Softwaretool DistrictHeatingSim - Dokumentation Kernfunktionen und -algorithmen

Dipl.-Ing. (FH) Jonas Pfeiffer

2024-09-24

Contents

1 Einleitung

Die Zielsetzungen des SMWK-Projektes fokussieren auf die Entwicklung und Erprobung innovativer Methoden und Werkzeuge für die Planung und Realisierung nachhaltiger Wärmenetze, um die Wärmeversorgung klimaneutral zu gestalten. Die Analyse der umfassenden Datenbasis für Gebäudehüllen, Wärmeverbräuche und Wärmenetze dient als Fundament für präzise Bedarfs- und Potentialanalysen. Auf dieser Basis sollen Werkzeuge entwickelt werden, die sowohl die Erfassung und Aufbereitung relevanter Daten als auch die ganzheitliche Simulation von Wärmenetzen ermöglichen. Diese Simulationen umfassen technische Aspekte der Netzgenerierung und -berechnung sowie ökonomische Betrachtungen zur Bewertung von Wärmeversorgungskonzepten. Die konsequente Weiterentwicklung dieser Ansätze mündet in die Realisierung einer integrierten Softwarelösung, die als zentrales Instrument dient, um die komplexe Aufgabe der Konzeptionierung, Bewertung und Optimierung von Wärmenetzen effizient und nutzerfreundlich zu unterstützen. Diese Softwarelösung bündelt damit die Kernaspekte des Projektes und bildet den roten Faden, der von den initialen Zielstellungen zur praktischen Umsetzung führt.

2 Geocoding

2.1 Skript: `geocodingETRS89.py`

Dieses Skript ermöglicht die Geokodierung von Adressen aus einer CSV-Datei und transformiert die Koordinaten in das ETRS89 / UTM Zone 33N-Koordinatensystem.

2.1.1 Funktion: `get_coordinates(address)`

Beschreibung:
Diese Funktion nimmt eine Adresse als Eingabe entgegen und versucht, die entsprechenden geografischen Koordinaten (Längen- und Breitengrad) mithilfe des Nominatim-Geokodierungsdienst abzurufen. Die erhaltenen Koordinaten werden vom WGS84 (GPS) Koordinatensystem in das ETRS89 / UTM Zone 33N-Koordinatensystem transformiert.

Parameter:

- `address (str)`: Die Adresse, für die die Koordinaten abgerufen werden sollen.

Rückgabewert:
Ein Tuple, das die UTM-Koordinaten (UTM_X und UTM_Y) in Metern enthält, also `(utm_x, utm_y)`. Falls die Geokodierung nicht erfolgreich ist, wird `(None, None)` zurückgegeben.

2.1.2 Funktion: `process_data(input_csv, output_csv)`

Beschreibung:
Diese Funktion liest Daten aus einer Eingabe-CSV-Datei, verarbeitet jede Zeile, geokodiert die Adressen und transformiert diese in UTM-Koordinaten. Die Originaldaten werden zusammen mit den transformierten UTM-Koordinaten in eine Ausgabe-CSV-Datei geschrieben.

Parameter:

- `input_csv (str)`: Pfad zur Eingabe-CSV-Datei, die die zu verarbeitenden Daten enthält. Die Datei sollte die Spalten "Land", "Bundesland", "Stadt", "Adresse" und eventuell zusätzliche Felder enthalten.

- `output_csv (str)`: Pfad zur Ausgabe-CSV-Datei, in der die verarbeiteten Daten mit den UTM-Koordinaten geschrieben werden.

Verhalten:

- Die Funktion öffnet die Eingabe-CSV-Datei zum Lesen und die Ausgabe-CSV-Datei zum Schreiben.
- Sie liest die Daten aus der Eingabe-CSV-Datei Zeile für Zeile, extrahiert relevante Informationen wie Land, Bundesland, Stadt und Adresse.
- Aus diesen Informationen wird ein vollständiger Adressstring erstellt.
- Die Funktion ruft `get_coordinates` auf, um die vollständige Adresse zu geokodieren und die UTM-Koordinaten zu erhalten.
- Die ursprünglichen Daten werden um zwei Spalten, "UTM_X" und "UTM_Y", erweitert, die die berechneten UTM-Koordinaten enthalten.
- Die verarbeiteten Daten werden in die Ausgabe-CSV-Datei geschrieben.
- Nach Abschluss gibt die Funktion "Verarbeitung abgeschlossen" aus.

Hinweis:
Am Ende des Skripts befindet sich ein Code-Schnipsel, der zeigt, wie die `process_data`-Funktion mit Eingabe- und Ausgabe-CSV-Dateipfaden aufgerufen wird. Dieser Teil kann aktiviert und angepasst werden, um die Geokodierung und Datenverarbeitung durchzuführen.

Zusammenfassung:
Diese Funktion ermöglicht die Geokodierung von Adressen aus einer CSV-Datei, das Abrufen der entsprechenden UTM-Koordinaten und das Erstellen einer neuen CSV-Datei mit den Originaldaten und den hinzugefügten UTM-Koordinaten. Die erstellte Datei kann für weitere räumliche Analysen oder Kartenanwendungen verwendet werden.

3 OSM

3.1 Skript: `import_osm_data_geojson.py`

Dieses Skript ist darauf ausgelegt, OpenStreetMap (OSM)-Daten herunterzuladen und in das GeoJSON-Format umzuwandeln. Der Fokus liegt dabei auf bestimmten Elementen (z. B. Straßen oder Gebäuden) innerhalb eines angegebenen Stadtgebiets. Das Skript enthält Funktionen zum Erstellen einer Overpass-Abfrage, zum Herunterladen von Daten, zur Verarbeitung der Daten in GeoJSON-Features und zum Speichern der resultierenden GeoJSON-Datei.

3.1.1 Funktion: `build_query(city_name, tags, element_type="way")`

Beschreibung:
Generiert eine Overpass-Abfrage, um OSM-Daten basierend auf festgelegten Kriterien wie Stadtname, Tags (Schlüssel-Wert-Paare) und Elementtyp abzurufen. Der Standardwert für `element_type` ist `"way"` für Straßen.

Rückgabewert:
Gibt die erstellte Abfrage als Zeichenfolge zurück.

3.1.2 Funktion: `download_data(query, element_type)`

Beschreibung:
Führt die Overpass-Abfrage aus, um OSM-Daten abzurufen und in GeoJSON-Features zu konvertieren. Die Funktion unterstützt zwei Elementtypen: `"way"` für Straßen und `"building"` für Gebäude.

Rückgabewert:
Gibt eine `GeoJSON-FeatureCollection` zurück, die die heruntergeladenen Daten enthält.

3.1.3 Funktion: `json_serial(obj)`

Beschreibung:
Ein JSON-Serialisierer, der zum Umgang mit Objekten dient, die nicht nativ in JSON serialisierbar sind. Diese Funktion konvertiert `Decimal`-Objekte in Gleitkommazahlen.

Anwendungsfall:
Wird in der Funktion `save_to_file` zur Serialisierung von `Decimal`-Objekten verwendet.

3.1.4 Funktion: `save_to_file(geojson_data, filename)`

Beschreibung:
Speichert die GeoJSON-Daten in einer angegebenen Datei mit richtiger Formatierung. Verwendet die Funktion `json_serial`, um `Decimal`-Objekte zu serialisieren.

3.1.5 Funktion: `run_here()`

Beschreibung:
Definiert Standardparameter für Stadtname und Tags (z. B. Straßentyp), erstellt eine Overpass-Abfrage, lädt OSM-Daten herunter und speichert diese als GeoJSON-Datei. Diese Funktion kann entkommentiert und ausgeführt werden, um das Skript mit vordefinierten Einstellungen auszuführen.

3.2 Hinweis

- Das Skript ermöglicht es, die Datenextraktion anzupassen, indem Stadt, OSM-Tags und der Elementtyp (`way` oder `building`) festgelegt werden.
- Die `run_here()`-Funktion kann entkommentiert und ausgeführt werden, um das Skript mit vordefinierten Parametern zu starten.

Zusammenfassung:
Dieses Skript ist ein nützliches Werkzeug, um OSM-Daten abzurufen und in das GeoJSON-Format umzuwandeln. Dadurch werden die Daten für verschiedene geospatiale Anwendungen und Analysen nutzbar.

4 Einleitung

Dieses Dokument beschreibt die Funktionen in den Skripten `MST_processing.py`, `simple_MST.py`, und `import_and_create_layers.py`. Die Skripte sind darauf ausgelegt, OSM-Daten zu verarbeiten, räumliche Netzwerke basierend auf Minimal Spanning Tree (MST)-Algorithmen zu erstellen und die Ergebnisse als GeoJSON-Dateien zu exportieren.

5 Skript: MST_processing.py

5.1 Übersicht

Das Skript `MST_processing.py` enthält Funktionen zur Nachbearbeitung von MST-Ergebnissen. Es erlaubt die Anpassung der MST-Segmente an Straßenverläufe und die Einführung von Zwischenpunkten zwischen gegebenen Punkten und den nächstgelegenen Straßen.

5.2 Funktion: `add_intermediate_points(points_gdf, street_layer, max_distance=200, point_interval=10)`

Beschreibung:
Diese Funktion fügt Zwischenpunkte zwischen gegebenen Punkten und den nächstgelegenen Straßen hinzu. Dies ist nützlich, um Netzwerke zu verfeinern und genauere Straßenanbindungen zu gewährleisten. Es werden Punkte in regelmäßigen Abständen (basierend auf `point_interval`) entlang der Verbindung zwischen den gegebenen Punkten und den Straßen hinzugefügt.

5.3 Funktion: `adjust_segments_to_roads(mst_gdf, street_layer, all_end_points_gdf, threshold=5)`

Beschreibung:
Passt die MST-Segmente so an, dass sie den Straßenlinien genauer folgen. Die Funktion iteriert durch die MST-Segmente und passt die Linien an die Straßen an, wenn sie eine bestimmte Distanz (`threshold`) überschreiten.

5.4 Funktion: `generate_mst(points)`

Beschreibung:
Erstellt einen Minimal Spanning Tree (MST) basierend auf einem Satz von Punkten. Die Kanten des MST werden als `LineString`-Objekte in einem `GeoDataFrame` gespeichert.

6 Skript: simple_MST.py

6.1 Übersicht

Das Skript `simple_MST.py` enthält die wesentlichen Funktionen zur Erstellung eines MST-Netzwerks und zur Verarbeitung von räumlichen Daten. Es verwendet verschiedene Algorithmen, um Netzwerke zu generieren und den Verlauf von Straßen in das Netzwerk zu integrieren.

6.2 Funktion: `create_offset_points(point, distance, angle_degrees)`

Beschreibung:
Erzeugt einen Punkt, der um einen bestimmten Abstand und Winkel von einem gegebenen Punkt versetzt ist. Diese Funktion wird verwendet, um neue Punkte in einem festgelegten Abstand von bestehenden Punkten zu generieren.

6.3 Funktion: `generate_network_fl(layer_points_fl, layer_wea, street_layer, algorithm="MST")`

Beschreibung:
Erzeugt ein Netzwerk aus den Fließpunkten (`layer_points_fl`) und den Wärmeaustauscherpunkten (`layer_wea`) unter Berücksichtigung der Straßenverläufe (`street_layer`). Die Funktion unterstützt verschiedene Algorithmen zur Netzwerkgenerierung, einschließlich MST und A*-Algorithmus.

6.4 Funktion: `generate_return_lines(layer_points_rl, layer_wea, fixed_distance_rl, fixed_angle_rl, street_layer)`

Beschreibung:
Erstellt Rücklaufleitungen basierend auf den Rücklaufpunkten (`layer_points_rl`) und Wärmeaustauscherpunkten (`layer_wea`). Die Leitungen werden versetzt und entlang der nächstgelegenen Straßen angeordnet.

7 Skript: `import_and_create_layers.py`

7.1 Übersicht

Das Skript `import_and_create_layers.py` dient der Verarbeitung von OSM-Daten und der Erstellung von Netzwerkschichten. Es kann Geodaten importieren, Schichten basierend auf den Daten erstellen und die Ergebnisse als GeoJSON-Dateien exportieren.

7.2 Funktion: `import_osm_street_layer(osm_street_layer_geojson_file)`

Beschreibung:
Importiert die OSM-Straßenschicht aus einer GeoJSON-Datei und gibt sie als `GeoDataFrame` zurück. Diese Funktion ermöglicht die Verwendung von OSM-Daten für die Netzwerkgenerierung.

7.3 Funktion: `generate_lines(layer, distance, angle_degrees, df=None)`

Beschreibung:
Generiert Linien, die von den Punkten in der `GeoDataFrame layer` um eine bestimmte Distanz und einen bestimmten Winkel versetzt sind. Die Funktion kann zusätzlich Attribute aus einer übergebenen Datenstruktur (`df`) übernehmen.

7.4 Funktion: `generate_and_export_layers(osm_street_layer_geojson_file_n data_csv_file_name, coordinates, base_path, fixed_angle=0, fixed_distance=1, algorithm="MST")`

Beschreibung:
Erzeugt die Netzwerkschichten basierend auf den gegebenen Daten und exportiert diese als GeoJSON-Dateien. Es verwendet verschiedene Algorithmen zur Netzwerkgenerierung, einschließlich des MST-Algorithmus.

8 Zusammenfassung

Die bereitgestellten Skripte bieten eine umfassende Lösung zur Erstellung von Netzwerken basierend auf räumlichen Daten und der Straßeninfrastruktur. Die Funktionen ermöglichen die flexible Anpassung der Netzwerke an Straßenverläufe, die Generierung von MST-Netzwerken sowie die einfache Verarbeitung und den Export der Ergebnisse in GeoJSON-Format.

9 Einleitung

Dieses Dokument beschreibt die Methodik zur Berechnung von Heizlastprofilen für Gebäude basierend auf CSV-Daten. Die Implementierung beinhaltet Funktionen zur Ermittlung des Heizwärme- und Warmwasserbedarfs von Gebäuden, welche die Berechnungsmethoden VDI 4655 und BDEW verwenden. Zusätzlich werden die Vor- und Rücklauftemperaturen der Heizsysteme berechnet.

10 Funktion `generate_profiles_from_csv`

Die Funktion `generate_profiles_from_csv` berechnet die Heizprofile eines Gebäudes auf Grundlage der folgenden Eingabedaten:

- **data**: Ein DataFrame mit Gebäudeinformationen, insbesondere:
  - Wärmebedarf in kWh
  - Gebäudetyp
  - Subtyp
  - Anteil des Warmwasserbedarfs am Gesamtwärmebedarf
  - Normaußentemperatur
- **TRY**: Pfad zu den Testreferenzjahresdaten (TRY), die stündliche Wetterdaten (z.B. Lufttemperaturen) enthalten.
- **calc_method**: Die Berechnungsmethode zur Ermittlung des Wärmebedarfs, basierend auf dem Gebäudetyp oder einer angegebenen Methode.

10.1 Berechnungslogik

Die Funktion führt folgende Schritte aus:

10.1.1 Aufteilung des Gesamtwärmebedarfs

Der Gesamtwärmebedarf wird in Heizwärme und Warmwasserbedarf aufgeteilt:

$$\text{Heizwärmebedarf} = \text{Gesamtwärmebedarf} \times (1 - \text{Warmwasserbedarf})$$

$$\text{Warmwasserbedarf} = \text{Gesamtwärmebedarf} \times \text{Warmwasserbedarf}$$

10.1.2 Berechnungsmethoden

Je nach Gebäudetyp wird die Berechnung entweder nach der Methode VDI 4655 oder BDEW durchgeführt.

VDI 4655 Für bestimmte Gebäudetypen, wie Einfamilienhäuser (EFH) und Mehrfamilienhäuser (MFH), kann die Methodik nach VDI 4655 verwendet werden. Diese Methode berechnet viertelstündliche Lastprofile für Heizung, Warmwasser und Strom. Eine ausführliche Beschreibung erfolgt in Kapitel 12.

BDEW Die Methodik der Standardlastprofile nach BDEW bieten hingegen deutlich heterogene Gebäudetypen. Neben Ein- und Mehrfamilienhäusern (HEF, HMF) sind das Gebäudenutzungstypen wie Gewerbebauten oder Bürobauten. Die Berechnungsmethode gibt stündliche Lastprofile für Heizung und Warmwasser aus. Eine ausführliche Beschreibung erfolgt in Kapitel 13.

Die spezifische Berechnungsmethode wird basierend auf dem Gebäudetyp ausgewählt.

10.1.3 Korrektur negativer Lasten

Um physikalisch unsinnige negative Werte zu vermeiden, werden alle negativen stündlichen Lasten auf 0 gesetzt:

$$\text{hourly\_heat\_demand\_total\_kW} = \max(0, \text{hourly\_heat\_demand\_total\_kW})$$

10.1.4 Umrechnung in Watt

Die berechneten Lasten in kW werden in Watt umgerechnet:

$$\text{total\_heat\_W} = \text{hourly\_heat\_demand\_total\_kW} \times 1000$$

10.2 Ausgabe

Die Funktion gibt folgende Werte zurück:

- **yearly_time_steps**: Stündliche Zeitpunkte über das Jahr hinweg.
- **total_heat_W**: Gesamtwärmelast in Watt.
- **heating_heat_W**: Heizwärmelast in Watt.
- **warmwater_heat_W**: Warmwasserlast in Watt.
- **max_heat_requirement_W**: Maximaler Wärmebedarf in Watt.
- **supply_temperature_curve**: Vorlauftemperaturkurve des Gebäudes.
- **return_temperature_curve**: Rücklauftemperaturkurve des Gebäudes.
- **hourly_air_temperatures**: Stündliche Außentemperaturen.

11 Funktion `calculate_temperature_curves`

Diese Funktion berechnet die Vor- und Rücklauftemperaturkurven eines Gebäudes basierend auf den stündlichen Lufttemperaturen.

11.1 Eingabe

- **data**: Ein DataFrame mit den Vor- und Rücklauftemperaturen der Heizsysteme sowie den Steigungen der Heizkurve für jedes Gebäude.
- **hourly_air_temperatures**: Array mit stündlichen Außentemperaturen.

11.2 Berechnung der Temperaturkurven

11.2.1 Temperaturdifferenz

Die Differenz zwischen Vor- und Rücklauftemperatur ΔT wird für jedes Gebäude berechnet:

$$\Delta T = VLT_{\max} - RLT_{\max}$$

Dabei ist $VLT_{\max}$ die maximale Vorlauftemperatur und $RLT_{\max}$ die maximale Rücklauftemperatur des Gebäudes.

