подібно, як для методу кінцевих елементів, виражені як відношення певної величини до сумарної потужності, що передається шару:

$$\begin{aligned} T_{n,out} = {} & T_{s,c,n}\left(\frac{\left|Q_{conv,n-1,out}\right|}{\left|Q_{conv,n,out}\right| + \left|Q_{rad,n}\right|}\right) \\ & + T_{rad,n}\left(\frac{\left|Q_{rad,n}\right|}{\left|Q_{conv,n,out}\right| + \left|Q_{rad,n}\right|}\right), K, \end{aligned} \tag{3.75}$$

Для розрахунку теплопередачі внутрішніх шарів, необхідно визначити температурні компоненти теплопровідності та теплоємності:

$$T_i = T_{i-1} - \frac{Q_{cond}\delta_i}{\lambda S}, K \tag{3.76}$$

де:

T_i – температура *i*-го шару, *K*;

Q_{cond} – величина теплопровідності в годину *n*, *Вт*;

δ_i – товщина шару, *м*.

$$T_i = T_{i-1} - \frac{Q_{cap}}{cm} \tag{3.77}$$

де:

c – коефіцієнт теплоємності матеріалу, *Дж/(кгК)*;

m – маса, *кг*.

Також, при розрахунку внутрішніх шарів, введено коефіцієнт зміни температури у просторовій координаті, запозичений з методу кінцевих різниць:

$$T_{c,i} = \frac{\frac{\lambda_i}{c_i \rho_i}}{2\left(\frac{\lambda_i}{c_i \rho_i} + 1\right)} \tag{3.78}$$

де:

ρ_i – густина матеріалу, з якого виконано шар, *кг/м*^{3}.

Урешті, температура внутрішніх шарів розраховується за формулою:

$$\begin{aligned} T_i = {} & \left(T_{n-1,i-1} - \frac{Q_{cond,n-1,i} \cdot \delta_i}{\lambda_i S}\right)\left(\frac{\left|Q_{cond,n-1,i}\right|}{2\left|Q_{cond,n-1,i}\right| + 2\left|Q_{cap,n-1,i}\right|}\right) \\ & + \left(T_{n,i-1} - \frac{Q_{cap,n-1}}{c_i m_i}\right)\left(\frac{\left|Q_{cap,n-1,i}\right|}{2\left|Q_{cond,n-1,i}\right| + 2\left|Q_{cap,n-1,i}\right|}\right) \\ & + \left(T_{n,i-1} - T_{c,i}\right)\left(\frac{\left|Q_{cond,n-1,i}\right|}{2\left|Q_{cond,n-1,i}\right| + 2\left|Q_{cap,n-1,i}\right|}\right) \\ & + \left(T_{n-1,i} - T_{c,n}\right)\left(\frac{\left|Q_{cap,n-1,i}\right|}{2\left|Q_{cond,n-1,i}\right| + 2\left|Q_{cap,n-1,i}\right|}\right), K \end{aligned} \tag{3.79}$$

Температура термальної маси будівлі, яку складають внутрішні стіни та об’єм повітря, розраховується окремо для кожної огороджувальної конструкції:

$$T_{n,in} = \left(T_{n,in-1} - \frac{Q_{conv,n-1,in}}{\alpha_{in} \cdot S}\right) \times$$
$$\times \left(\frac{|Q_{conv,n-1,in}|}{|Q_{conv,n-1,in}| + |Q_{cap,n-1,in}| + |Q_{rad,n-1,in}|}\right)$$
$$+ \left(T_{n,in-1} - \frac{Q_{cap,n-1,in}}{c_{in} m_{in}}\right)$$
$$\left(\frac{|Q_{cap,n-1,in}|}{|Q_{conv,n-1,in}| + |Q_{cap,n-1,in}| + |Q_{rad,n-1,in}|}\right)$$
$$+ \left(T^4_{n,in-1} - \frac{Q_{rad,n-1,in}}{\varepsilon \cdot \sigma_0 \cdot S}\right)^{\frac{1}{4}} \times$$
$$\times \left(\frac{|Q_{rad,n-1,in}|}{|Q_{conv,n-1,in}| + |Q_{cap,n-1,in}| + |Q_{rad,n-1,in}|}\right), K \quad (3.80)$$

Для схеми теплопередачі вікон (рис. 3.15) враховується особливість даної конструкції – її прозорість для випромінювальної енергії, що генерує Сонце, це показано стрілкою прямо до внутрішньої термальної маси будівлі крізь скло. Це відображається в формулі (3.81):

$$T_{n,in,w} = T_{n,in}\left(\frac{|Q_{conv,n,in}| + |Q_{cap,n,in}| + |Q_{rad,n,in}|}{|Q_{conv,n,in}| + |Q_{cap,n,in}| + |Q_{rad,n,in}| + |Q_{rad,n}|}\right)$$
$$+ T_{rad,n}\left(\frac{|Q_{rad,n}|}{|Q_{conv,n,in}| + |Q_{cap,n,in}| + |Q_{rad,n,in}| + |Q_{rad,n}|}\right), K \quad (3.81)$$

Відмінною особливістю в розрахунках стіни Trombe є те, що сонячне випромінювання, яке безпосередньо передається внутрішнім конструкціям для звичайних вікон, буде поглинатися масивною стіною. І вже звідти, через конвекцію, буде передаватися всередину будівлі.

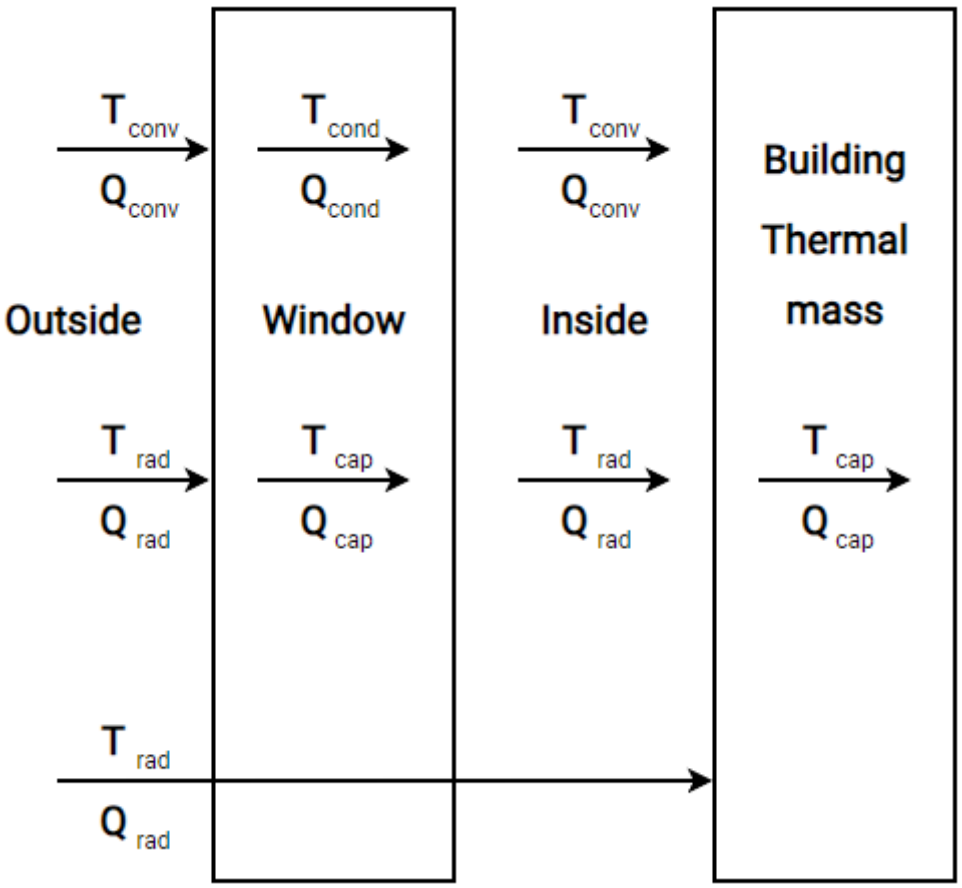

Рисунок 3.15 – Схема теплопередачі для вікна

Більшість шарів конструкції розраховуються за раніше продемонстрованими формулами. Температуру зовнішньої поверхні розраховують за формулами (3.72-3.75). Після цього розраховується температура шару скління за формулою (3.79). Температуру внутрішньої поверхні масивної стіни розраховують за формулою (3.80).

Для розрахунку температури на поверхні масивної стіни Тромбе використовується спеціальна формула, яка враховує всі процеси, що відбуваються всередині повітряного прошарку, це конвекція, конвективний теплообмін до поверхні стіни, теплообмін в повітря та його теплоємність:

$$
\begin{aligned}
&T_i \\
&= \left(T_{n-1,i-1}\right. \\
&\left. - \frac{Q_{cond,n-1,i} \cdot \delta_i}{\lambda_i S}\right)\left(\frac{\left|Q_{cond,n-1,i}\right|}{2\left|Q_{cond,n-1,i}\right| + 2\left|Q_{cap,n-1,i}\right| + \left|Q_{conv,n-1,i}\right|}\right) \\
&+ \left(T_{n,i-1} - \frac{Q_{cap,n-1}}{c_i m_i}\right)\left(\frac{\left|Q_{cap,n-1,i}\right|}{2\left|Q_{cond,n-1,i}\right| + 2\left|Q_{cap,n-1,i}\right| + \left|Q_{conv,n-1,i}\right|}\right) \\
&+ \left(T_{n,i-1} - T_{c,i}\right)\left(\frac{\left|Q_{cond,n-1,i}\right|}{2\left|Q_{cond,n-1,i}\right| + 2\left|Q_{cap,n-1,i}\right| + \left|Q_{conv,n-1,i}\right|}\right) \\
&+ \left(T_{n-1,i} - T_{c,n}\right)\left(\frac{\left|Q_{cap,n-1,i}\right|}{2\left|Q_{cond,n-1,i}\right| + 2\left|Q_{cap,n-1,i}\right| + \left|Q_{conv,n-1,i}\right|}\right) \\
&+ \left(T_{n,i-1} - \frac{Q_{conv,n-1,i}}{Nu \cdot S}\right), K
\end{aligned}
\tag{3.82}
$$

де:

Nu – коефіцієнт теплопередачі конвекцією всередині повітряного прошарку стіни Тромбе:

$$
Nu = 0{,}5 \cdot \left(\frac{g \cdot b \cdot (T_s - T_{env}) \cdot D^4}{8 \cdot v^2 \cdot h}\right)^{0.294} \tag{3.83}
$$

де:

g – прискорення вільного падіння, *м/с²*;

b – коефіцієнт об’ємного розширення середовища;

v – кінематична в'язкість середовища, *м²/с*;

T_s – температура поверхні стіни Тромбе, *K*;

T_{env} – температура оточуючого середовища, *K*;

D – товщина повітряного прошарку, *м*;

h – висота стіни Тромбе, *м*.

Після розрахунку термальної маси будівлі для кожної окремої огороджувальної конструкції, необхідно розрахувати загальну внутрішню

температуру. Знову ж таки, зрозуміло, що кожна така конструкція матиме відмінний вплив на загальний результат, тому знову вводиться ваговий коефіцієнт у вигляді відношення площі конкретної огороджувальної конструкції до загальної кондиціонованої площі:

$$T_{n,b} = \sum T_{n,in,x} \frac{S_x}{\sum S}, K \tag{3.84}$$

де:

S_x – площа конкретної огороджувальної конструкції, *м*^{2}.

Результатом розрахунку за формулою (4.24) буде матриця точок, що складається з двох стовпців: температура та час. Знаючи всі ці значення, можна перейти до розрахунку енергетичних потреб за методом градусо-секунд. Енергопотреба опалення [15]:

$$E_h = \frac{H_{tr}}{3600} \sum \left(T_h - T_{n,b}\right) \cdot (t_i - t_{i-1}), \text{Вт год}, \tag{3.85}$$

де:

T_h – встановлена температура опалення, *К*;

t_i – час, коли температура всередині будівлі має значення $T_{n,b}$, *с*;

H_{tr} – сумарний коефіцієнт теплопередачі теплопровідністю, *Вт/К*.

Сумарний коефіцієнт теплопередачі теплопровідністю розраховується як [15]:

$$H_{tr} = H_D + H_g, {}^{\text{Вт}}/_{\text{К}}, \tag{3.86}$$

де:

H_D – прямий узагальнений коефіцієнт теплопередачі теплопровідністю в зовнішнє середовище, *Вт/К*;

H_g – стаціонарний коефіцієнт теплопередачі теплопровідністю до ґрунту.

У свою чергу [15]:

$$H_x = \sum S_i \cdot U_i \, , {}^{\text{Вт}}/_{\text{К}}, \tag{3.87}$$

де:

S_i – площа і-го елементу будівлі, $м^2$;

U_i – приведений коефіцієнт теплопередачі і-го елемента огороджувальної конструкції будівлі, $Вт/(м^2 \cdot К)$.

Формула для розрахунку енергетичних потреб кондиціонування [15]:

$$E_c = \frac{H_{tr}}{3600} \sum \left(T_{n,b} - T_c\right) \cdot (t_i - t_{i-1}), Wh, \tag{3.88}$$

де:

T_c – встановлена температура охолодження, $К$.

Це основні формули для розрахунку. По суті, кожна формула формує певний блок теплопередачі, використовуючи який у правильній конфігурації, можна провести розрахунок енергетичних потреб для будь-якої будівлі. Використовуючи тільки їх, можна розрахувати внутрішню температуру стін, вікон, даху та підлоги (зміняться лише коефіцієнти), а потім погодинно розрахувати температуру внутрішньої маси будівлі, знаючи яку, можна розрахувати енергію. споживання та аналіз кривих зміни температури.

3.9 Висновки до третього розділу

У третьому розділі розглянуті методи дослідження процесів теплопередачі, як такі, що активно використовують у комп’ютерних системах, такі як, метод кінцевих елементів, кінцевих різниць, кінцевих об’ємів, так і інженерні, а саме, мережу термічних опорів.

Загальний підхід числових методів є наступним. По-перше, відбувається розділення області дослідження на дискретні елементи. В методі кінцевих різниць це робиться за допомогою різницевих сіток, в методі кінцевих елементів – за допомогою кінцевих елементів, а в методі кінцевих об’ємів – за допомогою об'ємів чи комірок. Основний поділ між типами сіток є на структуровані та неструктуровані. Структурована сітка – це сітка, у якій усі внутрішні вершини топологічно однакові. Така сітка є найпростішою для створення. Неструктурована сітка – це сітка, у якій вершини можуть мати довільні локальні околиці. Блоково-структурована або гібридна сітка утворена кількома невеликими структурованими сітками, об’єднаними в загальний неструктурований візерунок. Загалом, структуровані сітки пропонують простоту та легкий доступ до даних, тоді як неструктуровані сітки пропонують зручнішу адаптацію сітки (уточнення/зменшення визначень на основі початкового рішення) і краще підходять для складних доменів. Високоякісні гібридні сітки користуються перевагами обох підходів, але гібридна сітка не є повністю автоматичною.

Далі, відбувається апроксимація для диференціальних рівнянь. Методи кінцевих різниць та кінцевих об’ємів використовують різницеві або інтегральні апроксимації, тоді як метод кінцевих елементів використовує функції форми для апроксимації розподілу поліномів на кожному елементі.

Після цього, виникає система алгебраїчних рівнянь, яку необхідно розв'язати для отримання чисельного розв'язку.

Ці методи є потужними інструментами для моделювання та числового розв'язання різних фізичних систем, зокрема, для визначення температурних характеристик різних об’єктів. Перевага методів полягає в тому, що вони дають можливість глибоко досліджувати процеси всередині системи та отримувати точні значення для будь-якого відрізку часу. Але, головним чином, через свою ресурсоємність, ці методи не дуже підходять для вивчення великих об'єктів, таких як будівлі. Їх використання для цієї мети вимагає застосування обчислювальних серверів і займає багато часу.

Суть використання мережі теплових опорів полягає в розгляданні будь-яких огороджувальних конструкцій як елементу опору теплопередачі, за допомогою яких можливо сформувати таку структуру, яка є необхідною для розрахунку. Основою цього методу є використання математичних формул для процесів теплопередачі, описаних у першому розділі.

Перевагою цього методу є висока швидкість розрахунку та відсутність необхідності використання великих обчислювальних потужностей. Головним недоліком є те, що, найчастіше, для розрахунків використовуються середньомісячні значення основних показників, що не дає заглибитися у характеристики системи на певному, короткому відрізку часу і побачити переваги й недоліки системи при певних умовах, якими можуть стати зовнішня температура, інтенсивність сонячного випромінювання тощо.

Проаналізувавши переваги та недоліки всіх методів, вирішено частково інтегрувати числові методи у мережі теплових опорів у новому розробленому методі для розрахунку теплових характеристик будівлі.

Продемонстровано створення нового методу для розрахунку теплових та енергетичних характеристик будівлі з пасивною системою опалення. Даний метод є поєднанням числових та інженерних методів розрахунку, що дозволить моделі, побудованій на його основі, мати високу швидкість та точність виконання обчислень, а також, отримати можливість дослідження системи на певному, короткому відрізку часу і побачити переваги й недоліки системи при заданих вхідних умовах.

РОЗДІЛ 4
РОЗРОБКА КОМП'ЮТЕРНОЇ СИСТЕМИ

4.1 Алгоритм роботи та функції комп'ютерної системи

Для моделювання характеристик всіх наведених у другому розділі блоків структурних схем пропонується розробити комп'ютерну систему. Функції, що виконуватиме система, будуть наступними:

- Розрахунок енерговитрат будівлі з пасивною системою опалення з заданими користувачем характеристиками.
- Оптимізація параметрів пасивної системи опалення за енерговитратами.

Алгоритм роботи такої системи наступний:

1. Обирається місцевість, для якої проводитимуться розрахунки.

2. Обираються геометричні характеристики будівлі, якими є довжина, ширина та висота.

3. Обираються матеріали, з яких будуть виконані огороджувальні конструкції будівлі.

4. Обираються матеріали та розміри теплоізоляційного шару, адже погано втеплена будівля не зможе мати високу енергоефективність.

5. Обирається кількість та розташування віконних прорізів, а також тип склопакетів.

6. Система проводить розрахунок енерговитрат будівлі на опалення та кондиціонування, що є основною ціллю моделювання.

7. Система проводить оптимізацію пасивної системи опалення за енерговитратами.

У результаті, дослідник має повне уявлення про тепловий режим майбутньої будівлі, отримуючи всі необхідні значення та графіки.

4.2 Схеми оптимізації

Для можливості отримання характеристик будівлі, що сприятимуть мінімальним витратам тепла, необхідно оптимізувати результати моделювання. Запропоновану схему оптимізації параметрів пасивної системи опалення закритого типу зображена на рис. 4.1. Алгоритм її роботи наступний:

1. Задаються вхідні дані, що не будуть змінюватися впродовж дослідження: кліматичні дані; матеріал, з якого побудована акумулююча стіна. Матеріал повинен мати відносно високу густину та теплопровідність для того, щоб виконувати функцію акумулятора (найбільш розповсюджений матеріал для будівництва акумулюючих стін – залізобетон).

2. Задаються діапазони вхідних параметрів системи опалення: кількість шарів скління; товщина повітряного прошарку між склінням і акумлюючою стіною; товщина акумулюючої стіни. Вибір діапазонів значень ґрунтується на наступному: кліматичних умовах, для яких буде проводитися дослідження (чим холодніший клімат – тим більше шарів скління та товще акумулююча стіна); архітектурних особливостях будівлі (акумулююча стіна має межу мінімальної товщину через те, що крім опалювальної функції ще й несе навантаження); особливостях розпорядку дня людей, що будуть мешкати в оселі (чим товще акумулююча стіна, тим більшою буде затримка у передачі тепла від неї до кімнати).

3. Оптимізація виконується поступово для кожного з параметрів. Наприклад, для того, щоб отримати оптимальну товщину повітряного прошарку, необхідно прийняти значення інших параметрів за константу, і поступово виконати розрахунки для всіх значень товщини повітряного прошарку з заданого діапазону.

4. Пасивна система опалення з обраними параметрами моделюється.

5. Проводиться симуляція роботи пасивної системи опалення впродовж року. На цьому етапі отримуються графік розподілу температур всередині будівлі впродовж року.

6. Графік розподілу температур аналізується та розраховуються енерговитрати на опалення та кондиціонування, а також, сумарні енерговитрати.

7. Будується залежність сумарних річних енерговитрат від кожного значення обраного діапазону параметрів. Ця залежність перевіряється на наявність екстремумів, значення параметру у точці екстремуму і буде оптимальним.

8. У разі знаходження екстремуму, дослідження обраного параметру припиняється, оптимальне значення зберігається та використовується для дослідження інших параметрів.

9. Починається дослідження наступного параметру.

Комп'ютерна система, що реалізує використання запропонованої концептуальної схеми оптимізації параметрів пасивної системи опалення закритого типу дозволить прийняти математично обґрунтовані рішення щодо покращення енергоефективності будівель.

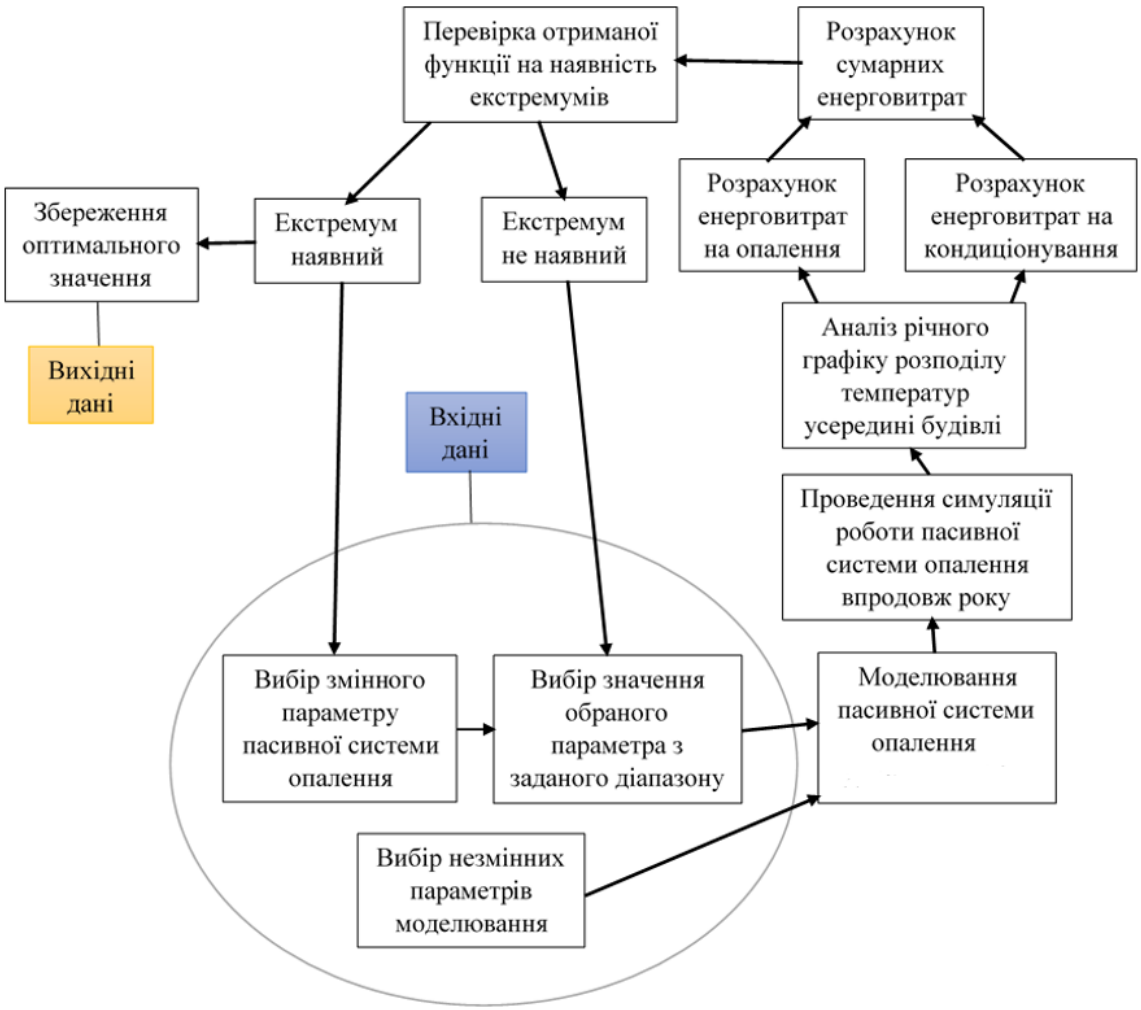

Рисунок 4.1 – Схема оптимізації параметрів пасивної системи опалення закритого типу

4.3 Реалізація комп’ютерної моделі

Вхідні дані, такі як, значення температури навколишнього середовища, температура ґрунту або потужності сонячного випромінювання, що показано на прикладі сонячного випромінювання для північної сторони, розраховуються заздалегідь у Excel і можуть використовуватися за допомогою пакету «pandas».

Імпортування з Excel та об’явлення температури навколишнього середовища:

```
TemperaturePerHour_data =
r"D:\Postgraduate\Postgraduate\Disertation\FinalVersion\TemperaturePerHour.
xlsx"
TemperaturePerHour = pd.read_excel(TemperaturePerHour_data)
T_street = TemperaturePerHour.iloc[:, 1]
```

Імпортування з Excel та об’явлення температури ґрунту:

```
SoilTemperaturePerHour_data =
r"D:\Postgraduate\Postgraduate\Disertation\FinalVersion\SoilTemperaturePerHour.xls
x"
SoilTemperaturePerHour = pd.read_excel(SoilTemperaturePerHour_data)
T_soil = SoilTemperaturePerHour.iloc[:, 1]
```

Імпортування з Excel та об’явлення даних про потужність сонячного випромінювання:

```
SolarIrradiationNorth_data =
r"D:\Postgraduate\Postgraduate\Disertation\FinalVersion\SolarIrradiation\SolarIrra
diationNorth.xlsx"
SolarIrradiationNorth = pd.read_excel(SolarIrradiationNorth_data)
I_n = SolarIrradiationNorth.iloc[:, 1]
```

Дані про сонячне випромінювання для інших сторін світу імпортуються таким же чином, і записуються як:

- SolarIrradiationEast_data – для сходу.
- SolarIrradiationWest_data – для заходу.
- SolarIrradiationSouth_data – для півдня.
- SolarIrradiationRoof_data – для даху.

Основні функції, створені для реалізації методу:

- Набір функцій, що описують розрахунок температурних характеристик стін, вікон, даху, підлоги, на виході повертають значення правої половини рівняння (3.82) для кожної окремої огороджувальної конструкції.
- Функція для розрахунку температурних характеристик стіни Тромбе виділений окремо, через особливості використання, продемонстровані далі.
- Функція для розрахунку енергетичних потреб будівлі за градусо-секундами, повертає у якості результату значення енергетичних потреб опалення (3.83) та кондиціонування (3.86).
- Функція оптимізації, принцип дії якої описано у розділі (4.2), що повертає, як результат, оптимізовані значення параметрів стіни Тромбе: товщина скла – зовнішнього шару стіни, товщина повітряного прошарку між склом та масивною стіною, товщина масивної стіни. Також, результатом є значення енергетичних потреб при оптимізованій пасивній системі опалення.

Початковим етапом реалізації функцій для розрахунку теплоенергетичних характеристик огороджувальних конструкцій є її об'явлення та визначення аргументів.

Наприклад, для північної стіни об'явлення виглядатиме наступним чином:

```
def north_wall_temp_calc(B_ins_w, B_l_w):
```

Аргументи функції наступні:

- Товщина шару ізоляції (B_ins_w).
- Товщина одного шару стіни (B_l_w).

Кожна стіна складається з 5 шарів.

Для розрахунку теплоенергетичнихвікон характеристик вікон, об'явлення функції виглядає наступним чином:

```
def north_windows_temp_calc(x):
```

Аргументом є кількість шарів скління (x).

Для підлоги по ґрунту, об'явлення функції виглядає наступним чином:

```
def floor_temp_calc(B_ins_f, B_l_f):
```

Для горища, об'явлення функції виглядає наступним чином:.

```
def floor_temp_calc(B_ins_r, B_l_r):
```

Аргументи цих двох функцій такі ж, як і для функції розрахунку стін.

Далі відбувається об’явлення всіх необхідних змінних, що будуть використані безпосередньо всередині функції. Початково, визначаються загальні характеристики огороджувальної конструкції. Для поверхні стін, і горищ – це:

- Значення потужності сонячного випромінювання (I_rad).
- Площа поверхні огороджувальної конструкції (S_w), що визначається як різниця площі певної сторони будинку, у даному випадку – це північна сторона і площа визначається як добуток довжини будівлі (l) та висоти (h), та площі вікон, розташованих з певної сторони будівлі (S_n_wd). Для підлоги по ґрунту – це тільки площа поверхні.

У програмному коді це має наступний вигляд:

```
I_rad = list() # solar irradiation
I_rad = I_n # solar irradiation
S_w = l * h - S_n_wd # north wall area
```

Також, поза функцією, об’являються коефіцієнти, запозичені з методу кінцевих різниць (формула (3.78)) для шару ізоляції, шару стіни та шару повітря, що необхідні для розрахунку температурних характеристик огороджувальних конструкцій будівлі:

```
# FDM coefficient for insulation layer
k_ins = ((L_ins) / (C_ins * r_ins)) / (2 * (((L_ins)/(C_ins * r_ins)) +1))
# FDM coefficient for wall layer
k_w = ((L_w) / (C_w*r_w)) / (2 * (((L_w) / (C_w * r_w)) + 1))
# FDM coefficient for air (inside building)
k_air = ((L_air) / (C_air * r_air)) / (2 * (((L_air)/(C_air * r_air))+1))
```

Для вікон, всередині функції об’являються додатково такі змінні як:

- Маса вікна (m_wd).
- Коефіцієнт, запозичений з методу кінцевих різниць, для вікна (k_wd), що враховує кількість шарів скла.
- Коефіцієнт прозорості, що враховує кількість шарів скла (R_wd_in).
- Коефіцієнт теплопровідності вікна, що враховує кількість шарів скла (L_wd).
- Товщина вікна (B_wd).

