Technische Universität München
Fakultät für Elektrotechnik und Informationstechnik
Lehrstuhl für Erneuerbare und Nachhaltige Energiesysteme

Spatial Complexity in Energy System Modeling

Kais Siala, M.Sc.

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender

Prof. Dr. Sebastian Steinhorst

Prüfer der Dissertation

1. Prof. Dr. Thomas Hamacher
2. Prof. Dr. Karen Pittel

Die Dissertation wurde am 24.06.2020 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 28.05.2021 angenommen.

Abstract

Energy systems are experiencing new trends, such as the increasing share of decentralized generation from renewable energy sources, the growing number of stakeholders due to sector coupling, and the improvements in data availability. These changes demand a revision of the way the spatial dimension is modeled. A review of the different concepts of space modeling in geography and in energy systems shows that there is a need for more flexibility in defining the scale and the shape of the spatial units, which should not be restricted to political and administrative divisions.

This thesis argues that there is an optimal spatial resolution for every research question, which mainly depends on the scope of the research and on accuracy and computational constraints. It proposes a systematic approach to reach the optimal resolution for the modeling of space. If the research question can be answered with one model, then it is possible to disaggregate the inputs, obtain high resolution data, and then aggregate it back according to the desired resolution. If more than one model is needed, model coupling can be used to expand the geographic scope and/or the model capabilities. This thesis presents different techniques for increasing and decreasing the spatial resolution of energy system data and gives recommendations for model coupling. Besides, three open-source tools for spatial data processing are shared online, each one with a detailed documentation included in the annex.

The techniques and the codes are applied in three different case studies. The first one describes how the photovoltaic technical potential in South-East Asia is estimated in a high spatial resolution using coarse atmospheric reanalysis data and other geographic information. Sites were classified based on their availabilities and their potential energy output, which could reach up to 430 TWh in the region under conservative assumptions. In the second application, a novel clustering algorithm to aggregate high resolution data is introduced and used to create model regions based on the homogeneity of the solar potential, wind potential, and total energy demand, respectively. Optimization models were generated with the same underlying high resolution data and for the same number of regions, by solely varying the shape of the spatial units. The comparison of the optimization results with a model using political divisions showed large discrepancies in the shares of renewable energy technologies in Europe in 2050. The final application coupled an optimization model of the European power system with a global, economy-wide input-output analysis. Due to differences in geographic scope and resolution, the model coupling was useful in determining the environmental and socio-economic impacts of the decarbonization of European countries on a global scale.

Ultimately, the techniques and open-source tools presented in this thesis facilitate the flexible modeling of space in energy system models and can provide critical information for policy-makers to make robust decisions on a regional level.

Zusammenfassung

Energiesysteme werden aktuell durch neue Entwicklungen, wie der zunehmende Anteil der dezentralen Erzeugung aus erneuerbaren Energiequellen, die wachsende Anzahl an Akteuren aufgrund der Sektorkopplung, sowie die Verbesserung der Datenverfügbarkeit und der Rechenleistung stark geprägt. Diese Entwicklungen stellen den Status Quo bei der Raummodellierung in Frage. Eine Gegenüberstellung der verschiedenen Konzepte der Raummodellierung in der Geographie und in Energiesystemen zeigt, dass mehr Flexibilität bei der Definition der Anzahl und der Form der räumlichen Einheiten erforderlich ist, und dass sie nicht auf politische und administrative Einheiten beschränkt werden sollten.

Die Hauptthese lautet, dass es für jede Forschungsfrage eine optimale räumliche Auflösung gibt, die hauptsächlich von ihren Zielen sowie von Genauigkeit- und Recheneinschränkungen abhängt. Ein systematischer Ansatz wird vorgeschlagen, um die optimale Auflösung zu bestimmen. Wenn die Forschungsfrage mit einem Modell beantwortet werden kann, ist es möglich, die Eingabedaten zu disaggregieren, hochaufgelöste Daten zu erhalten, und sie dann entsprechend der gewünschten Auflösung zu aggregieren. Wenn mehr als ein Modell benötigt wird, kann eine Modellkopplung verwendet werden, um den geografischen Umfang bzw. die Modellfunktionen zu erweitern. Es werden verschiedene Techniken zum Erhöhen und Verringern der räumlichen Auflösung beschrieben und Empfehlungen für die Modellkopplung ausgesprochen. Außerdem wurden drei Open-Source-Werkzeuge entwickelt und online gestellt, deren detaillierte Dokumentationen in den Anhang dieser Arbeit beigefügt wurden.

Die in dieser Arbeit beschriebenen Techniken und Werkzeuge wurden in drei Fallstudien angewendet. Die erste Fallstudie beschreibt, wie das technische Potenzial von Photovoltaik in Südostasien anhand von grobaufgelösten Reanalyse-Wetterdaten und anderen geografischen Informationen in einer hohen räumlichen Auflösung geschätzt wird. Potenzielle Standorte wurden nach ihrer Güte und Verfügbarkeit klassifiziert, und es hat sich ergeben, dass das Gesamtpotenzial von Photovoltaik in der Region unter bestimmten Annahmen 430 TWh erreichen kann. In der zweiten Fallstudie wird ein neuartiger Clustering-Algorithmus eingeführt, um hochaufgelöste Daten zu aggregieren und dann Modellregionen basierend auf der Homogenität des Sonnenpotentials, des Windpotentials bzw. des Gesamtenergiebedarfs zu erstellen. Dabei wurden Modelle mit denselben zugrunde liegenden hochaufgelösten Daten und für dieselbe Anzahl von Regionen erzeugt, indem lediglich die Form der räumlichen Einheiten variiert wurde. Die Ergebnisse dieser Optimierungsmodelle waren abhängig von der Wahl der Modellregionen, und im Jahr 2050 wurden große Unterschiede bei den Anteilen erneuerbarer Energietechnologien in Europa im Vergleich zum Modell mit politischen Einheiten festgestellt. In der dritten Fallstudie wurde ein Optimierungsmodell des europäischen Stromnetzes mit einer globalen, wirtschaftsweiten Input-Output-Analyse gekoppelt. Die Anwendung zeigt, wie die Unterschiede im geographischen Geltungsbereich und in der Auflösung genutzt werden können, um die ökologischen und sozioökonomischen Auswirkungen der Dekarbonisierung europäischer Länder auf globaler Ebene zu bestimmen.

Zum Schluss helfen die in dieser Arbeit vorgestellten Techniken und Werkzeuge dabei, die flexible Modellierung des Raums in Energiesystemmodellen zu erleichtern. Darüber hinaus können sie den politischen Entscheidungsträgern wichtige Informationen liefern, um auf regionaler Ebene fundierte Entscheidungen treffen zu können.

Résumé

De nouvelles tendances ont émergé dans les systèmes énergétiques, telles que la part croissante de la production décentralisée à partir de sources d'énergie renouvelables, le nombre croissant de protagonistes en raison du couplage sectoriel et les améliorations de la disponibilité des données et de la puissance de calcul. Ces tendances remettent en question le statu quo concernant la façon dont l'espace est modélisé. Un examen des différents concepts de la modélisation spatiale en géographie et dans les systèmes énergétiques montre qu'il faut une plus grande flexibilité dans la définition du nombre et de la forme des unités spatiales, et que celles-ci ne devraient pas se limiter aux divisions politiques et administratives.

La thèse soutient qu'il existe une résolution spatiale optimale pour chaque sujet de recherche qui dépend principalement de ses objectifs et des contraintes de précision et de calcul. Elle propose une approche systématique pour atteindre la résolution optimale. S'il est possible de répondre à la question de recherche avec un seul modèle, il est recommandé de désagréger les entrées, d'obtenir des données à haute résolution, puis de les agréger en fonction de la résolution souhaitée. Si plusieurs modèles sont nécessaires, le couplage de modèles peut être utilisé pour étendre la portée géographique et/ou les capacités du modèle. Différentes techniques pour augmenter et diminuer la résolution spatiale sont décrites et des recommandations pour le couplage de modèles sont fournies. Par ailleurs, trois outils open-source pour le traitement des données spatiales sont partagés en ligne, chacun avec une documentation détaillée incluse en annexe de cette thèse.

Trois applications de ces techniques et de ces outils sont incluses dans cette thèse. La première décrit comment le potentiel technique photovoltaïque en Asie du Sud-Est est estimé avec une résolution spatiale élevée à l'aide de données météorologiques de réanalyse à résolution grossière et d'autres informations géographiques. Cette étude a permis de classer les sites de projets potentiels selon leur qualité et leur disponibilité, et de quantifier le potentiel total de la région qui peut atteindre 430 TWh sous certaines conditions. Dans la deuxième application, un nouvel algorithme de clustering est introduit pour agréger des données haute résolution, puis utilisé pour créer des régions de modèle basées sur l'homogénéité du potentiel solaire, du potentiel éolien et de la demande totale d'énergie, respectivement. Les modèles ont été générés avec les mêmes données haute résolution et pour le même nombre de régions, en faisant seulement varier la forme des unités spatiales. Les résultats sont sensibles au choix des régions et des écarts importants dans les parts des technologies des énergies renouvelables en Europe ont été observés en 2050, par rapport au modèle utilisant des unités politiques. La dernière application montre comment le couplage d'un modèle d'optimisation du système électrique européen et d'une analyse économique globale des entrées-sorties peut être utile pour déterminer les impacts environnementaux et socio-économiques de la décarbonisation des pays européens à l'échelle mondiale.

En conclusion, les techniques et les outils open-source introduits dans cette thèse œuvrent à faciliter la modélisation flexible de l'espace dans les modèles de systèmes énergétiques, et peuvent fournir des informations essentielles aux décideurs politiques pour prendre des décisions solides au niveau régional.

ملخص

التّوجهات الجديدة في أنظمة الطّاقة تتضمّن رفع حصص مساهمة التّوليد اللاّمركزي للطّاقة إنطلاقاً من مصادر موزّعة للطّاقة المتجدّدة، علاوة على زيادة عدد المتدخّلين بسبب ترابط القطاعات، بالإضافة إلى تحسّن مستوى توفّر المعطيات والنّفاذ لها، و القدرات الحسابيّة لمعالجتها. هذه التّوجهات تطرح تحدّيات أمام التّقنيات الحاليّة المتّبعة لنمذجة المجال. مراجعة التّصورات الرّاهنة لنمذجة المجال جغرافيّاً وطاقيّاً تظهر الحاجة لمزيد من المرونة في تعريف الوحدات المجاليّة شكلاً وعدداً، في إتّجاه أن لا تقتصر على دوائر إدارية أو سياسيّة.

تناقش هذه الأطروحة وجود استبانة مجّاليّة مثلى لكل موضوع بحث، مرتبطة بشكل رئيسي بأهداف البحث كما بمقتضيات الدّقة والحساب. تقترح هذه الأطروحة مقاربة منهجية للتّوصل إلى الإستبانة المثلى. إذا كان السّؤال موضوع البحث يمكن اجابته بنموذج وحيد، فبالإمكان فصل المدخلات والحصول على بيانات ذات استبانة عالية، ثم دمج البيانات مجدّداً تبعاً لمستوى الإستبانة المنشود. إذا كانت هناك حاجة لأكثر من نموذج، يمكن إقران النّماذج لتعزيز النّطاق الجغرافي و\أو قدرات النّموذج. تصف الأطروحة تقنيات مختلفة للرفع أو الحدّ من الإستبانة المجاليّة، كما تقدّم توصيات لإقران النّماذج. تتضمّن الأطروحة كذلك لمحة عن ثلاث أدوات وقع تطويرها وإتاحتها للعموم في مصادر مفتوحة على الإنترنت، مع توثّيق مفصل لكلّ منها في الملاحق.

تحتوي الأطروحة على ثلاث تطبيقات لهذه التّقنيات والأدوات. تصف التّطبيقة الأولى كيف يمكن تقدير الإمكانات التّقنية للطّاقة الشّمسيّة في جنوب شرق آسيا وعرضها في إستبانة مجالية عالية، وذلك عبر دمج بيانات خام للطقس ومعطيات جغّرافية أخرى. تمكن الدّراسة من تحديد الأماكن المناسبة لإقامة مشاريع توليد الطّاقة الشّمسيّة وتّقييم النّاتج المحتمل اللّذي قد يبلغ 430 تيراوات-ساعة سنويّاً بالمنطقة. في المثال التّطبيقي الثّاني، وقع إعتماد خوارزميّة تجميع جديدة لإدماج معطيات ذات استبانة عاليّة، ثم استعمالها لتعريف مناطق نموذجية، على أساس تجانس الإمّكانات الطّاقية الشّمسيّة و الرّيحيّة والطّلب على الطّاقة، على التّوالي. وضعت هذه النّماذج فقط عبر تعديل شكل الوحدة المجاليّة، مع إعتماد نفس البيانات الأساسيّة ذات الإستبانة العالية ونفس عدد المناطق. نتائج نماذج التّحسين أظهرت إرتباطاً بإختيار المناطق، كما وقع رصد وجود تضارب واسع في نتائج هذه النّماذج، في ما يخص مساهمات تكنولوجيات الطّاقة المتجدّدة في أوروبا في غضون سنة 2050، وذلك بالمقارنة مع النّتائج المسجّلة عبر النّماذج المعتمدة على الحدود السياسيّة في تجزئة المجال الجغرافي. المثال التّطبيقي الأخير يظهر كيف أن إقتران نموذج تحسين المنظومة الكهربائية الأوروبية بتحليل إقتصادي شامل للمدخلات والمخرجات يمكن أن يكون فعال لإستشراف الآثار البيئيّة والإجتماعيّة والإقتصاديّة لمسار التخلص من ثاني أكسيد الكربون في الدول الأوربية على المستوى العالمي.

تمكّن التّقنيات والأدوات المدرجة بهذه الأطروحة من تيسير عملية النّمذجة المرنة للمجال في نماذج الأنظمة الطّاقية، كما توفّر معطيات هامّة لإرشاد آخذي القرار على المستوى الجهوي.

Acknowledgements

I would like to thank my advisor, Prof. Thomas Hamacher, for the opportunity to work at his chair and for the freedom to explore, try, make mistakes, and learn.

My mentor, Philipp Kuhn, deserves my gratitude for solving the administrative hurdles that came across my way.

I am thankful to my co-authors Jürgen Stich, Mohammad Youssef Mahfouz, Cristina de la Rúa, and Yolanda Lechón, for expanding my knowledge. This dissertation would not have been possible without them.

I also thank Sally M Benson from Stanford University, California, and Tobias Massier from TUMCREATE Ltd., Singapore, for hosting me as a visiting researcher in their research groups. Both experiences, which happened at different phases of my doctoral studies, have been very instructive and enriching.

My employment at TUM was supported by the Federal Ministry for Economic Affairs and Energy of Germany (BMWi) within the project "4NEMO - Research Network for the Development of New Methods in Energy System Modeling; research project: 0324008A". The research stays abroad were partially financed by the International Center and by the Graduate School at TUM. I am appreciative of the financial support of these institutions and their commitment to research and development.

To my colleagues and project partners, I would like to say that I would not have enjoyed going to work if it were not for you, and I am happy to count friends among you. My office mate and friend, Anahi Molar-Cruz, deserves a special mention for sustaining a good relationship and a peaceful work atmosphere that many would be envious of.

