Thermische Modellierung eines Passiv-Dämmhauses und integriertem Wärmebedarfsrechner

Bachelorarbeit zur Erlangung des Bachelor-Grades
Bachelor of Engineering im Studiengang Elektrotechnik
an der Fakultät für Informations-, Medien- und Elektrotechnik
der Technischen Hochschule Köln

vorgelegt von:	Andreas Kurz
Matrikel-Nr.:	11129904
Adresse:	Neuhöfferstraße 19
	50679 Köln
	Andreas.Kurz@smail.th-koeln.de

eingereicht bei:	Prof. Dr. Eberhard Waffenschmidt
Zweitgutachter:	Francisco Antonio Carrasco Serrano

Köln, 30.09.2023

Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbstständig verfasst zu haben. Alle Stellen, die wortwörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer oder der Verfasserin/des Verfassers selbst entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Ort, Datum	Rechtsverbindliche Unterschrift

Kurzfassung

Aufgrund der wahrscheinlich wachsenden Bevölkerungszahl in Deutschland und einem akuten Mangel an umweltschonenden Energieproduktionen wird der Bedarf an energieeffizienten und somit kostengünstigen Wohnraum in den kommenden Jahren zunehmen. [1]

Mit Erstellung dieser Bachelorarbeit soll gezeigt werden, dass mit Hilfe von selbsterstellten thermischen Simulationsmodellen die Temperatur der Raumluft eines Gebäudes, anhand verschiedener Faktoren, zielgenau berechnet werden kann und das diese Programme gegenüber kommerziellen Programmen einige Vorteile aufweisen. Mit thermischen Modellen ist man in der Lage, energieeffiziente Gebäude zu entwerfen und die möglichen Heizkosten vorab präzise berechnen zu können.

Für das Erstellen des Programms, bzw. der Algorithmen wurde die Entwicklungsumgebung Visual Studio Code verwendet. Als Programmiersprache dient Python, weil ein Programm, welches wiederum das thermische Simulationsprogramm aufruft, in Python geschrieben ist. Die Übergabeparameter wurden in einer Excel-Datei erstellt, welches mit Hilfe eines Frameworks namens Pandas eingelesen und verarbeitet wird.

Nach Fertigstellung des Programms sind die Raumtemperaturen recht präzise berechnet worden, zudem konnte der Heizbedarf, welcher benötigt wird um mit Hilfe von Heizkörpern eine vordefinierte Raumtemperatur zu erreichen, berechnet werden. Des weiteren ist das Programm erweiterbar, d.h. man kann es an neue Bedürfnisse des Anwenders anpassen und modifizieren.

Abstract

Due to an growing population of the humans in the world and a current shortage of environmentally friendly energy productions, the need of energy-efficient and thus affordable housing will increase in the following years. [1]

This bachelorthesis shall show that with the help of self-writtened thermal-simulation programs, the temperature of a building can be calculated accurate on the basis of various factors and that these programmes have some advantages over commercial programmes. With thermal simulated programs, the people are able to design energy-efficient buildings and to calculate the possible heating costs very precisely in advance.

The software called Visual Studio Code was used to create the programme and the algorithms. Python was used as the programming language because a program that in turn calls the thermal simulation program is written in Python. The parameters were created in an Excel file, which is read in and processed with the help of a framework called Pandas.

After completion of the programme, the room temperatures were calculated quite precisely, and it was also possible to calculate the heating requirement needed to achieve a predefined room temperature with the help of radiators. In addition, the program is expandable, this means it can be adapted and modified to meet the user's new needs.

Inhaltsverzeichnis

1 Einführung

Thermische Modulationsprogramme stellen ein wichtiges Instrument zur Erfassung von thermischen Prozessen dar. Diese Programme beinhalten das Potential, die Art und Weise zu revolutionieren, wie sich Gebäude jeglicher Art konstruieren, nutzen und effizienter gestalten lassen. Thermische Simulationsprogramme nutzen physikalische Eigenschaften, sowie Gegebenheiten, um die Raumtemperatur- und die Energieverwaltung in Gebäuden zu verbessern, den Komfort für die Bewohner zu erhöhen, die Produktion bei temperaturabhängigen Prozessen zu gewährleisten. Es entstehen eine Vielzahl von ökonomischen als auch ökologischen Vorteilen.

Ein Ziel von thermischen Simulationsprogrammen ist es, Wärmeübertragungsprozesse im Zusammenhang mit physikalischen Eigenschaften von Materialien und Komponenten besser zu verstehen und vorhersagen zu können. Solche berechneten Modelle spielen eine entscheidende Rolle in der Vorhersage eines Verhaltens von Systemen, der Elektronikproduktion, in der Verfahrenstechnik oder der Sensorik, unter verschiedenen, zum Teil auch dynamischen Bedingungen. [2]

Einer der bedeutendsten Vorteile eines thermischen Simulationsprogramms ist die Verbesserung der Energieeffizienz. Durch die Berechnung der zu erwarteten Raumtemperatur können Heizungs- und Kühlsysteme präzise Gesteuert werden und der Energiebedarf eines Gebäudes dadurch reduziert werden. Niedrige Betriebskosten sind dabei nur ein Vorteil für die Bewohner des Gebäudes, durch reduzierte CO_2-Emissionen aufgrund des verminderten Energiebedarfs sind thermische Simulationsprogramme ein wichtiges Werkzeug im Kampf gegen den Klimawandel.

1.1 Motivation

Im Jahr 2022 gaben deutsche Haushalte durchschnittlich 27,8 % ihres Einkommens für Wohnkosten aus. [3] Diese Zahl verdeutlicht die wachsende Herausforderung bezahlbaren Wohnraums. Doch hier bieten sich Möglichkeiten zur Besserung: Thermische Simulationsprogramme können maßgeblich zur Reduzierung der Energiebilanz beitragen. Indem diese Programme zum Einsatz kommen, eröffnen sich Potenziale, die Wohnkosten zu senken und erschwingliches Wohnen zu fördern.

Besonders beunruhigend ist, dass gegenwärtig 3,1 Millionen Haushalte sogar über 40 Prozent ihres Einkommens für die Miete aufbringen müssen. [4] Diese Situation verdeutlicht die Dringlichkeit, nach nachhaltigen Lösungen zu suchen. Hierbei können thermische Simulationsprogramme eine Schlüsselrolle spielen. Sie ermöglichen es, Wohngebäude energetisch zu optimieren, was nicht nur die Energiebilanz, sondern auch die Wohnkosten positiv beeinflusst.

Die fortschreitende Nutzung dieser Simulationsprogramme kann zu einer erfreulichen Entwicklung führen: Wohnen wird nicht nur bezahlbarer, sondern auch umweltfreundlicher. Diese Programme tragen dazu bei, die CO_2-Emissionen zu reduzieren, indem sie aufzeigen, wie Wohngebäude effizienter beheizt und gekühlt werden können. Insbesondere die Verbrennung von Brennstoffen zur Energieerzeugung stellt einen der größten Verursacher von CO_2-Emissionen dar. [5]

Durch eine verbesserte CO_2-Bilanz und energieeffiziente Maßnahmen tragen thermische Simulationsprogramme zur Schaffung einer nachhaltigen Wohnkultur bei. Sie bieten nicht nur wirtschaftliche Vorteile, sondern tragen auch zur Umweltschonung bei. Die Zusammenführung dieser Faktoren eröffnet die Möglichkeit, einen Weg zu finden, der nicht nur den Wohnkomfort steigert, sondern auch eine positive Auswirkung auf den Geldbeutel und die Umwelt hat.

Die Motivation dieser Bachelorarbeit ist es, die Bedeutung von thermischen Simulationsmodellen zu fördern, um bezahlbaren und klimafreundlichen Wohnraum zu ermöglichen.

1.2 Aufgabe und Aufbau dieser Arbeit

Mit Erstellung dieser Bachelorarbeit soll gezeigt werden, dass man für eine Berechnung der Raumtemperatur und einer benötigten Heizleistung, eigene Algorithmen erstellen kann, welche sich schnell modifizieren lassen und in anderen Algorithmen leicht einbauen lassen. Da die Einarbeitung in frei erwerblichen Programmen, welche für die thermische Modellierung von Gebäuden verwendet werden, für Laien kompliziert und unhandlich erscheinen mögen, soll gezeigt werden, dass die gewünschten Funktionen maßgeschneidert erstellt werden können. Dabei werden die Vorteile der Hochsprache Python, sowie der Entwicklungsumgebung Visual Studio Code erläutert. Diese Bachelorarbeit ist dabei in folgende Gliederpunkte aufgeteilt:

Einführung

Aufgabe ist es, die in meinem Exposé genannte Forschungsfrage zu beantworten:

„Wie und mit welchen Parametern lässt sich ein Programm zur thermischen Modellierung eines Hauses erstellen und welche Informationen lassen sich daraus zur Optimierung der Energieeffizienz herauslesen?“

Grundlagen der thermischen Übertragungswege

In diesem Kapitel werden die drei thermischen Übertragungswege erläutert. Das von mir erstellte Programm umfasst dabei die Konvektion und Strahlung. Als Beispiel soll ein stehendes Passiv-Dämmhaus verwendet werden, welches über die Außentemperatur und den Sonnenstrahlen Wärmeenergie aufnimmt, bzw. abgibt. Da in diesem Beispiel kein Wärmetransport von einem ruhenden Körper zu einem anderen ruhenden Körper erfolgt, wird die Konduktion nur beschrieben, findet für die Berechnung aber keine Beachtung.

Thermische Modelle

Anschließend werden die Parameter genannt und beschrieben, welche für die Berechnungen angewendet werden. Zudem werden spezifische Materialeigenschaften, wie der U-Wert, erläutert. Dieses Kapitel soll dem Leser die Möglichkeit bieten, den Umfang, also die benötigten Werte, welche für die Berechnungen essentiell sind, zu verdeutlichen.

Neues Design

Dies stellt den eigentlichen Hauptteil der Bachelorarbeit dar. Hier werden die Algorithmen genant, anhand derer die Temperaturen eines Gebäudes berechnet werden. Es wird genauer auf die Programmiersprache Python, die verwendete Entwicklungsumgebung, sowie auf die Vor- und Nachteile, die ein selbsterstelltes Programm besitzt, eingegangen. Ein Flussdiagramm visualisiert dabei die Logik, die das Programm besitzt.

Fazit

Abschließend werden die Ergebnisse der Algorithmen analysiert, dabei auf mögliche Schwachstellen eingegangen und letztlich eine Gegenüberstellung mit einem anderem Programm erstellt.

2 Thermische Übertragung

2.1 Grundlagen

Die Wärmeübertragung bildet ein fundamentales Konzept für das Verständnis der Energieübertragung in verschiedenen Medien. Es beschreibt die Übertragung von thermischer Energie von einem Ort höherer Temperatur zu einem Ort niedriger Temperatur. [6] Die Übertragung von thermischer Energie spielen in vielen Aspekten der natürlichen Umwelt eine entscheidende Rolle. In technologischen Konstruktionen wie der Isolierung von Gebäuden, in mechanischen Bauteilen wie Motoren und Maschinen, oder der Kühlung von Elektronikkomponenten wie Leiterplatten. Die Kenntnis der Wärmeübertragungsmechanismen ist entscheidend, um effiziente, langlebige und energieärmere Bauteile und Gebäude zu entwickeln. [7] Dabei lässt sich der Transfer von Wärme in drei Mechanismen unterteilen: [13]

- Konduktion
- Konvektion
- Strahlung

Die Mechanismen arbeiten oft in Kombination zueinander. Ein tiefes Verständnis dieser Mechanismen ermöglicht es uns, Energie effizienter zu nutzen, technologische Prozesse zu optimieren und eine Vielzahl von Anwendungen in Wissenschaft und Technik zu erstellen.

2.2 Thermische Übertragungswege

Bevor die drei Arten des Wärmetransport erläutert werden, sollten die Definition der Begriffe Temperatur und Wärme genannt werden. Die Temperatur ist ein Maß für die mittlere kinetische Energie von Teilchen in Gasen, Flüssigkeiten und Festkörpern. [8] Die SI-Einheit ist Kelvin [K]. Kelvin wird als Temperaturintervall verwendet. Diese Temperaturskala ist die Grundlage aller Temperaturmessungen. [9] Dabei sind 0 [K] der absolute Nullpunkt, bei dem sich Teilchen nicht mehr bewegen, obwohl Eigenfluktuationen dies Verhindern, weshalb der absolute Nullpunkt nie erreicht werden kann. 0 [K] sind zugleich -273,15 °Celsius. [10] Kelvin ist über die Boltzmann-Konstante definiert. [11] Ein Kelvin ist die Änderung der Temperatur, die wiederum einer Änderung der thermischen Energie um $1.38065 * 10^{-23}$ Joule entspricht. [12] Wärme ist somit ein Maß für die Energie, die benötigt wird, um die Temperatur eines Mediums zu ändern.

2.2.1 Konduktion

Wärmeleitung, auch Konduktion genannt, beschreibt den Prozess, bei dem Wärmeenergie durch direkten Kontakt zwischen Teilchen eines Materials übertragen wird, ohne dass die Teilchen selbst ihre Position wesentlich verändern. Die Wärme wird dabei von den schnell bewegten Teilchen in Regionen mit niedrigerer Energie, also einer niedrigeren Temperatur, weitergegeben. Diese thermische Transportmethode basiert auf der Bewegung von Teilchen, die ihre kinetische Energie übertragen, um ein Temperaturgefälle auszugleichen. Die Konduktion spielt in zahlreichen natürlichen und technischen Bereichen eine entscheidende Rolle. [13]

Jedes Material besitzt eine Wärmeleitfähigkeit. Es beschreibt die Fähigkeit, einen Wärmestrom, wie den elektrischen Strom durch el. Leiter, zu leiten. [14] Metalle zählen dabei zu den guten Wärmeleitern, sie leiten die Wärme sehr schnell und effizient von einem Ort höherer Temperatur, zu einem Ort mit einer niedrigeren Temperatur. Dagegen zählt z.B. Holz, Glas oder Luft zu den schlechten Wärmeleitern und könnten als Dämmstoffe verwendet werden. [15] Berechnet wird die Wärmeleitfähigkeit oftmals anhand folgender Formel: [37]

$$k = \frac{Q \times L}{A \times t \times (T2 - T1)}$$

Wärmeleitfähigkeit

Q ist hierbei die Übertragene Wärmeenergie in Joule [J], L die Länge des Materials in Metern [m], A die Querschnittsfläche in m², t dieund T_1 und T_2 die unterschiedlichen Temperaturen. Mit dieser Gleichung lässt sich die Wärmeleitfähigkeit eines Materials zu berechnen.

Die gute Wärmeleitfähigkeit von Metallen lässt sich z.B. beim morgendlichen trinken von Kaffee und/oder Tee spüren, wobei ein zunächst kühler Metalllöffel beim eintauchen in das heiße Getränk warm wird und sich weiterhin erhitzt, bis die Temperatur des Getränks mit der des Löffels identisch sind, ein thermisches Gleichgewicht demnach erreicht ist.

Technische Anwendungen der Wärmeleitung sind häufig zu finden. In der Elektronikindustrie, z.B. bei Halbleitern sind Wärmeleiter und Kühlkörper essentiell, um die Wärme von Mikrochips und Prozessoren in die Umgebungsluft abzuleiten, um Überhitzungen zu vermeiden. Im Gebäudebau dienen thermische Isolationsmedien wie Holz, Styropor, Zellulose und Hanffasern dem Temperaturerhalt im Gebäude. [16]

Zusammenfassend ist die Wärmeleitung ein wichtiger physikalischer Prozess, der die Übertragung von Wärmeenergie in Materialien beschreibt. Ihre Bedeutung reicht von alltäglichen Erfahrungen bis hin zu entscheidenden Anwendungen in verschiedenen Technologiebereichen. Die Formeln und Konzepte der Wärmeleitung ermöglichen es, diese Erscheinung zu verstehen und effektiv in vielen Bereichen einzusetzen.

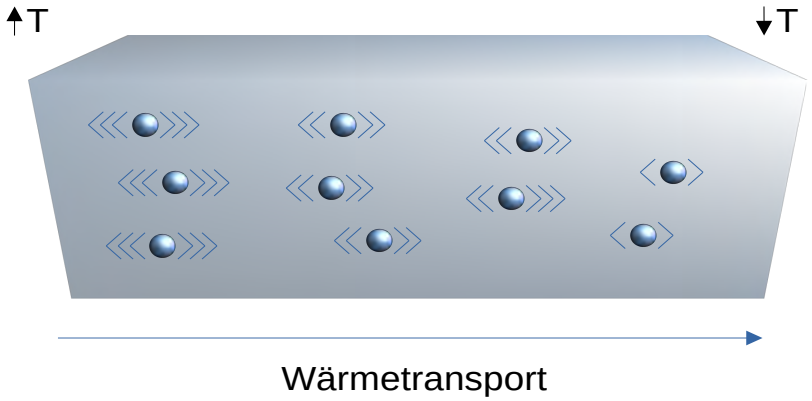

Abbildung 1: Konduktion

Quelle: Eigene Darstellung

2.2.2 Konvektion

Die Konvektion ist ein Prozess der Wärmeübertragung, der in der Natur und in technologischen Anwendungen eine bedeutende Rolle spielt. [17] Sie ist eng mit den Bewegungen von Flüssigkeiten und Gasen verbunden und veranschaulicht, wie Wärmeenergie in Strömungen transportiert wird.

Im Gegensatz zur Wärmeleitung, die auf direktem Kontakt zwischen Teilchen in Materialien basiert, und zur Strahlung, die ohne Medium auskommt, erfordert die Konvektion das Vorhandensein von ganzer Materie in Bewegung. Dieser Prozess ist in vielen natürlichen Phänomenen präsent, wie z.B. in Winden, Meeresströmungen und atmosphärischer Zirkulation. [18]

Die Konvektion kann in zwei Haupttypen unterteilt werden: natürliche Konvektion und erzwungene Konvektion.

Natürliche Konvektion, auch freie Konvektion genannt, tritt auf, wenn sich eine Flüssigkeit oder ein Gas aufgrund von Temperaturunterschieden selbst in Bewegung

setzt. Ein bekanntes Beispiel ist das erhitzen von Wasser in einem Kochtopf. [19] Aufgrund der Erwärmung des Wassers erhöht sich die kinetische Energie der Teilchen im Wasser, weshalb das Volumen größer wird und bei gleichbleibender Anzahl an Wassermolekülen die Dichte somit abnimmt. Warmes Wasser besitzt demnach eine geringere Dichte als kälteres Wasser, gefrorenes Wasser ist hierbei ausgenommen. Dies hat zur Folge, das sich warmes Wasser über kaltem Wasser befindet, warmes Wasser steigt also auf. Dies führt zu einer zirkulären Bewegung, Konvektionsstrom genannt.

Erzwungene Konvektion hingegen tritt auf, wenn die Bewegung eines Mediums bzw. Anteile davon, durch äußere Mechanismen ausgelöst wird. [19] Dies kann z.B. durch eine Pumpe oder einen Ventilator herbeigeführt werden. Die Klimaanlage eines Kraftfahrzeugs kann hierfür als Beispiel dienen. Ein Ventilator sorgt dafür, das Luft über einen Kühler hinweggeführt wird, wodurch die Wärme von der Kühlflüssigkeit abgeführt wird.

Um das Ausmaß an Konvektion angeben zu können, werden verschiedene Gleichungen verwendet. Am bekanntesten dient aber folgende Gleichung für den Wärmetransport: [38]

$$Q = h \times A \times (T_2 - T_1)$$

Wärmetransport

Q stellt hierbei die übertragene Wärme in Joule [J], h den U-Wert in $\frac{W}{m^2 \times K}$, A die Fläche in m^2 und T_2 - T_1 den Temperaturunterschied in Kelvin dar.

In der Technologie spielt Konvektion eine entscheidende Rolle. In Heizungsanlagen werden Konvektionsströme genutzt, um warme Luft durch Räume zu zirkulieren. In der Kältetechnik werden Kühlflüssigkeiten verwendet, um Wärme von einem Ort zu einem anderen zu transportieren.

Im Alltag begegnet uns Konvektion ebenfalls häufig. Ein weiteres Beispiel sind Meeresströmungen, die durch Temperaturunterschiede angetrieben werden und einen Einfluss auf das Klima haben. Das Aufsteigen warmer Luft und das Absteigen kühlerer Luft in Räumen sind ebenfalls Beispiele für natürliche Konvektion.

Zusammenfassend ist die Konvektion ein wesentlicher Mechanismus zur Übertragung von Wärmeenergie durch Bewegung von Fluiden. Sie manifestiert sich in natürlicher und erzwungener Form und spielt eine bedeutende Rolle in Technologie und Alltag, indem sie dazu beiträgt, Wärmeausgleich und Fluidbewegungen zu ermöglichen.