11.2.2 Vorlauftemperaturkurve

Die Vorlauftemperatur wird basierend auf der Außentemperatur und der Steigung der Heizkurve s berechnet. Für Außentemperaturen unterhalb der Normaußentemperatur $T_{\min}$ bleibt die Vorlauftemperatur konstant:

$$\text{Vorlauftemperatur} = VLT_{\max}, \quad \text{wenn } T_{\text{außen}} \leq T_{\min}$$

Wenn die Außentemperatur $T_{\text{außen}}$ größer ist als $T_{\min}$, wird die Vorlauftemperatur gemäß folgender Gleichung angepasst:

$$\text{Vorlauftemperatur} = VLT_{\max} + s \times (T_{\text{außen}} - T_{\min})$$

11.2.3 Rücklauftemperaturkurve

Die Rücklauftemperaturkurve wird durch Subtraktion der Temperaturdifferenz ΔT von der Vorlauftemperatur berechnet:

$$\text{Rücklauftemperatur} = \text{Vorlauftemperatur} - \Delta T$$

12 Berechnungsmethode: VDI 4655

Die VDI-Richtlinie 4655 beschreibt Verfahren zur Berechnung von Lastprofilen für Wohngebäude, insbesondere im Hinblick auf Heizwärme-, Warmwasser- und Strombedarfe. Das Ziel des hier beschriebenen Algorithmus ist es, diese Lastprofile auf Basis von Testreferenzjahrsdaten (TRY), Gebäude- und Haushaltsinformationen sowie Temperatur- und Wolkendaten zu erstellen.

Die Methode zur Berechnung von Wärmebedarfsprofilen basiert auf einer Aufteilung des Jahresbedarfs in Viertelstundenintervalle, wobei Faktoren für typische Verbrauchstage sowie saisonale und witterungsbedingte Einflüsse berücksichtigt werden.

12.1 Grundlage des Berechnungsalgorithmus nach VDI 4655

Die Berechnungsmethode der VDI 4655 orientiert sich an der Aufteilung des Jahresenergieverbrauchs (Heizung, Warmwasser, Strom) in detaillierte Viertelstundenprofile. Diese Profile werden durch die Kombination von Temperatur- und Wolkendaten mit typischen Verbrauchsprofilen für Gebäude- und Haushaltsarten erstellt. Die Berücksichtigung von saisonalen Schwankungen und verschiedenen Klimazonen ermöglicht eine realitätsnahe Simulation des Energieverbrauchs.

12.1.1 Testreferenzjahr (TRY) und Wetterdaten

Die Basis der Berechnungen bildet das sogenannte Testreferenzjahr (TRY), das Wetterdaten wie stündliche Temperaturen und Bewölkungsgrade enthält. Diese Daten werden verwendet, um den Einfluss der Außenbedingungen auf den Heizwärmebedarf sowie auf den Strombedarf für die Warmwasserbereitung zu modellieren.

Die Wetterdaten werden aus einer TRY-Datei eingelesen, und es werden die folgenden Größen extrahiert:

- **Temperatur (T)**: Die stündliche Außentemperatur wird zur Berechnung des Heizbedarfs genutzt.
- **Bewölkungsgrad (N)**: Der Bewölkungsgrad beeinflusst den Strombedarf für Licht und Geräte sowie den Heizbedarf.

12.1.2 Definition von Nutzungsprofilen

Für verschiedene Gebäudetypen und Haushaltsgrößen (z.B. Einfamilienhäuser, Mehrfamilienhäuser) werden in der VDI 4655 typische Tagesprofile definiert. Diese Profile spiegeln das Nutzungsverhalten über den Tag hinweg wider und variieren je nach Gebäudetyp, Tag (Werk-, Wochen- oder Feiertag) und Jahreszeit (Sommer, Übergangszeit, Winter).

Die Jahreszeit wird anhand der durchschnittlichen Tagestemperatur T_{avg} folgendermaßen bestimmt:

$$\text{Saison} = \begin{cases} \text{Winter (W)} & \text{wenn } T_{\text{avg}} < 5°\text{C} \\ \text{Übergangszeit (Ü)} & \text{wenn } 5°\text{C} \leq T_{\text{avg}} \leq 15°\text{C} \\ \text{Sommer (S)} & \text{wenn } T_{\text{avg}} > 15°\text{C} \end{cases}$$

Jeder Tag wird als Wochentag oder Wochenende/Feiertag klassifiziert, was zu einem kombinierten Profiltag führt, z.B. "WSH" (Winter, Wochentag, hoher Bewölkungsgrad).

12.2 Berechnungsansatz nach VDI 4655

12.2.1 Jahresenergieverbrauch und Aufteilung auf tägliche Profile

Der Jahresenergieverbrauch (JEV) wird für Heizung, Warmwasser und Strom separat angegeben. Dieser wird auf die Tage des Jahres verteilt, wobei tages-, saison- und klimazonenspezifische Faktoren berücksichtigt werden.

Die Tagesbedarfe für Heizung und Warmwasser werden folgendermaßen berechnet:

$$Q_{\text{Tag, Heizung}} = JEV_{\text{Heizung}} \cdot f_{\text{Heizung, TT}}$$

$$Q_{\text{Tag, WW}} = JEV_{\text{WW}} \cdot f_{\text{WW, TT}}$$

wobei:

- $f_{\text{Heizung, TT}}$ und $f_{\text{WW, TT}}$ spezifische Tagesfaktoren sind, die den Einfluss der Saison, des Tages und des Klimas berücksichtigen.

12.2.2 Berechnung stündlicher und viertelstündlicher Lastprofile

Nachdem der tägliche Energiebedarf ermittelt wurde, wird dieser auf stündliche und viertelstündliche Intervalle verteilt. Die Aufteilung erfolgt auf Basis der in der VDI 4655 definierten Standardlastprofile, die typische Nutzungszyklen im Tagesverlauf widerspiegeln. Dabei werden für jede Viertelstunde des Tages spezifische Lastfaktoren verwendet.

Für die viertelstündliche Aufteilung wird der Tagesbedarf Q_{Tag} auf 96 Viertelstunden des Tages verteilt:

$$Q_{\text{15min, Heizung}} = Q_{\text{Tag, Heizung}} \cdot f_{\text{15min, Heizung}}$$

$$Q_{\text{15min, WW}} = Q_{\text{Tag, WW}} \cdot f_{\text{15min, WW}}$$

Hierbei ist f_{15min} der Lastfaktor, der für jede Viertelstunde eines Tages gilt.

12.2.3 Korrektur der Lastprofile basierend auf tatsächlichem Verbrauch

Der tatsächliche Energieverbrauch kann von den Standardwerten der VDI 4655 abweichen. In diesem Fall erfolgt eine Korrektur der normierten viertelstündlichen Profile. Der korrigierte viertelstündliche Bedarf wird folgendermaßen berechnet:

$$Q_{\text{15min, korr}} = \frac{Q_{\text{15min, norm}}}{\sum Q_{\text{15min, norm}}} \cdot JEV$$

Hierbei wird der normierte viertelstündliche Bedarf $Q_{\text{15min, norm}}$ so skaliert, dass er den tatsächlichen Jahresenergieverbrauch JEV berücksichtigt.

12.3 Anwendungsbereiche des VDI 4655-Profils

Das VDI 4655-Verfahren wird vor allem zur Simulation und Modellierung von Energieverbrauchsprofilen in Wohngebäuden verwendet. Anwendungsbereiche sind:

- **Simulation von Lastprofilen**: Ermöglicht eine detaillierte Simulation des stündlichen oder viertelstündlichen Energiebedarfs für Heizung, Warmwasser und Strom.
- **Netzplanung und Dimensionierung**: Hilft bei der Planung und Dimensionierung von Heiz- und Stromnetzen, insbesondere in Fernwärme- oder Stromversorgungsnetzen.
- **Optimierung der Energienutzung**: Liefert eine Basis für die Optimierung der Energienutzung und die Integration erneuerbarer Energien.

12.4 Zusammenfassung

Das VDI 4655-Verfahren bietet eine strukturierte Methode zur Berechnung von detaillierten Energieverbrauchsprofilen für Heizung, Warmwasser und Strom. Durch die Berücksichtigung von Wetterdaten, Gebäudetypen und typischen Tages- und Jahresprofilen ermöglicht es eine realitätsnahe Simulation des Energiebedarfs von Wohngebäuden.

Das Verfahren ermöglicht die Erstellung viertelstündlicher Profile, die zur Netzplanung und zur Optimierung der Energieversorgung verwendet werden können. Besonders hervorzuheben ist die Anpassbarkeit des Verfahrens an unterschiedliche Klimazonen und Gebäudetypen, was es für eine Vielzahl von Anwendungsszenarien geeignet macht.

13 Berechnungsmethode: BDEW

Das BDEW-Standardlastprofilverfahren (SLP) ist eine weit verbreitete Methode zur Berechnung des stündlichen Wärmebedarfs eines Gebäudes basierend auf Jahresenergieverbrauch und Wetterbedingungen. Es wird verwendet, um typische Lastprofile für die Heizwärme (Raumwärme) und den Warmwasserbedarf eines Gebäudes zu erstellen. Dies ermöglicht eine detaillierte Simulation von Energieverbrauchsprofilen, die in der Energiewirtschaft genutzt werden können, um z.B. die Planung und Steuerung von Wärmenetzen zu optimieren.

Die Berechnung basiert auf einer Kombination von physikalischen und statistischen Modellen, die temperaturabhängige Profile und Nutzungsfaktoren über den Tages-, Wochen- und Jahresverlauf berücksichtigen. Das Ziel ist es, den gesamten jährlichen Wärmebedarf auf stündlicher Basis realitätsgetreu zu modellieren.

13.1 Grundlegende Komponenten der Wärmebedarfsberechnung

Der Gesamtwärmebedarf eines Gebäudes setzt sich aus zwei wesentlichen Komponenten zusammen:

- **Heizwärmebedarf (HWB)**: Die Energie, die benötigt wird, um die Raumtemperatur auf einem gewünschten Niveau zu halten. Diese hängt stark von der Außentemperatur, der Gebäudedämmung und den internen Wärmelasten ab.
- **Warmwasserbedarf (WWB)**: Der Energiebedarf für die Erwärmung des Brauchwassers für Haushaltszwecke. Dieser Bedarf ist im Gegensatz zum Heizwärmebedarf weitgehend unabhängig von der Außentemperatur, wird aber durch das Nutzungsverhalten bestimmt.

Das BDEW-SLP-Verfahren nutzt verschiedene Koeffizienten und Faktoren, um den Einfluss dieser beiden Komponenten auf den täglichen und stündlichen Wärmebedarf zu modellieren.

13.2 Jahreswärmebedarf und Tagesprofile

13.2.1 Ausgangspunkt: Der Jahreswärmebedarf (JWB)

Der Jahreswärmebedarf eines Gebäudes wird meist in Kilowattstunden (kWh) angegeben und beschreibt den gesamten Energieverbrauch für Heizung und Warmwasser über ein

Jahr. Dieser Wert wird in der Praxis z.B. durch Abrechnungsdaten oder Messungen ermittelt und stellt die Grundlage der weiteren Berechnungen dar.

13.2.2 Aufteilung in tägliche Profile

Der erste Schritt besteht darin, den Jahreswärmebedarf auf die einzelnen Tage des Jahres aufzuteilen. Diese Aufteilung erfolgt auf Basis der täglichen Temperaturdaten und der spezifischen Tagesprofile, die vom BDEW vorgegeben werden. Das Tagesprofil bestimmt, wie der Wärmebedarf an einem bestimmten Tag (z.B. ein Montag im Januar) aussieht.

Für jeden Tag wird der Heizwärmebedarf folgendermaßen berechnet:

$$Q_{\text{Tag, Heizung}} = f_{\text{Tagesprofil}} \cdot F_{\text{Tagesfaktor}} \cdot m_H \cdot T_{\text{avg}} + b_H$$

Hierbei sind:

- $f_{\text{Tagesprofil}}$: Ein spezifischer Koeffizient, der das Heizverhalten an einem bestimmten Tag beschreibt.
- $F_{\text{Tagesfaktor}}$: Ein tagesabhängiger Faktor, der den Einfluss des Wochentages oder Feiertags auf den Wärmebedarf darstellt.
- m_H und b_H: Lineare Koeffizienten, die den Temperaturverlauf über den Tag hinweg berücksichtigen.
- T_{avg}: Die Tagesdurchschnittstemperatur.

13.2.3 Temperaturabhängige Berechnung des Heizwärmebedarfs

Der Heizwärmebedarf ist eng an die Außentemperatur gekoppelt. Bei niedrigeren Außentemperaturen muss mehr Energie für das Heizen aufgewendet werden, um die Raumtemperatur konstant zu halten. Der Heizwärmebedarf wird durch eine temperaturabhängige Funktion modelliert:

$$Q_{\text{Heizung}}(T) = \frac{A}{1 + \left(\frac{B}{T_{\text{ref}} - 40}\right)^C} + m_H \cdot T_{\text{avg}} + b_H$$

wobei:

- A, B, und C profiltypische Koeffizienten sind, die das spezifische Heizverhalten des Gebäudes definieren.
- T_{ref} ist die Referenztemperatur, die aus den stündlichen Temperaturdaten berechnet wird.

Diese Funktion stellt sicher, dass bei extrem niedrigen Außentemperaturen der Heizbedarf stark ansteigt, während er bei höheren Temperaturen entsprechend abnimmt.

13.2.4 Berechnung des Warmwasserbedarfs

Der Warmwasserbedarf wird durch eine ähnliche Gleichung wie der Heizwärmebedarf modelliert, jedoch ist er weniger stark von der Außentemperatur abhängig. Die Berechnung erfolgt über die Gleichung:

$$Q_{\mathrm{WW}}(T) = m_W \cdot T_{\mathrm{avg}} + b_W$$

wobei:

- m_W und b_W lineare Koeffizienten für den Warmwasserbedarf sind.
- T_{avg} die Tagesdurchschnittstemperatur ist.

Für das Warmwasser ist die Temperaturabhängigkeit weniger relevant, da der Warmwasserbedarf eher durch das Nutzungsverhalten (z.B. morgendliches Duschen) bestimmt wird.

13.3 Tages- und Wochenfaktoren

Neben den Temperaturabhängigkeiten werden auch tages- und wochenabhängige Faktoren in die Berechnung einbezogen. Diese Faktoren spiegeln das typische Verbrauchsverhalten an verschiedenen Wochentagen wider. So ist der Wärmebedarf an einem Montag anders als an einem Sonntag, da der Montag typischerweise ein Arbeitstag ist und andere Heizmuster vorliegen.

Die tagesabhängigen Lastprofile werden über Faktoren F_{Tag} angepasst:

$$F_{\mathrm{Tag}} = F_{\mathrm{Wochentag}} \cdot F_{\mathrm{Wochenfaktor}} \cdot F_{\mathrm{Feiertag}}$$

Diese Faktoren berücksichtigen z.B. die geringere Nutzung an Wochenenden oder Feiertagen und reduzieren den berechneten Heiz- oder Warmwasserbedarf entsprechend.

13.4 Berechnung stündlicher Lastprofile

13.4.1 Aufteilung des Tageswärmebedarfs auf Stunden

Nachdem der Tageswärmebedarf für Heizung und Warmwasser berechnet wurde, wird dieser Bedarf auf die Stunden des Tages verteilt. Hierbei wird das typische Nutzungsverhalten im Tagesverlauf berücksichtigt, indem stündliche Koeffizienten f_{Stunde} verwendet werden. Diese Koeffizienten geben an, welcher Anteil des Tagesbedarfs in einer bestimmten Stunde auftritt.

Der stündliche Wärmebedarf wird mit der folgenden Interpolationsformel berechnet:

$$Q_{\mathrm{Heizung,\ Stunde}} = Q_{\mathrm{Tag,\ Heizung}} \cdot \left(f_{\mathrm{Stunde}} + \frac{T_{\mathrm{aktuell}} - T_{\mathrm{Grenze}}}{5} \cdot (f_{\mathrm{Stunde,\ T1}} - f_{\mathrm{Stunde,\ T2}}) \right)$$

Hierbei wird zwischen zwei Temperaturgrenzwerten T_{Grenze} interpoliert, um einen fließenden Übergang zwischen den stündlichen Lasten zu gewährleisten.

13.4.2 Stündliche Profile für Heizung und Warmwasser

Der Wärmebedarf wird für jede Stunde des Tages sowohl für die Heizung als auch für den Warmwasserbedarf berechnet. Diese stündlichen Profile sind wichtig, um den Verbrauch über den Tag hinweg detailliert abzubilden. Insbesondere bei stark schwankenden Außentemperaturen ergeben sich deutliche Unterschiede im stündlichen Heizwärmebedarf.

13.5 Anpassung des Warmwasseranteils am Gesamtwärmebedarf

Falls der tatsächliche Warmwasseranteil bekannt ist, kann dieser im Modell berücksichtigt werden. Der initiale Warmwasseranteil wird als Verhältnis des berechneten Warmwasserbedarfs zum gesamten Wärmebedarf ermittelt:

$$\text{WW-Anteil} = \frac{Q_{\text{WW}}}{Q_{\text{Heizung}} + Q_{\text{WW}}}$$

Falls der tatsächliche Warmwasseranteil vom berechneten abweicht, kann dieser durch einen Korrekturfaktor angepasst werden. Die Berechnung erfolgt durch Skalierung des berechneten Warmwasser- und Heizwärmebedarfs mit entsprechenden Korrekturfaktoren.

13.6 Zusammenfassung

Das BDEW-Lastprofilverfahren bietet eine detaillierte Methode zur Berechnung des stündlichen Wärmebedarfs auf Basis von Jahresverbrauchsdaten und Wetterdaten. Es berücksichtigt sowohl die Temperaturabhängigkeit des Heizbedarfs als auch tageszeitliche und wochenabhängige Faktoren. Das Verfahren eignet sich hervorragend zur Modellierung von Wärmeverbrauchspro die für die Steuerung und Optimierung von Wärmenetzen genutzt werden können.

Durch die Berücksichtigung von tages- und stundenbasierten Faktoren sowie der Anpassung an reale Verbrauchsdaten (z.B. durch den Warmwasseranteil) liefert das Verfahren genaue und praxisnahe Ergebnisse. Es ermöglicht eine präzise Abschätzung des stündlichen Wärmebedarfs für unterschiedliche Gebäudetypen und Nutzungsverhalten.