I am grateful to my students, from whom I learned a lot, and to my friends, who bore with me in the stressful phases (and there were quite a few).

I acknowledge the contribution of every journal reviewer, conference participant or any person who was critical of my work, even though they may not read this message: they all pushed me to pause, rethink, and improve.

I finally thank my parents and my siblings for their unconditional support. It is to them that I dedicate this work.

Contents

List of Figures

List of Tables

If I were a geographer

If I were a geographer
I would make up some space,
In my crowded maps, for a place
Just the way I prefer.

It could be anywhere, I guess
For it has no boundaries
But the limits of my fantasies
Which even I... transgress.

There would be no need
For a wall or a barbed wire.
Come and go when you desire!
Freedom is my only creed.

***

If I were a geographer
I would design a projection
To shape my place to perfection
Just the way I prefer.

Sometimes, I fancy a forest
Where the sunshine squeezes, with pain,
Between the leafy branches, in vain,
To tickle a quail in its nest.

At times, I picture a barren land
Where the wind blows, unhindered,
Then stops and looks, bewildered,
At the beautiful dunes of sand.

***

If I were a geographer
I would draw many a line
Linking every place to mine
Just the way I prefer.

Tell me, are you chasing a dream?
Or chased, looking for a refuge?
Are you fleeing a fire, a deluge?
Or acting on a providential scheme?

In any case, your visa's granted
With an unlimited permit of stay.
I declare, starting from today
That no one will ever be stranded.

Kais Siala, January 2020

Chapter 1

Introduction

Climate change mitigation is a challenge that requires acting on different fronts, in particular by overhauling existing energy-related policies. In order to provide evidence-based recommendations for policy-makers, energy system modelers rely on mathematical frameworks and computer software to explore different energy system designs and analyze their implications quantitatively (Strachan et al., 2016). The model frameworks require the simplification, abstraction and discretization of the physical world, in particular of space.

The spatial dimension is at the core of this thesis. After describing how space is commonly modeled in energy systems and in the related field of geography, I focus on the challenges of space modeling and identify the research gaps which I address throughout this work.

1.1 Space in geography and energy system modeling

The study of space has always been the main focus of the science of geography. Therefore, it is certainly useful to see how geographers model space, and compare it to spatial analysis in energy system models. In their book on Economic and Social Geography, Knowles and Wareing (1981) provide a comprehensive review of space modeling in geography, which I will mainly rely upon in the following sections, in addition to other sources.

Space in geography

Knowles and Wareing (1981, pp. 28–30) describe the evolution of geography over time in its relationship with space. They identify three distinct phases, which Wardenga (2002) (as cited by Fögele (2017, p. 65)) coined *space as a container*, *space as a system of relations*, and *space as a category of perception.*

Space as a container: According to Knowles and Wareing (1981), determinism was the main paradigm in geography up until the 1930s. It assumes that the physical conditions produce the human responses, so the aim of geographers was to identify the *natural* or *formal* regions[1], i.e. homogeneous regions with respect to a particular characteristic. Although determinism was gradually discredited, the main objective of geographers until the 1950s was the identification of formal space divisions.

[1]Knowles and Wareing (1981) define regions as the distinctive units or areas that divide space. Hence, the terms *spatial units*, *areas*, and *regions* can be used interchangeably.

Later, the focus of geographers shifted to *functional* units[2], i.e. units that operate as an integrated functioning whole. Knowles and Wareing (1981) highlight cities and the regions around them as a perfect example. The recent publication of Berdegué et al. (2019) proves that geographers are still interested in techniques to identify functional units based on urban areas.

Space as a system of relations: According to Knowles and Wareing (1981), the theories behind complex systems have influenced geography since the 1960s. They consider that space, despite its continuity over the surface of the earth, can be regarded as a system made up of many different subsystems. The differences between the subsystems emerge from particular physical and economic characteristics, for example. There is clearly a strong connection between this approach and von Bertalanffy's General System Theory, which is corroborated by Fögele (2017, p. 62). Geographers were then interested in the spatial organization of space, searching for order in the complex system and explaining locations and distributions.

Space as a category of perception: By the 1980s, the focus shifted to the analysis of the spatial structure of human behavior and how the perception of space affects it. The new trend was influenced by behavioral sciences, which emphasize the complexity of perception and behavior, framing them in probabilistic rather than deterministic terms (Knowles and Wareing, 1981).

Space as a construct: Wardenga (2002) completed the list with a more recent paradigm, namely *space as a construct.* Until the early 1980s, it was implicitely assumed that space existed in reality, whether this reality was objective (first two phases) or subjective (third phase). But then geographers started to analyze how different social systems (politics, economy, law, arts, science) view space, how each social system abstracts it and communicates about it, and for which purpose it "constructs" it. The main idea is that actions and communication do not occur in space, but actually shape it, so that it is part of them. Thus, regions can be understood as spatial structures of society produced by its collective actions (Wardenga, 2002).

Space in energy system modeling

Having the previous four concepts of space modeling in mind, I will describe how space is modeled in energy systems and discuss how this relates to geography.

Since its inception in the second half of the twentieth century, energy system modeling has mostly relied on political and administrative boundaries to divide the spatial scope. In a sense, this is similar to the concept of *functional units*, albeit the term has a different meaning. For energy system modelers providing policy advice, countries or their administrative subdivisions form functional units because energy policies are usually designed and rolled out at that level. In geography, however, the term refers to spatial agglomerations, because (human) geography focuses on human settlements in cities and suburban areas. The two definitions do not necessarily lead to overlapping regions, as noted by Berdegué et al. (2019).

There are numerous examples of energy system models that use political and administrative divisions as their spatial units. The European Commission relies on different models, such as JRC-EU-TIMES (Simoes et al., 2014), METIS (Kanellopoulos et al., 2019), POTEnCIA (Mantzos et al., 2017), and JRC-GEM-E3 (Rey Los Santos et al., 2018), which, despite the diversity of their techniques and objectives, use countries or groups of countries as their spatial units. The open-source model framework OSeMOSYS (Howells et al., 2011) has a model version for the EU countries plus Norway and Switzerland (OSeMBE), beside other versions for South America (SAMBA) and

[2]Not to be confused with functional units in life-cycle assessment, which measure the function of a system to enable the comparison of different systems

Africa (TEMBA), which use countries as model regions. Even for renewable potential analyses that do not require optimization or simulation models, energy system modelers often opt for the administrative divisions as spatial units. For example, Rauner et al. (2016) searched for hot spots of renewable power provision in Germany on the level of municipalities (>11000 regions), districts (>400 regions) and federal states (16 regions).

The practice of using political and administrative divisions in the context of energy system modeling is so common that modelers are usually content with a short mention of the spatial resolution in scientific reports and publications, without an explicit justification of their choice. Such a justification is usually reserved for the studies where political boundaries are not used.

This has not prevented energy system modelers from using other spatial resolutions. Looking at how the spatial aggregation of renewable energy sources affects the share of solar PV and wind, Krishnan and Cole (2016) used the ReEDS model of the USA with its native resolution of 134 load balancing areas, beside the aggregation at the state level (48 administrative divisions) and at the level of the thirteen members of the North American Electric Reliability Corporation (NERC). Technically, the balancing areas and the NERC-regions are neither political nor administrative divisions in the strict meaning of the terms, but they can be interpreted as functional units from the perspective of the power system operators. On a smaller scale, Candelise and Westacott (2017) focused on the integration of PV in South-West-England and divided the geographic scope into 1888 small communities of about 600 households, so called Lower Layer Super Output Areas. The spatial units can be smaller, even for large study areas. For instance, Perpiña Castillo et al. (2016) assessed the potential for solar power generation in the 28 EU countries at a resolution of 1 km^2. The modelers usually justify their choice of other spatial resolutions by the need to model solar PV and onshore wind capacity factors more accurately on a local scale (Krishnan and Cole, 2016), by the impacts of their integration into the distribution grid (Candelise and Westacott, 2017), and by the localized land restrictions that reduce their potential (Perpiña Castillo et al., 2016).

The concept of space as a system of relations is applied in models where the spatial organization of the energy system components and the distance between them are particularly important. This is the case for models that co-optimize generation and transmission capacity expansions, and for models that yield generic designs of the networks at different scales. Urban planning relies on this concept to ensure that goods and services are within reach of city inhabitants and would remain so under future growth scenarios.

Urban and regional planning is also the area where perception is becoming increasingly relevant for energy system models, as shown in the following two examples. First, land use planning and zoning are affected by the perception of the impacts of power generating technologies on the environment. Since wind turbines are thought to cause noise nuisance and visual disruption of the landscape (Eichhorn et al., 2019), many city planners have excluded these technologies or have allocated remote zones for their development, which should be reflected in the energy system models. Second, the optimal siting of electric charging stations within and between urban settlements may reduce the range anxiety that is preventing the wide adoption of electric vehicles, which are crucial to the decarbonization of the transport sector (Guo et al., 2018).

Figure 1.1 shows the different phases of space modeling in both fields and how they evolved over time. When comparing the ways geographers and energy system modelers divide space, it appears that:

- There are probably no or very few models that have used natural or formal regions, since that approach was already discredited by the time the field of energy system modeling emerged.
- Both sides have used functional units, albeit the term has two different meanings. In geography, it refers to spatial agglomerations, whereas for energy system modelers providing policy advice it usually means political and administrative units. The two definitions do not necessarily lead to overlapping regions, as noted by Berdegué et al. (2019).

- The concepts of space as a system of relations and as a perception were adopted in both fields at about the same time.
- The concept of space as a social construct can be applied in geography to analyze how other fields construct space, yet it has no direct application in quantitative energy system models.

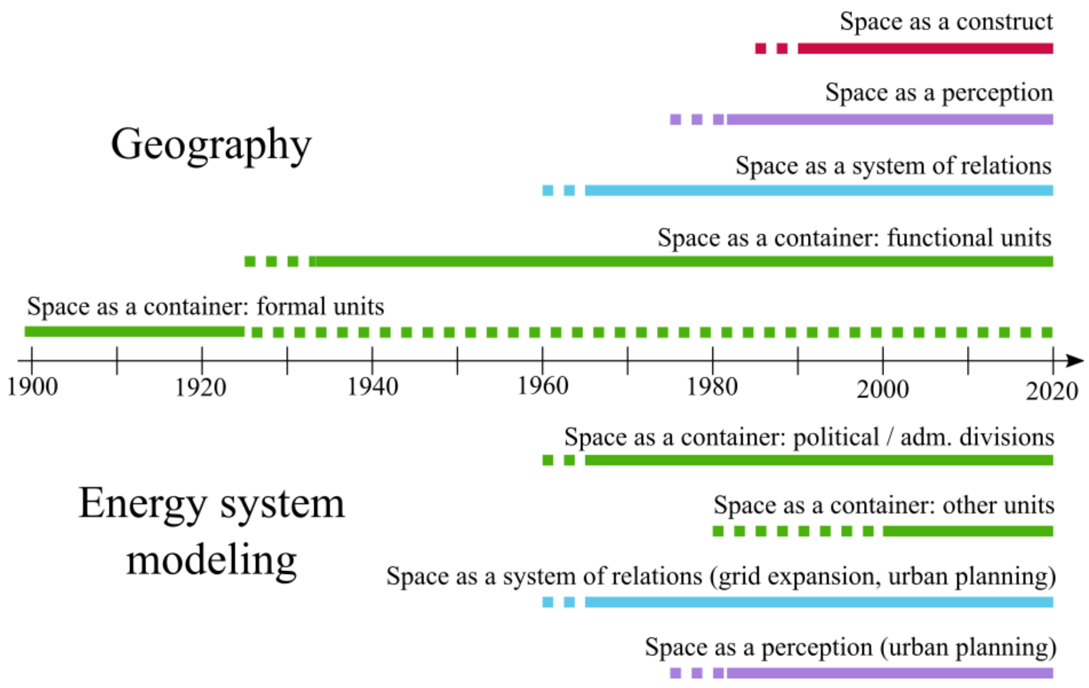


Figure 1.1: Overview of the evolution of the different concepts of space modeling in geography (top) and in energy system modeling (bottom). The dotted lines are used to mark the approximate period of time where the concepts emerged or were discredited, whereas the continuous lines mark the periods of time where the concepts were continuously used.

Despite the lack of a direct application of the last concept in energy system modeling, it can be used as an invitation for self-reflection: What does the way energy system modelers divide space say about the way they view it or want the rest of society to view it? Based on the previous examples of models and studies, it is clear that there is a strong emphasis on the political, administrative, and operational aspects of the spatial divisions. I can identify at least two weaknesses in this approach:

Weakness 1: The reliance on the same spatial units in most models and studies does not question the validity of this modeling choice. However, it is not clear whether the model results would change significantly by varying the number of the spatial units or their boundaries, or by using models at different scales and coupling them. Due to the importance of energy system models in shaping energy policies with direct implications on the environment and society, it is crucial to ensure the robustness of their results.

Weakness 2: There is no doubt that the political and operational aspects are important, especially since the models are used by policy-makers, operators and other stakeholders to guide their decisions. However, by using these spatial units, other aspects such as geographic characteristics are *de facto* neglected or relegated to a secondary role. So, is this prioritization justified? What current trends in the energy system are challenging this view?

The first weakness is related to the issues of choosing the appropriate scale and boundaries, which are also encountered in the field of geography. The latter is related to the different roles or functions of space, and their importance in the context of energy system modeling. I explain these issues in the following sections and, based on them, I derive the key research questions that I address in this thesis.

1.2 Common issues in geography and energy system modeling

Despite their different approaches in defining regions, the fields of geography and energy system modeling share a set of practical issues when dealing with space.

Scales

Viewing the geographic system as a whole that is made of components, which form systems in themselves, allows geographers to study systems at different scales. Knowles and Wareing (1981, pp. 27–28) provide the example of the worldwide transport network which is made of smaller systems that are complete in themselves, such as the British or the European rail systems. A similar classification of energy systems based on their scales is also possible. For the power system, the voltage level of the grid serves as a criterion to select the relevant lines at a given scale: whereas only lines above 220 kV are usually modeled at a European scale, the voltage level may not exceed 400 V for a single building. Hence, the level of the modeling detail varies with the scale of the system.

The scale also affects the way space is abstracted, i.e. the choice of the geometries that are used to represent spatial patterns. Knowles and Wareing (1981, p. 26) propose to use an analogy with the characteristics of geometrical space. Locations can be viewed as *points*, human interactions as *lines* of communication, and the regions where the interactions occur as *areas*. I have extended the analogy to the energy system by providing examples of objects and relations that can be modeled with the three basic geometries, as shown in Table 1.1.

Table 1.1: Spatial patterns in geography and energy system modeling, with the corresponding geometrical abstraction

Geographic pattern	Energy system examples	Corresponding geometry
Specific locations or general distributions	Locations of power plants or substations	Point
Economic and social interactions, flows	Transmission lines, electricity and gas trade, roads, freight routes, etc.	Line
Regions based on formal or functional units	Load regions, suitable areas for renewable capacity expansion, etc.	Area

Note that the choice of the suitable geometry depends on the scope and the scale of the system. At a national scale, the electric load is usually associated to regions, as in Table 1.1. But at the scale of one building, the electrical appliances can be modeled individually as single points.

