

In [43]:

```
import pypsa
from FLUCCoplus import plots
import matplotlib.pyplot as plt
import xarray as xr
import numpy as np
import cartopy.crs as ccrs
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Rectangle
from datetime import date, datetime, timedelta
import datetime as dt
from scipy.fft import fft, fftfreq
import math
import operator
import os.path as osp
n = pypsa.Network('elec_s_10_ec_lcopt_Co2L-1H.nc')
```

Organisation

Ordnerstruktur

environment

Gurobi-Solver

Webseite-Anmeldung

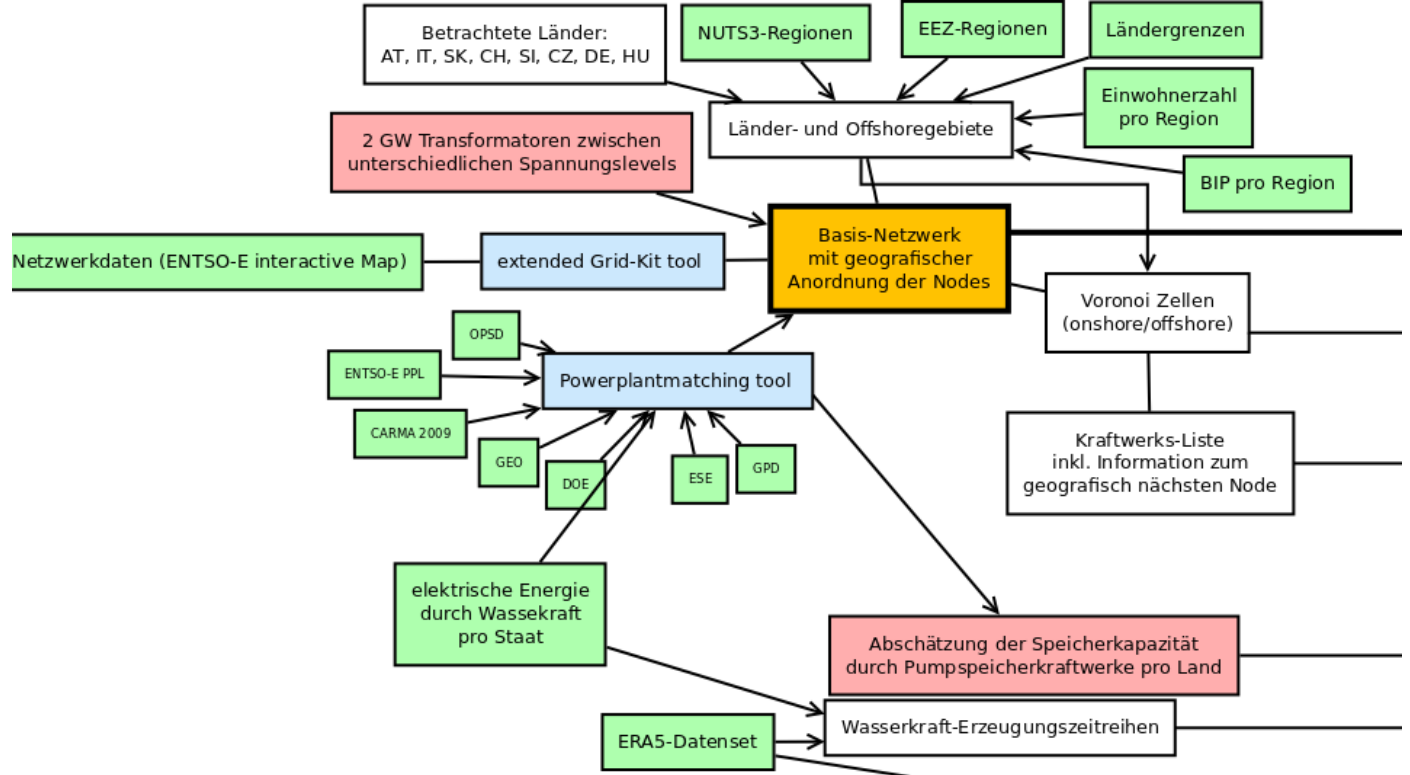
Arbeitsschritte des Workflows

Der gesamte Workflow vom Erstellen des Netzwerks bis zur Auswertung der Residuallast und der Beantwortung der Fragestellung wird in 5 Schritte geteilt:

- Erstellen des Basis-Netzwerks
- Erstellen der Verfügbarkeits- und Erzeugungsprofile
- Zusammenfügen des Netzwerks nach Ausbauszenario
- numerisches Lösen des Netzwerks
- Auswertung der Lösung

Erstellen des Basisnetzwerks

Ergebnis des Schrittes: Basis-Netzwerk mit geografischer Anordnung der Nodes



Erster Schritt des Workflows. Grüne Einträge stellen Inputs dar, deren Quellen in den nächsten Absätzen aufgelistet sind. Blaue Einträge kennzeichnen externe Tools, die für den Workflow verwendet werden und auf die inhaltlich nicht genauer eingegangen wird.

In diesem ersten Schritt werden Daten aus unterschiedlichen Quellen zusammengetragen, um die Netzstruktur, wie sie heutzutage besteht, abzubilden. Dabei werden

- geographische Länderdaten
- Netzwerkdaten
- Daten über bestehende konventionelle Kraftwerke

verarbeitet.

Geografische Daten

- Betrachtete Länder: werden in der config-datei des Workflows festgesetzt: Österreich, Italien, Slowakei, Schweiz, Slowenien, Tschechien, Deutschland, Ungarn
- Ländergrenzen: werden mit dem Skript `retrieve_databundle` heruntergeladen und im Skript `build_shapes` das erste Mal verarbeitet; Quelle: <https://www.naturalearthdata.com/downloads/10m-cultural-vectors/10m-admin-0-countries/>
- Ausschließliche Wirtschaftszonen für Deutschland und Italien (EEZ-Regionen): werden mit dem Skript `retrieve_databundle` heruntergeladen und im Skript `build_shapes` das erste Mal verarbeitet; Quelle: <http://www.marineregions.org/sources.php#unioneezcountry>
- NUTS3-Regionen der betrachteten Länder: werden mit dem Skript `retrieve_databundle` heruntergeladen und im Skript `build_shapes` das erste Mal verarbeitet; Quelle: <https://ec.europa.eu/eurostat/web/gisco/geodata/reference-data/administrative-units-statistical-units>
- Einwohnerzahl pro NUTS3-Region: werden mit dem Skript `retrieve_databundle` heruntergeladen; Quelle: http://appsso.eurostat.ec.europa.eu/nui/show.do?dataset=nama_10r_3popgdp&lang=en
- BIP pro NUTS3-Region: werden mit dem Skript `retrieve_databundle` heruntergeladen; Quellen: http://appsso.eurostat.ec.europa.eu/nui/show.do?dataset=nama_10r_3gdp&lang= , <https://www.bfs.admin.ch/bfs/en/home/news/whats-new.assetdetail.7786557.html>

Damit werden die betrachteten Staaten geografisch eingegrenzt, in NUTS3-Regionen geteilt und Informationen wie das BIP pro Kopf und die Einwohnerzahl zu den entsprechenden NUTS3-Regionen gespeichert.

Netzwerkdaten

Informationen über Netzwerktopologie werden der ENTSO-E Interactive Map entnommen. Diese stellt das elektrische Netz vom Stand Jänner 2020 dar. Diese OpenStreetMap-Daten werden mit dem externen Tool GridKit in ein elektrisches PyPSA-Netzwerk umgewandelt. Der Code des GridKit-Toolkits ist einsehbar: <https://github.com/pypsa/gridkit>.

Quelle der Netzwerkdaten: <https://www.entsoe.eu/data/map/> Die Daten werden im Skript `base_network` zu einem PyPSA-Netzwerk verarbeitet. Da in dieser Quelle keine Informationen über Transformatoren enthalten sind werden zwischen unterschiedlichen Spannungsebenen Transformatoren mit einer Kapazität von 2GW ins Netzwerk integriert. Durch diese hohe Einschätzung der Kapazitäten werden "künstliche" Einschränkungen vermieden und das Netzwerk somit numerisch einfacher lösbar.

Daten über bestehende konventionelle Kraftwerke

Um eine wahrheitsgetreue Abbildung konventioneller Kraftwerke zu erreichen werden 6 öffentlich zugängliche Datenbanken kombiniert. Als Basis dient die Transparency-Plattform der ENTSO-E. Bestehende Wasserkraftwerke sind durch diesen Prozess ebenfalls abgebildet und sind zwecks Übersicht in dem Workflow durch einen extra Punkt gekennzeichnet.

- ENTSO-E PPL: ENTSO-E Transparency Platform, Installed Capacity Per ProductionUnit, <https://transparency.entsoe.eu/generation/r2/installedCapacityPerProductionUnit/show> (2017).
- CARMA: K. Ummel, Carma revisited: An updated database of carbon dioxide emissions from power plants worldwide, SSRN Electronic Journal (304). doi:10.2139/ssrn.2226505.
- GEO: Global Energy Observatory, Google, KTH Royal Institute of Technology in Stockholm, Enipedia, World Resources Institute, 2018: Global Power Plant Database. online <https://datasets.wri.org/dataset/globalpowerplantdatabase>
- DOE: Sandia National Laboratories, 2020: DOE OE Global Energy Storage Database, online <https://www.sandia.gov/ess-ssl/global-energy-storage-database-home/>
- GPD: World Resource Institute, Climate Data and Tools Project, Global Power Plant Database 1.1.0, <http://datasets.wri.org/dataset/globalpowerplantdatabase> (2018).