14 Wärmeverlust durch Transmission

Der Wärmeverlust eines Gebäudes durch Transmission kann durch die folgende Gleichung beschrieben werden:

$$\dot{Q}_{\text{Trans}} = U \cdot A \cdot \Delta T \tag{1}$$

wobei:

- $\dot{Q}_{\text{Trans}}$: Wärmestrom (Watt, W)
- U: U-Wert des Bauteils (W/m^{2}K)
- A: Fläche des Bauteils (m^2)
- ΔT: Temperaturdifferenz zwischen Innen- und Außenseite des Bauteils (K)

Für ein Gebäude summieren sich die Wärmeverluste durch alle Bauteile:

$$\dot{Q}_{\text{Gesamt}} = \sum_i U_i \cdot A_i \cdot \Delta T \tag{2}$$

15 Wärmeverlust durch Lüftung

Der Wärmeverlust durch Lüftung wird durch den Luftaustausch zwischen Innenraum und Außenluft verursacht und ist wie folgt beschrieben:

$$\dot{Q}_{\text{Lüftung}} = 0.34 \cdot n \cdot V \cdot \Delta T \tag{3}$$

wobei:

- $\dot{Q}_{\text{Lüftung}}$: Lüftungswärmeverlust (W)
- n: Luftwechselrate h^{-1}
- V: Volumen des Gebäudes (m^3)
- ΔT: Temperaturdifferenz zwischen Innen- und Außenluft (K)

16 Maximaler Heizwärmebedarf

Der maximale Heizwärmebedarf ergibt sich aus der Summe der Wärmeverluste durch Transmission und Lüftung:

$$Q_{\text{max}} = \dot{Q}_{\text{Trans}} + \dot{Q}_{\text{Lüftung}} \tag{4}$$

17 Jährlicher Heizwärmebedarf

Der jährliche Heizwärmebedarf wird durch die Integration der stündlichen Heizbedarfe über das Jahr berechnet:

$$Q_{\text{Heizung}}(\text{Jahr}) = \sum_{\text{alle Stunden}} \max\left(0, m \cdot T_{\text{Außen}} + b\right) \tag{5}$$

wobei:

- m: Steigung der linearen Beziehung zwischen Heizbedarf und Außentemperatur
- $T_{\text{Außen}}$: Außentemperatur (K)
- b: Y-Achsenabschnitt, abhängig von den spezifischen U-Werten und der gewünschten Raumtemperatur

18 Jährlicher Warmwasserbedarf

Der jährliche Warmwasserbedarf wird als spezifischer Verbrauch pro Quadratmeter Wohnfläche berechnet:

$$Q_{\text{WW}} = WW_{\text{Bedarf}} \cdot A \cdot \text{Stockwerke} \tag{6}$$

wobei:

- Q_{WW}: Jährlicher Warmwasserbedarf (kWh)

- WW_{Bedarf}: Warmwasserbedarf pro Quadratmeter Wohnfläche (kWh/m^2)
- A: Grundfläche des Gebäudes (m^2)
- Stockwerke: Anzahl der Stockwerke

19 Gesamtwärmebedarf

Der gesamte jährliche Wärmebedarf des Gebäudes setzt sich aus dem jährlichen Heizwärmebedarf und dem jährlichen Warmwasserbedarf zusammen:

$$Q_{\text{Gesamt}} = Q_{\text{Heizung}} + Q_{\text{WW}} \tag{7}$$

Der Anteil des Warmwassers am gesamten Wärmebedarf kann berechnet werden durch:

$$\text{Warmwasseranteil} = \frac{Q_{\text{WW}}}{Q_{\text{Gesamt}}} \cdot 100 \tag{8}$$

20 Beispielrechnungen

20.1 Beispiel 1: Einfamilienhaus

Angenommen, ein Einfamilienhaus hat folgende Eigenschaften:

- Grundfläche: $A_{\text{Boden}} = 100\,\text{m}^2$
- Wandfläche: $A_{\text{Wand}} = 200\,\text{m}^2$
- Dachfläche: $A_{\text{Dach}} = 100\,\text{m}^2$
- Fensterfläche: $A_{\text{Fenster}} = 20\,\text{m}^2$
- Türfläche: $A_{\text{Tür}} = 5\,\text{m}^2$
- Volumen: $V = 400\,\text{m}^3$
- U-Werte: $U_{\text{Wand}} = 0.23\,\text{W/m}^2\text{K}, U_{\text{Dach}} = 0.19$ $\text{W/m}^2\text{K}, U_{\text{Fenster}} = 1.3$ $\text{W/m}^2\text{K}, U_{\text{Tür}} = 1.3\,\text{W/m}^2\text{K}$
- Luftwechselrate: $n = 0.5\,\text{h}^{-1}$
- Innentemperatur: $T_{\text{innen}} = 20\,^\circ\text{C}$
- Außentemperatur: $T_{\text{außen}} = -12\,^\circ\text{C}$ (Winter)

Berechnung der Flächen für die Wände ohne Fenster und Türen:

$$A_{\text{Wand, eff}} = A_{\text{Wand}} - A_{\text{Fenster}} - A_{\text{Tür}} = 200 - 20 - 5 = 175\,\text{m}^2 \tag{9}$$

Berechnung der Wärmeverluste über die einzelnen Bauteile:

- Wärmeverlust über die Wände:

$$\dot{Q}_{\text{Wand}} = U_{\text{Wand}} \cdot A_{\text{Wand, eff}} \cdot \Delta T = 0.23 \cdot 175 \cdot (20 - (-12)) = 835.8\,\text{W} \tag{10}$$

- Wärmeverlust über das Dach:

$$\dot{Q}_{\text{Dach}} = U_{\text{Dach}} \cdot A_{\text{Dach}} \cdot \Delta T = 0.19 \cdot 100 \cdot (20 - (-12)) = 608.4\,\text{W} \tag{11}$$

- Wärmeverlust über die Fenster:

$$\dot{Q}_{\text{Fenster}} = U_{\text{Fenster}} \cdot A_{\text{Fenster}} \cdot \Delta T = 1.3 \cdot 20 \cdot (20 - (-12)) = 832\,\text{W} \tag{12}$$

- Wärmeverlust über die Tür:

$$\dot{Q}_{\text{Tür}} = U_{\text{Tür}} \cdot A_{\text{Tür}} \cdot \Delta T = 1.3 \cdot 5 \cdot (20 - (-12)) = 208\,\text{W} \tag{13}$$

- Wärmeverlust über den Boden:

$$\dot{Q}_{\text{Boden}} = U_{\text{Boden}} \cdot A_{\text{Boden}} \cdot \Delta T = 0.31 \cdot 100 \cdot (20 - (-12)) = 992\,\text{W} \tag{14}$$

Der Gesamtwärmeverlust durch Transmission ist daher:

$$\dot{Q}_{\text{Trans}} = 835.8 + 608.4 + 832 + 208 + 992 = 3476.2\,\text{W} \tag{15}$$

Der Wärmeverlust durch Lüftung ist:

$$\dot{Q}_{\text{Lüftung}} = 0.34 \cdot 0.5 \cdot 400 \cdot (20 - (-12)) = 435.2\,\text{W} \tag{16}$$

Der maximale Heizwärmebedarf ist daher:

$$Q_{\text{max}} = \dot{Q}_{\text{Trans}} + \dot{Q}_{\text{Lüftung}} = 3476.2 + 435.2 = 3911.4\,\text{W} \tag{17}$$

20.2 Beispiel 2: Mehrfamilienhaus

Für ein Mehrfamilienhaus mit einer Grundfläche von $A_{\text{Boden}} = 500\,\text{m}^2$, einer Wandfläche von $A_{\text{Wand}} = 1000\,\text{m}^2$, einer Fensterfläche von $A_{\text{Fenster}} = 100\,\text{m}^2$, einer Türfläche von $A_{\text{Tür}} = 25\,\text{m}^2$, und einem Volumen von $V = 2000\,\text{m}^3$, sowie denselben U-Werten und Temperaturdifferenzen wie im vorherigen Beispiel, beträgt der maximale Heizwärmebedarf:

Berechnung der Flächen für die Wände ohne Fenster und Türen:

$$A_{\text{Wand, eff}} = A_{\text{Wand}} - A_{\text{Fenster}} - A_{\text{Tür}} = 1000 - 100 - 25 = 875\,\text{m}^2 \tag{18}$$

Berechnung der Wärmeverluste über die einzelnen Bauteile:

- Wärmeverlust über die Wände:

$$\dot{Q}_{\text{Wand}} = U_{\text{Wand}} \cdot A_{\text{Wand, eff}} \cdot \Delta T = 0.23 \cdot 875 \cdot (20 - (-12)) = 4179\,\text{W} \tag{19}$$

- Wärmeverlust über das Dach:

$$\dot{Q}_{\text{Dach}} = U_{\text{Dach}} \cdot A_{\text{Dach}} \cdot \Delta T = 0.19 \cdot 500 \cdot (20 - (-12)) = 3042\,\text{W} \tag{20}$$

- Wärmeverlust über die Fenster:

$$\dot{Q}_{\text{Fenster}} = U_{\text{Fenster}} \cdot A_{\text{Fenster}} \cdot \Delta T = 1.3 \cdot 100 \cdot (20 - (-12)) = 4160\,\text{W} \tag{21}$$

- Wärmeverlust über die Türen:

$$\dot{Q}_{\text{Tür}} = U_{\text{Tür}} \cdot A_{\text{Tür}} \cdot \Delta T = 1.3 \cdot 25 \cdot (20 - (-12)) = 1040\,\text{W} \tag{22}$$

- Wärmeverlust über den Boden:

$$\dot{Q}_{\text{Boden}} = U_{\text{Boden}} \cdot A_{\text{Boden}} \cdot \Delta T = 0.31 \cdot 500 \cdot (20 - (-12)) = 4960\,\text{W} \tag{23}$$

Der Gesamtwärmeverlust durch Transmission ist daher:

$$\dot{Q}_{\text{Trans}} = 4179 + 3042 + 4160 + 1040 + 4960 = 17381\,\text{W} \tag{24}$$

Der Wärmeverlust durch Lüftung ist:

$$\dot{Q}_{\text{Lüftung}} = 0.34 \cdot 0.5 \cdot 2000 \cdot (20 - (-12)) = 2176\,\text{W} \tag{25}$$

Der maximale Heizwärmebedarf ist daher:

$$Q_{\text{max}} = \dot{Q}_{\text{Trans}} + \dot{Q}_{\text{Lüftung}} = 17381 + 2176 = 19557\,\text{W} \tag{26}$$

21 Einleitung

Dieses Dokument beschreibt die Berechnungsmethode, die in der Datei `SanierungsanalysefuerGUI.py` implementiert ist. Die Berechnungen umfassen sowohl die Ermittlung des Wärmebedarfs eines Gebäudes vor und nach einer Sanierung als auch eine wirtschaftliche Analyse der Sanierung, einschließlich Amortisationszeit, Net Present Value (NPV), und Return on Investment (ROI).

22 Modellierung des Gebäudes und Berechnung des Wärmebedarfs

Die `Building`-Klasse repräsentiert ein Gebäude und berechnet den maximalen Heizwärmebedarf, den jährlichen Heizwärmebedarf sowie den jährlichen Warmwasserbedarf. Diese Berechnungen basieren auf den thermischen Eigenschaften des Gebäudes und den klimatischen Bedingungen (aus TRY-Daten).

22.1 U-Werte und Wärmeverluste

Die Wärmedurchgangskoeffizienten (U-Werte) beschreiben die Wärmeverluste durch die verschiedenen Bauteile des Gebäudes (z.B. Wände, Dach, Fenster, Türen, Boden). Der Wärmeverlust pro Kelvin Temperaturdifferenz wird berechnet als:

$$\text{Wärmeverlust}_{\text{Teil}} = A_{\text{Teil}} \times U_{\text{Teil}}$$

wobei A_{Teil} die Fläche des jeweiligen Bauteils und U_{Teil} der U-Wert des Bauteils ist.

Der gesamte Wärmeverlust pro Kelvin des Gebäudes ergibt sich als Summe der Wärmeverluste aller Bauteile:

$$Q_{\text{Verlust}} = Q_{\text{Wand}} + Q_{\text{Boden}} + Q_{\text{Dach}} + Q_{\text{Fenster}} + Q_{\text{Tür}}$$

Es handelt sich damit um eine vereinfachte Berechnung, die die Wärmebrücken und die thermische Speicherkapazität des Gebäudes nicht berücksichtigt.

22.2 Maximale Heizlast und Temperaturdifferenz

Die maximale Heizlast wird basierend auf der Normaußentemperatur $T_{\text{außen}}$ und der gewünschten Raumtemperatur T_{innen} berechnet. Die maximale Temperaturdifferenz ist:

$$\Delta T_{\text{max}} = T_{\text{innen}} - T_{\text{außen}}$$

Die maximale Heizlast des Gebäudes ergibt sich aus den Wärmeverlusten pro Kelvin multipliziert mit der maximalen Temperaturdifferenz:

$$Q_{\text{max}} = Q_{\text{Verlust}} \times \Delta T_{\text{max}}$$

22.3 Berechnung des Jahresheizbedarfs

Der jährliche Heizwärmebedarf wird basierend auf den stündlichen Außentemperaturen berechnet. Die Heizlast für jede Stunde wird berechnet als:

$$Q_{\text{Heizung, stunde}} = \max(m \times T_{\text{außen}} + b, 0)$$

wobei m und b durch lineare Regression bestimmt werden. Die Summe der Heizlasten aller Stunden ergibt den jährlichen Heizwärmebedarf.

22.4 Warmwasserbedarf

Der jährliche Warmwasserbedarf wird als Konstante pro Quadratmeter Gebäudefläche und pro Stockwerk berechnet:

$$Q_{\text{WW}} = \text{WW\_Bedarf\_pro\_m2} \times A_{\text{Boden}} \times \text{Stockwerke}$$

23 Wirtschaftliche Analyse

Die Klasse `SanierungsAnalyse` führt eine wirtschaftliche Analyse der Sanierung durch. Es werden verschiedene Indikatoren wie Amortisationszeit, Net Present Value (NPV) und Return on Investment (ROI) berechnet.

23.1 Kosteneinsparungen durch Sanierung

Die jährlichen Energiekosteneinsparungen durch die Sanierung werden berechnet als Differenz zwischen den Energiekosten vor und nach der Sanierung:

$$\Delta\text{Kosten} = Q_{\text{ref}} \times P_{\text{ref}} - Q_{\text{san}} \times P_{\text{san}}$$

wobei Q_{ref} und Q_{san} die Wärmebedarfe vor und nach der Sanierung sind, und P_{ref} und P_{san} die Energiepreise vor und nach der Sanierung.

23.2 Amortisationszeit

Die Amortisationszeit gibt an, nach wie vielen Jahren die Investitionskosten durch die jährlichen Kosteneinsparungen gedeckt sind. Sie wird berechnet als:

$$\text{Amortisationszeit} = \frac{\text{Investitionskosten}}{\Delta\text{Kosten}}$$

23.3 Net Present Value (NPV)

Der NPV gibt den heutigen Wert zukünftiger Cashflows (Kosteneinsparungen) an, abzüglich der Investitionskosten. Er wird mit folgender Formel berechnet:

$$\text{NPV} = \sum_{t=1}^{T} \frac{\Delta\text{Kosten}}{(1+r)^t} - \text{Investitionskosten}$$

wobei r der Diskontsatz und T die Anzahl der Jahre ist.

23.4 Return on Investment (ROI)

Der ROI gibt das Verhältnis zwischen den Kosteneinsparungen und den Investitionskosten an:

$$\text{ROI} = \frac{\Delta\text{Kosten} \times T - \text{Investitionskosten}}{\text{Investitionskosten}}$$

24 Lebenszykluskostenanalyse (LCCA)

Die Lebenszykluskostenanalyse berücksichtigt alle Kosten über die gesamte Lebensdauer der Sanierung, einschließlich Investitions-, Betriebs- und Instandhaltungskosten sowie den Restwert:

$$\text{LCCA} = \text{NPV der Cashflows} + \text{Restwert}$$

25 Zusammenfassung

Die `Sanierungsanalyse`-Berechnungsmethode ermöglicht es, den energetischen Zustand eines Gebäudes zu analysieren und die Kosten-Nutzen-Relation einer Sanierung zu bewerten. Die Berechnungen umfassen den Heiz- und Warmwasserbedarf des Gebäudes sowie wirtschaftliche Kennzahlen wie Amortisationszeit, NPV und ROI. Diese Methoden bieten eine fundierte Grundlage für Entscheidungen im Rahmen von Sanierungsprojekten.

26 Einleitung

Diese Dokumentation beschreibt die Funktionsweise der Python-Skripte, die zur thermischen und hydraulischen Netzwerksimulation sowie zur Initialisierung von Netzwerken auf Basis von GeoJSON-Daten verwendet werden. Zu den beschriebenen Skripten gehören:

- `pp_net_initialisation_geojson.py`
- `pp_net_time_series_simulation.py`

- `utilities.py`
- `config_plot.py`
- `controllers.py`

Die Skripte bieten Funktionen für die Initialisierung von Netzwerken, Zeitreihensimulationen, Datenvorverarbeitung und Netzwerkoptimierung auf Grundlage von GeoJSON-Daten.

27 Skript: `pp_net_initialisation_geojson.py`

27.1 Übersicht

Dieses Skript dient der Initialisierung eines Netzwerks basierend auf GeoJSON-Daten. Es lädt Daten aus GeoJSON-Dateien, die Informationen zu Wärmetauschern, Rohrleitungen und anderen Netzwerkelementen enthalten, und erstellt ein simuliertes Netzwerk. Dieses Netzwerk kann dann für thermische und hydraulische Berechnungen genutzt werden.

27.2 Wichtige Funktionen

27.2.1 Funktion: `initialize_geojson(vorlauf, ruecklauf, hast, erzeugeranlagen, ...)`

Beschreibung:
Diese Funktion lädt GeoJSON-Daten, die Informationen über Vorlauf- und Rücklaufleitungen, Wärmetauscher (HAST) und Erzeugeranlagen enthalten, und initialisiert damit ein Netzwerksimulationsmodell. Sie erstellt ein Netzwerk basierend auf den übergebenen GeoJSON-Daten und setzt Parameter wie Rohrleitungsdurchmesser und Wärmeflüsse.

Parameter:

- `vorlauf`: GeoJSON-Daten der Vorlaufleitungen.
- `ruecklauf`: GeoJSON-Daten der Rücklaufleitungen.
- `hast`: GeoJSON-Daten der Wärmetauscher.
- `erzeugeranlagen`: GeoJSON-Daten der Erzeugeranlagen.

27.2.2 Funktion: `create_network(gdf_flow_line, gdf_return_line, ...)`

Beschreibung:
Erstellt das `pandapipes`-Netzwerk basierend auf den übergebenen GeoJSON-Daten der Vor- und Rücklaufleitungen sowie den Wärmetauschern und Erzeugern. Dabei werden die Leitungen im Netz korrekt miteinander verbunden und Junction-Punkte gesetzt.

Parameter:

- `gdf_flow_line`: GeoDataFrame der Vorlaufleitungen.
- `gdf_return_line`: GeoDataFrame der Rücklaufleitungen.
- `gdf_hast`: GeoDataFrame der Wärmetauscher.

27.2.3 Funktion: `create_pipes(net, all_line_coords, all_line_lengths, ...)`

Beschreibung:
Diese Funktion fügt dem Netzwerk Rohre basierend auf den Koordinaten und Längen der Liniensegmente hinzu, die aus den GeoJSON-Daten extrahiert wurden.

Parameter:

- `net`: Das `pandapipes`-Netzwerkobjekt.
- `all_line_coords`: Liste der Rohrkoordinaten.
- `all_line_lengths`: Längen der Rohre.

28 Skript: `pp_net_time_series_simulation.py`

28.1 Übersicht

Das Skript `pp_net_time_series_simulation.py` führt Zeitreihenberechnungen für thermische und hydraulische Netzwerke durch. Es enthält Funktionen zur Aktualisierung von Steuerungen (z. B. für Rücklauf- und Vorlauftemperaturen) und zur Durchführung von Simulationen für Netzwerke über eine festgelegte Zeitperiode.

28.2 Wichtige Funktionen

28.2.1 Funktion: `update_const_controls(net, qext_w_profiles, time_steps, start, end)`

Beschreibung:
Aktualisiert konstante Steuerungen im Netzwerk mit neuen Daten für die Zeitreihenberechnung. Dies ist besonders nützlich, wenn sich die externen Wärmeprofile im Laufe der Zeit ändern.