Boundaries

While it may be convenient for Balta-Ozkan et al. (2015) to use the Oxford dictionary definition of a region as "an area, especially part of a country or the world having definable characteristics but not always fixed boundaries", quantitative analyses using computerized energy system models require clear delimitations of the boundaries between regions. Knowles and Wareing (1981, p. 30) acknowledge that the boundary issue is a challenge to the idealized concept of regions. On the one hand, certain features, such as rainfall, do not change abruptly across the borders of formal regions. On the other hand, fixed physical features, such as rivers, form simultaneously a natural border and the core of a characteristic region including the riverbeds. Even for functional regions, like cities, the delimitation can only be approximate since agglomerations usually overlap. For energy system modelers who rely on political and administrative divisions, the precise delimitation of the regions is usually not an issue. However, the use of political boundaries might affect the modeling of the energy system and distort its results, since the political regions do not necessarily reflect the flows and processes of the energy system in the best way. For the power system, it is recommended to use grid clustering algorithms to generate an equivalent, simplified electric grid and to infer the boundaries of well-connected regions out of it. Cao et al. (2019) and Biener and Garcia Rosas (2020) reviewed common network reduction techniques in energy system modeling. Figure 1.2 provides an example to illustrate how the use of administrative divisions may lead to the underestimation of grid bottlenecks.

Similarities and contrast

As Knowles and Wareing (1981) pointed out, geography has traditionally been concerned with extracting the *unique character* of an area. This has recently shifted to a search for similarities and orderly patterns in a complex system. In energy system modeling, the duality between similarity and contrast has usually not been determinant in defining regions. However, there are studies where the resource discrepancy between groups of regions is at the core of the research question, such as in the Desertec project (Düren, 2017) or in Díaz et al. (2017).

1.3 Trends and challenges specific to energy system modeling

Four decades ago, Nijkamp (1980) (as cited in Balta-Ozkan et al. (2015)) identified five issues related to energy and its interaction with space. These are the need to analyze how changes in the energy sector affect the spatial distribution of activites; to examine and compare regional, economic and technological variations; to explore different policy options; to investigate regional economic and environmental interlinkages; and to analyze the distributional impacts of energy issues (Balta-Ozkan et al., 2015, p. 501). These issues prove that the relevance of the spatial dimension is not recent. However, new trends have emerged that render the topic even more relevant.

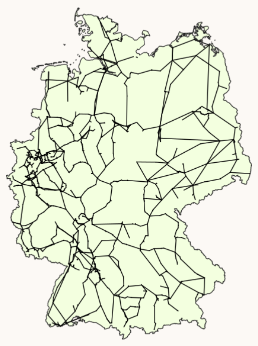
Increase of the share of renewable energy

Pfenninger et al. (2014) cites the provision of high resolution data, both in space and time, as one of the major challenges of energy systems modeling in recent years. The need for high resolution data is particularly driven by the increase of the share of variable renewable energy sources in the electricity sector, which has multiple implications.

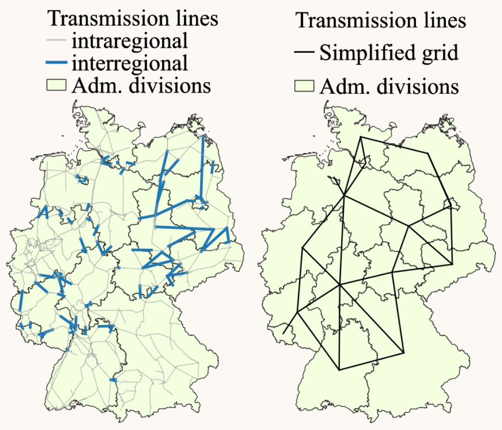
First, wind and solar power plants are usually located in areas with good potentials, which are not necessarily close to the load centers, and usually have a low overall power density per unit of area compared to conventional, centralized power plants. The decentralized nature of renewable

Example: Impact of boundaries on transmission grid modeling

Let us consider the geographic scope of Germany, and the transmission grid data from the open source project GridKit (Wiegmans, 2016). The transmission lines have been pre-processed using the tool pyPRIMA (Siala and Houmy, 2020, see Appendix C). The grid has 2517 nodes and an estimated total capacity of 3753 GVA for all lines. We are going to apply a simple grid reduction technique that only preserves interregional lines.

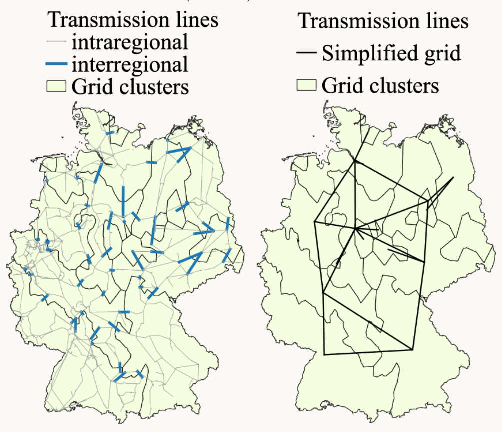

(a) Transmission grid of Germany according to GridKit.

When using the boundaries of the federal states to define 16 model regions, the intraregional transmission lines are ignored and all electricity suppliers and consumers are assumed to be connected to each other over a “copper plate”. By aggregating the capacities of the cross-border lines, we obtain a simplified transmission grid with a capacity of approximately 149 GVA.



(b) Administrative divisions of Germany, with the overlaying grid (left) and the simplified grid (right).

Using the clustering tool pyCLARA (Siala et al., 2020, see Appendix B), the transmission grid can be clustered to obtain model regions that preserve the critical grid bottlenecks. By applying the same aggregation method as before, we obtain a simplified transmission grid with a capacity of approximately 90 GVA. A lower interconnectivity might reflect the difficulty of integrating renewable energy better.



(c) Grid clusters of Germany, with the overlaying grid (left) and the simplified grid (right).

Figure 1.2: Impact of the choice of boundaries on the modeling of the transmission grid in Germany.

power supply and the spatial (and temporal) mismatch with the electricity demand leave a wide range of research questions open, such as whether energy autarky can be achieved at different spatial scales (Tröndle et al., 2019), or whether a combination of pumped hydroelectric storage and transmission lines is the cost-optimal solution (Sinn, 2017).

Second, the scale of the energy transition might have a non-negligible impact on the landscape and on the environment in general (Dijks et al., 2018). This leads to social acceptance issues (Batel, 2018), to the search for fair and socially acceptable distributions of the new energy infrastructure (Drechsler et al., 2017; Masurowski et al., 2016), and to the use of spatial synergies with other sectors of the economy like agriculture (Majumdar and Pasqualetti, 2018). Such analyses would not be possible without the growing body of literature on high resolution renewable energy potentials, such as the work done by Uslu et al. (2009) and more recently by Bosch et al. (2018) on the wind potential, and by Bódis et al. (2019) on the solar photovoltaic potential in Europe. In recent years, several webtools and global atlases of renewable potentials have been created, which are easy and free to use for research and other purposes (International Renewable Energy Agency, 2019; Pfenninger and Staffell, 2016; Solargis, 2019; Technical University of Denmark, 2019). By taking into account technical, environmental and sociopolitical constraints, researchers have attempted to narrow down the resource potential into technically-feasible and socially-acceptable potentials (Ryberg et al., 2018; Watson and Hudson, 2015).

Sector coupling

The decarbonization of the energy system goes beyond the power sector and concerns industry, transport, heating and cooling. There are currently two main strategies to achieve this: either through maximal electrification, where most of the final energy demand is electric, or through a mix of technologies that can deliver gas and liquid fuels beside electricity (Klein et al., 2017). Technologies such as heat pumps and battery electric vehicles are already enabling the electrification of the other sectors directly, whereas power-to-gas and power-to-liquid technologies will unlock the potential of indirect electrification, since they could use energy from renewable power plants to operate electrolysers and break water molecules into oxygen and hydrogen, a key element in synthetic fuels. In both cases, achieving a carbon-neutral energy system will require the expansion of renewable energy at a large scale. Ruhnau et al. (2019) reviewed multiple studies on the decarbonization of the German system and concluded that 400 TW h to 800 TW h are needed in addition to the current electricity consumption to cover the demand for heating and road transportation. Klein et al. (2017) included industrial feedstock, maritime transport and aviation, and estimated that around 2000 TW h of renewable power is needed, beside biomethane and geothermal energy. This is equivalent to 4–5 times the estimated base electricity demand of Germany in 2050. From the spatial perspective, an expansion of this magnitude will require the use of 2% of the area of Germany for onshore wind and utility-scale solar photovoltaic projects according to Klein et al. (2017). This again amplifies the importance of modeling the potential of these technologies with a high spatial resolution, to identify the best available project sites and minimize their spatial footprint.

Sector coupling poses another challenge from the spatial perspective. The different energy sectors have characteristics which cannot be modeled at one spatial scale. On one hand, heat demand for residential buildings can only be supplied economically through local heat generation, either at the building itself or at the district level if district-heating networks exist. On the other hand, energy-intensive industries cannot cover their energy requirements locally through decentralized renewable power production, hence they rely on power and gas transmission networks to connect them with utility-scale suppliers. Furthermore, modeling the transport sector can be on the scale of cities, countries, or the whole globe, depending on the mode of transportation. At each scale, there is a multitude of stakeholders. Thus, energy system modelers need more flexibility in designing their models to capture processes and flows at each spatial scale. Only with flexible spatial units that adapt to their scope can they represent the emerging trends in energy supply, as exemplified by the cellular approach of Benz et al. (2015) and Amado et al. (2017).

Data and computational power

The trend towards models with a high spatial resolution and which are capable of representing multiple energy sectors at different scales is partly slowed down by the unavailability of data or by limited computational power.

Energy system models are indeed data-intensive. Some of the data, such as transmission and distribution grids, is usually proprietary and confidential. Recently, there have been growing efforts from statistical offices and system operators to publish data publicly and freely. Yet, as Hirth (2020) has noted, the usage rights are not guaranteed unless an adequate open data license is provided, which is rarely the case. Besides, even if open data is available, it is usually at a low resolution and it requires some processing before it can be used in energy system models. One trend to follow in the near future is the proliferation of smart meters, which can collect high resolution data of high quality. However, whether energy system modelers will have access to this data remains doubtful.

The increase of the computational power since the 1960s has enabled the use of multi-regional energy system models to solve increasingly complex problems. Yet the modeling of multiple energy sectors with high spatial and temporal resolutions under uncertainty requires computational power that may exceed the capabilities of computers at the disposal of the modelers. Hence, many studies have attempted to identify the smallest number of regions for which the errors that arise from the simplification are acceptable. Hörsch and Brown (2017) clustered the European power grid and varied each time the number of regions, then compared the results in terms of installed power plant capacities, transmission grid expansion, and total system costs. They concluded that a higher number of regions improves the depiction of the potential of wind energy and of grid bottlenecks. Anderski et al. (2015) started from small administrative divisions of Europe, then reduced their number by merging regions heuristically within each country, so that the grid bottlenecks are preserved. Cao et al. (2019) classified and reviewed different techniques for speeding up energy system models, including the reduction of spatial units. They observed the evolution of computational indicators (number of equations, solving time, memory usage, etc.) and accuracy indicators (energy sources, storage, transmission, etc.). For the case of Germany, they observed that the accuracy of the results stabilizes above 100 regions, except for the transmission capacity, which is underestimated by more than 50% at that scale.

1.4 Research objectives and contributions

Research gaps

In light of the aforementioned literature on the issues faced by energy system modelers, it is worth reconsidering the two identified weaknesses in Section 1.1. The experiments involving a different number of spatial units came to the conclusion that the model results depend on the number of regions. Consequently, the choice of the spatial resolution matters. Moreover, the blanket assumption that models using political or administrative divisions (countries, federal states, etc.) are valid regardless of the specific research question does not hold true. Modelers are required to identify the minimum number of regions for which the obtained results are robust. This implies the ability to flexibly choose the spatial units in the model. But what the studies have not proved yet is that even for a fix number of regions, the spatial units may be shaped differently, and this may also impact the results. In fact, varying the number of regions may not be an option if the computational power is limited or if the data quality is uncertain at a high resolution, but varying their shape could still be an option.

Regarding the second weakness, it is clear that the political and operational aspects should not be the only criteria that matter when choosing the spatial units in energy system models. Previous studies have proven that the administrative units do not reflect the potential of renewable energy or the grid restrictions adequately. Meanwhile, geographic aspects such as renewable energy resources, land use constraints, and the spatial distribution of the energy demand are increasingly gaining importance. The relevance of these aspects differs depending on the research question, and should be reflected in the choice of the spatial units. However, the existing literature does not provide any systematic approach on how to reconcile all these aspects when defining the model regions.

Problem statement

For a given geographic scope, there is a large (in fact, infinite) number of ways of dividing it into discrete spatial units for energy system modeling. My hypothesis is that not all the different ways are equivalent, since they model aspects of the energy system (renewable energy supply, grid, demand, etc.) differently, which might be reflected in the results. Among all the possible options of dividing the spatial scope, and for each research question under certain constraints, there must be at least one spatial resolution that is *optimal*, given an objective function defining such optimality.

Therefore, the central research question I will answer throughout this thesis is: **how to reach the optimal spatial resolution for energy system modeling?** For that, I will go through the following steps:

1. Suggest a definition for the optimal spatial resolution,
2. Propose a set of methods to obtain the optimal spatial resolution systematically,
3. Provide case studies where the proposed methods are applied,
4. Justify quantitatively why the choice of spatial units matters.

Scope

The emphasis on optimality implies that the choice of spatial units is not arbitrary. In this thesis, I explore different ways of modeling the spatial scope of a given research question. Due to the increasing importance of renewable energy, I experiment in one of my publications with spatial units that are based on solar and wind potentials, and use them as model regions for an optimization of the European power system. The use of these natural, formal units, which used to be common in geography, is novel in energy system modeling. In the same experiment, I define spatial units which are based on the yearly energy demand. Some of them are a reflection of the urban agglomerations that geographers use as their functional units. In order to obtain these spatial units, I developed a new algorithm for clustering high resolution geographic data. By design, the algorithm searches for the similarities between the regions while ensuring the contiguity of the spatial units. It can be applied on different types of maps and is not only restricted to the power sector, so it is also suitable for studies on sector coupling.

The clustering requires the existence of high resolution data as an input. This is a crucial aspect of my work, as I believe that energy system modelers should be able to flexibly define their spatial units, regardless of the format of their input data. Therefore, I propose a methodology to convert the spatial resolution of any input data into the desired resolution of the energy system model. The conversion of low resolution data into a user-defined resolution usually goes through an intermediate step of high resolution data. I included a publication where I used such techniques to obtain high resolution potential maps for solar photovoltaics in South-East Asia.

Going from the high resolution data to a coarse, user-defined resolution is achieved through aggregation techniques. Theoretically, any scale that is larger than the smallest unit in the high resolution data is possible. However, the computational power could be a limiting factor here. Unlike previous studies, I did not focus on how results improve with a larger number of regions. Instead, I set a fix number of regions and focused on how the results vary depending on the shape of the regions. I thereby prove that the shape of the spatial units should be chosen carefully. By testing multiple spatial units, I perform a spatial sensitivity that identifies robust results, which are independent of the boundaries of the regions.

Finally, I expand the method to cover the cases where one model cannot answer the research question on its own, either because of its limited geographic scope or because it cannot model all the necessary scales and/or sectors. I therefore include a publication on model-coupling between a European power system optimization model and a global input-output table of the whole economy, as an example.