Die Datenbanken werden mit dem Powerplantmatching-Tool zusammengeführt und beispielsweise auf ähnliche oder doppelte Einträge überprüft. Der Code dafür ist veröffentlicht:

<https://github.com/FRESNA/powerplantmatching> Für Deutschland, Italien und Ungarn bestehen offizielle Auflistungen aktueller Kraftwerke, die verwendet werden. Quelle für diese Länder: OPSD: https://data.open-power-system-data.org/conventional_power_plants/2017-07-07

Die Kraftwerksdaten konventioneller Kraftwerke werden in dem Skript `build_powerplants` erstellt, als `.csv-Datei` abgespeichert und im dritten Schritt des Workflows weiter bearbeitet. Informationen über Wasserkraftwerke werden im Skript `build_hydro_profiles` extra behandelt was im zweiten Schritt des Workflows näher behandelt wird.



Ergebnis des Powerplantmatching-prozesses **Ergebnis des Powerplantmatching-Prozesses:** bestehende konventionelle Kraftwerke und Wasserkraftwerke, die aus unterschiedlichen Quellen zusammengetragen wurden und in das Netzwerk integriert werden. Sie können durch Aufrufen des externen Powerplantmatching-Tools grafisch aufbereitet werden.

Erstellen der Verfügbarkeits- und Erzeugungsprofile

Ergebnis des Schrittes: geografisch zuordenbare Verfügbarkeitsprofile für Wind- und Solarenergie und Erzeugungsprofile bestehender Wasserkraftwerke für ein Jahr mit einer zeitlichen Genauigkeit von einer Stunde.



Zweiter Schritt des Workflows: Die Voronoi-Zellen werden auf Basis des vorhandenen Netzwerks gebildet und dienen als geografische Ausgangswerte zur Einschätzung der Verfügbarkeit von Solar- und Windenergie. Grüne Einträge stellen Inputs dar. Rote Einträge sind Schritte, deren mathematischer bzw. programmiertechnischer Hintergrund für die weiteren Schritte besonders von Interesse ist und auf die deshalb genauer eingegangen wird.

Im zweiten Schritt werden auf Basis von Wetterdaten und geografischen Daten Zeitreihen für die Erzeugung elektrischer Energie erstellt. Dabei werden

- Solarenergie
- Windenergie
- Wasserkraft

getrennt behandelt. Für Solar- und Windenergie wird dabei von der maximal installierbaren Kapazität ausgegangen, die Erzeugungsprofile der Wasserkraft werden auf Basis der vorhandenen Kraftwerke erstellt.

Um die verfügbaren Potentiale geografisch zu einem Netzknotenpunkt zuteilen zu können werden durch das Skript `build_bus_regions` Voronoi-Zellen gebildet. In einer Voronoi-Zelle befinden sich dabei alle Punkte, die dem Netzknotenpunkt in dieser Zelle geografisch am nächsten sind. Die Fläche Österreichs wird dabei in 33 Zellen geteilt.

Verfügbarkeitsprofile für Solarenergie

Die maximal installierbare Kapazität an Solarenergie für eine Einstrahlungsdaten-Zelle wird durch

$$G_x^{max} = 0.01 * 145 \text{ MW/km}^2 * A_x$$

berechnet. Das technische Potential wird mit 145 MW/km² abgeschätzt. Bei dieser Abschätzung wird von einer Flächennutzung von 100% ausgegangen, welche durch den Faktor 0.01 auf 1% Flächennutzung durch Solarenergie beschränkt wird. Die Fläche A_x wird durch die Auswertung des CORINE Land Cover Datensets und des NATURA2000 Datensets bestimmt. Dabei kommen für die Nutzung durch Solarenergie Flächen, die mit "Künstlich angelegte nicht landwirtschaftlich genutzte Flächen", "Heterogene landwirtschaftliche Flächen" (abgesehen von Waldflächen, welche in einer genaueren Unterteilung der Flächennutzungen bestimmt sind), "Offene Flächen ohne oder mit geringer Vegetation", "Fels" und "Fels mit spärlicher Vegetation" gekennzeichnet sind, infrage. Das CORINE Landcover Datenset kann für Österreich unter <https://secure.umweltbundesamt.at/webgis-portal/corine/map.xhtml> grafisch aufbereitet betrachtet werden.

Die Wetterdatensets, die für die Berechnung der Kapazität für Solar- und Windenergie verwendet werden, haben eine geografische Auflösung von 30 km x 30 km und sind damit feiner aufgelöst als die Voronoi-Zellen. Die maximalen Kapazitäten pro Voronoi-Zellen werden deshalb für eine Voronoi-Zelle V mit

$$G_{V,x}^{max} = \mathcal{I}_{V,x} * c_x * G_x^{max}$$

Formel mit Lisa nochmal durchgehen

bestimmt, wobei die Indikatormatrix \mathcal{I} durch $\mathcal{I}_{V,x} = \text{area}(V \cap x) / \text{area}(x)$ bestimmt ist, mit $\mathcal{I}_{ij} \in (0, 1]$, und c_x die maximale Kapazität einer Zelle des Wetterdatensets ist.

Die maximal verfügbare Leistung an Solarenergie pro Voronoi-Zelle und Stunde wird mit der maximal installierbaren Leistung und Auswertung des SARA2-Datensets bestimmt. Referenzmodell ist eine kristalline Silizium-Solarzelle mit einem Aufstellwinkel von 35 deg. und südlicher Ausrichtung. Die technischen Daten der Zellen sind auf der atlite-Seite aufgelistet: <https://github.com/PyPSA/atlite/tree/master/atlite/resources/solarpanel>

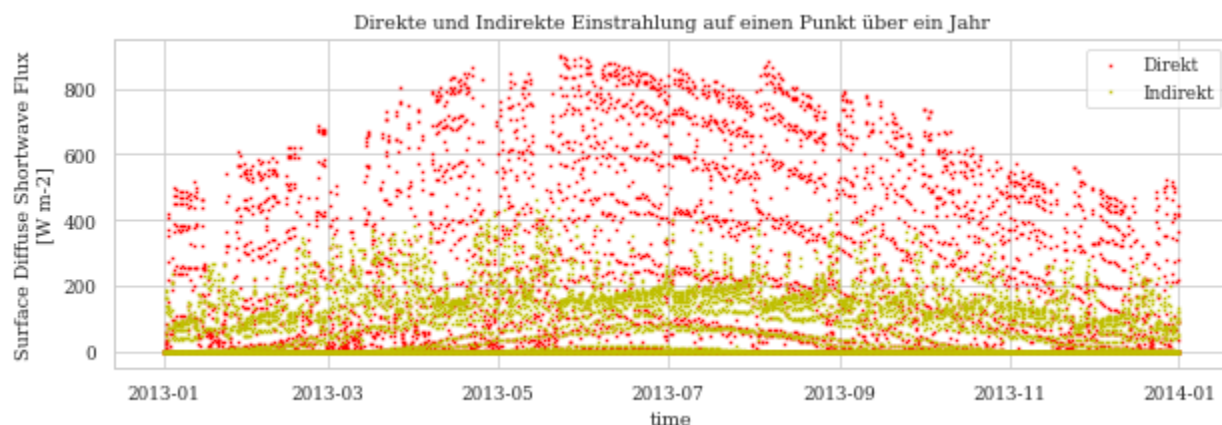
Das SARA2-Datenset enthält Informationen über solare Einstrahlung auf die Erdoberfläche. Die Daten stammen dabei von den beiden METEOSAT-Satelliten und decken in etwa den geografischen Bereich von Europa und Afrika ab. Es werden die Variablen, welche die direkte und indirekte Einstrahlung, sowie die Temperatur und die Albedo beschreiben, betrachtet. Die Energie direkter sowie indirekter Einstrahlung sind beispielhaft in der folgenden Grafik abgebildet.

In [44]:

```
sarah = xr.open_dataset('/home/max/Dokumente/FH/Master_thesis/NEW_pypsa-eur/pypsa-eur/cutout.nc')
sp=sarah.isel(x=40,y=8)
plt.figure(figsize=(10,3))
sp.plot.scatter(y = 'influx_direct', x = 'time', s = 2, marker = 'x', color = 'r', label='Direkt')
sp.plot.scatter(y = 'influx_diffuse', x = 'time', s = 2, marker = 'x', color = 'y', label='Indirekt')
plt.legend()
plt.title('Direkte und Indirekte Einstrahlung auf einen Punkt über ein Jahr')
```

Out[44]:

Text(0.5, 1.0, 'Direkte und Indirekte Einstrahlung auf einen Punkt über ein Jahr')



Verfügbarkeitsprofile für Windenergie

Die maximal installierbare Kapazität an Windenergie für eine Winddaten-Zelle wird durch

$$G_x^{max} = 0.3 * 10 \text{ MW/km}^2 * A_x$$

berechnet. Dabei ist das technische Potential mit 10 MW/km² abgeschätzt (Quelle: Y. Scholz, Renewable energy based electricity supply at low costs - Development of the REMix model and application for Europe, Ph.D. thesis, Universität Stuttgart (2012). doi:10.18419/opus-2015 .) Der Faktor 0.3, ergibt sich aus dem Miteinbeziehen gesellschaftlicher Akzeptanz und competing Flächennutzung. Die für die Windenergie nutzbare Fläche A_x einer Voronoi-Zelle x ergibt sich aus der Auswertung des CORINE Land Cover Datensets und des NATURA2000 Datensets. Flächen, die im CORINE Datenset als "Landwirtschaft" bzw. "Wälder und naturnahe Flächen" eingetragen sind werden und ein Abstand von mindestens 1 km zu "Städtisch geprägten Flächen" und "Industrie-, Gewerbe- und Verkehrsflächen" dafür in Betracht gezogen. Flächen, die als NATURA2000-Gebiet eingetragen sind werden exkludiert.