У програмному коді це має наступний вигляд:

```
I_rad = list() # solar irradiation
I_rad = I_n # solar irradation
S_wd = S_n_wd # windows area (m2)
m_wd = r_wd * S_wd * x  # window mass
k_wd = x * ((L_g) / (C_wd * r_wd)) / (2 * (((L_g) / (C_wd * r_wd)) + 1)) + k_air #
FDM coefficient for window
R_t_wd = (0.95 ** (x))  # transparency coefficient
L_wd = x * L_g + L_air  # window heat conductivity coefficient
B_wd = (x + 1) * B_gl  # window thickness
```

Далі, об’являються змінні для розрахунку температури зовнішнього шару огороджувальних конструкцій будівлі:

- Масив значень зовнішньої температури (T_w_out), для якого, одразу, розраховується температура в початковий момент часу за формулою (3.74).
- Масив значень температури випромінювання (T_w_irr).
- Масив значень температури зовнішнього шару стіни з урахуванням конвективної складової (T_w_out_w).
- Масив значень теплопередачі конвекцією (s_w_out).

У програмному коді це має наступний вигляд:

```
T_w_out = list() # outside temperaure (K)
T_w_out.insert(0, (T_street[0] - (((T_street[0])**4)*(R_0-R_out))**(1/4))) #
initial outside temperature (K)
T_w_irr = list() # irradiation temperature
T_w_out_w = list() # out wall temperature including radiation and convection
effects (K)
s_w_out = list() # out wall convection (W)
```

Далі, об’являються змінні для розрахунку температури внутрішніх шарів огороджувальних конструкцій будівлі:

- Масив значень температури шару (T_w_layer).
- Масив значень теплопередачі теплопровідністю (l_w_layer).
- Масив значень теплоємності (C_w_layer).

У програмному коді це має наступний вигляд:

```
T_w_1layer = list() # first layer temperature (K)
l_w_1layer = list() # first layer thermal conductivity (W)
C_w_1layer = list() # first layer heat capacity

T_w_2layer = list() # second layer temperature (K)
```

```
l_w_2layer = list() # second layer thermal conductivity (W)
C_w_2layer = list() # second layer heat capacity

T_w_3layer = list() # third layer temperature (K)
l_w_3layer = list() # third layer thermal conductivity (W)
C_w_3layer = list() # third layer heat capacity

T_w_4layer = list() # fourth layer temperature (K)
l_w_4layer = list() # fourth layer thermal conductivity (W)
C_w_4layer = list() # fourth layer heat capacity

T_w_5layer = list() # fifth layer temperature (K)
l_w_5layer = list() # fifth layer thermal conductivity (W)
C_w_5layer = list() # fifth layer heat capacity
```

Далі, об’являються змінні для розрахунку температури внутрішніх шарів огороджувальних конструкцій будівлі:

- Масив значень температури шару (T_w_in).
- Масив значень теплопередачі конвекцією (s_w_in).
- Масив значень теплопередачі випромінюванням від стін усередину будівлі (I_w_in).
- Масив значень теплоємності будівлі (C_w_in).
- Масив значень питомих температур (T_w_in_b)

У програмному коді це має наступний вигляд:

```
T_w_in = list() # inside building temperature (K)
s_w_in = list() # inside building convection (W)
I_w_in = list() # inside wall radiaion (W)
C_w_in = list() # inside building capacity (J)
T_w_in_b = list() # inside building specific temperature (K)
```

Далі, об’являються початкові температури кожного внутрішнього шару. Для цього, до початкової температури попереднього шару додається певна постійна, величина якої має вплив на результати тільки в перші дві-три розрахункові доби, з плином розрахунку, температури вирівнюються і приходять до певних значень.

У програмному коді це має наступний вигляд:

```
# first layer initial temperature (K)
T_w_1layer.insert(0, T_w_out[0] + 5)
# second layer initial temperature (K)
```

```
T_w_2layer.insert(0, T_w_1layer[0] + 4)
# second layer initial temperature (K)
T_w_3layer.insert(0, T_w_2layer[0] + 3)
# second layer initial temperature (K)
T_w_4layer.insert(0, T_w_3layer[0] + 2)
# second layer initial temperature (K)
T_w_5layer.insert(0, T_w_4layer[0] + 1)
# inside building initial temperature (K)
T_w_in.insert(0, T_w_5layer[0] - k_air)
```

Розрахунок зовнішнього шару виглядає наступним чином: цикл «for», у якому година за годиною визначаються і записуються у масиви значення наступних величин:

- Теплопередача конвекцією (s_w_out), що розраховується за формулою (1.3).
- Зовнішня температура (T_w_out), що розраховується за формулою (3.72).
- Теплопередача випромінюванням (T_w_irr), що розраховується за формулою (4.73).
- Температура зовнішньої поверхні стіни з урахуванням конвекції (T_w_out_w), що розраховується за формулою (4.75).

У програмному коді це має наступний вигляд:

```
for i in range(0, n+7): # out wall calculations
    s_w_out.insert(i, s_out * S_w * (T_street[i] - T_w_out[i]))
    T_w_out.insert(i+1, (T_street[i+1] - s_w_out[i] / (s_out * S_w)))
    T_w_irr.insert(i, ((T_w_out[i]**4 + (I_rad[i] * S_w / 3600) / (S_w *
R_0))**(1/4)))
    T_w_out_w.insert(i, (T_w_out[i] * (abs(s_w_out[i]) / (abs(s_w_out[i]) +
abs((I_rad[i] * S_w / 3600)))) + T_w_irr[i] * (abs((I_rad[i] * S_w / 3600)) /
(abs(s_w_out[i]) + abs((I_rad[i] * S_w / 3600))))))
```

Розрахунок внутрішнії шарів виконується за таким саме принципом, у масиви записуються значення величини:

- Теплопередачі теплопровідністю (l_w_layer), що розраховується за формулою (1.1).
- Теплоємності (C_w_layer), що розраховується за формулою (1.5).

- Температури внутрішнього шару (T_w_layer), що розраховується за формулою (3.79).

У програмному коді це має наступний вигляд:

```
for i in range(0, n+6): # first layer calculations
    l_w_1layer.insert(i, L_ins * S_w * (T_w_out_w[i] - T_w_1layer[i]) / B_ins_w)
    C_w_1layer.insert(i, (C_ins * r_ins * B_ins_w * S_w * (T_w_out_w[i] -
T_w_1layer[i])) / 3600)
    T_w_1layer.insert(i+1, ((T_w_out_w[i] - ((l_w_1layer [i] * B_ins_w) / (L_ins *
S_w))) * (abs(l_w_1layer[i]) / (2 * abs(l_w_1layer[i]) + 2 * abs(C_w_1layer[i])))))
+ ((T_w_out_w[i+1] - ((C_w_1layer[i] * 3600)/ (C_ins * r_ins * B_l * S_w))) *
(abs(C_w_1layer[i]) / (2 * abs(C_w_1layer[i]) + 2 * abs(l_w_1layer[i])))) +
((T_w_out_w[i+1] - k_ins) * (abs(l_w_1layer[i]) / (2 * abs(C_w_1layer[i]) + 2 *
abs(l_w_1layer[i])))) + ((T_w_1layer[i] - k_ins) * (abs(C_w_1layer[i]) / (2 *
abs(C_w_1layer[i]) + 2 * abs(l_w_1layer[i]))))
```

Продемонстровано тільки розрахунок першого шару, що є ідентичним до розрахунків всіх інших шарів.

Далі, розраховується температура внутрішньої маси будівлі, у масиви записуються значення величин:

- Теплопередачі конвекцією (s_w_in), що розраховується за формулою (1.3).
- Теплопередачі випромінюванням від стін до внутрішньої маси будівлі (I_w_in), що розраховується за формулою (1.4).
- Теплоємності внутрішньої маси будівлі (C_w_in), що розраховується за формулою (1.5).
- Температура внутрішньої маси будівлі для певної огороджувальної конструкції (T_w_in), що розраховується за формулою (3.80).

У програмному коді це має наступний вигляд:

```
for i in range(0, n+1): # inside building calculations
    s_w_in.insert(i, s_in_w * S_w * (T_w_5layer[i] - T_w_in[i]))
    I_w_in.insert(i, (T_w_5layer[i]**4 - T_w_in[i]**4) * S_w * R_in)
    C_w_in.insert(i, (C_w * r_w * B_in * S_in * (T_w_5layer[i] - T_w_in[i])) /
3600 + ((C_air * r_air * l * w * h) * (T_w_5layer[i] - T_w_in[i])) / 3600)
    T_w_in.insert(i+1, (T_w_5layer[i+1] - s_w_in[i] / (s_in_w * S_w)) *
(abs(s_w_in[i]) / (abs(s_w_in[i]) + abs(C_w_in[i]) + abs(I_w_in[i]))) +
(T_w_5layer[i+1] - (C_w_in[i] * 3600) / (C_w * r_w * B_in * S_in + C_air * r_air *
```

```
l * w * h)) * (abs(C_w_in[i]) / (abs(s_w_in[i]) + abs(C_w_in[i]) +
abs(I_w_in[i]))) + (((T_w_5layer[i+1])**4 - I_w_in[i] / (S_w * R_in))**(1/4)) *
(abs(I_w_in[i]) / (abs(s_w_in[i]) + abs(C_w_in[i]) + abs(I_w_in[i]))))
```

Далі, питома температура внутрішньої маси будівлі для певної огороджувальної конструкції, розраховується за формулою (3.82). У програмному коді це має наступний вигляд:

```
for i in range(0, n): # inside building specific temperature calculations
    T_w_in_b.insert(i, T_w_in[i] * S_w / S_in_b)
```

Функція повертає питому температуру внутрішньої маси будівлі для певної огороджувальної конструкції:

```
return(T_w_in_b)
```

Для функції розрахунку температурних характеристик вікон, за формулою (3.81), додатково враховується потужність сонячного випромінювання, що проходить крізь прозоре вікно напряму на поверхню внутрішніх стін будівлі. У програмному коді це має наступний вигляд:

```
for i in range(0, n+1): # inside building temperature including throuh-window
radiation calculation
    T_wd_rad_in.insert(i, T_wd_in[i] * ((abs(s_wd_in[i]) + abs(C_wd_in[i])+
abs(I_wd_in[i])) / ((abs(s_wd_in[i]) + abs(C_wd_in[i])+ abs(I_wd_in[i])) +
abs((R_t_wd * I_rad[i] * S_wd / 3600)))) + T_wd_irr[i] * (abs((R_t_wd * I_rad[i] *
S_wd / 3600)) / (abs(s_wd_in[i]) + abs(C_wd_in[i]) + abs(I_wd_in[i]) + abs((R_t_wd
* I_rad[i] * S_wd / 3600)))))
```

Усі інші розрахунки всередині функції розрахунку теплоенергетичних характеристик вікон, не відрізняються від розрахунків для стіни.

Об’явлення функції розрахунку теплоенергетичних характеристик стіни Тромбе виглядає наступним чином:

```
def Trombe_wall_temp_calc(B_wd_tr, B_g, B_tr):
```

Функція має три аргументи:

- Товщина скління зовнішнього шару (B_wd_tr).
- Товщина повітряного прошарку (B_g).
- Товщина масивної стіни (B_tr).

Об’явлення змінних загальних характеристик включає в себе:

- Маса зовнішнього скління стіни Тромбе (m_wd_tr).

- Коефіцієнт, запозичений з методу кінцевих різниць, для скління зовнішнього шару стіни Тромбе (k_wd).
- Коефіцієнт прозорості скління зовнішнього шару стін тромбе, що залежить від товщини скління (R_t_wd).
- Коефіцієнт теплопровідності скління (L_wd).

У програмному коді це має наступний вигляд:

```
m_wd_tr = r_wd * l * h * B_wd_tr  # Trombe wall glass' mass
k_wd_tr = (((L_g) / (C_wd * r_wd)) / (2 * (((L_g)/(C_wd * r_wd))+1))) # FDM
coefficient for Trombe wall
R_t_wd = (0.95 ** (B_wd_tr/0.02)) # transparency coefficient
L_wd = L_g # thermal conducivity coefficient
```

Розрахунок зовнішнього шару повністю співпадає з розрахунками цього ж шару для інших огороджувальних конструкцій, як і розрахунок шару скління. Для розрахунку шару повітряного прошарку, у масиви записуються значення величин:

- Теплопередача теплопровідністю (l_tr_mw_out), що розраховується за формулою (1.1).
- Теплоємність (C_tr_mw_out), що розраховується за формулою (1.5)
- Теплове випромінювання всередині повітряного прошарку (I_tr_mw_out), що розраховується за формулою (1.4)
- Коефіцієнт конвективної теплопередачі всередині повітряного прошарку (s_ag), що розраховується за формулою (3.83)
- Теплопередачіа конвекцією (s_tr_mw_out), що розраховується за формулою (1.3)
- Температура повітряного прошарку (T_tr_mw_out), що розраховується за формулою (3.82).

У цьому ж циклі, додатково розраховується температура поверхні масивної стіни (T_tr_mw_rad_out). Враховується потужність сонячного випромінювання, що проходить крізь скління зовнішнього шару на поверхню внутрішніх масивної стіни.

У програмному коді це має наступний вигляд:

```
for i in range(0, n+3): # air gap layer calculations
    l_tr_mw_out.insert(i, L_air * S_tr * (T_g_layer[i] - T_tr_mw_out[i]) / B_g)
    C_tr_mw_out.insert(i, (C_air * r_air * B_g * S_tr * (T_g_layer[i] -
```

```
T_tr_mw_out[i]) / 3600))
    I_tr_mw_out.insert(i, (T_g_layer[i]**4 - T_tr_mw_out[i]**4) * S_tr * R_wd_in)
    s_ag.insert(i, (0.5 * ((9.81 * 0.0367 * abs(T_tr_mw_out[i] - T_g_layer[i]) *
B_g**4) / (8 * 0.000016**2 * h))**0.294))
    s_tr_mw_out.insert(i, s_ag[i] * S_tr * (T_g_layer[i] - T_tr_mw_out[i]))
    T_tr_mw_out.insert(i+1, ((T_g_layer[i] - ((l_tr_mw_out[i] * B_g) / (L_air *
S_tr))) * (abs(l_tr_mw_out[i]) / (2 * abs(l_tr_mw_out[i]) + 2 *
abs(C_tr_mw_out[i]) + abs(s_tr_mw_out[i]) + abs(I_tr_mw_out[i])))) +
((T_g_layer[i+1] - ((C_tr_mw_out[i] * 3600)/ (C_air * r_air * B_g * S_tr))) *
(abs(C_tr_mw_out[i]) / (2 * abs(l_tr_mw_out[i]) + 2 * abs(C_tr_mw_out[i]) +
abs(s_tr_mw_out[i]) + abs(I_tr_mw_out[i])))) + ((T_g_layer[i+1] - k_air) *
(abs(l_tr_mw_out[i]) / (2 * abs(l_tr_mw_out[i]) + 2 * abs(C_tr_mw_out[i]) +
abs(s_tr_mw_out[i]) + abs(I_tr_mw_out[i])))) + ((T_tr_mw_out[i] - k_air) *
(abs(C_tr_mw_out[i]) / (2 * abs(l_tr_mw_out[i]) + 2 * abs(C_tr_mw_out[i]) +
abs(s_tr_mw_out[i]) + abs(I_tr_mw_out[i])))) + ((T_g_layer[i+1] - s_tr_mw_out[i] /
(s_ag[i] * S_tr)) * (abs(s_tr_mw_out[i]) / (2 * abs(l_tr_mw_out[i]) + 2 *
abs(C_tr_mw_out[i]) + abs(s_tr_mw_out[i]) + abs(I_tr_mw_out[i])))) +
(((T_g_layer[i+1]**4 - I_tr_mw_out[i] / (R_wd_in * S_tr))**(1/4)) *
(abs(I_tr_mw_out[i]) / (2 * abs(l_tr_mw_out[i]) + 2 * abs(C_tr_mw_out[i]) +
abs(s_tr_mw_out[i]) + abs(I_tr_mw_out[i]))))))
    T_tr_mw_rad_out.insert(i, (T_tr_mw_out[i] * ((abs(l_tr_mw_out[i]) +
abs(C_tr_mw_out[i]) + abs(I_tr_mw_out[i]) + abs(s_tr_mw_out[i])) /
(abs(l_tr_mw_out[i]) + abs(C_tr_mw_out[i]) + abs(I_tr_mw_out[i]) +
abs(s_tr_mw_out[i]) + abs((R_t_wd * I_s[i] * S_tr / 3600))))) + (T_tr_irr[i] *
(abs((R_t_wd * I_s[i] * S_tr / 3600)) / (abs(l_tr_mw_out[i]) + abs(C_tr_mw_out[i])
+ abs(I_tr_mw_out[i]) + abs(s_tr_mw_out[i]) + abs((R_t_wd * I_s[i] * S_tr /
3600))))))
```

Розрахунок температури внутрішньої поверхні масивної стіни відбувається аналогічно до внутрішніх шарів огороджувальних конструкцій. Для розрахунку температури внутрішнього шару, у масиви записуються значення величин:

- Теплопередачі конвекцією (s_tr_in), що розраховується за формулою (1.3).
- Теплопередача теплопровідністю (l_tr_in), що розраховується за формулою (1.1).
- Теплоємності внутрішньої маси будівлі (C_tr_in), що розраховується за формулою (1.5).
- Температура внутрішньої маси будівлі для стіни Тромбе (T_tr_in), що розраховується за формулою (3.80).

У програмному коді це має наступний вигляд:

```
for i in range(0, n+1): # inside building calculations
    s_tr_in.insert(i, s_in_tr * S_tr * (T_tr_mw_in[i] - T_tr_in[i]))
    I_tr_in.insert(i, (T_tr_mw_in[i]**4 - T_tr_in[i]**4) * S_tr * R_in)
    C_tr_in.insert(i, (C_w * r_w * B_in * S_in * (T_tr_mw_in[i] - T_tr_in[i])) /
3600 + ((C_air * r_air * l * w * h) * (T_tr_mw_in[i] - T_tr_in[i])) / 3600)
    T_tr_in.insert(i+1, (T_tr_mw_in[i+1] - s_tr_in[i] / (s_in_tr * S_tr)) *
(abs(s_tr_in[i]) / (abs(s_tr_in[i]) + abs(C_tr_in[i]) + abs(I_tr_in[i]))) +
(T_tr_mw_in[i+1] - (C_tr_in[i] * 3600) / (C_w * r_w * B_in * S_in + C_air * r_air
* l * w * h)) * (abs(C_tr_in[i]) / (abs(s_tr_in[i]) + abs(C_tr_in[i]) +
abs(I_tr_in[i]))) + (((T_tr_mw_in[i+1])**4 - I_tr_in[i] / (S_tr * R_in))**(1/4)) *
(abs(I_tr_in[i]) / (abs(s_tr_in[i]) + abs(C_tr_in[i]) + abs(I_tr_in[i]))))
```

Далі, питома температура внутрішньої маси будівлі для стіни Тромбе розраховується за формулою (3.82). У програмному коді це має наступний вигляд:

```
for i in range(0, n): # inside building specific temperature calculations
    T_tr_in_b.insert(i, T_tr_in[i] * S_tr / S_in_b)
```

Функція повертає питому температуру внутрішньої маси будівлі для стіни Тромбе:

```
return(T_tr_in_b)
```

Поруч з цією функцією, створена ще одна, з тими самими аргументами, для розрахунку температурних характеристик стіни Тромбе, при використанні ізоляції від зовнішнього сонячного випромінювання:

```
def Trombe_wall_0_temp_calc(B_wd_tr, B_g, B_tr):
```

Ця функція має ті самі аргументи, що і попередня. Її відмінністю є розрахунки для зовнішнього шару та шару повітряного прошарку стіни Тромбе, тобто тих, де має вплив інтенсивність сонячного випромінювання. Ця інтенсивність, для врахування наявності ізоляції, встановлюється на рівні 5 відсотків від номінальної.

У програмному коді це має наступний вигляд. Розрахунок температури зовнішнього шару:

```
for i in range(0, n + 5):  # out wall calculations
    s_tr_out.insert(i, s_out * S_tr * (T_street[i] - T_tr_out[i]))
    T_tr_out.insert(i + 1, (T_street[i + 1] - s_tr_out[i] / (s_out * S_tr)))
    T_tr_irr.insert(i, ((T_tr_out[i] ** 4 + (0.05 * I_s[i] * S_tr / 3600) / (S_tr
* R_0)) ** (1 / 4)))
    T_tr_out_w.insert(i, (T_tr_out[i] * (abs(s_tr_out[i]) / (abs(s_tr_out[i]) +
```

```
abs((0.05 * I_s[i] * S_tr / 3600)))) + T_tr_irr[i] * (abs((0.05 * I_s[i] * S_tr /
3600)) / (abs(s_tr_out[i]) + abs((0.05 * I_s[i] * S_tr / 3600))))))
```

Розрахунок температури зовнішньої поверхні масивної стіни з урахуванням інтенсивності сонячного випромінювання, при використанні ізоляції:

```
T_tr_mw_rad_out.insert(i, (T_tr_mw_out[i] * ((abs(l_tr_mw_out[i]) +
abs(C_tr_mw_out[i]) + abs(I_tr_mw_out[i]) + abs(s_tr_mw_out[i])) /
(abs(l_tr_mw_out[i]) + abs(C_tr_mw_out[i]) + abs(I_tr_mw_out[i]) +
abs(s_tr_mw_out[i]) + abs((0.05 * R_t_wd * I_s[i] * S_tr / 3600))))) +
(T_tr_irr[i] * (abs((0.05 * R_t_wd * I_s[i] * S_tr / 3600)) / (abs(l_tr_mw_out[i])
+ abs(C_tr_mw_out[i]) + abs(I_tr_mw_out[i]) + abs(s_tr_mw_out[i]) + abs((0.05 *
R_t_wd * I_s[i] * S_tr / 3600))))))
```

Функція розрахунку енергетичних потреб будівлі має дев’ять аргументів:

- Товщина шару стіни (B_l_w).
- Товщина шару ізоляції стіни (B_ins_w).
- Товщина шару підлоги по ґрунту (B_l_f).
- Товщина шару ізоляції підлоги по ґрунту (B_ins_f).
- Товщина шару горища (B_l_r).
- Товщина шару ізоляції горища (B_ins_r).
- Кількість шарів скла у вікнах (x).
- Товщина скління зовнішнього шару стіни Тромбе (B_wd_tr).
- Товщина повітряного прошарку стіни Тромбе (B_g).
- Товщина масивної стіни (B_tr).
- Температура за якої вмикається опалення будівлі (T_h).
- Температура за якої вмикається кондиціонування будівлі (T_c).

Об’являється функція розрахунку енергопотреб будівлі наступним чином:

```
def TW_build_en_cons_calc(B_l_w, B_ins_w,
                          B_l_f, B_ins_f,
                          B_l_r, B_ins_r,
                          x,
                          B_wd_tr, B_g, B_tr,
                          T_h, T_c):
```

Далі об’являються необхідні змінні:

- Масив значень внутрішньої температури (T_in).

- Масив значень температури внутрішньої маси будівлі для північної стіни (T_n_w_in).
- Масив значень температури внутрішньої маси будівлі для східної стіни (T_e_w_in).
- Масив значень температури внутрішньої маси будівлі для західної стіни (T_w_w_in).
- Масив значень внутрішньої маси будівлі для стіни Тромбе (T_tr_w_in).
- Масив значень температури внутрішньої маси будівлі для стіни Тромбе, при використанні ізоляції від зовнішнього сонячного випромінювання (T_tr_w_in_0).
- Масив значень температури внутрішньої маси будівлі для горища (T_f_in).
- Масив значень температури внутрішньої маси будівлі для підлоги по ґрунту (T_r_in).
- Масив значень температури внутрішньої маси будівлі для північного вікна (T_n_wd_in).
- Масив значень температури внутрішньої маси будівлі для східного вікна (T_e_wd_in).
- Масив значень температури внутрішньої маси будівлі для західного вікна (T_w_wd_in).
- Масив значень енергетичних потреб кондиціонування, розрахованих погодинно, за методом градусо-секунд (W_in_c).
- Масив значень енергетичних потреб опалення, розрахованих погодинно, за методом градусо-секунд (W_in_h).

У програмному коді це має наступний вигляд:

```
T_in = list()  # inside temperature (K)
T_n_w_in = list()  # inside north wall temperature (K)
T_e_w_in = list()  # inside eas wall temperature (K)
T_w_w_in = list()  # inside west wall temperature (K)
T_tr_w_in = list() # inside Trombe wall temperature (K)
T_tr_w_in_0 = list()  # inside isolated Trombe wall with temperature (K)
```

```
T_f_in = list()  # inside floor temperature (K)
T_r_in = list()  # inside roof temperature (K)
T_n_wd_in = list()  # inside north windows temperature (K)
T_e_wd_in = list()  # inside east windows temperature (K)
T_w_wd_in = list()  # inside west windows temperature (K)

W_in_c = list()  # cooling degree-seconds
W_in_h = list()  # heating degree-seconds calculations
```

Далі викликаються функції розрахунку питомої температури внутрішньої маси будівлі для кожної огороджувальної конструкції. Вони об’явлені як змінні для зручності подальших розрахунків:

- Північної стіни, об’явленої як (T_n_w_in).
- Східної стіни, об’явленої як (T_e_w_in).
- Західної стіни, об’явленої як (T_w_w_in).
- Підлоги по ґрунту, об’явленої як (T_f_in).
- Горища, об’явленої як (T_r_in).
- Стіни Тромбе, об’явленої як (T_tr_w_in).
- Стіни Тромбе, при використанні ізоляції від зовнішнього сонячного випромінювання, об’явленої як (T_tr_w_in_0).

У програмному коді це має наступний вигляд:

```
T_n_w_in = north_wall_temp_calc(B_ins_w, B_l_w)
T_e_w_in = east_wall_temp_calc(B_ins_w, B_l_w)
T_w_w_in = west_wall_temp_calc(B_ins_w, B_l_w)
T_f_in = floor_temp_calc(B_ins_f, B_l_f)
T_r_in = roof_temp_calc(B_ins_r, B_l_r)
T_tr_w_in = Trombe_wall_temp_calc(B_wd_tr, B_g, B_tr)
T_tr_w_in_0 = Trombe_wall_0_temp_calc(B_wd_tr, B_g, B_tr)
if S_n_wd > 0:
    T_n_wd_in = north_windows_temp_calc(x)
else:
    for i in range(0, n):
        T_n_wd_in.insert(i, 0)
if S_e_wd > 0:
    T_e_wd_in = east_windows_temp_calc(x)
else:
    for i in range(0, n):
        T_e_wd_in.insert(i, 0)
```

```
if S_w_wd > 0:
    T_w_wd_in = west_windows_temp_calc(x)
else:
    for i in range(0, n):
        T_w_wd_in.insert(i, 0)
```

Для перевірки наявності вікон у будівлі на певній стороні світу, використовується конструкція «if-else».

Використання вище згаданої ізоляції – імітації закривання доступу сонячного випромінювання до конструкції стіни Тромбе за допомогою непрозорого матеріалу, виконується наступним чином: коли сумарна температура внутрішньої маси будівлі перевищує температуру вмикання опалення – імітується використання ізоляції.

У програмному коді це має наступний вигляд:

```
for i in range(0, n): # solar insulation immitation
    if (T_n_w_in[i] + T_e_w_in[i] + T_w_w_in[i] +
        T_tr_w_in[i] + T_r_in[i] + T_f_in[i] +
        T_n_wd_in[i] + T_e_wd_in[i] + T_w_wd_in[i]) > T_c:
        T_in.insert(i, (T_n_w_in[i] + T_e_w_in[i] + T_w_w_in[i] +
                        T_tr_w_in_0[i] + T_r_in[i] + T_f_in[i] +
                        T_n_wd_in[i] + T_e_wd_in[i] + T_w_wd_in[i]))
    else:
        T_in.insert(i, (T_n_w_in[i] + T_e_w_in[i] + T_w_w_in[i] +
                        T_tr_w_in[i] + T_r_in[i] + T_f_in[i] +
                        T_n_wd_in[i] + T_e_wd_in[i] + T_w_wd_in[i]))
```

Після цього, проводиться розрахунок за методом градусо-секунд (формули (3.85-3.88)). Конструкція «if – else» використовується для задання температурних умов, усередині циклу «for» для наповнення масиву значень енергопотреби кондиціонування та опалення.

```
for i in range(0, n): # degree-seconds method (cooling)
    if T_in[i] - T_c > 0:
        W_in_c.insert(i, (T_in[i] - T_c) * 3600)
    else:
        W_in_c.insert(i, 0)

W_c = sum(W_in_c) * 158.9 / 3600 / 1000  # sum cooling energy consumption

for i in range(0, n): # degree-seconds method (heating)
```

```
    if T_h - T_in[i] > 0:
        W_in_h.insert(i, (T_h - T_in[i]) * 3600)
    else:
        W_in_h.insert(i, 0)

W_h = sum(W_in_h) * 158.9 / 3600 / 1000  # sum heating energy consumption
```

Далі, виводиться сумарне енергоспоживання:

```
W_sum = W_c + W_h  # sum energy consumption (kWh)
```

Функція повертає енергетичну потребу опалення, кондиціонування та сумарну енергетичну потребу:

```
return (W_c, W_h, W_sum, T_in)
```

Функція оптимізації параметрів пасивної системи опалення, має ті самі аргументи, що й функція розрахунку енергетичних потреб будівлі, її об'явлення виглядає наступним чином:

```
def Optimization(B_l_w,
                 B_ins_w,
                 B_l_f,
                 B_ins_f,
                 B_l_r,
                 B_ins_r,
                 x,
                 T_h,
                 T_c):
```

Усередині функції, спочатку об'являються необхідні змінні:

- Масив значень енергетичних потреб будівлі при різних значеннях товщини скління зовнішнього шару стіни Тромбе (Opt_wd).
- Масив значень енергетичних потреб будівлі при різних значеннях товщини повітряного прошарку стіни Тромбе (Opt_g).
- Масив значень енергетичних потреб будівлі при різних значеннях товщини масивної стіни (Opt_w).