There are different types of energy system models, and some of them may benefit from this thesis more than others. In most of my publications listed below, I used optimization models for expansion planning and economic dispatch for the power system, and this has informed my work in this thesis. The different techniques for pre-processing the data in order to generate models with the optimal spatial resolution are applicable for such models. Nevertheless, I do not restrict the scope of the thesis to that type of models only. Models that optimize the exact siting of renewable power plants can benefit from the techniques that generate high resolution technical potential data, whereas power flow models that have a very detailed power network can use the methods that disaggregate the load data to allocate it to the individual grid nodes. Energy system modelers that focus on other sectors can rely on existing analogies with the power sector to replicate the methodology. Last but not least, the chapter on model coupling expands the set of possibilities to a wide range of models beyond the scope of energy modeling.

Publications

This thesis includes three peer-reviewed journal publications, listed here in the order of appearance:

1. K. Siala and J. Stich. Estimation of the PV potential in ASEAN with a high spatial and temporal resolution. *Renewable Energy*, 88:445–456, 2016. doi: `10.1016/j.renene.2015.11.061`.

2. K. Siala and M. Y. Mahfouz. Impact of the choice of regions on energy system models. *Energy Strategy Reviews*, 25:75–85, Aug 2019. doi: `10.1016/j.esr.2019.100362`.

3. K. Siala, C. de la Rùa, Y. Lechòn, and T. Hamacher. Towards a sustainable European energy system: Linking optimization models with multi-regional input-output analysis. *Energy Strategy Reviews*, 26:100391, 2019. doi: `10.1016/j.esr.2019.100391`.

Other publications related to high resolution data and potential estimations using geographic information systems (GIS), in chronological order:

4. K. Siala and A. Molar-Cruz. Probabilistic GIS-based technical potential assessment of hydrothermal energy in Bavaria. In *Der Geothermiekongress 2017*, Munich, Germany, 2017. URL `https://mediatum.ub.tum.de/1395135`. Last viewed: June 1, 2020.

5. K. Siala. Increasing the spatial resolution of MERRA-2 reanalysis data for energy system modeling. In *European Geosciences Union General Assembly*, 2019a. URL `https://mediatum.ub.tum.de/doc/1506225/590329440273.pdf`.

6. K. Siala. Flexible spatial distribution of electricity demand for energy system models. In *38th International Energy Workshop*, Paris, France, 2019b. International Energy Agency.

URL `https://mediatum.ub.tum.de/doc/1506227/204630667544.pdf`. Last viewed: June 1, 2020.

Previous publication on the choice of spatial units, including preliminary results:

7. M. Y. Mahfouz and K. Siala. Impact of the Choice of Regions on Energy System Models. In *Energy Modelling Platform for Europe (EMP-E) 2018*, Brussels, 2018. TU Wien. URL `https://mediatum.ub.tum.de/doc/1464189/370483268925.pdf`. Last viewed: June 1, 2020.

Other publications on model coupling, in chronological order:

8. I. Boiarchuk, K. Siala, D. Hewes, R. Witzmann, and T. Hamacher. Investigation of the Performance of Nodal and Zonal ENTSO-E Transmission System Models in the Context of Large-Scale Integration of Renewables. In *Conference on Sustainable Energy Supply and Energy Storage Systems*, pages 289–295, Hamburg, 2018. VDE Verlag. ISBN 978-3-8007-4445-9.

9. K. Siala and C. de la Rùa. How are the benefits of Europe's decarbonization distributed? In *IAEE International Conference*, Montreal, Canada, 2019. International Association of Energy Economics. URL `https://mediatum.ub.tum.de/doc/1506226/716140601646.pdf`. Last viewed: June 1, 2020.

Other publications related to energy policy and public outreach, in chronological order:

10. T. Hartmann, K. Siala, P. Kuhn, M. Huber, L. Stolle, and T. Hamacher. Gesicherte Stromversorgung in Bayern [Reliable Power Supply in Bavaria]. Technical report, Bayerisches Staatsministerium für Wirtschaft und Medien, Energie und Technologie, Munich, Germany, April 2016.

11. S. Baur, J. Winklmaier, and K. Siala. Beitrag interdisziplinärer Hands-On-Projekte in der schulischen und universitären Lehre zur nachhaltigen Energiegewinnung in Entwicklungsländern [Contribution of interdisciplinary hands-on projects in secondary and tertiary education on sustainable energy supply in developing countries]. In *Technische Bildung im Spannungsfeld zwischen beruflicher und akademischer Bildung - Die Vielfalt der Wege zu technischer Bildung - 11. Ingenieurpädagogische Regionaltagung 2016*, pages 133–138. Ingenieur-Pädagogische Wissensgesellschaft - Technische Universität Hamburg-Harburg, 2017. ISBN 978-3-9818728-0-4.

12. K. Siala, E. Baik, M. Huber, T. Hamacher, and S. M. Benson. Optimizing the Californian Power System according to the Renewable Portfolio Standards for 2030 and beyond. In *37th International Energy Workshop*, 2018. URL `https://mediatum.ub.tum.de/node?id=1453494`. Last viewed: June 1, 2020.

13. G. Savvidis, K. Siala, C. Weissbart, L. Schmidt, F. Borggrefe, S. Kumar, K. Pittel, R. Madlener, and K. Hufendiek. The gap between energy policy challenges and model capabilities. *Energy Policy*, 125:503–520, 2019. doi: `10.1016/j.enpol.2018.10.033`.

14. S. Candas, K. Siala, and T. Hamacher. Sociodynamic modeling of small-scale PV adoption and insights on future expansion without feed-in tariffs. *Energy Policy*, 125:521–536, 2019. doi: `10.1016/j.enpol.2018.10.029`.

Source code repositories

In order to facilitate the pre-processing steps and enable other energy system modelers to flexibly define their spatial units, following the recommendations of this thesis, I shared my scripts as open-source code with an exhaustive documentation. I am the main author and maintainer of the following GitHub repositories:

- **tum-ens/pyGRETA** (**py**thon **G**enerator of **RE**newable **T**ime series and m**A**ps): A tool that generates high-resolution potential maps and time series for user-defined regions within the globe. See Appendix A for the documentation.
- **tum-ens/pyCLARA** (**py**thon **C**lustering of **L**ines **A**nd **RA**sters): A tool to cluster high-resolution spatial data (rasters or polylines connecting Voronoi polygons) into contiguous, homogeneous regions. See Appendix B for the documentation.
- **tum-ens/pyPRIMA** (**py**thon **PR**eprocessing of **I**nputs for **M**odel fr**A**meworks): A tool to automate the creation of energy system models using a common database. See Appendix C for the documentation.

1.5 Outline

This thesis is based on three peer-reviewed publications. The first two chapters cover the theoretical and methodological aspects of the thesis, while the following ones include the publications as three applications.

In Chapter 2, I start by defining the optimal spatial resolution. I propose an approach to reach that resolution in Chapter 3, where I also explain the different techniques for increasing, decreasing and combining the spatial resolutions. Chapter 4 includes a publication on the potential estimation of solar photovoltaics, and serves as an application for techniques of increasing resolutions. Chapter 5, on the impact of choice of regions on energy system models, introduces a novel clustering method of high resolution data and provides an example of methods for decreasing the resolution. In Chapter 6, I provide an application of coupling two models with different scopes and capabilities to answer a research question on the distributional effects of decarbonization. Finally, I discuss the outcomes of my work in Chapter 7 and conclude in Chapter 8.

Chapter 2

Optimal spatial resolution

The spatial resolution of an energy system model describes the number and shape (or nature) of the spatial units used in the model. In order to define optimality in spatial resolutions, I first describe the range of technically possible spatial resolutions. Then, I narrow it down gradually to reach what energy system modelers can practically achieve and what is suitable for their research questions.

2.1 Physical space and modeling complexity

Manson (2001) (as cited by Priesmann et al. (2019)) divides complexity research into *aggregated complexity* (focusing on overall system behavior), *deterministic complexity* (based on chaos and catastrophe theory), and *algorithmic complexity* (focusing on the effort required to solving mathematical problems). In the context of energy system modeling, Priesmann et al. (2019) define the complexity of the real-world energy system as *aggregated complexity* where decision-making agents and technology interact within physical and social networks, leading to dynamics, self-organization, path-dependency, emergence, co-evolution, as well as learning and adaption. They use the term *computational complexity* to denote the required effort for solving mathematical problems in energy system models. In fact, computerized energy system modeling is a process that involves the description of the states of a finite number of entities (technologies, agents, etc.) that exist within the scope of the system using discrete spatial and temporal units, with a degree of uncertainty. Thus, the computational complexity of an energy system model depends on the number of technologies and other entities that it describes, how detailed their mathematical description is (linear, nonlinear, etc.), the spatial and temporal resolution, and the uncertainty with which all this information is collected and processed. In short, the model complexity is a combination of *technological*, *spatial*, and *temporal* complexities under *uncertainty.*

The aspect of uncertainty provides the first clue on how to determine the technically possible discrete spatial units for energy system models. Although energy conversion processes may occur at the scale of molecules or ions, energy system modelers do not observe these processes for each single particle. Instead, they observe the overall balance whenever they perform a measurement of voltage and current on appliances (for electricity) or temperature and heat flow (for heat). The analogy applies to other sectors of the energy system as well. To limit the uncertainty and verify the validity of the model, the selected spatial unit has to be technically measurable. Hence, the smallest observable object in the energy system is its *elementary particle* and its size sets the minimum threshold for the scale of the spatial unit in the system model, because the states of smaller units cannot be validated. In the following, I use the scale of an electric household appliance (1×10^{-1} m to 1 m) as the smallest practical spatial unit for energy system modeling, since its state (e.g. voltage and current flow) can be determined with low uncertainty using commercially available electricity meters.

Figure 2.1 shows the range of possible spatial units in energy system models, for which it is technically possible to determine the state using common measuring techniques in the energy field. The top of the figure depicts the size of the geographic scope of five hypothetical energy system models. If the energy system is at the level of a building, the minimum number of units occurs when the model uses the building as one single unit. A 10 m × 10 m × 3 m building could have up to ≈3×10^5 spatial units the size of a light bulb if it is modeled in 3D. On higher levels, such as neighborhoods, cities, countries or continents, the maximum number of spatial units increases considerably.

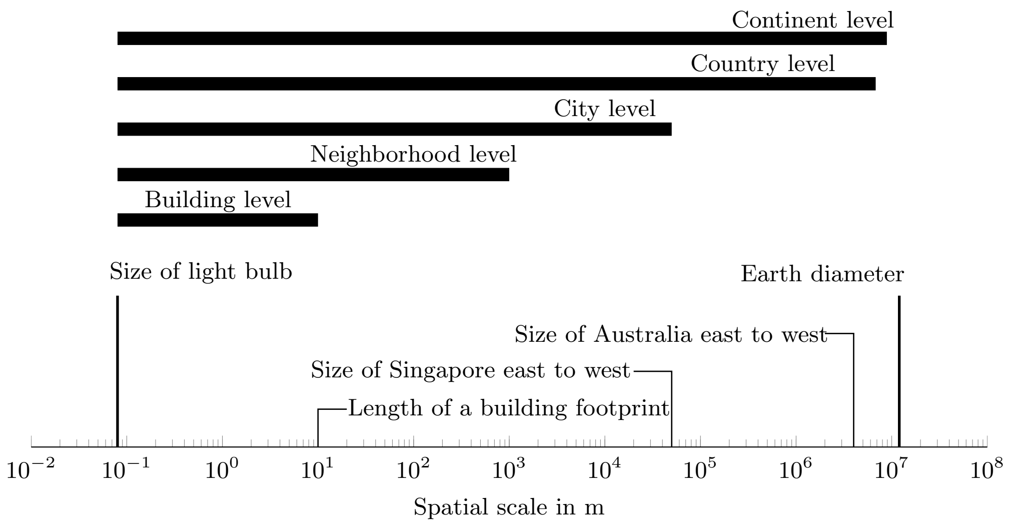


Figure 2.1: Range of technically possible discrete spatial units for energy system modeling, for which system states can be observed with low uncertainty

Many issues arise when using the technically possible maximum number of units:

- Although it is possible to collect measurement data on each appliance within a large area (e.g. country), such a process is costly due to the number of required measuring devices.
- This approach is also hardly feasible because it requires the processing of big data including personal information, which is legally contentious.
- Without measured data at the appliance level, the uncertainty of the model using the smallest spatial unit increases.
- The modeling of the individual appliances for systems with a large spatial scope might be unnecessary and irrelevant for the research question.

In order to overcome these issues, it is necessary to constrain the size of possible spatial units further by taking the aforementioned aspects into consideration. In the following sections, I explain how the research question, the computational capacity, and the trade-off with uncertainty can be used to systematically derive the optimal spatial resolution.

2.2 Characteristic properties of space

In an energy system model, a spatial unit, commonly referred to as model region or node, has a clear function: it sets an abstract boundary within which model relationships apply. In an optimization model for example, certain variables and constraints are defined for specific regions, whereas others apply for the whole system.

In the physical world, spatial units within the scope of the energy system may have various characteristics or features that define them. The next paragraphs contain a non-exhaustive list of characteristic properties that could be relevant for energy system modeling, followed by two illustrative examples on pages 19 and 20. The list is mainly inspired by the different sub-branches of physical and human geography.

Geophysical property: Spatial units can be grouped based on homogeneous geophysical properties. In topographic maps for example, isolines delimit areas with similar elevations. The range of possibilities is as broad as the field of physical geography itself: Climatology can help define spatial units with similar wind and solar potentials, geology is relevant for the study of geothermal energy, and biogeography can be used to define spatial units based on land use types.

Demographic property: Population and settlement geography can serve to determine areas with homogeneous demand properties, such as urban, suburban and rural areas.

Political property: The political property is a special case of the geographic property, related to political geography. Here, spatial units correspond to administrative divisions, such as political groups of countries, individual countries, states or provinces, municipalities, and districts or boroughs. In energy system modeling, this spatial division is relevant if there are policies that are only enforced at the level of the political units.

Geometric property: A spatial unit can have particular geometric characteristics regarding its shape (rectangles, hexagones, triangles, etc.) and its size (in units of length, area, or volume).

Operational property: A spatial unit with an operational property lies within the domain of an operator and is subject to operation constraints. In energy systems, this is the case for the domains of transmission and distribution system operators, balancing authorities, and energy utilities, to name a few examples.

Economic property: The economic property is relevant when the spatial unit corresponds to a market. Trade may be constrained between different markets, but no constraints should exist for market participants (suppliers and consumers) within a single market.

In order to answer a research question adequately, energy system modelers should identify the relevant aspects and translate them into spatial units with characteristic properties. The optimal spatial resolution is one that takes into account all the relevant aspects for the scope of the study. Such a process is trivial if there is only one relevant aspect, but gets complicated if more aspects with non-identical spatial units are considered. In the case of a two-dimensional scope, I recommend using the following algorithm to identify the optimal spatial resolution:

1. Define all relevant aspects to the research question.
2. Translate each aspect into a layer covering the scope of the study, and identify the spatial units within that layer according to their characteristic property.
3. Differentiate between spatial units that are divisible, and those that are not. A spatial unit is divisible if its parts inherit the same property after the division. For example, if a technology ban is enforced within a spatial unit, then it is also enforced within each of its parts. The same does not hold for a spatial unit where a technology quota exists.