Die maximal verfügbare Leistung an Windenergie pro Voronoi-Zelle pro Stunde wird auf Basis der maximalen Kapazität berechnet. Es wird die Leistungskurve der Vestas V112-Turbine mit einer Kapazität von 3MW und Narbenhöhe von 80m verwendet. Die Wetterdaten des ERA5-Datensets enthalten die Windstärke 100 m über dem Boden. Die Daten werden mit dem Logarithmus-Gesetz

$$u^{h1}(t) = \frac{u^{h2} \ln(h/z^0)}{\ln(\frac{h^{h2}}{z^0})}$$

extrapoliert, um die Windgeschwindigkeit in Narbenhöhe zu bekommen. Die technischen Daten der verwendeten Windturbine sind auf der Atlite-Seite aufgelistet:

<https://github.com/PyPSA/atlite/tree/master/atlite/resources/windturbine>

Quellen:

- SARAH2 Radiation Data Set: Pfeifroth, Uwe; Kothe, Steffen; Müller, Richard; Trentmann, Jörg; Hollmann, Rainer; Fuchs, Petra; Werscheck, Martin (2017): Surface Radiation Data Set - Heliosat (SARAH) - Edition 2, Satellite Application Facility on Climate Monitoring, DOI:10.5676/EUM_SAF_CM/SARAH/V002, https://doi.org/10.5676/EUM_SAF_CM/SARAH/V002.
- Capacity factors durch Weather Data Sets: ERA5-weather-data: Copernicus, European Commission, European Center for Medium Weather Forecast (ECMWF), 2019: ERA5-Land hourly data from 1981 to present, doi:10.24381/cds.e2161bac, downloaded from <https://cds.climate.copernicus.eu/cdsapp#!/dataset/reanalysis-era5-land?tab=overview>
- EEA, Corine Land Cover (CLC) 2012, version 18.5.1, <https://land.copernicus.eu/pan-european/corine-land-cover/clc-2012> (2012).
- EEA, Natura 2000 data - the European network of protected sites, <http://www.eea.europa.eu/data-and-maps/data/natura-7> (2016).

![[aitle]](Voronoi_cells.png)

Erzeugungsprofile der Wasserkraft

Erzeugungsprofile für die Erzeugung mittels Wasserkraftwerken werden auf eine andere Weise erstellt wie die Verfügbarkeitsprofile von Wind- und Solarenergie, da in dem Netzwerkmodell kein Ausbau der Wasserkraftkapazitäten betrachtet wird. Die bestehenden Kapazitäten an Wasserkraftwerken sind durch Ausführung des Skripts `build_powerplants`, die im ersten Schritt des Workflows durchgeführt wird, bekannt. Kapazitäten werden eingeteilt in Laufkraftwerke, Speicherwasser-Kraftwerke und Pumpspeicherkraftwerke. **Die verfügbaren Kapazitäten der Wasserkraft müssen in einem file des Workflows angepasst werden, sodass die Menge an elektrischer Energie, laut dem entsprechenden Szenario erstellt werden kann.** Die Erzeugungsprofile werden im Skript `build_hydro_profile` erstellt. Für die Erstellung der Zeitprofile werden die "Runoff"-Daten des ERA5 Datensets verwendet. Diese beinhalten alle Wassermengen, die beispielsweise durch Regen oder Schneeschmelze auf die Erde treffen und damit zur Stromerzeugung mittels Wasserkraftwerken beitragen. Die resultierenden Wassermengen werden nach Höhenmeter gewichtet und das Datenset normalisiert, um über den Zeitraum von einem Jahr die bekannte Menge elektrischer Energie zu erzeugen [Brown et al, 2017].

Abschätzung der Speicherkapazitäten von Pumpspeicherkraftwerken

Da keine öffentlich zugänglichen Daten zu den Kapazitäten bestehender Pumpspeicherkraftwerke verfügbar sind, werden diese auf Basis der gesamten Speicherkapazitäten eines Landes abgeschätzt und auf die verfügbaren Speicherwasser-Kraftwerke und Pumpspeicherkraftwerke aufgeteilt. Die geografische Aufteilung der Kapazitäten ist für das Projekt vernachlässigbar, da sie in einem späteren Schritt wieder zu einem Knotenpunkt zusammengefügt werden.

Quellen:

- D. Schlachtberger, T. Brown, S. Schramm, M. Greiner, The benefits of cooperation in a highly renewable european electricity network, Energy 134 (Supplement C) (2017) 469 – 481.
doi:10.1016/j.energy.2017.06.004
- Übersetzung Speicherwasserkraftwerk: https://austria-forum.org/af/AustriaWiki/Speicherkraftwerk_%28Wasser%29

Erstellen eines fiktiven optimierbaren Netzwerks

Ergebnis des Schrittes: PyPSA-Netzwerk, das die heutige Netzwerk- und Kraftwerksstruktur vereinfacht abbildet und in dem Solar- und Windenergie sowie entsprechende Speicherkapazitäten in Österreich optimiert werden können.



dritter Schritt des Workflows.png) **Dritter Schritt des Workflows:** Integration von Zeitreihen, Verbrauchsdaten und weiteren Netzwerkelementen

Integration der Verbrauchsdaten

Die Verbrauchsdaten der europäischen Länder werden von der Open-Power-System-Data-Seite geladen. Hierfür werden Verbrauchsdaten von 2019 verwendet. Da die Datensets nicht vollständig sind werden sie im Skript `build_load_data` zur weiteren Verarbeitung aufbereitet. Fehlende Daten von 3h oder weniger werden linear interpoliert, für längere Zeiten werden entsprechend ähnliche Ausschnitte des Datensets kopiert und die fehlenden Daten damit ersetzt. Innerhalb eines Landes würden die Daten mit einem Top-down Ansatz auf verschiedene Zellen, die von einem Netzknotenpunkt repräsentiert werden, verteilt. Dabei wird die Last entsprechend des BIP und der Bevölkerung pro Zelle aufgeteilt. Dieser Schritt wird im weiteren vernachlässigt, da Österreich von einem Knotenpunkt repräsentiert wird und die Verbrauchsdaten deshalb direkt aus der erwähnten Datei extrahiert und verarbeitet werden können.

Des weiteren wird in den betrachteten Szenarien ein Anstieg des Verbrauchs angenommen, was durch zwei Funktionen integriert wird. Zum einen werden die Verbrauchsdaten linear skaliert, zum anderen werden die Zeitreihen gemäß einer zeitlichen Verschiebung der Last modifiziert.

Sektoren, die den Stromverbrauch in Zukunft stark beeinflussen werden:

- Verkehr: starker Anstieg durch Elektromobilität (ca. $12e3$ - $29e3$ TJ)
 - geht man davon aus, dass ca. 500 PKWs pro 1000 Einwohner sein werden (2020: 570: <https://de.statista.com/statistik/daten/studie/288168/umfrage/pkw-dichte-in-oesterreich/>, Stand Oktober 2021), eine mittlere Fahrstrecke von 50 km/d und einem mittleren Energiebedarf von 12kWh pro 100 km, ergibt sich ein Elektrizitätsbedarf von **8.8 TWh** pro Jahr (ca. 10% des Energiebedarfs in AUT) => Energieszenario rechnet mit 8.07 MWh
 - gesteuerte Ladung in Schwachlastzeiten: **23h - 6h:** Laden mit geringer Ladeleistung (1kW/Auto) und am Tag **bis 18 h:** eine Nachladung von 50 km => 6kWh/Auto
 - tägliches Potential an Energie zum Verschieben: $27.6 * 10^3$ MWh, ein Großteil über Nacht (plausibel, ca 25%)
 - Referenzjahr: 2012
- Industrie: genereller Anstieg (höheres Wirtschaftswachstum in der Industrie) (ca. $1.01e4$ - $1.48e4$ TJ)

Deshalb werden die Lastzeitreihen mit folgenden Parametern modifiziert, welche in der Funktion `varring_adjustments` im Skript `build_load_data` implimentiert sind:

Tägliche Elemente → Anzahl der Stützstellen: 12 (2h-Genauigkeit);

Verkehr: Nachtladung und leichte Tagladungen (1.0) und Spitzenglättung, weil die gesamten Loads 15% gesteigert sind, also der Ausgleich sonst fehlt

Zeit	Begründung	Wert
0	e-Mobilität	1.1
2	e-Mobilität	1.1
4	e-Mobilität	1.1
6	e-Mobilität	1.1
8	Morgenspitze glätten	0.95
10	Morgenspitze glätten	0.95
12	Verstromung Industrie	1
14	Verstromung Industrie	1
16	Spitze glätten	0.95
18	Spitze glätten	0.95
20	1	1
22	e-Mobilität	1.1
Total	Durchschnitt	1.025

der Verbrauch wird damit um 2.5% angehoben.