Поряд з масивами значень енергетичних потреб будівлі, задаються масиви значень, з яких будуть обиратися оптимальні:

- Набір значень товщини скління зовнішнього шару стіни Тромбе у вигляді масиву (B_wd_tr).

- Набір значень товщини повітряного прошарку стіни Тромбе у вигляді масиву (B_g).
- Набір значень товщини масивної стіни у вигляді масиву (B_tr).

У програмному коді це має наступний вигляд:

```
Opt_wd = list() # Energy consumption with different Trombe wall window width
B_wd_tr = list([0.04, 0.06, 0.08, 0.1, 0.12]) # Trombe wall window width

Opt_g = list() # Energy consumption with different Trombe wall air gap width
B_g = list([0.05, 0.1, 0.15, 0.2, 0.25]) # Trombe wall air gap width

Opt_w = list() # Energy consumption with different Massive wall width
B_tr = list([0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6]) # Massive
wall width
```

Далі проводиться розрахунок оптимальної товщини скління зовнішнього шару стіни Тромбе. Для цього, функція для розрахунку енергетичних потреб будівлі викликається таку кількість разів, скільки значень у масиві даних на вибір для товщини скління зовнішнього шару (B_wd_tr). Значення товщини масивної стіни (B_tr) та повітряного прошарку (B_g) при цьому фіксуються на одному значенні.

У програмному коді це має наступний вигляд:

```
Opt_wd.insert(0, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                        x,
                                        B_wd_tr[0], B_g[2], B_tr[5],
                                        T_h, T_c)[2]))
Opt_wd.insert(1, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                        x,
                                        B_wd_tr[1], B_g[2], B_tr[5],
                                        T_h, T_c)[2]))
Opt_wd.insert(2, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                        x,
                                        B_wd_tr[2], B_g[2], B_tr[5],
                                        T_h, T_c)[2]))
Opt_wd.insert(3, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                        x,
```

```
                                        B_wd_tr[3], B_g[2], B_tr[5],
                                        T_h, T_c)[2]))
Opt_wd.insert(4, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                        x,
                                        B_wd_tr[4], B_g[2], B_tr[5],
                                        T_h, T_c)[2]))
```

Далі, оптимальне значення товщини скління зовнішнього шару стіни Тромбе (B_wd_tr_opt) визначається як значення товщини скління зовнішнього шару стіни Тромбе (B_wd_tr), при якому масив значень (Opt_wd) мав мінімальне значення, тобто, за якого загальна енергопотреба мала найменше значення.

У програмному коді це має наступний вигляд:

```
B_wd_tr_opt = B_wd_tr[(Opt_wd.index(min(Opt_wd)))]
```

Далі, таким же чином, розраховуються енергопотреби будівлі при різних значеннях товщини повітряного прошарку та оптимальне значення товщини повітряного прошарку. При цьому, приймається оптимальне значення товщини скління зовнішнього шару стіни Тромбе.

У програмному коді це має наступний вигляд:

```
Opt_g.insert(0, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                       x,
                                       B_wd_tr_opt, B_g[0], B_tr[5],
                                       T_h, T_c)[2]))
Opt_g.insert(1, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                       x,
                                       B_wd_tr_opt, B_g[1], B_tr[5],
                                       T_h, T_c)[2]))
Opt_g.insert(2, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                       x,
                                       B_wd_tr_opt, B_g[2], B_tr[5],
                                       T_h, T_c)[2]))
Opt_g.insert(3, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                       x,
                                       B_wd_tr_opt, B_g[3], B_tr[5],
                                       T_h, T_c)[2]))
```

```
Opt_g.insert(4, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                       x,
                                       B_wd_tr_opt, B_g[4], B_tr[5],
                                       T_h, T_c)[2]))

B_g_opt = B_g[(Opt_g.index(min(Opt_g)))]
```

Далі, таким же чином, розраховуються енергопотреби будівлі при різних значеннях товщини масивної стіни та оптимальне значення товщини масивної стіни. При цьому, приймається оптимальне значення товщини скління зовнішнього шару стіни Тромбе та товщини повітряного прошарку.

У програмному коді це має наступний вигляд:

```
Opt_w.insert(0, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                       x, B_wd_tr_opt, B_g_opt, B_tr[0], T_h,
T_c)[2]))
Opt_w.insert(1, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                       x, B_wd_tr_opt, B_g_opt, B_tr[1], T_h,
T_c)[2]))
Opt_w.insert(2, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                       x, B_wd_tr_opt, B_g_opt, B_tr[2], T_h,
T_c)[2]))
Opt_w.insert(3, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                       x, B_wd_tr_opt, B_g_opt, B_tr[3], T_h,
T_c)[2]))
Opt_w.insert(4, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                       x, B_wd_tr_opt, B_g_opt, B_tr[4], T_h,
T_c)[2]))
Opt_w.insert(5, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                       x, B_wd_tr_opt, B_g_opt, B_tr[5], T_h,
T_c)[2]))
Opt_w.insert(6, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                       x, B_wd_tr_opt, B_g_opt, B_tr[6], T_h,
T_c)[2]))
```

```
Opt_w.insert(7, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                       x, B_wd_tr_opt, B_g_opt, B_tr[7], T_h,
T_c)[2]))
Opt_w.insert(8, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                       x, B_wd_tr_opt, B_g_opt, B_tr[8], T_h,
T_c)[2]))
Opt_w.insert(9, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                       x, B_wd_tr_opt, B_g_opt, B_tr[9], T_h,
T_c)[2]))

B_tr_opt = B_tr[(Opt_w.index(min(Opt_w)))]
```

Після цього, визначаються енергетичні потреби будівлі при оптимальних параметрах стіни Тромбе:

- Значення енергопотреби кондиціонування за оптимальних значень параметрів стіни Тромбе (Opt_C).
- Значення енергопотреби опалення за оптимальних значень параметрів стіни Тромбе (Opt_H).
- Значення загальної енергопотреби за оптимальних значень параметрів стіни Тромбе (Opt_O).

У програмному коді це має наступний вигляд:

```
Opt_C = (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r, B_ins_r,
                               x,
                               B_wd_tr_opt, B_g_opt, B_tr_opt,
                               T_h, T_c)[0])
Opt_H = (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r, B_ins_r,
                               x,
                               B_wd_tr_opt, B_g_opt, B_tr_opt,
                               T_h, T_c)[1])
Opt_O = (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r, B_ins_r,
                               x,
                               B_wd_tr_opt, B_g_opt, B_tr_opt,
                               T_h, T_c)[2])
```

Функція повертає оптимізовані значення параметрів та енергопотреб:

- Оптимальне значення товщини скління зовнішнього шару стіни Тромбе (B_wd_tr_opt).
- Оптимальне значення товщини повітряного прошарку стіни Тромбе (B_g_opt).
- Оптимальне значення товщини масивної стіни (B_tr_opt).
- Значення енергетичних потреб кондиціонування за оптимальних значень параметрів стіни Тромбе (Opt_C).
- Значення енергетичних потреб опалення за оптимальних значень параметрів стіни Тромбе (Opt_H).
- Значення загальної енергетичних потреб за оптимальних значень параметрів стіни Тромбе (Opt_O).

У програмному коді це має наступний вигляд:

```
return B_wd_tr_opt, B_g_opt, B_tr_opt, Opt_C, Opt_H, Opt_O
```

Це всі основні функції, необхідні для виконання розрахунків та визначення температурних та теплоенергетичних характеристик будівлі з пасивною системою опалення.

4.4 Підтвердження валідності запропонованого методу

Для того, щоб довести ефективність та точність створеного методу, вирішено було порівняти розрахунки за даним методом, виконані за допомогою програми, із розрахунком за методом, описаним у ДСТУ Б А.2.2-12 2015. Енергетична ефективність будівель. Метод розрахунку енергоспоживання при опаленні, охолодженні, вентиляції, освітленні та гарячому водопостачанні [15]. Метод, описаний у державному стандарті може бути використаним лише для будівель із класичними системами опалення.

Для розрахунку енергетичних потреб будівель із класичними системами опалення програмним методом, була створена окрема функція. Така функція має ті самі аргументи, як і функція для розрахунку енергетичних потреб будівлі з

пасивною системою опалення, крім даних про, власне, стіну Тромбе. Її об’явлення виглядає наступним чином:

```
def reg_build_en_cons_calc(B_l_w, B_ins_w,
                           B_l_f, B_ins_f,
                           B_l_r, B_ins_r,
                           x,
                           T_h, T_c):
```

У всьому іншому, порядку та процесі розрахунку, ця функція також є ідентичною до функції для розрахунку енергетичних потреб будівлі з пасивною системою опалення. Єдиною відмінністю є наявність блоків південних стіни та вікон замість стіни Тромбе. Функція повертає ті самі змінні.

Теплоенергетичні характеристики матеріалів, використаних для імітованої будівлі, прийняті згідно з ДБН В.1.2-2:2006. Навантаження і впливи. Норми проектування [60]. Місцевість, для якої виконувалися розрахунки, значення зовнішньої температури та інтенсивність сонячного випромінювання – місто Херсон.

Загалом, буде представлена вибірка з п’яти будівель з різними вхідними параметрами.

Параметри для розрахунку енергетичних потреб першої будівлі з класичною системою опалення наступні:

- Довжина будівлі (l) – 10 м.
- Ширина будівлі (w) – 6 м.
- Висота будівлі (h) – 6 м.
- Площа північних вікон (S_n_wd) – 0 м^2.
- Площа східних вікон (S_e_wd) – 12 м^2.
- Площа західних вікон (S_w_wd) – 12 м^2.
- Площа південних вікон (S_s_wd) – 12 м^2.
- Матеріал виконання огороджувальних конструкцій – газобетон.
- Матеріал теплоізоляції стін, горища і підлоги по ґрунту – мінеральна вата.
- Товщина стін, горища і підлоги по ґрунту – 0,4 м.

- Товщина теплоізоляції стін, горища і підлоги по ґрунту – 0,1 м.

При виклику функції, вказані такі аргументи:

- Товщина шару стіни (B_l_w) – 0.1 м.
- Товщина шару ізоляції стіни (B_l_ins) – 0.1 м.
- Товщина шару підлоги по ґрунту (B_l_f) – 0.1 м.
- Товщина шару ізолції підлоги по ґрунту (B_f_ins) – 0.1 м.
- Товщина шару горища (B_l_r) – 0.1 м.
- Товщина шару ізоляції горища (B_r_ins) – 0.1 м.
- Кількість шарів скла (x) – 2.
- Температура увімкнення опалення (T_h) – 292, 15 К.
- Температура увімкнення охолодження (T_c) – 299,15 К.

У програмному коді, порівняння це має наступний вигляд:

```
Result_reg = (reg_build_en_cons_calc(0.1, 0.1,
                                     0.1, 0.1,
                                     0.1, 0.1,
                                     2,
                                     292.15, 299.15))
print(Result_reg[0], Result_reg[1], Result_reg[2])
```

Результати розрахунків продемонстровані на рис. 4.2.

```
2597.7636538155534 7950.443541015038 10548.207194830591
```

Рисунок 4.2 – Результати розрахунку енергетичних потреб першої будівлі з класичною системою опалення

У табл. 4.1 продемонстровано порівняння результатів розрахунку енергетичних потреб першої будівлі за ДСТУ Б А.2.2-12 2015 та за створеним методом.

Таблиця 4.1

Порівняння результатів розрахунку енергетичних потреб першої будівлі за ДСТУ Б А.2.2-12 2015 та за створеним методом

Енергетична потреба, кВт×год	Кондиціонування	Опалення	Загальна
Розрахунок за ДСТУ	3477	7445	11155
Розрахунок за створеним методом	2598	7950	10548

Подібність результатів розрахунків наступна:

- За енергетичною потребою на кондиціонування – 75 %.
- За енергетичною потребою на опалення – 94 %.
- За загальною енергетичною потребою – 97 %.

Параметри для розрахунку енергетичних потреб другої будівлі з класичною системою опалення наступні:

- Довжина будівлі (l) – 12 м.
- Ширина будівлі (w) – 8 м.
- Висота будівлі (h) – 6 м.
- Площа північних вікон (S_n_wd) – 10 м^2.
- Площа східних вікон (S_e_wd) – 10 м^2.
- Площа західних вікон (S_w_wd) – 10 м^2.
- Площа південних вікон (S_s_wd) – 20 м^2.
- Матеріал виконання огороджувальних конструкцій – керамічна порожниста кладка густиною 1400 кг/м3 (брутто) на цементно-піщаному розчині.
- Матеріал теплоізоляції стін, горища і підлоги по ґрунту – блоки пінополістирольні.
- Товщина стін, горища і підлоги по ґрунту – 0,5 м.
- Товщина теплоізоляції стін, горища і підлоги по ґрунту – 0,1 м.

При виклику функції, вказані такі аргументи:

- Товщина шару стіни (B_l_w) – 0.125 м.
- Товщина шару ізоляції стіни (B_l_ins) – 0.1 м.
- Товщина шару підлоги по ґрунту (B_l_f) – 0.125 м.
- Товщина шару ізолції підлоги по ґрунту (B_f_ins) – 0.1 м.
- Товщина шару горища (B_l_r) – 0.125 м.
- Товщина шару ізоляції горища (B_r_ins) – 0.1 м.
- Кількість шарів скла (x) – 2.
- Температура увімкнення опалення (T_h) – 292, 15 К.
- Температура увімкнення охолодження (T_c) – 299,15 К.

У програмному коді, порівняння це має наступний вигляд:

```
Result_reg = (reg_build_en_cons_calc(0.125, 0.1,
                                     0.125, 0.1,
                                     0.125, 0.1,
                                     2,
                                     292.15, 299.15))
```

Результати розрахунків продемонстровані на рис. 4.3.

```
2720.998929915425 8077.937767093771 10798.936697009196
```

Рисунок 4.3 – Результати розрахунку енергетичних потреб другої будівлі з класичною системою опалення

У табл. 4.2 продемонстровано порівняння результатів розрахунку енергетичних потреб другої будівлі за ДСТУ Б А.2.2-12 2015 та за створеним методом.

Таблиця 4.2

Порівняння результатів розрахунку енергетичних потреб другої будівлі за ДСТУ Б А.2.2-12 2015 та за створеним методом

Енергетична потреба, кВт×год	Кондиціонування	Опалення	Загальна
Розрахунок за ДСТУ	3260	10277	13537

Продовження табл. 4.2

Енергетична потреба, кВт×год	Кондиціонування	Опалення	Загальна
Розрахунок за створеним методом	2721	8078	10798

Подібність результатів розрахунків наступна:

- За енергетичною потребою на кондиціонування – 83 %.
- За енергетичною потребою на опалення – 79 %.
- За загальною енергетичною потребою – 80 %.

Параметри для розрахунку енергетичних потреб третьої будівлі з класичною системою опалення наступні:

- Довжина будівлі (l) – 15 м.
- Ширина будівлі (w) – 8 м.
- Висота будівлі (h) – 3 м.
- Площа північних вікон (S_n_wd) – 5 м^2.
- Площа східних вікон (S_e_wd) – 10 м^2.
- Площа західних вікон (S_w_wd) – 10 м^2.
- Площа південних вікон (S_s_wd) – 30 м^2.
- Матеріал виконання огороджувальних конструкцій – керамзитобетон на керамзитовому піску.
- Матеріал теплоізоляції стін, горища і підлоги по ґрунту – плити зі скляного штапельного волокна.
- Товщина стін, горища і підлоги по ґрунту – 0,4 м.
- Товщина теплоізоляції стін, горища і підлоги по ґрунту – 0,2 м.

При виклику функції, вказані такі аргументи:

- Товщина шару стіни (B_l_w) – 0.1 м.
- Товщина шару ізоляції стіни (B_l_ins) – 0.2 м.
- Товщина шару підлоги по ґрунту (B_l_f) – 0.1 м.
- Товщина шару ізолції підлоги по ґрунту (B_f_ins) – 0.2 м.
- Товщина шару горища (B_l_r) – 0.1 м.

- Товщина шару ізоляції горища (B_r_ins) – 0.2 м.
- Кількість шарів скла (x) – 2.
- Температура увімкнення опалення (T_h) – 292, 15 К.
- Температура увімкнення охолодження (T_c) – 299,15 К.

У програмному коді, порівняння це має наступний вигляд:

```
Result_reg = (reg_build_en_cons_calc(0.1, 0.2,
                                     0.1, 0.2,
                                     0.1, 0.2,
                                     2,
                                     292.15, 299.15))
```

Результати розрахунків продемонстровані на рис. 4.4.

```
3237.665425072036 8150.325459875771 11387.990884947807
```

Рисунок 4.4 – Результати розрахунку енергетичних потреб третьої будівлі з класичною системою опалення

У табл. 4.3 продемонстровано порівняння результатів розрахунку енергетичних потреб третьої будівлі за ДСТУ Б А.2.2-12 2015 та за створеним методом.

Таблиця 4.3

Порівняння результатів розрахунку енергетичних потреб третьої будівлі за ДСТУ Б А.2.2-12 2015 та за створеним методом

Енергетична потреба, кВт×год	Кондиціонування	Опалення	Загальна
Розрахунок за ДСТУ	3043	8522	11565
Розрахунок за створеним методом	3238	8150	11378

Подібність результатів розрахунків наступна:

- За енергетичною потребою на кондиціонування – 94 %.
- За енергетичною потребою на опалення – 96 %.
- За загальною енергетичною потребою – 98 %.

Параметри для розрахунку енергетичних потреб четвертої будівлі з класичною системою опалення наступні:

- Довжина будівлі (l) – 10 м.
- Ширина будівлі (w) – 10 м.
- Висота будівлі (h) – 3 м.
- Площа північних вікон (S_n_wd) – 6 м^2.
- Площа східних вікон (S_e_wd) – 12 м^2.
- Площа західних вікон (S_w_wd) – 12 м^2.
- Площа південних вікон (S_s_wd) – 20 м^2.
- Матеріал виконання огороджувальних конструкцій – Залізобетон.
- Матеріал теплоізоляції стін, горища і підлоги по ґрунту – плити негорючі теплоізоляційні базальто-волокнисті.
- Товщина стін, горища і підлоги по ґрунту – 0,2 м.
- Товщина теплоізоляції стін, горища і підлоги по ґрунту – 0,3 м.

При виклику функції, вказані такі аргументи:

- Товщина шару стіни (B_l_w) – 0.05 м.
- Товщина шару ізоляції стіни (B_l_ins) – 0.3 м.
- Товщина шару підлоги по ґрунту (B_l_f) – 0.05 м.
- Товщина шару ізолції підлоги по ґрунту (B_f_ins) – 0.3 м.
- Товщина шару горища (B_l_r) – 0.05 м.
- Товщина шару ізоляції горища (B_r_ins) – 0.3 м.
- Кількість шарів скла (x) – 2.
- Температура увімкнення опалення (T_h) – 292, 15 К.
- Температура увімкнення охолодження (T_c) – 299,15 К.

У програмному коді, порівняння це має наступний вигляд:

```
Result_reg = (reg_build_en_cons_calc(0.05, 0.3,
                                     0.05, 0.3,
                                     0.05, 0.3,
                                     2,
                                     292.15, 299.15))
```

Результати розрахунків продемонстровані на рис. 4.5.

```
3418.118464954099 8265.074889444082 11683.193354398181
```

Рисунок 4.5 – Результати розрахунку енергетичних потреб четвертої будівлі з класичною системою опалення

У табл. 4.4 продемонстровано порівняння результатів розрахунку енергетичних потреб четвертої будівлі за ДСТУ Б А.2.2-12 2015 та за створеним методом.

Таблиця 4.4

Порівняння результатів розрахунку енергетичних потреб четвертої будівлі за ДСТУ Б А.2.2-12 2015 та за створеним методом

Енергетична потреба, кВт×год	Кондиціонування	Опалення	Загальна
Розрахунок за ДСТУ	3135	7170	10305
Розрахунок за створеним методом	3418	8265	11683

Подібність результатів розрахунків наступна:

- За енергетичною потребою на кондиціонування – 92 %.
- За енергетичною потребою на опалення – 87 %.
- За загальною енергетичною потребою – 88 %.

Параметри для розрахунку енергетичних потреб п’ятої будівлі з класичною системою опалення наступні:

- Довжина будівлі (l) – 9 м.
- Ширина будівлі (w) – 9м.
- Висота будівлі (h) – 6 м.
- Площа північних вікон (S_n_wd) – 10 м^2.
- Площа східних вікон (S_e_wd) – 15 м^2.
- Площа західних вікон (S_w_wd) – 15 м^2.

- Площа південних вікон (S_s_wd) – 20 м^2.
- Матеріал виконання огороджувальних конструкцій – перлітобетон.
- Матеріал теплоізоляції стін, горища і підлоги по ґрунту – плити теплоізоляційні очеретяні.
- Товщина стін, горища і підлоги по ґрунту – 0,4 м.
- Товщина теплоізоляції стін, горища і підлоги по ґрунту – 0,1 м.

При виклику функції, вказані такі аргументи:

- Товщина шару стіни (B_l_w) – 0.1 м.
- Товщина шару ізоляції стіни (B_l_ins) – 0.1 м.
- Товщина шару підлоги по ґрунту (B_l_f) – 0.1 м.
- Товщина шару ізолції підлоги по ґрунту (B_f_ins) – 0.1 м.
- Товщина шару горища (B_l_r) – 0.1 м.
- Товщина шару ізоляції горища (B_r_ins) – 0.1 м.
- Кількість шарів скла (x) – 2.
- Температура увімкнення опалення (T_h) – 292, 15 К.
- Температура увімкнення охолодження (T_c) – 299,15 К.

У програмному коді, порівняння це має наступний вигляд:

```
Result_reg = (reg_build_en_cons_calc(0.1, 0.1,
                                     0.1, 0.1,
                                     0.1, 0.1,
                                     2,
                                     292.15, 299.15))
```

Результати розрахунків продемонстровані на рис. 4.6.

```
2960.757729212933 8006.068032664785 10966.825761877719
```

Рисунок 4.6 – Результати розрахунку енергетичних потреб п’ятої будівлі з класичною системою опалення

У табл. 4.5 продемонстровано порівняння результатів розрахунку енергетичних потреб п’ятої будівлі за ДСТУ Б А.2.2-12 2015 та за створеним методом.

Таблиця 4.5

Порівняння результатів розрахунку енергетичних потреб п’ятої будівлі за ДСТУ Б А.2.2-12 2015 та за створеним методом

Енергетична потреба, кВт×год	Кондиціонування	Опалення	Загальна
Розрахунок за ДСТУ	3168	10583	13751
Розрахунок за створеним методом	2960	8006	10967

Подібність результатів розрахунків наступна:

- За енергетичною потребою на кондиціонування – 93 %.
- За енергетичною потребою на опалення – 76 %.
- За загальною енергетичною потребою – 80 %.

Отже, за результатами розрахунку п’яти різних будівель, середня подібність складає:

- За енергетичною потребою на кондиціонування – 87 %.
- За енергетичною потребою на опалення – 86 %.
- За загальною енергетичною потребою – 89 %.

Такий рівень подібності дозволяє говорити, що створений метод є достатньо ефективним для використання при розрахунках енергетичних потреб будівель. Таким чином, можна зробити припущення, що він буде таким же ефективним і для розрахунків теплоенергетичних характеристик пасивних систем опалення.

4.5 Ефективність пасивної системи опалення

Далі необхідно визначити ефективність застосування пасивної системи опалення типу стіна Тромбе, для цього викликається функція оптимізації.

Для першої будівлі:

```
Result_Tr = (Optimization(0.1, 0.1,
                          0.1, 0.1,
                          0.1, 0.1,
                          2,
                          292.15, 299.15))

print(Result_Tr)
```

Результати розрахунків продемонстровані на рис. 4.7.

```
(0.12, 0.05, 0.6, 1415.8347372656276, 7203.053686446274, 8618.888423711902)
```

Рисунок 4.7 – Результати розрахунку енергетичних потреб першої будівлі з пасивною системою опалення

Отже, оптимальними параметрами стіни тромбе є наступні:

- Товщина скління зовнішнього шару стіни Тромбе – 0,12 м.
- Товщина повітряного прошарку стіни Тромбе – 0,05 м.
- Товщина масивної стіни – 0,6 м.

У табл. 4.6 продемонстровано ефективність пасивної системи опалення, для цього виконане порівняння енергетичних потреб із будівлею з класичною системою опалення.

Таблиця 4.6

Порівняння енергетичних потреб першої будівлі з класичною системою опалення та пасивною системою опалення типу стіна Тромбе

	Кондиціонування	Опалення	Загальна
Енергетична потреба будівлі з класичною системою опалення, кВт×год	2598	7950	10548
Енергетична потреба будівлі з пасивною системою опалення, кВт×год	1416	7203	8619

Продовження табл. 4.6

	Кондиціонування	Опалення	Загальна
Економія, кВт×год	1182	747	1929
Економія, %	45	9	18

Для другої будівлі:

```
Result_Tr = (Optimization(0.125, 0.1,
                          0.125, 0.1,
                          0.125, 0.1,
                          2,
                          292.15, 299.15))

print(Result_Tr)
```

Результати розрахунків продемонстровані на рис. 4.8.

```
(0.12, 0.05, 0.35, 1609.6023282307985, 7470.548043523582, 9080.15037175438)
```

Рисунок 4.8 – Результати розрахунку енергетичних потреб другої будівлі з пасивною системою опалення

Отже, оптимальними параметрами стіни тромбе є наступні:

- Товщина скління зовнішнього шару стіни Тромбе – 0,12 м.
- Товщина повітряного прошарку стіни Тромбе – 0,05 м.
- Товщина масивної стіни – 0,35 м.

У табл. 4.7 продемонстровано ефективність пасивної системи опалення, для цього виконане порівняння енергетичних потреб із будівлею з класичною системою опалення.

Таблиця 4.7

Порівняння енергетичних потреб другої будівлі з класичною системою опалення та пасивною системою опалення типу стіна Тромбе

	Кондиціонування	Опалення	Загальна
Енергетична потреба будівлі з класичною системою опалення, кВт×год	2721	8078	10798
Енергетична потреба будівлі з пасивною системою опалення, кВт×год	1610	7471	9080
Економія, кВт×год	1111	607	1718
Економія, %	41	8	16

Для третьої будівлі:

```
Result_Tr = (Optimization(0.1, 0.2,
                          0.1, 0.2,
                          0.1, 0.2,
                          2,
                          292.15, 299.15))

print(Result_Tr)
```

Результати розрахунків продемонстровані на рис. 4.9.

```
(0.12, 0.05, 0.25, 2208.4536227369435, 7868.956235086664, 10077.409857823608)
```

Рисунок 4.9 – Результати розрахунку енергетичних потреб третьої будівлі з пасивною системою опалення

Отже, оптимальними параметрами стіни тромбе є наступні:

- Товщина скління зовнішнього шару стіни Тромбе – 0,12 м.
- Товщина повітряного прошарку стіни Тромбе – 0,05 м.
- Товщина масивної стіни – 0,35 м.

У табл. 4.8 продемонстровано ефективність пасивної системи опалення, для цього виконане порівняння енергетичних потреб із будівлею з класичною системою опалення.

Таблиця 4.8

Порівняння енергетичних потреб третьої будівлі з класичною системою опалення та пасивною системою опалення типу стіна Тромбе

	Кондиціонування	Опалення	Загальна
Енергетична потреба будівлі з класичною системою опалення, кВт×год	3238	8150	11378
Енергетична потреба будівлі з пасивною системою опалення, кВт×год	2208	7869	10077
Економія, кВт×год	1030	281	1301
Економія, %	32	3	11

Для четвертої будівлі:

```
Result_Tr = (Optimization(0.05, 0.3,
                          0.05, 0.3,
                          0.05, 0.3,
                          2,
                          292.15, 299.15))

print(Result_Tr)
```

Результати розрахунків продемонстровані на рис. 4.10.

```
(0.12, 0.05, 0.45, 2603.3509725278113, 8030.475206564026, 10633.826179091837)
```

Рисунок 4.10 – Результати розрахунку енергетичних потреб четвертої будівлі з пасивною системою опалення

Отже, оптимальними параметрами стіни тромбе є наступні:

- Товщина скління зовнішнього шару стіни Тромбе – 0,12 м.

- Товщина повітряного прошарку стіни Тромбе – 0,05 м.
- Товщина масивної стіни – 0,45 м.

У табл. 4.9 продемонстровано ефективність пасивної системи опалення, для цього виконане порівняння енергетичних потреб із будівлею з класичною системою опалення.

Для п’ятої будівлі:

```
Result_Tr = (Optimization(0.1, 0.1,
                          0.1, 0.1,
                          0.1, 0.1,
                          2,
                          292.15, 299.15))

print(Result_Tr)
```

Результати розрахунків продемонстровані на рис. 4.11.

```
(0.12, 0.05, 0.25, 1907.5643028778645, 7538.978332139023, 9446.542635016887)
```

Рисунок 4.11 – Результати розрахунку енергетичних потреб п’ятої будівлі з пасивною системою опалення

Таблиця 4.9

Порівняння енергетичних потреб четвертої будівлі з класичною системою опалення та пасивною системою опалення типу стіна Тромбе

	Кондиціонування	Опалення	Загальна
Енергетична потреба будівлі з класичною системою опалення, кВт×год	3418	8265	11683
Енергетична потреба будівлі з пасивною системою опалення, кВт×год	2603	8030	10634
Економія, кВт×год	815	235	1049
Економія, %	24	3	9

Отже, оптимальними параметрами стіни тромбе є наступні:

- Товщина скління зовнішнього шару стіни Тромбе – 0,12 м.
- Товщина повітряного прошарку стіни Тромбе – 0,05 м.
- Товщина масивної стіни – 0,45 м.