4. Superpose all the layers and observe their intersections. The layers are compatible only if the boundaries of the spatial units overlap or lie within a divisible spatial unit.
    a) If all layers are compatible, then the spatial units derived out of their intersection are the smallest possible set of model regions for the research question. It is possible to find an optimal spatial resolution for the problem.
    b) If at least two layers are not compatible, then move to the next step.
5. Prioritize the conflicting layers based on the importance of their properties.
6. Repeat the following steps for all conflicting layers:
    a) Check whether it is possible to model the property of the spatial units in the layer differently, such as an attribute for example, instead of a model region on its own.
        i. If that is possible, then perform the conversion and ignore that layer.
        ii. If that is not possible, move to the next step.
    b) If there are no more conflicting layers, go back to step (4). Otherwise continue to the next step.
7. If there are still two or more conflicted layers, then it is not possible to find an optimal resolution for the research question. However, there are at least two approaches to find the closest spatial resolution to optimality:
    - Ignore layers whose property have a low priority and take into consideration the error that might arise from this simplification.
    - Create different models, each one of them ignoring one of the conflicted layers, and use them as a sensitivity analysis of the model.

Following the previous algorithm, the modeler would obtain an eventually large set of spatial units that can be used as model regions. The spatial resolution would be optimal for the research question, or close to optimality. However, the availability of accurate inputs to feed into the energy system model and the computational capability of running it might add further constraints to the problem.

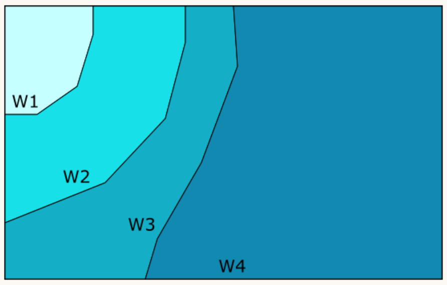
2.3 Accuracy and computational complexity

Energy system models rely on several inputs which are not always provided in a high spatial resolution, or in a resolution that is compatible with the optimal resolution for the research question. Section 3.2 in the next chapter lists some methods to increase the spatial resolution. Whereas the original inputs might have been obtained through reliable sources that performed measurements and checked the validity of the values, the values obtained through algorithms that increase the resolution are less accurate and might lack validation. In fact, the statistical estimation errors increase with increased resolution, as Böhme (2013, p. 116) showed for heat consumption data of buildings in Oldenburg, Germany. Thus, there might be a trade-off between the flexibility in using the optimal resolution and the uncertainty affecting the input data. The best solution is to search for alternative, reliable high resolution input data. If that is not an option, then the modeler has to choose between sacrificing the spatial resolution that is recommended for the research question and applying the algorithms to transform the inputs and increase their uncertainty. The decision should be based on the estimated error that would occur in either case.

The computational power that is at the modeler's disposal could also be a major limiting factor. In fact, the number of model variables and constraints increases if the number of model

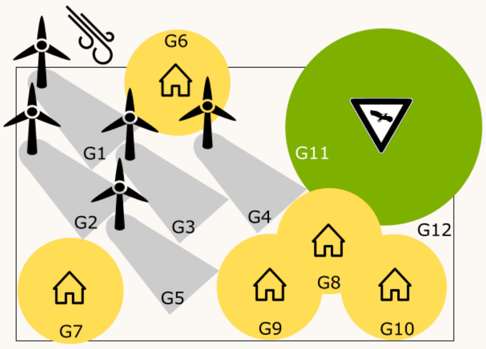
Example: Wind potential in a district

Geophysical property: For the resource potential of wind, information about the wind speed is crucial. For the scope of this example, four different wind classes exist.



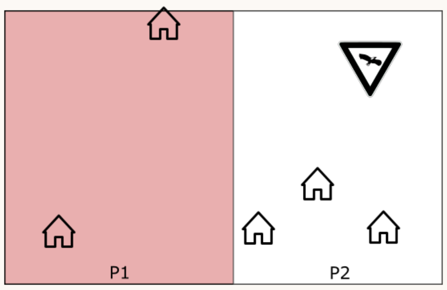
(a) The spatial units W1–W4 represent the four different wind classes

Geometric property: When estimating the technical potential of onshore wind in an area where the turbines should be placed at a certain distance from dwellings or protected natural reserves, it is possible to use circles around the excluded sites with a radius equal to the minimum distance as a buffer. And if each wind turbine has a minimal area requirement with a special shape, that shape can be used as a pattern to identify the maximum number of wind turbines that can be built.



(b) Schematic representation of the problem of placing wind turbines. The spatial units G1–G11 have a geometric function, whereas G12 covers the rest of the scope

Political property: The district is made of two municipalities, which have different policies toward wind power. The municipality P1 encourages wind expansion through tax exemption, whereas municipality P2 levies a high building permit fee.



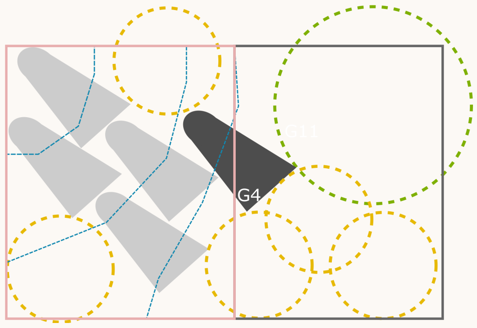
(c) The spatial units P1 and P2 represent two municipalities

Figure 2.2: Relevant spatial functions for the problem “Wind potential in a district” (steps 1–2).

Example: Wind potential in a district – continued

Divisibility: The spatial units W1–W4, G6–G12, and P1–P2 are divisible, because their parts can inherit their properties. This is not the case for G1–G5, which are not divisible.

Compatibility: The units G1–G5 need to be checked for compatibility with the other layers. Despite the fact that they intersect with W1–W4, this is not an issue because the potential is determined by the exact location of the turbines, so the geometry of their wake is irrelevant. The units G1, G2, G3, and G5 lie completely in P1, but G4 lies partially in P1 and P2, which may cause a conflict in the modeling.



Only the spatial unit G4 may cause a conflict because it is not divisible, yet it lies on P1 and P2 at the same time

Prioritization: By choice, the layer with the geometries has a higher priority than the layer of the political boundaries.

Conversion: It is possible to convert the information from the political layer into a cost attribute for the wind turbine in G4. Its investment cost is partly subsidized by the share of its area in P1, but that is offset by the higher permit fee for taking space in P2.

Further conflicts: No more conflicts exist.

Figure 2.3: Optimal spatial units for the problem "Wind potential in a district" (steps 3–7).

regions increases, which could lead to higher memory and CPU usage. The relationship is usually not linear and depends on the actual problem formulation. Cao et al. (2019) showed the impact of varying the number of regions from one to 488 on computational indicators (solver ticks, number of equations, number of nonzeros, solver memory, and wall-clock time) for the REMix dispatch model of Germany. The reference model with 488 regions had more than 3×10^7 variables, 9×10^6 constraints, and almost 7×10^7 nonzeros. It used 79 GB of memory, and solving it lasted 3.6 hours.

High memory and CPU usage lead to longer computation time and eventually to the intractability of the problem. Hence, it is important that the modeler identifies the maximum number of regions n_{max} that can be used, either by inspection of the problem or through trial and error. If n_{max} exceeds the number of spatial units of the optimal resolution based on the research question, then no change is required. But if that number is lower, then I recommend reducing the resolution by merging regions while taking into account the layer priorities as defined in the algorithm of the previous section.

It is important to note, though, that the computational complexity does not only depend on the number of model regions. The technical and temporal resolution also affect it. Therefore, the maximum number of regions n_{max} actually varies depending on the number of modeled technologies and time steps. In order to be able to model more regions, the modeler can reduce the number of technologies, simplify their equations, or run the model for a few representative time steps. It is also possible to shift the complexity from one aspect to another: As already mentioned in the algorithm, sometimes it is possible to model the function of the spatial units in the layer differently, such as an attribute of a technology for example, instead of a model region on its own. The right approach depends on the research question and its objectives.

Priesmann et al. (2019) and Cao et al. (2019) reviewed other complexity reduction techniques beyond the spatial, temporal, and technological aspects, such as model decomposition and solver tuning. Whereas the spatial[1], temporal and technological aggregations are approximations, the problem reformulation and the different types of decomposition mostly lead to accurate equivalent solutions. Therefore, these options should be explored in order to reduce the overall computational complexity and provide more leeway for space modeling.

2.4 Additional aspects

The previous sections have explained the most relevant aspects that affect the choice of the optimal resolution for energy system modeling. Other aspects that also come into play are listed here briefly:

- Data availability: The difficulty to obtain inputs or to validate them may restrict the choice of the spatial resolution (assuming that there is no way to convert the data into another resolution.)
- Legal aspect: Some datasets may be missing proper licensing information, or are subject to licenses that restrict the ability to edit, process, and reuse the data.
- Economic aspect: Data acquisition in the desired resolution may cost money, or human resources and computer power to collect it, convert it into another resolution, and use it in the models.
- Ease of interpretation: It is easier to understand, interpret, and communicate the results for regions for which there are reference points, such as prior knowledge about the regions or accumulated expertise in modeling them.

Chapter summary

In this chapter, I claim that the optimal spatial resolution is mainly dictated by the limits of technical observability, the scope of the research question, the required accuracy, and the computational complexity. I argue that the optimal spatial resolution has the following properties:

1. The optimal spatial resolution is a physically observable unit of space. Energy system modelers do not have to observe the states of the energy system for each model region by means of direct measurements, but such an observation should be technically possible.
2. The optimal spatial resolution respects all the aspects that are relevant for the research question. Spatial units can have different properties, but it is up to the modeler to judge their relevance and prioritize them in case of incompatibility and conflicts.
3. There is a trade-off between higher resolution and accuracy. The optimal spatial resolution relies on validated input data, or on data that has been transformed with validated methods.

[1] Priesmann et al. (2019) did not mention the spatial aggregation explicitly, but the claim can be extrapolated to include it, since it is analogous to temporal and technical aggregations.

4. Using the optimal spatial resolution, it is possible to run the models and obtain results in a reasonable amount of time. The maximum number of model regions depends on the technical and temporal complexities of the model as well.
5. Ideally, the input data is easily obtained for the desired spatial resolution at low cost and with a permissive license.

After establishing the wishlist for the optimal spatial resolution, it is time to reflect about how to achieve it. Hence, the next chapter provides an overview of modeling techniques that enable the transformation of the data and the use of the optimal resolution.

Chapter 3

Modeling techniques

In this chapter, I describe how to flexibly move from one resolution to another by presenting the different methods I used in the enclosed papers.

3.1 Proposed approach

Generally, it is possible to differentiate between the spatial resolution(s) of the inputs, the spatial resolution(s) of the energy system model(s), and the spatial resolution(s) of the outputs. At all three stages, multiple resolutions are possible. For the sake of simplicity, I will assume in the following examples that there is only one input with one particular spatial resolution. I will explain later how to deal with multiple inputs, which is actually the most common case.

If the research question can be answered using one energy system model, i.e. if there is only one objective (optimization, simulation, etc.), and the whole problem can be formulated and solved by running one command, there is only one spatial resolution at the second stage. In the trivial case, the resolutions at the three stages are identical, as shown in Figure 3.1.

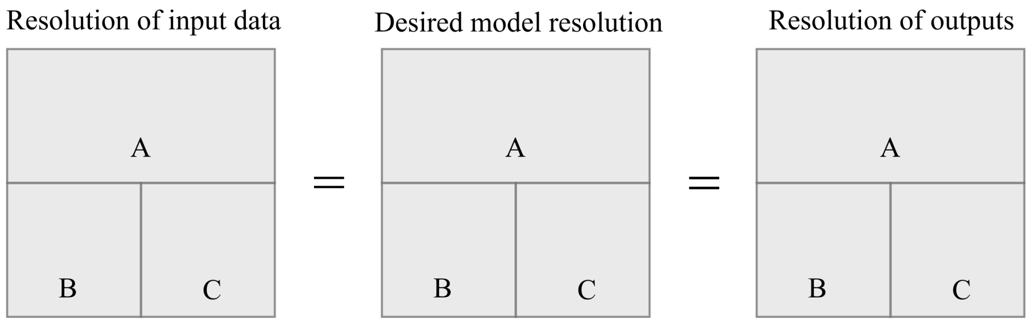


Figure 3.1: Trivial case: the spatial resolutions of the input, the model, and the outputs are identical. The rectangles denoted with a letter are the spatial units for which the input data is available (left), or which are optimal for use in energy system models as defined in the previous chapter (center), or for which the outputs are provided (right). The gray area corresponds to the spatial scope of the problem.

However, in the general case, the spatial resolutions at the three stages may differ. The spatial units can have different shapes and/or numbers, as shown in Figure 3.2.

In order to achieve a high flexibility in switching from one resolution into the other, I propose to add an intermediate step with very high resolution data between the first and second stage. Several techniques can be used to increase the resolution of the input data and disaggregate it into

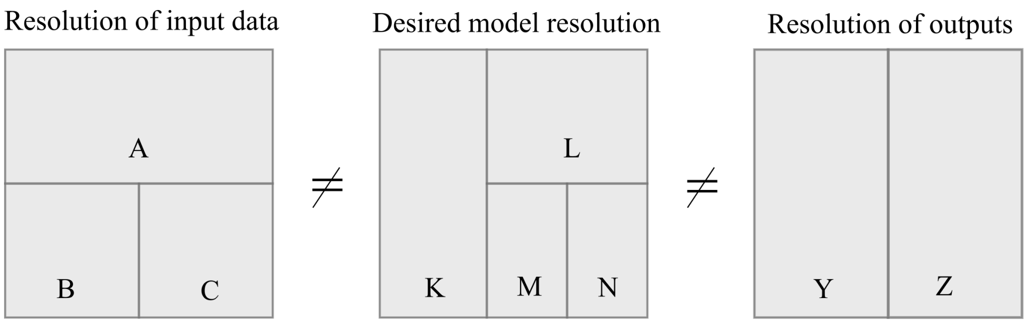


Figure 3.2: General case: the spatial resolutions of the input, the model, and the outputs may be different in shape and number.

smaller parts, which are later grouped and aggregated again to decrease the resolution according to the desired model resolution. The same applies between the second and third stage, i.e. for converting the model results into the resolution of the outputs. The methods to use for increasing and decreasing the spatial resolution depend on the data types and geometries (multipolygons, multilines, points, rasters) to be converted. Figure 3.3 shows the intermediate step using a raster of high resolution data as an example.

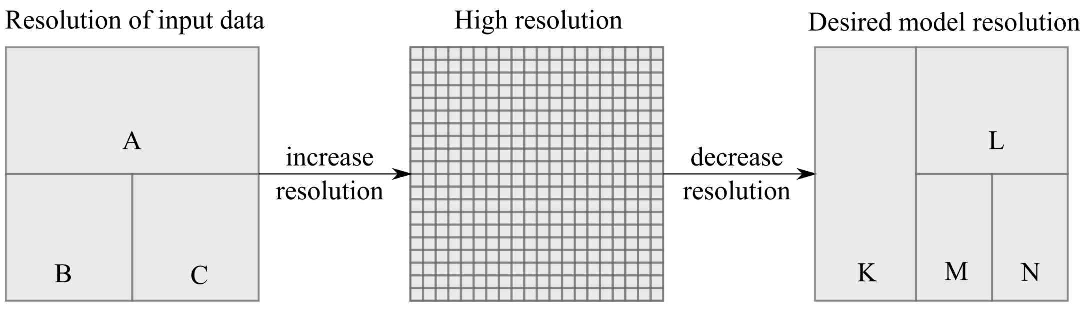


Figure 3.3: Proposed approach: increase the resolution to obtain high resolution data, then decrease it again to reach the desired resolution.