Saisonale Elemente → keine hohe Anzahl an Stützstellen notwendig: 6 Stützstellen gewählt, die alle zwei Monate darstellen

Zeit	Begründung	Wert
Jänner	Ausgleich zu Sommer	0.95
März	Ausgleich zu Sommer	0.95
Mai	Juni: Anstieg durch Klimaanlage	1
Juli	Anstieg durch Klimaanlage	1.1
September	Anfang September Klimaanlage	1
November	Ausgleich zu Sommer	1
Total	1	1

der Verbrauch bleibt damit insgesamt gleich.

Bei der Modellierung der täglichen Änderungen werden trotz zweistündlicher Auflösung von 24 Stunden 12 Stützstellen verwendet, da sich durch die unterschiedlichen Werte und das Problem des Overfittings ohne eine kompliziertere Veränderung der modellierenden Funktion ansonsten harmonische Schwingungen ergeben, welche die Modifikation der Verbraucherdaten verfälschen würde.

Optimierbare Elemente

Zusätzlich zu den Verbraucherdaten werden in das Netzwerk zu jedem Netzknotenpunkt Generatoren-Elemente und Speicher-Elemente hinzugefügt. Die Elemente bilden damit Solarkraftwerke, Windkraftwerke, Batteriespeicher und Wasserstoffspeicher ab, die noch nicht zur Energieerzeugung bzw. -speicherung beitragen. Der Ausbau dieser kann in weiterer Folge optimiert werden. → **Evtl. erst im nächsten Schritt,**

nachdem die 10 nodes gebildet wurden? (Schlussendlich ist es egal, weil sie sowieso wenn dann addiert werden müssten)

Quellen:

- Verbrauchsdaten: Open-Power-System-Data: https://data.open-power-system-data.org/time_series/2019-06-05, Stand Oktober 2021
- Abschätzung der zeitlichen Verschiebung der Last(1): Brauner, G. (2009): Energiebereitstellung für die Elektromobilität; doi:10.1007/s00502-009-0682-9
- Abschätzung der zeitlichen Verschiebung der Last(2): Kalt, G.; Baumann, M.; Eggler, L.; Holzmann, A.; Pauritsch, G. (2016): Energieszenario für Österreich: Entwicklung von Energienachfrage und Energieaufbringung bis 2030; Bundesministerium für Wissenschaft, Forschung und Wirtschaft.
- **Quelle für den Anteil von Kühlanlagen wieder finden!**

Vereinfachen des Netzwerks

Ziel des Schrittes: eine Netzwerkbeschreibung mit optimierbarem PyPSA-Netzwerk, die numerisch lösbar ist.

vierter Schritt des Workflows.png) **Vierter Schritt des Workflows:** Vereinfachung des PyPSA Netzwerks um eine numerische Lösung möglich zu machen


Vereinfachung der Netzstruktur

Die Netzstruktur bildet in passender geografischer Aufteilung alle Netzebenen ab 220 kV ab. Diese werden in diesem Schritt auf eine äquivalente 380 kV Struktur vereinfacht. Das hat für die Ergebnisse des Netzmodells keine direkte Auswirkung, da die Verbindungen der Netzknotenpunkte fiktiv sind, was daraus resultiert, dass jedes Land von einem Netzknotenpunkt repräsentiert wird. Numerisch ist das resultierende Modell jedoch leichter und stabiler lösbar.

Warum werden vorher die Transformatoren ins Netzmodell integriert, wenn danach das ganze Modell auf eine 380 kV-Ebene reduziert wird und Transformationen wegfallen müssen?

Clustering

Das Netzwerk wird zu den 10 Netzknotenpunkten, die Österreich und die Nachbarländer, wie im ersten Schritt des Workflows erklärt, abbilden, vereinfacht. Dabei werden Deutschland und Italien von zwei Netzknotenpunkten repräsentiert und es ergibt sich folgende Zellen-Struktur für das Modell des elektrischen Netzwerks.

Zellen-Struktur des optimierbaren Netzwerks **Zellen-Struktur des optimierbaren Netzwerks:** betrachtetes Gebiet, wobei jede Zelle von einem Netzknotenpunkt im elektrischen Netzwerk repräsentiert wird. Deutschland und Italien werden von je 2 Netzknotenpunkten repräsentiert.

Warum entsteht da ein Gebiet in der Steiermark, das nicht vom Österreichischen Netzknotenpunkt repräsentiert wird?

Jede der abgebildeten Zellen wird durch einen Netzknotenpunkt repräsentiert. Diese werden durch das elektrische Netzwerk miteinander verbunden, was in der folgenden Abbildung zu sehen ist. Dabei sind alle Netzknotenpunkte in das Netzwerk integriert und mit den Nodes, die die jeweils benachbarten Zellen repräsentieren verbunden. Die Breite der Verbindungen stellt ist direkt proportional zur maximal transportierbaren Scheinleistung. Sardinien ist mit einer HVDC-Verbindung mit dem italienischen Festland verbunden, welche in dem Plot durch die grüne Farbe gekennzeichnet ist. Ebenso besteht zwischen den Nodes

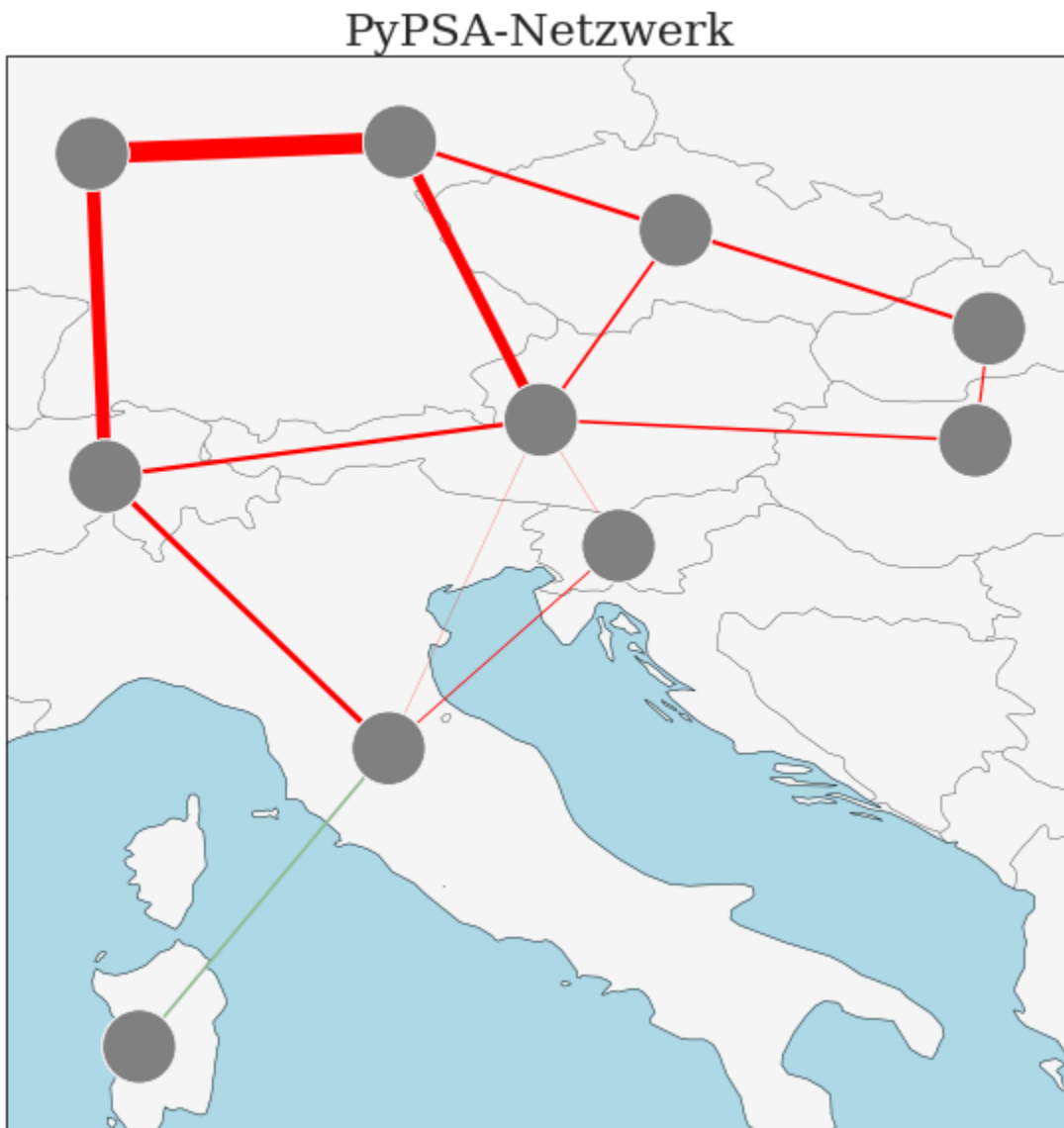
von Italien und der Schweiz neben der Wechselstromverbindung auch eine Gleichstromverbindung.
Informationen über bestehende Gleichstromverbindungen werden im Skript `prepare_links_p_nom` von https://en.wikipedia.org/wiki/List_of_HVDC_projects#Europe geladen und verarbeitet.

```
In [45]: fig, ax = plt.subplots(figsize = (10,10), subplot_kw = {"projection": ccrs.PlateCarree()})
n.plot(ax = ax, bus_colors = 'gray',
      line_widths = n.lines.s_nom/3e3,
      line_colors = 'red',
      color_geomap = True,
      line_cmap = plt.cm.viridis,
      bus_sizes = 0.2)
plt.title("PyPSA-Netzwerk", fontsize = 23)
```

```
Out[45]: Text(0.5, 1.0, 'PyPSA-Netzwerk')

/home/max/anaconda3/envs/pypsa-eur/lib/python3.8/site-packages/cartopy/mpl/geoaxes.py:387:
MatplotlibDeprecationWarning:
```


The 'inframe' parameter of `draw()` was deprecated in Matplotlib 3.3 and will be removed two minor releases later. Use `Axes.redraw_in_frame()` instead. If any parameter follows 'inframe', they should be passed as keyword, not positionally.