У табл. 4.10 продемонстровано ефективність пасивної системи опалення, для цього виконане порівняння енергетичних потреб із будівлею з класичною системою опалення.

Таблиця 4.10

Порівняння енергетичних потреб п’ятої будівлі з класичною системою опалення та пасивною системою опалення типу стіна Тромбе

	Кондиціонування	Опалення	Загальна
Енергетична потреба будівлі з класичною системою опалення, кВт×год	2960	8006	10967
Енергетична потреба будівлі з пасивною системою опалення, кВт×год	1908	7539	9447
Економія, кВт×год	1052	467	1520
Економія, %	36	6	14

Середнє значення скорочення енергетичних потреб за рахунок використання пасивної системи опалення типу стіна Тромбе показане в табл. 4.11.

Таблиця 4.11

Середнє значення скорочення енергетичних потреб за рахунок використання пасивної системи опалення типу стіна Тромбе

	Кондиціонування	Опалення	Загальна
Економія, кВт×год	1038	467	1503
Економія, %	36	6	14

За результатами оптимізації, товщина скління зовнішнього шару стіни Тромбе обрана максимальною абсолютно для всіх будівель, що означає, що теплова ізоляція важливіша для мінімізації енергетичних потреб, ніж прозорість і максимізація нагріву внутрішньої маси будівлі від сонячного випромінювання.

Товщина повітряного прошарку обрана мінімальною для всіх будівель, адже чим менший об'єм повітря існує між стіною та склінням – тим більше він прогріється під дією сонячного випромінювання.

Товщина масивної стіни обрана максимальною тільки для першої будівлі, що знову таки каже про важливість теплової ізоляції, яку забезпечує товста стіна. Для інших будівель, це значення було обрано з середини масиву, що показує прагнення системи знайти баланс між витратами на кондиціонування та опалення, адже, чим тонша масивна стіна – тим менша енергетичнична потреба кондиціонування, чим вона товща – тим менша енергетичнична потреба опалення.

За результатами дослідження, отримано досить високу середню економію енергоресурсів за використання пасивної системи опалення типу стіна Тромбе, більше 1500 кВт×год на рік, в основному, за рахунок зниження енергетичних потреб кондиціонування, більше, ніж на 1000 кВт×год на рік, що відповідає 36 відсоткам. Витрати на опалення, у свою чергу, скорочуються більше, ніж на 450 кВт×год на рік, проте, у відсотковому еквіваленті це значення досягає всього 6 відсотків. Це означає, що в кліматі обраної місцевості, пасивна система опалення типу стіна Тромбе проявляє свою ефективність, в основному, не нагріваючи внутрішню масу будівлі, а попереджаючи перегрів будівлі в теплий період року.

4.6 Висновки до четвертого розділу

У четвертому розділі продемонстровано повний цикл роботи над комп'ютерною системою, реалізованою за допомогою мови програмування «Python». Показано реалізацію комп'ютерної моделі за допомогою мови програмування «Python». Пояснено використання та порядок виклику всіх необхідних функцій.

Доказано точність створеного методу. Виконане порівняння результатів розрахунку за створеним методом і за методом, представленим у ДСТУ Б А.2.2-12 2015. У результаті, подібність за енергетичною потребою кондиціонування склала, в середньому, 86 відсотків, за енергетичною потребою опалення – 87 відсотків, за загальною енергетичною потребою – 89 відсотків, що дозволяє вважати його ефективність достатньою для використання при розрахунках теплових та енергетичних характеристик будівлі.

Продемонстровано результати роботи програми. За результатами оптимізації, товщина скління зовнішнього шару стіни Тромбе обрана максимальною абсолютно для всіх будівель, що означає, що теплова ізоляція важливіша для мінімізації енергетичних потреб, ніж прозорість і максимізація нагріву внутрішньої маси будівлі від сонячного випромінювання. Товщина повітряного прошарку обрана мінімальною для всіх будівель, що пояснюється тим, що менший об’єм повітря існує між стіною та склінням – тим більше він прогрівється під дією сонячного випромінювання. Товщина масивної стіни обрана з середини масиву для всіх будівель, крім однієї, що демонструє прагнення системи знайти баланс між витратами на кондиціонування та опалення, адже, чим тонша масивна стіна – тим менша енергетична потреба кондиціонування, чим вона товща – тим менша енергетична потреба опалення.

За результатами дослідження, отримано досить високу середню економію енергоресурсів за використання пасивної системи опалення типу стіна Тромбе, більше 1500 кВт×год на рік, в основному, за рахунок зниження енергетичних потреб кондиціонування, більше, ніж на 1000 кВт×год на рік, що відповідає 36 відсоткам. Витрати на опалення, у свою чергу, скорочуються більше, ніж на 450 кВт×год на рік, проте, у відсотковому еквіваленті це значення досягає всього 6 відсотків. Це означає, що в кліматі обраної місцевості, пасивна система опалення типу стіна Тромбе проявляє свою ефективність, в основному, не нагріваючи внутрішню масу будівлі, а попереджаючи перегрів будівлі в теплий період року.

ВИСНОВКИ

У роботі досліджено теоретичні основи процесів теплопередачі: теплопровідності, конвекції, випромінювання, теплоємності. Для кожного з процесів, у ході роботи, створюється математично обґрунтований програмний блок. Із сукупності таких блоків утворюється система, дослідження якої є метою виконання роботи.

Житлова будівля, а також, пасивна система опалення в ній, розглядається в роботі як об’єкт моделювання та дослідження з точки зору теплових процесів, що відбуваються всередині. Представлені структурні схеми окремих складових досліджуваної системи, що складаються з блоків, кожен із яких представляє окремий процес теплопередачі.

Розглянуті методи дослідження процесів теплопередачі, як числові так і інженерні. Усі методи є потужними інструментами для моделювання та числового розв'язання різних фізичних систем, зокрема, для визначення температурних характеристик різних об’єктів. Перевага числових методів полягає в тому, що вони дають можливість глибоко досліджувати процеси всередині системи та отримувати точні значення для будь-якого відрізку часу. Але, головним чином, через свою ресурсоємність, ці методи не дуже підходять для вивчення великих об'єктів, таких як будівлі. Їх використання для цієї мети вимагає застосування обчислювальних серверів і займає багато часу. Перевагою інженерних методів є висока швидкість розрахунку та відсутність необхідності використання великих обчислювальних потужностей. Головним недоліком є те, що, найчастіше, для розрахунків використовуються середньомісячні значення основних показників, що не дає заглибитися у характеристики системи на певному, короткому відрізку часу і побачити переваги й недоліки системи при певних умовах, якими можуть стати зовнішня температура, інтенсивність сонячного випромінювання тощо.

Продемонстровано створення нового методу для розрахунку теплових та енергетичних характеристик будівлі з пасивною системою опалення. Даний метод є поєднанням числових та інженерних методів розрахунку, що дозволяє йому мати

високу швидкість та точність виконання обчислень, а також, отримати можливість дослідження системи на певному, короткому відрізку часу і побачити переваги й недоліки системи при заданих вхідних умовах. Усередині методу показана схема оптимізації параметрів пасивної системи опалення за показниками енергетичних потреб будівлі.

Показано реалізацію комп'ютерної моделі за допомогою мови програмування «Python». Пояснено використання та порядок виклику всіх необхідних функцій.

Доказано точність створеного методу. Виконане порівняння результатів розрахунку за створеним методом і за методом, представленим у ДСТУ Б А.2.2-12 2015. У результаті, подібність за енергетичною потребою кондиціонування склала, в середньому, 86 відсотків, за енергетичною потребою опалення – 87 відсотків, за загальною енергетичною потребою – 89 відсотків, що дозволяє вважати його ефективність достатньою для використання при розрахунках теплових та енергетичних характеристик будівлі.

Продемонстровано результати роботи програми. За результатами оптимізації, товщина скління зовнішнього шару стіни Тромбе обрана максимальною абсолютно для всіх будівель, що означає, що теплова ізоляція важливіша для мінімізації енергетичних потреб, ніж прозорість і максимізація нагріву внутрішньої маси будівлі від сонячного випромінювання. Товщина повітряного прошарку обрана мінімальною для всіх будівель, що пояснюється тим, що менший об'єм повітря існує між стіною та склінням – тим більше він прогрівться під дією сонячного випромінювання. Товщина масивної стіни обрана з середини масиву для всіх будівель, крім однієї, що демонструє прагнення системи знайти баланс між витратами на кондиціонування та опалення, адже, чим тонша масивна стіна – тим менша енергетична потреба кондиціонування, чим вона товща – тим менша енергетична потреба опалення.

За результатами дослідження, отримано досить високу середню економію енергоресурсів за використання пасивної системи опалення типу стіна Тромбе, більше 1500 кВт×год на рік, в основному, за рахунок зниження енергетичних

потреб кондиціонування, більше, ніж на 1000 кВт×год на рік, що відповідає 36 відсоткам. Витрати на опалення, у свою чергу, скорочуються більше, ніж на 450 кВт×год на рік, проте, у відсотковому еквіваленті це значення досягає всього 6 відсотків. Це означає, що в кліматі обраної місцевості, пасивна система опалення типу стіна Тромбе проявляє свою ефективність, в основному, не нагріваючи внутрішню масу будівлі, а попереджаючи перегрів будівлі в теплий період року.

Основним перспективним напрямком розвитку запропонованої системи, на даний момент, є створення баз даних температури та інтенсивності сонячного випромінювання місцевостей, для яких імовірні проведення розрахунків.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Y. A. Çengel, Introduction to Thermodynamics and Heat Transfer, 2nd ed. 2009, p. 865.

2. Y. A. Çengel and M. A. Boles, Thermodynamics: An Engineering Approach, 8th ed. New York, NY, USA: McGraw-Hill, 2015.

3. Y. A. Çengel and A. J. Ghajar, Heat and Mass Transfer: Fundamentals and Applications. New York, NY, USA: McGraw-Hill Education, 2015.

4. J. H. Lienhard V and J. H. Lienhard IV, A Heat Transfer Textbook, 6th ed. Cambridge, MA, USA: Phlogiston Press, 2024.

5. P. K. Pandranki, "Combined Modes of Heat Transfer in Complex Systems: A Comprehensive Review," International Journal of Scientific Research and Reviews, vol. 12, no. 4, 2023.

6. Coupled conduction and convection. – веб-сайт. URL: https://www.thermopedia.com/content/671/ (Дата звернення 17.06.2024).

7. Combined conduction and convection heat transfer in heat sinks – веб-сайт. URL: https://resources.system-analysis.cadence.com/blog/msa2022-combined-conduction-and-convection-heat-transfer-in-heat-sinks (Дата звернення 17.06.2024).

8. Coupled radiation and convection – веб-сайт. URL: https://www.thermopedia.com/content/204/ (Дата звернення 17.06.2024).

9. Coupled radiation, convection and conduction – веб-сайт. URL: https://www.thermopedia.com/content/205/ (Дата звернення 17.06.2024).

10. Coupled (combined) radiation and conduction – веб-сайт. URL: https://thermopedia.com/content/203/ (Дата звернення 17.06.2024).

11. Computer simulation – веб-сайт. URL: https://citizendium.org/wiki/Computer_simulation (Дата звернення 17.06.2024).

12. What is the difference between deterministic and stochastic models? – веб-сайт. URL: https://www.linkedin.com/advice/1/what-difference-between-deterministic-stochastic-c5ate (Дата звернення 17.06.2024).

13. What Are Dynamic Models? – веб-сайт. URL: https://assets.press.princeton.edu/chapters/s8124.pdf (Дата звернення 17.06.2024).

14. Do You Want To Know About Continuous And Discrete Modeling – веб-сайт. URL: https://controlscience.ir/basics/continuesdiscretemodels/ (Дата звернення 17.06.2024).

15. ДСТУ Б А.2.2-12 2015. Енергетична ефективність будівель. Метод розрахунку енергоспоживання при опаленні, охолодженні, вентиляції, освітленні та гарячому водопостачанні. ДП «Державний науково-дослідний інститут будівельних конструкцій»; ТК 302 «Енергоефективність будівель і споруд», ПК4 «Енергетична паспортизація будівель», Київ, 2016, 145 с.

16. Energy efficiency in buildings – веб-сайт. URL: https://www.unido.org/sites/default/files/2009-02/Module18_0.pdf (Дата звернення 17.06.2024).

17. Energy balance - new methodology – веб-сайт. URL: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Energy_balance_-_new_methodology#What_is_an_energy_balance.3F (Дата звернення 17.06.2024).

18. F. P. Incropera, D. P. DeWitt, et al., Introduction to Heat Transfer, 5th ed. Hoboken, NJ, USA: Wiley, 2007.

19. J. P. Holman, Heat Transfer, 10th ed. New York, NY, USA: McGraw-Hill, 2010.

20. Y. A. Çengel and A. J. Ghajar, Heat and Mass Transfer: Fundamentals and Applications, 5th ed. New York, NY, USA: McGraw-Hill Professional, 2014.

21. J. H. Lienhard, A Heat Transfer Textbook. Englewood Cliffs, NJ, USA: Prentice-Hall, 1987.

22. A. Bejan and A. D. Kraus, Heat Transfer Handbook. Hoboken, NJ, USA: John Wiley & Sons, 2003.

23. J. A. Duffie and W. A. Beckman, Solar Engineering of Thermal Processes, 4th ed. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2013, p. 910.

24. F. Manzano-Agugliaro, F. G. Montoya, A. Sabio-Ortega, and A. García-Cruz, "Review of Bioclimatic Architecture Strategies for Achieving Thermal Comfort," Renewable and Sustainable Energy Reviews, vol. 49, pp. 736-755, 2015.

25. B. Widera, "Bioclimatic Architecture," Journal of Civil Engineering and Architecture Research, vol. 4, pp. 567-578, 2015.

26. U. A. Mohammed and H. Z. Alibaba, "Application of Bioclimatic Design Strategies to Solve Thermal Discomfort in Maiduguri Residences, Borno State Nigeria," Imperial Journal of Interdisciplinary Research, vol. 1, pp. 227-233, 2018.

27. A. T. Adekunle and M. Nikolopoulou, "Thermal Comfort, Summertime Temperatures and Overheating in Prefabricated Timber Housing," Building and Environment, vol. 103, pp. 21-35, 2016.

28. Numerical Analysis – веб-сайт. URL: https://www.vedantu.com/maths/numerical-analysis (Дата звернення 17.06.2024).

29. The Application of Numerical Methods in Heat Transfer Problems – веб-сайт. URL: https://resources.system-analysis.cadence.com/thermal/msa2022-the-application-of-numerical-methods-in-heat-transfer-problems (Дата звернення 17.06.2024).

30. Numerical heat transfer – веб-сайт. URL: https://www.thermopedia.com/content/991/ (Дата звернення 17.06.2024).

31. How can you choose the best numerical method for heat transfer simulations? – веб-сайт. URL: https://www.linkedin.com/advice/3/how-can-you-choose-best-numerical-method-tcgse (Дата звернення 17.06.2024).

32. The Application of Numerical Methods in Heat Transfer Problems – веб-сайт. URL: https://resources.system-analysis.cadence.com/blog/msa2022-the-application-of-numerical-methods-in-heat-transfer-problems (Дата звернення 17.06.2024).

33. J. Y. Murthy and S. R. Mathur, Numerical Methods in Heat, Mass, and Momentum Transfer. School of Mechanical Engineering, Purdue University, 2002.

34. D. Nance, Finite Volume Algorithms for Heat Conduction, 2010.

35. D. Nance, "Finite Volume Solution of the Heat Conduction Equation - An Application in 3D," 2020, doi: 10.13140/RG.2.2.29422.84807.

36. Finite difference methods for diffusion processes. – веб-сайт. URL: http://hplgit.github.io/num-methods-for-PDEs/doc/pub/diffu/sphinx/index.html (Дата звернення 17.06.2024).

37. G. W. Recktenwald, "Finite-Difference Approximations to the Heat Equation," Mechanical Engineering, no. 1, Portland State University, 2004, pp. 27.

38. A Gallery of finite element solvers – веб-сайт. URL: https://fenicsproject.org/pub/tutorial/html/._ftut1006.html (Дата звернення 17.06.2024).

39. Finite Difference Method – веб-сайт. URL: https://pythonnumericalmethods.studentorg.berkeley.edu/notebooks/chapter23.03-Finite-Difference-Method.html (Дата звернення 17.06.2024).

40. G. Nervadof. (2020). Solving 2D Heat Equation Numerically using Python – веб-сайт. URL: https://levelup.gitconnected.com/solving-2d-heat-equation-numerically-using-python-3334004aa01a.

41. Yoann Mocquin. (2022). 300-Times Faster Resolution of Finite-Difference Method Using NumPy – веб-сайт. URL: https://towardsdatascience.com/300-times-faster-resolution-of-finite-difference-method-using-numpy-de28cdade4e1 (Дата звернення 17.06.2024).

42. M. Bern and P. Plassmann, "Mesh Generation," in Handbook of Computational Geometry, J.-R. Sack and J. Urrutia, Eds. North-Holland, 2000, pp. 291-332, doi: 10.1016/B978-044482537-7/50007-3.

43. J. Tu, G.-H. Yeoh, and C. Liu, CFD Mesh Generation: A Practical Guideline, in Computational Fluid Dynamics, J. Tu, G.-H. Yeoh, and C. Liu, Eds. Butterworth-Heinemann, 2018, pp. 125-154, doi: 10.1016/B978-0-08-101127-0.00004-0.

44. L. Su, "A Radial Basis Function (RBF)-Finite Difference (FD) Method for the Backward Heat Conduction Problem," Applied Mathematics and Computation, vol. 354, pp. 232-247, 2019, doi: 10.1016/j.amc.2019.02.035.

45. A. Kværnø, Partial Differential Equations and Finite Difference Methods, 2020.

46. J. Parkinson, Solving Partial Differential Equations Using the Finite Difference Method and the Fourier Spectral Method, 2022.

47. V. John, Finite Difference Methods (FDM).

48. A. Powell, Finite Difference Solution of the Heat Equation, 2002.

49. R. Eymard, T. Gallouët, and R. Herbin, "Finite Volume Methods," in Handbook of Numerical Analysis, vol. 7, J. L. Lions and P. Ciarlet, Eds. Elsevier, 2000, pp. 713-1020.

50. L. Chen, Finite Volume Methods, 2011.

51. A. Outhwaite, "Numerical Solutions for 1D Conduction Using the Finite Volume Method," 2017, doi: 10.13140/RG.2.2.32005.45287.

52. R. Morasata and K. M. Güleren, "Solution of the Two-Dimensional Steady State Heat Conduction Using the Finite Volume Method," 2016.

53. S. C. Mishra and H. K. Roy, "Solving Transient Conduction and Radiation Heat Transfer Problems Using the Lattice Boltzmann Method and the Finite Volume Method," Journal of Computational Physics, vol. 223, no. 1, pp. 89-107, 2007, doi: 10.1016/j.jcp.2006.08.021.

54. O. Akeremale, O. A. Olaiju, and Y. Hoe, "Finite Element Method for Heat Transfer Phenomenon on a Closed Rectangular Plate," Eureka: Physics and Engineering, 2020.

55. C. Sert, Formulation of FEM for One-Dimensional Problems.

56. B. Mebrate, "Numerical Solution of a One Dimensional Heat Equation with Dirichlet Boundary Conditions," American Journal of Applied Mathematics, vol. 3, no. 6, p. 305, 2015, doi: 10.11648/j.ajam.20150306.20.

57. M. Asadzadeh, An Introduction to the Finite Element Method (FEM) for Differential Equations, 2014.

58. Finite element basis functions – веб-сайт. URL: https://hplgit.github.io/INF5620/doc/pub/H14/fem/html/._main_fem004.html#fem:approx:fe (Дата звернення 17.06.2024).

59. L. Wald, Basics in Solar Radiation at Earth Surface - Revised Version #2, 2019.

60. ДБН В.1.2-2:2006. Навантаження і Впливи. Норми Проектування. ВАТ Укрндіпроектстальконструкція ім. В.М. Шимановського, Київ, 2007, 75 с.

ДОДАТОК А

Список опублікованих праць за темою дисертації

Публікації в наукових фахових виданнях України:

1. Андронова О.В., Курак В.В., Сокол К.І. Тепловий режим будівлі з пасивною системою опалення.О.В. *Вісник Херсонського національного технічного університету. Інженерні науки*. № 1(72), Ч. 1. ХНТУ. 2020. с. 9-17.
URL: https://journals.kntu.net.ua/index.php/visnyk/article/view/467/493
DOI: https://doi.org/10.35546/kntu2078-4481.2020.1.1.1

2. Сокол К.І., Огнєва О.Є. Розробка системи комп'ютерного моделювання для оптимізації параметрів пасивної системи опалення закритого типу. *Проблеми інформаційних технологій*. 2020. № 27. С. 69-77.
URL: https://journals.kntu.net.ua/index.php/pit/article/view/705
DOI: https://doi.org/10.35546/2313-0687.2020.27.69-77

3. Сокол К.І., Огнєва О.Є., Андронова О.В. Проектування комп’ютерної системи для дослідження теплового режиму будівлі з пасивною системою опалення закритого типу. *Вісник Херсонського національного технічного університету. Інформаційні технології*. № 1(80). ХНТУ. 2022. с. 61-67.
URL: https://ojs.kntu.net.ua/index.php/visnyk/article/view/236
DOI: https://doi.org/10.35546/kntu2078-4481.2022.1

4. Сокол К.І. Програмна реалізація методу розрахунку енергопотреб будівлі. *Вісник Хмельницького національного університету. Серія: технічні науки.* № 335(3.1). ХНУ. 2024. с. 209-215.
URL: https://heraldts.khmnu.edu.ua/index.php/heraldts/article/view/229
DOI: https://doi.org/10.31891/2307-5732-2024-335-3-29

Матеріали і тези Всеукраїнських наукових та науково-практичних конференцій:

1. Сокол К.І., Огнєва О.Є. Використання SMART GRIDS у системи опалення будівель / К.І. Сокол, О.Є. Огнєва // Матеріали всеукраїнської науково-технічної

конференції «Інформаційні технології та програмування», 30.11.2020р. – Херсон:ХНТУ, 2020. – с.86-87.

2. Сокол К.І., Огнєва О.Є. Проектування комп'ютерної системи для імітації роботи та дослідження ефективності пасивної системи опалення / К.І. Сокол, О.Є. Огнєва // Матеріали ІІ Всеукраїнської науково-практичної конференції молодих вчених, аспірантів і студентів «Молодь і наука», 24.05.2021р.- Херсон:ХНТУ, 2020.- с.12-13.

3. Сокол К., Огнєва О. Проектування додатку для розрахунку показників моделі будівлі з пасивною системою опалення закритого типу / К. Сокол., О. Огнєва // Матеріали Всеукраїнської науково-технічна конференція «Інформаційні технології та програмування», 30.11.2021. – Херсон: ХНТУ, 2021.

Матеріали і тези Міжнародних наукових та науково-практичних конференцій:

1. Sokol K. Ohnieva O. Development of the structural scheme for passive closed type passive heating system's parameters optimization / Інтелектуальні системи прийняття рішень і проблеми обчислювального інтелекту: Матеріали міжнар. наук. конф., с. Залізний Порт, 25-29 травня 2021 р. – Херсон: Видавництво ФОП Вишемирський В. С., 2021. – с.111-113.

2. Сокол К., Огнєва О. Концепція додатку для проектування будівель з пасивними системами опалення / К. Сокол., О. Огнєва // Матеріали міжнародної науково-практичної конференції «Синергія науки і бізнесу у повоєнному відновленні Херсонщини», 26-28.04.23. Видавництво Олді+., 2023. – с. 207-211.

3. Sokol K., Ohnieva O., Dostalek L. Determining the effectiveness of using passive closed-loop heating systems to prevent the risk of building overcooling during energy system stability issues.. International Workshop on Computational & Information Technologies for Risk-Informed Systems, December 21-22, 2023.

