Dokumentation zur Anwendung des Energiesystemmodells in Stadt-Land-Energie (2025), **1**, 1–8

REINER LEMOINE INSTITUT

Dokumentation zur Anwendung des Energiesystemmodells in Stadt-Land-Energie

Inhaltsverzeichnis

1. ESM-Guide zur Berechnung robuster, regionaler Energiesystemmodelle

Das vorliegende Material dient als begleitende Unterstützung für die Modellierung eines Stadt-Land-Nexus im Rahmen des Projektes „Stadt-Land-Energie“ . Der Best-Practice-Guide ermöglicht es potenziellen Anwender:innen den Aufbau der untersuchten Modelle in den Stadt-Land-Case-Studies zu verstehen und dieses Wissen in die Adaption des Energiesystemmodells für eine spezifische Region einzubringen.

Die im Szenario-Explorer dargestellten Ergebnisse basieren auf den Berechnungen eines Energiesystemmodells. Je nach Case-Study werden regionsspezifische Systemgrenzen sowie technische und ökonomische Rahmenbedingenen identifiziert und in das Modell implementiert.

Das Open-Source-Energiesystemmodell **oemof-B3** bildet die Grundlage für das Modell. Das Multi-Knoten-Modell ermöglicht es, unterschiedliche Regionen als Knoten abzubilden und diese mit Übertragungsleitungen zu verknüpfen. Abbildung 1 zeigt die konzeptionelle Struktur des Modells. Strom und Wärme sind als Bedarf definiert, die über Busse mit verschiedenen Energieträgern verbunden sind. Die Energieträger stehen im Austausch mit Umwandlungstechnologien und Energiequellen, wodurch das Modell den Energiebedarf auf unterschiedliche Weise decken kann.

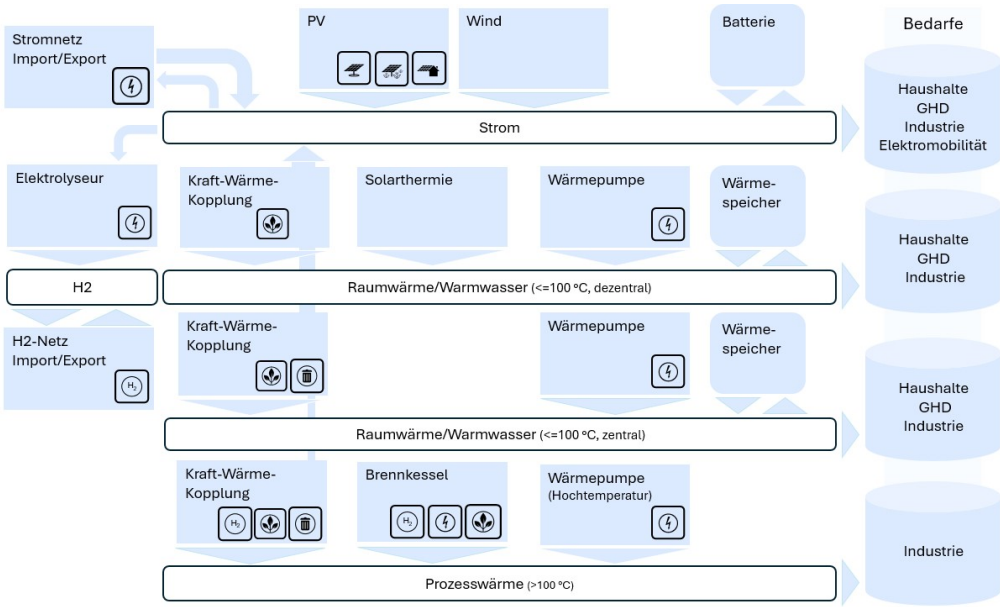

Abbildung 1. Konzeptbild des Energiesystemmodells für die Case Study Berlin-Brandenburg

Für die Optimierung des Energiesystemmodells der Case Studies wird die Pipeline für Webapps **apipe** angewendet. Die apipe ist eine vom Reiner Lemoine Institut entwickelte Open-Source-Datenpipeline für Webanwendungen. Sie unterliegt der AGPL-3.0-Lizenz, wobei der Code in einem öffentlichen Repository auf Github [1] verfügbar ist. Das Modell muss zunächst lokal aufgesetzt werden, bevor die Modelldaten für eigene Zwecke oder spezifische Fragestellungen angepasst werden können.