The strength of this approach resides in its versatility: multiple inputs with different resolutions can be used, as long as we can find a suitable method to disaggregate the data and aggregate it again according to the desired resolution. A selection of methods for increasing and decreasing the spatial resolutions, which are used in the publications that are enclosed in this thesis, are presented briefly in the following sections of this chapter.

Often times, the research question requires the use of more than one energy system model. This is the case if there are multiple objectives (e.g. long-term optimization for expansion planning and short-term simulation of operation), or if the research question requires models with different temporal and spatial scope, or when the problem is too complex to be solved in one run, so that it is decomposed in one master problem and multiple subproblems. In all these cases, two or more models can be coupled. In order to reach the desired resolution of each model, the previously introduced approach can be applied. Then, the models are run seperately, sequentially or in parallel, depending on the nature of the model coupling. In the simplest case, the outputs of one model are the inputs of the other. Feedback loops may exist, and in the case of decomposition problems, the outcome of the models is only known at the end of all model runs, because the subproblems are not solved explicitely. The model coupling affects the modeling of space at that stage.

If the models share exactly the same geographic scope but have different spatial resolutions, then it is important that they share information that can be interpreted by both models, i.e. that is provided in a resolution compatible with each one of theirs. Here again, an intermediate step with a high resolution as described previously is one way of achieving that target. If the models do not have the same scope, but the spatial units of one model are a subset of the other, then the information exchange is straightforward for the common model regions and inexistent for the others. In the general case, the modeler may have to deal with different geographic scopes and different resolutions within the shared area, as displayed in Figure 3.4.

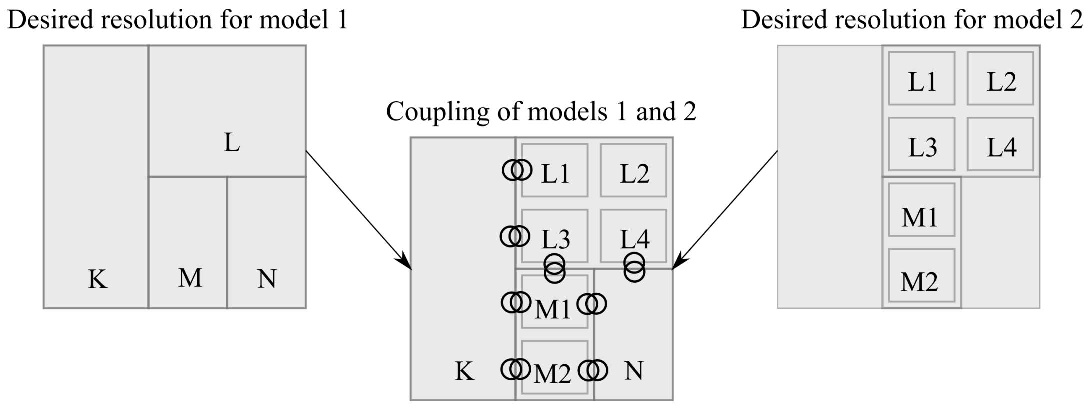

Desired resolution for model 1
L
K
M
N
Coupling of models 1 and 2
L1
L2
L3
L4
M1
M2
K
N
Desired resolution for model 2
L1
L2
L3
L4
M1
M2

Figure 3.4: Schema of a model coupling operation, where model 1 has a larger geographic scope but a coarser spatial resolution than model 2. The coupling symbols (interlinked circles) highlight the locations where the models need to exchange information in their respective resolutions.

To conclude, starting from the case of a single input, one model, and a single output having different spatial resolutions, I showed that it is possible to reconcile the spatial resolutions by going through an intermediate step of high resolution data. This approach allows several inputs to have different resolutions as well. The same applies by analogy to the outputs. In case of model coupling, the approach is extended to deal with the cases of different geographic scopes and/or different model regions within the same scope. Examples of methods for increasing and decreasing the spatial resolutions and for model coupling are provided in the next sections.

3.2 Increasing the spatial resolution

The purpose of this step is to transition into a state with higher information content for the same geographic scope, i.e. a higher information density. Here is a non-exhaustive list of techniques that I have used in the enclosed papers to disaggregate data and increase its resolution, together with illustrative examples in Figures 3.5 and 3.6:

Value repetition: This is the most trivial method. No data is added; instead, the information is saved in a less efficient way by repeating it in regular steps. For example, a low-resolution two-dimensional raster of 2×3 cells can be converted into a 20×30 high-resolution raster that is hundred times larger simply by repeating each cell 10 times. This is sometimes necessary when the raster inputs cover the same scope, but have different resolutions, so the low-resolution rasters are first converted to the higher resolution, before any matrix operations (element-wise sum, multiplication or comparison) can be performed on them. In the publication in Chapter 4, all the rasters (in particular the MERRA reanalysis data) are first converted into rasters with 15 arcsec resolution.

Interpolation and kriging: Interpolation is a method by which the value of a point for which no observation is available is estimated based on the values of neighboring observations. It can be applied if there are discrete observations and boundary conditions, if the values that are interpolated are in a continuous domain, and if neighbors are more likely to have similar values and affect each other than remote locations, which is roughly Tobler's First Law of Geography[1]. Kriging is a special interpolation for which a stochastic process and prior covariances are used to model the interpolated values. It delivered better results than other interpolating methods when I applied it to data from weather stations in the publication in Chapter 4.

Combination of data sources with different resolutions: This is an extension of the first method, *value repetition*. After making sure that the datasets have the same size, a new dataset is created as a function of two or more datasets, using element-wise operations. For example, the wind speed in a particular location depends on the wind speed at 50 m and on the elevation at that location. Thus, by combining the low-resolution wind speed data of MERRA with the high-resolution elevation data, I obtain high-resolution wind speed data. Other applications of this method abound in the publication in Chapter 4 and in the Python tools documented in Annex A and in Annex C.

Example: Techniques for increasing the spatial resolution (Data)

The hourly wind speed data from MERRA-2 (Gelaro et al., 2017) has a spatial resolution of 0.5° × 0.625°. Hence, six grid cells are needed for the geographic scope of Samoa, as shown in the top figure.
Land use data in 15 arcsec resolution (Broxton et al., 2014) can be used because the wind gradient and the roughness of the terrain vary depending on the land use type. If used, the Hellmann coefficients (indicators of terrain roughness) are assumed to vary between 0.10 for water and 0.40 for urban areas. Otherwise, a default value of 0.25 is used.
The terrain elevation in 15 arcsec resolution (de Ferranti and Hormann, 2015) may be used to correct the wind speed and match the wind distribution of the Global Wind Atlas (Technical University of Denmark, 2019).

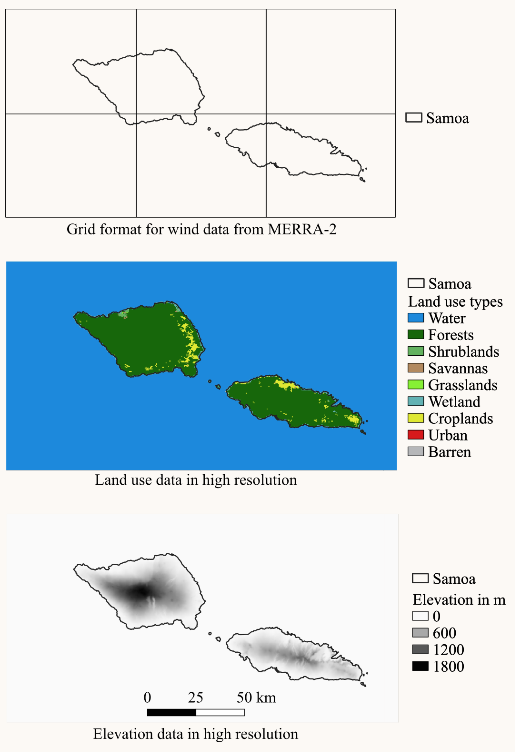


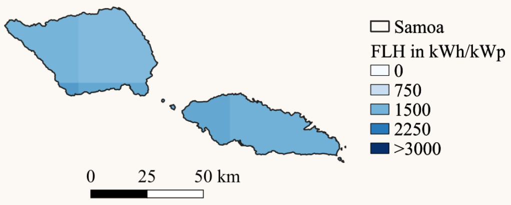
Figure 3.5: Data sets with different spatial resolutions for wind potential estimation for Samoa.

[1] "Everything is related to everything else, but near things are more related than distant things." (Tobler, 1970)

Example: Techniques for increasing the spatial resolution (Comparison)

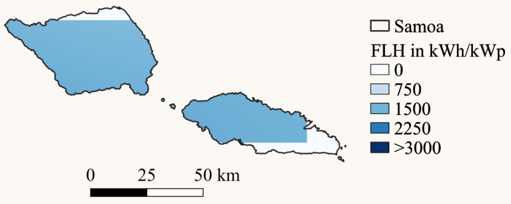
The following calculations are done using wind speed data for 2015 which is fed into the tool pyGRETA (see Appendix A). In all three cases, wind turbines with a hub height of 80 m are used.

If *value repetition* is chosen, then the hourly wind speed data from the six grid cells is repeated to reach the resolution of 15 arcsec. The only advantage is that it enables the masking with the shapefile of the land area.



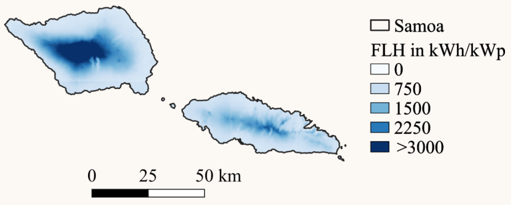
(a) Onshore wind potential of Samoa using value repetition of wind speed data.

Linear interpolation smoothens the data between the six grid cells. However, it does not reflect how wind speed is actually distributed, because it ignores the orography and elevation of the terrain. Besides, the behavior on the boundaries should either be set by the user, or the geographic scope should be enlarged to avoid missing values as in Figure 3.5b.



(b) Onshore wind potential of Samoa using linear interpolation of wind speed data.

The *combination of data sources* requires more computational effort than a simple value repetition. The advantage lies in the differenciation between the land use types and the terrain elevations. In this case, since the dominant land use types, forests and croplands, are assumed to have Hellmann coefficients of 0.25 and 0.20 respectively, the impact of the land use map is hardly visible. The mountainous areas, which tend to have a higher wind speed at hub height than low-lying coastal areas, have a higher wind potential, which is visible in the dark blue colors that replicate the shapes of the elevation map.



(c) Onshore wind potential of Samoa using a combination of wind speed data, land use data, and elevation data.

Figure 3.6: Comparison of techniques for increasing the spatial resolution on the onshore wind potential estimation for Samoa.

3.3 Decreasing the spatial resolution

As opposed to the previous section, the purpose here is to reduce the amount of information and extract a small portion of it which is representative for the whole model region. Before listing the aggregation methods that are used in the publications, I would like to emphasize the distinction between *user-defined aggregation*, where the modeler sets the shape and number of the regions to be used in the model, and *data-driven clustering*, where the shape and/or number of regions is not known in advance and is determined based on the data itself. Chapter 5 presents a method of clustering geographic information on a high resolution, which relies on the Python tool documented in Annex B. In either case, there is a reduction of the amount of information. This can be done through a statistical summary or through sampling. Figure 3.7 provides an illustrative example of the techniques.

Statistical summary: Here, one value is selected to summarize a set of input values using a suitable statistical function such as the mode, mean, median, standard deviation, maximum, or minimum. The choice of the function depends on the data type and the objective. If the high resolution data is a land use map with various land use categories, then the mode of the data points within the boundaries of the model region determines the most common land use type. For the total electricity demand of that region, the sum of the values of each data point is calculated, whereas for the wind potential expressed in full-load hours, the average is suitable. However, for the peak electricity demand or for the potential of the best location, an aggregation using the maximum is more appropriate.

Sampling: Instead of summarizing all the values using statistical functions, the modeler can select a sample of values at different intervals or quantiles. The downsampling at regular intervals can be used on high resolution raster maps to reduce their size by only keeping data points on a coarse grid. This method is simple, yet it may cause a loss of critical information in the intervals that have been skipped, especially if applied on unsorted data. Instead, I recommend sorting the data and extracting information at particular quantiles, whenever possible. This method gives more control to the modeler on which information to keep or to dismiss. I use it to generate time series for renewable energy sources in Chapter 4 and in Annex A, for example.

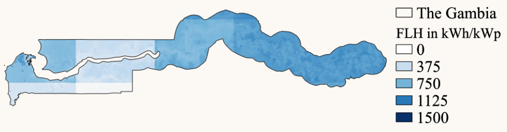
3.4 Combining spatial resolutions

As explained in Section 3.1, there are many cases where two or more models (or instances of the same model) have to be coupled together to answer the research question. The models may have different scopes and/or different spatial resolutions within the shared scope. There are no clear rules or standards on how to couple models, and if there were any, they would have to cover a wide range of cases for all possible applications and combinations of models. Instead, I propose to consider the following aspects, based on my experience in coupling an expansion planning optimization model of Europe with an Input-Output Analysis of the whole world, which is the topic of the publication in Chapter 6.

Shared scope: It is important to clearly define the shared scope of the models. Regions that are used in one model but not in the other are not necessarily superfluous; in fact, the purpose of the coupling may be actually to extend the geographic scope of one model using the other. In the case of the aforementioned publication, the optimization model used the EU28 countries except Malta and Cyprus, plus Norway and Switzerland as model regions for electricity system expansion planning. The Input-Output Analysis is based on a database with global coverage, where 27 EU

Example: Techniques for decreasing the spatial resolution

The map in Figure 3.6a is obtained by replicating the approach of *combination of data sources* on the case of The Gambia. Despite the use of elevation and land use data, the original resolution of the wind data is still visible (unlike Figure 3.5c for Samoa). This is because the roughness of the terrain is almost homogeneous in The Gambia, and the elevation is rather low (<85 m).



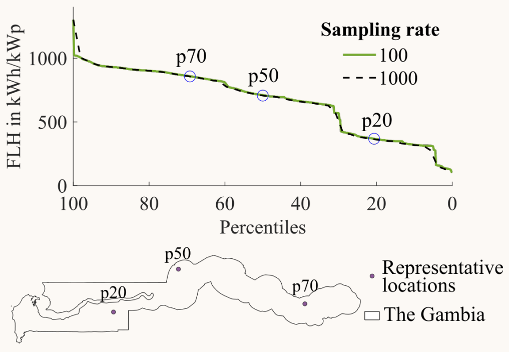
(a) Onshore wind potential of The Gambia using a combination of wind speed data, land use data, and elevation data.

Statistical summary: Table 3.1 provides an example of a statistical summary of the data from Figure 3.6a.