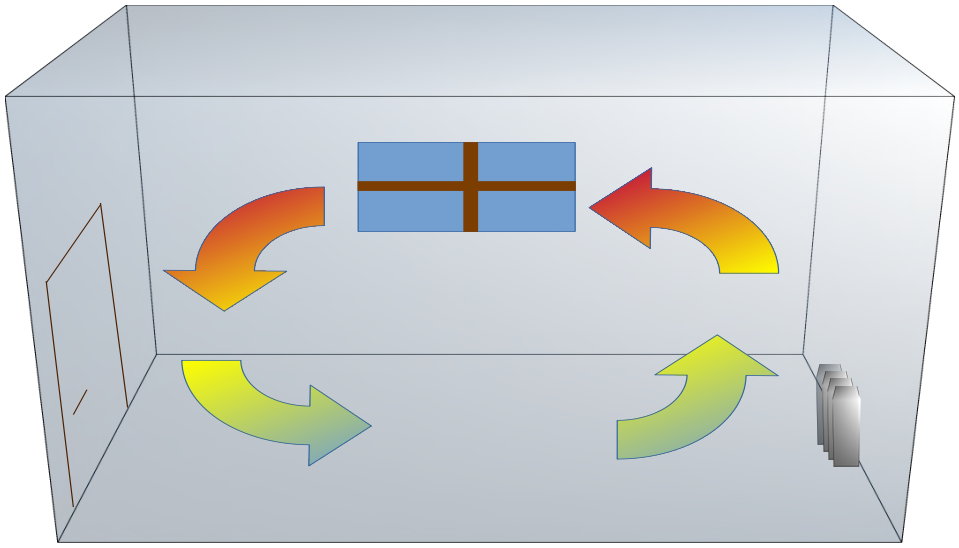

Abbildung 2: Konvektion

Quelle: Eigene Darstellung

2.2.3 Wärmestrahlung

Die Wärmeübertragung durch Strahlung ist, anders als die Konduktion und Konvektion, ein Prozess, der ganz ohne Materie fungiert. Es basiert auf Emission und Absorption elektromagnetischer Wellen und kann sich somit selbst in einem Vakuum ausbreiten.

Ein materieller Träger ist also nicht von Nöten. [20] Die Wärmestrahlung ist in zahlreichen Bereichen des Alltags von großer Bedeutung.

Strahlung bezieht sich auf die Übertragung von Energie in Form elektromagnetischer Wellen, wobei die Wärmeübertragung durch Infrarotstrahlung besonders relevant ist. Alle Teilchen mit einer Temperatur über dem absoluten Nullpunkt von 0 °K oder -273,15°C, emittieren aufgrund ihrer Eigenbewegung elektromagnetische Strahlung. [20] Die Strahlungsenergie ist also abhängig von der kinetischen Energie und somit von der Temperatur eines Gegenstandes. [21]

Je höher die Temperatur, desto mehr Strahlungsenergie wird emittiert. Diese Infrarotwellen breiten sich mit Lichtgeschwindigkeit aus und können von Materialien absorbiert werden, wodurch sich die Temperatur des Gegenstandes erhöht, oder aber auch reflektiert oder durchgelassen werden. [20]

Im Alltag tritt der Wärmetransport durch Strahlung in zahlreichen Bereichen und Technologien auf. Ein klassischen Beispiel sind die Sonnenstrahlen, welche auf die Erdoberfläche treffen und die Erde erwärmt. Abgesehen von der natürlichen Wärme wird dies genutzt, um Strom aus Solaranlagen zu gewinnen. In der Medizintechnik wird Strahlungsenergie verwendet, um bei Bildgebenden Verfahren, wie der Infrarot Thermografie, wird Strahlung verwendet, um unterschiedliche Gewebe anhand deren unterschiedlichen Temperatur sichtbar zu machen.

In der Bautechnik besitzt die Wärmeübertragung durch Infoartstrahlung ebenfalls eine besondere Bedeutung. Baustoffe, die Strahlung gut reflektieren und kaum absorbieren, dienen als Wärmedämmstoffe. [22]

Die Strahlungswärmeübertragung bleibt ein aktives Forschungsgebiet mit ständiger Weiterentwicklung und Innovation. Neue Materialien, die die Emission oder Absorption von Strahlung beeinflussen, werden erforscht, um effizientere Wärmedämmung oder Energieerzeugung zu ermöglichen. Fortschritte in der Strahlungswärmeübertragung haben das Potenzial, das Verständnis der Energieübertragung zu vertiefen und neue Technologien zu schaffen.

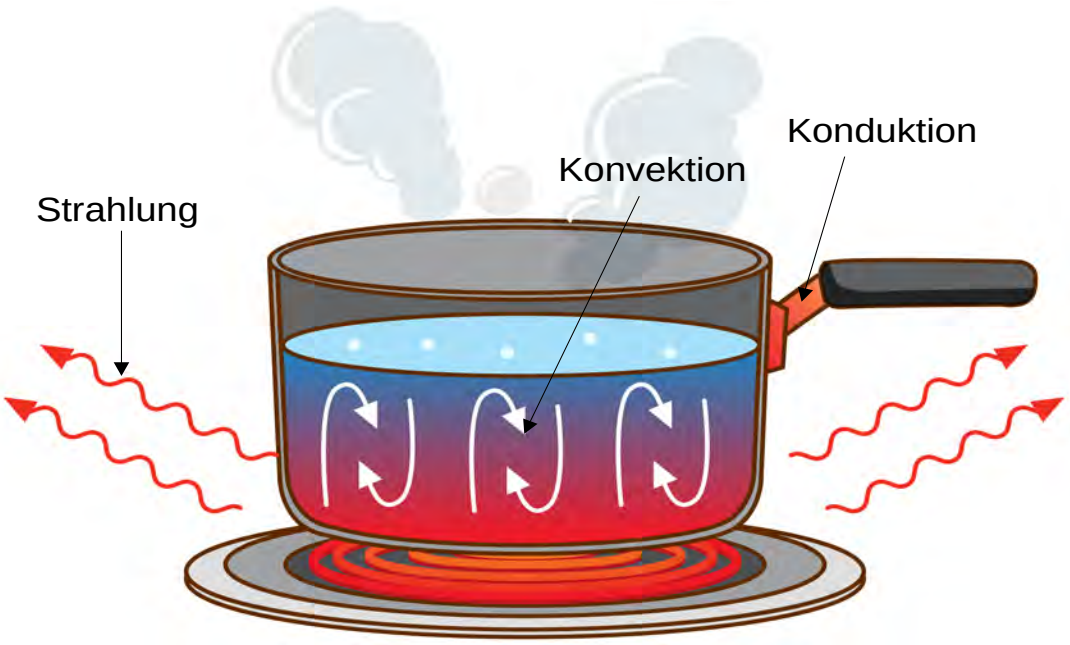

Abbildung 3: Wärmestransport

Quelle: In Anlehnung an https://de.freepik.com/vektoren-kostenlos/topf-mit-kochendem-wasser-auf-induktionsherd-anfassen_20721623.htm#query=hei%C3%9Fer%20Topf&position=17&from_view=search&track=ais

3 Parameter der thermischen Modelle

Für möglichst exakte Werte zur Berechnung von Raumtemperaturen und einer benötigten Heizleistung, werden verschiedene Parameter benötigt. Da es eine sehr große Anzahl an Faktoren gibt, die die Raumtemperatur beeinflussen, wurde mein thermisches Simulationsprogramm so programmiert, das es mit den meiner Meinung nach wichtigsten Parametern auskommt. Diese Parameter werden aus einer Excel-Datei eingelesen, welche im Nachhinein genauer erläutert wird. Man kann die Werte anhand zeitabhängigen und zeitunabhängigen Parametern gruppieren.

3.1 Zeitunabhängige Parameter

Zeitunabhängige Parameter spielen eine entscheidende Rolle in thermischen Gebäudesimulationen, da sie dazu beitragen, das Verhalten von Gebäuden unter verschiedenen Bedingungen zu analysieren, ohne die Komplexität einer zeitlichen Dynamik berücksichtigen zu müssen. Diese Parameter sind essenziell, um das thermische Verhalten eines Gebäudes zu verstehen, seine Energieeffizienz zu bewerten und Optimierungen im Design und Betrieb vorzunehmen.

Ein zentraler zeitunabhängiger Parameter ist das Baumaterial des Gebäudes. Es umfasst Gruppierungen wie Wände, Fenster, Dach und Boden, die den Wärmeaustausch zwischen dem Gebäudeinneren und der Umwelt beeinflussen. Der U-Wert, auch als Wärmedurchgangskoeffizient bekannt, ist ein Schlüsselparameter, der angibt, wie viel Wärme durch eine bestimmte Fläche der Gebäudehülle fließt. [23] Dabei bedeutet ein niedriger U-Wert eine gute Wärmedämmung, was wiederum eine effizientere Nutzung des Heiz- und Kühlsystems zur Folge haben kann, da es weniger Interaktionen mit der äußeren Luft gibt.

Die physikalischen Eigenschaften der verwendeten Baumaterialien eines Gebäudes sind ausschlaggebend für die Raumtemperatur. Die Wärmekapazität und die Wärmeleitfähigkeit eines Stoffes wie Beton, Ziegelstein oder Dämmstoffe wie Polyurethan-Hartschaum bestimmen das Speichervermögen von Wärme, sowie die Eigenschaft, Energie in Form von Wärme kaum durch das Material hindurch fließen zu lassen. Temperaturschwankungen können damit reduziert werden und Heiz- und Kühlprozesse angepasst, also effizienter Umgesetzt werden.

Die Ausrichtung, die Form und die Lage eines Hauses hat ebenfalls einen Einfluss auf die Raumtemperatur. Die Ausrichtung eines Gebäudes, sowie die Lage haben einen Einfluss auf die Menge an Solarstrahlung, welche auf das Gebäude auftreffen. [24] Geometrische Aspekte, wie die Raumgröße und Form des Gebäudes, beeinflussen das Luftvolumen, welches auf einer bestimmten Temperatur gehalten werden soll. Zudem beeinflusst die Form eines Raumes ebenfalls die Luftzirkulation und damit direkt die Wärmeverteilung.

In thermischen Gebäudesimulationen werden diese zeitunabhängigen und somit mehr oder weniger konstanten Parameter in mathematischen Modellen verwendet, um das

Verhalten eines Gebäudes unter gewünschten Bedingungen vorherzusagen. In der Bauwirtschaft sind Bauingenieure und Architekten in der Lage, Gebäude effizienter zu entwerfen, den Energieverbrauch besser zu planen und den allgemeinen Wohnkomfort zu steigern. Durch das anwenden von verschiedenen Ausgangssituationen ist es möglich, verschiedene Szenarien zu entwerfen und damit Optimierungen vorzunehmen, bevor ein Gebäude gebaut wird, was zu kosteneffizienteren und auch nachhaltigeren Ergebnissen führen kann.

3.1.1 Der U-Wert

Die Effizienz von Gebäuden in Bezug auf Wärmeenergie spielt eine immer größere Rolle in einer modernen Gesellschaft. Um Energiekosten zu senken, den Umweltschutz zu fördern und den Komfort zu erhöhen, wird die Wärmedämmung von Gebäuden zu einem zentralen Anliegen. Ein entscheidender Faktor, der dabei berücksichtigt wird, ist der U-Wert. Der U-Wert ist ein Maß für die Wärmeleitfähigkeit von Bauteilen und ein Schlüsselkonzept in der Wärmedämmung.

Der U-Wert, auch als Wärmedurchgangskoeffizient bezeichnet, ist ein Kennwert, der angibt, wie gut oder schlecht ein Bauteil Wärme leitet. Er wird in Watt pro Quadratmeter und Kelvin gemessen und gibt an, wie viel Wärmeenergie pro Flächeneinheit bei einem Temperaturunterschied von 1 Kelvin zwischen den beiden Seiten eines Bauteils hindurchgeht. Ein niedriger U-Wert ist stellvertretend für eine gute Wärmedämmung, während ein hoher U-Wert auf eine schlechte Isolierung hinweist. Zu berücksichtigen ist dies zur Verminderung von Wärmeverlusten im Winter und um im Sommer die Überhitzung von Räumen zu reduzieren. [23]

Um den U-Wert ermitteln zu können, berechnet man zunächst den Wärmedurchlasswiderstand. Der Wärmedurchlasswiderstand R ist ein Maß für den Widerstand eines Materials gegen die Wärmeenergie und somit der Quotient aus der Dicke eines Materials d und der Wärmeleitfähigkeit λ: [23]

$$R = \frac{d}{\lambda} \qquad \left[\frac{m}{\left(\frac{W}{m * K}\right)} \right]$$

Wärmedurchlasswiderstand

Anschließend addiert man die Wärmeübergangswiderstände (α) der an das Bauteil angrenzenden Gase. Der U-Wert ist nun der Kehrwert der Summe aus der Wärmedurchlasswiderstand und den beiden Wärmeübergangswiderständen: [23]

$$U = \frac{1}{R + \frac{1}{\alpha 1} + \frac{1}{\alpha 2}} \quad \left[\frac{W}{m^2 K} \right]$$

U-Wert

Dies ist eine vereinfachte Form zur Berechnung des U-Wertes, da hier der Baustoff homogen sein muss. Bei Baumaterialien mit z.B. Wärmebrücken, ist die Berechnung des U-Wertes aufwendiger, weshalb dafür spezielle Programme verwendet werden.

Viele Länder haben Baustandards und Vorschriften eingeführt, die minimale Anforderungen an die Wärmedämmung von Gebäuden festlegen. Diese Vorschriften basieren oft auf maximal zulässige U-Werte für verschiedene Bauteile. Durch die Einhaltung dieser Standards kann der Energieverbrauch von Gebäuden erheblich reduziert werden. In Deutschland gibt es keine allgemeine Dämmpflicht, sondern ein Gebäudeenergiegesetz, welches maximale U-Werte für das Dachgeschoss vorschreibt. [25]

Die Technologien im Bereich Wärmedämmung und Bauphysik entwickeln sich dabei ständig weiter, so haben alte Ziegelsteine, welche Anfang des 20. Jahrhunderts oft im Häuserbau verwendet wurden, einen U-Wert von circa 1,1 $\frac{W}{m^2 K}$, [26] während heutige Polyurethan-Hartschaumisolierungen einen U-Wert von ca. 0,24 $\frac{W}{m^2 K}$ aufweisen. [27]

Zukünftige Materialien werden die Wärmedämmung wahrscheinlich noch wirksamer gestalten können und die Energieeffizienz von Gebäuden weiterhin steigern.

Abschließend lässt sich feststellen, dass der U-Wert ein zentrales Konzept in der Wärmedämmung von Gebäuden darstellt. Der Wert ermöglicht eine objektive Bewertung der Wärmedämmfähigkeit von Bauteilen und spielt eine entscheidende Rolle bei der Gestaltung energieeffizienter Gebäude. In einer Zeit, in der Nachhaltigkeit und Energieeinsparung immer wichtiger werden, ist das Verständnis des U-Werts unerlässlich, um die Gebäude der Zukunft energieeffizient zu entwerfen.

3.2 Zeitabhängige Parameter

Als Pendant zu den Zeitunabhängigen Parametern gibt es die zeitabhängigen Parameter. Sie spielen ebenso eine entscheidende Rolle in einer möglichst präzisen Modellierung des thermischen Verhaltens eines Gebäudes. Bei thermischen Gebäudesimulationen berücksichtigen zeitgebundene Parameter die dynamischen Veränderungen in physikalischen Prozessen und Gegebenheiten wie der Temperatur, Energien und Sonnenstrahlen. Diese Parameter sind ebenso wichtig, um ein möglichst genaues thermisches Modell zu erhalten.

Ein zeitabhängiger Parameter ist die Globale Sonnenstrahlung, englisch ‘Global Horizontal Irradiance’, kurz GHI. GHI ist die gesamte menge an Sonnenstrahlen, die auf die Erde auftreffen. Bei den eintreffenden Sonnenstrahlen unterscheidet man dabei zwischen der Direktstrahlung und der Diffusstrahlung. Die Direktstrahlung ist, wie die Bezeichnung bereits vermuten lässt, die Sonnenstrahlung, welche direkt, also ohne Unterbrechungen durch z.B. Wolken, von der Sonne auf die Erde gelangen. Dem gegenüber steht die Diffusstrahlung, welches durch Teilchen wie Staub oder Wassermolekülen in Wolken gebrochen werden. Die Globalstrahlung ist dabei abhängig von der Jahres- und Tageszeit, was wiederum einen direkten Einfluss auf die Lufttemperatur an der Erdoberfläche und der Raumtemperatur eines Gebäudes hat. [28]

Ein weiterer zeitabhängiger Faktor ist zudem die Windgeschwindigkeit und die Windrichtung. Da diese beiden Parameter aber bei meinem Modell nicht mit berücksichtigt werden, bleiben diese außer acht.

Ein weiterer zeitabhängiger Faktor ist das thermische Trägheitsverhalten eines jeden Gebäudes. Dieses Verhalten wird unter anderem durch die vorhandenen

Baumaterialien und der Effizient der Wärmeleitung beeinflusst. Dieses Trägheitsverhalten sorgt dafür, das die Raumtemperatur nicht unmittelbar auf äußere Einflüsse wie die Außentemperatur und die Sonnenstrahlung reagiert, sondern zeitversetzt. So wird bei den Konduktion die kinetische Energie nicht sofort auf ein anderes Material übertragen, sondern Schrittweise. Dies führt zu einer Verzögerung der Anpassung der Innentemperatur eines Gebäudes. [29] Dieses Verhalten kann man sich zu Nutze machen, da es zur Folge hat, das sie Raumtemperatur ein wenig mehr konstant bleibt und man somit die Heiz- und Kühlsysteme nicht kontinuierlich Ein- bzw. Ausschalten muss.

In thermischen Modellierungsprogrammen könnten zudem Nutzerprofile erstellt werden. Das Verhalten der Personen innerhalb eines Gebäudes beeinflusst die gewünschte und die tatsächliche Raumtemperatur. So führen Aktivitäten wie Kochen, Duschen, Beleuchtungen oder die Verwendung von elektronischen Geräten zu einer Wärmeabgabe an die Umgebungsluft, was zu einer Erhöhung der Raumtemperatur führt. Diese internen Wärmequellen variieren je nach Tageszeit und Wochentag. Zudem ist die gewünschte Raumtemperatur abhängig von der Tageszeit, so können Heizkörper in der Zeit, in der die Hausbewohner nicht anwesend sind, aufgrund von Arbeit oder ähnlichem, ausgeschaltet bleiben. Bei kleineren, selbsterstellten Modellierungsprogrammen ist es recht einfach, Nutzerprofile anzulegen und zu Berücksichtigen. Dadurch können Simulationen erzeugt werden, die eine recht präzise Vorhersage zu den benötigten Mengen an Energien erstellen, die Energieeffizienz kann dadurch gesteigert, die Wohnkosten verringert und das allgemeine Wohnkomfort erhöht werden.

Eine Simulationssoftware ermöglicht es, zeitabhängige Parameter in komplexen Gebäudemodellen zu berücksichtigen. Ingenieuren und Architekten ist es somit möglich, präzisere Vorhersagen über das thermische Verhalten eines Gebäudes zu entwerfen, zudem die benötigten Strommengen vorherzusagen und gegebenenfalls die Energieeffizienz von Häusern zu verbessern. Die Berücksichtigung von zeitabhängigen Faktoren erschaffen realistischere Simulationen und liefern umfassendere Informationen, die für eine umweltschonende Gebäudeplanung unerlässlich sind.

Insgesamt zeigen die Parameter der thermischen Gebäudesimulationen, wie komplex und vielschichtig das Verhalten von Gebäuden unter verschiedenen Bedingungen sein

kann. Ihre Berücksichtigung trägt dazu bei, Gebäude energieeffizienter, komfortabler und umweltfreundlicher zu gestalten, indem sie das Verständnis für die Wechselwirkungen zwischen Gebäude, Klima und Nutzungsverhalten vertiefen.

3.3 Verwendete Eingabeparameter

Die oben genannten Parameter sind in einer Excel-Datei aufgelistet. Diese Datei beinhaltet mehrere Arbeitsblätter, sogenannte Sheets. Das erste Sheets, mit der Bezeichnung “House_Location” beinhaltet die geografischen Daten des Hauses. So wird dort der Name der Ortschaft, in dem sich das Haus befindet, die Latitude, Longitude und Altitude aufgelistet. Das darauffolgendem Sheet beinhaltet die konstante physikalischen Eigenschaften des Gebäudes und der Baumaterialien. Dieses Arbeitsblatt, mit der Bezeichnung “House_parameters” beinhaltet die Flächen und Dicken der Wände, Böden, Decken und Fensterscheiben, die Dicke und die Wärmeleitfähigkeit der Objekte. Im folgendem Arbeitsblatt, mit der Bezeichnung “Weather”, sind die Außentemperaturen und der Global Horizontal Irrediance (GHI) gespeichert. Diese dynamischen Parameter sind über einem Zeitraum von einem Jahr hinweg, in einem 1-Stunde-Takt eingetragen. Diese Datensätze dokumentieren den gesamten stündlichen verlauf der Außentemperatur und der Stärke der GHI, an einem, im ersten Sheet definierten Ort. Die Excel-Datei beinhaltet noch weitere Arbeitsblätter und Parameter, welche für die Berechnung der Raumtemperatur und der benötigten Heizleistung aber irrelevant sind.