ДОДАТОК Б

Лістинг коду програми

```
# temperature per hour data

TemperaturePerHour_data =
r"D:\Postgraduate\Postgraduate\Disertation\FinalVersion\TemperaturePerHour.xlsx"
TemperaturePerHour = pd.read_excel(TemperaturePerHour_data)
T_street = TemperaturePerHour.iloc[:, 1]

# soil temperature per hour data

SoilTemperaturePerHour_data =
r"D:\Postgraduate\Postgraduate\Disertation\FinalVersion\SoilTemperaturePerHour.xls
x"
SoilTemperaturePerHour = pd.read_excel(SoilTemperaturePerHour_data)
T_soil = SoilTemperaturePerHour.iloc[:, 1]

# north solar irradiation

SolarIrradiationNorth_data =
r"D:\Postgraduate\Postgraduate\Disertation\FinalVersion\SolarIrradiation\SolarIrra
diationNorth.xlsx"
SolarIrradiationNorth = pd.read_excel(SolarIrradiationNorth_data)
I_n = SolarIrradiationNorth.iloc[:, 1]

SolarIrradiationEast_data =
r"D:\Postgraduate\Postgraduate\Disertation\FinalVersion\SolarIrradiation\SolarIrra
diationEast.xlsx" # input data for east wall
SolarIrradiationEast = pd.read_excel(SolarIrradiationEast_data)
I_e = SolarIrradiationEast.iloc[:, 1]

SolarIrradiationWest_data =
r"D:\Postgraduate\Postgraduate\Disertation\FinalVersion\SolarIrradiation\SolarIrra
diationWest.xlsx" # input data for west wall
SolarIrradiationWest = pd.read_excel(SolarIrradiationWest_data)
I_w = SolarIrradiationWest.iloc[:, 1]

SolarIrradiationRoof_data =
r"D:\Postgraduate\Postgraduate\Disertation\FinalVersion\SolarIrradiation\SolarIrra
diationRoof.xlsx" # input data for roof
SolarIrradiationRoof = pd.read_excel(SolarIrradiationRoof_data)
I_r = SolarIrradiationRoof.iloc[:, 1]

SolarIrradiationSouth_data =
r"D:\Postgraduate\Postgraduate\Disertation\FinalVersion\SolarIrradiation\SolarIrra
diationSouth.xlsx" # input data for Trombe wall
SolarIrradiationSouth = pd.read_excel(SolarIrradiationSouth_data)
I_s = SolarIrradiationSouth.iloc[:, 1]

# north wall calculations

def north_wall_temp_calc(B_ins_w, B_l_w):

        I_rad = list() # solar irradiation
        I_rad = I_n # solar irradiation
        S_w = l * h - S_n_wd # north wall area

        T_w_out = list() # outside temperaure (K)
        # initial outside temperature (K)
```

```
        T_w_out.insert(0, (T_street[0] - (((T_street[0])**4)*(R_0-R_out))**(1/4)))
        T_w_irr = list() # irradiation temperature
        T_w_out_w = list() # out wall temperature including radiation and
convection effects (K)
        s_w_out = list() # out wall convection (W)

        T_w_1layer = list() # first layer temperature (K)
        l_w_1layer = list() # first layer thermal conductivity (W)
        C_w_1layer = list() # first layer heat capacity

        T_w_2layer = list() # second layer temperature (K)
        l_w_2layer = list() # second layer thermal conductivity (W)
        C_w_2layer = list() # second layer heat capacity

        T_w_3layer = list() # third layer temperature (K)
        l_w_3layer = list() # third layer thermal conductivity (W)
        C_w_3layer = list() # third layer heat capacity

        T_w_4layer = list() # fourth layer temperature (K)
        l_w_4layer = list() # fourth layer thermal conductivity (W)
        C_w_4layer = list() # fourth layer heat capacity

        T_w_5layer = list() # fifth layer temperature (K)
        l_w_5layer = list() # fifth layer thermal conductivity (W)
        C_w_5layer = list() # fifth layer heat capacity

        T_w_in = list() # inside building temperature (K)
        s_w_in = list() # inside building convection (W)
        I_w_in = list() # inside wall radiaion (W)
        C_w_in = list() # inside building capacity (J)

        T_w_in_b = list() # inside building specific temperature (K)

        # first layer initial temperature (K)
        T_w_1layer.insert(0, T_w_out[0] + 5)
        # second layer initial temperature (K)
        T_w_2layer.insert(0, T_w_1layer[0] + 4)
        # second layer initial temperature (K)
        T_w_3layer.insert(0, T_w_2layer[0] + 3)
        # second layer initial temperature (K)
        T_w_4layer.insert(0, T_w_3layer[0] + 2)
        # second layer initial temperature (K)
        T_w_5layer.insert(0, T_w_4layer[0] + 1)
        # inside building initial temperature (K)
        T_w_in.insert(0, T_w_5layer[0] - k_air)

        for i in range(0, n+7): # out wall calculations
            s_w_out.insert(i, s_out * S_w * (T_street[i] - T_w_out[i]))
            T_w_out.insert(i+1, (T_street[i+1] - s_w_out[i] / (s_out * S_w)))
            T_w_irr.insert(i, ((T_w_out[i]**4 + (I_rad[i] * S_w / 3600) / (S_w *
R_0))**(1/4)))
            T_w_out_w.insert(i, (T_w_out[i] * (abs(s_w_out[i]) / (abs(s_w_out[i])
+ abs((I_rad[i] * S_w / 3600)))) + T_w_irr[i] * (abs((I_rad[i] * S_w / 3600)) /
(abs(s_w_out[i]) + abs((I_rad[i] * S_w / 3600))))))

        for i in range(0, n+6): # first layer calculations
            l_w_1layer.insert(i, L_ins * S_w * (T_w_out_w[i] - T_w_1layer[i]) /
B_ins_w)
            C_w_1layer.insert(i, (C_ins * r_ins * B_ins_w * S_w * (T_w_out_w[i] -
T_w_1layer[i])) / 3600)
            T_w_1layer.insert(i+1, ((T_w_out_w[i] - ((l_w_1layer [i] * B_ins_w) /
(L_ins * S_w))) * (abs(l_w_1layer[i]) / (2 * abs(l_w_1layer[i]) + 2 *
abs(C_w_1layer[i])))) + ((T_w_out_w[i+1] - ((C_w_1layer[i] * 3600)/ (C_ins * r_ins
```

```
* B_l * S_w))) * (abs(C_w_1layer[i]) / (2 * abs(C_w_1layer[i]) + 2 *
abs(l_w_1layer[i])))) + ((T_w_out_w[i+1] - k_ins) * (abs(l_w_1layer[i]) / (2 *
abs(C_w_1layer[i]) + 2 * abs(l_w_1layer[i])))) + ((T_w_1layer[i] - k_ins) *
(abs(C_w_1layer[i]) / (2 * abs(C_w_1layer[i]) + 2 * abs(l_w_1layer[i]))))))

        for i in range(0, n+5): # second layer calculations
            l_w_2layer.insert(i, L_w * S_w * (T_w_1layer[i] - T_w_2layer[i]) /
B_l_w)
            C_w_2layer.insert(i, (C_w * r_w * B_l_w * S_w * (T_w_1layer[i] -
T_w_2layer[i])) / 3600)
            T_w_2layer.insert(i+1, ((T_w_1layer[i] - ((l_w_2layer [i] * B_l_w) /
(L_w * S_w))) * (abs(l_w_2layer[i]) / (2 * abs(l_w_2layer[i]) + 2 *
abs(C_w_2layer[i])))) + ((T_w_1layer[i+1] - ((C_w_2layer[i] * 3600)/ (C_w * r_w *
B_l * S_w))) * (abs(C_w_2layer[i]) / (2 * abs(C_w_2layer[i]) + 2 *
abs(l_w_2layer[i])))) + ((T_w_1layer[i+1] - k_w) * (abs(l_w_2layer[i]) / (2 *
abs(C_w_2layer[i]) + 2 * abs(l_w_2layer[i])))) + ((T_w_2layer[i] - k_w) *
(abs(C_w_2layer[i]) / (2 * abs(C_w_2layer[i]) + 2 * abs(l_w_2layer[i]))))))

        for i in range(0, n+4): # third layer calculations
            l_w_3layer.insert(i, L_w * S_w * (T_w_2layer[i] - T_w_3layer[i]) /
B_l_w)
            C_w_3layer.insert(i, (C_w * r_w * B_l_w * S_w * (T_w_2layer[i] -
T_w_3layer[i])) / 3600)
            T_w_3layer.insert(i+1, ((T_w_2layer[i] - ((l_w_3layer [i] * B_l_w) /
(L_w * S_w))) * (abs(l_w_3layer[i]) / (2 * abs(l_w_3layer[i]) + 2 *
abs(C_w_3layer[i])))) + ((T_w_2layer[i+1] - ((C_w_3layer[i] * 3600)/ (C_w * r_w *
B_l * S_w))) * (abs(C_w_3layer[i]) / (2 * abs(C_w_3layer[i]) + 2 *
abs(l_w_3layer[i])))) + ((T_w_2layer[i+1] - k_w) * (abs(l_w_3layer[i]) / (2 *
abs(C_w_3layer[i]) + 2 * abs(l_w_3layer[i])))) + ((T_w_3layer[i] - k_w) *
(abs(C_w_3layer[i]) / (2 * abs(C_w_3layer[i]) + 2 * abs(l_w_3layer[i]))))))

        for i in range(0, n+3): # fourth layer calculations
            l_w_4layer.insert(i, L_w * S_w * (T_w_3layer[i] - T_w_4layer[i]) /
B_l_w)
            C_w_4layer.insert(i, (C_w * r_w * B_l_w * S_w * (T_w_3layer[i] -
T_w_4layer[i])) / 3600)
            T_w_4layer.insert(i+1, ((T_w_3layer[i] - ((l_w_4layer [i] * B_l_w) /
(L_w * S_w))) * (abs(l_w_4layer[i]) / (2 * abs(l_w_4layer[i]) + 2 *
abs(C_w_4layer[i])))) + ((T_w_3layer[i+1] - ((C_w_4layer[i] * 3600)/ (C_w * r_w *
B_l * S_w))) * (abs(C_w_4layer[i]) / (2 * abs(C_w_4layer[i]) + 2 *
abs(l_w_4layer[i])))) + ((T_w_3layer[i+1] - k_w) * (abs(l_w_4layer[i]) / (2 *
abs(C_w_4layer[i]) + 2 * abs(l_w_4layer[i])))) + ((T_w_4layer[i] - k_w) *
(abs(C_w_4layer[i]) / (2 * abs(C_w_4layer[i]) + 2 * abs(l_w_4layer[i]))))))

        for i in range(0, n+2): # fifth layer calculations
            l_w_5layer.insert(i, L_w * S_w * (T_w_4layer[i] - T_w_5layer[i]) /
B_l_w)
            C_w_5layer.insert(i, (C_w * r_w * B_l_w * S_w * (T_w_4layer[i] -
T_w_5layer[i])) / 3600)
            T_w_5layer.insert(i+1, ((T_w_4layer[i] - ((l_w_5layer [i] * B_l_w) /
(L_w * S_w))) * (abs(l_w_5layer[i]) / (2 * abs(l_w_5layer[i]) + 2 *
abs(C_w_5layer[i])))) + ((T_w_4layer[i+1] - ((C_w_5layer[i] * 3600)/ (C_w * r_w *
B_l * S_w))) * (abs(C_w_5layer[i]) / (2 * abs(C_w_5layer[i]) + 2 *
abs(l_w_5layer[i])))) + ((T_w_4layer[i+1] - k_w) * (abs(l_w_5layer[i]) / (2 *
abs(C_w_5layer[i]) + 2 * abs(l_w_5layer[i])))) + ((T_w_5layer[i] - k_w) *
(abs(C_w_5layer[i]) / (2 * abs(C_w_5layer[i]) + 2 * abs(l_w_5layer[i]))))))

        for i in range(0, n+1): # inside building calculations
            s_w_in.insert(i, s_in_w * S_w * (T_w_5layer[i] - T_w_in[i]))
            I_w_in.insert(i, (T_w_5layer[i]**4 - T_w_in[i]**4) * S_w * R_in)
            C_w_in.insert(i, (C_w * r_w * B_in * S_in * (T_w_5layer[i] -
T_w_in[i])) / 3600 + ((C_air * r_air * l * w * h) * (T_w_5layer[i] - T_w_in[i])) /
3600)
```

```
            T_w_in.insert(i+1, (T_w_5layer[i+1] - s_w_in[i] / (s_in_w * S_w)) *
(abs(s_w_in[i]) / (abs(s_w_in[i]) + abs(C_w_in[i]) + abs(I_w_in[i]))) +
(T_w_5layer[i+1] - (C_w_in[i] * 3600) / (C_w * r_w * B_in * S_in + C_air * r_air *
l * w * h)) * (abs(C_w_in[i]) / (abs(s_w_in[i]) + abs(C_w_in[i]) +
abs(I_w_in[i]))) + (((T_w_5layer[i+1])**4 - I_w_in[i] / (S_w * R_in))**(1/4)) *
(abs(I_w_in[i]) / (abs(s_w_in[i]) + abs(C_w_in[i]) + abs(I_w_in[i]))))

        for i in range(0, n): # inside building specific temperature calculations
            T_w_in_b.insert(i, T_w_in[i] * S_w / S_in_b)

        return(T_w_in_b)

# east wall calculations

def east_wall_temp_calc(B_ins_w, B_l_w):

        I_rad = list() # solar irradiation

        T_w_out = list() # outside temperaure (K)
        T_w_out.insert(0, (T_street[0]-(((T_street[0])**4)*(R_0-R_out))**(1/4))) #
initial outside temperature (K)
        T_w_irr = list() # irradiation temperature
        T_w_out_w = list() # out wall temperature including radiation and
convection effects (K)
        s_w_out = list() # out wall convection (W)

        T_w_1layer = list() # first layer temperature (K)
        l_w_1layer = list() # first layer thermal conductivity (W)
        C_w_1layer = list() # first layer heat capacity

        T_w_2layer = list() # second layer temperature (K)
        l_w_2layer = list() # second layer thermal conductivity (W)
        C_w_2layer = list() # second layer heat capacity

        T_w_3layer = list() # third layer temperature (K)
        l_w_3layer = list() # third layer thermal conductivity (W)
        C_w_3layer = list() # third layer heat capacity

        T_w_4layer = list() # fourth layer temperature (K)
        l_w_4layer = list() # fourth layer thermal conductivity (W)
        C_w_4layer = list() # fourth layer heat capacity

        T_w_5layer = list() # fifth layer temperature (K)
        l_w_5layer = list() # fifth layer thermal conductivity (W)
        C_w_5layer = list() # fifth layer heat capacity

        T_w_in = list() # inside building temperature (K)
        s_w_in = list() # inside building convection (W)
        I_w_in = list() # inside wall radiaion (W)
        C_w_in = list() # inside building capacity (J)

        T_w_in_b = list() # inside building specific temperature (K)

        T_w_1layer.insert(0, T_w_out[0] + 5) # first layer initial temperature (K)
        T_w_2layer.insert(0, T_w_1layer[0] + 4) # second layer initial temperature
(K)
        T_w_3layer.insert(0, T_w_2layer[0] + 3) # second layer initial temperature
(K)
        T_w_4layer.insert(0, T_w_3layer[0] + 2) # second layer initial temperature
(K)
        T_w_5layer.insert(0, T_w_4layer[0] + 1) # second layer initial temperature
(K)
        T_w_in.insert(0, T_w_5layer[0] - k_air) # inside building initial
```

```
temperature (K)

        S_w = w * h - S_e_wd # east wall area
        I_rad = I_e # solar irradiation

        for i in range(0, n+7): # out wall calculations
            s_w_out.insert(i, s_out * S_w * (T_street[i] - T_w_out[i]))
            T_w_out.insert(i+1, (T_street[i+1] - s_w_out[i] / (s_out * S_w)))
            T_w_irr.insert(i, ((T_w_out[i]**4 + (I_rad[i] * S_w / 3600) / (S_w *
R_0))**(1/4)))
            T_w_out_w.insert(i, (T_w_out[i] * (abs(s_w_out[i]) / (abs(s_w_out[i])
+ abs((I_rad[i] * S_w / 3600)))) + T_w_irr[i] * (abs((I_rad[i] * S_w / 3600)) /
(abs(s_w_out[i]) + abs((I_rad[i] * S_w / 3600))))))

        for i in range(0, n+6): # first layer calculations
            l_w_1layer.insert(i, L_ins * S_w * (T_w_out_w[i] - T_w_1layer[i]) /
B_ins_w)
            C_w_1layer.insert(i, (C_ins * r_ins * B_ins_w * S_w * (T_w_out_w[i] -
T_w_1layer[i])) / 3600)
            T_w_1layer.insert(i+1, ((T_w_out_w[i] - ((l_w_1layer [i] * B_ins_w) /
(L_ins * S_w))) * (abs(l_w_1layer[i]) / (2 * abs(l_w_1layer[i]) + 2 *
abs(C_w_1layer[i])))) + ((T_w_out_w[i+1] - ((C_w_1layer[i] * 3600)/ (C_ins * r_ins
* B_l * S_w))) * (abs(C_w_1layer[i]) / (2 * abs(C_w_1layer[i]) + 2 *
abs(l_w_1layer[i])))) + ((T_w_out_w[i+1] - k_ins) * (abs(l_w_1layer[i]) / (2 *
abs(C_w_1layer[i]) + 2 * abs(l_w_1layer[i])))) + ((T_w_1layer[i] - k_ins) *
(abs(C_w_1layer[i]) / (2 * abs(C_w_1layer[i]) + 2 * abs(l_w_1layer[i])))))

        for i in range(0, n+5): # second layer calculations
            l_w_2layer.insert(i, L_w * S_w * (T_w_1layer[i] - T_w_2layer[i]) /
B_l_w)
            C_w_2layer.insert(i, (C_w * r_w * B_l_w * S_w * (T_w_1layer[i] -
T_w_2layer[i])) / 3600)
            T_w_2layer.insert(i+1, ((T_w_1layer[i] - ((l_w_2layer [i] * B_l_w) /
(L_w * S_w))) * (abs(l_w_2layer[i]) / (2 * abs(l_w_2layer[i]) + 2 *
abs(C_w_2layer[i])))) + ((T_w_1layer[i+1] - ((C_w_2layer[i] * 3600)/ (C_w * r_w *
B_l * S_w))) * (abs(C_w_2layer[i]) / (2 * abs(C_w_2layer[i]) + 2 *
abs(l_w_2layer[i])))) + ((T_w_1layer[i+1] - k_w) * (abs(l_w_2layer[i]) / (2 *
abs(C_w_2layer[i]) + 2 * abs(l_w_2layer[i])))) + ((T_w_2layer[i] - k_w) *
(abs(C_w_2layer[i]) / (2 * abs(C_w_2layer[i]) + 2 * abs(l_w_2layer[i])))))

        for i in range(0, n+4): # third layer calculations
            l_w_3layer.insert(i, L_w * S_w * (T_w_2layer[i] - T_w_3layer[i]) /
B_l_w)
            C_w_3layer.insert(i, (C_w * r_w * B_l_w * S_w * (T_w_2layer[i] -
T_w_3layer[i])) / 3600)
            T_w_3layer.insert(i+1, ((T_w_2layer[i] - ((l_w_3layer [i] * B_l_w) /
(L_w * S_w))) * (abs(l_w_3layer[i]) / (2 * abs(l_w_3layer[i]) + 2 *
abs(C_w_3layer[i])))) + ((T_w_2layer[i+1] - ((C_w_3layer[i] * 3600)/ (C_w * r_w *
B_l * S_w))) * (abs(C_w_3layer[i]) / (2 * abs(C_w_3layer[i]) + 2 *
abs(l_w_3layer[i])))) + ((T_w_2layer[i+1] - k_w) * (abs(l_w_3layer[i]) / (2 *
abs(C_w_3layer[i]) + 2 * abs(l_w_3layer[i])))) + ((T_w_3layer[i] - k_w) *
(abs(C_w_3layer[i]) / (2 * abs(C_w_3layer[i]) + 2 * abs(l_w_3layer[i])))))

        for i in range(0, n+3): # fourth layer calculations
            l_w_4layer.insert(i, L_w * S_w * (T_w_3layer[i] - T_w_4layer[i]) /
B_l_w)
            C_w_4layer.insert(i, (C_w * r_w * B_l_w * S_w * (T_w_3layer[i] -
T_w_4layer[i])) / 3600)
            T_w_4layer.insert(i+1, ((T_w_3layer[i] - ((l_w_4layer [i] * B_l_w) /
(L_w * S_w))) * (abs(l_w_4layer[i]) / (2 * abs(l_w_4layer[i]) + 2 *
abs(C_w_4layer[i])))) + ((T_w_3layer[i+1] - ((C_w_4layer[i] * 3600)/ (C_w * r_w *
B_l * S_w))) * (abs(C_w_4layer[i]) / (2 * abs(C_w_4layer[i]) + 2 *
abs(l_w_4layer[i])))) + ((T_w_3layer[i+1] - k_w) * (abs(l_w_4layer[i]) / (2 *
```

```
abs(C_w_4layer[i]) + 2 * abs(l_w_4layer[i])))) + ((T_w_4layer[i] - k_w) *
(abs(C_w_4layer[i]) / (2 * abs(C_w_4layer[i]) + 2 * abs(l_w_4layer[i])))))

        for i in range(0, n+2): # fifth layer calculations
            l_w_5layer.insert(i, L_w * S_w * (T_w_4layer[i] - T_w_5layer[i]) /
B_l_w)
            C_w_5layer.insert(i, (C_w * r_w * B_l_w * S_w * (T_w_4layer[i] -
T_w_5layer[i])) / 3600)
            T_w_5layer.insert(i+1, ((T_w_4layer[i] - ((l_w_5layer [i] * B_l_w) /
(L_w * S_w))) * (abs(l_w_5layer[i]) / (2 * abs(l_w_5layer[i]) + 2 *
abs(C_w_5layer[i])))) + ((T_w_4layer[i+1] - ((C_w_5layer[i] * 3600)/ (C_w * r_w *
B_l * S_w))) * (abs(C_w_5layer[i]) / (2 * abs(C_w_5layer[i]) + 2 *
abs(l_w_5layer[i])))) + ((T_w_4layer[i+1] - k_w) * (abs(l_w_5layer[i]) / (2 *
abs(C_w_5layer[i]) + 2 * abs(l_w_5layer[i])))) + ((T_w_5layer[i] - k_w) *
(abs(C_w_5layer[i]) / (2 * abs(C_w_5layer[i]) + 2 * abs(l_w_5layer[i])))))

        for i in range(0, n+1): # inside building calculations
            s_w_in.insert(i, s_in_w * S_w * (T_w_5layer[i] - T_w_in[i]))
            I_w_in.insert(i, (T_w_5layer[i]**4 - T_w_in[i]**4) * S_w * R_in)
            C_w_in.insert(i, (C_w * r_w * B_in * S_in * (T_w_5layer[i] -
T_w_in[i])) / 3600 + ((C_air * r_air * l * w * h) * (T_w_5layer[i] - T_w_in[i])) /
3600)
            T_w_in.insert(i+1, (T_w_5layer[i+1] - s_w_in[i] / (s_in_w * S_w)) *
(abs(s_w_in[i]) / (abs(s_w_in[i]) + abs(C_w_in[i]) + abs(I_w_in[i]))) +
(T_w_5layer[i+1] - (C_w_in[i] * 3600) / (C_w * r_w * B_in * S_in + C_air * r_air *
l * w * h)) * (abs(C_w_in[i]) / (abs(s_w_in[i]) + abs(C_w_in[i]) +
abs(I_w_in[i]))) + (((T_w_5layer[i+1])**4 - I_w_in[i] / (S_w * R_in))**(1/4)) *
(abs(I_w_in[i]) / (abs(s_w_in[i]) + abs(C_w_in[i]) + abs(I_w_in[i]))))

        for i in range(0, n): # inside building specific temperature calculations
            T_w_in_b.insert(i, T_w_in[i] * S_w / S_in_b)

        return(T_w_in_b)

# west wall calculations

def west_wall_temp_calc(B_ins_w, B_l_w):

        I_rad = list() # solar irradiation

        T_w_out = list() # outside temperaure (K)
        T_w_out.insert(0, (T_street[0]-(((T_street[0])**4)*(R_0-R_out))**(1/4))) #
initial outside temperature (K)
        T_w_irr = list() # irradiation temperature
        T_w_out_w = list() # out wall temperature including radiation and
convection effects (K)
        s_w_out = list() # out wall convection (W)

        T_w_1layer = list() # first layer temperature (K)
        l_w_1layer = list() # first layer thermal conductivity (W)
        C_w_1layer = list() # first layer heat capacity

        T_w_2layer = list() # second layer temperature (K)
        l_w_2layer = list() # second layer thermal conductivity (W)
        C_w_2layer = list() # second layer heat capacity

        T_w_3layer = list() # third layer temperature (K)
        l_w_3layer = list() # third layer thermal conductivity (W)
        C_w_3layer = list() # third layer heat capacity

        T_w_4layer = list() # fourth layer temperature (K)
        l_w_4layer = list() # fourth layer thermal conductivity (W)
        C_w_4layer = list() # fourth layer heat capacity
```

```
        T_w_5layer = list() # fifth layer temperature (K)
        l_w_5layer = list() # fifth layer thermal conductivity (W)
        C_w_5layer = list() # fifth layer heat capacity

        T_w_in = list() # inside building temperature (K)
        s_w_in = list() # inside building convection (W)
        I_w_in = list() # inside wall radiaion (W)
        C_w_in = list() # inside building capacity (J)

        T_w_in_b = list() # inside building specific temperature (K)

        T_w_1layer.insert(0, T_w_out[0] + 5) # first layer initial temperature (K)
        T_w_2layer.insert(0, T_w_1layer[0] + 4) # second layer initial temperature
(K)
        T_w_3layer.insert(0, T_w_2layer[0] + 3) # second layer initial temperature
(K)
        T_w_4layer.insert(0, T_w_3layer[0] + 2) # second layer initial temperature
(K)
        T_w_5layer.insert(0, T_w_4layer[0] + 1) # second layer initial temperature
(K)
        T_w_in.insert(0, T_w_5layer[0] - k_air) # inside building initial
temperature (K)

        S_w = w * h - S_w_wd # east wall area
        I_rad = I_w # solar irradiation

        for i in range(0, n+7): # out wall calculations
            s_w_out.insert(i, s_out * S_w * (T_street[i] - T_w_out[i]))
            T_w_out.insert(i+1, (T_street[i+1] - s_w_out[i] / (s_out * S_w)))
            T_w_irr.insert(i, ((T_w_out[i]**4 + (I_rad[i] * S_w / 3600) / (S_w *
R_0))**(1/4)))
            T_w_out_w.insert(i, (T_w_out[i] * (abs(s_w_out[i]) / (abs(s_w_out[i])
+ abs((I_rad[i] * S_w / 3600)))) + T_w_irr[i] * (abs((I_rad[i] * S_w / 3600)) /
(abs(s_w_out[i]) + abs((I_rad[i] * S_w / 3600))))))

        for i in range(0, n+6): # first layer calculations
            l_w_1layer.insert(i, L_ins * S_w * (T_w_out_w[i] - T_w_1layer[i]) /
B_ins_w)
            C_w_1layer.insert(i, (C_ins * r_ins * B_ins_w * S_w * (T_w_out_w[i] -
T_w_1layer[i])) / 3600)
            T_w_1layer.insert(i+1, ((T_w_out_w[i] - ((l_w_1layer [i] * B_ins_w) /
(L_ins * S_w))) * (abs(l_w_1layer[i]) / (2 * abs(l_w_1layer[i]) + 2 *
abs(C_w_1layer[i])))) + ((T_w_out_w[i+1] - ((C_w_1layer[i] * 3600)/ (C_ins * r_ins
* B_l * S_w))) * (abs(C_w_1layer[i]) / (2 * abs(C_w_1layer[i]) + 2 *
abs(l_w_1layer[i])))) + ((T_w_out_w[i+1] - k_ins) * (abs(l_w_1layer[i]) / (2 *
abs(C_w_1layer[i]) + 2 * abs(l_w_1layer[i])))) + ((T_w_1layer[i] - k_ins) *
(abs(C_w_1layer[i]) / (2 * abs(C_w_1layer[i]) + 2 * abs(l_w_1layer[i])))))

        for i in range(0, n+5): # second layer calculations
            l_w_2layer.insert(i, L_w * S_w * (T_w_1layer[i] - T_w_2layer[i]) /
B_l_w)
            C_w_2layer.insert(i, (C_w * r_w * B_l_w * S_w * (T_w_1layer[i] -
T_w_2layer[i])) / 3600)
            T_w_2layer.insert(i+1, ((T_w_1layer[i] - ((l_w_2layer [i] * B_l_w) /
(L_w * S_w))) * (abs(l_w_2layer[i]) / (2 * abs(l_w_2layer[i]) + 2 *
abs(C_w_2layer[i])))) + ((T_w_1layer[i+1] - ((C_w_2layer[i] * 3600)/ (C_w * r_w *
B_l * S_w))) * (abs(C_w_2layer[i]) / (2 * abs(C_w_2layer[i]) + 2 *
abs(l_w_2layer[i])))) + ((T_w_1layer[i+1] - k_w) * (abs(l_w_2layer[i]) / (2 *
abs(C_w_2layer[i]) + 2 * abs(l_w_2layer[i])))) + ((T_w_2layer[i] - k_w) *
(abs(C_w_2layer[i]) / (2 * abs(C_w_2layer[i]) + 2 * abs(l_w_2layer[i])))))

        for i in range(0, n+4): # third layer calculations
```

```
            l_w_3layer.insert(i, L_w * S_w * (T_w_2layer[i] - T_w_3layer[i]) /
B_l_w)
            C_w_3layer.insert(i, (C_w * r_w * B_l_w * S_w * (T_w_2layer[i] -
T_w_3layer[i])) / 3600)
            T_w_3layer.insert(i+1, ((T_w_2layer[i] - ((l_w_3layer [i] * B_l_w) /
(L_w * S_w))) * (abs(l_w_3layer[i]) / (2 * abs(l_w_3layer[i]) + 2 *
abs(C_w_3layer[i])))) + ((T_w_2layer[i+1] - ((C_w_3layer[i] * 3600)/ (C_w * r_w *
B_l * S_w))) * (abs(C_w_3layer[i]) / (2 * abs(C_w_3layer[i]) + 2 *
abs(l_w_3layer[i])))) + ((T_w_2layer[i+1] - k_w) * (abs(l_w_3layer[i]) / (2 *
abs(C_w_3layer[i]) + 2 * abs(l_w_3layer[i])))) + ((T_w_3layer[i] - k_w) *
(abs(C_w_3layer[i]) / (2 * abs(C_w_3layer[i]) + 2 * abs(l_w_3layer[i])))))

        for i in range(0, n+3): # fourth layer calculations
            l_w_4layer.insert(i, L_w * S_w * (T_w_3layer[i] - T_w_4layer[i]) /
B_l_w)
            C_w_4layer.insert(i, (C_w * r_w * B_l_w * S_w * (T_w_3layer[i] -
T_w_4layer[i])) / 3600)
            T_w_4layer.insert(i+1, ((T_w_3layer[i] - ((l_w_4layer [i] * B_l_w) /
(L_w * S_w))) * (abs(l_w_4layer[i]) / (2 * abs(l_w_4layer[i]) + 2 *
abs(C_w_4layer[i])))) + ((T_w_3layer[i+1] - ((C_w_4layer[i] * 3600)/ (C_w * r_w *
B_l * S_w))) * (abs(C_w_4layer[i]) / (2 * abs(C_w_4layer[i]) + 2 *
abs(l_w_4layer[i])))) + ((T_w_3layer[i+1] - k_w) * (abs(l_w_4layer[i]) / (2 *