28.2.2 Funktion: `update_return_temperature_controller(net, supply_temperature_heat_con return_temperature_heat_consumer, time_steps, start, end)`

Beschreibung:
Aktualisiert die Steuerung der Rücklauftemperatur für Wärmekonsumenten. Die Funktion ermöglicht es, die Rücklauftemperatur in Abhängigkeit von der Zeit und der Systemlast dynamisch zu steuern.

28.2.3 Funktion: `thermohydraulic_time_series_net()`

Beschreibung:
Führt eine thermohydraulische Zeitreihensimulation für das gesamte Netzwerk durch. Diese Simulation berücksichtigt die Änderungen in der Vorlauf- und Rücklauftemperatur sowie den Druck im Netzwerk.

29 Skript: utilities.py

29.1 Übersicht

Das utilities.py-Skript enthält Hilfsfunktionen, die in verschiedenen Netzwerksimulationen und Optimierungsalgorithmen verwendet werden. Dazu gehören Funktionen zur Berechnung der Leistung von Wärmepumpen (Coefficient of Performance, COP) und zur Netzwerkoptimierung.

29.2 Wichtige Funktionen

29.2.1 Funktion: COP_WP(VLT_L, QT, values)

Beschreibung:
Berechnet den COP (Coefficient of Performance) einer Wärmepumpe basierend auf der Vorlauftemperatur VLT_L und der Quelle QT. Diese Funktion wird in der Simulation verwendet, um die Effizienz von Wärmepumpen zu bestimmen.

29.2.2 Funktion: net_optimization(net, v_max_pipe, v_max_heat_exchanger, ...)

Beschreibung:
Optimiert das Netzwerk durch Anpassung der Rohrdurchmesser und der Wärmetauscher, um die maximalen Geschwindigkeiten in den Rohren und Wärmetauschern einzuhalten. Diese Funktion ist entscheidend, um sicherzustellen, dass das Netzwerk effizient arbeitet, ohne die technischen Grenzwerte zu überschreiten.

30 Skript: config_plot.py

30.1 Übersicht

Das config_plot.py-Skript enthält Konfigurations- und Visualisierungsoptionen für Netzwerksimulationen. Es bietet Funktionen zur grafischen Darstellung von Netzwerken und Simulationsergebnissen.

30.2 Wichtige Funktionen

30.2.1 Funktion: plot_network(net, output_file)

Beschreibung:
Erstellt eine grafische Darstellung des Netzwerks und speichert das Ergebnis in einer Datei. Diese Funktion ist hilfreich, um die Struktur des Netzwerks und die Ergebnisse der Simulation visuell zu überprüfen.

30.3 Skript: controllers.py

30.4 Übersicht

Das controllers.py-Skript definiert verschiedene Steuerungsmechanismen für die Netzwerksimulation. Dazu gehören Controller für Temperaturen und Durchflüsse, die in der Simulation verwendet werden, um realistische Systemverhalten zu modellieren.

30.5 Wichtige Funktionen

30.5.1 Funktion: `TemperatureController(...)`

Beschreibung:
Dieser Controller steuert die Temperatur an bestimmten Punkten im Netzwerk und ermöglicht es, Temperaturprofile zu definieren, die über die Zeit dynamisch angepasst werden können.

31 Fazit

Die beschriebenen Skripte bieten eine umfassende Lösung für die Initialisierung, Simulation und Optimierung thermischer und hydraulischer Netzwerke. Mit Hilfe der GeoJSON-basierten Netzwerkinitialisierung und der Zeitreihensimulation können realistische Szenarien für Wärmeverteilnetze modelliert und analysiert werden. Die Skripte ermöglichen eine detaillierte Analyse von Lastprofilen, Druckverlusten und Temperaturverteilungen im Netz.

32 Wirtschaftlichkeitsrechnung nach VDI 2067

32.1 Einleitung

Die Berechnung der Wirtschaftlichkeit technischer Anlagen ist ein zentraler Bestandteil des Energiemanagements. Die Annuitätsmethode gemäß VDI 2067 ermöglicht es, die Gesamtkosten einer technischen Anlage über die gesamte Nutzungsdauer zu erfassen und zu bewerten. Die Kosten umfassen die kapitalgebundenen, bedarfsgebundenen und betriebsgebundenen Kosten, sowie Erlöse.

32.2 Die Annuität

Die Annuität bezeichnet eine jährliche Zahlung, die Kapital- und Betriebskosten sowie Wartungskosten und gegebenenfalls Erlöse berücksichtigt. Die Berechnung der Annuität basiert auf den folgenden Komponenten:

32.2.1 Formel zur Berechnung der Annuität

Die Annuität A_N wird durch die Summe der folgenden Komponenten bestimmt:

$$A_N = A_{N,K} + A_{N,V} + A_{N,B} + A_{N,S} - A_{N,E}$$

wobei:

- $A_{N,K}$: Kapitalgebundene Kosten
- $A_{N,V}$: Bedarfsgebundene Kosten
- $A_{N,B}$: Betriebsgebundene Kosten
- $A_{N,S}$: Sonstige Kosten
- $A_{N,E}$: Erlöse

32.2.2 Kapitalgebundene Kosten

Die kapitalgebundenen Kosten $A_{N,K}$ umfassen die Investitionskosten und den Restwert der Anlage:

$$A_{N,K} = (A_0 - R_W) \cdot a$$

wobei:

- A_0 die Anfangsinvestition ist,
- R_W der Restwert der Anlage nach Ablauf der Nutzungsdauer T ist,
- a der Annuitätsfaktor ist:
$$a = \frac{q-1}{1-q^{-T}}$$
- q der Zinsfaktor ist, also $q = 1 + \text{Zinssatz}$.

32.2.3 Bedarfsgebundene Kosten

Die bedarfsgebundenen Kosten $A_{N,V}$ werden aus dem Energiebedarf und den Energiekosten berechnet:

$$A_{N,V} = \text{Energiebedarf} \cdot \text{Energiekosten} \cdot a \cdot b_V$$

wobei:

$$b_V = \frac{1 - \left(\frac{r}{q}\right)^T}{q - r}$$

und r der Preissteigerungsfaktor (Inflation) ist.

32.2.4 Betriebsgebundene Kosten

Die betriebsgebundenen Kosten $A_{N,B}$ setzen sich aus den Betriebskosten und den Wartungskosten zusammen:

$$A_{N,B} = (\text{Bedienaufwand} \cdot \text{Stundensatz} + A_0 \cdot (f_{\text{Inst}} + f_{\text{W\_Insp}})/100) \cdot a \cdot b_B$$

wobei:

- f_{Inst}: Installationsfaktor,
- $f_{\text{W\_Insp}}$: Wartungs- und Inspektionsfaktor.

32.2.5 Sonstige Kosten

Sonstige Kosten $A_{N,S}$ können in ähnlicher Weise berechnet werden, wobei keine weiteren Parameter in diesem Beispiel angegeben sind.

32.2.6 Erlöse

Falls Erlöse $A_{N,E}$ vorhanden sind (z.B. durch den Verkauf von Energie), werden diese von der Annuität abgezogen:

$$A_{N,E} = E_1 \cdot a \cdot b_E$$

32.2.7 Rückgabewert

Die Gesamtannuität wird als Summe der Komponenten berechnet. Sie ergibt die jährlichen Gesamtkosten oder Erträge der Anlage:

$$A_N = -(A_{N,K} + A_{N,V} + A_{N,B} + A_{N,S} - A_{N,E})$$

32.3 Zusammenfassung

Die Annuitätsberechnung gemäß VDI 2067 bietet eine umfassende Methode, um die Kosten und Erlöse einer technischen Anlage über die gesamte Nutzungsdauer zu bewerten. Durch die Anwendung von Kapitalwertfaktoren und Preissteigerungsfaktoren können die jährlichen Belastungen und Einsparungen realitätsnah abgebildet werden.

33 HeatPump Klasse

Die `HeatPump`-Klasse repräsentiert ein Wärmepumpensystem und bietet Methoden zur Berechnung verschiedener Leistungs- und Wirtschaftlichkeitskennzahlen. Die Klasse ist modular aufgebaut und ermöglicht die Anpassung an unterschiedliche Wärmequellen und Anwendungsfälle. Nachfolgend werden die wichtigsten Attribute und Methoden der Klasse detailliert beschrieben.

33.1 Attribute

- `name (str)`: Der Name der Wärmepumpe.
- `spezifische_Investitionskosten_WP (float)`: Spezifische Investitionskosten der Wärmepumpe pro kW. Standardwert: 1000 €/kW.
- `Nutzungsdauer_WP (int)`: Nutzungsdauer der Wärmepumpe in Jahren. Standardwert: 20 Jahre.
- `f_Inst_WP (float)`: Installationsfaktor für die Wärmepumpe. Standardwert: 1.
- `f_W_Insp_WP (float)`: Wartungs- und Inspektionsfaktor für die Wärmepumpe. Standardwert: 1.5.
- `Bedienaufwand_WP (float)`: Betriebsaufwand für die Wärmepumpe in Stunden. Standardwert: 0.
- `f_Inst_WQ (float)`: Installationsfaktor für die Wärmequelle. Standardwert: 0.5.
- `f_W_Insp_WQ (float)`: Wartungs- und Inspektionsfaktor für die Wärmequelle. Standardwert: 0.5.
- `Bedienaufwand_WQ (float)`: Betriebsaufwand für die Wärmequelle in Stunden. Standardwert: 0.
- `Nutzungsdauer_WQ_dict (dict)`: Wörterbuch, das die Nutzungsdauer verschiedener Wärmequellen (z.B. Abwärme, Flusswasser) enthält.
- `co2_factor_electricity (float)`: CO_2-Emissionsfaktor für Strom in tCO_2/MWh. Standardwert: 2.4 tCO_2/MWh.

33.2 Methoden

- `calculate_COP(VLT_L, QT, COP_data)`: Berechnet die Leistungszahl (COP) der Wärmepumpe, indem die COP-Daten basierend auf Vorlauftemperaturen (`VLT_L`) und Quellentemperaturen (`QT`) interpoliert werden.

  Diese Methode verwendet eine zweidimensionale Interpolation basierend auf vorgegebenen Vorlauf- und Quellentemperaturen. Der COP wird für jede Kombination von Vorlauftemperatur und Quellentemperatur bestimmt. Der Interpolationsalgorithmus verwendet Gitterdaten aus einer Datei oder einem Dataset, in dem die Kennlinien der Wärmepumpe enthalten sind:

  $$COP = f(\texttt{VLT\_L}, \texttt{QT})$$

  Wo f die Interpolationsfunktion ist, die die Kennlinien der Wärmepumpe verwendet, um den entsprechenden COP zu berechnen.

- `calculate_heat_generation_costs(Wärmeleistung, Wärmemenge, Strombedarf, spez_Investitionskosten_WQ, Strompreis, q, r, T, BEW, stundensatz)`: Berechnet die gewichteten Durchschnittskosten der Wärmeerzeugung (WGK) der Wärmepumpe auf Basis der thermischen Leistung, der Investitionskosten und der Betriebskosten.

  - **Wärmeleistung (float)**: Erzeugte Wärmeleistung in kW.
  - **Wärmemenge (float)**: Gesamte Wärmemenge, die von der Wärmepumpe produziert wurde, in MWh.
  - **Strombedarf (float)**: Strombedarf der Wärmepumpe in MWh.
  - **spez_Investitionskosten_WQ (float)**: Spezifische Investitionskosten für die Wärmequelle.
  - **Strompreis (float)**: Strompreis in €/MWh.
  - **q (float)**: Kapitalrückgewinnungsfaktor.
  - **r (float)**: Preissteigerungsfaktor.
  - **T (int)**: Betrachtungszeitraum in Jahren.
  - **BEW (float)**: Abzinsungsfaktor.
  - **stundensatz (float)**: Arbeitskosten pro Stunde in €/Stunde.

  Diese Methode berechnet die Gesamtkosten der Wärmeerzeugung, indem die Investitionskosten der Wärmepumpe und der Wärmequelle über die Lebensdauer des Systems mit den Betriebskosten kombiniert werden. Die jährlichen Kosten werden mit dem Annuitätenfaktor berechnet:

  $$E1\_WP = \frac{\texttt{Investitionskosten\_WP} + \texttt{Betriebskosten}}{\texttt{Wärmemenge}}$$

  und die spezifischen Wärmeerzeugungskosten (`WGK`) werden folgendermaßen berechnet:

  $$\texttt{WGK\_Gesamt} = \frac{\texttt{E1\_WP} + \texttt{E1\_WQ}}{\texttt{Wärmemenge}}$$

33.3 Nutzung der Methoden

Beispiel zur Berechnung des COP und der Wärmeerzeugungskosten (WGK):

```
heat_pump = HeatPump(name="Luft-Wärmepumpe", spezifische_Investitionskosten_WP=1200)

# COP-Berechnung
VLT_L = np.array([40, 50, 60])
QT = 10
COP_data = np.array([[0, 35, 45, 55, 65],
                     [5, 3.6, 3.4, 3.2, 3.0, 2.8],
                     [10, 4.0, 3.8, 3.6, 3.4, 3.2],
                     [15, 4.4, 4.2, 4.0, 3.8, 3.6]])

COP_L, adjusted_VLT_L = heat_pump.COP_WP(VLT_L, QT, COP_data)

# WGK-Berechnung
Wärmeleistung = 50  # kW
Wärmemenge = 120  # MWh
Strombedarf = 40  # MWh
Strompreis = 80  # €/MWh
q = 0.03
r = 0.02
T = 20
BEW = 0.95
stundensatz = 50
spez_Investitionskosten_WQ = 600

WGK_Gesamt_a = heat_pump.WGK(Wärmeleistung, Wärmemenge, Strombedarf, spez_Investition
```

In diesem Beispiel wird der COP der Wärmepumpe auf Basis der Vorlauf- und Quellentemperaturen berechnet. Anschließend werden die gewichteten Durchschnittskosten der Wärmeerzeugung (WGK) unter Berücksichtigung von Investitions- und Betriebskosten der Wärmepumpe und der Wärmequelle berechnet.

34 Geothermal Klasse

Die `Geothermal`-Klasse modelliert ein geothermisches Wärmepumpensystem und erbt von der `HeatPump`-Basis-Klasse. Sie enthält Methoden zur Simulation des geothermischen Wärmeentzugsprozesses und zur Berechnung verschiedener ökonomischer und ökologischer Kennzahlen.

34.1 Attribute

- `Fläche (float)`: Verfügbare Fläche für die geothermische Installation in Quadratmetern.
- `Bohrtiefe (float)`: Bohrtiefe der geothermischen Sonden in Metern.

- `Temperatur_Geothermie (float)`: Temperatur der geothermischen Quelle in Grad Celsius.
- `spez_Bohrkosten (float)`: Spezifische Bohrkosten pro Meter. Standardwert: 100 €/m.
- `spez_Entzugsleistung (float)`: Spezifische Entzugsleistung pro Meter. Standardwert: 50 W/m.
- `Vollbenutzungsstunden (float)`: Vollbenutzungsstunden pro Jahr. Standardwert: 2400 Stunden.
- `Abstand_Sonden (float)`: Abstand zwischen den Sonden in Metern. Standardwert: 10 m.
- `min_Teillast (float)`: Minimale Teillast als Anteil der Nennlast. Standardwert: 0,2.
- `co2_factor_electricity (float)`: CO_2-Emissionsfaktor für Stromverbrauch, in tCO_2/MWh. Standardwert: 0,4 tCO_2/MWh.
- `primärenergiefaktor (float)`: Primärenergiefaktor für den Stromverbrauch. Standardwert: 2,4.

34.2 Methoden

- `calculate_operation(Last_L, VLT_L, COP_data, duration)`: Simuliert den geothermischen Wärmeentzugsprozess und berechnet die erzeugte Wärmemenge, den Strombedarf und weitere Leistungskennzahlen.
  - **Last_L (array-like)**: Lastprofil in kW.
  - **VLT_L (array-like)**: Vorlauftemperaturen in Grad Celsius.
  - **COP_data (array-like)**: Daten zur Leistungszahl (COP) zur Interpolation.
  - **duration (float)**: Dauer des Zeitschritts in Stunden.

  Diese Methode berechnet den geothermischen Wärmeertrag auf Basis der Quelltemperatur und der spezifischen Entzugsleistung pro Meter. Die Entzugsleistung wird als:

  $$\texttt{Entzugsleistung} = \texttt{Bohrtiefe} \times \texttt{spez\_Entzugsleistung} \times \texttt{Anzahl\_Sonden}$$

  berechnet, wobei die Anzahl der Sonden von der verfügbaren Fläche und dem Abstand der Sonden abhängt.
- `calculate(VLT_L, COP_data, Strompreis, q, r, T, BEW, stundensatz, duration, general_results)`: Berechnet die ökonomischen und ökologischen Kennzahlen für das geothermische Wärmepumpensystem.
  - **VLT_L (array-like)**: Vorlauftemperaturen in Grad Celsius.
  - **COP_data (array-like)**: COP-Daten zur Leistungsberechnung.
  - **Strompreis (float)**: Strompreis in €/MWh.

- **q (float)**: Kapitalrückgewinnungsfaktor.
- **r (float)**: Preissteigerungsfaktor.
- **T (int)**: Betrachtungszeitraum in Jahren.
- **BEW (float)**: Abzinsungsfaktor für Betriebskosten.
- **stundensatz (float)**: Stundensatz in €/Stunde.
- **duration (float)**: Dauer jedes Simulationsschritts in Stunden.
- **general_results (dict)**: Allgemeine Ergebnisse, inklusive Lastprofil.

Diese Methode berechnet die gewichteten Durchschnittskosten der Wärmeerzeugung (WGK) und die CO_2-Emissionen basierend auf dem Stromverbrauch. Die spezifischen CO_2-Emissionen werden wie folgt berechnet:

$$\texttt{spec\_co2\_total} = \frac{\texttt{co2\_emissions}}{\texttt{Wärmemenge\_Geothermie}}\,\mathrm{tCO_2/MWh}$$

- `to_dict()`: Wandelt die Objektattribute in ein Wörterbuch um.
- `from_dict(data)`: Erstellt ein Objekt aus einem Wörterbuch von Attributen.

34.3 Ökonomische und ökologische Überlegungen

Die `Geothermal`-Klasse berechnet die Wärmegestehungskosten (WGK), welche die Kosten für Bohrung, Installation, Betrieb und Stromverbrauch berücksichtigen. Sie berechnet außerdem die spezifischen CO_2-Emissionen basierend auf dem Stromverbrauch sowie den Primärenergieverbrauch unter Verwendung eines Primärenergiefaktors.

34.4 Nutzungsbeispiel

Das folgende Beispiel zeigt, wie die `Geothermal`-Klasse initialisiert und verwendet wird, um die Leistung eines geothermischen Systems zu berechnen:

```
geothermal_system = Geothermal(
    name="Geothermal Heat Pump",
    Fläche=500,  # m²
    Bohrtiefe=150,  # m
    Temperatur_Geothermie=10,  # °C
    spez_Bohrkosten=120,  # €/m
    spez_Entzugsleistung=55  # W/m
)
results = geothermal_system.calculate(
    VLT_L=temperature_profile,
    COP_data=cop_profile,
    Strompreis=100,  # €/MWh
    q=0.04, r=0.02, T=20,
    BEW=0.9,
    stundensatz=50,
    duration=1,
    general_results=load_data
)
```

In diesem Beispiel wird ein geothermisches System mit einer Fläche von 500 m^2 und einer Bohrtiefe von 150 m simuliert. Die Leistung und die ökonomischen Kennzahlen des Systems werden anhand der eingegebenen Daten berechnet.