Datetime	temp_air	ghi
01.01.2021 05:00	7.1	0
01.01.2021 06:00	7.0	0
01.01.2021 07:00	7.2	0
01.01.2021 08:00	7.1	0
01.01.2021 09:00	7.0	9
01.01.2021 10:00	7.3	28
01.01.2021 11:00	7.4	44
01.01.2021 12:00	7.4	50
01.01.2021 13:00	7.6	46

4 EnergyPlus

Bei EnergyPlus handelt es sich um ein kostenlosen Simulationsprogramm, mitfinanziert vom U.S. Departement of Energy im Jahre 1996. [30] Es ermöglicht die detaillierte Analyse der Energieeffizienz von Gebäuden, womit man ebenfalls in der Lage ist, die thermische Leistung, die Temperatur in mehreren Räumen und den Energieverbrauch von elektrischen Heiz- und Kühlkörpern zu berechnen. [31]

4.1 Funktionalitäten

EnergyPlus ist ein umfassendes Programm, mit dem man unter anderem folgende Aspekte untersuchen kann:

- Thermische Leistung:

  Durch die Berechnung des Wärmeflusses durch Gegenstände wie Wände, Dächer und Fenster lässt sich die thermische Effizienz von Gebäuden bewerten.

- Energieverbrauch:

  Mit Hilfe von EnergyPlus lässt sich der Stromverbrauch von verschiedenen elektrischen Energiequellen berechnen. Darunter Heiz- und Kühlkörper, Beleuchtungen und elektrische Haushaltsgeräte.

- HVAC-Systeme:

  Die Software ermöglicht die Modellierung von Heizungs-, Lüftungs- und Klimaanlagensystemen. Aufgrund dessen ermöglicht das Programm dem Anwender, verschiedene Strategien bei der Planung des Hausbaus und bei der Regulierung der Wärmeflüsse zu berücksichtigen.

- Erneuerbare Energien:

  Die Einbindung von erneuerbaren Energien wie Solar- und Windkraft als Stromquellen für Gebäude ermöglicht dem Benutzer die Berechnung des Energieverbrauchs und der Energieerzeugung.

- Komfort:

  Zu Guter Letzt ist das Programm hilfreich, um die persönlichen Vorlieben bezüglich der Raumtemperatur, der Luftfeuchtigkeit, der Luftströmungen und weiteren Parametern zu berechnen und ein Gebäude zu entwerfen, welches die eigenen Vorlieben des Anwenders berücksichtigt.

4.2 Vorteile

EnergyPlus bietet einige Vorteile gegenüber einem selbsterstellten Programm. So sind die Ergebnisse recht genau, da man viele Eingangsparameter besitzt und das Programm seit einiger Zeit auf dem Markt ist und geupdatet wird. Zudem bietet das Programm eine breite Palette an Konfigurationsmöglichkeiten an, so lassen sich z.B. Objekte wie Fenster, mit in die Berechnungen berücksichtigen. Zudem ist man in der Lage, verschiedene Gebäudetypen und Energiequellen mit zu berücksichtigen. Da EnergyPlus zudem ein Open-Source-Produkt und schon seit einigen Jahren auf dem Markt ist, existiert bereits eine recht große Community, in der man Antworten zu Problemstellungen bekommt.

4.3 Nachteile

Mit EnergyPlus lassen sich sehr umfangreiche Berechnungen ausführen. Dies hat zur Folge, dass die eigens dafür benötigten Inputparameter als .idf Dateien eingelesen werden. Diese Dateien beinhalten eine große Menge an Daten, was bei einer Simulation von einfachen Konzepten hinderlich sein kann, da man in der Regel mit dem Dateityp nicht vertraut ist und man sehr viele Werte für die Parameter angeben muss. Anbei ein Abbild einer .idf-Datei. Es fällt auf, dass unerfahrene Nutzer mit der Struktur der Datei und der Fülle an Parametern leicht überfordert sein können und gegebenenfalls Werte zu falschen Parametern angeben, was zu einer ungenauen Berechnung führen kann. Des weiteren ist EnergyPlus eine eigenständige Anwendung, welches externe .idf Dateien einließt. Somit ist eine Einbettung in ein bereits vorhandenes System sehr schwierig Umsetzbar und für die Berechnung der Raumtemperatur ist der Zeitaufwand sehr groß.

```
Table:IndependentVariable,
  Cool Cap Mod func of Temperature_IndependentVariable1,  !- Name
  Linear,                  !- Interpolation Method
  Constant,                !- Extrapolation Method
  13.0,                    !- Minimum Value
  23.89,                   !- Maximum Value
  ,                        !- Normalization Reference Value
  Temperature,             !- Unit Type
  ,                        !- External File Name
  ,                        !- External File Column Number
  ,                        !- External File Starting Row Number
  13.00000,                !- Value 1
  17.00000,                !- Value 2
  19.44440,                !- <none>
  21.00000,                !- <none>
  23.90000;                !- <none>

Table:IndependentVariable,
  Cool Cap Mod func of Temperature_IndependentVariable2,  !- Name
  Linear,                  !- Interpolation Method
  Constant,                !- Extrapolation Method
  -10.0,                   !- Minimum Value
  46.0,                    !- Maximum Value
  ,                        !- Normalization Reference Value
  Temperature,             !- Unit Type
  ,                        !- External File Name
  ,                        !- External File Column Number
  ,                        !- External File Starting Row Number
  -10.00000,               !- Value 1
  15.00000,                !- Value 2
  18.00000,                !- <none>
  24.00000,                !- <none>
  30.00000,                !- <none>
  35.00000,                !- <none>
  38.00000,                !- <none>
  46.00000;                !- <none>

Table:IndependentVariableList,
  Cool Cap Mod func of Temperature_IndependentVariableList,  !- Name
  Cool Cap Mod func of Temperature_IndependentVariable1,  !- Independent Variable 1 Name
  Cool Cap Mod func of Temperature_IndependentVariable2;  !- Independent Variable 2 Name

Table:Lookup,
  Cool Cap Mod func of Temperature,  !- Name
  Cool Cap Mod func of Temperature_IndependentVariableList,  !- Independent Variable List Name
  ,                        !- Normalization Method
  ,                        !- Normalization Divisor
  ,                        !- Minimum Output
  ,                        !- Maximum Output
  Dimensionless,           !- Output Unit Type
  ,                        !- External File Name
  ,                        !- External File Column Number
  ,                        !- External File Starting Row Number
  1.00,                    !- Output Value 1
```

Abbildung 4: .idf-Inputfile

Quelle: Eigene Darstellung

Zudem ist das Programm sehr komplex, was den Umgang für einfache Nutzer schwierig gestaltet. Zudem ist der Zeitaufwand für die Berechnung der genauen Ergebnisse recht hoch, und bei komplexen Gebäuden, mit vielen verschiedenen Komponenten, benötigt man eine große Rechenleistung. EnergyPlus bietet eine GUI, welches dem Nutzer unter anderem die Auswahl der Eingabedatei ermöglicht. Diese GUI ist leider nicht modifizierbar, bei unerfahrenen Nutzern können somit Unachtsamkeiten und damit falsche Berechnungen entstehen. Abkürzungen wie “EDD” und “Slab Out” sind für die meisten Menschen Fremdwörter, die genaue Bedeutung muss also vorab ausfindig gemacht werden.

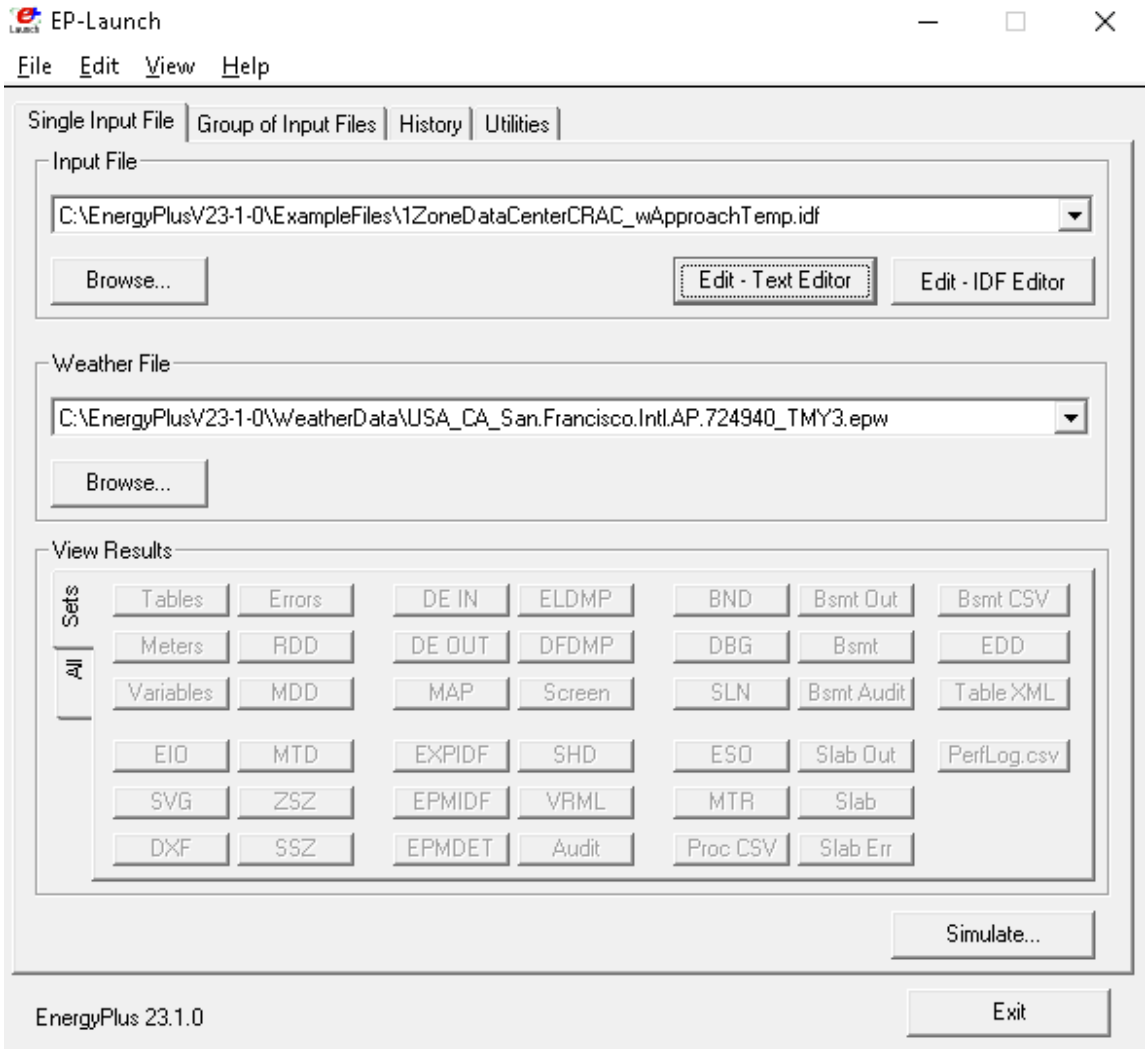

Abbildung 5: EnergyPlus-GUI

Quelle: Eigene Darstellung

Zudem ist das Programm für Entwickler nur mit speziellen Decompilern modifizierbar, sofern man die Rechte dafür hat und genug Zeit findet, den Quelltext zu verstehen, weshalb man bei neueren Methodiken zur Berechnung von z.B. thermischen Flüssen oder den Energieverbrauch, dass Programm nicht selber anpassen kann. Letzten Endes gibt es scheinbar keine Logfile bei einer Eingabe mit fehlerhaften Parametern, weshalb eine Fehlerdiagnose nur sehr schwer durchführbar ist. Selbst bei der Berechnung der mitgegebenen Default-Dateien erscheinen einem 146 Warnungen und

die Output-Files sind sehr schwer zu lesen. Berechnungen mit zweifelhaften Ergebnissen werden in den Outputfiles angezeigt, müssen in den Dateien aber erst gefunden werden.

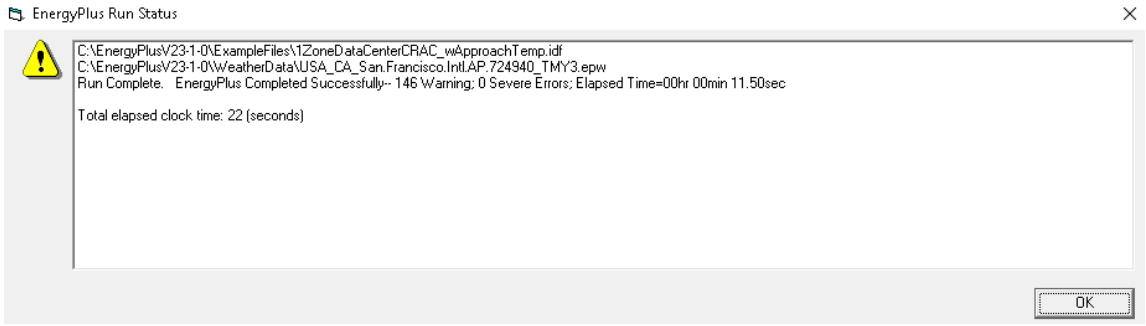

Abbildung 6: EnergyPlus Fehlermeldung

Quelle: Eigene Darstellung

Eine Ausgabe als HTML über den Browser ist etwas angenehmer, leider wird man auch hier mit einer Vielzahl an Daten konfrontiert:

End Uses By Space Type

	Space Type	Electricity [GJ]	Natural Gas [GJ]	Gasoline [GJ]	Diesel [GJ]	Coal [GJ]	Fuel Oil No 1 [GJ]	Fuel Oil No 2 [GJ]	Propane [GJ]	Other Fuel 1 [GJ]	Other Fuel 2 [GJ]	District Cooling [GJ]	District Heating [GJ]	Water [m3]
Heating	Unassigned	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Cooling	Unassigned	37.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Interior Lighting	Unassigned	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Exterior Lighting	Unassigned	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Interior Equipment	GENERAL	407.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Exterior Equipment	Unassigned	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Fans	Unassigned	22.67	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Pumps	Unassigned	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Heat Rejection	Unassigned	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Humidification	Unassigned	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Heat Recovery	Unassigned	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Water Systems	Unassigned	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Refrigeration	Unassigned	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Generators	Unassigned	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Normalized Metrics

Utility Use Per Conditioned Floor Area

	Electricity Intensity [MJ/m2]	Natural Gas Intensity [MJ/m2]	Gasoline Intensity [MJ/m2]	Diesel Intensity [MJ/m2]	Coal Intensity [MJ/m2]	Fuel Oil No 1 Intensity [MJ/m2]	Fuel Oil No 2 Intensity [MJ/m2]	Propane Intensity [MJ/m2]	Other Fuel 1 Intensity [MJ/m2]	Other Fuel 2 Intensity [MJ/m2]	District Cooling Intensity [MJ/m2]	District Heating Intensity [MJ/m2]	Water Intensity [m3/m2]
Lighting	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
HVAC	257.34	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Other	1752.39	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Total	2009.73	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Abbildung 7: EnergyPlus Output als HTML

Quelle: Eigene Darstellung

4.4 Fazit

Für den erfahrenen Nutzer ist EnergyPlus ein sehr umfangreiches Programm, um die Energieeffizienz selbst komplexeren Gebäuden berechnen zu können. Die Auswahl an Methoden zur Kalkulation von etlichen Aspekten, wie z.B. das Heiz- und Kühlsystem, die Verwendung verschiedener Typen an Energieerzeugern und das Einbinden von z.B. Luftströmungen lassen EnergyPlus zu einem sehr wirksamen Tool werden. Leider ist das Programm sehr starr und komplex, weshalb es einiges an Zeit benötigt, sich in die Funktionen und Wirkungsweisen des Tools einzuarbeiten. Für die einfache Berechnung einer Raumtemperatur und einer Heizleistung eines Gebäudes ist das Programm zu schwerfällig.

5 Individuelle Softwarelösung

Es existieren kommerzielle Programme, die in der Lage sind, die Raumtemperaturen anhand verschiedener Parameter berechnen zu können. Diese Programme haben ihre Vor- und Nachteile im Vergleich zu selbst geschriebenen Anwendungen. Anhand des im folgenden Kapitel 5.1 genannten Beispiels wird aber deutlich, dass kommerzielle Software nicht stets die praktischste Lösung darstellen, weshalb es oftmals sinnvoller sein kann, eigene Algorithmen zu kreieren.

5.1 Fallbeispiel

Als Fallbeispiel soll ein Modellhaus aus der Stadt Valparaiso in Chile genommen werden. Die Raumtemperatur dieses Modells wurde über ein Jahr hinweg stündlich gemessen, zudem die Außentemperatur und der benötigte elektronische Verbrauch in Watt. [32] Dieses Modell soll als Maßstab dienen. Ein in Python geschriebenes Programm, welches in dem Framework GEKKO implementiert ist, und unter anderem die Raumtemperatur von Gebäuden berechnen soll, simuliert verschiedene Verhaltensmuster von Häusern, im Hinblick auf den Energieverbrauch. Es berechnet zudem den Alterungsprozess von Lithium-Ionen-Akkumulatoren. Da mehrere Faktoren einen Einfluss auf den Energiebedarf eines Gebäudes haben, unter Anderem die momentane Raumtemperatur und dem damit einhergehenden Bedarf an Heiz- bzw. Kühlsystemen, wird eine Funktion benötigt, die die Raumtemperatur und den benötigten Heizbedarf berechnet, um eine vordefinierte Raumtemperatur halten zu können.

5.2 Gründe für maßgeschneiderte Software

Selbstgeschriebene Programme und kommerzielle Software haben ihre spezifischen Vor- und Nachteile zum Erfüllen von den zu lösenden Aufgaben. Hier sind einige wichtige Aspekte, die bei der Entscheidung zwischen diesen beiden Varianten berücksichtigt werden sollten. Dabei beziehen sich folgende Argumente jeweils auf das im Punkt 5.1 genannte Fallbeispiel. Andere Situationen haben wiederum ihre eigenen Anforderungen, weshalb diese Argumente in anderen Fällen irrelevant sein könnten.

Selbstgeschriebene Programme können genau an die spezifischen Anforderungen und Bedürfnisse eines Projekts geschrieben werden. Ein selbstgeschriebenes Programm ermöglicht die Erfüllung von individuellen Anforderungen. Da zudem die Umsetzung der Lösung maßgeschneidert ist, ist der Umgang mit der Anwendung leicht handhabbar und es ist eine schnelle Berechnung möglich. Zudem hat der Anwender die Kontrolle über das Programm, so dass Modifikationen und Erweiterungen kontinuierlich und nach eigenem Ermessen umgesetzt werden können, ohne auf Updates von Unternehmen warten zu müssen. Dies betrifft auch Sicherheitslücken. Da man zugriff auf den Quellcode eines selbstentworfenden Programms hat, ist man in der Lage, den Code jederzeit zu debuggen und Schwachstellen zu identifizieren und womöglich auch zu beheben. Zudem ist die Anzahl der geschriebenen Codezielen geringer, da sie maßgeschneidert erstellt wurden. Je weniger Zeilen an Code, desto geringer ist die Möglichkeit für das Auftreten von Schwachstellen. Lange Wartezeiten, bis Unternehmen Sicherheitsupdates herausgeben, entfällt bei einem selbstgeschriebenem Programm.

Das Erstellen eines eigenen Programms kann aber sehr viel Zeit und damit auch recht viel Geld kosten. Selbst die Wartung, die man anschließend im Idealfall selber durchführt, kann sehr zeitaufwändig sein. Erschwert wird es, wenn der Entwickler nach einiger Zeit seinen Code nicht mehr nachvollziehen kann. Die Definition der Problemstellungen, das Ausfindig-machen der Lösungswege, das Design, die anschließende Implementierung, sowie das debuggen benötigen zeit und insbesondere Erfahrungen. Bei unerfahrenen Entwicklern kann das Ergebnis ungenau, wenn nicht sogar falsch sein. Eventuelle Werkzeuge wie Entwicklungsumgebungen und dergleichen müssen ebenfalls vorhanden sein und gegebenenfalls bezahlt werden.