abs(C_w_4layer[i]) + 2 * abs(l_w_4layer[i])))) + ((T_w_4layer[i] - k_w) *
(abs(C_w_4layer[i]) / (2 * abs(C_w_4layer[i]) + 2 * abs(l_w_4layer[i])))))

        for i in range(0, n+2): # fifth layer calculations
            l_w_5layer.insert(i, L_w * S_w * (T_w_4layer[i] - T_w_5layer[i]) /
B_l_w)
            C_w_5layer.insert(i, (C_w * r_w * B_l_w * S_w * (T_w_4layer[i] -
T_w_5layer[i])) / 3600)
            T_w_5layer.insert(i+1, ((T_w_4layer[i] - ((l_w_5layer [i] * B_l_w) /
(L_w * S_w))) * (abs(l_w_5layer[i]) / (2 * abs(l_w_5layer[i]) + 2 *
abs(C_w_5layer[i])))) + ((T_w_4layer[i+1] - ((C_w_5layer[i] * 3600)/ (C_w * r_w *
B_l * S_w))) * (abs(C_w_5layer[i]) / (2 * abs(C_w_5layer[i]) + 2 *
abs(l_w_5layer[i])))) + ((T_w_4layer[i+1] - k_w) * (abs(l_w_5layer[i]) / (2 *
abs(C_w_5layer[i]) + 2 * abs(l_w_5layer[i])))) + ((T_w_5layer[i] - k_w) *
(abs(C_w_5layer[i]) / (2 * abs(C_w_5layer[i]) + 2 * abs(l_w_5layer[i])))))

        for i in range(0, n+1): # inside building calculations
            s_w_in.insert(i, s_in_w * S_w * (T_w_5layer[i] - T_w_in[i]))
            I_w_in.insert(i, (T_w_5layer[i]**4 - T_w_in[i]**4) * S_w * R_in)
            C_w_in.insert(i, (C_w * r_w * B_in * S_in * (T_w_5layer[i] -
T_w_in[i])) / 3600 + ((C_air * r_air * l * w * h) * (T_w_5layer[i] - T_w_in[i])) /
3600)
            T_w_in.insert(i+1, (T_w_5layer[i+1] - s_w_in[i] / (s_in_w * S_w)) *
(abs(s_w_in[i]) / (abs(s_w_in[i]) + abs(C_w_in[i]) + abs(I_w_in[i]))) +
(T_w_5layer[i+1] - (C_w_in[i] * 3600) / (C_w * r_w * B_in * S_in + C_air * r_air *
l * w * h)) * (abs(C_w_in[i]) / (abs(s_w_in[i]) + abs(C_w_in[i]) +
abs(I_w_in[i]))) + (((T_w_5layer[i+1])**4 - I_w_in[i] / (S_w * R_in))**(1/4)) *
(abs(I_w_in[i]) / (abs(s_w_in[i]) + abs(C_w_in[i]) + abs(I_w_in[i]))))

        for i in range(0, n): # inside building specific temperature calculations
            T_w_in_b.insert(i, T_w_in[i] * S_w / S_in_b)

        return(T_w_in_b)

# south wall calculations

def south_wall_temp_calc(B_ins_w, B_l_w):

        I_rad = list() # solar irradiation

        T_w_out = list() # outside temperaure (K)
```

```
        T_w_out.insert(0, (T_street[0]-(((T_street[0])**4)*(R_0-R_out))**(1/4))) #
initial outside temperature (K)
        T_w_irr = list() # irradiation temperature
        T_w_out_w = list() # out wall temperature including radiation and
convection effects (K)
        s_w_out = list() # out wall convection (W)

        T_w_1layer = list() # first layer temperature (K)
        l_w_1layer = list() # first layer thermal conductivity (W)
        C_w_1layer = list() # first layer heat capacity

        T_w_2layer = list() # second layer temperature (K)
        l_w_2layer = list() # second layer thermal conductivity (W)
        C_w_2layer = list() # second layer heat capacity

        T_w_3layer = list() # third layer temperature (K)
        l_w_3layer = list() # third layer thermal conductivity (W)
        C_w_3layer = list() # third layer heat capacity

        T_w_4layer = list() # fourth layer temperature (K)
        l_w_4layer = list() # fourth layer thermal conductivity (W)
        C_w_4layer = list() # fourth layer heat capacity

        T_w_5layer = list() # fifth layer temperature (K)
        l_w_5layer = list() # fifth layer thermal conductivity (W)
        C_w_5layer = list() # fifth layer heat capacity

        T_w_in = list() # inside building temperature (K)
        s_w_in = list() # inside building convection (W)
        I_w_in = list() # inside wall radiaion (W)
        C_w_in = list() # inside building capacity (J)

        T_w_in_b = list() # inside building specific temperature (K)

        T_w_1layer.insert(0, T_w_out[0] + 5) # first layer initial temperature (K)
        T_w_2layer.insert(0, T_w_1layer[0] + 4) # second layer initial temperature
(K)
        T_w_3layer.insert(0, T_w_2layer[0] + 3) # second layer initial temperature
(K)
        T_w_4layer.insert(0, T_w_3layer[0] + 2) # second layer initial temperature
(K)
        T_w_5layer.insert(0, T_w_4layer[0] + 1) # second layer initial temperature
(K)
        T_w_in.insert(0, T_w_5layer[0] - k_air) # inside building initial
temperature (K)

        S_w = l * h - S_s_wd # north wall area
        I_rad = I_s # solar irradiation

        for i in range(0, n+7): # out wall calculations
            s_w_out.insert(i, s_out * S_w * (T_street[i] - T_w_out[i]))
            T_w_out.insert(i+1, (T_street[i+1] - s_w_out[i] / (s_out * S_w)))
            T_w_irr.insert(i, ((T_w_out[i]**4 + (I_rad[i] * S_w / 3600) / (S_w *
R_0))**(1/4)))
            T_w_out_w.insert(i, (T_w_out[i] * (abs(s_w_out[i]) / (abs(s_w_out[i])
+ abs((I_rad[i] * S_w / 3600)))) + T_w_irr[i] * (abs((I_rad[i] * S_w / 3600)) /
(abs(s_w_out[i]) + abs((I_rad[i] * S_w / 3600))))))

        for i in range(0, n+6): # first layer calculations
            l_w_1layer.insert(i, L_ins * S_w * (T_w_out_w[i] - T_w_1layer[i]) /
B_ins_w)
            C_w_1layer.insert(i, (C_ins * r_ins * B_ins_w * S_w * (T_w_out_w[i] -
T_w_1layer[i])) / 3600)
```

```
            T_w_1layer.insert(i+1, ((T_w_out_w[i] - ((l_w_1layer [i] * B_ins_w) /
(L_ins * S_w))) * (abs(l_w_1layer[i]) / (2 * abs(l_w_1layer[i]) + 2 *
abs(C_w_1layer[i])))) + ((T_w_out_w[i+1] - ((C_w_1layer[i] * 3600)/ (C_ins * r_ins
* B_l * S_w))) * (abs(C_w_1layer[i]) / (2 * abs(C_w_1layer[i]) + 2 *
abs(l_w_1layer[i])))) + ((T_w_out_w[i+1] - k_ins) * (abs(l_w_1layer[i]) / (2 *
abs(C_w_1layer[i]) + 2 * abs(l_w_1layer[i])))) + ((T_w_1layer[i] - k_ins) *
(abs(C_w_1layer[i]) / (2 * abs(C_w_1layer[i]) + 2 * abs(l_w_1layer[i])))))

        for i in range(0, n+5): # second layer calculations
            l_w_2layer.insert(i, L_w * S_w * (T_w_1layer[i] - T_w_2layer[i]) /
B_l_w)
            C_w_2layer.insert(i, (C_w * r_w * B_l_w * S_w * (T_w_1layer[i] -
T_w_2layer[i])) / 3600)
            T_w_2layer.insert(i+1, ((T_w_1layer[i] - ((l_w_2layer [i] * B_l_w) /
(L_w * S_w))) * (abs(l_w_2layer[i]) / (2 * abs(l_w_2layer[i]) + 2 *
abs(C_w_2layer[i])))) + ((T_w_1layer[i+1] - ((C_w_2layer[i] * 3600)/ (C_w * r_w *
B_l * S_w))) * (abs(C_w_2layer[i]) / (2 * abs(C_w_2layer[i]) + 2 *
abs(l_w_2layer[i])))) + ((T_w_1layer[i+1] - k_w) * (abs(l_w_2layer[i]) / (2 *
abs(C_w_2layer[i]) + 2 * abs(l_w_2layer[i])))) + ((T_w_2layer[i] - k_w) *
(abs(C_w_2layer[i]) / (2 * abs(C_w_2layer[i]) + 2 * abs(l_w_2layer[i])))))

        for i in range(0, n+4): # third layer calculations
            l_w_3layer.insert(i, L_w * S_w * (T_w_2layer[i] - T_w_3layer[i]) /
B_l_w)
            C_w_3layer.insert(i, (C_w * r_w * B_l_w * S_w * (T_w_2layer[i] -
T_w_3layer[i])) / 3600)
            T_w_3layer.insert(i+1, ((T_w_2layer[i] - ((l_w_3layer [i] * B_l_w) /
(L_w * S_w))) * (abs(l_w_3layer[i]) / (2 * abs(l_w_3layer[i]) + 2 *
abs(C_w_3layer[i])))) + ((T_w_2layer[i+1] - ((C_w_3layer[i] * 3600)/ (C_w * r_w *
B_l * S_w))) * (abs(C_w_3layer[i]) / (2 * abs(C_w_3layer[i]) + 2 *
abs(l_w_3layer[i])))) + ((T_w_2layer[i+1] - k_w) * (abs(l_w_3layer[i]) / (2 *
abs(C_w_3layer[i]) + 2 * abs(l_w_3layer[i])))) + ((T_w_3layer[i] - k_w) *
(abs(C_w_3layer[i]) / (2 * abs(C_w_3layer[i]) + 2 * abs(l_w_3layer[i])))))

        for i in range(0, n+3): # fourth layer calculations
            l_w_4layer.insert(i, L_w * S_w * (T_w_3layer[i] - T_w_4layer[i]) /
B_l_w)
            C_w_4layer.insert(i, (C_w * r_w * B_l_w * S_w * (T_w_3layer[i] -
T_w_4layer[i])) / 3600)
            T_w_4layer.insert(i+1, ((T_w_3layer[i] - ((l_w_4layer [i] * B_l_w) /
(L_w * S_w))) * (abs(l_w_4layer[i]) / (2 * abs(l_w_4layer[i]) + 2 *
abs(C_w_4layer[i])))) + ((T_w_3layer[i+1] - ((C_w_4layer[i] * 3600)/ (C_w * r_w *
B_l * S_w))) * (abs(C_w_4layer[i]) / (2 * abs(C_w_4layer[i]) + 2 *
abs(l_w_4layer[i])))) + ((T_w_3layer[i+1] - k_w) * (abs(l_w_4layer[i]) / (2 *
abs(C_w_4layer[i]) + 2 * abs(l_w_4layer[i])))) + ((T_w_4layer[i] - k_w) *
(abs(C_w_4layer[i]) / (2 * abs(C_w_4layer[i]) + 2 * abs(l_w_4layer[i])))))

        for i in range(0, n+2): # fifth layer calculations
            l_w_5layer.insert(i, L_w * S_w * (T_w_4layer[i] - T_w_5layer[i]) /
B_l_w)
            C_w_5layer.insert(i, (C_w * r_w * B_l_w * S_w * (T_w_4layer[i] -
T_w_5layer[i])) / 3600)
            T_w_5layer.insert(i+1, ((T_w_4layer[i] - ((l_w_5layer [i] * B_l_w) /
(L_w * S_w))) * (abs(l_w_5layer[i]) / (2 * abs(l_w_5layer[i]) + 2 *
abs(C_w_5layer[i])))) + ((T_w_4layer[i+1] - ((C_w_5layer[i] * 3600)/ (C_w * r_w *
B_l * S_w))) * (abs(C_w_5layer[i]) / (2 * abs(C_w_5layer[i]) + 2 *
abs(l_w_5layer[i])))) + ((T_w_4layer[i+1] - k_w) * (abs(l_w_5layer[i]) / (2 *
abs(C_w_5layer[i]) + 2 * abs(l_w_5layer[i])))) + ((T_w_5layer[i] - k_w) *
(abs(C_w_5layer[i]) / (2 * abs(C_w_5layer[i]) + 2 * abs(l_w_5layer[i])))))

        for i in range(0, n+1): # inside building calculations
            s_w_in.insert(i, s_in_w * S_w * (T_w_5layer[i] - T_w_in[i]))
            I_w_in.insert(i, (T_w_5layer[i]**4 - T_w_in[i]**4) * S_w * R_in)
```

```
            C_w_in.insert(i, (C_w * r_w * B_in * S_in * (T_w_5layer[i] -
T_w_in[i])) / 3600 + ((C_air * r_air * l * w * h) * (T_w_5layer[i] - T_w_in[i])) /
3600)
            T_w_in.insert(i+1, (T_w_5layer[i+1] - s_w_in[i] / (s_in_w * S_w)) *
(abs(s_w_in[i]) / (abs(s_w_in[i]) + abs(C_w_in[i]) + abs(I_w_in[i]))) +
(T_w_5layer[i+1] - (C_w_in[i] * 3600) / (C_w * r_w * B_in * S_in + C_air * r_air *
l * w * h)) * (abs(C_w_in[i]) / (abs(s_w_in[i]) + abs(C_w_in[i]) +
abs(I_w_in[i]))) + (((T_w_5layer[i+1])**4 - I_w_in[i] / (S_w * R_in))**(1/4)) *
(abs(I_w_in[i]) / (abs(s_w_in[i]) + abs(C_w_in[i]) + abs(I_w_in[i]))))

        for i in range(0, n): # inside building specific temperature calculations
            T_w_in_b.insert(i, T_w_in[i] * S_w / S_in_b)

        return(T_w_in_b)

# floor calculations

def floor_temp_calc(B_ins_f, B_l_f):

        T_f_1layer = list() # first layer temperature (K)
        l_f_1layer = list() # first layer thermal conductivity (W)
        C_f_1layer = list() # first layer heat capacity

        T_f_2layer = list() # second layer temperature (K)
        l_f_2layer = list() # second layer thermal conductivity (W)
        C_f_2layer = list() # second layer heat capacity

        T_f_3layer = list() # third layer temperature (K)
        l_f_3layer = list() # third layer thermal conductivity (W)
        C_f_3layer = list() # third layer heat capacity

        T_f_4layer = list() # fourth layer temperature (K)
        l_f_4layer = list() # fourth layer thermal conductivity (W)
        C_f_4layer = list() # fourth layer heat capacity

        T_f_5layer = list() # fifth layer temperature (K)
        l_f_5layer = list() # fifth layer thermal conductivity (W)
        C_f_5layer = list() # fifth layer heat capacity

        T_f_in = list() # inside building temperature (K)
        s_f_in = list() # inside building convection (W)
        I_f_in = list() # inside wall radiaion (W)
        C_f_in = list() # inside building capacity

        T_f_in_b = list() # inside building specific temperature (K)

        T_f_1layer.insert(0, T_soil[0] - k_ins) # first layer initial temperature
(K)
        T_f_2layer.insert(0, T_f_1layer[0] - k_w) # second layer initial
temperature (K)
        T_f_3layer.insert(0, T_f_2layer[0] - k_w) # third layer initial
temperature (K)
        T_f_4layer.insert(0, T_f_3layer[0] - k_w) # fourth layer initial
temperature (K)
        T_f_5layer.insert(0, T_f_4layer[0] - k_w) # fifth layer initial
temperature (K)
        T_f_in.insert(0, T_f_5layer[0] - k_air) # inside building initial
temperature (K)

        for i in range(0, n+6): # first layer calculations
            l_f_1layer.insert(i, L_ins * S_f * (T_soil[i] - T_f_1layer[i]) /
B_ins_f)
            C_f_1layer.insert(i, (C_ins * r_ins * B_ins_f * S_f * (T_soil[i] -
```

```
T_f_1layer[i])) / 3600)
            T_f_1layer.insert(i+1, ((T_soil[i] - ((l_f_1layer [i] * B_ins_f) /
(L_ins * S_f))) * (abs(l_f_1layer[i]) / (2 * abs(l_f_1layer[i]) + 2 *
abs(C_f_1layer[i])))) + ((T_soil[i+1] - ((C_f_1layer[i] * 3600)/ (C_ins * r_ins *
B_l * S_f))) * (abs(C_f_1layer[i]) / (2 * abs(C_f_1layer[i]) + 2 *
abs(l_f_1layer[i])))) + ((T_soil[i+1] - k_ins) * (abs(l_f_1layer[i]) / (2 *
abs(C_f_1layer[i]) + 2 * abs(l_f_1layer[i])))) + ((T_f_1layer[i] - k_ins) *
(abs(C_f_1layer[i]) / (2 * abs(C_f_1layer[i]) + 2 * abs(l_f_1layer[i])))))

        for i in range(0, n+5): # second layer calculations
            l_f_2layer.insert(i, L_w * S_f * (T_f_1layer[i] - T_f_2layer[i]) /
B_l_f)
            C_f_2layer.insert(i, (C_w * r_w * B_l_f * S_f * (T_f_1layer[i] -
T_f_2layer[i])) / 3600)
            T_f_2layer.insert(i+1, ((T_f_1layer[i] - ((l_f_2layer [i] * B_l_f) /
(L_w * S_f))) * (abs(l_f_2layer[i]) / (2 * abs(l_f_2layer[i]) + 2 *
abs(C_f_2layer[i])))) + ((T_f_1layer[i+1] - ((C_f_2layer[i] * 3600)/ (C_w * r_w *
B_l * S_f))) * (abs(C_f_2layer[i]) / (2 * abs(C_f_2layer[i]) + 2 *
abs(l_f_2layer[i])))) + ((T_f_1layer[i+1] - k_w) * (abs(l_f_2layer[i]) / (2 *
abs(C_f_2layer[i]) + 2 * abs(l_f_2layer[i])))) + ((T_f_2layer[i] - k_w) *
(abs(C_f_2layer[i]) / (2 * abs(C_f_2layer[i]) + 2 * abs(l_f_2layer[i])))))

        for i in range(0, n+4): # third layer calculations
            l_f_3layer.insert(i, L_w * S_f * (T_f_2layer[i] - T_f_3layer[i]) /
B_l_f)
            C_f_3layer.insert(i, (C_w * r_w * B_l_f * S_f * (T_f_2layer[i] -
T_f_3layer[i])) / 3600)
            T_f_3layer.insert(i+1, ((T_f_2layer[i] - ((l_f_3layer [i] * B_l_f) /
(L_w * S_f))) * (abs(l_f_3layer[i]) / (2 * abs(l_f_3layer[i]) + 2 *
abs(C_f_3layer[i])))) + ((T_f_2layer[i+1] - ((C_f_3layer[i] * 3600)/ (C_w * r_w *
B_l * S_f))) * (abs(C_f_3layer[i]) / (2 * abs(C_f_3layer[i]) + 2 *
abs(l_f_3layer[i])))) + ((T_f_2layer[i+1] - k_w) * (abs(l_f_3layer[i]) / (2 *
abs(C_f_3layer[i]) + 2 * abs(l_f_3layer[i])))) + ((T_f_3layer[i] - k_w) *
(abs(C_f_3layer[i]) / (2 * abs(C_f_3layer[i]) + 2 * abs(l_f_3layer[i])))))

        for i in range(0, n+3): # fourth layer calculations
            l_f_4layer.insert(i, L_w * S_f * (T_f_3layer[i] - T_f_4layer[i]) /
B_l_f)
            C_f_4layer.insert(i, (C_w * r_w * B_l_f * S_f * (T_f_3layer[i] -
T_f_4layer[i])) / 3600)
            T_f_4layer.insert(i+1, ((T_f_3layer[i] - ((l_f_4layer [i] * B_l_f) /
(L_w * S_f))) * (abs(l_f_4layer[i]) / (2 * abs(l_f_4layer[i]) + 2 *
abs(C_f_4layer[i])))) + ((T_f_3layer[i+1] - ((C_f_4layer[i] * 3600)/ (C_w * r_w *
B_l * S_f))) * (abs(C_f_4layer[i]) / (2 * abs(C_f_4layer[i]) + 2 *
abs(l_f_4layer[i])))) + ((T_f_3layer[i+1] - k_w) * (abs(l_f_4layer[i]) / (2 *
abs(C_f_4layer[i]) + 2 * abs(l_f_4layer[i])))) + ((T_f_4layer[i] - k_w) *
(abs(C_f_4layer[i]) / (2 * abs(C_f_4layer[i]) + 2 * abs(l_f_4layer[i])))))

        for i in range(0, n+2): # fifth layer calculations
            l_f_5layer.insert(i, L_w * S_f * (T_f_4layer[i] - T_f_5layer[i]) /
B_l_f)
            C_f_5layer.insert(i, (C_w * r_w * B_l_f * S_f * (T_f_4layer[i] -
T_f_5layer[i])) / 3600)
            T_f_5layer.insert(i+1, ((T_f_4layer[i] - ((l_f_5layer [i] * B_l_f) /
(L_w * S_f))) * (abs(l_f_5layer[i]) / (2 * abs(l_f_5layer[i]) + 2 *
abs(C_f_5layer[i])))) + ((T_f_4layer[i+1] - ((C_f_5layer[i] * 3600)/ (C_w * r_w *
B_l * S_f))) * (abs(C_f_5layer[i]) / (2 * abs(C_f_5layer[i]) + 2 *
abs(l_f_5layer[i])))) + ((T_f_4layer[i+1] - k_w) * (abs(l_f_5layer[i]) / (2 *
abs(C_f_5layer[i]) + 2 * abs(l_f_5layer[i])))) + ((T_f_5layer[i] - k_w) *
(abs(C_f_5layer[i]) / (2 * abs(C_f_5layer[i]) + 2 * abs(l_f_5layer[i])))))

        for i in range(0, n+1): # inside building calculations
            s_f_in.insert(i, s_in_f * S_f * (T_f_5layer[i] - T_f_in[i]))
```

```
            I_f_in.insert(i, (T_f_5layer[i]**4 - T_f_in[i]**4) * S_f * R_in)
            C_f_in.insert(i, (C_w * r_w * B_in * S_in * (T_f_5layer[i] -
T_f_in[i])) / 3600 + ((C_air * r_air * l * w * h) * (T_f_5layer[i] - T_f_in[i])) /
3600)
            T_f_in.insert(i+1, (T_f_5layer[i+1] - s_f_in[i] / (s_in_f * S_f)) *
(abs(s_f_in[i]) / (abs(s_f_in[i]) + abs(C_f_in[i]) + abs(I_f_in[i]))) +
(T_f_5layer[i+1] - (C_f_in[i] * 3600) / (C_w * r_w * B_in * S_in + C_air * r_air *
l * w * h)) * (abs(C_f_in[i]) / (abs(s_f_in[i]) + abs(C_f_in[i]) +
abs(I_f_in[i]))) + (((T_f_5layer[i+1])**4 - I_f_in[i] / (S_f * R_in))**(1/4)) *
(abs(I_f_in[i]) / (abs(s_f_in[i]) + abs(C_f_in[i]) + abs(I_f_in[i]))))

        for i in range(0, n): # inside building specific temperature calculations
            T_f_in_b.insert(i, T_f_in[i] * S_f / S_in_b)

        return(T_f_in_b)

# roof calculations

def roof_temp_calc(B_ins_r, B_l_r):

        I_rad = list() # solar irradiation
        I_rad = I_r

        T_r_out = list() # outside temperaure (K)
        T_r_out.insert(0, (T_street[0]-(((T_street[0])**4)*(R_0-R_out))**(1/4))) #
initial outside temperature (K)
        T_r_irr = list() # irradiation temperature
        T_r_out_w = list() # out wall temperature including radiation and
convection effects (K)
        s_r_out = list() # out wall convection (W)

        T_r_1layer = list() # first layer temperature (K)
        l_r_1layer = list() # first layer thermal conductivity (W)
        C_r_1layer = list() # first layer heat capacity

        T_r_2layer = list() # second layer temperature (K)
        l_r_2layer = list() # second layer thermal conductivity (W)
        C_r_2layer = list() # second layer heat capacity

        T_r_3layer = list() # third layer temperature (K)
        l_r_3layer = list() # third layer thermal conductivity (W)
        C_r_3layer = list() # third layer heat capacity

        T_r_4layer = list() # fourth layer temperature (K)
        l_r_4layer = list() # fourth layer thermal conductivity (W)
        C_r_4layer = list() # fourth layer heat capacity

        T_r_5layer = list() # fifth layer temperature (K)
        l_r_5layer = list() # fifth layer thermal conductivity (W)
        C_r_5layer = list() # fifth layer heat capacity

        T_r_in = list() # inside building temperature (K)
        s_r_in = list() # inside building convection (W)
        I_r_in = list() # inside wall radiaion (W)
        C_r_in = list() # inside building capacity

        T_r_in_b = list() # inside building specific temperature (K)

        T_r_1layer.insert(0, T_r_out[0] + 13) # first layer initial temperature
(K)
        T_r_2layer.insert(0, T_r_1layer[0] + 8) # second layer initial temperature
(K)
        T_r_3layer.insert(0, T_r_2layer[0] + 4) # second layer initial temperature
```

```
(K)
        T_r_4layer.insert(0, T_r_3layer[0] + 3) # second layer initial temperature
(K)
        T_r_5layer.insert(0, T_r_4layer[0] + 1) # second layer initial temperature
(K)
        T_r_in.insert(0, T_r_5layer[0] - k_air) # inside building initial
temperature (K)

        for i in range(0, n+7): # out wall calculations
            s_r_out.insert(i, s_out * S_r * (T_street[i] - T_r_out[i]))
            T_r_out.insert(i+1, (T_street[i+1] - s_r_out[i] / (s_out * S_r)))
            T_r_irr.insert(i, ((T_r_out[i]**4 + (I_rad[i] * S_r / 3600) / (S_r *
R_0))**(1/4)))
            T_r_out_w.insert(i, (T_r_out[i] * (abs(s_r_out[i]) / (abs(s_r_out[i])
+ abs((I_rad[i] * S_r / 3600)))) + T_r_irr[i] * (abs((I_rad[i] * S_r / 3600)) /
(abs(s_r_out[i]) + abs((I_rad[i] * S_r / 3600))))))

        for i in range(0, n+6): # first layer calculations
            l_r_1layer.insert(i, L_ins * S_r * (T_r_out_w[i] - T_r_1layer[i]) /
B_ins_r)
            C_r_1layer.insert(i, (C_ins * r_ins * B_ins_r * S_r * (T_r_out_w[i] -
T_r_1layer[i])) / 3600)
            T_r_1layer.insert(i+1, ((T_r_out_w[i] - ((l_r_1layer [i] * B_ins_r) /
(L_ins * S_r))) * (abs(l_r_1layer[i]) / (2 * abs(l_r_1layer[i]) + 2 *
abs(C_r_1layer[i])))) + ((T_r_out_w[i+1] - ((C_r_1layer[i] * 3600)/ (C_ins * r_ins
* B_l * S_r))) * (abs(C_r_1layer[i]) / (2 * abs(C_r_1layer[i]) + 2 *
abs(l_r_1layer[i])))) + ((T_r_out_w[i+1] - k_ins) * (abs(l_r_1layer[i]) / (2 *
abs(C_r_1layer[i]) + 2 * abs(l_r_1layer[i])))) + ((T_r_1layer[i] - k_ins) *
(abs(C_r_1layer[i]) / (2 * abs(C_r_1layer[i]) + 2 * abs(l_r_1layer[i])))))

        for i in range(0, n+5): # second layer calculations
            l_r_2layer.insert(i, L_w * S_r * (T_r_1layer[i] - T_r_2layer[i]) /
B_l_r)
            C_r_2layer.insert(i, (C_w * r_w * B_l_r * S_r * (T_r_1layer[i] -
T_r_2layer[i])) / 3600)
            T_r_2layer.insert(i+1, ((T_r_1layer[i] - ((l_r_2layer [i] * B_l_r) /
(L_w * S_r))) * (abs(l_r_2layer[i]) / (2 * abs(l_r_2layer[i]) + 2 *
abs(C_r_2layer[i])))) + ((T_r_1layer[i+1] - ((C_r_2layer[i] * 3600)/ (C_w * r_w *
B_l * S_r))) * (abs(C_r_2layer[i]) / (2 * abs(C_r_2layer[i]) + 2 *
abs(l_r_2layer[i])))) + ((T_r_1layer[i+1] - k_w) * (abs(l_r_2layer[i]) / (2 *
abs(C_r_2layer[i]) + 2 * abs(l_r_2layer[i])))) + ((T_r_2layer[i] - k_w) *
(abs(C_r_2layer[i]) / (2 * abs(C_r_2layer[i]) + 2 * abs(l_r_2layer[i])))))

        for i in range(0, n+4): # third layer calculations
            l_r_3layer.insert(i, L_w * S_r * (T_r_2layer[i] - T_r_3layer[i]) /
B_l_r)
            C_r_3layer.insert(i, (C_w * r_w * B_l_r * S_r * (T_r_2layer[i] -
T_r_3layer[i])) / 3600)
            T_r_3layer.insert(i+1, ((T_r_2layer[i] - ((l_r_3layer [i] * B_l_r) /
(L_w * S_r))) * (abs(l_r_3layer[i]) / (2 * abs(l_r_3layer[i]) + 2 *
abs(C_r_3layer[i])))) + ((T_r_2layer[i+1] - ((C_r_3layer[i] * 3600)/ (C_w * r_w *
B_l * S_r))) * (abs(C_r_3layer[i]) / (2 * abs(C_r_3layer[i]) + 2 *
abs(l_r_3layer[i])))) + ((T_r_2layer[i+1] - k_w) * (abs(l_r_3layer[i]) / (2 *
abs(C_r_3layer[i]) + 2 * abs(l_r_3layer[i])))) + ((T_r_3layer[i] - k_w) *
(abs(C_r_3layer[i]) / (2 * abs(C_r_3layer[i]) + 2 * abs(l_r_3layer[i])))))

        for i in range(0, n+3): # fourth layer calculations
            l_r_4layer.insert(i, L_w * S_r * (T_r_3layer[i] - T_r_4layer[i]) /
B_l_r)
            C_r_4layer.insert(i, (C_w * r_w * B_l_r * S_r * (T_r_3layer[i] -
T_r_4layer[i])) / 3600)
            T_r_4layer.insert(i+1, ((T_r_3layer[i] - ((l_r_4layer [i] * B_l_r) /
(L_w * S_r))) * (abs(l_r_4layer[i]) / (2 * abs(l_r_4layer[i]) + 2 *
```

```
abs(C_r_4layer[i])))) + ((T_r_3layer[i+1] - ((C_r_4layer[i] * 3600)/ (C_w * r_w *
B_l * S_r))) * (abs(C_r_4layer[i]) / (2 * abs(C_r_4layer[i]) + 2 *
abs(l_r_4layer[i])))) + ((T_r_3layer[i+1] - k_w) * (abs(l_r_4layer[i]) / (2 *
abs(C_r_4layer[i]) + 2 * abs(l_r_4layer[i])))) + ((T_r_4layer[i] - k_w) *
(abs(C_r_4layer[i]) / (2 * abs(C_r_4layer[i]) + 2 * abs(l_r_4layer[i]))))))

        for i in range(0, n+2): # fifth layer calculations
            l_r_5layer.insert(i, L_w * S_r * (T_r_4layer[i] - T_r_5layer[i]) /
B_l_r)
            C_r_5layer.insert(i, (C_w * r_w * B_l_r* S_r * (T_r_4layer[i] -
T_r_5layer[i])) / 3600)
            T_r_5layer.insert(i+1, ((T_r_4layer[i] - ((l_r_5layer [i] * B_l_r) /
(L_w * S_r))) * (abs(l_r_5layer[i]) / (2 * abs(l_r_5layer[i]) + 2 *
abs(C_r_5layer[i])))) + ((T_r_4layer[i+1] - ((C_r_5layer[i] * 3600)/ (C_w * r_w *
B_l * S_r))) * (abs(C_r_5layer[i]) / (2 * abs(C_r_5layer[i]) + 2 *
abs(l_r_5layer[i])))) + ((T_r_4layer[i+1] - k_w) * (abs(l_r_5layer[i]) / (2 *
abs(C_r_5layer[i]) + 2 * abs(l_r_5layer[i])))) + ((T_r_5layer[i] - k_w) *
(abs(C_r_5layer[i]) / (2 * abs(C_r_5layer[i]) + 2 * abs(l_r_5layer[i]))))))

        for i in range(0, n+1): # inside building calculations
            s_r_in.insert(i, s_in_r * S_r * (T_r_5layer[i] - T_r_in[i]))
            I_r_in.insert(i, (T_r_5layer[i]**4 - T_r_in[i]**4) * S_r * R_in)