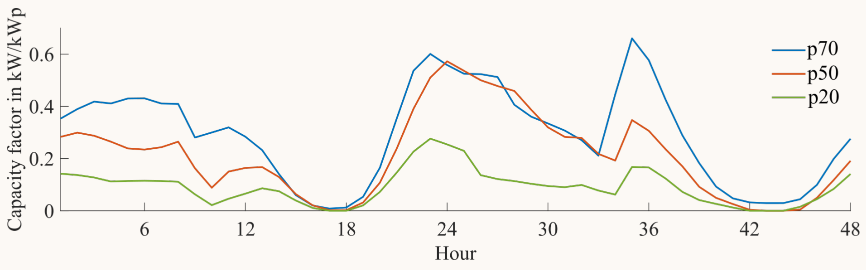
Table 3.1: Statistical summary for the wind potential in The Gambia.

Number of pixels	Area (km^2)	Mean FLH (kWh/kWp)	Max FLH (kWh/kWp)	Energy potential (TWh)
51 304	10 671	667	1302	57

Sampling: After sorting the FLH values, sampling is performed to identify representative locations. A high sampling rate (r=1000) in this case leads to an overestimation of the performance of good wind locations. Three points from the major plateaus in the curve on Figure 3.6b are chosen to represent the different wind site qualities in the country. The corresponding pixels in the map are identified, and time series for their locations are generated.



(b) Sorted wind full-load hours for The Gambia.



(c) Representative wind time series for The Gambia, for the locations of the percentiles q70, q50 and q20 and the first 48 hours of 2015.

Figure 3.7: Overview of techniques for decreasing the spatial resolution of the onshore wind potential estimation for The Gambia.

countries (EU28 minus Croatia) are modeled individually. The others are grouped in mega-regions, such as “rest of world”, so they could not be used in the analysis. Hence, the shared scope was 25 European countries (EU28 minus Malta, Cyprus and Croatia). Even though Norway, Switzerland and Croatia were not part of the analysis, they were used anyway in the optimization model due to their strong ties with neighboring countries in the European electricity grid. Their outputs were not shared with the Input-Output model, though.

Syntax and semantic errors: Model coupling is prone to errors if the models use spatial units with different names. Dictionaries might be necessary to convert names from one naming convention to the other. For example, the optimization model of Europe used two-character code names for countries, whereas the Input-Output table used three-character codes. This would cause a syntax error in extracting the data, unless a dictionary is used. However, the same names may also refer to different things, causing semantic errors. This was the case for France, Spain and Portugal, for instance. In the optimization model, the regions that are referred to are limited to the European continent, excluding French Guiana, Spanish enclaves in Africa, the Balearic islands, Madeira and the Azores. The Input-Output tables, which are based on national accounts, include information from these areas. The modeler has to be aware of the semantic differences, in order to eventually remedy them.

Data sharing interface: For the model coupling to succeed, modelers need to identify the possible synergies between the models. In case of different types of models, these are endogeneous variables in one model A that can be used as exogeneous parameters in another model B. For example, the electricity mix delivered by an optimization model (endogeneous result) can be used in an Input-Output table as a parameter. In the case where models A and B have different geographic scopes, the synergy could lie in the additional regions that are not shared by A and B. To use the same example of the publication in Chapter 6, the Input-Output table had regions outside of Europe, which was the scope of the optimization model. In order to facilitate the communication between the models, a data sharing interface can be used which includes the outputs of model A in a usable format for model B, taking into account the differences in syntax and units. The interface can include a feature to allow the communication in both directions, which is necessary if the model coupling involves a feedback loop.

Chapter summary

In this chapter, I propose a systematic approach for transforming data into the optimal spatial resolution. If the research question can be answered with one model, then it is possible to first disaggregate the data, obtain high resolution data in an intermediate step, and finally aggregate it again according to the desired resolution. If more than one model is needed, model coupling can be used to expand the modeling capabilities and/or the geographic scope.

I also provide an overview of modeling techniques that enable the transformation of the data. Value repetition, interpolation, and the combination of data sources with different resolutions can be used to increase the spatial resolution. Data aggregation, whether it is user-defined or data-driven, could be achieved through statistical summaries or sampling. Despite the lack of standards for model coupling, I provide some guidance on how to define the shared scope, how to avoid syntax and semantic errors, and how to manage data sharing between the models.

Chapter 4

Increasing the spatial resolution: Application for potential estimation

The first publication included in this dissertation is on the estimation of the PV potential in South East Asia. It illustrates how it is possible to combine information from different data sources, some of it in low resolution, in order to obtain a dataset in high spatial resolution, as described in Section 3.2.

The publication is based on the work I did in 2014/2015 for my Master's thesis "Determination of the Power Generation Potential of Solar PV and Wind in South East Asia". As a first author, I was responsible for the idea, structure and content of the paper. I also wrote the Matlab code which was used to obtain the results, created the figures and tables, and took care of the communication with the journal and the reviewers. My co-author, Jürgen Stich, contributed with suggestions for the structure, and proofread the manuscript and the answers to the reviewers.

Since the article was published, I have continued developing the code with the help of my students. It is now available as an open-source tool written in Python and published in GitHub. Some of the new features include the ease of use for any region within the globe, multiple options to correct the data, the possibility to match full-load hour statistics from the International Renewable Energy Agency (IRENA), and improved computation speed. The latest documentation of the tool is included in Annex A.

Title: Estimation of the PV potential in ASEAN with a high spatial and temporal resolution

Authors: Kais Siala, Jürgen Stich

Journal: Renewable Energy (Elsevier)

Publication date: January 2016

Permission to use: Elsevier publishing guidelines allow authors to "include their articles in full or in part in a thesis or dissertation for non-commercial purposes".

Abstract

Further exploitation of renewable energy sources for power generation could mitigate the current rapid increase of greenhouse gas emissions in South East Asia. In this context, solar PV is a promising option as PV system costs have been declining continuously over the past. In order to define strategies towards a low-carbon future power supply, detailed information on the potential power output of solar PV is essential. Therefore, this paper analyses the resource and technical potential of solar PV in South East Asia in high temporal and spatial resolution. An empirical, climate-based Ångström-Prescott model is proposed in order to adjust MERRA solar radiation data. The possible power output of PV is derived considering topographic and land-use constraints as well as technological characteristics of typical PV systems. Java, central Myanmar and eastern Thailand were identified to be the best locations for PV use, with capacity factors exceeding 15 %. Due to the large land area which is suitable for PV installations, South East Asia offers an abundant theoretical potential of solar PV, amounting to 430 TW h with conservative assumptions.

Keywords: PV potential; solar radiation; MERRA; GIS

4.1 Introduction

Satellite and retrospective-analysis (re-analysis) data have been widely used to estimate solar radiation potential (Kennedy et al., 2011; Olseth and Skartveit, 2001; Perez et al., 2002), particularly for PV applications (Richardson and Andrews, 2014; Stein et al., 2010). One of their biggest advantages are their easy accessibility and their high temporal resolution. For instance, the Modern Era Retrospective-Analysis for Research data (MERRA) provided by NASA is publicly available in an hourly resolution and covers the period stretching from 1979 until now (Rienecker et al., 2011). Many studies have assessed the suitability of MERRA data for energy potential estimation with mixed opinions. Whereas some consider the results acceptable within a certain error margin (Richardson and Andrews, 2014), others pointed out the overprediction of clear-sky conditions and the high inaccuracy even within climatically homogeneous areas (Boilley and Wald, 2015).

In order to solve these problems, it is necessary to compare MERRA data with solar radiation measurements, examine the nature and location of the discrepancies, and correct them. In South East Asia, solar radiation measurements are only available at a limited number of stations, which do not cover the region entirely. On the other hand, measurements of sunshine hours are available at a greater number of stations. Using the Ångström-Prescott (A-P) empirical correlation between irradiation and sunshine hours (Ångström, 1924; Prescott, 1940), it is possible to obtain an accurate estimation of the monthly global horizontal irradiation in ASEAN with a high spatial resolution. There are many A-P empirical correlations with different spatial scales in the literature (Bakirci, 2009). Generally, A-P coefficients for one particular location can be used for another site, provided that the coefficients are adjusted to account for the climatic differences (Sen, 1998). In this paper, the authors suggest using a climate-based empirical A-P correlation, i.e. monthly coefficients will be derived for each climate zone in ASEAN. In fact, a strong correlation exists between the A-P empirical model (based on solar radiation) and the Köppen-Geiger climate classification (based on temperature and precipitation) (Polo et al., 2015). Developing an A-P model based on climate zones allows to aggregate different locations with similar climatic conditions and therefore the analysis of areas of a large extent while keeping the A-P model itself simple. The derived monthly A-P coefficients for each climatic zone can be used afterwards to adjust the hourly data provided by MERRA for the entire ASEAN region. To our knowledge, no A-P model exists in the literature which is based on climatic zones. Furthermore, we have not found studies that estimate the solar PV potential of South East Asia with similar approaches. In this region, there are less weather station compared to other parts of the world (e.g., North America or Europe), and only a few of them are measuring solar radiation. Therefore, estimating the solar PV potential using adjusted retrospective-analysis data is particularly important for South East Asia.

Based on the global horizontal irradiance, it is possible to estimate the irradiance on a tilted surface, provided that solar radiation components (direct and diffuse) are known. However, direct and diffuse radiation are rarely measured. Thus, numerous decomposition models based on other parameters were developed to estimate the direct and diffuse fractions. Wong and Chow (2001) made a useful review of the different decomposition models available in the literature.

In addition to accurate solar radiation data, estimating the PV potential requires the setting of technical constraints to model the behaviour of the PV system under realistic conditions. Numerous PV potential studies have analysed the influence of technical constraints, particularly the impact of heating losses (Skoplaki and Palyvos, 2009), siting and shading (Ruiz-Arias et al., 2010), and tilting and orientation (Page, 2003). In this paper, all aforementioned aspects are considered and their impact on the final result is evaluated.

Thus, the objectives of this study are (i) to calibrate the A-P model coefficients for each climatic zone in ASEAN, (ii) estimate the monthly solar radiation in ASEAN with a high spatial resolution, (iii) use the monthly values to correct hourly MERRA data, and (iv) estimate the PV technical potential in ASEAN under realistic constraints.

4.2 Data

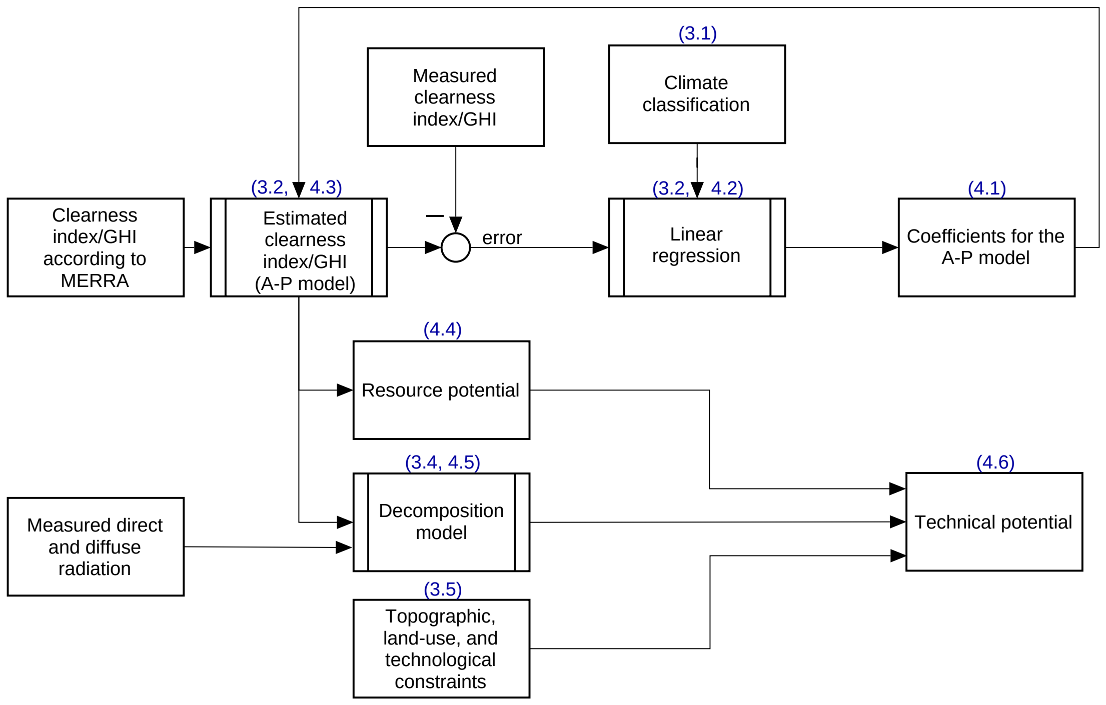
Here is a short description of the data sources used in this paper:

- MERRA data: The re-analysis data provided by NASA include hourly average top of the atmosphere irradiance (TOA) and global horizontal irradiance (GHI) since 1979, on a spatial resolution of $\frac{1}{2}^\circ$ latitude and $\frac{2}{3}^\circ$ longitude (Rienecker et al., 2011).
- Solar radiation measurements: Measurements of the GHI are available for 20 stations across ASEAN and its neighbouring countries on a daily and/or monthly basis at different periods of time (Institute for Climate and Atmospheric Sciences, 2015; World Radiation Data Centre, 2015). Additionally, hourly measurements for direct, diffuse, and global radiation are provided for Bukit Kototabang, Indonesia, since 1996 (World Radiation Data Centre, 2015).
- Sunshine hours measurements: Long-term monthly mean values of sunshine hours for 249 stations in ASEAN and its neighbouring countries are part of the CLIMWAT 2.0 database (Water Resources, Development and Management Service, 2015).
- Climate zones classification: For this paper, a Köppen-Geiger climate classification map with a $\frac{1}{10}^\circ$ resolution was obtained from the literature (Peel et al., 2007).
- Land-use data: Broxton et al. (2014) provided a dataset for global land cover with 17 classes of land-use at 15 arcsec ($\frac{1}{240}^\circ$) resolution. Additionally, data regarding protected areas was gathered from IUCN[1] and UNEP[2] (IUCN and UNEP, 2014).
- Topographic data: The SRTM[3] data, originally produced by NASA, is resampled at a resolution of 15 arcsec to reduce computation time (Jarvis et al., 2008).
- PV module characteristics: In this paper, the technical characteristics of Yingli Solar YL260C-30b PV modules are used (Yingli Solar, 2014).

[1] International Union for Conservation of Nature
[2] United Nations Environment Programme
[3] 2000 Shuttle Radar Topography Mission

4.3 Methods

The framework of the study is displayed in Fig. 4.1. The numbers in parentheses refer to the corresponding sections in the paper.



Measured clearness index/GHI
(3.1)
Climate classification
(3.2, 4.3)
(3.2, 4.2)
(4.1)
Clearness index/GHI according to MERRA
Estimated clearness index/GHI (A-P model)
−
error
Linear regression
Coefficients for the A-P model
(4.4)
Resource potential
(3.4, 4.5)
(4.6)
Measured direct and diffuse radiation
Decomposition model
Technical potential
(3.5)
Topographic, land-use, and technological constraints

Figure 4.1: Framework of the methodology.