Kommerzielle Programme hingegen haben der Vorteil, das sie in der Regel ab Erhalt einsatzbereit sind. Es erfordert somit keine Entwicklungszeit, was in Fällen, in denen man den Einsatz einer Anwendung zeitnah benötigt, essentiell ist. Der Aufwand für Updates entfällt ebenfalls, da renommierte Softwareanbieter kontinuierliche Updates anbieten. Zudem erhält man oftmals einen Support. Sollte man Programme anwenden oder modifizieren, welche man nicht selber geschrieben hat, kann es schwierig werden, die Logiken der Algorithmen nachzuvollziehen. Sollte der Entwickler dabei nicht mehr zu erreichen sein, kann es sehr aufwendig werden, nicht selbst-geschriebene Software upzudaten. [33] Bei kommerziellen Programmen ist es möglich, einen Support zu erhalten und somit zumindest den Umgang des Programms verstehen zu können. Da gewerbliche Software im besten Falle bereits ausreichen getestet werden und bereits zum Einsatz kamen, sind sie oftmals sehr zuverlässig.

Gewerbliche Programme kosten aber in der Regel Geld und erfüllen die Anforderungen nicht stets vollständig. Bei möglichen Modifikationen ist man von Updates des Anbieters abhängig. Ein weiterer Nachteil ist der Sicherheits- und Datenschutzaspekt. Sollte man Anwendungen von Unternehmen erwerben, sollte sichergestellt werden, dass die Umsetzung der Sicherheits- und Datenschutzvorschriften die eigenen Bedürfnisse befriedigen. Da gewerbliche Software zudem als ganze Anwendungen im Umlauf sind, der Fall aus Punkt 5.1 aber lediglich eine Funktion benötigt, welche in einen bereits vorhandene Algorithmus eingebettet werden muss, hätte ein Programm dafür gewerblich in Auftrag gegeben werden müssen, was ebenfalls Zeit und Geld gekostet hätte.

Die Wahl zwischen einem selbstgeschriebenen Programm und einem kommerziellen Programm hängt von verschiedenen Faktoren ab, einschließlich der Komplexität der Anforderungen, verfügbarer Ressourcen, Zeit und des technischen Know-hows. In manchen Fällen kann eine Kombination beider Ansätze, maßgeschneiderte Lösungen für kritische Prozesse und kommerzielle Software für allgemeine Aufgaben, die beste Option sein. In Fallbeispiel 5.1 ist eine kommerzielle Anwendung, aufgrund der Implementierbarkeit und zukünftigen Modifizierbarkeit aber ausgeschlossen.

5.3 Python

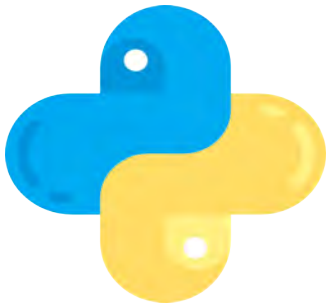

Abbildung 8: Python-Logo

Quelle: https://de.freepik.com/icon/python_5968350#fromView=resource_detail&position=8

Bei Python handelt es sich um eine Hochsprache. Sie zählt zu den bekanntesten, am weitesten verbreitetsten und zu den mächtigsten Programmiersprachen, weltweit. Entwickelt wurde sie in den 1990er Jahren von dem niederländischen Softwareentwickler Guido van Rossum. [34] Python wird, auch dank einer großen Anzahl an Bibliotheken, in zahlreichen Anwendungen eingesetzt, von Frontend-Entwicklungen wie reinen Webanwendungen, über Datenbanken und gilt als die beliebteste Sprache in der künstlichen Intelligenz. Als Bibliotheken bezeichnet man eine Sammlung von wiederverwendbaren Funktionen oder Programmier-Klassen, die zum Einsatz kommen, wenn häufig wiederholende Aufgaben in einem Algorithmus auftreten. Bibliotheken bieten eine Möglichkeit, Code zu organisieren, zu strukturieren und zu teilen, was die Effizienz und Produktivität bei der Entwicklung von Softwareanwendungen erhöhen kann. Ein weiterer entscheidender Vorteil der Hochsprache ist die einfach Lesbarkeit. Die Syntax, die Struktur und Regeln einer Programmiersprache, ist sehr klar und verständlich. Der Code ist oft weniger komplex als in anderen Programmiersprachen, was das Lernen und Verstehen erleichtert. Eine

Besonderheit von Python in der Syntax ist die Verwendung von Einrückungen. Die Einrückungen, durch Tabs oder Leerzeichen erstellt, sind Teil der logischen Gliederung der Algorithmen, was in anderen Sprachen lediglich eine Hilfe darstellt, um den Code lesbarer zu gestalten. Dies erzwingt somit eine konsistente Formatierung.

Aufgrund der großen Anzahl an Bibliotheken und Frameworks, bietet Python bietet eine beeindruckende Bandbreite an Anwendungsgebieten. Frameworks stellen eine strukturierte und vordefinierte Umgebung dar, welche als Grundlage für den Entwurf einer Anwendung dient. [35] Es enthält dabei eine Sammlung an Bibliotheken und Tools die den Entwicklern helfen, Anwendungen effizienter zu erstellen. In der Datenanalyse, insbesondere im Umgang mit Datenbanken, ist die Bibliothek NumPy für numerische Berechnungen und pandas für Datenmanipulation äußerst beliebt. Auch das von mir erstellte Programm verwendet pandas, um das Einlesen der Eingabeparametern zu vereinfachen. Machine Learning und künstliche Intelligenz werden durch Bibliotheken wie TensorFlow entworfen. [36] Des weiteren existieren große Communitys, welche in zahlreichen Bereichen tätig sind. Es wurden zahlreiche Webseiten und Online-Videos erstellt, in denen Dokumente, Tutorials, Foren und Kurse angeboten werden, die es selbst erfahrenen Entwicklern ermöglichen, ihr wissen zu erweitern. Aufgrund der großen Anzahl an Unterstützung ist Python eine der beliebtesten Hochsprachen von Softwareentwicklern. Hinzu kommt, das Python unter Windows, Ubuntu und auch unter Android ausgeführt werden kann. Mit der zunehmenden Bedeutung von Datenanalyse, künstlicher Intelligenz und maschinellem Lernen wird Python weiterhin eine wichtige Rolle in der Welt der Programmierung spielen.

Python ist eine vielseitige Programmiersprache, die für die Erstellung einer hohen Anzahl an Anwendungen geeignet ist. Die sehr präzise und leicht verständliche Syntax, die große Gemeinschaft und die umfangreichen Bibliotheken und Frameworks machen sie zu einer guten Wahl für unerfahrene und auch sehr erprobte Entwickler.

5.4 Visual Studio Code

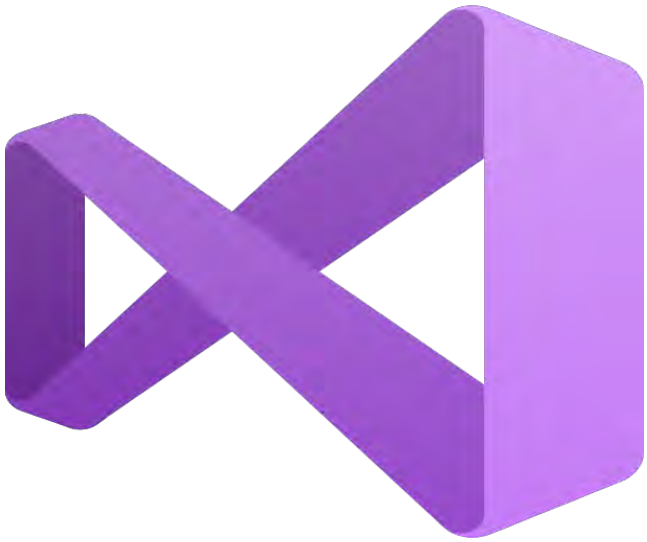

Abbildung 9: VS-Code Logo

Quelle: https://de.freepik.com/icon/visuelle-grundlagen_5968389#fromView=resource_detail&position=3

In der Softwareentwicklung ist eine leistungsstarke, leicht zu bedienende und schnell Operierende Entwicklungsumgebung von entscheidender Bedeutung. Neben Visual Studio und Eclipse ist Visual Studio Code, oft einfach VS-Code genannt, eine der beliebtesten Editoren. Entwickelt wurde die Umgebung von Microsoft. VS Code bietet ein breites Spektrum an Funktionen und Bibliotheken, mit dem sich zahlreiche Compiler, also Werkzeuge, die Quellcode in eine Maschinensprache übersetzen, einbinden lassen. Deshalb ist VS Code für eine Vielzahl an Programmiersprachen geeignet. VS Code ist ein plattformübergreifender Editor, darunter auch maxOS. Da Visual Studio auf Linux-Distributionen lediglich unter Verwendung von Programmen, die ein Windows-Betriebssystemen virtualisieren, sogenannten Emulatoren, oder aber mit Hilfe von Programmen, welche Windows-API-Aufrufe auf Betriebssystemaufrufe des Host-Betriebssystems abbilden, um Windows-Anwendungen auf Linux ausführen zu können, verwendbar ist, ist VS Code eine gute Alternative auf Linux-Systemen. Es bietet zudem verschiedene Syntax-Hervorhebungen, womit einem die Struktur eines Quellcodes durch farbliche Markierungen hervorgehoben wird. Ein weiteres nützliches Tool ist Intellisense, das einem automatische Vervollständigungen des Codes anbietet. Zudem gibt es verschiedene Debugger, womit Entwickler den Ablauf eines Programm

zeilenweise nachverfolgen kann, um mögliche Schwachstellen, Bugs genant, zu finden. VS Code besitzt aufgrund der großen Community eine Vielzahl an Erweiterungen, die den Funktionsumfang der Entwicklungsumgebung stark erweitern können. VS Code gehört wohl zu den am bekanntesten Entwicklungsumgebungen. Aufgrund der stetig wachsenden Gemeinschaft und ständigen Verbesserungen, wird es in den kommenden Jahren weiterhin eine wichtige Rolle in der Softwareentwicklung spielen. Der Editor wird sowohl von Anfängern, als auch von sehr Erfahrenen Entwicklern verwendet. Mit seinen umfangreichen Bibliotheken, Werkzeugen und anderen Erweiterungen, sowie der Möglichkeit für plattformübergreifende Anwendungsentwicklungen macht es zu einer guten Wahl für Entwickler. Die Fähigkeit, individuellen Bedürfnissen gerecht zu werden, macht es zu einem unverzichtbaren Werkzeug in der heutigen Welt der Softwareentwicklung.

5.5 Methodik

Um die Raumtemperatur eines Gebäudes berechnen zu können, sollten alle Wärmequellen in Betracht gezogen werden, die die Raumtemperatur beeinflussen. Da wäre zum Einen die Außentemperatur, welche durch Konvektion durch die Wärme, die Decke und den Fußboden in das Haus gelangt. Hinzu kommt die Wärmeenergie durch die Sonne, welche das Haus erwärmen und die Gebäudehülle Energie in Form von Wärme an die Raumtemperatur abgibt. Die Temperatur der Wände, der Decke und des Fußbodens, welche zusammen nachfolgend als Gebäudehülle bezeichnet wird, ist demnach mitverantwortlich, für die Temperatur, im inneren. Des weiteren spielen innere Wärmequellen, wie Lebewesen und elektronische Geräte, eine Rolle bei der Temperaturerhöhung des Raumes. Um die Raumtemperatur berechnen zu können, müssen diese Faktoren berücksichtigt werden.

5.5.1 Annahmen

Um die Raumtemperatur eines Gebäudes präzise berechnen zu können, benötigt man die Werte zahlreiche Parameter. Je mehr Parameter in Berechnungen einfließen, desto mehr zeit wird benötigt, Algorithmen zu erstellen, die Wahrscheinlichkeit für Fehler und Sicherheitslücken vergrößert sich und die anschließende Modifizierbarkeit wird oftmals

erschwert, da Quellcodes komplizierter werden, je komplexer sie werden. Da für dieses Programm eine vordefinierte Excel-Datei vorhanden ist, in der die Parameter fest stehen, ist es ersichtlich, dass eine 100%-Genauigkeit für die Ergebnisse nicht erreicht werden kann. Um das mögliche Parameter zu ersetzen und um das Programm in der kurzen Zeit entwickeln zu können, wurde deshalb von zahlreichen Annahmen ausgegangen, die Einfluss auf die Präzision der Ergebnisse haben werden. So wird angenommen, das sich in dem Gebäude Personen oder andere Lebewesen aufhalten, elektrische Geräte oder andere Objekte, die Wärme von sich geben, sie wirken wie Heizkörper. Ein entscheidender Wärmefluss erfolgt zudem über die Wände anhand der Außentemperatur der Luft, sowie anhand der Wärmestrahlung, die die Wände abgeben, sobald die Sonne scheint. Die Wärme wird also ausschließlich über Konvektion und Wärmestrahlung abgeben. Die Außenluft und die Raumluft sind zudem homogen, d.h. die Dichte ist an jeder Stelle konstant und die Art der Moleküle ist überall identisch, da es andernfalls zu Fluktuationen kommen kann, was wiederum zu Winden führen kann, die wiederum einen Einfluss auf die Temperatur haben. Des weiteren erleichtert es die Berechnung des Wärmeflusses durch die Wand, wenn davon ausgegangen werden kann, dass die Temperatur der Luft und die Temperatur der Wände, des Daches und des Boden überall konstant ist. Die Dichte der Wände, des Daches und des Fußbodens ist zudem an jeder Stelle einer Einheit ebenfalls homogen. Die Temperatur der Wand berechnet sich aus dem Mittelwert des innen und der Außentemperatur. Für den Falls das die Sonne scheint, addiert sich der zu dem Mittelwert noch die Wärmeenergie der Sonnenstrahlen. Dies erleichtert die Berechnung der abstrahlenden Wärme der Wände. Die Temperatur der Luft und der Objekte des Gebäudes, sind zudem über eine Stunde konstant und erreichen ihren Wert ohne Zeitverzögerung bei jeder vollen Stunde. Die auftretenden Temperaturschwankungen würden die Berechnungen ohne Sensoren andernfalls äußerst herausfordernd gestalten. Für die Konvektion des Wärmestroms durch die Objekte des Gebäudes benötigt man unter anderem die Wärmeübergangskoeffizienten. Diese Koeffizienten stellen einen Wert dar, mit dessen man die Intensität einer Wärmeabgabe an Grenzflächen bestimmen kann, anhand gasförmiger oder flüssiger Medien. [13] Sie sind abhängig von der Dichte, der Wärmeleitfähigkeit und der spezifischen Wärmekapazität, eine physikalische Eigenschaft, die angibt, wie viel Energie benötigt wird, um die Temperatur einer bestimmten Menge eines Stoffes um 1° Kelvin zu erhöhen. Die Wärmeübergangskoeffizienten sind somit Materialabhängig. In dem Projekt wird davon ausgegangen, dass die Wärmeübergangszahlen an den Innenseiten

der Wände und der Fensterscheiben, stets identisch sind mit den Wärmeübergangszahl an den Innenseiten der anderen wände und Fensterscheiben. Das selbe wird auch für die Außenseiten angenommen. Da der Wärmestrom durch ein Objekt auch von der Dicke des Gegenstandes anhängig ist, wurde davon ausgegangen, das die Dicke an jeder Stelle einer Wand, der Decke oder des Bodens, sowie der Fensterscheiben an jeder stelle gleich groß ist und alle Objekte Gerade sind, d.h. keine Krümmungen vorkommen. Der Wärmefluss verläuft zudem stets senkrecht zu den Wänden, den Fensterscheiben, der Decke und dem Fußboden. Der Wärmefluss durch die Objekte trifft in der Realität auf einen Wärmewiderstand, wie in Kapitel 3.1.1 beschrieben. Aufgrund der thermischen Widerstände entstehen auch Wärmeverluste, wobei ausgegangen wird, dass diese vollständig als Wärmeenergie an die Umgebungsluft abgegeben werden.. Winde geben wiederum mehr Energie ab als ruhende Luft. Diese auftretenden kinetischen Energien müssten zusätzlich berechnet werden, wurden der Vereinfachung halber aber weggelassen. Das bedeutet, das die Außenluft und die Luft in dem Raum, stets ruht. Das Haus besteht zudem aus lediglich einem Raum, welcher wiederum über der Erde schwebt, also keinen Bodenkontakt hat und voll und ganz von der Außenluft umgeben ist. Schließlich wird davon ausgegangen, dass die Sonnenstrahlen in einem konstanten Winkel von 45° auf das Gebäude strahlen. Zudem treten keine Verschattungen auf, d.h. die Sonnenstrahlen werden von den ganzen Flächen absorbiert. Es wurde zudem davon ausgegangen, dass der Absorptionsgrad aller Oberflächen identisch sei, weshalb an jedem Objekt der gleiche Anteil an Sonnenenergie reflektiert und absorbiert wird. Diese Annahmen führen dazu, das das Programm recht schnell arbeitet, leichter zu ändern ist, in einem kürzeren Zeitraum erstellt werden kann, aber leider auch keine exakten Ergebnisse liefert.

5.5.2 Berechnung

Aufgrund der in Kapitel 5.5.2 genannten Annahmen konnten die Berechnungen für die Raumtemperatur stark vereinfacht werden. In diesem Abschnitt wird der gesamte Ablauf des Programms dargestellt. Das Programm wird durch einen Funktionsaufruf gestartet.

Abbildung 10: Aufrufsfunktion

Quelle: Eigene Darstellung

Der Funktion wird der Pfad zu der Eingabedatei gegeben. Als Rückgabewerte erhält man die Raumtemperatur und die benötigte Heizenergie als Listen. Nun werden zunächst die Vier Excelsheets mit den Bezeichnungen „House_location“, „House_parameters“ und „Weather“eingelesen. Dabei müssen im Sheet „House_parameters“ folgende Parameter als Überschrift in der ersten Zeile vorhanden sein:

Index Angle WaA WaX WaK WaH WaD WiA WiX WiK WiH WiD

Dabei erscheint in der Spalte ‘’Index’ die Bezeichnungen der Objekte, in dem Fallbeispiel aus Kapitel 5.1 die Bezeichnungen WallN, WallE, WallS, WallW, Roof und Floor. Die Bezeichnung Wall wird mit Wa und Windows mit Wi abgekürzt. Der Buchstabe A bedeutet dabei Area, also die Fläche des Objektes in m² aus der Spalte

Index, X für die Dicke in m, K für die Wärmeleitfähigkeit in $\frac{W}{m \times K}$, H für die spezifische Wärmekapazität in $\frac{J}{Kg \times K}$ und der Buchstabe D für die Dichte des Objektes in $\frac{Kg}{m^3}$.

Die spez. Wärmekapazität beschreibt die Wärmemenge Q, welche benötigt wird, um die Masse eines Stoffes um eine Temperaturdifferenz von einem Kelvin zu erwärmen. [23] In der Spalte WaA werden somit die Flächen der Objekte in m² angegeben. Das Programm achtet darauf, das alle Werte eingetragen wurden, abgesehen von der Spalte Angle, da davon ausgegangen wird, das das Gebäude stets eine Würfelform hat und somit alle im 90° Winkel zueinander stehen. Sollte ein Wert in den anderen Spalten vergessen werden, wird das Sheet als fehlerhaft markiert, ein boolescher Parameter der Klasse „clsHouseParameters“, mit der Bezeichnung boError auf True gesetzt und das Programm wird abgebrochen. In der Konsolenanwendung wird zudem eine passende Meldung ausgegeben. Im Sheet “Weather” werden unter anderem folgende Parameter eingelesen: “DateTime”, “temp_air” und “GHI”. In der Spalte DateTime werden die Zeitstempel im Excel-Format “DD.MM.JJJJ HH.MM” angegeben. Das bedeutet, die Uhrzeit 17:16 am 10.07.2023 würde folgendermaßen eingefügt werden: “10.07.2023 17:16”. Die Zeiten werden dabei im Stundentakt eingeben, demnach für jede Stunde eine Spalte. In der Spalte temp_air wird die Temperatur der jeweiligen Stunde in °Celsius angegeben sowie der GHI-Wert, der die Gesamtmenge an Sonnenstrahlung, die auf eine horizontale Fläche auf der Erdoberfläche eintrifft, in $\frac{W}{m^2}$. Dabei ist zu beachten, dass in den drei genannten Spalten kein Wert fehlen darf, ansonsten wird ein Klassenspezifischer Parameter mit der Bezeichnung “boError” auf True gesetzt und das Programm wird im Anschluss abgebrochen. Sollten die Arbeitsblätter fehlerfrei eingelesen worden sein, werden anschließend die Volumen der Objekte, also der einzelnen Wände, Fensterscheiben, des Daches und des Fußbodens berechnet. Dabei wird folgende Formel verwendet:

$$V[m^3] = Fläche[m^2] * Dicke[m]$$

Volumen 1

Anschließend wird die Masse der einzelnen Komponenten berechnet:

$$Masse\,[kg] = Volumen\,[m^3] * Dichte\,[\frac{Kg}{m}^3]$$

Masse

Daraufhin werden für die einzelnen Elemente die U-Werte aus Punkt 3.1.1 berechnet:

$$U = \frac{1}{R + \frac{1}{\alpha 1} + \frac{1}{\alpha 2}} \;\; [\frac{W}{m^2 K}]$$

Da die Sonnenstrahlen nicht alle Objekte erreichen, wird anschließend die Fläche der Wand im Osten, der Wand im Süden und der Wand im Westen zusammen addiert und anschließend durch 3 Geteilt um einen Mittelwert der Flächen zu erhalten, die im Kontakt zur Sonne stehen. Dieser Mittelwert wird mit der Fläche des Daches addiert, da das Dach ständigem Sonneneinstrahlungen ausgesetzt ist. Anschließend wird die selbe Berechnung mit den passenden U-Werten der Objekte durchgeführt, die von der Sonne bestrahlt werden. Daraufhin wird das Raumvolumen berechnet. Da davon ausgegangen wird, das es sich bei dem Gebäude stets um ein rechteckförmiges Objekt handelt, wäre die Formel

$$Volumen\,[m^3] = Länge\,A\,[m] \times Länge\,B\,[m] \times Länge\,C\,[m]$$

Volumen 2

ausreichend gewesen. Es wurde sich aber für etwaige zukünftige Objekte für folgende Formel entschieden:

$$V\,[m^3] = (\frac{Fläche\,Dach\,[m^2] + Fläche\,Boden\,[m^2]}{2}) \times \frac{\sum Höhe\,Wände\,[m]}{4}$$

Volumen 3

Damit kann man sich an das Volumen von Körpern annähern, deren Seiten unterschiedlich Hoch sind.