[1] J. Amme, M.-C. Gering, A. Schilling, D. Starzl, and H. Bartels, äpipe,Äug. 26, 2025. [Online]. Available: https://github.com/rl-institut/apipe

1.1 Installation und Setup

Für die Installation des Modells sind einige Voraussetzungen zu berücksichtigen. Derzeit ist die Anwendung ausschließlich unter dem Betriebssystem Linux möglich. Für das Clonen des Projekts wird ein GitHub-Konto benötigt und die Installation von Anaconda ist erforderlich. Für die Ausführung der Optimierung wird eine Python-Version 3.10 benötigt.

Das Projekt kann mit der Schritt-für-Schritt-Anleitung im Installationsguide auf dem Repository apipe aufgesetzt werden. Im Anschluss der Installation muss auf den Branch **sle-features/case-study-1** navigiert werden. Hierfür ist folgender Befehl auszuführen:

—-
conda checkout sle-features/case-study-1
—-

Auf der Open Energy Platform können die Eingangsdaten für die Case Study 1 als oemof-tabular Datapackage heruntergeladen werden. Die Datensätze, welche die Rohdaten enthalten, können auf Anfrage vom Reiner Lemoine Institut als .zip- Datei zur Verfügung gestellt werden. Diese müssen im Anschluss entpackt und in die jeweiligen Dateipfade abgelegt werden.

—-
apipe / store / raw
—-

—-
pipe / store / datasets
—-

Im Projekt wird mit de Workflow Snakemake gearbeitet. Das Ausführen von Prozessschritten erfolgt mit Snakemake-Befehlen im folgenden Format:

—-
snakemake -j1 make_esys_appdata
—-

Der Ausdruck **-j1** gibt die Anzahl der CPU-Kerne an, die für die Pipeline-Ausführung verwendet werden sollen. Für die Modellierung im Projekt ist in der Regel ein Kern als Standardwert festgelegt. Anwender:innen ist es jedoch freigegeben die Anzahl der CPU-Kerne selbst zu wählen. Der Befehl **make_esys_appdata** überprüft anhand eines Testlaufs, ob die Datensätze im richtigen Dateipfad abgelegt sind.

Für den Aufbau eines Energiesystemmodells werden Informationen über die Technologien und das Energiesystem in Form von **scalars** und Zeitreihen **time series** benötigt. Diese können mithilfe der folgenden Befehle als leere .csv-Dateien erstellt werden. Die Dateien *empty_scalars.csv* und *empty_time_series.csv* geben den Anwender:innen eine Übersicht über die Komponenten und Parameter, die sich im Energiesystemmodell befinden und entsprechend angepasst werden können.

—-
snakemake -j1 create_empty_scalars
—-

—-
snakemake -j1 create_empty_ts

—-

Falls das manuelle Ausfüllen der **scalars** umgangen werden soll, kann mit dem Befehl **write_default_scalars** eine automatich ausgefüllte *default_scalars.csv* Datei erstellt werden. In diesem Fall wird für jeden Parameter der Wert entweder auf 0 oder NaN gesetzt.

—-
snakemake -j1 write_default_scalars
—-

Das Lösen des Energiesystemmodells mit dem Befehl:
—-
snakemake -j1 postprocessed_esys_appdata
—-

Sollte das Energiesystemmodell auf Basis des oemof-tabular Datapckages aus gestartet werden, muss die Flag *–rerun-incomplete* hinter den Befehl gesetzt werden, das verhindert, dass die Pipeline die vorangestellten Prozessierungsschritte abfragt und ausführt:

—-
snakemake -j1 postprocessed_esys_appdata –rerun-incomplete
—-

Die generierten Ergebnisse können mit folgenden Befehl aus dem Ergebnisordner entfernt werden. Dieser Befehl erweist sich als nützliche Funktion für Anwender:innen, die das Modell debuggen.

—-
snakemake -j1 clean
—-

Die Dokumentation für das Aufsetzen des Energiesystemmodells und Antworten zu den meist gestellten Fragen, kann hier abgerufen werden. Näheres zu den Befehlen mit Snakemake und das Abrufen von spezifischen Datensets können im Workflow nachgelesen werden.

