Berechnungsmodelle und Optimierungsfunktionen für Wärmeerzeugungssysteme in DistrictHeatSim - Dokumentation

Dipl.-Ing. (FH) Jonas Pfeiffer

2024-09-10

Contents

1 Einleitung

Dieses Dokument beschreibt die mathematischen Modelle und Optimierungsfunktionen, die zur Berechnung und Bewertung verschiedener Wärmeerzeugungssysteme verwendet werden. Der Code konzentriert sich auf die Optimierung von Energieeffizienz und Wirtschaftlichkeit und wendet dabei die VDI 2067-Richtlinien für die Kostenanalyse an. Für die Berechnungen werden die Python-Bibliotheken NumPy, SciPy und Pandas verwendet. Die Berechnungsmodelle sind in Klassen organisiert, die spezifische Wärmeerzeugungssysteme repräsentieren.

2 Wirtschaftlichkeitsrechnung nach VDI 2067

2.1 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.

2.2 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:

2.2.1 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

2.2.2 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 $q = 1 + \text{Zinssatz}$.

2.2.3 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.

2.2.4 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.

2.2.5 Sonstige Kosten

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

2.2.6 Erlöse

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

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

2.2.7 Rückgabewert

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

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

2.3 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.

3 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.

3.1 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.

3.2 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:

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

3.3 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_Investitionskoster
```

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.

4 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.

4.1 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.

4.2 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.

4.3 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.

4.4 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 m² und einer Bohrtiefe von 150 m simuliert. Die Leistung und die ökonomischen Kennzahlen des Systems werden anhand der eingegebenen Daten berechnet.

5 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.

5.1 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.

5.2 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.

5.3 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.

5.4 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.

6 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.

6.1 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.

6.2 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.

6.3 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.

6.4 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.

7 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.

7.1 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.

7.2 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.

7.3 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.

7.4 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.

8 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.

8.1 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.

8.2 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:

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

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

8.3 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.

8.4 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 m^3 simuliert. Die ökonomische und ökologische Leistung des Systems wird anhand der bereitgestellten Eingaben berechnet.

9 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.

9.1 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 €/m^3.
- `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.

9.2 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.

9.3 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.

9.4 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 m³ simuliert. Das System enthält einen 100 m³ großen Wärmespeicher. Leistung und Kosten werden basierend auf den eingegebenen Parametern berechnet.

10 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.

10.1 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.

10.2 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.

10.2.1 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:

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

10.2.2 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_Investitionskosten`) 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.

10.2.3 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.

10.3 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.

10.4 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.

11 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.

11.1 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 €/m².
- `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).

11.2 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 Rücklauftemperaturen, 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.

11.3 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.

11.3.1 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.

11.3.2 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.

11.3.3 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.

11.3.4 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:

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

11.3.5 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.

11.3.6 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.

11.4 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.

11.5 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 m² und einem Speichervolumen von 50 m³ simuliert. Die Leistungs- und Kostenkennzahlen werden basierend auf den bereitgestellten Eingabedaten berechnet.

12 Optimierungsfunktion für den Erzeugermix

12.1 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.

12.2 Mathematisches Modell

12.2.1 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.

12.2.2 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.

12.2.3 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.

12.3 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:

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

12.4 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.

13 Berechnungsfunktion für den Erzeugermix

13.1 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.

13.2 Mathematisches Modell

13.2.1 Zielgrößen

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

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

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

13.2.2 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.

13.2.3 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

13.3 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.

13.4 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.

14 Fazit

Dieses Dokument bietet einen umfassenden Überblick über die Modelle und Optimierungsstrategien, die für Wärmeerzeugungssysteme verwendet werden. Zukünftige Arbeiten werden das System erweitern, um die Systeme genau zu modellieren und die Optimierungsfunktionen zu verbessern.