Das Volumen des Gebäudes soll dem Luftvolumen innerhalb des Gebäudes gleichgesetzt werden.

Anschließend wird der Wärmedurchfluss durch die Objekte berechnet, die mit der Außenluft in Kontakt stehen, was in dem Fallbeispiel aus Punkt 5.1 alle Flächen betrifft.

$$Q[\frac{W}{m^2}] = U - Wert[\frac{W}{m^2 \times K}] * Flächen[m^2] * (T_1 - T_2[°K])$$

Diese Formel wird nun nach T umgestellt. Dabei ist T1 stets die Höhere Temperatur. Da sich der Wärmestrom ausschließlich in Richtung der kälteren Temperatur bewegt, um ein Gleichgewicht zwischen der Temperaturen in zwei Systemen zu schaffen. Es wird also darauf geachtet, ob die Außentemperatur größer ist, als die vorherige Raumtemperatur, welche vor einer Stunde in dem Gebäude errechnet wurde.

Mit diesen Berechnungen erlangt man die Temperaturen der Raumluft, welche der Außentemperatur hinterherläuft. Die Differenzen sind aber minimal, weshalb es schwierig ist, die Unterschiede deutlich auf einem Diagramm, aufgezeichnet über einen Längeren Zeitraum, darzustellen.

Zeit	Außentemperatur [°C]	Raumtemperatur [°C]
2021-01-01 00:00:00	14,904833	14,904833
2021-01-01 01:00:00	14,427500	14,430512
2021-01-01 02:00:00	14,664667	14,663195
2021-01-01 03:00:00	15,209000	15,205583
2021-01-01 04:00:00	15,048667	15,049703
2021-01-01 05:00:00	14,643833	14,646396
2021-01-01 06:00:00	14,573333	14,573798
2021-01-01 07:00:00	14,802333	14,800902
2021-01-01 08:00:00	15,376500	15,373009
2021-01-01 09:00:00	16,318833	16,313452
2021-01-01 10:00:00	16,846167	16,844764
2021-01-01 11:00:00	17,688500	17,686691
2021-01-01 12:00:00	18,559500	18,558371
2021-01-01 13:00:00	21,202167	21,189000
2021-01-01 14:00:00	20,997667	21,003834

2021-01-01 15:00:00	22,444333	22,440266
2021-01-01 16:00:00	23,296333	23,295943
2021-01-01 17:00:00	22,311667	22,322163
2021-01-01 18:00:00	21,491000	21,499577
2021-01-01 19:00:00	18,362000	18,383988
2021-01-01 20:00:00	16,926833	16,936458
2021-01-01 21:00:00	16,099333	16,104699
2021-01-01 22:00:00	15,898000	15,899310
2021-01-01 23:00:00	16,165167	16,163497

Die gemessenen Raumtemperaturen aus dem Fallbeispiel 5.1 zeigen aber höhere Werte an, da die Temperatur der Raumluft nicht nur abhängig ist von der Außentemperatur, sondern unter anderem auch von Sonnenstrahlen, die auf die Wände und Fenster scheinen, die Wände erhitzen und die Wände die Wärme an die Raumluft abgibt. Fensterscheiben hingegen, lassen einen Teil der Sonnenstrahlen hindurch und erhitzen damit den Innenraum stärker.

Für den Fall, das Sonnenstrahlen vorhanden sind, der ausgelesene Wert in der Spalte „GHI“ demnach nicht 0 ist, sollte die Energie berechnet werden, die die Objekte des Gebäudes, welche in Kontakt mit den Sonnenstrahlen sind, aufgenommen wird. [40]

$$Q[J] = GHI \times (1 - Reflexionsgrad) * \sin(Einfallswinkel)$$

Energie aus Sonnenstrahlen

Die aufgenommene Energie kann mit der Energie, die die Objekte durch den Wärmestrom aufnehmen, addiert werden. Anschließend kann die Temperatur der Wände und der Decke geschätzt werden, indem der Durchschnitt der Außen- und Innentemperatur berechnet und anschließend die Temperaturdifferenz aus folgender Formel verwendet wird:

$$dT[^\circ K] = \frac{Q[J]}{M[Kg] \times C[\frac{J}{Kg \times K}]}$$

Temperaturdifferenz

Dadurch kann ein Schätzwert für die Temperatur der Wände, welche durch die Sonne erhitzt werden, ermittelt werden.

Des weiteren kann die Wärmeenergie berechnet werden, die die Strahlen an die Raumluft abgeben. Dieses Gesetz beschreibt die Menge an Wärmeenergie, die ein schwarzer Körper pro Flächeneinheit und pro Zeit ausstrahlt. Ein schwarzer Körper ist ein ideales Konstrukt, welches alle einfallende Strahlung absorbiert und keine reflektiert. Da die Wände und das Dach aber Sonnenstrahlen teilweise reflektieren, muss ein Emissionsgrad geschätzt werden. Das Stefan-Boltzmann-Gesetz lautet folgendermaßen: [39]

$$P[W] = Fläche[m^2] \times StefanBoltzmannKonstante \times Emissionsgrad \times Oberflächentemperatur^4[K]$$

Stefan-Boltzmann-Gesetz

Da die gezeigten Ergebnisse von den gemessenen Werten aus dem Fallbeispiel aus Kapitel 5.1 aber stets um circa 5 °Celsius abweicht, scheinen weitere Parameter einen relevanten Einfluss auf die Innentemperatur zu haben. Diese Parameter können z.B. Personen und elektrische Gegenstände sein, die Temperatur an ihre Umgebung, demnach der Raumluft abgeben.

Die Ergebnisse der berechneten Werte weichen leider stark von den gegeben Werten des Fallbeispiels aus Kapitel 5.1 ab. Je mehr Parameter verwendet werden, desto präziser sollten die Ergebnisse werden. Leider hat es die Auswirkung, dass je größer die Anzahl der Eingangsparameter ist, desto komplexer und zahlreicher werden die Algorithmen, was eine Verschlechterung der Geschwindigkeit, der Laufzeit der Anwendung haben kann. Außerdem können dadurch mehr Fehlerquellen entstehen.

Um ein vereinfachtes Modell einer Gleichung zu erhalten, die mit den gegeben Parametern eine Raumtemperatur berechnen kann, kann folgende Formeln verwendet werden: [42]

$$T_{Raum}^{(x+1)}[\circ C] = T_{Raum}^{(x)} + \frac{dt}{C_{Raum}} \times \left(\frac{(T_{Wand} - W_{Raum})}{R_{Raum}} \times \Omega_{el} + Sol \times GHI \right)^{(x)}$$

Innentemperatur

$$T_{Wand}^{(x+1)}[\circ C] = T_{Wand}^{(x)} + \frac{dt}{C_{Wand}} \times \left(\frac{(T_{Raum} - W_{Wand})}{R_{Raum}} + \frac{(T_{Außen} - W_{Wand})}{R_{Außen}} + \Omega_{el} + Sol \times GHI^{2} \right)^{(x)}$$

Wandtemperatur

Dabei stellt T_{Raum} die Raumtemperatur und T_{Wand} die Temperatur der Wände, Decke und des Fußbodens dar. C_x die spezifische Wärmekapazität der Objekte, bei C_i ist es die spezifische Wärmekapazität von Luft, welche in Abhängigkeit der Luftfeuchtigkeit und der Zusammensetzung der Gasmoleküle, sowie dem Druck [10] ist. Zudem stellt es die spez. Wärmekapazität der inneren Gegenstände des Raumes dar, das können z.B. Möbel darstellen. Als grober Richtwert kann aber eine spezifische Wärmekapazität von 1,020 $\frac{KJ}{Kg \times K}$ für die Luft verwendet werden, [11] da diese den Großteil eines Raumes ausmacht. Dies ist allerdings ein grober Schätzwert, da die spez. Wärmekapazität aller Gegenstände in dem Raum mit berücksichtigt werden sollten.

R_{Raum} ist der Wärmeübergangskoeffizient zwischen der Raumluft und der Wände, der Decke, sowie des Fußbodens, demnach alle Innenflächen und $R_{Außen}$ der Wärmeübergangskoeffizient zwischen der Außenluft und den Außenflächen. Ω_{el} ist die elektrische Energie in [W], welche in dem Raum durch el. Geräte verbraucht wird und GHI stellt die Global Horizontal Irradiance dar, Sol den Solarenergiedurchlassgrad.

Hier wird die Raumtemperatur anhand der vorherigen Raumtemperatur berechnet, welche als Startpunkt mit der Außentemperatur gleichgesetzt wurde. Addiert wird dann der Quotient aus der Temperaturdifferenz zwischen der Raum- und der Wandtemperatur aus dem vorherigen Zeitpunkt sowie dem Wärmeübergangskoeffizienten der Raumluft und der Innenseite der Wände. Dies wird mit der el. Energie, welche in dem Raum multipliziert wird sowie einem Faktor, da nicht die gesamte el. Leistung in Wärmeenergie übergeht. In diesem Fall wurde der Faktor auf 0,5 gesetzt, da damit die Übereinstimmung der Ergebnisse den gemessenen Werten aus Fallbeispiel 5.1 am größten ist. Multipliziert wird es mit der GHI, welche somit senkrecht auf das Gebäude strahlt, sowie dem Solarenergiedurchlassgrad der

Fensterscheiben. Das Ergebnis wird mit der spez. Wärmekapazität aller Gegenstände in dem Raum sowie der Raumluft dividiert und mit der vorher berechneten Raumluft addiert.

5.5.2.1 Heizleistung

Um die Heizleistung eines Raumes berechnen zu können, die benötigt wird, um die Raumtemperatur auf einen gewünschten Wert erhöhen zu können, wird die Summe aus den Transmissionswärmeverlusten und den Lüftungswärmeverlusten berechnet. [43] Um den Transmissionswärmeverlust berechnen zu können, kann folgende Formel verwendet werden:

$$Ptrans[W] = Raumgrundfläche[m^2] \times U - Wert\ der\ Wände[W/m^2 * K] \times dT[K]$$

Transmissionswärmeverlust

Die Lüftungswärmeverluste lassen sich folgendermaßen berechnen:

$$Pluft[W] = Raumvolumen[m^3] \times spez.\ Wärmekapazität\ Raumluft[J/kg * K] \times dT[K]$$

Lüftungswärmeverlust

Bei einer gewünschten Temperatur von 25 °C werden in dem Fallbeispiel aus 5.1 folgende Wärmeenergien benötigt:

Zeit	Raumtemperatur [°C]	benötigte Heizleistung [W]
2016-01-01 00:00:00	21.88	25468,57
2016-01-01 01:00:00	22.27	25455,7
2016-01-01 02:00:00	22.45	25440,12
2016-01-01 03:00:00	22.53	25372,93
2016-01-01 04:00:00	22.30	25321,02
2016-01-01 05:00:00	22.13	25312,79
2016-01-01 06:00:00	22.05	25303,4

2016-01-01 07:00:00	21.99	25221,91
2016-01-01 08:00:00	22	24854,89
2016-01-01 09:00:00	22.24	23943,34
2016-01-01 10:00:00	22.56	22472,99
2016-01-01 11:00:00	22.79	20695,21
2016-01-01 12:00:00	23.48	19007,99
2016-01-01 13:00:00	24.26	16826,21
2016-01-01 14:00:00	24.79	14713,39
2016-01-01 15:00:00	25.04	12492
2016-01-01 16:00:00	25.54	10493,23
2016-01-01 17:00:00	25.93	9003,4
2016-01-01 18:00:00	26.19	8024,71
2016-01-01 19:00:00	26.12	8026,72
2016-01-01 20:00:00	25.63	8374,44
2016-01-01 21:00:00	25.07	8886,57
2016-01-01 22:00:00	24.63	9451,5
2016-01-01 23:00:00	24.58	10006,35

Die Ergebnisse repräsentieren dabei den Heizbedarf, der benötigt wird, um die Raumtemperatur stündlich von dem berechneten Wert auf den gewünschten Temperatur zu erhitzen. Die Temperatur wird im Alltag allerdings anschließend, abhängig vom Dämmgrad der Wände etc. gehalten, weshalb ein Mittelwert dieser Temperaturen nicht den Heizbedarf eines Raumes entspricht.

5.5.3 Flussdiagramm

Da das beschreiben von Algorithmen und Formeln recht umständlich sein kann, wurde ein Flussdiagramm angefertigt. Flussdiagramme bieten eine visuelle Darstellung eines Programmablaufs. Dies erleichtert Softwareentwicklern das Verständnis der logischen Struktur eines Programms und hilft, komplexe Zusammenhänge auf einen Blick zu erfassen. Sie werden in der Regel angefertigt, bevor die eigentliche Programmierung erfolgt. Dies dient der Planung und Analyse der Problemstellungen. Probleme und zukünftige Fehler können somit eventuell erkannt und gelöst oder umgangen werden. Durch das Erstellen von klaren Pfaden lassen sich unklare Verzweigungen oder

inkonsistente Abläufe rechtzeitig finden und beheben. Es dient aber auch der besseren Kommunikation zwischen den Entwicklern in Projektgruppen.

Ein Flussdiagramm beginnt stets mit einem Oval, als Startpunkt. Entscheidungen, engl. Precisions werden als Quadrate dargestellt, Ausführungen, dies können Berechnungen sein, werden ebenfalls als Quadrate dargestellt, aber mit runden Ecken. Die einzelnen Aktionen, jeweils stellvertretend für ein Symbol, werden mit Pfeilen verknüpft. Somit ist der Pfad, also der Weg von einer Aktion zu der nächsten klar definiert. Precisions haben stets mindestens zwei Abzweigungen. Als einfachstes Beispiel ist eine Entscheidung stellvertretend für eine Ja – Nein Abfrage. Der zu der Antwort gehörige Pfad wird mit dem Wert bezeichnet. Unten, in Abbildung 11, als Beispiel zu sehen sind die Begriffe „Ja“ und „Nein“ .

Insgesamt stellen Flussdiagramme ein wertvolles Werkzeug für die Planung, Kommunikation und Analyse in der Softwareentwicklung dar. Sie sind jedoch nicht ohne ihre Einschränkungen und erfordern sorgfältige Abwägung und Pflege, um ihren Nutzen ganz auszuschöpfen.

Die folgenden Flussdiagramme wurden online unter https://online.visual-paradigm.com erstellt:

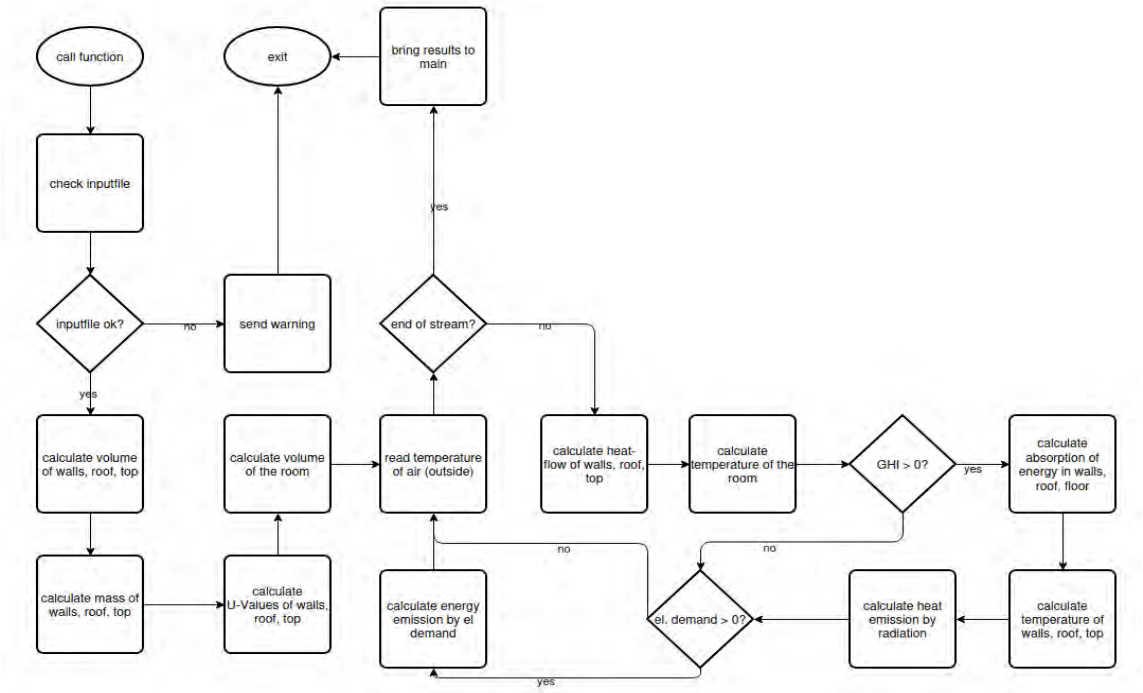

Abbildung 11: Flussdiagramm anhand Konvektion

Quelle: Eigene Darstellung, Zuhilfenahme von online.visual-paradigm.com

Dieses Flussdiagramm zeigt das logische Vorgehen bei der Berechnung der Raumtemperatur anhand der Außentemperatur über die Konvektionen durch die Wände, Fensterscheiben, dem Dach und dem Fußboden. Bei vorhandenem GHI wird zudem die Wärmeaufnahme der Objekte aufgrund der Sonnenstrahlen berechnet.