35 WasteHeatPump Klasse

Die `WasteHeatPump`-Klasse modelliert ein Wärmepumpensystem zur Rückgewinnung von Abwärme und erbt von der `HeatPump`-Basisklasse. Sie enthält Methoden zur Simulation der Leistung der Wärmepumpe sowie zur Berechnung verschiedener ökonomischer und ökologischer Kennzahlen auf Basis der Abwärmenutzung.

35.1 Attribute

- `Kühlleistung_Abwärme (float)`: Kühlleistung der Abwärmepumpe in kW.
- `Temperatur_Abwärme (float)`: Temperatur der Abwärmequelle in Grad Celsius.
- `spez_Investitionskosten_Abwärme (float)`: Spezifische Investitionskosten der Abwärmepumpe pro kW. Standardwert: 500 €/kW.
- `spezifische_Investitionskosten_WP (float)`: Spezifische Investitionskosten der Wärmepumpe pro kW. Standardwert: 1000 €/kW.
- `min_Teillast (float)`: Minimale Teillast als Anteil der Nennlast. Standardwert: 0,2.
- `co2_factor_electricity (float)`: CO_2-Faktor für den Stromverbrauch in tCO_2/MWh. Standardwert: 0,4 tCO_2/MWh.
- `primärenergiefaktor (float)`: Primärenergiefaktor für den Stromverbrauch. Standardwert: 2,4.

35.2 Methoden

- `calculate_heat_pump(VLT_L, COP_data)`: Berechnet die Wärmelast, den Stromverbrauch und die angepassten Vorlauftemperaturen für die Abwärmepumpe.
  - **VLT_L (array-like)**: Vorlauftemperaturen in Grad Celsius.
  - **COP_data (array-like)**: COP-Daten zur Interpolation.

  Gibt die Wärmelast und den Stromverbrauch für die Abwärmepumpe zurück.
- `calculate_waste_heat(Last_L, VLT_L, COP_data, duration)`: Berechnet die Abwärme und weitere Leistungskennzahlen für die Wärmepumpe.
  - **Last_L (array-like)**: Lastanforderung in kW.
  - **VLT_L (array-like)**: Vorlauftemperaturen in Grad Celsius.
  - **COP_data (array-like)**: COP-Daten zur Leistungsberechnung.
  - **duration (float)**: Dauer des Zeitschritts in Stunden.

Gibt die erzeugte Wärmemenge, den Strombedarf, die Wärmeleistung und die elektrische Leistung zurück.

- `calculate(VLT_L, COP_data, Strompreis, q, r, T, BEW, stundensatz, duration, general_results)`: Berechnet die ökonomischen und ökologischen Kennzahlen für die Abwärmepumpe.
  - **VLT_L (array-like)**: Vorlauftemperaturen in Grad Celsius.
  - **COP_data (array-like)**: COP-Daten zur Leistungsberechnung.
  - **Strompreis (float)**: Strompreis in €/MWh.
  - **q (float)**: Kapitalrückgewinnungsfaktor.
  - **r (float)**: Preissteigerungsfaktor.
  - **T (int)**: Betrachtungszeitraum in Jahren.
  - **BEW (float)**: Abzinsungsfaktor für Betriebskosten.
  - **stundensatz (float)**: Arbeitskosten pro Stunde in €/Stunde.
  - **duration (float)**: Dauer jedes Simulationsschritts in Stunden.
  - **general_results (dict)**: Allgemeine Ergebnisse, inklusive Lastprofil.

  Gibt ein Wörterbuch mit den berechneten Kennzahlen, einschließlich Wärmemenge, Strombedarf, CO_2-Emissionen und Primärenergieverbrauch, zurück.
- `to_dict()`: Wandelt die Objektattribute in ein Wörterbuch um.
- `from_dict(data)`: Erstellt ein Objekt aus einem Wörterbuch von Attributen.

35.3 Ökonomische und ökologische Überlegungen

Die `WasteHeatPump`-Klasse berechnet die Wärmegestehungskosten (WGK) für die Abwärmepumpe, die die Installations-, Betriebs- und Stromkosten berücksichtigen. Die Klasse berechnet auch die spezifischen CO_2-Emissionen basierend auf dem Stromverbrauch sowie den Primärenergieverbrauch des Systems.

35.4 Nutzungsbeispiel

Das folgende Beispiel zeigt, wie die `WasteHeatPump`-Klasse initialisiert und verwendet wird, um die Leistung eines Abwärmenutzungssystems zu berechnen:

```
waste_heat_pump = WasteHeatPump(
    name="Waste Heat Pump System",
    Kühlleistung_Abwärme=100,  # kW
    Temperatur_Abwärme=60  # °C
)
results = waste_heat_pump.calculate(
    VLT_L=temperature_profile,
    COP_data=cop_profile,
    Strompreis=150,  # €/MWh
    q=0.05, r=0.02, T=20,
```

```
    BEW=0.85,
    stundensatz=45,
    duration=1,
    general_results=load_profile
)
```

In diesem Beispiel wird ein Abwärmenutzungssystem mit einer Kühlleistung von 100 kW und einer Abwärmequellentemperatur von 60°C erstellt. Die Leistungskennzahlen und wirtschaftlichen Bewertungen werden basierend auf den Eingabedaten berechnet.

36 RiverHeatPump Klasse

Die `RiverHeatPump`-Klasse modelliert ein Wärmepumpensystem, das Flusswasser als Wärmequelle nutzt, und erbt von der `HeatPump`-Basisklasse. Sie enthält Methoden zur Berechnung der Leistung der Wärmepumpe sowie zur Ermittlung wirtschaftlicher und ökologischer Kennzahlen.

36.1 Attribute

- `Wärmeleistung_FW_WP (float)`: Wärmeleistung der Flusswasser-Wärmepumpe in kW.
- `Temperatur_FW_WP (float)`: Temperatur des Flusswassers in Grad Celsius.
- `dT (float)`: Temperaturdifferenz für den Betrieb. Standardwert: 0.
- `spez_Investitionskosten_Flusswasser (float)`: Spezifische Investitionskosten der Flusswasser-Wärmepumpe in €/kW. Standardwert: 1000 €/kW.
- `spezifische_Investitionskosten_WP (float)`: Spezifische Investitionskosten der Wärmepumpe in €/kW. Standardwert: 1000 €/kW.
- `min_Teillast (float)`: Minimale Teillast als Bruchteil der Nennlast. Standardwert: 0,2.
- `co2_factor_electricity (float)`: CO_2-Faktor für den Stromverbrauch in tCO_2/MWh. Standardwert: 0,4.
- `primärenergiefaktor (float)`: Primärenergiefaktor für den Stromverbrauch. Standardwert: 2,4.

36.2 Methoden

- `calculate_heat_pump(Wärmeleistung_L, VLT_L, COP_data)`: Berechnet die Kühlleistung, den Stromverbrauch und die angepassten Vorlauftemperaturen.
  - **Wärmeleistung_L (array-like)**: Wärmeleistungsprofil.
  - **VLT_L (array-like)**: Vorlauftemperaturen.
  - **COP_data (array-like)**: COP-Daten zur Interpolation.

Gibt die Kühlleistung, den Stromverbrauch und die angepassten Vorlauftemperaturen zurück.

- `calculate_river_heat(Last_L, VLT_L, COP_data, duration)`: Berechnet die Abwärme und weitere Leistungskennzahlen für die Flusswasser-Wärmepumpe.
  - **Last_L (array-like)**: Lastanforderung in kW.
  - **VLT_L (array-like)**: Vorlauftemperaturen.
  - **COP_data (array-like)**: COP-Daten zur Leistungsberechnung.
  - **duration (float)**: Dauer jedes Zeitschritts in Stunden.

  Gibt die erzeugte Wärmemenge, den Strombedarf, die Wärmeleistung, die elektrische Leistung, die Kühlenergie und die Kühlleistung zurück.

- `calculate(VLT_L, COP_data, Strompreis, q, r, T, BEW, stundensatz, duration, general_results)`: Berechnet die wirtschaftlichen und ökologischen Kennzahlen für die Flusswasser-Wärmepumpe.
  - **VLT_L (array-like)**: Vorlauftemperaturen.
  - **COP_data (array-like)**: COP-Daten zur Interpolation.
  - **Strompreis (float)**: Strompreis in €/MWh.
  - **q (float)**, **r (float)**, **T (int)**, **BEW (float)**, **stundensatz (float)**: Wirtschaftliche Parameter.
  - **duration (float)**: Simulationsdauer in Stunden.
  - **general_results (dict)**: Wörterbuch mit Lastprofilen und anderen Ergebnissen.

  Gibt ein Wörterbuch mit den berechneten Ergebnissen, einschließlich der wirtschaftlichen und ökologischen Kennzahlen, zurück.

- `to_dict()`: Wandelt die Objektattribute in ein Wörterbuch um.
- `from_dict(data)`: Erstellt ein Objekt aus einem Wörterbuch von Attributen.

36.3 Ökonomische und ökologische Überlegungen

Die `RiverHeatPump`-Klasse bietet eine Methode zur Berechnung der **Wärmegestehungskosten (WGK)**, die die Investitionskosten, den Stromverbrauch und betriebliche Faktoren berücksichtigt. Zudem werden die spezifischen CO_2-Emissionen und der Primärenergieverbrauch der Wärmepumpe berechnet.

36.4 Nutzungsbeispiel

Das folgende Beispiel zeigt, wie die `RiverHeatPump`-Klasse initialisiert und verwendet werden kann, um die Leistung einer Flusswasser-Wärmepumpe zu simulieren:

```
river_heat_pump = RiverHeatPump(
    name="Flusswärmepumpe",
    Wärmeleistung_FW_WP=300,  # kW
    Temperatur_FW_WP=12  # °C
)
results = river_heat_pump.calculate(
    VLT_L=temperature_forward,
    COP_data=cop_data,
    Strompreis=100,  # €/MWh
    q=0.03, r=0.02, T=20, BEW=0.8,
    stundensatz=50,
    duration=1,
    general_results=load_profile
)
```

In diesem Beispiel wird eine Flusswasser-Wärmepumpe mit einer Wärmeleistung von 300 kW und einer Flusswassertemperatur von 12°C simuliert. Die Leistungskennzahlen werden basierend auf den bereitgestellten Daten berechnet.

37 AqvaHeat Klasse

Die `AqvaHeat`-Klasse modelliert ein Wärmepumpensystem, das Vakuum-Eis-Schlamm-Generatoren zur Wärmerückgewinnung nutzt, und erbt von der `HeatPump`-Basisklasse. Sie enthält Methoden zur Berechnung der Leistung der Wärmepumpe sowie zur Ermittlung wirtschaftlicher und ökologischer Kennzahlen.

37.1 Attribute

- `Wärmeleistung_FW_WP (float)`: Wärmeleistung der Wärmepumpe.
- `Temperatur_FW_WP (float)`: Temperatur der Wärmequelle (z.B. Flusswasser) in Grad Celsius.
- `dT (float)`: Temperaturdifferenz im Betrieb. Standardwert: 2,5.
- `spez_Investitionskosten_Flusswasser (float)`: Spezifische Investitionskosten der Wärmepumpe in €/kW. Standardwert: 1000 €/kW.
- `spezifische_Investitionskosten_WP (float)`: Spezifische Investitionskosten der Wärmepumpe in €/kW. Standardwert: 1000 €/kW.
- `min_Teillast (float)`: Minimale Teillast als Bruchteil der Nennlast. Standardwert: 1 (keine Teillast).
- `co2_factor_electricity (float)`: CO_2-Faktor für den Stromverbrauch in tCO_2/MWh. Standardwert: 0,4.
- `primärenergiefaktor (float)`: Primärenergiefaktor für den Stromverbrauch. Standardwert: 2,4.

37.2 Methoden

- `Berechnung_WP(Wärmeleistung_L, VLT_L, COP_data)`: Berechnet die Kühlleistung, den Stromverbrauch und die angepassten Vorlauftemperaturen.

  - **Wärmeleistung_L (array-like)**: Wärmeleistungsprofil.
  - **VLT_L (array-like)**: Vorlauftemperaturen.
  - **COP_data (array-like)**: COP-Daten zur Interpolation.

  Gibt die Kühlleistung, den Stromverbrauch und die angepassten Vorlauftemperaturen zurück.

- `calculate(output_temperatures, COP_data, duration, general_results)`: Berechnet die wirtschaftlichen und ökologischen Kennzahlen für das AqvaHeat-System.

  - **output_temperatures (array-like)**: Vorlauftemperaturen.
  - **COP_data (array-like)**: COP-Daten zur Interpolation.
  - **duration (float)**: Dauer jedes Zeitschritts in Stunden.
  - **general_results (dict)**: Wörterbuch mit den Ergebnissen, wie z.B. Restlasten.

  Gibt ein Wörterbuch mit den berechneten Ergebnissen zurück, einschließlich der erzeugten Wärmemenge, des Strombedarfs, der Primärenergie und CO_2-Emissionen.

- `to_dict()`: Wandelt die Objektattribute in ein Wörterbuch um.

- `from_dict(data)`: Erstellt ein Objekt aus einem Wörterbuch von Attributen.

37.3 Ökonomische und ökologische Überlegungen

Die `AqvaHeat`-Klasse bietet eine Methode zur Berechnung der **gewichteten Durchschnittskosten der Wärmeerzeugung (WGK)**, die die Investitionskosten, den Stromverbrauch und betriebliche Faktoren berücksichtigt. Zusätzlich werden die spezifischen CO_2-Emissionen und der Primärenergieverbrauch des Systems berechnet.

37.4 Nutzungsbeispiel

Das folgende Beispiel zeigt, wie die `AqvaHeat`-Klasse initialisiert und verwendet werden kann, um die Leistung eines AqvaHeat-Systems zu simulieren:

```
aqva_heat_pump = AqvaHeat(
    name="AqvaHeat-System",
    nominal_power=100  # kW
)
results = aqva_heat_pump.calculate(
    output_temperatures=temperature_profile,
    COP_data=cop_profile,
    duration=1,
    general_results=load_profile
)
```

In diesem Beispiel wird ein AqvaHeat-System mit einer Nennleistung von 100 kW simuliert. Die Leistungskennzahlen werden basierend auf den bereitgestellten Daten berechnet.

38 CHP Klasse

Die `CHP`-Klasse modelliert ein Blockheizkraftwerk (BHKW), das sowohl thermische als auch elektrische Energie bereitstellt. Die Klasse enthält Methoden zur Berechnung der Leistung, des Brennstoffverbrauchs, der ökonomischen Kennzahlen und der Umweltauswirkungen. Das System kann mit oder ohne Speicher betrieben werden und unterstützt sowohl gas- als auch holzgasbetriebene BHKWs.

38.1 Attribute

- `name (str)`: Name des BHKW-Systems.
- `th_Leistung_BHKW (float)`: Thermische Leistung des BHKWs in kW.
- `spez_Investitionskosten_GBHKW (float)`: Spezifische Investitionskosten für gasbetriebene BHKWs in €/kW. Standard: 1500 €/kW.
- `spez_Investitionskosten_HBHKW (float)`: Spezifische Investitionskosten für holzgasbetriebene BHKWs in €/kW. Standard: 1850 €/kW.
- `el_Wirkungsgrad (float)`: Elektrischer Wirkungsgrad des BHKWs. Standard: 0,33.
- `KWK_Wirkungsgrad (float)`: Gesamtwirkungsgrad des BHKWs (kombinierte Wärme- und Stromerzeugung). Standard: 0,9.
- `min_Teillast (float)`: Minimale Teillast als Anteil der Nennlast. Standard: 0,7.
- `speicher_aktiv (bool)`: Gibt an, ob ein Speichersystem aktiv ist. Standard: `False`.
- `Speicher_Volumen_BHKW (float)`: Speichervolumen in Kubikmetern. Standard: 20 m^3.
- `T_vorlauf (float)`: Vorlauftemperatur in Grad Celsius. Standard: 90°C.
- `T_ruecklauf (float)`: Rücklauftemperatur in Grad Celsius. Standard: 60°C.
- `initial_fill (float)`: Anfangsfüllstand des Speichers als Bruchteil des maximalen Füllstands.
- `min_fill (float)`: Minimaler Füllstand des Speichers.
- `max_fill (float)`: Maximaler Füllstand des Speichers.
- `spez_Investitionskosten_Speicher (float)`: Spezifische Investitionskosten für den Speicher in €/m^3.
- `BHKW_an (bool)`: Gibt an, ob das BHKW eingeschaltet ist.

- `thermischer_Wirkungsgrad (float)`: Thermischer Wirkungsgrad des BHKWs.
- `el_Leistung_Soll (float)`: Zielwert der elektrischen Leistung des BHKWs in kW.
- `Nutzungsdauer (int)`: Lebensdauer des BHKWs in Jahren.
- `f_Inst (float)`: Installationsfaktor.
- `f_W_Insp (float)`: Wartungs- und Inspektionsfaktor.
- `Bedienaufwand (float)`: Arbeitsaufwand für den Betrieb.
- `co2_factor_fuel (float)`: CO_2-Emissionsfaktor für den Brennstoff in tCO_2/MWh.
- `primärenergiefaktor (float)`: Primärenergiefaktor für den Brennstoff.
- `co2_factor_electricity (float)`: CO_2-Emissionsfaktor für Strom in tCO_2/MWh. Standard: 0,4 tCO_2/MWh.

38.2 Methoden

- `simulate_operation(Last_L, duration)`: Berechnet die Wärme- und Stromerzeugung des BHKWs ohne Speichersystem.
  - **Last_L (array)**: Lastprofil in kW.
  - **duration (float)**: Dauer jedes Zeitschritts in Stunden.

  Diese Methode berechnet die thermische und elektrische Leistung sowie den Brennstoffverbrauch des BHKWs. Die Berechnung der erzeugten Wärme basiert auf dem thermischen Wirkungsgrad:

$$\texttt{Wärmemenge\_BHKW} = \sum_{t=1}^{n} \left(\frac{\texttt{Wärmeleistung\_kW}[t]}{1000} \right) \times \texttt{duration}$$

  Der Brennstoffbedarf wird auf Basis des kombinierten Wirkungsgrads (`KWK_Wirkungsgrad`) berechnet:

$$\texttt{Brennstoffbedarf\_BHKW} = \frac{\texttt{Wärmemenge\_BHKW} + \texttt{Strommenge\_BHKW}}{\texttt{KWK\_Wirkungsgrad}}$$

- `simulate_storage(Last_L, duration)`: Berechnet die Wärme- und Stromerzeugung des BHKWs mit einem Speichersystem.
  - **Last_L (array)**: Lastprofil des Systems in kW.
  - **duration (float)**: Dauer jedes Zeitschritts in Stunden.