Das Modell des Netzes ist damit soweit vereinfacht, dass im nächsten Schritt mathematische Gleichungen darauf angewendet werden können und das resultierende mathematische Problem an einen Solver übergeben werden kann.

Numerische Lösung des mathematischen Problems

Ziel dieses Schrittes: Aufstellen einer Matrix, die das elektrische Netzwerk mathematisch beschreibt und Lösen dieser, sodass Zeitleisten elektrischer Erzeugung und Verbrauch sowie Informationen über Lastflüsse in einer stündlichen Genauigkeit für einen Zeitraum von einem Jahr vorliegen.

 fünfter Schritt des Workflows.png) **Fünfter Schritt des Workflows:** Numerische Lösung des Problems mit einem geeigneten Solver; dabei werden unterschiedliche Gleichungen und Bedingungen an die Lösung berücksichtigt.

In diesem Schritt wird das in den ersten vier Schritten definierte Problem numerisch gelöst. Je mehr Bedingungen an die Lösung gestellt werden, desto komplizierter ist die numerische Lösung und desto mehr Zeit nimmt diese in Anspruch. Dafür ist ein entsprechender Solver und genügend Rechenleistung notwendig. Die Bedingungen, die das Problem definieren können in drei thematische Bereiche gegliedert werden, die in die Lösung mit einfließen.

Lastflussgleichungen

Die numerische Lösung wird durch eine Linearisierung der Lastflussgleichungen erreicht. Die Linearisierung ist dann anwendbar, wenn der Wirkwiderstand R gegenüber des Blindwiderstands X verschwindend klein ist, Differenzen der Spannungswinkel klein genug sind, um $\sin \alpha \approx \alpha$ annehmen zu können und "reactive power decouples from active power", also wenn

$$r_l \ll |x_l|$$

$$\sin(\theta_l - \theta_k) \approx \theta_l - \theta_k$$

. Ziel ist es, den Lastfluss zwischen den Nodes bei bekannter Leistung p_i (mit $p_i \in [-\infty, \infty]$) an den Netzknoten konsistent zu berechnen.

Mathematische Definition des Lastflussproblems:

$i = 1, \dots, N$ Netzknotenpunkte ($N = 10$)

$l = 1, \dots, L$ Netzwerkverbindungen ($L = 13$)

θ_i = Spannungswinkel an Netzknotenpunkt i

K_{il} = Incidence-Matrix mit der Eigenschaft:

$$K_{il} = \begin{cases} 1 & \text{wenn } l \text{ beim Punkt } i \text{ startet} \\ -1 & \text{wenn } l \text{ beim Punkt } i \text{ endet} \\ 0 & \text{sonst} \end{cases}$$

die Dimension der Matrix ist $\dim K = N - 1$, da eine Zeile immer von den anderen Zeilen linear abhängig ist.

$C_{lc} = \ker K$ = Ringmatrix, welche alle Cycles in einem Netzwerk abbildet, also wenn die Netzwerkverbindung innerhalb einer begrenzten Anzahl an Nodes wieder zu ihrem Startnode zurück kommt. Für die Dimension der Matrix gilt $\dim C = L - (N - 1)$

- Energieerhaltung: $\sum_i p_i = 0$
- Kirchhoff'sche Knotenregel: $p_i = \sum_l K_{il} f_l$. Damit sind $N - 1 = 9$ unabhängige Gleichungen definiert.

- Kirchhoff'sche Maschenregel: $\sum_l c_{lc} \sum_i K_{il} \theta_i = 0$. Sie definiert $L - (N - 1) = 13 - 9 = 4$ unabhängige Gleichungen.

Damit sind 13 unanabhängige Gleichungen definiert, was ausreichend ist um die 13 Lastflüsse zu berechnen. Das mathematische Problem ist damit ausreichend determiniert.

Randwerte

Dem aus den Lastflussgleichungen resultierenden Gleichungssystem werden über das Skript `solve_network` fixierte Randwerte zugeteilt. Auf der Verbraucherseite sind die Lastprofile pro betrachtetem Land fixiert, was bedeutet, dass sie zu jedem Zeitpunkt (snapshot) an jedem Netzknotenpunkt gedeckt werden müssen. Dieser "Randwert" resultiert eigentlich direkt aus den oben angeführten Gleichungen, ist aber hier nochmal extra angeführt um zu betonen, dass die Last initial als unelastisch in das Modell einfließt.

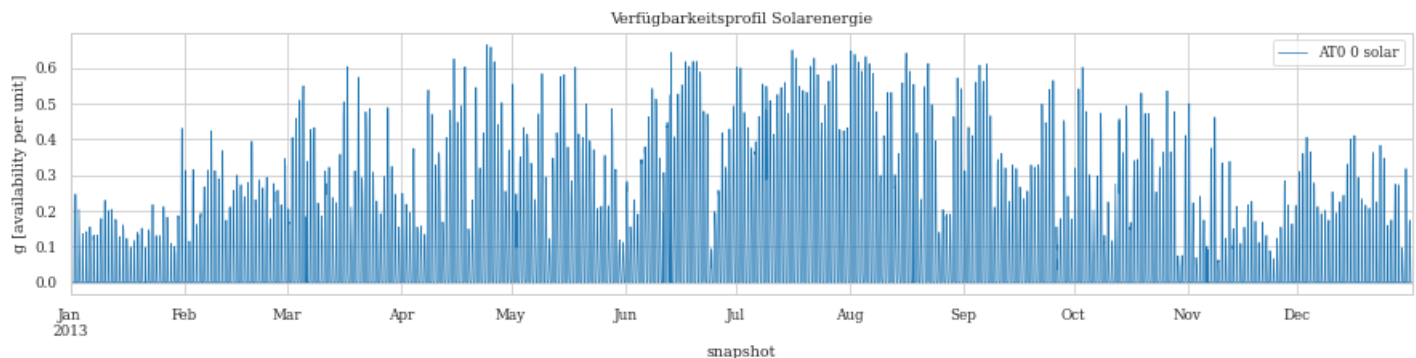
Wie im zweiten und im dritten Schritt des Workflows angemerkt sind in das PyPSA-Netzwerk optimierbare Elemente integriert, deren Kapazität initial nicht festgelegt ist, sondern anhand anderer Constraints festgelegt wird. Pro Node sind das je ein Generator für Wind- bzw. Solarenergie sowie je eine Speichereinheit für Batterie- sowie Wasserstoffspeicher. Im zweiten Schritt des Workflows werden die Verfügbarkeitsprofile \bar{g} für Solar- und Windenergie erstellt. Sie begrenzen nun gemeinsam mit der maximal installierbaren Kapazität $G_{n,r}$ die maximale Stromerzeugung $g_{n,r,t}$ pro Energieträger r an einem Netzknotenpunkt n zu jedem Zeitpunkt t durch

$$g_{n,r,t} \leq \bar{g}_{n,r,t} G_{n,r} \text{ mit } \bar{g} \in (0, 1)$$

. Beispielhaft ist hier das Verfügbarkeitsprofil für Solarenergie gezeigt.

```
In [46]: n.generators_t.p_max_pu.filter(like = 'AT0 0 solar').plot(figsize = (15,3), title = 'Verfü
plt.ylabel('g [availability per unit]')

Out[46]: Text(0, 0.5, 'g [availability per unit]')
```



An dieser Stelle fließen auch die Werte der Inputszenarien, welche die erzeugte Menge elektrischer Energie innerhalb eines Jahres für eine bestimmte Technologie festlegen in das Modell ein. Sie werden als extra Constraints im Skript `solve_network` angeführt. Für die Erzeugungsanlagen Solar, Wind und Wasser werden pro Technologie zwei Constraints hinzugefügt, deren Funktionen im Folgenden für die Solarenergie gezeigt sind.

```
def constraint_solar(n, config): config = n.config gen_constraints = config['minimal_generation'] solar_var = get_var(n,
'Generator', 'p').filter(like='solar').filter(like='AT') lhs = linexpr((1, solar_var)).sum(axis=1).sum(axis=0) rhs =
(config['minimal_generation'].get('solar')) define_constraints(n, lhs, '>=', rhs, 'solar_constraint') def constraint_solarMAX(n,
config): # in einem 10%-RANGE config = n.config gen_constraints = config['minimal_generation'] solar_var = get_var(n,
'Generator', 'p').filter(like='solar').filter(like='AT') lhs = linexpr((1, solar_var)).sum(axis=1).sum(axis=0) rhs =
(((config['minimal_generation'].get('solar'))*1.02) define_constraints(n, lhs, '<=', rhs, 'solar_constraint')
```