Da die Ergebnisse den gemessenen Werten stark abwichen, wurde folgende Logik verwendet:

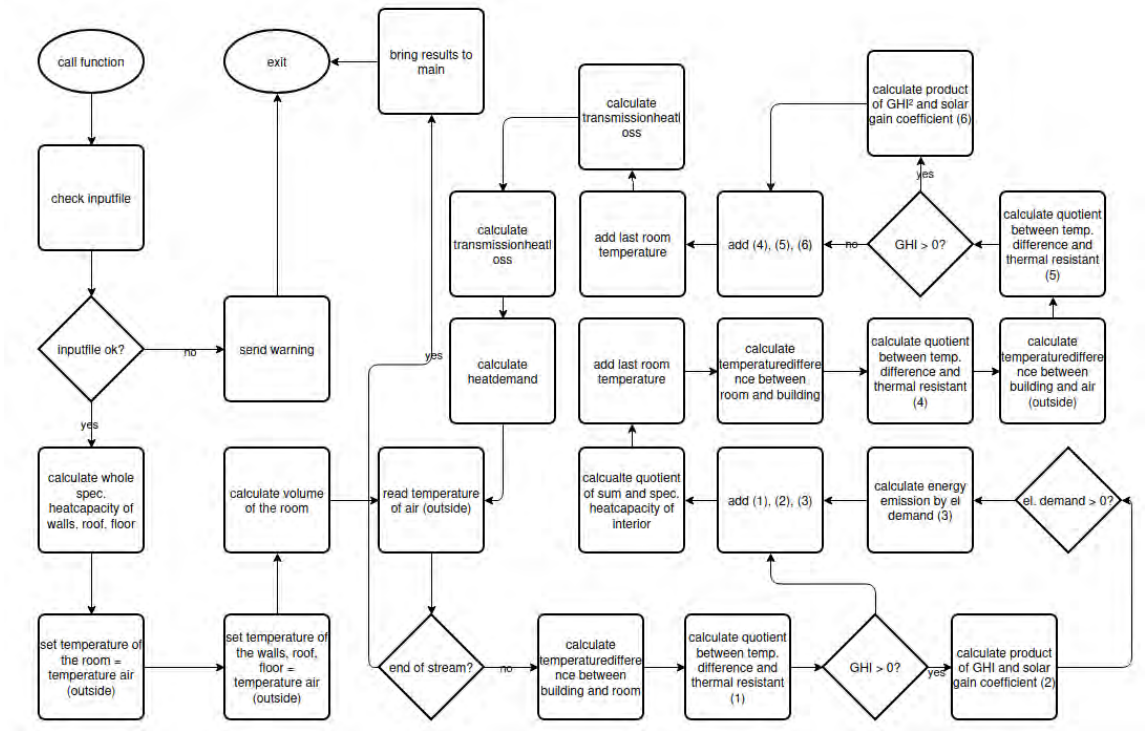

Abbildung 12: Flussdiagramm anhand Allgemeiner-Formel

Quelle: Eigene Darstellung, Zuhilfenahme von online.visual-paradigm.com

5.6 Ergebnis

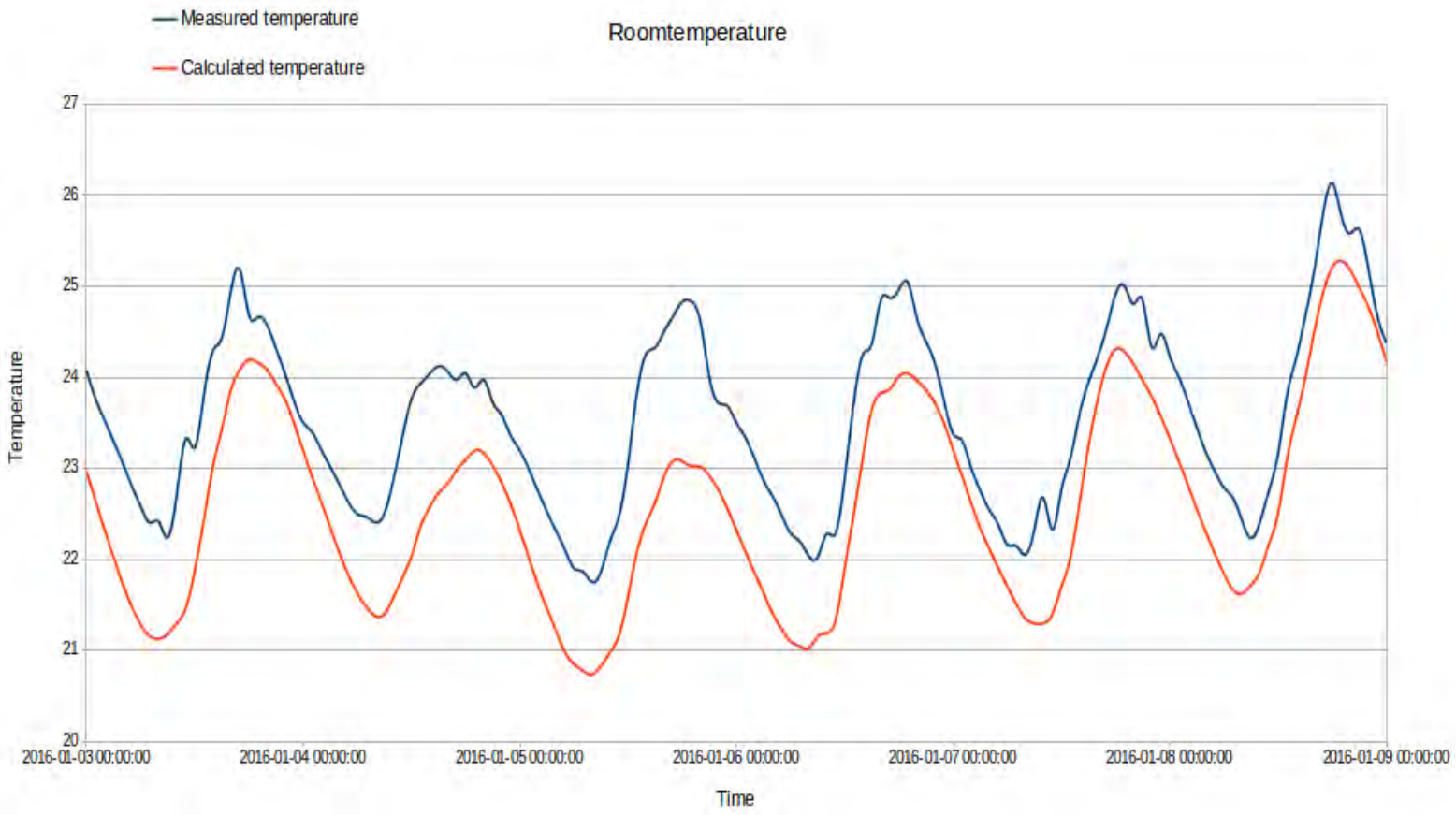

Abbildung 13: Vergleich gemessene und berechnete Raumtemperaturen

Quelle: Eigene Darstellung

Die Kalkulationen des Wärmetransports der Außentemperatur durch die Wände, Fensterscheiben, der Decke und den Fußboden hat gezeigt, dass die berechneten Werte der Außentemperatur sehr ähnlich sind und der Außentemperatur hinterher läuft. Da der Wärmetransport abhängig ist von dem U-Wert, haben die folgenden Faktoren einen Einfluss auf den Wärmetransport: Die Dicke der Materialien, die Fläche und die

Wärmeübergangskoeffizienten. Anhand der Gleichungen lässt sich erkennen, dass je größer die Breite und die Länge der Medien ist, umso geringer ist der U-Wert und desto weniger Wärme pro Zeiteinheit wird transportiert. Ebenso hat die Fläche einen Einfluss auf die Wärmeleitfähigkeit, je größer die Fläche, desto geringer die Wärmeleitfähigkeit. Es wurde zudem deutlich, dass die Außentemperatur einen großen Einfluss auf die Raumtemperatur hat, aber innere Heizkörper, dies können alle Lebewesen und elektrische Geräte sein, sowie die Sonnenstrahlen beeinflussen die Raumtemperatur ebenfalls. Um eine konstante Raumtemperatur halten zu können, sind die verwendeten Baumaterialien somit sehr entscheidend.

Die anschließend berechneten Werte mit Berücksichtigungen des GHI und der internen Heizkörper, mit Hilfe der verallgemeinerten Formel sind den tatsächlich gemessenen Werten sehr ähnlich, Die Differenzen liegen ca. zwischen 0.1 und 2 [K]. Die Ursachen der Unterschiede sind in den Annahmen und den damit einhergehenden Ungenauigkeiten verschuldet. So wurde davon ausgegangen, dass die Sonnenstrahlen mit einem Winkel von 90° auf das Haus eintreffen, als spezifische Wärmekapazität C_i der Raumluft und allen Gegenständen innerhalb des Raumes wurde geschätzt, Die solaren Wärmegewinnungskoeffizienten wie auch die thermischen Widerstände wurden ebenfalls geschätzt. Zudem wurde davon ausgegangen, dass 50% der el. Leistung, welche verbraucht wird, an Heizenergie an die Raumluft abgegeben wird. Dies ist aber abhängig von den jeweiligen el. Gerät. Die berechneten Ergebnisse sind den gemessenen Werten sehr ähnlich, weshalb davon ausgegangen werden kann, das die Berechnung der Innentemperatur anhand weniger Parameter für bestimmte Anwendungsfälle ausreichend ist.

6 Fazit

Mit dieser Bachelorarbeit soll die Forschungsfrage „Wie und mit welchen Parametern lässt sich ein Programm zur thermischen Modellierung eines Hauses erstellen und welche Informationen lassen sich daraus zur Optimierung der Energieeffizienz herauslesen“ beantwortet werden. Am Anfang der Arbeit wurden die drei verschiedenen Wärmetransportarten beschreiben. Dabei wurde ersichtlich, dass Wärme über verschiedene Wege transportiert wird und von mehreren Faktoren abhängig ist. So ist der Aggregatzustand der beteiligten Materialien genauso entscheidend von der Art und des Wärmetransports, wie die physikalischen Eigenschaften der Medien. Im darauffolgendem Kapitel 3 wurden einzelne Parameter genannt, welche Einfluss auf den Wärmestrom haben. Die hohe Anzahl der physikalischen Eigenschaften verdeutlicht die Schwierigkeit, ein System zu entwickeln, welches in der Lage ist, einen Temperaturtransport genau zu berechnen. Eine Aufteilung in zeitabhängige und zeitunabhängige Parameter hat zudem gezeigt, dass die Temperatur innerhalb eines Gebäudes zudem von persönlichen Aktivitäten beeinflusst wird. So kann die Raumtemperatur in einem Badezimmer, während eines Duschvorgangs höher sein, als die Temperatur des Badezimmers, während sich keine Person darin aufhält. Dies hat zur folge, dass Programm, welche mit wenigen Parametern arbeiten, eine genaue Raumtemperatur nicht berechnen können. Anhand der verwendeten Gleichungen zum Berechnung des Wärmestromes anhand der Konvektion, bei dem die berechneten Werte der Innentemperatur der vorgegebenen Außentemperatur sehr ähnlich waren, lässt sich sagen, dass der U-Wert einen entscheidenden Einfluss auf die Raumtemperatur hat. Zudem sind die Fläche der Wände und die Dicke relevant für die Wärmeleitfähigkeit und dem Wärmetransport. Baumaterialien mit einem größeren Abstand von der Außenseite zur Innenseite, dämmern besser, während große Bauflächen mehr Sonnenenergie aufnehmen und mehr Wärme transportieren. Es wurde deutlich, dass es mehrere Möglichkeiten gibt, den Energieverbrauch zu optimieren, indem man unter anderem auf die Baumaterialien, sowie auf die Aktivitäten und der damit einhergehenden Temperaturänderung in Räumen achtet und den Heizbedarf dementsprechend anpasst. Zudem wurde ersichtlich, dass auch mit wenigen Parametern, wie der spezifischen Wärmekapazität der Baumaterialien, der

Außentemperatur, oder dem GHI, eine Raumtemperatur auf wenige Grad Celsius genau berechnet werden kann.

7 Abkürzungsverzeichnis

Bzw.	Beziehungsweise
C	Celsius
ca.	Circa
CPU	Central Processing Unit
chem.	chemisch
d.h.	das heißt
el.	Elektrisch / elektro
GHI	Global Horizontal Irradiance
i.d.R.	in der Regel
K	Kelvin
Kg	Kilogramm
m	Meter
magn.	magnetisch
s	Sekunde
W	Watt
z.B.	zum Beispiel

Abbildungsverzeichnis

Formelverzeichnis

8 Literaturverzeichnis

1. https://www.umweltbundesamt.de/daten/private-haushalte-konsum/strukturdaten-privater-haushalte/bevoelkerungsentwicklung-struktur-privater#immer-mehr-ein-personenhaushalte-in-deutschland, (31.07.2023)
2. https://www.ipm.fraunhofer.de/de/gf/gastechnologie-spektroskopie/komp/simul-ausw/thermische-simulation.html, (31.07.2023)
3. https://www.destatis.de/DE/Presse/Pressemitteilungen/2023/03/PD23_129_12_63.html, (31.07.2023)
4. https://www.destatis.de/DE/Presse/Pressemitteilungen/2023/03/PD23_129_12_63.html, (31.07.2023)
5. https://www.bmwk.de/Redaktion/DE/Infografiken/Industrie/treibhausgasemissionen-deutschland-nach-sektoren.html, (31.07.2023)
6. Paul A. Tipler, Gene Mosca, Physik für Wissenschaftler und Ingenieure, Auflage 6, Seite 782
7. Wärmemanagement bei Leiterplatten, TEC REPORT Ausgabe 01, WÜRTH ELEKTRONIK, Seite 2
8. Bergmann, Schaefer, Lehrbuch der Experimentalphysik, Band 1, Seite 638
9. Bergmann, Schaefer, Lehrbuch der Experimentalphysik, Band 1, Seite 609
10. Paul A. Tipler, Gene Mosca, Physik für Wissenschaftler und Ingenieure, Auflage 6, Seite 671
11. Joachim Fischer, Bernd Fellmuth, Christof Gaiser: Wie viel Energie steckt in der Temperatur? Bestimmung der Boltzmann-Konstante, PTB-Mitteilungen 126 (2016), Heft 2, Seite 94
12. Bernd Fellmuth, Wolfgang Buck, Joachim Fischer, Christof Gaiser, Joachim Seidel: Neudefinition der Basiseinheit Kelvin, PTB-Mitteilungen 117 (2007), Heft 3, Seite 292
13. Paul A. Tipler, Gene Mosca, Physik für Wissenschaftler und Ingenieure, Auflage 6, Seite 781
14. Bergmann, Schaefer, Lehrbuch der Experimentalphysik, Band 1, Seite 680
15. Paul A. Tipler, Gene Mosca, Physik für Wissenschaftler und Ingenieure, Auflage 6, Seite 782
16. Margit Pfundstein, Roland Gellert, Martin H. Spitzner, Alexander Rudolphi, Dämmstoffe Grundlagen Materialien Anwendungen, Detail Praxis,1. Auflage 2007, Seite 8
17. Bergmann, Schaefer, Lehrbuch der Experimentalphysik, Band 1, Seite 676
18. Bergmann, Schaefer, Lehrbuch der Experimentalphysik, Band 1, Seite 675
19. Bergmann, Schaefer, Lehrbuch der Experimentalphysik, Band 1, Seite 675
20. https://www.bfs.de/DE/themen/opt/ir/einfuehrung/einfuehrung_node.html, (14.08.2023)
21. Bergmann, Schaefer, Lehrbuch der Experimentalphysik, Band 1, Seite 692