2. Modellanpassungen

Anwender:innen können für ihre eigenen Projekte Daten des Modells anpassen. Diese teilen sich in regionsspezifische sowie nicht-regionsspezifische Daten und Einspeise- und Verbrauchszeitreihen auf. Für das Verständnis werden im folgenden Abschnitt die jeweiligen Kategorien vorgestellt und die darin enthaltenen Datenpunkte erläutert.

2.1 Nicht-regionsspezifische Daten

Die Nicht-regionsspezifischen Daten beinhalten die Informationen, die unabhängig von den untersuchten Regionen in die Modellierung einfließen. Diese sind in der Datei *raw_costs_efficiencies.csv* zusammengefasst und beinhalten technische Daten zu den Technologien des Energiesystems und Informationen zu Emissionswerten. Jeder Parameter steht für ein bestimmtes Merkmal oder eine Kennzahl innerhalb der Datenstruktur. Die Spalte **var_name** gibt das Merkmal der Komponente an. Eine Übersicht zu allen Merkmalen mit Kurzbeschreibung ist in Tabelle 2 dargelegt.

Tabelle 2. Parameterübersicht in raw_costs_efficiencies.csv

var_name	Bezeichnung
capacity_cost_overnight	Investitionskosten
carrier_cost	Energieträgerkosten
fixom_cost	Fixe Kosten
marginal_cost	Grenzkosten
storage_capacity_cost_overnight	Investitionskosten des Speichers
storage_fixom_cost	Fixe Kosten des Speichers
wacc	Weighted Average Cost of Capital
expandable	Ausbaufähigkeit als boolean
lifetime	Lebensdauer
condensing_efficiency	Kondensationswirkungsgrad
effciency	Wirkungsgrad
electric_efficiency	elektrischer Wirkungsgrad
thermal_efficiency	thermischer Wirkungsgrad
loss_rate	Verlustrate
emissions_1990	Emissionen im Jahr 1990
emission_reduction_factor	Emissionsreduktionsfaktor
emissions_not_modeled	nicht modellierte Emissionen

Das Format der Daten muss für die Modellierung beibehalten werden, um die Parameter der Technolgien zuzuordnen. In Tabelle 3 wird für eine Kraft-Wärme-Kopplungsanlage der Eintrag aus raw_costs _efficiencies.csv exemplarisch dargestellt. Im gegebenen Beispiel handelt es sich um eine Gegendruck-KWK im Hochtemperaturbereich für ein zukünftiges Szenario in 2045. Für die Komponente sind die Investitionskosten (capacity_cost_overnight) in EUR/_e spezifiziert. Da die Technologiedaten unabhängig der Region gültig sind, wird unter dem Parameter **name** keinen Wert eingetragen.

Eine ausführliche Beschreibung der Parameter und deren zugehöriges Daten-Format ist in der Dokumentation unter Scalars zu finden.

Tabelle 3. Beispiel aus raw_costs_efficiencies.csv

Parameter	Wert
id_scal	51
scenario_key	2045_scenario
name	
var_name	capacity_cost_overnight
carrier	biomass_solid
region	ALL
tech	extchp_heat_high
type	extraction
var_value	3390.8967
var_unit	EUR/kW_e
source	PyPSA: Danish Energy Agency, technology_data_for_el_and_dh.xlsx
comment	09a Wood Chips, Large 50 degree: Nominal investment

Die *technology_data.json*, beinhaltet Angaben zu den Volllaststunden, Leistungsdichten und Speicherdimensionierung für Erneuerbare Energien und Speicher. Eine Übersicht zu den Datensätzen und Einheiten sind in Tabelle 4 abgebildet. Zusätzliche Informationen und Annahmen zur Case Study Oderland-Spree, können aus den Technologiedaten entnommen werden.