  Diese Methode berechnet die Speichernutzung und das Füllniveau basierend auf der erzeugten Wärme und dem Lastprofil. Die Wärmespeicherkapazität in kWh wird berechnet:

$$\texttt{speicher\_kapazitaet} = \texttt{Speicher\_Volumen\_BHKW} \times 4186 \times (\texttt{T\_vorlauf} - \texttt{T\_ruecklauf})/3600$$

- `calculate_heat_generation_costs(Wärmemenge, Strommenge, Brennstoffbedarf, Brennstoffkosten, Strompreis, q, r, T, BEW, stundensatz)`: Berechnet die gewichteten Durchschnittskosten für das BHKW.
  - **Wärmemenge (float)**: Erzeugte Wärmemenge in MWh.
  - **Strommenge (float)**: Erzeugte Strommenge in MWh.
  - **Brennstoffbedarf (float)**: Brennstoffverbrauch in MWh.
  - **Brennstoffkosten (float)**: Brennstoffkosten in €/MWh.
  - **Strompreis (float)**: Strompreis in €/MWh.
  - **q (float)**, **r (float)**: Faktoren für Kapitalrückgewinnung und Preissteigerung.
  - **T (int)**: Zeitperiode in Jahren.
  - **BEW (float)**: Abzinsungsfaktor.
  - **stundensatz (float)**: Arbeitskosten in €/Stunde.

  Diese Methode berechnet die spezifischen Wärmeerzeugungskosten (`WGK_BHKW`) auf Basis der Investitions- und Betriebskosten:

$$\texttt{WGK\_BHKW} = \frac{\texttt{A\_N}}{\texttt{Wärmemenge}}$$

- `calculate(Gaspreis, Holzpreis, Strompreis, q, r, T, BEW, stundensatz, duration, general_results)`: Führt eine vollständige Simulation des BHKWs durch, einschließlich der Berechnung der ökonomischen und ökologischen Kennzahlen.
  - **Gaspreis (float)**: Gaspreis in €/MWh.
  - **Holzpreis (float)**: Preis für Holzgas in €/MWh.
  - **Strompreis (float)**: Strompreis in €/MWh.
  - **q (float)**, **r (float)**, **T (int)**, **BEW (float)**, **stundensatz (float)**: Parameter für die Kostenberechnung.
  - **duration (float)**: Simulationsdauer in Stunden.
  - **general_results (dict)**: Wörterbuch mit allgemeinen Ergebnissen wie Lastprofilen.

  Die Methode berechnet zudem die spezifischen CO_2-Emissionen und den Primärenergieverbrauch:

$$\texttt{co2\_emissions} = \texttt{Brennstoffbedarf} \times \texttt{co2\_factor\_fuel}$$

$$\texttt{primärenergie} = \texttt{Brennstoffbedarf} \times \texttt{primärenergiefaktor}$$

38.3 Ökonomische und ökologische Überlegungen

Die `CHP`-Klasse ermöglicht die Berechnung der Wärmegestehungsksoten und der spezifischen CO_2-Emissionen eines BHKW-Systems. Diese Berechnungen berücksichtigen die Brennstoffkosten, die Stromerzeugung sowie die Arbeits- und Betriebskosten. Darüber hinaus werden die CO_2-Einsparungen durch die Stromerzeugung und der Primärenergieverbrauch des Systems ermittelt.

38.4 Nutzungsbeispiel

Das folgende Beispiel zeigt die Initialisierung und Verwendung der `CHP`-Klasse zur Simulation eines gasbetriebenen BHKWs:

```
chp_system = CHP(
    name="Gas-BHKW",
    th_Leistung_BHKW=200,  # kW
    speicher_aktiv=True,
    Speicher_Volumen_BHKW=30  # m³
)
results = chp_system.calculate(
    Gaspreis=60,  # €/MWh
    Holzpreis=40,  # €/MWh
    Strompreis=100,  # €/MWh
    q=0.03, r=0.02, T=15, BEW=0.8,
    stundensatz=50,
    duration=1,
    general_results=load_profile
)
```

In diesem Beispiel wird ein gasbetriebenes BHKW mit einer thermischen Leistung von 200 kW und einem Speichervolumen von 30 m³ simuliert. Die ökonomische und ökologische Leistung des Systems wird anhand der bereitgestellten Eingaben berechnet.

39 BiomassBoiler Klasse

Die `BiomassBoiler`-Klasse modelliert ein Biomassekesselsystem und enthält Methoden zur Simulation der Leistung des Kessels, des Brennstoffverbrauchs, der Speicherintegration sowie zur ökonomischen und ökologischen Analyse.

39.1 Attribute

- `name (str)`: Name des Biomassekesselsystems.
- `P_BMK (float)`: Kesselleistung in kW.
- `Größe_Holzlager (float)`: Größe des Holzlagers in Kubikmetern.
- `spez_Investitionskosten (float)`: Spezifische Investitionskosten für den Kessel in €/kW.
- `spez_Investitionskosten_Holzlager (float)`: Spezifische Investitionskosten für das Holzlager in €/m³.
- `Nutzungsgrad_BMK (float)`: Wirkungsgrad des Biomassekessels.
- `min_Teillast (float)`: Minimale Teillast in Relation zur Nennlast.
- `speicher_aktiv (bool)`: Gibt an, ob ein Speichersystem aktiv ist.

- `Speicher_Volumen (float)`: Volumen des Wärmespeichers in Kubikmetern.
- `T_vorlauf (float)`: Vorlauftemperatur in Grad Celsius.
- `T_ruecklauf (float)`: Rücklauftemperatur in Grad Celsius.
- `initial_fill (float)`: Anfangsfüllstand des Speichers in Relation zum Gesamtvolumen.
- `min_fill (float)`: Minimaler Füllstand des Speichers in Relation zum Gesamtvolumen.
- `max_fill (float)`: Maximaler Füllstand des Speichers in Relation zum Gesamtvolumen.
- `spez_Investitionskosten_Speicher (float)`: Spezifische Investitionskosten für den Wärmespeicher in €/m^3.
- `BMK_an (bool)`: Gibt an, ob der Kessel in Betrieb ist.
- `opt_BMK_min (float)`: Minimale Kesselleistung für die Optimierung.
- `opt_BMK_max (float)`: Maximale Kesselleistung für die Optimierung.
- `opt_Speicher_min (float)`: Minimale Speicherkapazität für die Optimierung.
- `opt_Speicher_max (float)`: Maximale Speicherkapazität für die Optimierung.
- `Nutzungsdauer (int)`: Nutzungsdauer des Biomassekessels in Jahren.
- `f_Inst (float)`: Installationsfaktor.
- `f_W_Insp (float)`: Wartungs- und Inspektionsfaktor.
- `Bedienaufwand (float)`: Arbeitsaufwand für den Betrieb.
- `co2_factor_fuel (float)`: CO_2-Faktor für den Brennstoff in tCO_2/MWh.
- `primärenergiefaktor (float)`: Primärenergiefaktor für den Brennstoff.

39.2 Methoden

- `simulate_operation(Last_L, duration)`: Simuliert den Betrieb des Biomassekessels über ein gegebenes Lastprofil und eine bestimmte Zeitdauer.
  - **Last_L (array)**: Lastprofil des Systems in kW.
  - **duration (float)**: Dauer jedes Zeitschritts in Stunden.

  Diese Methode simuliert den Betrieb des Kessels und berechnet die erzeugte Wärmemenge sowie den Brennstoffbedarf. Die Formel für die Wärmeerzeugung ist:

$$\texttt{Wärmemenge\_BMK} = \sum_{t=1}^{n} \left(\frac{\texttt{Wärmeleistung\_kW}[t]}{1000} \right) \times \texttt{duration}$$

Die Brennstoffmenge wird auf Basis des Wirkungsgrades (`Nutzungsgrad_BMK`) berechnet:

$$\texttt{Brennstoffbedarf\_BMK} = \frac{\texttt{Wärmemenge\_BMK}}{\texttt{Nutzungsgrad\_BMK}}$$

- `simulate_storage(Last_L, duration)`: Simuliert den Betrieb des Speichersystems und passt die Kesselleistung zur Optimierung des Speicherbetriebs an.
  - **Last_L (array)**: Lastprofil des Systems in kW.
  - **duration (float)**: Dauer jedes Zeitschritts in Stunden.

  Diese Methode berechnet die Speicherfüllstände, indem sie die Wärmemenge, die in den Speicher geladen oder daraus entnommen wird, auf Basis der Vor- und Rücklauftemperaturen bestimmt. Das Speichervolumen in kWh wird folgendermaßen berechnet:

  $$\texttt{speicher\_kapazitaet} = \texttt{Speicher\_Volumen} \times 4186 \times (\texttt{T\_vorlauf} - \texttt{T\_ruecklauf})/3600$$

  Dabei ist 4186 die spezifische Wärmekapazität von Wasser in J/kgK.

- `calculate_heat_generation_costs(Wärmemenge, Brennstoffbedarf, Brennstoffkosten, q, r, T, BEW, stundensatz)`: Berechnet die gewichteten Durchschnittskosten der Wärmeerzeugung (WGK) basierend auf den Investitionskosten, Brennstoffkosten und Betriebskosten des Systems.
  - **Wärmemenge (float)**: Erzeugte Wärmemenge in kWh.
  - **Brennstoffbedarf (float)**: Brennstoffverbrauch in MWh.
  - **Brennstoffkosten (float)**: Kosten des Biomassebrennstoffs in €/MWh.
  - **q (float)**: Kapitalrückgewinnungsfaktor.
  - **r (float)**: Preissteigerungsfaktor.
  - **T (int)**: Zeitperiode in Jahren.
  - **BEW (float)**: Betriebskostenfaktor.
  - **stundensatz (float)**: Stundensatz für den Arbeitsaufwand.

  Die Methode berechnet die Investitionskosten des Kessels, des Holzlagers und des Speichers und verwendet die Annuitätenmethode zur Berechnung der jährlichen Kapitalrückzahlung:

  $$\texttt{A\_N} = \text{annuität}(\texttt{Investitionskosten}, \texttt{Nutzungsdauer}, \texttt{f\_Inst}, \texttt{f\_W\_Insp}, \texttt{Bedienaufwand}, q, r, T)$$

  Die spezifischen Wärmeerzeugungskosten werden durch Division der Gesamtkosten durch die erzeugte Wärmemenge berechnet:

  $$\texttt{WGK\_BMK} = \frac{\texttt{A\_N}}{\texttt{Wärmemenge}}$$

- `calculate(Holzpreis, q, r, T, BEW, stundensatz, duration, general_results)`: Führt eine vollständige Simulation des Biomassekessels durch, einschließlich der Berechnung der Wärmeerzeugung und ökonomischer Parameter.

- **Holzpreis (float)**: Preis des Brennstoffs (Holzpellets) in €/MWh.
- **q (float)**: Kapitalrückgewinnungsfaktor.
- **r (float)**: Preissteigerungsfaktor.
- **T (int)**: Zeitperiode in Jahren.
- **BEW (float)**: Betriebskostenfaktor.
- **stundensatz (float)**: Stundensatz für den Arbeitsaufwand.
- **duration (float)**: Dauer jedes Simulationsschritts in Stunden.
- **general_results (dict)**: Wörterbuch mit allgemeinen Ergebnissen, wie z.B. Restlasten.

Diese Methode berechnet die Leistung des Kessels, den Brennstoffverbrauch und die Wärmemenge. Falls ein Speicher aktiviert ist, wird die Methode `storage()` aufgerufen. Zusätzlich werden die spezifischen CO_2-Emissionen und der Primärenergieverbrauch berechnet:

$$\texttt{co2\_emissions} = \texttt{Brennstoffbedarf} \times \texttt{co2\_factor\_fuel}$$

$$\texttt{primärenergie} = \texttt{Brennstoffbedarf} \times \texttt{primärenergiefaktor}$$

Diese Berechnungen ermöglichen eine umfassende Analyse der ökologischen Auswirkungen des Systems.

39.3 Ökonomische und ökologische Überlegungen

Die `BiomassBoiler`-Klasse ermöglicht die Berechnung der **Wärmegestehungskosten (WGK)** unter Berücksichtigung der Investitions-, Installations- und Betriebskosten sowie des Brennstoffverbrauchs. Die spezifischen CO_2-Emissionen werden auf Basis des verbrannten Brennstoffs berechnet, und der **Primärenergieverbrauch** wird anhand der erzeugten Wärmemenge und des Primärenergiefaktors ermittelt.

39.4 Nutzungsbeispiel

Die `BiomassBoiler`-Klasse kann verwendet werden, um die Leistung eines Biomasseheizsystems mit oder ohne Speicher zu simulieren. Das folgende Beispiel zeigt, wie die Klasse initialisiert und verwendet werden kann:

```
biomass_boiler = BiomassBoiler(
    name="Biomassekessel",
    P_BMK=500,  # kW
    Größe_Holzlager=50,  # m³
    Nutzungsgrad_BMK=0.85,
    Speicher_Volumen=100,  # m³
    speicher_aktiv=True
)
results = biomass_boiler.calculate(
    Holzpreis=20,  # €/MWh
    q=0.05, r=0.03, T=15, BEW=1.1,
    stundensatz=50,
```

```
    duration=1,
    general_results=load_profile
)
```

In diesem Beispiel wird ein Biomassekessel mit einer Leistung von 500 kW und einem Holzlager von 50 m^3 simuliert. Das System enthält einen 100 m^3 großen Wärmespeicher. Leistung und Kosten werden basierend auf den eingegebenen Parametern berechnet.

40 GasBoiler Klasse

Die `GasBoiler`-Klasse repräsentiert ein Gaskesselsystem, das dazu dient, die Leistung, Kosten und Emissionen eines Gaskessels in einem Heizsystem zu berechnen und zu simulieren. Die Klasse umfasst zentrale ökonomische, betriebliche und ökologische Faktoren und ermöglicht eine umfassende Analyse in Energiesystemen.

40.1 Attribute

- `name (str)`: Name des Gaskesselsystems.
- `spez_Investitionskosten (float)`: Spezifische Investitionskosten für den Gaskessel in €/kW.
- `Nutzungsgrad (float)`: Wirkungsgrad des Gaskessels, der typischerweise zwischen 0,8 und 1,0 liegt. Er repräsentiert das Verhältnis von nutzbarer Wärmeleistung zur gesamten zugeführten Energie.
- `Faktor_Dimensionierung (float)`: Dimensionierungsfaktor, der eine eventuelle Überdimensionierung berücksichtigt.
- `Nutzungsdauer (int)`: Lebensdauer des Gaskessels in Jahren. Standardmäßig 20 Jahre.
- `f_Inst (float)`: Installationsfaktor, der zusätzliche Kosten aufgrund von Installationskomplexität repräsentiert.
- `f_W_Insp (float)`: Inspektionsfaktor, der periodische Wartungs- und Inspektionskosten berücksichtigt.
- `Bedienaufwand (float)`: Betriebskosten in Form von Arbeitsaufwand.
- `co2_factor_fuel (float)`: CO_2-Emissionsfaktor für den Brennstoff (Erdgas), typischerweise in tCO_2/MWh.
- `primärenergiefaktor (float)`: Primärenergiefaktor für den Brennstoff, der die Menge an Primärenergie darstellt, die benötigt wird, um eine Einheit nutzbare Energie (MWh) zu erzeugen. Dieser Faktor berücksichtigt Energieverluste in der Brennstofflieferkette.

40.2 Methoden

Die `GasBoiler`-Klasse enthält mehrere Methoden, die die Auslegung und Berechnung eines Gaskessels im Detail beschreiben. Im Folgenden werden die mathematischen Grundlagen und Berechnungslogiken der wichtigsten Methoden erläutert.

40.2.1 simulate_operation(Last_L, duration)

Diese Methode simuliert den Betrieb des Gaskessels basierend auf einem gegebenen Lastprofil und der Betriebsdauer. Sie berechnet die Wärmeerzeugung des Kessels, den Brennstoffbedarf sowie die maximale Leistung. Die wichtigsten Schritte der Berechnung sind:

- **Berechnung der Wärmeleistung:** Zunächst wird das gegebene Lastprofil `Last_L` verwendet, um die stündliche Wärmeleistung in kW zu bestimmen. Da negative Lasten (falls vorhanden) keinen Sinn ergeben, wird die Funktion `np.maximum()` verwendet, um negative Werte auf 0 zu setzen:

$$\texttt{Wärmeleistung\_kW} = \max(\texttt{Last\_L}, 0)$$

- **Berechnung der Wärmemenge:** Die Wärmemenge (`Wärmemenge_Gaskessel`) wird über die Summe der stündlichen Wärmeleistung, multipliziert mit der Simulationsdauer `duration` in Stunden, berechnet:

$$\texttt{Wärmemenge\_Gaskessel} = \sum_{t=1}^{n} \left(\frac{\texttt{Wärmeleistung\_kW}[t]}{1000} \right) \times \texttt{duration}$$

  Dabei wird die Wärmeleistung von kW in MWh umgerechnet (Faktor 1000).

- **Berechnung des Gasbedarfs:** Der Gasbedarf (`Gasbedarf`) wird aus der erzeugten Wärmemenge und dem Wirkungsgrad (`Nutzungsgrad`) des Gaskessels berechnet:

$$\texttt{Gasbedarf} = \frac{\texttt{Wärmemenge\_Gaskessel}}{\texttt{Nutzungsgrad}}$$

  Der Wirkungsgrad berücksichtigt die Verluste, die bei der Umwandlung von Brennstoff in nutzbare Wärme entstehen.

- **Maximale Leistung:** Die maximale Leistung des Gaskessels (`P_max`) wird basierend auf der maximalen Last im Profil und einem Dimensionierungsfaktor berechnet, der mögliche Überdimensionierungen des Kessels berücksichtigt:

$$\texttt{P\_max} = \max(\texttt{Last\_L}) \times \texttt{Faktor\_Dimensionierung}$$

40.2.2 calculate_heat_generation_cost(Brennstoffkosten, q, r, T, BEW, stundensatz)

Diese Methode berechnet die Wärmegestehungskosten (**WGK**). Diese beinhalten sowohl Investitions- als auch Betriebskosten, um die tatsächlichen Kosten der Wärmeerzeugung pro MWh zu ermitteln. Die Berechnung erfolgt in mehreren Schritten:

- **Berechnung der Investitionskosten:** Die spezifischen Investitionskosten (spez_Investitionsk werden mit der maximalen Leistung des Kessels (P_max) multipliziert, um die gesamten Investitionskosten (Investitionskosten) zu erhalten:

$$\texttt{Investitionskosten} = \texttt{spez\_Investitionskosten} \times \texttt{P\_max}$$

- **Annuität:** Um die jährlichen Kapitalrückzahlungen zu berechnen, wird der Annuitätenfaktor verwendet, der sowohl die Kapitalrückzahlung über die Lebensdauer des Kessels als auch Installations- und Wartungskosten berücksichtigt. Der Annuitätenfaktor A_N berechnet sich mit der Funktion annuität, die auf den Kapitalrückgewinnungsfaktor (q), die Lebensdauer (T), und die Installations- und Wartungskosten (f_Inst und f_W_Insp) eingeht:

$$\texttt{A\_N} = \text{annuität}(\texttt{Investitionskosten}, \texttt{Nutzungsdauer}, \texttt{f\_Inst}, \texttt{f\_W\_Insp}, \texttt{Bedienaufwand}, q, r, T)$$

- **Berechnung der Wärmegestehungskosten:** Die jährlichen Gesamtkosten werden durch die erzeugte Wärmemenge geteilt, um die spezifischen Wärmegestehungskosten (WGK_GK) zu erhalten:

$$\texttt{WGK\_GK} = \frac{\texttt{A\_N}}{\texttt{Wärmemenge\_Gaskessel}}$$

  Diese Kosten beinhalten die Investitionskosten, Betriebskosten und den Gaspreis.