Für die Energiemenge, die aus dem Generator 'solar' in Österreich stammt, wird mit der ersten Funktion ein minimaler Wert definiert, welcher in der Config-Datei in *MWh* angegeben werden kann. Die Summe über

die stündliche Leistung (lhs) wird der minimalen Energiemenge über ein Jahr (rhs) gegenübergestellt und darf diese nicht unterschreiten. Die lhs ist dabei so definiert, dass die Constraint bei höherer geografischer Auflösung gleich angewendet werden kann, weshalb zwei Summen notwendig sind. Wird mit einer niedrigeren zeitlichen Auflösung modelliert muss der Wert für die minimale Erzeugung durch die zeitliche Auflösung dividiert werden. Bei einer zeitlichen Genauigkeit von 6h wäre beispielsweise in der Config-Datei für die Energiemenge `minimal_generation: solar: E[MWh]/6`. Die Funktion `constraint_solarMAX()` definiert die maximale Erzeugungsmenge, die gleich der minimalen Erzeugungsmenge plus 10% ist. Damit wird garantiert, dass die Erzeugungswerte, die in den zu modellierenden Szenarien angegeben sind, auf eine Genauigkeit von 2% modelliert werden. Da das gesamte Netzwerk zudem auf minimale Kosten optimiert wird und die Betriebskosten für Solar- und Windkraftwerke in der Config-Datei angepasst sind kann grundsätzlich davon ausgegangen werden, dass die Energiemenge pro Jahr mit einer höheren Genauigkeit erreicht wird. Für `minimal_generation: solar: 2.7` werden beispielsweise durch Solarkraftwerke in Österreich in dem Modell nach der Optimierung

```
In [47]: n.generators_t.p.filter(like='AT0 0 solar').sum().sum()
```

```
Out[47]: 27000004.806650948
```

MWh elektrischer Energie erzeugt. Die Eingaben für `minimal generation` in der Config-Datei sollen also genau den in einem zu modellierenden Szenario angegebenen Energiemengen entsprechen, um eine genaue Modellierung des Szenarios zu erreichen.

Für den folgenden Workflow weniger von Interesse ist die Randbedingung, die den maximalen CO₂ Ausstoß der Energieerzeugung festlegt. Änderungen wirken sich auf alle freien Variablen der Stromerzeugung bzw. Speicherung aus, die im Betrieb einen CO₂-Ausstoß aufweisen, also konventionelle Kraftwerke und Speicher, wie Batteriespeicher und Wasserstoffspeicher.

Optimierung nach geringsten Kosten

Der ursprüngliche Workflow von PyPSA-Eur dient dazu, die Kosten eines elektrischen Energiesystems zu minimieren. Dabei wird ein CO₂-Limit beachtet und Solar- bzw. Windkraftwerke entsprechend dieses Limits ausgebaut. Der Prozess der Kostenminimierung wurde aus Gründen der Einfachheit nicht umgangen, sondern ist weiterhin Teil des Workflows und wird vom Solver gemeinsam mit den in den letzten Absätzen beschriebenen Constraints und der Lösung der Matrix, die die Netzwerkgleichungen beschreibt, durchgeführt.

Solver

Einfache Netzwerke, wie ein auf Österreich beschränktes Netzwerk ohne weitere Vorgaben können mit frei verfügbaren Solvern gelöst werden, für das vorliegende Netzwerk wurde ein proprietärer Solver verwendet. Dabei kommen die Solver *Gurobi* und *CPLEX* infrage. Gurobi stellt auf beschränkte Zeit gratis Studenten-Lizenzen zur Verfügung, die für die Berechnungen verwendet wurden.

Die aus den Bedingungen und Netzwerkgleichungen resultierende Matrix deckt einen relativ großen Zahlenbereich ab, weshalb beispielsweise eine Anpassung der CO₂-Constraints an die Größenordnung der bestehenden Constraints notwendig ist, um das Modell numerisch stabil zu halten. Die angeführten Statistiken der Netzwerkmatrix können vom Gurobi-Solver angezeigt werden:

```
Coefficient statistics:
  Matrix range      [1e-02, 3e+02]
  Objective range   [9e-03, 2e+05]
```

Bounds range	[3e+00, 9e+09]
RHS range	[5e-01, 3e+07]

Die großen Intervalle der "Matrix range" von 10^4 und "RHS range" in der Größenordnung von 10^8 zeigen die schlechte Kondition des Problems. Entsprechend ist neben der Anpassung der Solver-Variablen noch die Variable `NumericFocus = 1` im `config.yaml` File hinzugefügt, womit zwar die Rechenzeit erhöht wird, numerische Probleme aber vorgebeugt werden. Für den Schritt des Netzwerk lösens ist somit in etwa mit einer Rechenzeit von 3 bis 4 Stunden zu rechnen.

Quellen:

- Brown et al: PyPSA: Python for Power System Analysis
- Hörsch et al: Linear Optimal Power Flow Using Cycle Flows
- Hörsch et al: PyPSA-Eur An Open Optimisation Model of the European Transmission System

Sechster Schritt des Workflows

Ziel dieses Schrittes: Auswertung des elektrischen Netzwerks mit Fokus auf Zeitanalysen und Beantwortung der wissenschaftlichen Fragestellungen

Die Auswertung dient konkret dazu, die wissenschaftliche Fragestellung der Arbeit und thematisch zusammengehörige relevante Fragestellungen zu beantworten. Die wissenschaftliche Fragestellung lautet: Welche Residuallast nicht erneuerbarer Energieerzeugung ergibt sich bei unterschiedlichen Ausbauraten fluktuierender erneuerbarer Energieträger im österreichischen Stromnetz und wie verhält sich die zeitliche Auflösung der Residuallast? Um eine grundsätzliche Beschaffenheit der nicht-erneuerbaren Residuallast beschreiben zu können stellen sich zudem noch untergeordnete Fragen:

- In welchen zeitlichen Abständen ist generell mit einem erhöhten Bedarf an nicht erneuerbarer Residuallast zu rechnen? Damit verbunden: Welche Frequenzen der Residuallast sind zu erkennen?
- Welche elektrische Leistungsbereiche müssen wie oft durch nicht erneuerbar erzeugten Strom abgedeckt werden?
- Welche Zeitdauern der nicht-erneuerbaren Residuallast müssen überbrückt werden und zu welcher Tages- bzw. Jahreszeit sind diese Zeiten häufig?


Zur Beantwortung dieser Fragestellungen werden die Jahresdauerlinie der Residuallast dargestellt und eine Frequenzanalyse durchgeführt.

Zudem wird die Frage, wie sich das Angebot von Strom aus fluktuierenden erneuerbaren Quellen auf die Anforderungen an die Energieflexibilität auswirkt behandelt. Dazu wurden folgende untergeordnete Fragestellungen formuliert:

- Welche Zeitdauern der nicht-erneuerbaren Residuallast müssen überbrückt werden und zu welcher Tages- bzw. Jahreszeit sind diese Zeiten häufig?
- Wie unterscheidet sich die Beschaffenheit vorherrschender Frequenzen der Residuallast von erkennbaren Frequenzen anderer relevanter Netzzeitreihen?
- Welche Menge an Residuallast kann durch kurzfristige Lastverschiebung mit erneuerbarer Energie gedeckt werden?
- In welcher Weise kann die Residuallast nicht erneuerbarer Energieträger als Basis für Schaltsignale möglicher Lastverschiebungspotentiale verwendet werden?

- Welche Anforderungen an die Lastverschiebungspotentiale (in Plusenergiequartieren) stellen sich durch die Auswertung der Residuallast?

Um diese Fragestellungen beantworten zu können wird die Zeitreihe der Residuallast ausgewertet und Zeitdauern von Residuallast-Phasen, Anzahl an Nulldurchgängen der Residuallast und statistische Daten evaluiert. Zudem werden aus der Residuallast und dem zeitlichen Mittelwert dieser Schaltsignale erstellt, die als einfache Schaltsignale für Lastverschiebungspotentiale dienen können. Außerdem wird ein Skript codiert, mit dem einfache Lastverschiebungen modelliert werden und die Potentiale kurzfristiger sowie langfristiger Lastverschiebung abgeschätzt werden.

 sechster Schritt des Workflows.png) **Sechster Schritt des Workflows:** Analyse des Netzwerks und Beantwortung der wissenschaftlichen Fragestellung und thematisch damit zusammenhängender Fragestellungen, die in den blauen Ellipsen angegeben sind.

Das im vorherigen Schritt numerisch gelöste Netzwerk ist in einer netCDF-Datei gespeichert und kann in einem Python-Skript analysiert werden. Die Datei muss dafür mit `pypsa.Network("path_to_file")` importiert werden, wie dies am Anfang dieses Notebooks getan wurde. Der Pfad zur Netzwerkdatei muss dafür nicht absolut sein. Der filename muss dafür im file `config_auswertung.yaml` bei `network: filename` angegeben werden. Die mathematische Auswertung wird dabei als vom restlichen Workflow getrennt behandelt, damit Netzwerke verschiedener Szenarien einfach analysiert und verglichen werden können.

Programm- Funktionen der Auswertung

In den folgenden Absätzen werden die verschiedenen Funktionen, die die mathematische Auswertung der vorhin genannten Punkte ausführen, beschrieben und an Code-Beispielen erklärt. Voraussetzung dafür ist das Erstellen der Zeitreihe nicht-erneuerbarer Residuallast, welche am Beginn des Skripts

`Netzwerkanalyse/Auswertung.py` erstellt wird. Die Residuallast besteht aus der Last, die nicht durch Strom erzeugt mit erneuerbaren Energieträgern in Österreich oder Strom aus Speichereinheiten in Österreich gedeckt werden kann. Zeitreihen erneuerbarer Energieträger und Speicher werden deshalb summiert und von der Zeitreihe des elektrischen Energieverbrauchs abgezogen. Dabei werden die Energieträger Solar, Wind, Run of River und Biomasse als erneuerbare Erzeugungsanlagen gezählt, Speicherwasserkraftwerke und Pumpspeicherkraftwerke zählen wie Batterispeicher und Wasserstoffspeicher zu den Speichereinheiten.