22. Margit Pfundstein, Roland Gellert, Martin H. Spitzner, Alexander Rudolphi, Dämmstoffe Grundlagen Materialien Anwendungen, Detail Praxis,1. Auflage 2007, Seite 34
23. Margit Pfundstein, Roland Gellert, Martin H. Spitzner, Alexander Rudolphi, Dämmstoffe Grundlagen Materialien Anwendungen, Detail Praxis,1. Auflage 2007, Seite 10
24. Michael Seidel, Band 2, Wärmeübertragung, Seite 24
25. https://www.gesetze-im-internet.de/geg/__47.html, (06.08.2023)
26. https://www.hausjournal.net/u-wert-ziegel, (06.08.2023)
27. https://www.effizienzhaus-online.de/polyurethan-pur/, (06.08.2023)
28. https://www.dwd.de/DE/leistungen/solarenergie/globalstrahlung.html, (07.08.2023)
29. https://www.renovablesverdes.com/de/inercia-termica/, (14.08.2023)
30. https://energyplus.net/licensing, (14.07.2023)
31. https://energyplus.net/, (14.07.2023)
32. Bachelorarbeit „Análisis Comparativo entre la Modelación y Medición de la Calidad Térmica de Viviendas Sociales“, Francisco Antonio Carrasco Serrano, Novemebr 2016
33. Robert C. Martin, Clean Code, Refactoring, Patterns, Testen und Techniken für sauberen Code, 1. Auflage 2009, Seite 29
34. https://news.microsoft.com/de-de/developer-stories-guido-van-rossum/, (10.08.2023)
35. Alessandro Pasetti, Software Frameworks and Embedded Control Systems, Springer, 2002, Seite 8
36. Hope, Tom ; Resheff, Yehezkel S., Einführung in TensorFlow: Deep-Learning-Systeme programmieren, trainieren, skalieren und deployen, O'Reilly Verlag, 2018
37. https://www.chemie.de/lexikon/W%C3%A4rmeleitf%C3%A4higkeit.html, (06.08.2023)
38. Bergmann, Schaefer, Lehrbuch der Experimentalphysik, Band 1, Seite 676
39. https://www.cosmos-indirekt.de/Physik-Schule/W%C3%A4rmestrahlung, (07.08.2023)
40. https://www.cosmos-indirekt.de/Physik-Schule/Solarkonstante#google_vignette, (07.08.2023)
41. Bergmann, Schaefer, Lehrbuch der Experimentalphysik, Band 1, Seite 745
42. https://buildingenergygeeks.org/a-simple-rc-model-python.html, (09.08.2023)
43. https://www.bosch-homecomfort.com/de/de/wohngebaeude/wissen/heizungsratgeber/heizleistung-berechnen/, (09.07.2023)

```
#-----------------------------------------------------------------------------------------------------------
--------------------
#Aufruf
#Andreas Kurz
#Version 1.5
#-----------------------------------------------------------------------------------------------------------
--------------------

import ThermalHouse

flArrHeatPower = []
flArrTemperatures = []
strExcelPath = './PathToExcel.xlsx'

flArrTemperatures,flArrHeatPower = ThermalHouse.fncMain(strExcelPath)
```

```
#-----------------------------------------------------------------------------------------------------------
--------------------
#MainFunction
#Andreas Kurz
#Version 1.5
#-----------------------------------------------------------------------------------------------------------
--------------------

import TH_CheckInputFile
import pandas as pd
import numpy as np


class ClsMass:
    def __init__(self):
        self.flWallN = None
        self.flWallE = None
        self.flWallS = None
        self.flWallW = None
        self.flRoof = None
        self.flFloor = None
        self.flWindowN = None
        self.flWindowE = None
        self.flWindowS = None
        self.flWindowW = None
        self.flWindowRoof = None
        self.flWindowFloor = None
        self.flTotalMass = None
        self.boError = False

    def fncGetWallN(self):
        return self.flWallN
    def fncGetWallE(self):
        return self.flWallE
    def fncGetWallS(self):
        return self.flWallS
    def fncGetWallW(self):
        return self.flWallW
    def fncGetWindowN(self):
        return self.flWindowN
    def fncGetWindowE(self):
        return self.flWindowE
    def fncGetWindowS(self):
        return self.flWindowS
    def fncGetWindowW(self):
        return self.flWindowW
    def fncGetWindowRoof(self):
        return self.flWindowRoof
    def fncGetWindowFloor(self):
        return self.flWindowFloor
    def fncGetRoof(self):
        return self.flRoof
    def fncGetFloor(self):
        return self.flFloor
    def fncGetTotalMass(self):
        return self.flTotalMass
    def fncGetError(self):
        return self.boError

    def fncReadValues(self,Volumes,Parameters):
        try:
            flMass = 0.0
            self.flTotalMass = 0.0
            flDensity = Parameters.fncGetWaD()
            flVolumesN = Volumes.fncGetWallN()
            flVolumesE = Volumes.fncGetWallE()
            flVolumesS = Volumes.fncGetWallS()
            flVolumesW = Volumes.fncGetWallW()
            flVolumesRoof = Volumes.fncGetRoof()
            flVolumesFloor = Volumes.fncGetFloor()

            if len(flDensity) == 6:
```

```
    for intCounter in range(0,len(flDensity)):
        if intCounter == 0:
            flMass = calculateMass(flVolumesN,flDensity[intCounter])
            if flMass < 0:
                self.boError = True
            else:
                self.flWallN = flMass
                self.flTotalMass = self.flTotalMass + flMass
        elif intCounter == 1:
            flMass = calculateMass(flVolumesE,flDensity[intCounter])
            if flMass < 0:
                self.boError = True
            else:
                self.flWallE = flMass
                self.flTotalMass = self.flTotalMass + flMass
        elif intCounter == 2:
            flMass = calculateMass(flVolumesS,flDensity[intCounter])
            if flMass < 0:
                self.boError = True
            else:
                self.flWallS = flMass
                self.flTotalMass = self.flTotalMass + flMass
        elif intCounter == 3:
            flMass = calculateMass(flVolumesW,flDensity[intCounter])
            if flMass < 0:
                self.boError = True
            else:
                self.flWallW = flMass
                self.flTotalMass = self.flTotalMass + flMass
        elif intCounter == 4:
            flMass = calculateMass(flVolumesRoof,flDensity[intCounter])
            if flMass < 0:
                self.boError = True
            else:
                self.flRoof = flMass
                self.flTotalMass = self.flTotalMass + flMass
        elif intCounter == 5:
            flMass = calculateMass(flVolumesFloor,flDensity[intCounter])
            if flMass < 0:
                self.boError = True
            else:
                self.flFloor = flMass
                self.flTotalMass = self.flTotalMass + flMass
        else:
            self.boError = True
else:
    self.boError = True

flMass = 0.0
flDensity = Parameters.fncGetWiD()
flVolumesN = Volumes.fncGetWindowN()
flVolumesE = Volumes.fncGetWindowE()
flVolumesS = Volumes.fncGetWindowS()
flVolumesW = Volumes.fncGetWindowW()
flVolumesRoof = Volumes.fncGetWindowRoof()
flVolumesFloor = Volumes.fncGetWindowFloor()

if len(flDensity) == 6:
    for intCounter in range(0,len(flDensity)):
        if intCounter == 0:
            flMass = calculateMass(flVolumesN,flDensity[intCounter])
            if flMass < 0:
                self.boError = True
            else:
                self.flWindowN = flMass
                self.flTotalMass = self.flTotalMass + flMass
        elif intCounter == 1:
            flMass = calculateMass(flVolumesE,flDensity[intCounter])
            if flMass < 0:
                self.boError = True
            else:
                self.flWindowE = flMass
                self.flTotalMass = self.flTotalMass + flMass
```

```
                    elif intCounter == 2:
                        flMass = calculateMass(flVolumesS,flDensity[intCounter])
                        if flMass < 0:
                            self.boError = True
                        else:
                            self.flWindowS = flMass
                            self.flTotalMass = self.flTotalMass + flMass
                    elif intCounter == 3:
                        flMass = calculateMass(flVolumesW,flDensity[intCounter])
                        if flMass < 0:
                            self.boError = True
                        else:
                            self.flWindowW = flMass
                            self.flTotalMass = self.flTotalMass + flMass
                    elif intCounter == 4:
                        flMass = calculateMass(flVolumesRoof,flDensity[intCounter])
                        if flMass < 0:
                            self.boError = True
                        else:
                            self.flWindowRoof = flMass
                            self.flTotalMass = self.flTotalMass + flMass
                    elif intCounter == 5:
                        flMass = calculateMass(flVolumesFloor,flDensity[intCounter])
                        if flMass < 0:
                            self.boError = True
                        else:
                            self.flWindowFloor = flMass
                            self.flTotalMass = self.flTotalMass + flMass
                    else:
                        self.boError = True
            else:
                self.boError = True

        except:
            print("Error in 'ClsMass', Function 'fncReadValues'")

class ClsThermalResistance:
    def __init__(self):
        self.flWallN = None
        self.flWallE = None
        self.flWallS = None
        self.flWallW = None
        self.flRoof = None
        self.flFloor = None
        self.flTotal = None
        self.boError = False

    def fncGetWallN(self):
        return self.flWallN
    def fncGetWallE(self):
        return self.flWallE
    def fncGetWallS(self):
        return self.flWallS
    def fncGetWallW(self):
        return self.flWallW
    def fncGetRoof(self):
        return self.flRoof
    def fncGetFloor(self):
        return self.flFloor
    def fncGetTotal(self):
        return self.flTotal
    def fncGetError(self):
        return self.boError

    def fncReadValues(self,flArrThickness,flArrThermalConductivity):
        try:
            flResistense = 0.0
            self.flTotal = 0.0

            for intCounter in range(0,len(flArrThickness)):
                if intCounter == 0:
                    flResistense =
calculateHeatTransferResistors(flArrThickness[intCounter],flArrThermalConductivity[intCounter])
```

```
                    if flResistense < 0:
                        self.boError = True
                    else:
                        self.flWallN = flResistense
                elif intCounter == 1:
                    flResistense =
calculateHeatTransferResistors(flArrThickness[intCounter],flArrThermalConductivity[intCounter])
                    if flResistense < 0:
                        self.boError = True
                    else:
                        self.flWallE = flResistense
                elif intCounter == 2:
                    flResistense =
calculateHeatTransferResistors(flArrThickness[intCounter],flArrThermalConductivity[intCounter])
                    if flResistense < 0:
                        self.boError = True
                    else:
                        self.flWallS = flResistense
                elif intCounter == 3:
                    flResistense =
calculateHeatTransferResistors(flArrThickness[intCounter],flArrThermalConductivity[intCounter])
                    if flResistense < 0:
                        self.boError = True
                    else:
                        self.flWallW = flResistense
                elif intCounter == 4:
                    flResistense =
calculateHeatTransferResistors(flArrThickness[intCounter],flArrThermalConductivity[intCounter])
                    if flResistense < 0:
                        self.boError = True
                    else:
                        self.flRoof = flResistense
                elif intCounter == 5:
                    flResistense =
calculateHeatTransferResistors(flArrThickness[intCounter],flArrThermalConductivity[intCounter])
                    if flResistense < 0:
                        self.boError = True
                    else:
                        self.flFloor = flResistense
                else:
                    self.boError = True

                if self.boError == False:
                    self.flTotal = self.flTotal + flResistense

        except:
            print("Error in 'ClsThermalResistance', Function 'fncReadValues'")

class ClsVolume:
    def __init__(self):
        self.flWallN = None
        self.flWallE = None
        self.flWallS = None
        self.flWallW = None
        self.flRoof = None
        self.flFloor = None
        self.flWindowN = None
        self.flWindowE = None
        self.flWindowS = None
        self.flWindowW = None
        self.flWindowRoof = None
        self.flWindowFloor = None
        self.boError = False

    def fncGetWallN(self):
        return self.flWallN
    def fncGetWallE(self):
        return self.flWallE
    def fncGetWallS(self):
        return self.flWallS
    def fncGetWallW(self):
        return self.flWallW
    def fncGetRoof(self):
```

```
        return self.flRoof
    def fncGetFloor(self):
        return self.flFloor
    def fncGetWindowN(self):
        return self.flWindowN
    def fncGetWindowE(self):
        return self.flWindowE
    def fncGetWindowS(self):
        return self.flWindowS
    def fncGetWindowW(self):
        return self.flWindowW
    def fncGetWindowRoof(self):
        return self.flWindowRoof
    def fncGetWindowFloor(self):
        return self.flWindowFloor
    def fncGetError(self):
        return self.boError

    def fncReadValues(self,Parameters):
        try:
            flVolume = 0.0
            flArrArea = Parameters.fncGetWaA()
            flArrThickness = Parameters.fncGetWaX()

            if len(flArrArea) == len(flArrThickness):
                self.flWallN = calculateVolume(flArrArea[0],flArrThickness[0])
                self.flWallE = calculateVolume(flArrArea[1],flArrThickness[1])
                self.flWallS = calculateVolume(flArrArea[2],flArrThickness[2])
                self.flWallW = calculateVolume(flArrArea[3],flArrThickness[3])
                self.flRoof  = calculateVolume(flArrArea[4],flArrThickness[4])
                self.flFloor = calculateVolume(flArrArea[5],flArrThickness[5])
            else:
                self.boError = True

            flVolume = 0.0
            flArrArea = Parameters.fncGetWiA()
            flArrThickness = Parameters.fncGetWiX()

            if len(flArrArea) == len(flArrThickness):
                self.flWindowN = calculateVolume(flArrArea[0],flArrThickness[0])
                self.flWindowE = calculateVolume(flArrArea[1],flArrThickness[1])
                self.flWindowS = calculateVolume(flArrArea[2],flArrThickness[2])
                self.flWindowW = calculateVolume(flArrArea[3],flArrThickness[3])
                self.flWindowRoof = calculateVolume(flArrArea[4],flArrThickness[4])
                self.flWindowFloor =calculateVolume(flArrArea[5],flArrThickness[5])
            else:
                self.boError = True
        except:
            print("Error in 'ClsVolume', Function 'fncReadValues'")


'''
-------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------
Functions
'''
#def fncIsNumeric(stValue):
#    try:
#        float(stValue)
#        return True
#    except ValueError:
#        return False


#Calculate Volume
def calculateVolume(flArea,flThickness):
    try:
        flVolume = flArea * flThickness
        return flVolume
    except:
        print("Error in 'calculateVolume'")

#calculate Mass
```

```
def calculateMass(flVolume,flDensity):
    try:
        flMass = flVolume * flDensity
        return flMass
    except:
        print("Error in 'calculateMass'")

#calculate HeatAmount Q
def calculateHeatAmount(flUValue,flThermalDifference,flArea,intTime):
    try:
        flHeatAmount = flUValue * flThermalDifference * flArea * intTime
        return flHeatAmount
    except:
        print("Error in 'calculateHeatAmount'")

#calculate Wärmedurchlasswiderstände
def calculateHeatTransferResistors(flThickness,flThermalConductivity):
    try:
        if flThermalConductivity > 0:
            flHeatTransferResistor = ( flThickness * flThermalConductivity ) + 0.13 + 0.04
 #Luftwiderstand Innen + Luftwiderstand Aussen
        else:
            flHeatTransferResistor = -1

        return flHeatTransferResistor
    except:
        print("Error in 'calculateHeatTransferResistors'")

#calculate Walltemperature
def calculateAverage(flFirst,flSecond):
    try:
        flWallTemperature = (flFirst + flSecond) / 2

        return flWallTemperature
    except:
        print("Error in 'calculateAverage'")

#calculate Konvektion
#Q = U * A * (T2 - T1)
def calculateKonvektion(flUValue, flArea,flTemperature1,flTemperature2):
    try:
        if flTemperature1 > flTemperature2:
            flKonvektion = flUValue * flArea * (flTemperature1 - flTemperature2)
        else:
            flKonvektion = flUValue * flArea * (flTemperature2 - flTemperature1)

        return flKonvektion
    except:
        print("Error in 'calculateKonvektion'")

#calculate U-Value
def calculateUValue(flThickness, flThermalConductivity,
flHeatTransferCoefficientIn,flHeatTransferCoefficientOut):
    try:
        flUValue = 0.0
        if flHeatTransferCoefficientOut != 0  and flHeatTransferCoefficientIn != 0 and
flThermalConductivity != 0 and ((flThickness/flThermalConductivity) + (1/flHeatTransferCoefficientIn) +
(1/flHeatTransferCoefficientOut)) != 0:
            flUValue = 1 / ((flThickness/flThermalConductivity) + (1/flHeatTransferCoefficientIn) +
(1/flHeatTransferCoefficientOut))
        else:
            flUValue = 0.0

        return flUValue
    except:
        print("Error in 'calculateUValue'")

#calculate RoomVolume(flAreas)
def calculateRoomVolume(flAreas):
    try:
        flRoomVolume = 0.0
        flAverageAreaRoofAndTop = 0.0
        flHighWallN = 0.0
```

```
        flHighWallE = 0.0
        flHighWallS = 0.0
        flHighWallW = 0.0
        flAreaRoof = 0.0
        flAreaFloor = 0.0

        #library math is not supported, i use instead: x**0.5
        flHighWallN = flAreas[0]**0.5
        flHighWallE = flAreas[1]**0.5
        flHighWallS = flAreas[2]**0.5
        flHighWallW = flAreas[3]**0.5

        flAreaRoof = flAreas[4]
        flAreaFloor = flAreas[5]

        flAverageAreaRoofAndTop = (flAreaRoof + flAreaFloor) / 2
        flRoomVolume =  flAverageAreaRoofAndTop  * ((flHighWallN + flHighWallE + flHighWallS +
flHighWallW) / 4)

        return flRoomVolume
    except:
        print("Error in 'calculateRoomVolume'")

#calculate Heatamount throug the sun ( Q_sonne = GHI * A * (1 - e))
# or Qrad = ε * σ * A * (Tout^4 - Tin^4)
def calculateHeatAmountThroughSun(flGHI, flArea, flEmission):
    try:
        flHeatAmount = 0.0

        flHeatAmount = flGHI * flArea * (1 - flEmission)

        return flHeatAmount
    except:
        print("Error in 'calculateHeatAmountThroughSun'")

#calculate Roomtemperature Tin [°C]
# T =
def calculateRoomTemperature(flHeatAmount, flHeatCapacity, flMass,flTemperatureWall):
    try:
        if (flHeatCapacity * flMass) != 0:
            flRoomtemperature = ( flHeatAmount / (flHeatCapacity * flMass) ) + flTemperatureWall
        else:
            flRoomtemperature = -500

        flRoomtemperature = flRoomtemperature
        return flRoomtemperature
    except:
        print("Error in 'calculateRoomTemperature'")

#calculate Avereage of Areas with Emissions
def calculateAreasWithEmission(flAreas, flEmissionWall, flEmissionWindow):
    try:
        flAreaWithEmission = 0.0
        #Wall E - Window E
        flAreaWithEmission = (flAreas[1] - flAreas[7]) * (1 - flEmissionWall)
        #Window E
        flAreaWithEmission = flAreaWithEmission + (flAreas[7] * (1 - flEmissionWindow))
        #Wall S - Window S
        flAreaWithEmission = flAreaWithEmission + ((flAreas[2] - flAreas[8]) * (1 - flEmissionWall))
        #Window S
        flAreaWithEmission = flAreaWithEmission + (flAreas[8] * (1 - flEmissionWindow))
        #Wall W - Window W
        flAreaWithEmission = flAreaWithEmission + ((flAreas[3] - flAreas[9]) * (1 - flEmissionWall))
        #Window W
        flAreaWithEmission = flAreaWithEmission + (flAreas[9] * (1 - flEmissionWindow))

        flAreaWithEmission = flAreaWithEmission / 3

        #Roof - Window Roof
        flAreaWithEmission = flAreaWithEmission + ((flAreas[4] - flAreas[10]) * (1 - flEmissionWall))
        #Window Roof
        flAreaWithEmission = flAreaWithEmission + (flAreas[10] * (1 - flEmissionWindow))
```

```
        return flAreaWithEmission
    except:
        print("Error in 'calculateAreasWithEmission'")

#Calculate Areas with Air contact
def calculateAreasWithAirContact(flAreas):
    try:
        flAreaWithAirContact = []
        #Wall N - Window N
        flAreaWithAirContact.append(flAreas[0] - flAreas[6])
        #Wall E - Window E
        flAreaWithAirContact.append(flAreas[1] - flAreas[7])
        #Wall S - Window S
        flAreaWithAirContact.append(flAreas[2] - flAreas[8])
        #Wall W - Window W
        flAreaWithAirContact.append(flAreas[3] - flAreas[9])
        #Roof - Window Roof
        flAreaWithAirContact.append(flAreas[4] - flAreas[10])
        #Floor - Window Floor
        flAreaWithAirContact.append(flAreas[5] - flAreas[11])
         #Window N
        flAreaWithAirContact.append(flAreas[6])
        #Window E
        flAreaWithAirContact.append(flAreas[7])
        #Window S
        flAreaWithAirContact.append(flAreas[8])
        #Window W
        flAreaWithAirContact.append(flAreas[9])
        #Window Roof
        flAreaWithAirContact.append(flAreas[10])
        #Window Floor
        flAreaWithAirContact.append(flAreas[11])

        return flAreaWithAirContact
    except:
        print("Error in 'calculateAreasWithAirContact'")

#calculate Avereage of HeatCapacitiy from Areas with Sun
def calculateAvereageHeatCapacitiy(flArrHeatCapacity):
    try:
        flHeatCapacitiy = 0.0
        #Wall E - Window E
        flHeatCapacitiy = (flArrHeatCapacity[1] - flArrHeatCapacity[7])
        #Window E
        flHeatCapacitiy = flHeatCapacitiy + flArrHeatCapacity[7]
        #Wall S - Window S
        flHeatCapacitiy = flHeatCapacitiy + (flArrHeatCapacity[2] - flArrHeatCapacity[8])
        #Window S
        flHeatCapacitiy = flHeatCapacitiy + flArrHeatCapacity[8]
        #Wall W - Window W
        flHeatCapacitiy = flHeatCapacitiy + (flArrHeatCapacity[3] - flArrHeatCapacity[9])
        #Window WS
        flHeatCapacitiy = flHeatCapacitiy + flArrHeatCapacity[9]

        #flHeatCapacitiy = flHeatCapacitiy / 3

        #Roof - Window Roof
        flHeatCapacitiy = flHeatCapacitiy + (flArrHeatCapacity[4] - flArrHeatCapacity[10])
        #Window Roof
        flHeatCapacitiy = flHeatCapacitiy + flArrHeatCapacity[10]

        return flHeatCapacitiy
    except:
        print("Error in 'calculateAvereageHeatCapacitiy'")
'''
-------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------
main programm
declare variables
'''
def fncMain(strExcelPath):

    boEnd = False
```

```
    boError = False
    flArrRoomTemperatures = []
    flArrThickness = []
    flArrHeatConductivity = []
    flAreas = []
    flArrAreasWithAirContact = []
    flArrHeatTransferCoefficientIn = []
    flArrHeatTransferCoefficientOut = []
    flArrMass = []
    flArrUValue = []
    flArrGHI = []
    flHeatingAmount = 0.0
    flTransmissionsHeatLoss = 0.0
    flVentilationHeatLoss = 0.0
    flRoomArea = 0.0
    flInnerGainFactor = 0.5
    flTemperatureDifference = 0.0
    flRoomVolume = []
    flRequiredHeatPower = []
    flTemperatureDifference = 0.0
    flRequiredHeatPower = []
    flResistor = 0.01
    flHeatTransferCoefficientUpToDown = 8.1
    flHeatTransferCoefficientDownToUp = 5.8
    flHeatTransferCoefficientOutWall = 23
    flHeatTransferCoefficientInWall = 8.1
    flHeatTransferCoefficientOutWindow = 12
    flHeatTransferCoefficientInWindows = 8.1
    flSpecificHeatCapacityAir = 1020
    flWall = []
    flSolarGainCoefficient = 0.6
    flResistor = 0.01
    flArrHeatCapacity = []
    flArrElectricalDemand = []
    flArrTemperatureOut = []
    flArrRoomTemperatures = []
    flUValue = 0.0
    flUValueFromObjectsWithSun = 0.0
    flAreaWhole = 0.0
    flWholeHeatCapacity = 0.0
    flWholeMass = 0.0
    flDesiredTemperatureMin = 25
    flDesiredTemperatureMax = 0.0
    flAreaFromObjectsWithSunContact = 0.0
#-----------------------------------------------------------------------------------------------------
-------------------------------------------------
# Start Program

    while boEnd == False:

        #Get ExcelFile  clsLocation,
        clsParameter, clsWeather, clsDemand , clsHVAC =
TH_CheckInputFile.fncMainCheckInputFile(strExcelPath)

        if clsParameter.fncGetError() == False and clsWeather.fncGetError() == False and
clsDemand.fncGetError() == False:

            #Calculate Volume
            clsVolumes = ClsVolume()
            clsVolumes.fncReadValues(clsParameter)
            if clsVolumes.fncGetError() == True:
                boError = True
                break

            #Calculate Mass
            clsMass = ClsMass()
            clsMass.fncReadValues(clsVolumes,clsParameter)
            if clsMass.fncGetError() == True:
                boError = True
                break

            flArrHeatCapacity = clsParameter.fncGetWaH()
            flArrTemperatureOut = clsWeather.fncGetTempAir()
```

```
            flArrElectricalDemand = clsDemand.fncGetEd()

            for intX in range(0,12):
                if intX < 4:
                    flArrHeatTransferCoefficientIn.append(flHeatTransferCoefficientInWall)
                    flArrHeatTransferCoefficientOut.append(flHeatTransferCoefficientOutWall)
                elif intX == 4 or intX == 10:
                    flArrHeatTransferCoefficientIn.append(flHeatTransferCoefficientUpToDown)
                    flArrHeatTransferCoefficientOut.append(flHeatTransferCoefficientUpToDown)
                elif intX == 5 or intX == 11:
                    flArrHeatTransferCoefficientIn.append(flHeatTransferCoefficientDownToUp)
                    flArrHeatTransferCoefficientOut.append(flHeatTransferCoefficientDownToUp)
                else:
                    flArrHeatTransferCoefficientIn.append(flHeatTransferCoefficientInWindows)
                    flArrHeatTransferCoefficientOut.append(flHeatTransferCoefficientOutWindow)

            flArrGHI = clsWeather.fncGetGHI()
            flAreas = clsParameter.fncGetWaA() + clsParameter.fncGetWiA()
            flArrHeatCapacity = clsParameter.fncGetWaH() + clsParameter.fncGetWiH()
            flArrThickness = clsParameter.fncGetWaX() + clsParameter.fncGetWiX()
            flArrHeatConductivity = clsParameter.fncGetWaK() + clsParameter.fncGetWiK()
            flArrMass.append(clsMass.fncGetWallN())
            flArrMass.append(clsMass.fncGetWallE())
            flArrMass.append(clsMass.fncGetWallS())
            flArrMass.append(clsMass.fncGetWallW())
            flArrMass.append(clsMass.fncGetRoof())
            flArrMass.append(clsMass.fncGetFloor())
            flArrMass.append(clsMass.fncGetWindowN())
            flArrMass.append(clsMass.fncGetWindowE())
            flArrMass.append(clsMass.fncGetWindowS())
            flArrMass.append(clsMass.fncGetWindowW())
            flArrMass.append(clsMass.fncGetWindowRoof())
            flArrMass.append(clsMass.fncGetWindowFloor())

            #calculate U-Value from every Object
            for intX in range(0,len(flArrThickness)):
                if flArrThickness[intX] != 0 and flArrHeatConductivity[intX] != 0:

 flArrUValue.append(calculateUValue(flArrThickness[intX],flArrHeatConductivity[intX],flArrHeatTransferCoe
fficientIn[intX],flArrHeatTransferCoefficientOut[intX]))

            #get U-Values from Objects with Suncontact
            flUValueFromObjectsWithSun = flArrUValue[1]
            flUValueFromObjectsWithSun = flUValueFromObjectsWithSun + flArrUValue[2]
            flUValueFromObjectsWithSun = flUValueFromObjectsWithSun + flArrUValue[3]
            flUValueFromObjectsWithSun =  flUValueFromObjectsWithSun / 3
            flUValueFromObjectsWithSun = flUValueFromObjectsWithSun + flArrUValue[4]

            #get Areas fom Objects with Suncontact
            flAreaFromObjectsWithSunContact = flAreas[1]
            flAreaFromObjectsWithSunContact = flAreaFromObjectsWithSunContact + flAreas[2]
            flAreaFromObjectsWithSunContact = flAreaFromObjectsWithSunContact + flAreas[3]
            flAreaFromObjectsWithSunContact = flAreaFromObjectsWithSunContact / 3
            flAreaFromObjectsWithSunContact = flAreaFromObjectsWithSunContact + flAreas[4]

            flArrAreasWithAirContact = calculateAreasWithAirContact(flAreas)

            #calculate capazity from walls, roof, floor
            for intX in range(0,len(flArrHeatCapacity)):
                if intX < 6:
                    flWholeHeatCapacity = flWholeHeatCapacity + flArrHeatCapacity[intX]

            flWholeHeatCapacity = flWholeHeatCapacity / 6

            for intX in range(0,len(flArrUValue)):
                flUValue = flUValue + flArrUValue[intX]

            for intX in range(0,len(flArrAreasWithAirContact)):
                flAreaWhole = flAreaWhole + flArrAreasWithAirContact[intX]

            for intX in range(0,len(flArrMass)):
                flWholeMass = flWholeMass + flArrMass[intX]
```

```
            if len(flAreas) > 4:
                if flAreas[4] == flAreas[5]:
                    flRoomArea = flAreas[4]
                else:
                    boError = True
                    break
            else:
                boError = True
                break

            flRoomVolume = calculateRoomVolume(flAreas)

#-------------------------------------------------------------------------------------------------------------
-------------------------
# Function that will calculate the indoor temperature for a given value of R and C
            flDesiredTemperatureMin = clsHVAC.fncGetMin()
            flDesiredTemperatureMax = clsHVAC.fncGetMax()
            flWall.append(flArrTemperatureOut[0])
            flArrRoomTemperatures.append(flArrTemperatureOut[0])
            flWall.append(flArrTemperatureOut[0])
            flArrRoomTemperatures.append(flArrTemperatureOut[0])

            for intCounter in range(1,len(flArrTemperatureOut)):
                #Wall
                Twall2 = (flArrRoomTemperatures[intCounter] - flWall[intCounter]) / flResistor
                Twall2 = Twall2 + ((flArrTemperatureOut[intCounter] - flWall[intCounter]) / flResistor)
                Twall2 = Twall2 + (flSolarGainCoefficient * flArrGHI[intCounter])
                Twall2 = (Twall2 / flWholeHeatCapacity) + flWall[intCounter-1]
                Twall2 =
calculateAverage(flArrRoomTemperatures[intCounter],flArrTemperatureOut[intCounter])
                flWall.append(Twall2)
#-------------------------------------------------------------------------------------------------------------
-------------------------------------------------
# Calculate Heating Energy
                if  flArrRoomTemperatures[intCounter] > flDesiredTemperatureMin:
                    flRequiredHeatPower.append(0)
                else:
                    if flArrRoomTemperatures[intCounter]> flDesiredTemperatureMax:
                        #Cooling, no Power needed
                        flRequiredHeatPower.append(0)
                    else:
                        flTemperatureDifference = flDesiredTemperatureMin -
flArrRoomTemperatures[intCounter]
                        flTransmissionsHeatLoss = flRoomArea * flUValue * flTemperatureDifference
                        flVentilationHeatLoss = flRoomVolume * (flSpecificHeatCapacityAir/1000) *
flTemperatureDifference
                        flHeatingAmount = flTransmissionsHeatLoss + flVentilationHeatLoss
                        flRequiredHeatPower.append(round(flHeatingAmount,2))

                TIn2 = (Twall2 - flArrRoomTemperatures[intCounter-1]) / flResistor
                #el Demand as Heating Power
                TIn2 = TIn2 + flArrElectricalDemand[intCounter] * flInnerGainFactor
                TIn2 = TIn2 + (flSolarGainCoefficient * flArrGHI[intCounter])
                TIn2 = (TIn2 / (flSpecificHeatCapacityAir) ) +  flArrRoomTemperatures[intCounter]
                flArrRoomTemperatures.append(TIn2)

        boEnd = True
#-------------------------------------------------------------------------------------------------------------
-------------------------------------------------
    if boError == True:
        flArrRoomTemperatures = []

    else:

        del flArrRoomTemperatures[0]
        del flArrRoomTemperatures[1]

    return flArrRoomTemperatures,flRequiredHeatPower
```

```
#-----------------------------------------------------------------------------------------------------------
--------------------
#CheckInputFile
#Andreas Kurz
#Version 1.5
#-----------------------------------------------------------------------------------------------------------
--------------------
from datetime import datetime
import pandas

class ClsHVAC:
    def __init__(self):
        self.intDesiredTemperatureMin = None
        self.intDesiredTemperatureMax = None
        self.boError = False

    def fncGetMin(self):
        return self.intDesiredTemperatureMin

    def fncGetMax(self):
        return self.intDesiredTemperatureMax

    def fncGetError(self):
        return self.boError

    def InsertValue(self,Value):
        try:
            SaveValue = 0
            if type(Value) == int or type(Value) == float:
                    SaveValue = Value
            elif type(Value) == str:
                if Value.isdigit() == True:
                    flValue = float(Value)
                    SaveValue = flValue
                elif fncIsNumeric(Value) == True:
                    flValue = float(Value)
                    SaveValue = flValue
                else:
                    self.boError = True
            else:
                self.boError = True
            return SaveValue
        except:
            print("Error in 'ClsHVAC', Function 'InsertValue'")

    def fncReadValues(self, Sheet):
        varValues = []
        NumberColumns = len(Sheet.columns)
        for index, row in Sheet.iterrows():
            varValues.clear()
            for intX in range(0,NumberColumns,1):
                varValues.append(row[intX])

            if varValues[0] == "Lower_lim":
                self.intDesiredTemperatureMin = self.InsertValue(varValues[1])
            elif varValues[0] == "Upper_lim":
                self.intDesiredTemperatureMax = self.InsertValue(varValues[1])


class ClsHouseParameters:
    def __init__(self):
        self.intArrAngle = []
        self.intArrWaA = []
        self.intArrWaX = []
        self.intArrWaK = []
        self.intArrWaH = []
        self.intArrWaD = []
        self.intArrWiA = []
        self.intArrWiX = []
        self.intArrWiK = []
        self.intArrWiH = []
        self.intArrWiD = []
```

```
        self.flTotalArea = None
        self.flTotalHeatCapacity = None
        self.boError = False

    def fncGetAngle(self):
        return self.intArrAngle
    def fncGetWaA(self):
        return self.intArrWaA
    def fncGetWaX(self):
        return self.intArrWaX
    def fncGetWaK(self):
        return self.intArrWaK
    def fncGetWaH(self):
        return self.intArrWaH
    def fncGetWaD(self):
        return self.intArrWaD
    def fncGetWiA(self):
        return self.intArrWiA
    def fncGetWiX(self):
        return self.intArrWiX
    def fncGetWiK(self):
        return self.intArrWiK
    def fncGetWiH(self):
        return self.intArrWiH
    def fncGetWiD(self):
        return self.intArrWiD
    def fncGetTotalArea(self):
        return self.flTotalArea
    def fncGetTotalHeatCapacity(self):
        return self.flTotalHeatCapacity
    def fncGetError(self):
        return self.boError

    def InsertValue(self,Value, flMin, flMax):
        try:
            SaveValue = None
            if type(Value) == int or type(Value) == float:
                if fncCheckValue(Value,flMin, flMax) == False:
                    SaveValue = Value
                else:
                    self.boError = True
            elif type(Value) == str:
                if Value.isdigit() == True:
                    flValue = float(Value)
                    if fncCheckValue(flValue,flMin, flMax) == False:
                        SaveValue = flValue
                    else:
                        self.boError = True
                elif fncIsNumeric(Value) == True:
                    flValue = float(Value)
                    if fncCheckValue(flValue,flMin, flMax) == False:
                        SaveValue = flValue
                    else:
                        self.boError = True
                else:
                    self.boError = True
            else:
                self.boError = True
            return SaveValue
        except:
            print("Error in 'ClsHouseParameters', Function 'InsertValue'")

    def fncReadValues(self, Sheet):
        self.flTotalArea = 0.0
        self.flTotalHeatCapacity = 0.0
        for index, row in Sheet.iterrows():
            intX = -1
            for header in list(Sheet.columns):
                intX = intX + 1
                if header == "Angle":
                    flMin = 0.0
                    flMax = 359.0
                    if pandas.isna(row[intX]) != True:
```

```
                self.intArrAngle.append(self.InsertValue(row[intX], flMin, flMax))

        elif header == "WaA":
            flMin = 1.0
            flMax = 500
            if pandas.isna(row[intX]) != True:
                self.intArrWaA.append(self.InsertValue(row[intX], flMin, flMax))
                self.flTotalArea = self.flTotalArea + self.intArrWaA[index]
            else:
                self.boError = True
        elif header == "WaX":
            flMin = 0.001
            flMax = 1
            if pandas.isna(row[intX]) != True:
                self.intArrWaX.append(self.InsertValue(row[intX], flMin, flMax))
            else:
                self.boError = True
        elif header == "WaK":
            flMin = 0.01
            flMax = 5
            if pandas.isna(row[intX]) != True:
                self.intArrWaK.append(self.InsertValue(row[intX], flMin, flMax))
            else:
                self.boError = True
        elif header == "WaH":
            flMin = 1000
            flMax = 200000
            if pandas.isna(row[intX]) != True:
                self.intArrWaH.append(self.InsertValue(row[intX], flMin, flMax))

                if self.intArrWaH[index] != None:
                    self.flTotalHeatCapacity = self.flTotalHeatCapacity + self.intArrWaH[index]
            else:
                self.boError = True
        elif header == "WaD":
            flMin = 1
            flMax = 3000
            if pandas.isna(row[intX]) != True:
                self.intArrWaD.append(self.InsertValue(row[intX], flMin, flMax))
            else:
                self.boError = True
        elif header == "WiA":
            flMin = 0
            flMax = 500
            if pandas.isna(row[intX]) != True:
                self.intArrWiA.append(self.InsertValue(row[intX], flMin, flMax))
            else:
                self.boError = True
        elif header == "WiX":
            flMin = 0
            flMax = 1
            if pandas.isna(row[intX]) != True:
                self.intArrWiX.append(self.InsertValue(row[intX], flMin, flMax))
            else:
                self.boError = True
        elif header == "WiK":
            flMin = 0
            flMax = 0.9
            if pandas.isna(row[intX]) != True:
                self.intArrWiK.append(self.InsertValue(row[intX], flMin, flMax))
            else:
                self.boError = True
        elif header == "WiH":
            flMin = 0
            flMax = 20000000
            if pandas.isna(row[intX]) != True:
                self.intArrWiH.append(self.InsertValue(row[intX], flMin, flMax))
            else:
                self.boError = True
        elif header == "WiD":
            flMin = 0
            flMax = 3000
            if pandas.isna(row[intX]) != True:
```

```
                    self.intArrWiD.append(self.InsertValue(row[intX], flMin, flMax))
                else:
                    self.boError = True

#Sheet Weather
class ClsWeather:
    def __init__(self):
        self.dtArrDateTime = []
        self.intArrYear = []
        self.intArrMonth = []
        self.intArrDay = []
        self.intArrHour = []
        self.intArrTempAir = []
        self.intArrGHI = []
        self.intArrDNI = []
        self.intArrDHI = []
        self.boError = False

    def fncGetDateTime(self):
        return self.dtArrDateTime
    def fncGetYear(self):
        return self.intArrYear
    def fncGetMonth(self):
        return self.intArrMonth
    def fncGetDay(self):
        return self.intArrDay
    def fncGetHour(self):
        return self.intArrHour
    def fncGetTempAir(self):
        return self.intArrTempAir
    def fncGetGHI(self):
        return self.intArrGHI
    def fncGetDNI(self):
        return self.intArrDNI
    def fncGetDHI(self):
        return self.intArrDHI
    def fncGetError(self):
        return self.boError

    def InsertValue(self,Value, flMin, flMax):
        try:
            SaveValue = None
            if type(Value) == int or type(Value) == float:
                if fncCheckValue(Value,flMin, flMax) == False:
                    SaveValue = Value
            elif type(Value) == str:
                if Value.isdigit() == True or fncIsNumeric(Value) == True:
                    flValue = float(Value)
                    if fncCheckValue(flValue,flMin, flMax) == False:
                        SaveValue = flValue
                    else:
                        self.boError = True
                else:
                    self.boError = True
            else:
                self.boError = True
            return SaveValue
        except:
            print("Error in 'ClsWeather', Function 'InsertValue'")

    def fncReadValues(self, Sheet):
        try:
            intX = 0

            for index, row in Sheet.iterrows():
                intX = -1
                for header in list(Sheet.columns):
                    intX = intX + 1
                    if header == "DateTime":
                        if pandas.isna(row[intX]) != True:
                            if type(row[intX]) == datetime or type(row[intX]) == pandas.Timestamp or
type(row[intX]) == str:
                                self.dtArrDateTime.append(row[intX])
```

```
                        else:
                            self.boError = True
                    else:
                        self.boError = True
                elif header == "year":
                    flMin = 0
                    flMax = 0
                    if pandas.isna(row[intX]) != True:
                        self.intArrYear.append(self.InsertValue(row[intX], flMin, flMax))
                elif header == "month":
                    flMin = 1
                    flMax = 12
                    if pandas.isna(row[intX]) != True:
                        self.intArrMonth.append(self.InsertValue(row[intX], flMin, flMax))
                elif header == "day":
                    flMin = 1
                    flMax = 31
                    if pandas.isna(row[intX]) != True:
                        self.intArrDay.append(self.InsertValue(row[intX], flMin, flMax))
                elif header == "hour":
                    flMin = 1
                    flMax = 24
                    if pandas.isna(row[intX]) != True:
                        self.intArrHour.append(self.InsertValue(row[intX], flMin, flMax))
                elif header == "temp_air":
                    flMin = -50.0
                    flMax = 80.0
                    if pandas.isna(row[intX]) != True:
                        self.intArrTempAir.append(self.InsertValue(row[intX], flMin, flMax))
                    else:
                        self.boError = True
                elif header == "ghi":
                    flMin = 0
                    flMax = 1500
                    if pandas.isna(row[intX]) != True:
                        self.intArrGHI.append(self.InsertValue(row[intX], flMin, flMax))
                    else:
                        self.boError = True
                elif header == "dni":
                    flMin = 0
                    flMax = 1500
                    if pandas.isna(row[intX]) != True:
                        self.intArrDNI.append(self.InsertValue(row[intX], flMin, flMax))
                elif header == "dhi":
                    flMin = 0
                    flMax = 1500
                    if pandas.isna(row[intX]) != True:
                        self.intArrDHI.append(self.InsertValue(row[intX], flMin, flMax))
        except:
            print("Error in 'ClsWeather', Function 'fncReadValues'")

#Sheet Demand
class ClsDemand:
    def __init__(self):
        self.dtArrDateTime = []
        self.intArrPV = []
        self.intArrEd = []
        self.intArrHd = []
        self.boError = False

    def fncGetDateTime(self):
        return self.dtArrDateTime
    def fncGetPV(self):
        return self.intArrPV
    def fncGetEd(self):
        return self.intArrEd
    def fncGetHd(self):
        return self.intArrHd
    def fncGetError(self):
        return self.boError

    def InsertValue(self,Value, flMin, flMax):
        try:
```

```
            SaveValue = None
            if type(Value) == int or type(Value) == float:
                if fncCheckValue(Value,flMin, flMax) == False:
                    SaveValue = Value
            elif type(Value) == str:
                if Value.isdigit() == True or fncIsNumeric(Value) == True:
                    flValue = float(Value)
                    if fncCheckValue(flValue,flMin, flMax) == False:
                        SaveValue = flValue
                    else:
                        self.boError = True
                else:
                    self.boError = True
            else:
                self.boError = True
            return SaveValue
        except:
            print("Error in 'ClsDemand', Function 'InsertValue'")

    def fncReadValues(self, Sheet):
        try:
            intX = 0

            for index, row in Sheet.iterrows():
                intX = -1
                for header in list(Sheet.columns):
                    intX = intX + 1
                    if header == "DateTime":
                        if pandas.isna(row[intX]) != True:
                            if type(row[intX]) == datetime or type(row[intX]) == pandas.Timestamp or
type(row[intX]) == str:
                                self.dtArrDateTime.append(row[intX])
                            else:
                                self.boError = True
                    elif header == "PV":
                        flMin = 0
                        flMax = 0
                        if pandas.isna(row[intX]) != True:
                            self.intArrPV.append(self.InsertValue(row[intX], flMin, flMax))
                    elif header == "Ed":
                        flMin = 0
                        flMax = 0
                        if pandas.isna(row[intX]) != True:
                            self.intArrEd.append(self.InsertValue(row[intX], flMin, flMax))
                    elif header == "Hd":
                        flMin = 0
                        flMax = 0
                        if pandas.isna(row[intX]) != True:
                            self.intArrHd.append(self.InsertValue(row[intX], flMin, flMax))

        except:
            print("Error in 'ClsDemand', Function 'fncReadValues'")

'''
-------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------
Functions
'''
def fncIsNumeric(stValue):
    try:
        float(stValue)
        return True
    except ValueError:
        return False

#control Angle 0 -270
def fncCheckValue(flValue, flMin, flMax):
    try:
        boError = False
        if flMin != 0 and flMax != 0:
            if flValue < flMin or flValue > flMax:
                boError = True
```

```
        return boError
    except:
        print("Error in 'fncCheckValue'")

#read Excel
def readXLSX(strExcelPath):
    try:
        XLSXFile = pandas.read_excel(strExcelPath, sheet_name=None)
        return XLSXFile
    except:
        print("Could not find the Excelfile or a Sheetname")

#Control Excelfile
def controlExcel(XLSXFile,strArrSheetNames):
    try:
        boErrorInExcel = False
        boFound = True
        intCounter = -1

        if len(XLSXFile) > 3:
            for worksheet in XLSXFile.items():
                if boFound == True:
                    boFound = False
                    intCounter = intCounter + 1
                else:
                    boErrorInExcel = True

                if intCounter == 3:
                    break
                for intX in range(len(strArrSheetNames)):
                    if  worksheet[0] == strArrSheetNames[intX]:
                        boFound = True
                        break

        return boErrorInExcel
    except:
        print("Error in 'controlExcel'")

#read Sheets
def readSheets(fileXLSX, strSheetName):
    try:
        Sheet = fileXLSX[strSheetName]
        return Sheet
    except:
        print("Error in 'readSheets'")

#Control Rows and Columns In Sheet
def controlValuesInExcel(Sheet, Columns, Rows):
    try:
        boError = False
        if Sheet.shape[1] >= Columns:
            if Rows != 0:
                if Sheet.shape[0] != Rows:
                    boError = True
        else:
            #wrong number of Columns
            boError = True
        return boError
    except:
        print("Error in 'controlValuesInExcel'")

#check headers in first sheet
def checkHeaders(Sheet, strArrHeaders):
    try:
        boError = False
        intCounter = 0

        for header in list(Sheet.columns):
            for intX in range(0,len(strArrHeaders)):
                if header == strArrHeaders[intX]:
                    intCounter = intCounter + 1
                    break
```

```
        if intCounter != len(strArrHeaders):
            boError = True

        return boError
    except:
        print("Error in 'checkHeaders'")

'''
-------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------
main programm
declare variables
'''

def fncMainCheckInputFile(strExcelPath):

    boEnd = False
    strArrSheetNames = ["House_location", "House_parameters", "Weather", "Demand"]
    strArrHeadersHouseParameters = ["Index", "Angle", "WaA", "WaX", "WaK", "WaH", "WaD", "WiA", "WiX",
"WiK", "WiH","WiD"]
    strArrHeadersWeather = ["DateTime", "temp_air", "ghi"]
    strArrHeadersDemand = ["DateTime", "PV", "Ed", "Hd"]
    intHouseParametersColumns = 0
    intWeatherColumns = 0
    intDemandColumns = 0

#-------------------------------------------------------------------------------------------------------------
-------------------------------------------------
# Start Program

    #read Excel
    XLSXFile = readXLSX(strExcelPath)

    while boEnd == False:
        #control Excel
        if controlExcel(XLSXFile,strArrSheetNames) == True:
            break

        #read all Sheets
        shtHouseParameters = readSheets(XLSXFile,"House_parameters")
        shtWeather = readSheets(XLSXFile,"Weather")
        shtDemand = readSheets(XLSXFile,"Demand")
        shtHVAC = readSheets(XLSXFile, "HVAC")

        clsParameter = ClsHouseParameters()
        clsParameter.fncReadValues(shtHouseParameters)

        clsHVAC = ClsHVAC()
        clsHVAC.fncReadValues(shtHVAC)

        clsWeather = ClsWeather()
        clsWeather.fncReadValues(shtWeather)

        clsDemand = ClsDemand()
        clsDemand.fncReadValues(shtDemand)

        #end loop
        boEnd = True

    return clsParameter, clsWeather, clsDemand, clsHVAC
```