40.2.3 calculate(Gaspreis, q, r, T, BEW, stundensatz, duration, Last_L, general_results)

Diese Methode führt eine vollständige Berechnung der Systemleistung und Kostenanalyse durch. Sie kombiniert die oben beschriebenen Schritte und berechnet die wichtigsten ökonomischen und ökologischen Kennzahlen:

- **Berechnung der Wärmemenge und des Gasbedarfs:** Die Methode ruft Gaskessel() auf, um die Wärmemenge und den Gasbedarf zu berechnen.

- **Berechnung der CO_2-Emissionen:** Die CO_2-Emissionen werden auf Basis des Gasverbrauchs und des spezifischen CO_2-Faktors für Erdgas (co2_factor_fuel) berechnet:

$$\texttt{co2\_emissions} = \texttt{Gasbedarf} \times \texttt{co2\_factor\_fuel}$$

  Um die spezifischen CO_2-Emissionen pro erzeugte Wärmeeinheit (in tCO_2/MWh) zu ermitteln, werden die gesamten CO_2-Emissionen durch die Wärmemenge geteilt:

$$\texttt{spec\_co2\_total} = \frac{\texttt{co2\_emissions}}{\texttt{Wärmemenge\_Gaskessel}}$$

- **Primärenergieverbrauch:** Der Primärenergieverbrauch wird durch Multiplikation des Gasverbrauchs mit dem Primärenergiefaktor (primärenergiefaktor) berechnet:

$$\texttt{primärenergie} = \texttt{Gasbedarf} \times \texttt{primärenergiefaktor}$$

- **Ergebnisse:** Am Ende werden die berechneten Werte in einem Wörterbuch (`results`) zurückgegeben, das die Wärmemenge, die zeitlich aufgelöste Wärmeleistung (`Wärmeleistung_L`), den Brennstoffbedarf, die gewichteten Durchschnittskosten (`WGK`), die spezifischen CO_2-Emissionen und den Primärenergieverbrauch enthält.

Die vollständige Berechnungsmethode ermöglicht die Simulation der Leistung eines Gaskessels über einen bestimmten Zeitraum und liefert umfassende ökonomische und ökologische Kennzahlen, die für eine energetische Bewertung entscheidend sind.

40.3 Ökonomische und ökologische Überlegungen

Die `GasBoiler`-Klasse wurde entwickelt, um sowohl die ökonomischen als auch die ökologischen Auswirkungen eines Gaskesselsystems zu simulieren. Die **Wärmegestehungskosten (WGK)** berücksichtigen sowohl Investitionskosten als auch Betriebskosten, einschließlich Brennstoffpreise, Arbeitskosten und Wartung. Zudem werden die **CO_2-Emissionen** des Systems basierend auf dem Brennstoffverbrauch und dem spezifischen CO_2-Faktor für Erdgas berechnet, um eine Analyse des ökologischen Fußabdrucks des Systems zu ermöglichen. Der **Primärenergieverbrauch** wird ebenfalls berechnet, um Einblicke in die Gesamtenergieeffizienz und Nachhaltigkeit des Systems zu geben.

40.4 Nutzungsbeispiel

Das folgende Beispiel zeigt, wie die `GasBoiler`-Klasse initialisiert und verwendet werden kann:

```
gas_boiler = GasBoiler(
    name="Gasheizkessel",
    spez_Investitionskosten=35,  # €/kW
    Nutzungsgrad=0.92,  # 92% Effizienz
    Faktor_Dimensionierung=1.1  # Leichte Überdimensionierung
)

results = gas_boiler.calculate(
    Gaspreis=30,  # €/MWh
    q=0.03, r=0.02, T=20, BEW=1,
    stundensatz=50,
    duration=1,
    Last_L=load_profile,
    general_results={'Restlast_L': residual_load}
)
```

In diesem Beispiel wird der Gaskessel mit einem Wirkungsgrad von 92% und einer leichten Überdimensionierung dimensioniert. Die Berechnungsmethode schätzt die Wärmeerzeugung, den Gasbedarf, die CO_2-Emissionen und die gewichteten Durchschnittskosten der Wärmeerzeugung basierend auf einem Lastprofil und allgemeinen Systemparametern.

41 Einleitung

Dieses Dokument beschreibt detailliert den Algorithmus zur Berechnung der Solarstrahlung, die auf eine geneigte Oberfläche trifft. Der Algorithmus berücksichtigt Wetterdaten, die geografische Lage, den Einfallswinkel der Sonnenstrahlen, die Neigung der Kollektorfläche und den Albedo-Effekt. Diese Methode wird zur Simulation von Solaranlagen verwendet, insbesondere für solarthermische Anwendungen in Wärmenetzen.

Die Berechnungsgrundlage stammt aus dem Ertragsberechnungsprogramm für Solarthermie in Wärmenetzen ScenoCalc Fernwärme 2.0 `https://www.scfw.de/`.

42 Berechnung der Solarstrahlung

Die Hauptfunktion des Algorithmus ist `Berechnung_Solarstrahlung`, die die Strahlungsintensität auf einer geneigten Kollektorfläche berechnet. Die Berechnung erfolgt in mehreren Schritten, die im Folgenden beschrieben werden.

42.1 Berechnung des Tagwinkels und der Zeitkorrektur

Der Tag des Jahres N wird in einen Winkel B umgerechnet:

$$B = \frac{360 \times (N - 1)}{365}$$

Dieser Winkel ist notwendig, um die Sonnenposition im Jahreszyklus zu berechnen.

Die Zeitkorrektur E, die die Abweichungen zwischen Sonnenzeit und Standardzeit berücksichtigt, wird mit folgender Formel berechnet:

$$E = 229.2 \cdot (0.000075 + 0.001868\cos(B) - 0.032077\sin(B) - 0.014615\cos(2B) - 0.04089\sin(2B))$$

42.2 Berechnung der Sonnenzeit

Die Sonnenzeit wird unter Berücksichtigung der geografischen Länge L, der Standardlänge des Zeitzonenmeridians L_{std} und der Zeitkorrektur E berechnet:

$$t_{\text{solar}} = \frac{(t_{\text{Uhrzeit}} - 0.5) \cdot 3600 + E \cdot 60 + 4 \cdot (L_{std} - L) \cdot 60}{3600}$$

42.3 Sonnenzenitwinkel und Deklination der Sonne

Die Deklination der Sonne δ wird als Funktion des Tages des Jahres berechnet:

$$\delta = 23.45 \cdot \sin\left(\frac{360 \cdot (284 + N)}{365}\right)$$

Der Sonnenzenitwinkel SZA beschreibt den Winkel zwischen dem Lot auf die Erdoberfläche und den Sonnenstrahlen:

$$SZA = \arccos\left(\cos(\phi)\cos(h)\cos(\delta) + \sin(\phi)\sin(\delta)\right)$$

wobei ϕ die geografische Breite und h der Stundenwinkel der Sonne ist:

$$h = -180 + t_{\text{solar}} \times \frac{180}{12}$$

42.4 Berechnung des Sonnenazimutwinkels

Der Sonnenazimutwinkel γ_S beschreibt den Winkel zwischen der Sonne und einer Referenzrichtung (in der Regel Süden) auf der horizontalen Ebene. Der Azimutwinkel wird benötigt, um die Position der Sonne relativ zur Oberfläche der Erdkruste zu bestimmen. Er wird berechnet unter Berücksichtigung des Stundenwinkels h und des Sonnenzenitwinkels SZA:

$$\gamma_S = \text{sgn}(h) \cdot \arccos\left(\frac{\cos(SZA) \cdot \sin(\phi) - \sin(\delta)}{\sin(SZA) \cdot \cos(\phi)}\right)$$

Hierbei wird die Signum-Funktion verwendet, um die korrekte Richtung des Azimutwinkels zu bestimmen, abhängig davon, ob die Sonne östlich oder westlich des Meridians steht.

42.5 Berechnung des Einfallswinkels auf die Kollektorfläche

Der Einfallswinkel der Sonnenstrahlen auf eine geneigte Kollektorfläche I_aC beschreibt den Winkel zwischen der Kollektorfläche und den einfallenden Strahlen der Sonne. Dieser Winkel beeinflusst maßgeblich die Intensität der direkten Sonneneinstrahlung, die auf die Kollektorfläche trifft. Der Einfallswinkel wird durch folgende Formel berechnet:

$$I_aC = \arccos\left(\cos(SZA) \cdot \cos(CTA) + \sin(SZA) \cdot \sin(CTA) \cdot \cos(\gamma_S - \gamma_C)\right)$$

wobei: - SZA der Sonnenzenitwinkel ist, - CTA der Neigungswinkel der Kollektorfläche, - γ_S der Sonnenazimutwinkel und - γ_C der Azimutwinkel des Kollektors.

Je kleiner der Einfallswinkel, desto mehr direkte Strahlung trifft auf die Kollektorfläche. Diese Berechnung berücksichtigt sowohl die Neigung als auch die Ausrichtung der Kollektorfläche, was insbesondere bei nicht optimal nach Süden ausgerichteten Kollektoren von Bedeutung ist.

Zusätzlich wird der Einfallswinkel in Bezug auf die Ost-West-Orientierung Incidence_angle_EW und die Nord-Süd-Orientierung Incidence_angle_NS berechnet. Diese Winkel dienen dazu, die Effekte der Strahlungsmodifikatoren (IAM) in beide Richtungen zu berechnen. Für die Ost-West-Orientierung lautet die Formel:

$$f_{\text{EW}} = \arctan\left(\frac{\sin(SZA) \cdot \sin(\gamma_S - \gamma_C)}{\cos(I_aC)}\right)$$

und für die Nord-Süd-Orientierung:

$$f_{\text{NS}} = -(180/\pi) \cdot \arctan\left(\tan(SZA) \cdot \cos(\gamma_S - \gamma_C)\right) - CTA$$

Diese Funktionen stellen sicher, dass der Einfallswinkel korrekt auf die jeweiligen Ausrichtungen des Kollektors angewendet wird.

42.6 Berechnung der direkten und diffusen Strahlung

Die direkte Strahlung auf die horizontale Fläche *Gbhoris* wird berechnet als:

$$Gbhoris = D_L \cdot \cos(SZA)$$

Die diffuse Strahlung *Gdhoris* ergibt sich als Differenz zwischen der globalen Strahlung G und der direkten Strahlung:

$$Gdhoris = G - Gbhoris$$

42.7 Atmosphärischer Diffusanteil und Gesamtstrahlung

Der atmosphärische Diffusanteil Ai wird basierend auf der horizontalen Direktstrahlung $Gbhoris$ und der Solarkonstanten (1367 W/m²) wie folgt berechnet:

$$Ai = \frac{Gbhoris}{1367 \cdot (1 + 0.033 \cdot \cos(360 \cdot N/365)) \cdot \cos(SZA)}$$

Die Gesamtstrahlung GT_HGk auf der geneigten Oberfläche wird berechnet durch:

$$GT_HGk = Gbhoris \cdot R_b + Gdhoris \cdot Ai \cdot R_b + Gdhoris \cdot (1 - Ai) \cdot 0.5 \cdot (1 + \cos(CTA)) + G \cdot \text{Albedo} \cdot 0.5 \cdot (1 - \cos(CT$$

Hierbei beschreibt R_b das Verhältnis der Strahlungsintensität auf der geneigten Fläche zur horizontalen Fläche.

43 Zusammenfassung

Dieser Algorithmus berechnet die Solarstrahlung auf geneigten Flächen, basierend auf physikalischen Modellen und atmosphärischen Einflüssen. Durch die Berücksichtigung von direkter, diffuser und reflektierter Strahlung kann der Energieertrag einer Solaranlage realistisch simuliert werden. Diese Berechnung ist entscheidend für die Planung und Optimierung solarthermischer Anlagen.

44 SolarThermal Klasse

Die `SolarThermal`-Klasse modelliert ein solarthermisches System und enthält Methoden zur Berechnung der Leistung, der wirtschaftlichen Kennzahlen und der Umweltauswirkungen. Die Klasse unterstützt verschiedene Arten von Sonnenkollektoren (z. B. Flachkollektoren und Vakuumröhrenkollektoren) und enthält Parameter für die Integration eines Speichersystems.

44.1 Attribute

- `name (str)`: Name der Solarthermieanlage.
- `bruttofläche_STA (float)`: Brutto-Kollektorfläche der Solarthermieanlage in Quadratmetern.
- `vs (float)`: Volumen des Speichersystems in Kubikmetern.
- `Typ (str)`: Typ des Sonnenkollektors, z.B. "Flachkollektor" oder "Vakuumröhrenkollektor".
- `kosten_speicher_spez (float)`: Spezifische Kosten für das Speichersystem in €/m³.
- `kosten_fk_spez (float)`: Spezifische Kosten für Flachkollektoren in €/m².
- `kosten_vrk_spez (float)`: Spezifische Kosten für Vakuumröhrenkollektoren in €/m².
- `Tsmax (float)`: Maximale Speichertemperatur in Grad Celsius.

- `Longitude (float)`: Längengrad des Installationsortes.
- `STD_Longitude (float)`: Standardlängengrad der Zeitzone.
- `Latitude (float)`: Breitengrad des Installationsortes.
- `East_West_collector_azimuth_angle (float)`: Azimutwinkel des Sonnenkollektors in Grad.
- `Collector_tilt_angle (float)`: Neigungswinkel des Sonnenkollektors in Grad.
- `Tm_rl (float)`: Mittlere Rücklauftemperatur in Grad Celsius.
- `Qsa (float)`: Anfangsleistung.
- `Vorwärmung_K (float)`: Vorwärmung in Kelvin.
- `DT_WT_Solar_K (float)`: Temperaturdifferenz über den Solar-Wärmetauscher in Kelvin.
- `DT_WT_Netz_K (float)`: Temperaturdifferenz über den Netz-Wärmetauscher in Kelvin.
- `opt_volume_min (float)`: Minimales Optimierungsvolumen in Kubikmetern.
- `opt_volume_max (float)`: Maximales Optimierungsvolumen in Kubikmetern.
- `opt_area_min (float)`: Minimale Optimierungsfläche in Quadratmetern.
- `opt_area_max (float)`: Maximale Optimierungsfläche in Quadratmetern.
- `kosten_pro_typ (dict)`: Wörterbuch, das die spezifischen Kosten für verschiedene Arten von Sonnenkollektoren enthält.
- `Kosten_STA_spez (float)`: Spezifische Kosten für die Solarthermieanlage in €/m^2.
- `Nutzungsdauer (int)`: Lebensdauer der Solarthermieanlage in Jahren (Standardwert: 20 Jahre).
- `f_Inst (float)`: Installationsfaktor.
- `f_W_Insp (float)`: Wartungs- und Inspektionsfaktor.
- `Bedienaufwand (float)`: Betriebsaufwand für das System.
- `Anteil_Förderung_BEW (float)`: Fördersatz für das Erneuerbare-Energien-Gesetz.
- `Betriebskostenförderung_BEW (float)`: Betriebskostenzuschuss pro MWh thermischer Energie.
- `co2_factor_solar (float)`: CO_2-Faktor für Solarenergie (typisch 0 für Solarwärme).
- `primärenergiefaktor (float)`: Primärenergiefaktor (typisch 0 für Solarthermie).

44.2 Methoden

- `calculate_heat_generation_costs(q, r, T, BEW, stundensatz)`: Berechnet die Wärmegestehungsksoten (WGK) basierend auf den Investitions- und Betriebskosten des Systems sowie auf der Förderfähigkeit nach dem BEW.
  - **q (float)**: Kapitalrückgewinnungsfaktor.
  - **r (float)**: Preissteigerungsfaktor.
  - **T (int)**: Betrachtungszeitraum in Jahren.
  - **BEW (str)**: Angabe der Betrachtung der Förderung nach BEW ("Ja" oder "Nein").
  - **stundensatz (float)**: Stundensatz für Arbeitsaufwand.

  Gibt die Wärmegesethungskosten des Systems basierend auf Investitionen, Förderungen und Betriebskosten zurück.

- `calculate(VLT_L, RLT_L, TRY, time_steps, calc1, calc2, q, r, T, BEW, stundensatz, duration, general_results)`: Simuliert die Leistung des solarthermischen Systems über einen bestimmten Zeitraum und berücksichtigt dabei Vorlauf- und Rücklauftemperature Wetterdaten und Betriebskosten. Die Berechnung erfolgt in einer ausgelagerten Funktion. Dies wird im Abschnitt "Ertragsberechnung" genauer erläutert.
  - **VLT_L (array)**: Array von Vorlauftemperaturen in Grad Celsius.
  - **RLT_L (array)**: Array von Rücklauftemperaturen in Grad Celsius.
  - **TRY (array)**: Testreferenzjahr-Wetterdaten.
  - **time_steps (array)**: Array von Zeitschritten für die Simulation.
  - **calc1 (float)**, **calc2 (float)**: Zusätzliche Berechnungsparameter.
  - **q (float)**, **r (float)**, **T (int)**, **BEW (str)**, **stundensatz (float)**: Parameter für die Kostenberechnung.
  - **duration (float)**: Dauer jedes Simulationszeitschritts.
  - **general_results (dict)**: Wörterbuch, das allgemeine Ergebnisse aus der Simulation enthält, wie z.B. Restlasten.

  Gibt ein Dictionary mit den Ergebnissen der Simulation zurück, einschließlich Wärmeerzeugung, spezifischen CO_2-Emissionen, Primärenergieverbrauch und Speicherstatus.

- `to_dict()`: Wandelt das `SolarThermal`-Objekt in ein Wörterbuch um, um eine einfache Serialisierung und Speicherung zu ermöglichen.

- `from_dict(data)`: Erstellt ein `SolarThermal`-Objekt aus einem Wörterbuch von Attributen.

44.3 Ertragsberechnung

Im folgenden wird die Berechung der Solarthermie erläutert. Die Datengrundlage für die Berechnung ist das Testreferenzjahr (TRY), das Wetterdaten wie Temperatur, Windgeschwindigkeit

und Strahlungsdaten enthält. Die Berechnung erfolgt in mehreren Schritten, die im Folgenden beschrieben werden. Es werden die charakteristischen Parameter der Solarkollektoren, die Speichergrößen und die Systemverluste in die Berechnung einbezogen. Die Berechnung erfolgt basierend auf physikalischen Modellen, die den Energiefluss durch die Solarkollektoren, die Wärmeübertragung im Speicher und die Rohrleitungsverluste abbilden.