In [70]:

```
renewables = ['solar', 'wind', 'ror', 'biomass']
stores = ['H2O', 'battery']
store_units = ['PHS', 'hydro']
load = n.loads_t.p.filter(like='AT')
renload = n.generators_t.p.filter(like = 'geothermal').filter(like = 'AT').sum(axis=1)
for x in renewables:
    renload += n.generators_t.p.filter(like = x).filter(like = 'AT').sum(axis=1)
for x in stores:
    renload += n.stores_t.p.filter(like=x).filter(like='AT').sum(axis=1)
for x in store_units:
    renload += n.storage_units_t.p.filter(like=x).filter(like='AT').sum(axis=1)
renload[renload<0]=0
renload = renload.to_frame(name = 'AT0 0')
res = load.values - renload ###Zeile wurde fürs notbook angepasst!
```

Aufgrund der unterschiedlichen Modellierung der Speichereinheiten von Wasserspeicherkraftwerken und Pumpspeicherkraftwerken zu Batteriespeicher- bzw. Wasserstoffspeicheranlagen muss hier zwischen `stores` und `stores_units` unterschieden werden.

Das weitere Skript `Auswertung.py` ist in die Bereiche `_BasicData`, `_Residualload`, `Nullldurchgang`, `Jahresdauerlinie`, `Zeitdauern`, `Lastverschiebungssimulation`, `Frequenzanalyse` und `Schaltsignale` geteilt. Dabei

besteht jeder Teil aus den Funktionen, welche die Auswertung bzw. Bearbeitung der Daten durchführen und einem Skript, durch das die bearbeiteten Daten entsprechend graphisch aufbereitet und gespeichert werden. Die Funktionen rufen sich dabei selbst auf und die Eingabe, welche thematischen Bereiche ausgewertet werden sollen kann im `config_auswertung.yaml` file durch Eingabe von `true` oder `false` getroffen werden.

Die Ergebnisse werden dann in mehrfacher Form ausgegeben. Zum Einen wird ein Ordner mit Plots erstellt, welcher alle grafischen Aufbereitungen enthält, zum Anderen werden alle relevanten Outputs als csv-Datei gespeichert. Zusätzlich gibt es ein Excel-File, das die Ergebnisse auf mehreren Tabellenblättern enthält. Dieses wird, genauso wie die csv-Dateien, im Ordner `_resultfiles` gespeichert.

Basic Data

Speicherort der Ausgaben:

- Plots/Basic_Data
- Zusammenfassung Netzwerk.txt

Es werden die Zeitreihen der Erzeugung erneuerbarer elektrischer Energie in Österreich und der elektrischen Last in Österreich geplottet. Ebenso werden die Stromerzeugungstechnologien nach installierter Leistung und Stromerzeugung innerhalb eines Jahres aufgelistet. Dies kann beispielsweise als erster Check verwendet werden, in welcher Genauigkeit das gewünschte Szenario modelliert werden konnte. Die Daten werden dafür direkt aus den Pandas-Dataframes bzw. Dictionarys der importierten nc-Datei verwendet, werden in `['solar', 'wind', 'ROR', 'coal', 'gas', 'rest']` gruppiert und direkt geplottet bzw. in der Datei `Zusammenfassung Netzwerk.txt` gespeichert.

Residuallast

Speicherort der Ausgaben:

- Plots/Residuallast
- RES_pro_Monat.csv
- RES_pro_Tag.csv
- RES_pro_Stunde.csv
- RL_nach_Tageszeit_pro_Monat.csv

ungefährer Ort im Skript: 210 - 350

Die Auswertung der Residuallast findet auf unterschiedlichen zeitlichen Skalen statt. Die Funktionen `Residuallast_per_hour(jahr, res)`, `Residuallast_per_day(jahr, silvester, pres, einjahr)` und `Residuallast_per_month` werten die Residuallast auf einer stündlichen, einer 24-stündlichen und einer monatlichen Genauigkeit aus. Dabei wird für die langfristige Auswertung die Leistung der stündlichen Residuallast über 24 Stunden oder über 720 ± 24 Stunden (abhängig davon, wie viele Tage das jeweilige Monat hat) summiert. Die stündlichen Daten werden direkt aus dem nc-File exportiert, für die Gruppierung der Last nach Tagen wird eine for-Schleife verwendet, die jeweils über einen Tag läuft. Die Gruppierung der Residuallast nach Monat kann mit der pandas-funktion `df.groupby(index)` durchgeführt werden. Grafisch aufbereitet werden die Ergebnisse von der Funktion `plot_Residuallast(counts, zeitraum, einheit)`, die sich im Skript am Ende des Abschnittes "Residuallast" befindet.

Für eine Analyse der tages- und jahreszeitlichen Schwankungen der Residuallast werden monatlich die Residuallast-Zeiten ausgewertet. Dabei wird auf die Auswertung der Leistung der Last verzichtet. Es wird an jedem Tag d in einem Monat m unterschieden, ob zum jeweiligen Tageszeitpunkt t auf nicht-erneuerbare

Residuallast zurückgegriffen wird, oder nicht. Die Variable x_t für einen bestimmten Tag d eines Monats m ergibt sich dann aus

$$x_{m,d,t} = \begin{cases} 1 & \text{if Residuallast} > 0 \\ 0 & \text{if Residuallast} \leq 0 \end{cases}$$

Der angeführte Code, welcher der Code der Funktion

`Residuallast_yn_per_day_per_hour(res, jahr)` ist summiert über alle Tage eines Monats, damit das Ergebnis die Anzahl der Tage in einem Monat darstellt, an denen zu dieser Uhrzeit auf nicht-erneuerbare Residuallast zurückgegriffen wird. Jeder Eintrag $x_{m,t}$ der Matrix $X^{12 \times 24}$ wird also durch

$$x_{m,t} = \sum_d x_{m,d,t} \quad \text{mit } m \in [0, 11], t \in [0, 23]$$

erstellt. Das Ergebnis der Funktion, X ist hier zu sehen. Dafür wurde der Code der Funktion leicht verändert, um im Notebook ein Ergebnis anzuzeigen. Alle eingefügten Commands sind extra gekennzeichnet.

In [88]:

```

#### Beginn
monate = np.linspace(0,8760, 13, dtype = int, endpoint = False)
months = ['Jänner', 'Februar', 'März', 'April', 'Mai', 'Juni', 'Juli', 'August', 'September', 'Oktober', 'November', 'Dezember']
#### End: Zeilen wurden fürs Notebook eingefügt
nach_monat_zeit = [[0] * 24, [0] * 24, [0] * 24, [0] * 24, [0] * 24, [0] * 24, [0] * 24, [0] * 24, [0] * 24, [0] * 24, [0] * 24, [0] * 24, [0] * 24]
for v in range(1, res.size):
    if (res['AT0 0'].iloc[v] > 0):
        point=res.index[v]
        for i in range(0, len(nach_monat_zeit)):
            if (v >= monate[i] and v < monate[i+1]):
                for ask_run in range(0,24):
                    if ask_run == point.hour:
                        nach_monat_zeit[i][ask_run] += 1
frame = pd.DataFrame(nach_monat_zeit, columns= range(0,24), index = months)
frame #### Zeile fürs Notebook hinzugefügt

```

Out[88]:

	0	1	2	3	4	5	6	7	8	9	...	14	15	16	17	18	19	20	21	22	23
Jänner	19	18	19	20	22	24	24	25	21	21	...	21	26	25	24	23	19	16	19	18	20
Februar	18	20	21	20	22	24	23	21	19	20	...	20	25	26	26	24	22	22	23	19	19
März	18	19	18	18	19	21	21	19	18	16	...	18	19	22	23	22	18	18	18	19	19
April	19	19	18	21	22	23	24	24	19	19	...	21	23	18	16	14	14	14	15	17	18
Mai	7	6	7	7	7	7	12	9	6	5	...	8	10	9	6	5	5	4	5	6	6
Juni	10	10	10	10	11	12	12	11	4	5	...	11	13	10	10	10	11	10	11	11	10
Juli	8	7	7	7	9	7	12	13	9	5	...	11	16	10	9	7	7	6	6	6	7
August	5	5	7	6	11	9	10	13	10	4	...	14	14	7	4	5	5	5	4	6	5
September	9	8	8	8	11	14	12	18	12	10	...	21	18	10	8	8	8	9	9	9	9
Oktober	14	14	13	12	14	18	13	15	14	12	...	17	14	15	17	12	12	12	12	13	13
November	25	24	24	24	27	27	25	23	22	19	...	23	23	25	24	23	22	21	24	24	24
Dezember	25	24	25	25	26	28	24	23	25	25	...	25	24	23	23	24	24	24	25	28	26

12 rows \times 24 columns

Weiters sind im Abschnitt "Residual load" die Funktionen `count_per_month(v,df)` , `ND_und_Res_per_month_per_hour()` und `plot_ND_und_RES_mh(monat, df, df)` zu finden. Diese Auswertungen dienen ebenfalls der Beantwortung der Fragestellung, wie die Residuallast grundsätzlich