Tabelle 4. Übersicht der technology_data.json

Parameter	Bezeichnung	Einheit
`full_load_hours`	Jährliche Volllaststunden verschiedener Technologien	h/a
`power_density`	Leistungsdichte pro Flächeneinheit (z. B. PV, Wind)	MW/km^2
`nominal_power_per_unit`	Nennleistung einer einzelnen Erzeugungseinheit	MW
batteries		
`nominal_power_per_storage_capacity`	Verhältnis von Lade-/Entladeleistung zur Speicherkapazität (C-Rate)	1/h
`storage_capacity_per_pv_power`	Verhältnis von Speicherkapazität zur installierten PV-Leistung	MWh/MWp
hot_water_storages		
`nominal_power_per_storage_capacity`	Verhältnis Lade-/Entladeleistung zu Speicherkapazität von Wärmespeichern	1/h

2.2 Regionsspezifische Daten

Für die individuellen Regionen unterscheiden sich die Energiebedarfe, die Lastprofile, die Potenziale für den Ausbau von Erneuerbarer Energien und die Commodities. Die Anpassung dieser Datensätze können von den Anwender:innen in der Datei *region_specific_scalars.csv* vorgenommen werden. Strom- und Wärmebedarf sind in der .csv-Datei unter den Einträgen **var_name:amount**, Lastprofile unter **var_name:profile** und Ausbaupotenziale von Erneuerbaren Energien unter **var_name:capacity_potential** aufzufinden. Die als „Commodities“ bezeichneten Ressourcen einer Region umfassen die verfügbaren Kapazitäten unterschiedlicher Energieträger. Im Parameter **name** werden die Datensätze einer Region und Technologie zugeordnet. Eine Übersicht zu den Datensätzen mit ihren zugehörigen Datenpunkten ist in Tabelle 5 aufgelistet.

In der Dokumentation "Regionsspezifischer Datensatz Case Study: Oderland_Spree" sind die Datenpunkte für die Case Study 1 mit ihren Quellen und weiterführend die Berechnungen und Annahmen hinterlegt.

2.3 Einspeise- und Verbrauchszeitreihen

Die Zeitreihen, welche die Einspeisung und den Verbrauch abbilden, sind in der *region_specific_scalars.csv* unter dem Namen **var_name:profile** zu identifizieren. Dazu gehören zum einen die normierten Einspeisezeitreihen der Erneuerbaren Energien. Im Datensatz befinden sich die Zeitreihen für Wind, Dach-PV, Freiflächen-PV sowie Agri-PV und Solarthermie. Des Weiteren werden im Modell normierte COP-Zeitreihen für Wärmepumpen implementiert. Die Wärmepumpen differenzieren sich in ihrer Anwendung im Energiesystem. Dazu gehören Hochtemperatur- und Großwärmepumpen sowie dezentrale Wärmepumpen

3. Robustheit

Für die Durchführung von Robustheitsuntersuchungen stellen wir ein Add-In zur Verfügung, das eine effiziente warmgestarte Reoptimierungsstrategie in Bezug auf Perturbationen der Kosten oder Nachfrage durchführt. Das Python-Skript ist in oemof-B3 und apipe integriert, kann aber auch mit wenigen Änderungen unabhängig davon verwendet werden.

Tabelle 5. Übersicht der Datenpunkte in `region_specific_scalars.csv`

Kategorie	Parameter
Energiebedarfe	Gesamtstrombedarf, normiert
	Gesamtwärmebedarf Niedrigtemperatur zentral
	Gesamtwärmebedarf Niedrigtemperatur dezentral
	Gesamtwärmebedarf Prozess
Lastprofile	Stromlastprofile, normiert
	Wärmelastprofile Niedrigtemperatur zentral und dezentral
	Wärmelastprofile Prozess
Ausbaupotenziale	Wärmepumpen
	KWK
	Brennkessel
	Speicher
	Elektrolyseur
Feste Kapazitäten	KWK
	Speicher
	Export
	Commodities

Das Robustheits-Add-In gliedert sich in drei Komponenten:

- **master.py**: Hauptskript zur Reoptimierung,
- **CostDemand_JSONFile**: Datei im JSON-Format, in der die Rahmenbedingungen für Perturbationen von Kosten oder Nachfrage notiert werden,
- **_models.py**: Modifikation von oemof-solph um Low-Level-Warmstarts zu ermöglichen.

Die Warmstart-Features werden durch persistente Schnittstellen zu den LP/MIP-Lösern Gurobi (kommerziell) und SCIP (open source) realisiert. Im Git-Repository befinden sich Beispieldateien und eine Erklärung des Installationsprozesses in der Readme-Datei.