Yield calculation program for solar thermal energy in heating networks (calculation basis: ScenoCalc District Heating 2.0, `https://www.scfw.de/`)

Eingabeparameter

Die Funktion `Berechnung_STA` verwendet die folgenden Eingabeparameter:

- **Bruttofläche_STA**: Die Bruttofläche der Solaranlage in Quadratmetern.
- **VS**: Speichervolumen der Solaranlage in Litern.
- **Typ**: Der Typ der Solaranlage (`"Flachkollektor"` oder `"Vakuumröhrenkollektor"`).
- **Last_L**: Array des Lastprofils in Watt.
- **VLT_L, RLT_L**: Vorlauf- und Rücklauftemperaturprofil.
- **TRY**: Testreferenzjahr-Daten (Temperatur, Windgeschwindigkeit, Direktstrahlung, Globalstrahlung).
- **time_steps**: Zeitstempel.
- **Longitude, Latitude**: Geografische Koordinaten des Standorts.
- **Albedo**: Reflektionsgrad der Umgebung.
- **Tsmax**: Maximale Speichertemperatur in Grad Celsius.
- **East_West_collector_azimuth_angle, Collector_tilt_angle**: Azimut- und Neigungswinkel des Kollektors.

Die Parameter wie Vorwärmung, Temperaturdifferenzen in Wärmetauschern und Speichervolumen können optional angepasst werden.

44.3.1 Definition von Solarkollektoren und ihren Eigenschaften

Je nach Kollektortyp (`Flachkollektor` oder `Vakuumröhrenkollektor`) werden verschiedene Kollektoreigenschaften wie die optische Effizienz, Wärmekoeffizienten und Aperaturflächen verwendet. Beispielsweise werden für Flachkollektoren die Eigenschaften des `Vitosol 200-F XL13` verwendet:

$$\eta_0 = 0.763, \quad K_{\theta,\text{diff}} = 0.931, \quad c_1 = 1.969, \quad c_2 = 0.015$$

Für Vakuumröhrenkollektoren werden spezifische Eigenschaften wie der optische Wirkungsgrad η_0, sowie die Wärmeverluste a_1 und a_2 berücksichtigt. Diese Parameter werden verwendet, um die Kollektorleistung zu berechnen, abhängig von den Umgebungsbedingungen und der Strahlung.

44.3.2 Berechnung der Solarstrahlung

Die Funktion `Berechnung_Solarstrahlung`, die in einem separaten Skript definiert ist, wird aufgerufen, um die direkte, diffuse und reflektierte Strahlung auf die geneigte Oberfläche zu berechnen. Diese Funktion verwendet geometrische Modelle zur Bestimmung des Einfallswinkels der Sonnenstrahlen auf die Kollektorfläche und berechnet den Strahlungsfluss unter Berücksichtigung der Neigungs- und Azimutwinkel des Kollektors.

Die Rückgabe dieser Funktion umfasst:

- **GT_H_Gk**: Die Gesamtstrahlung auf der geneigten Oberfläche.
- **GbT**: Direkte Strahlung auf der geneigten Fläche.
- **GdT_H_Dk**: Diffuse Strahlung auf der geneigten Fläche.
- **K_beam**: Modifizierte Strahlungsintensität durch den Einfallswinkel.

44.3.3 Berechnung der Kollektorfeldleistung

Die Leistung des Kollektorfelds wird berechnet, indem der Wirkungsgrad des Kollektors und die auf die geneigte Fläche einfallende Strahlung verwendet werden. Die Berechnung der Leistung für die Kollektorfläche erfolgt unter Berücksichtigung von Strahlungsverlusten, Kollektoreffizienz und thermischen Verlusten:

$$P_{\text{Kollektor}} = (\eta_0 \cdot K_{\theta,\text{beam}} \cdot G_b + \eta_0 \cdot K_{\theta,\text{diff}} \cdot G_d) - c_1 \cdot (T_\text{m} - T_{\text{Luft}}) - c_2 \cdot (T_\text{m} - T_{\text{Luft}})^2$$

Dabei ist G_b die direkte Strahlung und G_d die diffuse Strahlung, während c_1 und c_2 die Wärmeverluste des Kollektors darstellen. T_m ist die mittlere Temperatur im Kollektor und T_{Luft} die Umgebungstemperatur.

44.3.4 Berechnung der Rohrleitungsverluste

Die Verluste in den Verbindungsleitungen werden unter Berücksichtigung der Rohrlänge, des Durchmessers und der Wärmedurchgangskoeffizienten berechnet. Die Formel zur Berechnung der Verluste in den erdverlegten Rohren ist wie folgt:

$$P_{\text{RVT}} = L_{\text{Rohr}} \cdot \left(\frac{2\pi \cdot D_{\text{Rohr}} \cdot K_{\text{Rohr}}}{\log\left(\frac{D_{\text{Rohr}}}{2}\right)} \right) \cdot (T_{\text{Vorlauf}} - T_{\text{Luft}})$$

44.3.5 Speicherberechnung

Das Speichervolumen und die Temperatur des Speichers beeinflussen die Menge der nutzbaren Wärmeenergie. Die gespeicherte Wärmemenge wird anhand der Wärmekapazität und der Temperaturdifferenz berechnet:

$$Q_{\text{Speicher}} = m_{\text{Speicher}} \cdot c_p \cdot \Delta T$$

wobei m_{Speicher} die Masse des Wassers im Speicher ist, c_p die spezifische Wärmekapazität von Wasser (ca. 4.18 kJ/kgK) und ΔT die Temperaturdifferenz zwischen der Vorlauf- und Rücklauftemperatur darstellt.

44.3.6 Wärmeoutput und Stagnation

Der Wärmeoutput der Solaranlage wird als Funktion der Kollektorleistung und der Speicherverluste berechnet. Falls die Speichertemperatur das zulässige Maximum erreicht, tritt Stagnation auf, und die Kollektorfeldertrag wird auf null gesetzt.

Der Gesamtwärmeoutput wird über die Simulationszeit summiert:

$$Q_{\text{output}} = \sum_{i=1}^{n} \frac{P_{\text{Kollektor},i} \cdot \Delta t}{1000}$$

Dabei ist $P_{\text{Kollektor},i}$ die Kollektorleistung zum Zeitpunkt i, und Δt die Zeitschrittweite.

44.4 Wirtschaftliche und ökologische Überlegungen

Die `SolarThermal`-Klasse enthält Methoden zur Berechnung der **Wärmegesethungskosten (WGK)**, die die Installationskosten, Betriebskosten und Förderungen gemäß BEW berücksichtigt. Die spezifischen CO_2-Emissionen des Systems werden als Emissionen pro erzeugter Wärmeeinheit berechnet, und der **Primärenergieverbrauch** wird basierend auf der Wärmeerzeugung des Systems ermittelt.

44.5 Nutzungsbeispiel

Diese Klasse ist anpassungsfähig für verschiedene solarthermische Konfigurationen. Das folgende Beispiel zeigt, wie die Klasse initialisiert und verwendet werden kann:

```
solar_system = SolarThermal(
    name="SolarThermie-Anlage",
    bruttofläche_STA=500,  # m²
    vs=50,  # m³ Speicher
    Typ="Flachkollektor",
    Tsmax=90,
    Longitude=-14.42,
    STD_Longitude=-15,
    Latitude=51.17,
    East_West_collector_azimuth_angle=0,
    Collector_tilt_angle=36
)
results = solar_system.calculate(
    VLT_L=temperature_forward,
    RLT_L=temperature_return,
    TRY=weather_data,
    time_steps=steps,
    calc1=0.8, calc2=1.2,
    q=0.03, r=0.02, T=20, BEW="Ja",
    stundensatz=50,
    duration=1,
    general_results=load_profile
)
```

In diesem Beispiel wird eine Solarthermieanlage mit Flachkollektoren auf einer Fläche von 500 m^2 und einem Speichervolumen von 50 m^3 simuliert. Die Leistungs- und Kostenkennzahlen werden basierend auf den bereitgestellten Eingabedaten berechnet.

45 Einleitung

Dieses Dokument beschreibt die Berechnung der Solarstrahlung und des photovoltaischen Ertrags basierend auf dem Skript `photovoltaics.py`. Die Berechnung nutzt Wetterdaten des Testreferenzjahrs (TRY) zur Bestimmung von Strahlungsintensität, Umgebungstemperatur und Windgeschwindigkeit. Diese Parameter werden verwendet, um die Einstrahlung auf geneigte Flächen und den photovoltaischen Ertrag zu berechnen.

Die Solarstrahlungsberechnung erfolgt auf Basis des Scenocalc Fernwärme 2.0 Modells `https://www.scfw.de/`, während die PV-Berechnung nach der Berechnungsvorschrift von PVGIS durchgeführt wird `https://joint-research-centre.ec.europa.eu/photovoltaic-geog getting-started-pvgis/pvgis-data-sources-calculation-methods_en`.

46 Berechnung der Photovoltaik-Leistung

Die Hauptfunktion `Calculate_PV` im Skript berechnet die PV-Leistung für ein gegebenes System und geographische Lage. Dabei werden spezifische Systemparameter wie die Bruttofläche, die Albedo und der Kollektorneigungswinkel berücksichtigt.

46.1 Eingangsparameter

Die Funktion `Calculate_PV` verwendet die folgenden Eingangsparameter:

- `TRY_data`: Pfad zu den TRY-Daten.
- `Gross_area`: Bruttofläche des PV-Systems in Quadratmetern.
- `Longitude`: Geographische Länge des Standorts.
- `STD_Longitude`: Standardlänge für die Zeitzone.
- `Latitude`: Geographische Breite des Standorts.
- `Albedo`: Albedo-Wert (Reflexionsfaktor der Umgebung).
- `East_West_collector_azimuth_angle`: Azimutwinkel des Kollektors in der Ost-West-Richtung.
- `Collector_tilt_angle`: Neigungswinkel des Kollektors.

46.2 Berechnung der Solarstrahlung

Die zuvor erläuterte Funktion `calculate_solar_radiation` wird verwendet, um die Solarstrahlung auf der geneigten Kollektorfläche zu berechnen. Dabei werden sowohl die direkte als auch die diffuse Strahlung sowie Reflexionen durch die Umgebung (Albedo) berücksichtigt. Die Solarstrahlung G_T wird durch folgende Gleichung bestimmt:

$$G_T = Gbhoris \cdot R_b + Gdhoris \cdot Ai \cdot R_b + Gdhoris \cdot (1 - Ai) \cdot 0.5 \cdot (1 + \cos(CTA)) + G \cdot \text{Albedo} \cdot 0.5 \cdot (1 - \cos(CTA))$$

wobei:

- $Gbhoris$: Direkte Strahlung auf die horizontale Fläche,
- $Gdhoris$: Diffuse Strahlung,
- R_b: Verhältnis der Strahlungsintensität auf der geneigten Fläche zur horizontalen Fläche,
- Ai: Atmosphärischer Diffusanteil,
- CTA: Neigungswinkel des Kollektors.

46.3 Photovoltaik-Leistungsberechnung

Die Photovoltaik-Leistung wird basierend auf der berechneten Solarstrahlung, der Bruttofläche und den spezifischen Systemparametern berechnet. Die nominale Effizienz η_{nom} des PV-Moduls wird dabei durch Temperatur- und Strahlungseinflüsse modifiziert. Die Systemverluste (typisch 14%) werden ebenfalls berücksichtigt. Die Leistungsberechnung erfolgt nach folgender Formel:

$$P_{\text{PV}} = G_T \times \text{Fläche} \times \eta_{\text{nom}} \times (1 - \text{Systemverluste})$$

Hierbei ist G_T die berechnete Strahlungsintensität in $\frac{\text{kW}}{\text{m}^2}$, und die Verluste basieren auf verschiedenen Faktoren wie Modultemperatur und Strahlungsintensität.

46.4 Berechnung der Modultemperatur

Die Modultemperatur T_m wird als Funktion der Umgebungstemperatur T_a, der Strahlungsintensität G_T und der Windgeschwindigkeit W berechnet:

$$T_m = T_a + \frac{G_T}{U_0 + U_1 \cdot W}$$

wobei U_0 und U_1 Parameter sind, die den temperaturabhängigen Leistungsverlust beschreiben.

46.5 Relative Effizienz

Die relative Effizienz η_{rel} des PV-Moduls hängt von der Strahlungsintensität G_1 und der Modultemperatur T_m ab und wird wie folgt berechnet:

$$\eta_{\text{rel}} = 1 + k_1 \ln(G_1) + k_2(\ln(G_1))^2 + k_3 T_1 m + k_4 T_1 m \ln(G_1) + k_5 T_m (\ln(G_1))^2 + k_6 T_m^2$$

Dabei sind $k_1, k_2, k_3, k_4, k_5, k_6$ Konstanten, die den Einfluss der Modultemperatur und Strahlungsintensität auf die Effizienz berücksichtigen.

47 Ergebnisse und Berechnungen für Gebäude

Die Funktion berechnet die jährliche PV-Ausbeute.

Die Ergebnisse umfassen:

- Jährliche PV-Ausbeute in kWh,
- Maximale PV-Leistung in W,
- Stündliche PV-Leistung in W

48 Zusammenfassung

Die in `photovoltaics.py` implementierte Methode berechnet die Solarstrahlung und den photovoltaischen Ertrag auf Grundlage der spezifischen Systemparameter und Wetterdaten. Durch Berücksichtigung der Einstrahlung, der Modultemperatur und der Systemverluste wird eine realistische Schätzung des jährlichen PV-Ertrags ermöglicht. Diese Methode kann auf verschiedene Standorte und Gebäudetypen angewendet werden.

49 Optimierungsfunktion für den Erzeugermix

49.1 Einleitung

Die Berechnungsfunktion `Berechnung_Erzeugermix` ermittelt die Energieerzeugung für einen vorgegebenen Mix an Technologien. Ziel ist es, die Wärmeerzeugung für ein bestimmtes Lastprofil unter Einbeziehung verschiedener Kosten-, Effizienz- und Emissionsfaktoren zu berechnen.

49.2 Mathematisches Modell

49.2.1 Eingangsparameter

Die Berechnungsfunktion nimmt eine Reihe von Eingangsparametern an, die die technologischen und ökonomischen Bedingungen beschreiben. Diese beinhalten unter anderem:

- **tech_order**: Liste der zu betrachtenden Technologien.
- **initial_data**: Tuple bestehend aus Zeitpunkten, Lastprofil, Vorlauf- und Rücklauftemperaturen.
- **Gaspreis**, **Strompreis**, **Holzpreis**: Energiekosten in €/kWh.
- **BEW**: Förderung BEW Ja/Nein.
- **kapitalzins**, **preissteigerungsrate**, **betrachtungszeitraum**: Finanzielle Parameter für die Kostenberechnung.

49.2.2 Berechnungslogik

Die Funktion berechnet zunächst die Jahreswärmebedarfe basierend auf dem Lastprofil L und der zeitlichen Auflösung:

$$\text{Jahreswärmebedarf} = \frac{\sum L}{1000} \cdot \text{duration}$$

Die Wärmebedarfsfunktion läuft über eine Schleife für jede Technologie in der `tech_order`. Je nach Art der Technologie (Solarthermie, Abwärme, Geothermie usw.) wird ein spezifisches Berechnungsmodell angewandt.

49.2.3 Technologiespezifische Berechnung

Jede Technologie verwendet unterschiedliche Berechnungsmodelle:

- **Solarthermie**: Berechnet den Ertrag basierend auf der Vorlauftemperatur und der solaren Einstrahlung aus dem Testreferenzjahr (TRY).
- **Wärmepumpen** und **Abwärme**: Verwenden den COP-Wert (*Coefficient of Performance*) und Strompreis zur Ermittlung der Betriebsaufwendungen.
- **Blockheizkraftwerke (BHKW)**: Berücksichtigen sowohl thermische als auch elektrische Leistungen, sowie den Brennstoffverbrauch.

49.3 Kapital- und Emissionskosten

Neben den Betriebskosten werden auch kapitalgebundene und emissionsbasierte Kosten berechnet. Der kapitalgebundene Kostenanteil ergibt sich aus:

$$A_{N,K} = A_0 \cdot \frac{(q-1)}{1-q^{-T}}$$

wobei $q = 1 + \text{Zinsrate}$.

Die spezifischen CO2-Emissionen werden pro erzeugte Wärmemenge berechnet:

$$\text{CO2\_Emissionen} = \frac{\sum \text{Wärmemenge}_i \cdot \text{spec\_co2}_i}{\text{Jahreswärmebedarf}}$$

49.4 Zusammenfassung

Die Funktion `Berechnung_Erzeugermix` führt eine detaillierte Berechnung der Energieerzeugung durch, indem sie mehrere Technologien gleichzeitig berücksichtigt. Die Berechnung erfolgt basierend auf stündlichen Daten für Lastprofile, Temperaturen und Emissionen.

50 Berechnungsfunktion für den Erzeugermix

50.1 Einleitung

Die Optimierungsfunktion `optimize_mix` verwendet mathematische Optimierungstechniken, um den Mix aus Energieerzeugungstechnologien zu optimieren. Das Ziel der Optimierung ist es, die Kosten, CO2-Emissionen und den Primärenergieverbrauch zu minimieren, indem verschiedene Technologien mit unterschiedlichen Parametern berücksichtigt werden.

50.2 Mathematisches Modell

50.2.1 Zielgrößen

Die Optimierung basiert auf der Minimierung einer gewichteten Summe von drei Zielgrößen:

$$\text{Ziel} = w_{\text{WGK}} \cdot \text{WGK\_Gesamt} + w_{\text{CO2}} \cdot \text{CO2\_Emissionen\_Gesamt} + w_{\text{Primärenergie}} \cdot \text{Primärenergie\_Faktor\_Gesa}$$

Hierbei sind $w_{\text{WGK}}, w_{\text{CO2}}, w_{\text{Primärenergie}}$ die Gewichte, die den Einfluss der jeweiligen Zielgröße auf das Gesamtergebnis steuern.

50.2.2 Optimierungsverfahren

Die Optimierung erfolgt mittels des `SLSQP`-Algorithmus, der für nichtlineare Probleme mit Nebenbedingungen geeignet ist. Der Algorithmus sucht nach den optimalen Parametern für die Technologien (z.B. Fläche für Solarthermie, Leistung für BHKW), die die gewichtete Summe der Zielgrößen minimieren.

50.2.3 Nebenbedingungen

Für jede Technologie werden Schranken (*bounds*) für die zu optimierenden Parameter definiert, um physikalisch sinnvolle Werte sicherzustellen. Zum Beispiel:

- Für die Fläche eines Solarthermie-Systems: min_area $\leq$ Fläche $\leq$ max_area
- Für die Leistung eines BHKW: min_Leistung $\leq$ Leistung $\leq$ max_Leistung

50.3 Ergebnis

Nach erfolgreicher Optimierung gibt die Funktion die optimierten Parameter für jede Technologie zurück. Diese Parameter minimieren die gewichtete Summe der Kosten, CO2-Emissionen und des Primärenergieverbrauchs.

50.4 Zusammenfassung

Die Funktion `optimize_mix` erlaubt eine simultane Optimierung mehrerer Technologien basierend auf benutzerdefinierten Zielgrößen. Durch die Verwendung von mathematischen Optimierungsverfahren wie `SLSQP` werden die besten Kombinationen von Technologien und Parametern ermittelt.

51 Fazit