            C_r_in.insert(i, (C_w * r_w * B_in * S_in * (T_r_5layer[i] -
T_r_in[i])) / 3600 + ((C_air * r_air * l * w * h) * (T_r_5layer[i] - T_r_in[i])) /
3600)
            T_r_in.insert(i+1, (T_r_5layer[i+1] - s_r_in[i] / (s_in_r * S_r)) *
(abs(s_r_in[i]) / (abs(s_r_in[i]) + abs(C_r_in[i]) + abs(I_r_in[i]))) +
(T_r_5layer[i+1] - (C_r_in[i] * 3600) / (C_w * r_w * B_in * S_in + C_air * r_air *
l * w * h)) * (abs(C_r_in[i]) / (abs(s_r_in[i]) + abs(C_r_in[i]) +
abs(I_r_in[i]))) + (((T_r_5layer[i+1])**4 - I_r_in[i] / (S_r * R_in))**(1/4)) *
(abs(I_r_in[i]) / (abs(s_r_in[i]) + abs(C_r_in[i]) + abs(I_r_in[i]))))

        for i in range(0, n): # inside building specific temperature calculations
            T_r_in_b.insert(i, T_r_in[i] * S_r / S_in_b)

        return(T_r_in_b)

# north windows calculations

def north_windows_temp_calc(x):

        I_rad = list() # solar irradiation
        I_rad = I_n # solar irradation
        S_wd = S_n_wd # windows area (m2)
        m_wd = r_wd * S_wd * x  # window mass
        # FDM coefficient for window
        k_wd = x * ((L_g) / (C_wd * r_wd)) / (2 * (((L_g) / (C_wd * r_wd)) + 1)) +
k_air
        R_t_wd = (0.95 ** (x))  # transparency coefficient
        L_wd = x * L_g + L_air  # window heat conductivity coefficient
        B_wd = (x + 1) * B_gl  # window thickness

        T_wd_rad_in_b = list() # inside building specific temperature (K)

        T_wd_out = list() # outside temperaure (K)
        T_wd_out.insert(0, (T_street[0]-(((T_street[0])**4)*(R_0-
R_wd_out))**(1/4))) # initial outside temperature (K)
        T_wd_irr = list() # irradiation temperature
        T_wd_out_w = list() # out wall temperature including radiation and
convection effects (K)
        s_wd_out = list() # out wall convection (W)

        T_wd_1layer = list() # first layer temperature (K)
```

```
        l_wd_1layer = list() # first layer thermal conductivity (W)
        C_wd_1layer = list() # first layer heat capacity

        T_wd_in = list() # inside building temperature (K)
        s_wd_in = list() # inside building convection (W)
        I_wd_in = list() # inside building radiation (W)
        C_wd_in = list() # inside building capacity (J)

        T_wd_rad_in = list() # inside building temperature including throuh-window
radiation (K)

        T_wd_1layer.insert(0, T_wd_out[0] + 1) # first layer initial temperature
(K)
        T_wd_in.insert(0, T_wd_1layer[0] - k_air) # inside building initial
temperature (K)

        for i in range(0, n+4): # out wall calculations
            s_wd_out.insert(i, s_out * S_wd * (T_street[i] - T_wd_out[i]))
            T_wd_out.insert(i+1, (T_street[i+1] - s_wd_out[i] / (s_out * S_wd)))
            T_wd_irr.insert(i, ((T_wd_out[i]**4 + (I_rad[i] * S_wd / 3600) / (S_wd
* R_0))**(1/4)))
            T_wd_out_w.insert(i, (T_wd_out[i] * (abs(s_wd_out[i]) /
(abs(s_wd_out[i]) + abs((I_rad[i] * S_wd / 3600)))) + T_wd_irr[i] * (abs((I_rad[i]
* S_wd / 3600)) / (abs(s_wd_out[i]) + abs((I_rad[i] * S_wd / 3600))))))

        for i in range(0, n+3): # first layer calculations
            l_wd_1layer.insert(i, L_wd * S_wd * (T_wd_out_w[i] - T_wd_1layer[i]) /
B_wd)
            C_wd_1layer.insert(i, (C_wd * m_wd * (T_wd_out_w[i] - T_wd_1layer[i]))
/ 3600)
            T_wd_1layer.insert(i+1, ((T_wd_out_w[i] - ((l_wd_1layer [i] * B_wd) /
(L_wd * S_wd))) * (abs(l_wd_1layer[i]) / (2 * abs(l_wd_1layer[i]) + 2 *
abs(C_wd_1layer[i])))) + ((T_wd_out_w[i+1] - ((C_wd_1layer[i] * 3600) / (C_wd *
m_wd))) * (abs(C_wd_1layer[i]) / (2 * abs(C_wd_1layer[i]) + 2 *
abs(l_wd_1layer[i])))) + ((T_wd_out_w[i+1] - k_wd) * (abs(l_wd_1layer[i]) / (2 *
abs(C_wd_1layer[i]) + 2 * abs(l_wd_1layer[i])))) + ((T_wd_1layer[i] - k_wd) *
(abs(C_wd_1layer[i]) / (2 * abs(C_wd_1layer[i]) + 2 * abs(l_wd_1layer[i])))))

        for i in range(0, n+2): # inside building calculations
            s_wd_in.insert(i, s_in_wd * S_wd * (T_wd_1layer[i] - T_wd_in[i]))
            I_wd_in.insert(i, (T_wd_1layer[i]**4 - T_wd_in[i]**4) * S_wd *
R_wd_in)
            C_wd_in.insert(i, (C_w * r_w * B_in * S_in * (T_wd_1layer[i] -
T_wd_in[i])) / 3600
                           + ((C_air * r_air * l * w * h) * (T_wd_1layer[i] -
T_wd_in[i])) / 3600)
            T_wd_in.insert(i+1, (T_wd_1layer[i+1] - s_wd_in[i] / (s_in_wd * S_wd))
*
                           (abs(s_wd_in[i]) / (abs(s_wd_in[i]) + abs(C_wd_in[i]) +
abs(I_wd_in[i])))
                           + (T_wd_1layer[i+1] - (C_wd_in[i] * 3600) / (C_w * r_w
* B_in * S_in + C_air * r_air * l * w * h)) *
                           (abs(C_wd_in[i]) / (abs(s_wd_in[i]) + abs(C_wd_in[i]) +
abs(I_wd_in[i])))
                           + (((T_wd_1layer[i+1])**4 - I_wd_in[i] / (S_wd *
R_wd_in))**(1/4)) *
                           (abs(I_wd_in[i]) / (abs(s_wd_in[i]) + abs(C_wd_in[i]) +
abs(I_wd_in[i]))))

        for i in range(0, n+1): # inside building temperature including throuh-
window radiation calculation
            T_wd_rad_in.insert(i, T_wd_in[i] * ((abs(s_wd_in[i]) +
abs(C_wd_in[i])+ abs(I_wd_in[i])) / ((abs(s_wd_in[i]) + abs(C_wd_in[i])+
```

```
abs(I_wd_in[i])) + abs((R_t_wd * I_rad[i] * S_wd / 3600)))) + T_wd_irr[i] *
(abs((R_t_wd * I_rad[i] * S_wd / 3600)) / (abs(s_wd_in[i]) + abs(C_wd_in[i]) +
abs(I_wd_in[i]) + abs((R_t_wd * I_rad[i] * S_wd / 3600)))))

        for i in range(0, n): # inside building specific temperature calculations
            T_wd_rad_in_b.insert(i, T_wd_rad_in[i] * S_wd / S_in_b)

        return(T_wd_rad_in_b)

# east windows calculations

def east_windows_temp_calc(x):

    # west windows calculations

        I_rad = list() # solar irradiation
        I_rad = I_e # solar irradation
        S_wd = S_e_wd # windows area (m2)
        m_wd = r_wd * S_wd * x  # window mass
        k_wd = x * ((L_g) / (C_wd * r_wd)) / (2 * (((L_g) / (C_wd * r_wd)) + 1)) +
k_air  # FDM coefficient for window
        R_t_wd = (0.95 ** (x))  # radiation coefficient
        L_wd = x * L_g + L_air  # window heat conductivity coefficient
        B_wd = (x + 1) * B_gl  # window thickness

        T_wd_rad_in_b = list() # inside building specific temperature (K)

        T_wd_out = list() # outside temperaure (K)
        T_wd_out.insert(0, (T_street[0]-(((T_street[0])**4)*(R_0-
R_wd_out))**(1/4))) # initial outside temperature (K)
        T_wd_irr = list() # irradiation temperature
        T_wd_out_w = list() # out wall temperature including radiation and
convection effects (K)
        s_wd_out = list() # out wall convection (W)

        T_wd_1layer = list() # first layer temperature (K)
        l_wd_1layer = list() # first layer thermal conductivity (W)
        C_wd_1layer = list() # first layer heat capacity

        T_wd_in = list() # inside building temperature (K)
        s_wd_in = list() # inside building convection (W)
        I_wd_in = list() # inside building radiation (W)
        C_wd_in = list() # inside building capacity (J)

        T_wd_rad_in = list() # inside building temperature including throuh-window
radiation (K)

        T_wd_1layer.insert(0, T_wd_out[0] + 1) # first layer initial temperature
(K)
        T_wd_in.insert(0, T_wd_1layer[0] - k_air) # inside building initial
temperature (K)

        for i in range(0, n+4): # out wall calculations
            s_wd_out.insert(i, s_out * S_wd * (T_street[i] - T_wd_out[i]))
            T_wd_out.insert(i+1, (T_street[i+1] - s_wd_out[i] / (s_out * S_wd)))
            T_wd_irr.insert(i, ((T_wd_out[i]**4 + (I_rad[i] * S_wd / 3600) / (S_wd
* R_0))**(1/4)))
            T_wd_out_w.insert(i, (T_wd_out[i] * (abs(s_wd_out[i]) /
(abs(s_wd_out[i]) + abs((I_rad[i] * S_wd / 3600)))) + T_wd_irr[i] * (abs((I_rad[i]
* S_wd / 3600)) / (abs(s_wd_out[i]) + abs((I_rad[i] * S_wd / 3600))))))

        for i in range(0, n+3): # first layer calculations
            l_wd_1layer.insert(i, L_wd * S_wd * (T_wd_out_w[i] - T_wd_1layer[i]) /
```

```
B_wd)
            C_wd_1layer.insert(i, (C_wd * m_wd * (T_wd_out_w[i] - T_wd_1layer[i]))
/ 3600)
            T_wd_1layer.insert(i+1, ((T_wd_out_w[i] - ((l_wd_1layer [i] * B_wd) /
(L_wd * S_wd))) * (abs(l_wd_1layer[i]) / (2 * abs(l_wd_1layer[i]) + 2 *
abs(C_wd_1layer[i])))) + ((T_wd_out_w[i+1] - ((C_wd_1layer[i] * 3600) / (C_wd *
m_wd))) * (abs(C_wd_1layer[i]) / (2 * abs(C_wd_1layer[i]) + 2 *
abs(l_wd_1layer[i])))) + ((T_wd_out_w[i+1] - k_wd) * (abs(l_wd_1layer[i]) / (2 *
abs(C_wd_1layer[i]) + 2 * abs(l_wd_1layer[i])))) + ((T_wd_1layer[i] - k_wd) *
(abs(C_wd_1layer[i]) / (2 * abs(C_wd_1layer[i]) + 2 * abs(l_wd_1layer[i])))))

        for i in range(0, n+2): # inside building calculations
            s_wd_in.insert(i, s_in_wd * S_wd * (T_wd_1layer[i] - T_wd_in[i]))
            I_wd_in.insert(i, (T_wd_1layer[i]**4 - T_wd_in[i]**4) * S_wd *
R_wd_in)
            C_wd_in.insert(i, (C_w * r_w * B_in * S_in * (T_wd_1layer[i] -
T_wd_in[i])) / 3600 + ((C_air * r_air * l * w * h) * (T_wd_1layer[i] -
T_wd_in[i])) / 3600)
            T_wd_in.insert(i+1, (T_wd_1layer[i+1] - s_wd_in[i] / (s_in_wd * S_wd))
* (abs(s_wd_in[i]) / (abs(s_wd_in[i]) + abs(C_wd_in[i]) + abs(I_wd_in[i]))) +
(T_wd_1layer[i+1] - (C_wd_in[i] * 3600) / (C_w * r_w * B_in * S_in + C_air * r_air
* l * w * h)) * (abs(C_wd_in[i]) / (abs(s_wd_in[i]) + abs(C_wd_in[i]) +
abs(I_wd_in[i]))) + (((T_wd_1layer[i+1])**4 - I_wd_in[i] / (S_wd *
R_wd_in))**(1/4)) * (abs(I_wd_in[i]) / (abs(s_wd_in[i]) + abs(C_wd_in[i]) +
abs(I_wd_in[i]))))

        for i in range(0, n+1): # inside building temperature including throuh-
window radiation calculation
            T_wd_rad_in.insert(i, T_wd_in[i] * ((abs(s_wd_in[i]) +
abs(C_wd_in[i])+ abs(I_wd_in[i])) / ((abs(s_wd_in[i]) + abs(C_wd_in[i])+
abs(I_wd_in[i])) + abs((R_t_wd * I_rad[i] * S_wd / 3600)))) + T_wd_irr[i] *
(abs((R_t_wd * I_rad[i] * S_wd / 3600)) / (abs(s_wd_in[i]) + abs(C_wd_in[i]) +
abs(I_wd_in[i]) + abs((R_t_wd * I_rad[i] * S_wd / 3600)))))

        for i in range(0, n): # inside building specific temperature calculations
            T_wd_rad_in_b.insert(i, T_wd_rad_in[i] * S_wd / S_in_b)

        return(T_wd_rad_in_b)

# west windows calculations

def west_windows_temp_calc(x):

        I_rad = list() # solar irradiation
        I_rad = I_w # solar irradation
        S_wd = S_w_wd # windows area (m2)
        m_wd = r_wd * S_wd * x  # window mass
        k_wd = x * ((L_g) / (C_wd * r_wd)) / (2 * (((L_g) / (C_wd * r_wd)) + 1)) +
k_air  # FDM coefficient for window
        R_t_wd = (0.95 ** (x))  # radiation coefficient
        L_wd = x * L_g + L_air  # window heat conductivity coefficient
        B_wd = (x + 1) * B_gl  # window thickness

        T_wd_rad_in_b = list() # inside building specific temperature (K)

        T_wd_out = list() # outside temperaure (K)
        T_wd_out.insert(0, (T_street[0]-(((T_street[0])**4)*(R_0-
R_wd_out))**(1/4))) # initial outside temperature (K)
        T_wd_irr = list() # irradiation temperature
        T_wd_out_w = list() # out wall temperature including radiation and
convection effects (K)
        s_wd_out = list() # out wall convection (W)
```

```
        T_wd_1layer = list() # first layer temperature (K)
        l_wd_1layer = list() # first layer thermal conductivity (W)
        C_wd_1layer = list() # first layer heat capacity

        T_wd_in = list() # inside building temperature (K)
        s_wd_in = list() # inside building convection (W)
        I_wd_in = list() # inside building radiation (W)
        C_wd_in = list() # inside building capacity (J)

        T_wd_rad_in = list() # inside building temperature including throuh-window
radiation (K)

        T_wd_1layer.insert(0, T_wd_out[0] + 1) # first layer initial temperature
(K)
        T_wd_in.insert(0, T_wd_1layer[0] - k_air) # inside building initial
temperature (K)

        for i in range(0, n+4): # out wall calculations
            s_wd_out.insert(i, s_out * S_wd * (T_street[i] - T_wd_out[i]))
            T_wd_out.insert(i+1, (T_street[i+1] - s_wd_out[i] / (s_out * S_wd)))
            T_wd_irr.insert(i, ((T_wd_out[i]**4 + (I_rad[i] * S_wd / 3600) / (S_wd
* R_0))**(1/4)))
            T_wd_out_w.insert(i, (T_wd_out[i] * (abs(s_wd_out[i]) /
(abs(s_wd_out[i]) + abs((I_rad[i] * S_wd / 3600)))) + T_wd_irr[i] * (abs((I_rad[i]
* S_wd / 3600)) / (abs(s_wd_out[i]) + abs((I_rad[i] * S_wd / 3600))))))

        for i in range(0, n+3): # first layer calculations
            l_wd_1layer.insert(i, L_wd * S_wd * (T_wd_out_w[i] - T_wd_1layer[i]) /
B_wd)
            C_wd_1layer.insert(i, (C_wd * m_wd * (T_wd_out_w[i] - T_wd_1layer[i]))
/ 3600)
            T_wd_1layer.insert(i+1, ((T_wd_out_w[i] - ((l_wd_1layer [i] * B_wd) /
(L_wd * S_wd))) * (abs(l_wd_1layer[i]) / (2 * abs(l_wd_1layer[i]) + 2 *
abs(C_wd_1layer[i])))) + ((T_wd_out_w[i+1] - ((C_wd_1layer[i] * 3600) / (C_wd *
m_wd))) * (abs(C_wd_1layer[i]) / (2 * abs(C_wd_1layer[i]) + 2 *
abs(l_wd_1layer[i])))) + ((T_wd_out_w[i+1] - k_wd) * (abs(l_wd_1layer[i]) / (2 *
abs(C_wd_1layer[i]) + 2 * abs(l_wd_1layer[i])))) + ((T_wd_1layer[i] - k_wd) *
(abs(C_wd_1layer[i]) / (2 * abs(C_wd_1layer[i]) + 2 * abs(l_wd_1layer[i])))))

        for i in range(0, n+2): # inside building calculations
            s_wd_in.insert(i, s_in_wd * S_wd * (T_wd_1layer[i] - T_wd_in[i]))
            I_wd_in.insert(i, (T_wd_1layer[i]**4 - T_wd_in[i]**4) * S_wd *
R_wd_in)
            C_wd_in.insert(i, (C_w * r_w * B_in * S_in * (T_wd_1layer[i] -
T_wd_in[i])) / 3600 + ((C_air * r_air * l * w * h) * (T_wd_1layer[i] -
T_wd_in[i])) / 3600)
            T_wd_in.insert(i+1, (T_wd_1layer[i+1] - s_wd_in[i] / (s_in_wd * S_wd))
* (abs(s_wd_in[i]) / (abs(s_wd_in[i]) + abs(C_wd_in[i]) + abs(I_wd_in[i]))) +
(T_wd_1layer[i+1] - (C_wd_in[i] * 3600) / (C_w * r_w * B_in * S_in + C_air * r_air
* l * w * h)) * (abs(C_wd_in[i]) / (abs(s_wd_in[i]) + abs(C_wd_in[i]) +
abs(I_wd_in[i]))) + (((T_wd_1layer[i+1])**4 - I_wd_in[i] / (S_wd *
R_wd_in))**(1/4)) * (abs(I_wd_in[i]) / (abs(s_wd_in[i]) + abs(C_wd_in[i]) +
abs(I_wd_in[i]))))

        for i in range(0, n+1): # inside building temperature including throuh-
window radiation calculation
            T_wd_rad_in.insert(i, T_wd_in[i] * ((abs(s_wd_in[i]) +
abs(C_wd_in[i])+ abs(I_wd_in[i])) / ((abs(s_wd_in[i]) + abs(C_wd_in[i])+
abs(I_wd_in[i])) + abs((R_t_wd * I_rad[i] * S_wd / 3600)))) + T_wd_irr[i] *
(abs((R_t_wd * I_rad[i] * S_wd / 3600)) / (abs(s_wd_in[i]) + abs(C_wd_in[i]) +
abs(I_wd_in[i]) + abs((R_t_wd * I_rad[i] * S_wd / 3600)))))

        for i in range(0, n): # inside building specific temperature calculations
```

```
            T_wd_rad_in_b.insert(i, T_wd_rad_in[i] * S_wd / S_in_b)

        return(T_wd_rad_in_b)

# south windows calculations

def south_windows_temp_calc(x):

        I_rad = list() # solar irradiation
        I_rad = I_s # solar irradation
        S_wd = S_s_wd # windows area (m2)
        m_wd = r_wd * S_wd * x  # window mass
        k_wd = x * ((L_g) / (C_wd * r_wd)) / (2 * (((L_g) / (C_wd * r_wd)) + 1)) +
k_air  # FDM coefficient for window
        R_t_wd = (0.95 ** (x))  # radiation coefficient
        L_wd = x * L_g + L_air  # window heat conductivity coefficient
        B_wd = (x + 1) * B_gl  # window thickness

        T_wd_rad_in_b = list() # inside building specific temperature (K)

        T_wd_out = list() # outside temperaure (K)
        T_wd_out.insert(0, (T_street[0]-(((T_street[0])**4)*(R_0-
R_wd_out))**(1/4))) # initial outside temperature (K)
        T_wd_irr = list() # irradiation temperature
        T_wd_out_w = list() # out wall temperature including radiation and
convection effects (K)
        s_wd_out = list() # out wall convection (W)

        T_wd_1layer = list() # first layer temperature (K)
        l_wd_1layer = list() # first layer thermal conductivity (W)
        C_wd_1layer = list() # first layer heat capacity

        T_wd_in = list() # inside building temperature (K)
        s_wd_in = list() # inside building convection (W)
        I_wd_in = list() # inside building radiation (W)
        C_wd_in = list() # inside building capacity (J)

        T_wd_rad_in = list() # inside building temperature including throuh-window
radiation (K)

        T_wd_1layer.insert(0, T_wd_out[0] + 1) # first layer initial temperature
(K)
        T_wd_in.insert(0, T_wd_1layer[0] - k_air) # inside building initial
temperature (K)

        for i in range(0, n+4): # out wall calculations
            s_wd_out.insert(i, s_out * S_wd * (T_street[i] - T_wd_out[i]))
            T_wd_out.insert(i+1, (T_street[i+1] - s_wd_out[i] / (s_out * S_wd)))
            T_wd_irr.insert(i, ((T_wd_out[i]**4 + (I_rad[i] * S_wd / 3600) / (S_wd
* R_0))**(1/4)))
            T_wd_out_w.insert(i, (T_wd_out[i] * (abs(s_wd_out[i]) /
(abs(s_wd_out[i]) + abs((I_rad[i] * S_wd / 3600)))) + T_wd_irr[i] * (abs((I_rad[i]
* S_wd / 3600)) / (abs(s_wd_out[i]) + abs((I_rad[i] * S_wd / 3600))))))

        for i in range(0, n+3): # first layer calculations
            l_wd_1layer.insert(i, L_wd * S_wd * (T_wd_out_w[i] - T_wd_1layer[i]) /
B_wd)
            C_wd_1layer.insert(i, (C_wd * m_wd * (T_wd_out_w[i] - T_wd_1layer[i]))
/ 3600)
            T_wd_1layer.insert(i+1, ((T_wd_out_w[i] - ((l_wd_1layer [i] * B_wd) /
(L_wd * S_wd))) * (abs(l_wd_1layer[i]) / (2 * abs(l_wd_1layer[i]) + 2 *
abs(C_wd_1layer[i])))) + ((T_wd_out_w[i+1] - ((C_wd_1layer[i] * 3600) / (C_wd *
m_wd))) * (abs(C_wd_1layer[i]) / (2 * abs(C_wd_1layer[i]) + 2 *
```

```
abs(l_wd_1layer[i])))) + ((T_wd_out_w[i+1] - k_wd) * (abs(l_wd_1layer[i]) / (2 *
abs(C_wd_1layer[i]) + 2 * abs(l_wd_1layer[i])))) + ((T_wd_1layer[i] - k_wd) *
(abs(C_wd_1layer[i]) / (2 * abs(C_wd_1layer[i]) + 2 * abs(l_wd_1layer[i])))))

        for i in range(0, n+2): # inside building calculations
            s_wd_in.insert(i, s_in_wd * S_wd * (T_wd_1layer[i] - T_wd_in[i]))
            I_wd_in.insert(i, (T_wd_1layer[i]**4 - T_wd_in[i]**4) * S_wd *
R_wd_in)
            C_wd_in.insert(i, (C_w * r_w * B_in * S_in * (T_wd_1layer[i] -
T_wd_in[i])) / 3600 + ((C_air * r_air * l * w * h) * (T_wd_1layer[i] -
T_wd_in[i])) / 3600)
            T_wd_in.insert(i+1, (T_wd_1layer[i+1] - s_wd_in[i] / (s_in_wd * S_wd))
* (abs(s_wd_in[i]) / (abs(s_wd_in[i]) + abs(C_wd_in[i]) + abs(I_wd_in[i]))) +
(T_wd_1layer[i+1] - (C_wd_in[i] * 3600) / (C_w * r_w * B_in * S_in + C_air * r_air
* l * w * h)) * (abs(C_wd_in[i]) / (abs(s_wd_in[i]) + abs(C_wd_in[i]) +
abs(I_wd_in[i]))) + (((T_wd_1layer[i+1])**4 - I_wd_in[i] / (S_wd *
R_wd_in))**(1/4)) * (abs(I_wd_in[i]) / (abs(s_wd_in[i]) + abs(C_wd_in[i]) +
abs(I_wd_in[i]))))

        for i in range(0, n+1): # inside building temperature including throuh-
window radiation calculation
            T_wd_rad_in.insert(i, T_wd_in[i] * ((abs(s_wd_in[i]) +
abs(C_wd_in[i])+ abs(I_wd_in[i])) / ((abs(s_wd_in[i]) + abs(C_wd_in[i])+
abs(I_wd_in[i])) + abs((R_t_wd * I_rad[i] * S_wd / 3600)))) + T_wd_irr[i] *
(abs((R_t_wd * I_rad[i] * S_wd / 3600)) / (abs(s_wd_in[i]) + abs(C_wd_in[i]) +
abs(I_wd_in[i]) + abs((R_t_wd * I_rad[i] * S_wd / 3600)))))

        for i in range(0, n): # inside building specific temperature calculations
            T_wd_rad_in_b.insert(i, T_wd_rad_in[i] * S_wd / S_in_b)

        return(T_wd_rad_in_b)

# Trombe wall calculations

def Trombe_wall_temp_calc(B_wd_tr, B_g, B_tr):

        m_wd_tr = r_wd * l * h * B_wd_tr  # Trombe wall glass' mass
        k_wd_tr = (((L_g) / (C_wd * r_wd)) / (2 * (((L_g)/(C_wd * r_wd))+1))) #
FDM coefficient for Trombe wall
        R_t_wd = (0.95 ** (B_wd_tr/0.02)) # transparency coefficient
        L_wd = L_g # thermal conducivity coefficient

        T_tr_in_b = list() # inside building specific temperature (K)

        T_tr_out = list() # outside temperaure (K)
        T_tr_out.insert(0, (T_street[0]-(((T_street[0])**4)*(R_0-
R_wd_out))**(1/4))) # initial outside temperature (K)
        T_tr_irr = list() # irradiation temperature
        T_tr_out_w = list() # out wall temperature including radiation and
convection effects (K)
        s_tr_out = list() # out wall convection (W)

        T_g_layer = list() # glass layer temperature (K)
        l_g_layer = list() # glass layer thermal conductivity (W)
        C_g_layer = list() # glass layer heat capacity

        T_tr_mw_out = list() # air gap temperature (K)
        l_tr_mw_out = list() # air gap thermal conductivity (W)
        C_tr_mw_out = list() # air gap heat capacity (J)
        I_tr_mw_out = list() # air gap radiation (W)
        s_tr_mw_out = list() # air gap convection (W)
        s_ag = list() # air gap convection coefficient
        T_tr_mw_rad_out = list() # outside massive wall layer temperature (K)
```

```
        T_tr_mw_in = list() # massive wall inside temperature (K)
        l_tr_mw_in = list() # massive wall thermal conductivity (W)
        C_tr_mw_in = list() # massive wall heat capacity

        T_tr_in = list() # inside building temperature (K)
        s_tr_in = list() # inside building convection (W)
        I_tr_in = list() # Inside building radiation (W)
        C_tr_in = list() # inside building capacity (J)

        T_g_layer.insert(0, T_tr_out[0] + 7) # glass layer initial temperature (K)
        T_tr_mw_out.insert(0, T_g_layer[0] + 1) # air gap initial temperature (K)
        T_tr_mw_in.insert(0, T_tr_mw_out[0] + 7) # massive wall inside initial
temperature (K)
        T_tr_in.insert(0, T_tr_mw_in[0] - k_air) # inside building initial
temperature (K)

        for i in range(0, n+5): # out wall calculations
            s_tr_out.insert(i, s_out * S_tr * (T_street[i] - T_tr_out[i]))
            T_tr_out.insert(i+1, (T_street[i+1] - s_tr_out[i] / (s_out * S_tr)))
            T_tr_irr.insert(i, ((T_tr_out[i]**4 + (I_s[i] * S_tr / 3600) / (S_tr *
R_0))**(1/4)))
            T_tr_out_w.insert(i, (T_tr_out[i] * (abs(s_tr_out[i]) /
(abs(s_tr_out[i]) + abs((I_s[i] * S_tr / 3600))))
                                  + T_tr_irr[i] * (abs((I_s[i] * S_tr / 3600)) /
(abs(s_tr_out[i]) + abs((I_s[i] * S_tr / 3600))))))

        for i in range(0, n+4): # glass layer calculations
            l_g_layer.insert(i, L_wd * S_tr * (T_tr_out_w[i] - T_g_layer[i]) /
B_wd_tr)
            C_g_layer.insert(i, (C_wd * m_wd_tr * (T_tr_out_w[i] - T_g_layer[i]))
/ 3600)
            T_g_layer.insert(i+1, ((T_tr_out_w[i] - ((l_g_layer [i] * B_wd_tr) /
(L_wd * S_tr))) *
                                   (abs(l_g_layer[i]) / (2 * abs(l_g_layer[i]) + 2
* abs(C_g_layer[i])))))
                               + ((T_tr_out_w[i+1] - ((C_g_layer[i] * 3600) / (C_wd
* m_wd_tr))) *
                                  (abs(C_g_layer[i]) / (2 * abs(C_g_layer[i]) + 2 *
abs(l_g_layer[i])))))
                               + ((T_tr_out_w[i+1] - k_wd_tr) * (abs(l_g_layer[i]) /
(2 * abs(C_g_layer[i]) + 2 * abs(l_g_layer[i])))))
                               + ((T_g_layer[i] - k_wd_tr) * (abs(C_g_layer[i]) / (2
* abs(C_g_layer[i]) + 2 * abs(l_g_layer[i])))))

        for i in range(0, n+3): # air gap layer calculations
            l_tr_mw_out.insert(i, L_air * S_tr * (T_g_layer[i] - T_tr_mw_out[i]) /
B_g)
            C_tr_mw_out.insert(i, (C_air * r_air * B_g * S_tr * (T_g_layer[i] -
T_tr_mw_out[i]) / 3600))
            I_tr_mw_out.insert(i, (T_g_layer[i]**4 - T_tr_mw_out[i]**4) * S_tr *
R_wd_in)
            s_ag.insert(i, (0.5 * ((9.81 * 0.0367 * abs(T_tr_mw_out[i] -
T_g_layer[i]) * B_g**4) / (8 * 0.000016**2 * h))**0.294))