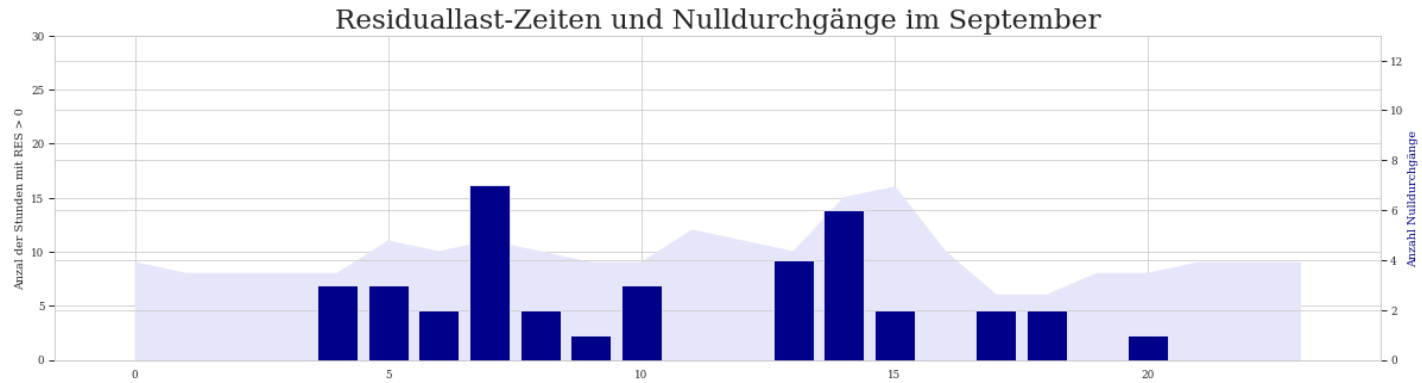
Residuallast_yn_per_day_per_hour() erklärt wurde, kann ein umfassenderer Überblick über die zeitlichen Unterschiede der Notwendigkeit nicht-erneuerbarer Residuallast verschaffen werden.

```
def count_per_month(v,nach_monat_zeit):
    point = res.index[v]
    for i in range(0, len(nach_monat_zeit)):
        if (v >= monate[i] and v < monate[i+1]):
            for ask_run in range(0, 24):
                if ask_run == point.hour:
                    nach_monat_zeit[i][ask_run] +=1
    return(nach_monat_zeit)

def plot_ND_und_RES_mh(monat, ND_frame, RES_frame):
    fig,ax = plt.subplots(figsize=(20,5))
    title2 = ''
    if monat != 'nan':
        title2 = (' im ' + monat)
    plt.title('Residuallast-Zeiten und Nulldurchgänge' + title2, fontsize = 23)
    ax.plot(range(0, 24),
            RES_frame.loc[monat],
            color = 'lavender')
    ax.set_ylabel('Anzal der Stunden mit RES > 0')
    plt.ylim([0,30])
    plt.fill_between(range(0,24), RES_frame.loc[monat],
                     color = 'lavender')
    ax2 = ax.twinx()
    ax2.bar(range(0,24),
            ND_frame.loc[monat],
            color = 'darkblue')
    ax2.set_ylabel('Anzahl Nulldurchgänge', color = 'darkblue')
    plt.ylim([0,13])

ND_nmz = [[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24]
RES_nmz = [[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24,[0]*24]
for v in range(1, res.size):
    if (res['AT0 0'].iloc[v-1] < 0 and res['AT0 0'].iloc[v] > 0):
        ND_nmz = count_per_month(v, ND_nmz)
    elif(res['AT0 0'].iloc[v] > 0):
        RES_nmz = count_per_month(v, RES_nmz)

ND_frame = pd.DataFrame(ND_nmz, columns = range(0,24), index = months)
RES_frame = pd.DataFrame(RES_nmz, columns = range(0,24), index = months)
plot_ND_und_RES_mh('September', ND_frame, RES_frame) #code fürs Notebook verändert
# die erste Variable kann verändert werden, um Ergebnisse anderer Moante zu erhalten
```



Der Parameter "Nulldurchgang" wird auch für weitere Analysen herangezogen, die im folgenden Abschnitt erläutert werden.

Nulldurchgang

Speicherort der Ausgaben:

- Plots/Nulldurchgang
- ND_per_day.csv
- ND_per-day_nach_Monat.csv

ungefährer Ort im Skript: 350 - 420

Neben dem Zeitpunkt der Nulldurchgänge der Residuallast ist auch die Anzahl der Nulldurchgänge in einem bestimmten Intervall interessant zu analysieren. Sie geben Aufschluss darauf, wie stabil Phasen positiver bzw. negativer Residuallast sind. Daraus ist abzuleiten, welche zeitliche Flexibilität Lastverschiebungspotentiale aufweisen müssen, um Lasten, die in dem Modell nicht-erneuerbar gedeckt werden effizient verschieben zu können.

Die Funktion der Residuallast wird in den Skripten auf Nulldurchgänge, also wenn gilt, dass $f(t-1) < 0$ AND $f(t) \geq 0$, untersucht die Nulldurchgänge pro Tag gezählt. Zudem wird in der Funktion `Nulldurchgang_per_month` die durchschnittliche Anzahl an Nulldurchgängen an einem Tag pro Monat ausgegeben. Die Ergebnisse werden grafisch mit einer eigenen Funktion aufbereitet und als csv-Datei gespeichert.

Jahresdauerlinie

Speicherort der Ausgaben:

- Plots/Jahresdauerlinie
- JDL - erzeugung erneuerbarer Energie.csv
- JDL - negativen Residuallast.csv
- JDL - positiven Residuallast.csv
- JDL - Residuallast.csv

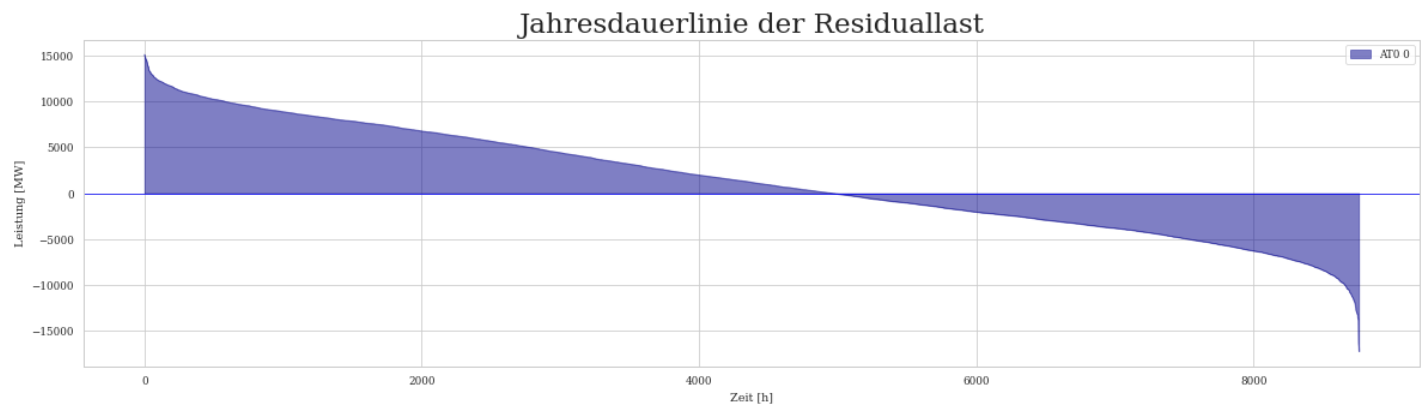
ungefährer Ort im Skript: 420 - 450

Die Darstellung der Jahresdauerlinie eignet sich, um die Last mit den Zeiträumen in Verbindung zu interpretieren und abschätzen zu können, wie viel Energie in welchen Zeiträumen fließt. Der Fokus der Arbeit liegt auf der zeitlichen Auswertung der Daten, es ist aber wichtig, die Komponente der Leistung ebenfalls in die Betrachtungen einfließen zu lassen, **da sie gemeinsam mit der Komponente der Zeit die Energie bestimmt, die generell gedeckt werden muss.**

Die Funktion `Jahresdauerlinie(frame, art)`, welche die Jahresdauerlinie erstellt kann für unterschiedliche Größen aufgerufen werden. Die diskreten Werte der übergebenen Frames werden nach Wert geordnet, graphisch aufbereitet und als Grafik und csv-Datei gespeichert. Der Code zeigt die Funktion und ein Beispiel der Jahresdauerlinie der Residuallast.

In [115...

```
def Jahresdauerlinie(frame, art):
    for i in range(0, res.size):
        if frame.iloc[i]['AT0 0'] < 0:
            art = 'Residuallast'
    sort = frame.sort_values(by = 'AT0 0',
                             ascending = False,
                             ignore_index = True)
    sort.plot.area(stacked = False,
                   figsize = (20,5),
                   ylabel = 'Leistung [MW]',
                   xlabel = 'Zeit [h]',
                   color = 'darkblue')
    plt.axhline(y = 0, c = 'blue')
    plt.title('Jahresdauerlinie der ' + art, fontsize = 23)
    # der restliche Teil der Funktion wurde für das Notebook weggelassen
    Jahresdauerlinie(res, 'Residuallast')
```



Frequenzanalyse

Jahresdauerlinie

Einfache Lastverschiebungspotentiale

Statistische Auswertung nach Tageszeitpunkt

Signale für mögliche Lastverschiebungspotentiale

Weitere Informationen und allgemeine Quellen für diese Zusammenfassung

- [readthedocs pypsa-eur](#)
- [readthedocs pypsa](#)
- [Veröffentlichung pypsa-eur](#)
- [Veröffentlichung pypsa](#)