            s_tr_mw_out.insert(i, s_ag[i] * S_tr * (T_g_layer[i] -
T_tr_mw_out[i]))
            T_tr_mw_out.insert(i+1, ((T_g_layer[i] - ((l_tr_mw_out[i] * B_g) /
(L_air * S_tr))) * (abs(l_tr_mw_out[i]) / (2 * abs(l_tr_mw_out[i]) + 2 *
abs(C_tr_mw_out[i]) + abs(s_tr_mw_out[i]) + abs(I_tr_mw_out[i])))) +
((T_g_layer[i+1] - ((C_tr_mw_out[i] * 3600)/ (C_air * r_air * B_g * S_tr))) *
(abs(C_tr_mw_out[i]) / (2 * abs(l_tr_mw_out[i]) + 2 * abs(C_tr_mw_out[i]) +
abs(s_tr_mw_out[i]) + abs(I_tr_mw_out[i])))) + ((T_g_layer[i+1] - k_air) *
(abs(l_tr_mw_out[i]) / (2 * abs(l_tr_mw_out[i]) + 2 * abs(C_tr_mw_out[i]) +
```

```
abs(s_tr_mw_out[i]) + abs(I_tr_mw_out[i])))) + ((T_tr_mw_out[i] - k_air) *
(abs(C_tr_mw_out[i]) / (2 * abs(l_tr_mw_out[i]) + 2 * abs(C_tr_mw_out[i]) +
abs(s_tr_mw_out[i]) + abs(I_tr_mw_out[i])))) + ((T_g_layer[i+1] - s_tr_mw_out[i] /
(s_ag[i] * S_tr)) * (abs(s_tr_mw_out[i]) / (2 * abs(l_tr_mw_out[i]) + 2 *
abs(C_tr_mw_out[i]) + abs(s_tr_mw_out[i]) + abs(I_tr_mw_out[i])))) +
(((T_g_layer[i+1]**4 - I_tr_mw_out[i] / (R_wd_in * S_tr))**(1/4)) *
(abs(I_tr_mw_out[i]) / (2 * abs(l_tr_mw_out[i]) + 2 * abs(C_tr_mw_out[i]) +
abs(s_tr_mw_out[i]) + abs(I_tr_mw_out[i]))))))
            T_tr_mw_rad_out.insert(i, (T_tr_mw_out[i] * ((abs(l_tr_mw_out[i]) +
abs(C_tr_mw_out[i]) + abs(I_tr_mw_out[i]) + abs(s_tr_mw_out[i])) /
(abs(l_tr_mw_out[i]) + abs(C_tr_mw_out[i]) + abs(I_tr_mw_out[i]) +
abs(s_tr_mw_out[i]) + abs((R_t_wd * I_s[i] * S_tr / 3600))))) + (T_tr_irr[i] *
(abs((R_t_wd * I_s[i] * S_tr / 3600)) / (abs(l_tr_mw_out[i]) + abs(C_tr_mw_out[i])
+ abs(I_tr_mw_out[i]) + abs(s_tr_mw_out[i]) + abs((R_t_wd * I_s[i] * S_tr /
3600))))))

        for i in range(0, n+2): # third layer calculations
            l_tr_mw_in.insert(i, L_tr * S_tr * (T_tr_mw_rad_out[i] -
T_tr_mw_in[i]) / B_tr)
            C_tr_mw_in.insert(i, (C_tr * r_tr * B_tr * S_tr * (T_tr_mw_rad_out[i]
- T_tr_mw_in[i])) / 3600)
            T_tr_mw_in.insert(i+1, ((T_tr_mw_rad_out[i] - ((l_tr_mw_in[i] * B_tr)
/ (L_tr * S_tr))) *
                                   (abs(l_tr_mw_in[i]) / (2 * abs(l_tr_mw_in[i])
+ 2 * abs(C_tr_mw_in[i]))))
                                + ((T_tr_mw_rad_out[i+1] - ((C_tr_mw_in[i] * 3600)/
(C_tr * r_tr * B_tr * S_tr))) *
                                   (abs(C_tr_mw_in[i]) / (2 * abs(l_tr_mw_in[i]) + 2
* abs(C_tr_mw_in[i]))))
                                + ((T_tr_mw_rad_out[i+1] - k_tr) *
                                   (abs(l_tr_mw_in[i]) / (2 * abs(l_tr_mw_in[i]) + 2
* abs(C_tr_mw_in[i]))))
                                + ((T_tr_mw_in[i] - k_tr) *
                                   (abs(C_tr_mw_in[i]) / (2 * abs(l_tr_mw_in[i]) + 2
* abs(C_tr_mw_in[i])))))

        for i in range(0, n+1): # inside building calculations
            s_tr_in.insert(i, s_in_tr * S_tr * (T_tr_mw_in[i] - T_tr_in[i]))
            I_tr_in.insert(i, (T_tr_mw_in[i]**4 - T_tr_in[i]**4) * S_tr * R_in)
            C_tr_in.insert(i, (C_w * r_w * B_in * S_in * (T_tr_mw_in[i] -
T_tr_in[i])) / 3600 + ((C_air * r_air * l * w * h) * (T_tr_mw_in[i] - T_tr_in[i]))
/ 3600)
            T_tr_in.insert(i+1, (T_tr_mw_in[i+1] - s_tr_in[i] / (s_in_tr * S_tr))
* (abs(s_tr_in[i]) / (abs(s_tr_in[i]) + abs(C_tr_in[i]) + abs(I_tr_in[i]))) +
(T_tr_mw_in[i+1] - (C_tr_in[i] * 3600) / (C_w * r_w * B_in * S_in + C_air * r_air
* l * w * h)) * (abs(C_tr_in[i]) / (abs(s_tr_in[i]) + abs(C_tr_in[i]) +
abs(I_tr_in[i]))) + (((T_tr_mw_in[i+1])**4 - I_tr_in[i] / (S_tr * R_in))**(1/4)) *
(abs(I_tr_in[i]) / (abs(s_tr_in[i]) + abs(C_tr_in[i]) + abs(I_tr_in[i]))))

        for i in range(0, n): # inside building specific temperature calculations
            T_tr_in_b.insert(i, T_tr_in[i] * S_tr / S_in_b)

        return(T_tr_in_b)


def Trombe_wall_0_temp_calc(B_wd_tr, B_g, B_tr):

    m_wd_tr = r_wd * l * h * B_wd_tr  # Trombe wall glass' mass
    k_wd_tr = ((L_g) / (C_wd * r_wd)) / (2 * (((L_g) / (C_wd * r_wd)) + 1))  # FDM
coefficient for Trombe wall
    R_t_wd = (0.95 ** (B_wd_tr / 0.02))  # radiation coefficient
    L_wd = L_g  # thermal conducivity coefficient
```

```
    T_tr_in_b = list()  # inside building specific temperature (K)

    T_tr_out = list()  # outside temperaure (K)
    T_tr_out.insert(0,
                    (T_street[0] - (((T_street[0]) ** 4) * (R_0 - R_wd_out)) ** (1
/ 4)))  # initial outside temperature (K)
    T_tr_irr = list()  # irradiation temperature
    T_tr_out_w = list()  # out wall temperature including radiation and convection
effects (K)
    s_tr_out = list()  # out wall convection (W)

    T_g_layer = list()  # glass layer temperature (K)
    l_g_layer = list()  # glass layer thermal conductivity (W)
    C_g_layer = list()  # glass layer heat capacity

    T_tr_mw_out = list()  # air gap temperature (K)
    l_tr_mw_out = list()  # air gap thermal conductivity (W)
    C_tr_mw_out = list()  # air gap heat capacity (J)
    I_tr_mw_out = list()  # air gap radiation (W)
    s_tr_mw_out = list()  # air gap convection (W)
    s_ag = list()  # air gap convection coefficient
    T_tr_mw_rad_out = list()  # outside massive wall layer temperature (K)

    T_tr_mw_in = list()  # massive wall inside temperature (K)
    l_tr_mw_in = list()  # massive wall thermal conductivity (W)
    C_tr_mw_in = list()  # massive wall heat capacity

    T_tr_in = list()  # inside building temperature (K)
    s_tr_in = list()  # inside building convection (W)
    I_tr_in = list()  # Inside building radiation (W)
    C_tr_in = list()  # inside building capacity (J)

    T_g_layer.insert(0, T_tr_out[0] + 7)  # glass layer initial temperature (K)
    T_tr_mw_out.insert(0, T_g_layer[0] + 1)  # air gap initial temperature (K)
    T_tr_mw_in.insert(0, T_tr_mw_out[0] + 7)  # massive wall inside initial
temperature (K)
    T_tr_in.insert(0, T_tr_mw_in[0] - k_air)  # inside building initial
temperature (K)

    for i in range(0, n + 5):  # out wall calculations
        s_tr_out.insert(i, s_out * S_tr * (T_street[i] - T_tr_out[i]))
        T_tr_out.insert(i + 1, (T_street[i + 1] - s_tr_out[i] / (s_out * S_tr)))
        T_tr_irr.insert(i, ((T_tr_out[i] ** 4 + (0.05 * I_s[i] * S_tr / 3600) /
(S_tr * R_0)) ** (1 / 4)))
        T_tr_out_w.insert(i, (T_tr_out[i] * (abs(s_tr_out[i]) / (abs(s_tr_out[i])
+ abs((0.05 * I_s[i] * S_tr / 3600)))) + T_tr_irr[i] * (abs((0.05 * I_s[i] * S_tr
/ 3600)) / (abs(s_tr_out[i]) + abs((0.05 * I_s[i] * S_tr / 3600))))))

    for i in range(0, n + 4):  # glass layer calculations
        l_g_layer.insert(i, L_wd * S_tr * (T_tr_out_w[i] - T_g_layer[i]) /
B_wd_tr)
        C_g_layer.insert(i, (C_wd * m_wd_tr * (T_tr_out_w[i] - T_g_layer[i])) /
3600)
        T_g_layer.insert(i + 1, ((T_tr_out_w[i] - ((l_g_layer[i] * B_wd_tr) /
(L_wd * S_tr))) *
                                 (abs(l_g_layer[i]) / (2 * abs(l_g_layer[i]) + 2 *
abs(C_g_layer[i])))) +
                         ((T_tr_out_w[i + 1] - ((C_g_layer[i] * 3600) / (C_wd *
m_wd_tr))) *
                          (abs(C_g_layer[i]) / (2 * abs(C_g_layer[i]) + 2 *
abs(l_g_layer[i])))) +
                         ((T_tr_out_w[i + 1] - k_wd_tr) *
                          (abs(l_g_layer[i]) / (2 * abs(C_g_layer[i]) + 2 *
```

```
abs(l_g_layer[i]))))) +
                          ((T_g_layer[i] - k_wd_tr) *
                           (abs(C_g_layer[i]) / (2 * abs(C_g_layer[i]) + 2 *
abs(l_g_layer[i])))))

    for i in range(0, n + 3):  # second layer calculations
        l_tr_mw_out.insert(i, L_air * S_tr * (T_g_layer[i] - T_tr_mw_out[i]) /
B_g)
        C_tr_mw_out.insert(i, (C_air * r_air * B_g * S_tr * (T_g_layer[i] -
T_tr_mw_out[i]) / 3600))
        I_tr_mw_out.insert(i, (T_g_layer[i] ** 4 - T_tr_mw_out[i] ** 4) * S_tr *
R_wd_in)
        s_ag.insert(i, (0.5 * ((9.81 * 0.0367 *
                                abs(T_tr_mw_out[i] - T_g_layer[i]) * B_g ** 4) /
(8 * 0.000016 ** 2 * h)) ** 0.294))
        s_tr_mw_out.insert(i, s_ag[i] * S_tr * (T_g_layer[i] - T_tr_mw_out[i]))
        T_tr_mw_out.insert(i + 1, ((T_g_layer[i] - ((l_tr_mw_out[i] * B_g) /
(L_air * S_tr))) *
                                   (abs(l_tr_mw_out[i]) /
                                    (2 * abs(l_tr_mw_out[i]) + 2 *
abs(C_tr_mw_out[i]) + abs(s_tr_mw_out[i]) + abs(I_tr_mw_out[i])))) +
                           ((T_g_layer[i + 1] - ((C_tr_mw_out[i] * 3600) / (C_air
* r_air * B_g * S_tr))) *
                            (abs(C_tr_mw_out[i]) /
                             (2 * abs(l_tr_mw_out[i]) + 2 * abs(C_tr_mw_out[i]) +
abs(s_tr_mw_out[i]) + abs(I_tr_mw_out[i])))) +
                           ((T_g_layer[i + 1] - k_air) *
                            (abs(l_tr_mw_out[i]) /
                             (2 * abs(l_tr_mw_out[i]) + 2 * abs(C_tr_mw_out[i]) +
abs(s_tr_mw_out[i]) + abs(I_tr_mw_out[i])))) +
                           ((T_tr_mw_out[i] - k_air) *
                            (abs(C_tr_mw_out[i]) /
                             (2 * abs(l_tr_mw_out[i]) + 2 * abs(C_tr_mw_out[i]) +
abs(s_tr_mw_out[i]) + abs(I_tr_mw_out[i])))) +
                           ((T_g_layer[i + 1] - s_tr_mw_out[i] / (s_ag[i] * S_tr))
*
                            (abs(s_tr_mw_out[i]) /
                             (2 * abs(l_tr_mw_out[i]) + 2 * abs(C_tr_mw_out[i]) +
abs(s_tr_mw_out[i]) + abs(I_tr_mw_out[i])))) +
                           (((T_g_layer[i + 1] ** 4 - I_tr_mw_out[i] / (R_wd_in *
S_tr)) ** (1 / 4)) *
                            (abs(I_tr_mw_out[i]) /
                             (2 * abs(l_tr_mw_out[i]) + 2 * abs(C_tr_mw_out[i]) +
abs(s_tr_mw_out[i]) + abs(I_tr_mw_out[i])))))

        T_tr_mw_rad_out.insert(i, (T_tr_mw_out[i] * ((abs(l_tr_mw_out[i]) +
abs(C_tr_mw_out[i]) + abs(I_tr_mw_out[i]) + abs(s_tr_mw_out[i])) /
(abs(l_tr_mw_out[i]) + abs(C_tr_mw_out[i]) + abs(I_tr_mw_out[i]) +
abs(s_tr_mw_out[i]) + abs((0.05 * R_t_wd * I_s[i] * S_tr / 3600))))) +
(T_tr_irr[i] * (abs((0.05 * R_t_wd * I_s[i] * S_tr / 3600)) / (abs(l_tr_mw_out[i])
+ abs(C_tr_mw_out[i]) + abs(I_tr_mw_out[i]) + abs(s_tr_mw_out[i]) + abs((0.05 *
R_t_wd * I_s[i] * S_tr / 3600))))))

    for i in range(0, n + 2):  # third layer calculations
        l_tr_mw_in.insert(i, L_tr * S_tr * (T_tr_mw_rad_out[i] - T_tr_mw_in[i]) /
B_tr)
        C_tr_mw_in.insert(i, (C_tr * r_tr * B_tr * S_tr * (T_tr_mw_rad_out[i] -
T_tr_mw_in[i])) / 3600)
        T_tr_mw_in.insert(i + 1, ((T_tr_mw_rad_out[i] - ((l_tr_mw_in[i] * B_tr) /
(L_tr * S_tr))) * (
                          abs(l_tr_mw_in[i]) / (2 * abs(l_tr_mw_in[i]) + 2 *
abs(C_tr_mw_in[i])))) + ((T_tr_mw_rad_out[i + 1] - ((C_tr_mw_in[i] * 3600) / (C_tr
* r_tr * B_tr * S_tr))) * (abs(C_tr_mw_in[i]) / (2 * abs(l_tr_mw_in[i]) + 2 *
```

```
abs(C_tr_mw_in[i])))) + ((T_tr_mw_rad_out[i + 1] - k_tr) * ( abs(l_tr_mw_in[i]) /
(2 * abs(l_tr_mw_in[i]) + 2 * abs(C_tr_mw_in[i])))) + ((T_tr_mw_in[i] - k_tr) *
(abs(C_tr_mw_in[i]) / (2 * abs(l_tr_mw_in[i]) + 2 * abs(C_tr_mw_in[i])))))

    for i in range(0, n + 1):  # inside building calculations
        s_tr_in.insert(i, s_in_tr * S_tr * (T_tr_mw_in[i] - T_tr_in[i]))
        I_tr_in.insert(i, (T_tr_mw_in[i] ** 4 - T_tr_in[i] ** 4) * S_tr * R_in)
        C_tr_in.insert(i, (C_w * r_w * B_in * S_in * (T_tr_mw_in[i] - T_tr_in[i]))
/ 3600 + (
                       (C_air * r_air * l * w * h) * (T_tr_mw_in[i] - T_tr_in[i])) /
3600)
        T_tr_in.insert(i + 1, (T_tr_mw_in[i + 1] - s_tr_in[i] / (s_in_tr * S_tr))
* (
                       abs(s_tr_in[i]) / (abs(s_tr_in[i]) + abs(C_tr_in[i]) +
abs(I_tr_in[i]))) + (
                                          T_tr_mw_in[i + 1] - (C_tr_in[i] * 3600) / (
                                              C_w * r_w * B_in * S_in + C_air * r_air * l
* w * h)) * (
                                          abs(C_tr_in[i]) / (abs(s_tr_in[i]) +
abs(C_tr_in[i]) + abs(I_tr_in[i]))) + (
                                          ((T_tr_mw_in[i + 1]) ** 4 - I_tr_in[i] / (S_tr
* R_in)) ** (1 / 4)) * (
                                          abs(I_tr_in[i]) / (abs(s_tr_in[i]) +
abs(C_tr_in[i]) + abs(I_tr_in[i]))))

    for i in range(0, n):  # inside building specific temperature calculations
        T_tr_in_b.insert(i, T_tr_in[i] * S_tr / S_in_b)

    return (T_tr_in_b)

# regular building energy consumptions calculations

def reg_build_en_cons_calc(B_l_w, B_ins_w,
                           B_l_f, B_ins_f,
                           B_l_r, B_ins_r,
                           x,
                           T_h, T_c):

    T_in = list() # inside temperature (K)
    T_n_w_in = list() # inside north wall temperature (K)
    T_e_w_in = list() # inside eas wall temperature (K)
    T_w_w_in = list() # inside west wall temperature (K)
    T_s_w_in = list() # inside south wall temperature (K)
    T_f_in = list() # inside floor temperature (K)
    T_r_in = list() # inside roof temperature (K)
    T_n_wd_in = list() # inside north windows temperature (K)
    T_e_wd_in = list() # inside east windows temperature (K)
    T_w_wd_in = list() # inside west windows temperature (K)
    T_s_wd_in = list() # inside south windows temperature (K)

    W_in_c = list() # cooling degree-seconds
    W_in_h = list() # heating degree-seconds calculations

    T_n_w_in = north_wall_temp_calc(B_ins_w, B_l_w)

    T_e_w_in = east_wall_temp_calc(B_ins_w, B_l_w)

    T_w_w_in = west_wall_temp_calc(B_ins_w, B_l_w)

    T_s_w_in = south_wall_temp_calc(B_ins_w, B_l_w)

    T_f_in = floor_temp_calc(B_ins_f, B_l_f)
```

```
    T_r_in = roof_temp_calc(B_ins_r, B_l_r)

    if S_n_wd > 0:
       T_n_wd_in = north_windows_temp_calc(x)
    else:
        for i in range(0, n):
            T_n_wd_in.insert(i, 0)

    if S_e_wd > 0:
       T_e_wd_in = east_windows_temp_calc(x)
    else:
        for i in range(0, n):
            T_e_wd_in.insert(i, 0)

    if S_w_wd > 0:
       T_w_wd_in = west_windows_temp_calc(x)
    else:
        for i in range(0, n):
            T_w_wd_in.insert(i, 0)

    if S_s_wd > 0:
       T_s_wd_in = south_windows_temp_calc(x)
    else:
        for i in range(0, n):
            T_s_wd_in.insert(i, 0)

    for i in range(0, n): # inside temperature calculations
        T_in.insert(i, T_n_w_in[i] + T_e_w_in[i] + T_w_w_in[i] + T_s_w_in[i] +
T_r_in[i] + T_f_in[i] + T_n_wd_in[i] + T_e_wd_in[i] + T_w_wd_in[i] + T_s_wd_in[i])

    for i in range(0, n):
        if T_in[i] - T_c > 0:
           W_in_c.insert(i, (T_in[i] - T_c) * 3600)
        else: W_in_c.insert(i, 0)

    W_c = sum(W_in_c) * 158.9 / 3600 / 1000 # sum cooling energy consumption

    for i in range(0, n):
        if T_h - T_in[i] > 0:
           W_in_h.insert(i, (T_h - T_in[i]) * 3600)
        else: W_in_h.insert(i, 0)

    W_h = sum(W_in_h) * 158.9 / 3600 / 1000 # sum heating energy consumption

    W_sum = W_c + W_h # sum energy consumption

    return(W_c, W_h, W_sum, T_in)


# Trombe wall building energy consumptions calculations

def TW_build_en_cons_calc(B_l_w, B_ins_w,
                          B_l_f, B_ins_f,
                          B_l_r, B_ins_r,
                          x,
                          B_wd_tr, B_g, B_tr,
                          T_h, T_c):

    T_in = list()  # inside temperature (K)
    T_n_w_in = list()  # inside north wall temperature (K)
    T_e_w_in = list()  # inside eas wall temperature (K)
    T_w_w_in = list()  # inside west wall temperature (K)
    T_tr_w_in = list() # inside Trombe wall temperature (K)
```

```
T_tr_w_in_0 = list()  # inside isolated Trombe wall with temperature (K)
T_f_in = list()  # inside floor temperature (K)
T_r_in = list()  # inside roof temperature (K)
T_n_wd_in = list()  # inside north windows temperature (K)
T_e_wd_in = list()  # inside east windows temperature (K)
T_w_wd_in = list()  # inside west windows temperature (K)

W_in_c = list()  # cooling degree-seconds
W_in_h = list()  # heating degree-seconds calculations

T_n_w_in = north_wall_temp_calc(B_ins_w, B_l_w)
T_e_w_in = east_wall_temp_calc(B_ins_w, B_l_w)
T_w_w_in = west_wall_temp_calc(B_ins_w, B_l_w)
T_f_in = floor_temp_calc(B_ins_f, B_l_f)
T_r_in = roof_temp_calc(B_ins_r, B_l_r)
T_tr_w_in = Trombe_wall_temp_calc(B_wd_tr, B_g, B_tr)
T_tr_w_in_0 = Trombe_wall_0_temp_calc(B_wd_tr, B_g, B_tr)
if S_n_wd > 0:
    T_n_wd_in = north_windows_temp_calc(x)
else:
    for i in range(0, n):
        T_n_wd_in.insert(i, 0)
if S_e_wd > 0:
    T_e_wd_in = east_windows_temp_calc(x)
else:
    for i in range(0, n):
        T_e_wd_in.insert(i, 0)
if S_w_wd > 0:
    T_w_wd_in = west_windows_temp_calc(x)
else:
    for i in range(0, n):
        T_w_wd_in.insert(i, 0)

for i in range(0, n): # solar insulation immitation
    if (T_n_w_in[i] + T_e_w_in[i] + T_w_w_in[i] +
        T_tr_w_in[i] + T_r_in[i] + T_f_in[i] +
        T_n_wd_in[i] + T_e_wd_in[i] + T_w_wd_in[i]) > T_c:
        T_in.insert(i, (T_n_w_in[i] + T_e_w_in[i] + T_w_w_in[i] +
                        T_tr_w_in_0[i] + T_r_in[i] + T_f_in[i] +
                        T_n_wd_in[i] + T_e_wd_in[i] + T_w_wd_in[i]))
    else:
        T_in.insert(i, (T_n_w_in[i] + T_e_w_in[i] + T_w_w_in[i] +
                        T_tr_w_in[i] + T_r_in[i] + T_f_in[i] +
                        T_n_wd_in[i] + T_e_wd_in[i] + T_w_wd_in[i]))

for i in range(0, n): # degree-seconds method (cooling)
    if T_in[i] - T_c > 0:
        W_in_c.insert(i, (T_in[i] - T_c) * 3600)
    else:
        W_in_c.insert(i, 0)

W_c = sum(W_in_c) * 158.9 / 3600 / 1000  # sum cooling energy consumption

for i in range(0, n): # degree-seconds method (heating)
    if T_h - T_in[i] > 0:
        W_in_h.insert(i, (T_h - T_in[i]) * 3600)
    else:
        W_in_h.insert(i, 0)

W_h = sum(W_in_h) * 158.9 / 3600 / 1000  # sum heating energy consumption

W_sum = W_c + W_h  # sum energy consumption (kWh)
```

```
    return (W_c, W_h, W_sum, T_in)

# Optimization

def Optimization(B_l_w,
                 B_ins_w,
                 B_l_f,
                 B_ins_f,
                 B_l_r,
                 B_ins_r,
                 x,
                 T_h,
                 T_c):

    Opt_wd = list() # Energy consumption with different Trombe wall window width
    B_wd_tr = list([0.04, 0.06, 0.08, 0.1, 0.12]) # Trombe wall window width

    Opt_g = list() # Energy consumption with different Trombe wall air gap width
    B_g = list([0.05, 0.1, 0.15, 0.2, 0.25]) # Trombe wall air gap width

    Opt_w = list() # Energy consumption with different Massive wall width
    B_tr = list([0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6]) # Massive
wall width

    Opt_wd.insert(0, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                            x,
                                            B_wd_tr[0], B_g[2], B_tr[5],
                                            T_h, T_c)[2]))
    Opt_wd.insert(1, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                            x,
                                            B_wd_tr[1], B_g[2], B_tr[5],
                                            T_h, T_c)[2]))
    Opt_wd.insert(2, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                            x,
                                            B_wd_tr[2], B_g[2], B_tr[5],
                                            T_h, T_c)[2]))
    Opt_wd.insert(3, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                            x,
                                            B_wd_tr[3], B_g[2], B_tr[5],
                                            T_h, T_c)[2]))
    Opt_wd.insert(4, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                            x,
                                            B_wd_tr[4], B_g[2], B_tr[5],
                                            T_h, T_c)[2]))

    B_wd_tr_opt = B_wd_tr[(Opt_wd.index(min(Opt_wd)))]

    Opt_g.insert(0, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                           x,
                                           B_wd_tr_opt, B_g[0], B_tr[5],
                                           T_h, T_c)[2]))
    Opt_g.insert(1, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                           x,
                                           B_wd_tr_opt, B_g[1], B_tr[5],
                                           T_h, T_c)[2]))
    Opt_g.insert(2, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
```

```
B_ins_r,
                                          x,
                                          B_wd_tr_opt, B_g[2], B_tr[5],
                                          T_h, T_c)[2]))
    Opt_g.insert(3, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                          x,
                                          B_wd_tr_opt, B_g[3], B_tr[5],
                                          T_h, T_c)[2]))
    Opt_g.insert(4, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                          x,
                                          B_wd_tr_opt, B_g[4], B_tr[5],
                                          T_h, T_c)[2]))

    B_g_opt = B_g[(Opt_g.index(min(Opt_g)))]

    Opt_w.insert(0, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                          x, B_wd_tr_opt, B_g_opt, B_tr[0], T_h,
T_c)[2]))
    Opt_w.insert(1, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                          x, B_wd_tr_opt, B_g_opt, B_tr[1], T_h,
T_c)[2]))
    Opt_w.insert(2, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                          x, B_wd_tr_opt, B_g_opt, B_tr[2], T_h,
T_c)[2]))
    Opt_w.insert(3, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                          x, B_wd_tr_opt, B_g_opt, B_tr[3], T_h,
T_c)[2]))
    Opt_w.insert(4, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                          x, B_wd_tr_opt, B_g_opt, B_tr[4], T_h,
T_c)[2]))
    Opt_w.insert(5, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                          x, B_wd_tr_opt, B_g_opt, B_tr[5], T_h,
T_c)[2]))
    Opt_w.insert(6, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                          x, B_wd_tr_opt, B_g_opt, B_tr[6], T_h,
T_c)[2]))
    Opt_w.insert(7, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                          x, B_wd_tr_opt, B_g_opt, B_tr[7], T_h,
T_c)[2]))
    Opt_w.insert(8, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                          x, B_wd_tr_opt, B_g_opt, B_tr[8], T_h,
T_c)[2]))
    Opt_w.insert(9, (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r,
B_ins_r,
                                          x, B_wd_tr_opt, B_g_opt, B_tr[9], T_h,
T_c)[2]))

    B_tr_opt = B_tr[(Opt_w.index(min(Opt_w)))]

    Opt_C = (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r, B_ins_r,
                                   x,
                                   B_wd_tr_opt, B_g_opt, B_tr_opt,
```

```
                                T_h, T_c)[0])
Opt_H = (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r, B_ins_r,
                                x,
                                B_wd_tr_opt, B_g_opt, B_tr_opt,
                                T_h, T_c)[1])
Opt_O = (TW_build_en_cons_calc(B_l_w, B_ins_w, B_l_f, B_ins_f, B_l_r, B_ins_r,
                                x,
                                B_wd_tr_opt, B_g_opt, B_tr_opt,
                                T_h, T_c)[2])

return B_wd_tr_opt, B_g_opt, B_tr_opt, Opt_C, Opt_H, Opt_O
```