4.3.1 Climate classification of ASEAN

The Köppen-Geiger climate classification is a widely used climate classification system, which describes climatic conditions defined by precipitation and temperature and their seasonal fluctuations. Based on the updated GIS world map of the Köppen-Geiger climate classification (Peel et al., 2007), it appears that ASEAN can be divided into four major climate zones[4]:

1. tropical rainforest (Af), characterised by a year-round hot and wet climate (e.g. Sumatra);
2. temperate dry winter (Cwa), characterised by a temperate climate with a dry winter and a wet hot summer (e.g. northern Vietnam);
3. tropical monsoon (Am), with year-round hot temperatures and a moderately-dry season (e.g. southern Thailand);
4. tropical savannah (Aw), with year-round hot temperatures and a pronounced dry season (e.g. central Myanmar).

Some areas in northern Myanmar and northern Philippines experience a temperate climate without dry seasons, but these areas are relatively small and do not include any station with solar radiation measurements. Thus, their climate will be assimilated to the one of the closest major climate zone.

[4] exact metrics for the classification are available in (Peel et al., 2007, p. 1636)

Additionally, a buffer zone of 60 km on the border between two different climate zones was added. The climate in these buffer zones is assumed to be equally influenced by the neighbouring zones (see Eq. (4.4)). The resulting classification is displayed in Fig. 4.2, where each major climate zone and each buffer zone has its own colour. For example, the buffer zone between the tropical rainforest (Af) and the tropical monsoon (Am) is noted Af/Am and is coloured in cyan.

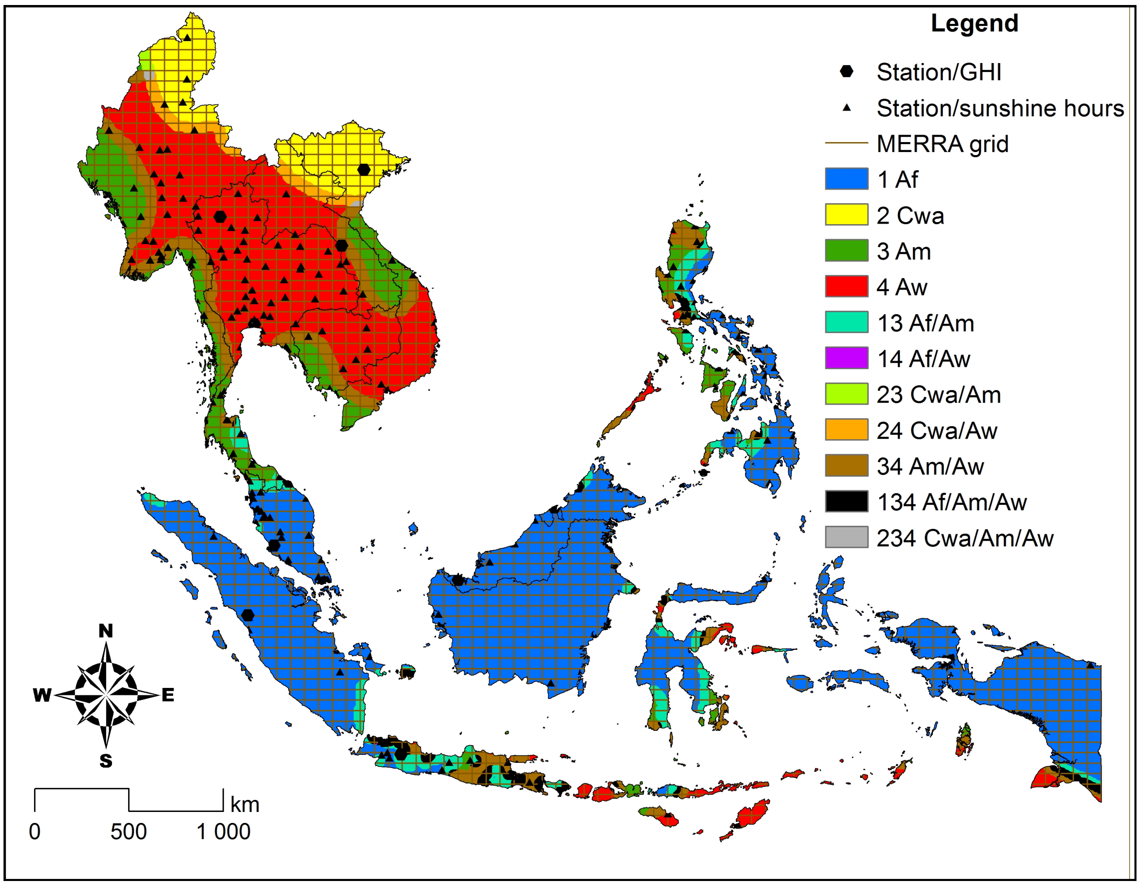


Figure 4.2: Climate zones in ASEAN.

4.3.2 Climate-based A-P model for ASEAN

The general equation for the climate-based A-P model is defined as follows:

$$k_{t,m} = a_{z,m} + b_{z,m} \cdot S_{f,m} \qquad [-] \qquad (4.1)$$

where $k_{t,m}$ stands for the monthly average daily clearness index, $a_{z,m}$ and $b_{z,m}$ for the monthly, zone-dependent model coefficients, and $S_{f,m}$ for the monthly sunshine fraction. The latter can be estimated through the following equation:

$$S_{f,m} = \frac{(S_d)_m}{(S_{0,d})_m} \qquad [-] \qquad (4.2)$$

with the monthly average daily sunshine hours $(S_d)_m$ (obtained from Water Resources, Development and Management Service (2015)) and the monthly average daytime length $(S_{0,d})_m$ (computed using further equations from the literature (Page, 2003)).

Since seasons do not occur at the same time at different locations within the same climatic zone, the monthly averages for clearness indices and sunshine fractions have to be shifted in time according to the seasons. In this paper, the sequence of monthly averages starts with the month with the lowest clearness index or with the lowest sunshine fraction, in the absence of the former.

The coefficients $a_{z,m}$ and $b_{z,m}$ are calibrated by using the least squares regression analysis at the 20 stations where both global irradiation and sunshine hours measurements are available. Therefore, the monthly clearness indices at these stations need to be computed first using the following equation:

$$k_{t,m} = \frac{(G_{t,d})_m}{(G_{0,d})_m} \qquad [-] \quad (4.3)$$

where $(G_{t,d})_m$ is the monthly average daily global horizontal irradiation at the surface (obtained from the WRDC database (World Radiation Data Centre, 2015)) and $(G_{0,d})_m$ the monthly average daily GHI on top of the atmosphere (computed using further equations from the literature (Page, 2003)).

As mentioned above, stations located in the buffer zones are affected by their neighbouring climates. They are also used to calibrate the coefficients, and their influence depends on the climates they are affected by. The weighing factor, $w_{station \in z}$, is inversely proportional to the number of major climates affecting each location. Thus, Eq. (4.1) becomes:

$$\hat{k}_{t,m} = \sum_{z=1}^{4} (a_{z,m} + b_{z,m} \cdot S_{f,m}) \cdot w_{station \in z} \qquad [-] \quad (4.4)$$

For instance, the monthly clearness index for a station located in the buffer zone Af/Am (between the zones 1 and 3) is calculated as follows:

$$\begin{aligned} \hat{k}_{t,m} =& (a_{1,m} + b_{1,m} \cdot S_{f,m}) \cdot \frac{1}{2} \\ &+ (a_{3,m} + b_{3,m} \cdot S_{f,m}) \cdot \frac{1}{2} \end{aligned} \qquad [-] \quad (4.5)$$

The goal of the least square regression is to minimise the root mean square error (RMSE) between the measured and the estimated clearness indices for each station, which is defined as follows:

$$RMSE_{station} = \sqrt{\frac{\sum_{m=1}^{12} (k_{t,m} - \hat{k}_{t,m})^2}{12}} \qquad [-] \quad (4.6)$$

After calibrating $a_{z,m}$ and $b_{z,m}$ for each zone and month, it is now possible to estimate the monthly average value of the clearness index in each location in ASEAN. Therefore, the monthly average daily sunshine fraction data from 249 stations in ASEAN and bordering regions is first interpolated using Empirical Bayesian Kriging with a resolution of 15 arcsec (approximately 500 m at the Equator), then used in Eq. (4.1) to estimate the monthly clearness indices with the same resolution.

4.3.3 Correction of MERRA data

For the following section, the MERRA data for the year 2007 is used, since its yearly mean GHI is the closest to the 10-year average between 2004 and 2013 in ASEAN. The hourly clearness indices calculated using MERRA data are adjusted by monthly correction factors for each climatic zone. The correction factors are calculated by dividing the monthly average daily clearness indices $k_{t,m}$, which were previously derived using the A-P model, by the monthly average daily clearness indices based on MERRA data. The correction is shown in Eq. (4.7):

$$k_{t,h} = \frac{\langle G_{t,h} \rangle_{MERRA}}{\langle G_{0,h} \rangle_{MERRA}} \cdot \frac{k_{t,m}}{\frac{\langle (G_{t,d})_m \rangle_{MERRA}}{\langle (G_{0,d})_m \rangle_{MERRA}}} \qquad [-] \quad (4.7)$$

The purpose of this adjustment is not to correct the hourly values of the data for the year 2007, but to obtain an hourly time series in accordance with the monthly, long-term estimates of the clearness indices for every location in ASEAN.

4.3.4 Estimation of the hourly global-to-diffuse ratio

For an accurate estimation of the PV potential on a tilted surface, it is necessary to decompose the GHI into its direct and diffuse components, and then use them to estimate the incident radiation on the PV panel. In ASEAN, only the station of Bukit Kototabang, Indonesia is part of the Global Atmosphere Watch (GAW), which provides publicly available hourly measurement data of global, direct, and diffuse radiation. In the lack of other sources, this data is used to derive a function to estimate the diffuse fraction for a given clearness index and elevation angle, based on the model proposed by Reindl et al. (1990) and shown in Eq. (4.8):

$$k_{d,h} = p_i + q_i \cdot k_{t,h} + r_i \cdot sin(\alpha), \quad l_{inf,i} \leq k_{t,h} < l_{sup,i} \qquad (4.8)$$

where p_i, q_i, and r_i are empirical coefficients to be determined by least square regression, and $l_{inf,i}$ and $l_{sup,i}$ are the limits of the range R_i. Two changes were made to this general equation:

- In contrast to the piecewise function proposed by Reindl et al., the ranges depend on the sine of the elevation angle instead of the clearness index. This change was conducted because the clearness indices in the data of Bukit Kototabang are rather low (more than 93 % are smaller than 0.3). Splitting the data according to the sine of the elevation angle leads to three sets of similar cardinality.
- In order to be able to use the formula for locations other than Bukit Kototabang, it is necessary to normalise it to the maximal clearness index that occurs within a year. For example, a clearness index of 0.5 might mean a relatively clear day in Bukit Kototabang, because the maximum within a year lies at 0.65. In another location with less cloudy days, the maximum could be 0.8, so a clearness index of 0.5 would occur in a relatively cloudy day. The normalisation alleviates this issue.

The general equation for the diffuse fraction is thus shown in Eq. (4.9).

$$k_{d,h} = p_i + q_i \cdot \frac{k_{t,h}}{max(k_{t,h})_y} + r_i \cdot sin(\alpha), \quad l_{inf,i} \leq sin(\alpha) < l_{sup,i} \qquad (4.9)$$

The maximal clearness index $max(k_{t,h})_y$ for Bukit Kototabang can be derived from the measurement data set itself. For other locations in ASEAN, it will be assumed that $max(k_{t,h})_y = \langle max(k_{t,h})_y \rangle_{MERRA}$. The elevation angle depends on the latitude and the time of the year, and can be calculated using models from the literature (Page, 2003).

4.3.5 Technical constraints and properties

In order to estimate the technical potential for a given radiation data set, the following constraints are adopted:

- Terrain constraints: Protected areas, water bodies, and wetlands are excluded, as well as terrains with a slope higher than 3 %, similarly to the approach adopted by NREL[5] (Lopez et al., 2012).
- Spacing: The PV panels are installed in parallel rows such that no inter-row shading occurs for at least six (6) hours during the shortest day of the year. Inter-row shading losses which occur early in the morning or late in the evening are neglected.
- Topographic shading: The shading by objects on the horizon (mountains, hills, etc.) affects the direct radiation. For the sake of simplicity, the azimuth angle of the sun is rounded to the next possible discrete value corresponding to due East, South, West, or North (90°, 180°, 270°, and 0°, respectively), so that the calculation of the shadow length becomes a one-dimensional problem:

  $$l_{shadow}(i,j) = \frac{\Delta h(i,j)}{tan(\alpha)} \qquad [\mathrm{m}] \quad (4.10)$$

  where α is the elevation angle and Δh the height difference between two pixels (i,j) in the sun ray direction. The shading losses can be expressed as:

  $$L_{shading} = \sum_{E,S,W,N} \sum_{j=i+1}^{i+horizon} \frac{l_{shadow}(i,j)}{\Delta l(i,j)} \qquad [\mathrm{m}] \quad (4.11)$$

  where *horizon* stands for the pixel number located at the optical horizon, assumed to be $\frac{1}{48}$ ° (five pixels away from i), and Δl the distance between the centres of the pixels i and j in the Sun beam direction. There is a trade-off between higher accuracy and computation time when setting the horizon distance.
- Tilt and orientation: The tilting angle β is set equal to the latitude ϕ but at least equal to 15°. The panels are always facing to the Equator, i.e. the surface azimuth angle ψ_S is 180° in the northern hemisphere and 0° in the southern hemisphere. Both angles have an impact on the incident radiation (Page, 2003):

  $$\begin{aligned} G_{incident,h} =& [(1-k_{d,h}) \cdot \frac{cos(\alpha)\cdot cos(\psi-\psi_S)\cdot sin(\beta) + sin(\alpha)\cdot cos(\beta)}{sin(\alpha)} \\ &+ k_{d,h} \cdot \frac{1+cos(\beta)}{2} \\ &+ \rho \cdot \frac{1-cos(\beta)}{2}] \cdot k_{t,h} \cdot G_{0,h} \end{aligned} \qquad [\mathrm{W\,m^{-2}}] \quad (4.12)$$

  where ψ is the azimuth angle of the sun and ρ the albedo of the ground (0.65 for rooftops, otherwise 0.25).
- System performance: The module efficiency is 16 %, and the DC/AC conversion efficiency is assumed to be 75 %.
- Heating losses: The rated temperature of the module is $T_{STC} = 25\,°\mathrm{C}$, with a temperature loss coefficient of $f_T = -0.42\,\%/°\mathrm{C}$. The actual operating cell temperature is estimated using the following equation (Skoplaki and Palyvos, 2009):

  $$T_{cell,h} = \langle T_{ambient,h} \rangle_{MERRA} + f_{Ross} \cdot G_{incident,h} \qquad [°\mathrm{C}] \quad (4.13)$$

  where $\langle T_{ambient,h} \rangle_{MERRA}$ is the hourly temperature according to MERRA, f_{Ross} the Ross coefficient (assumed to be 0.0260 in cities and 0.0208 elsewhere), and $G_{incident,h}$ the incident

[5]National Renewable Energy Laboratory (USA)

irradiance on the PV panel in W m^{-2}. Using $T_{cell,h}$, it is now possible to estimate the heating losses:

$$L_{heating} = \begin{cases} (T_{cell,h} - T_{STC}) \cdot f_T, & T_{cell,h} > T_{STC} \\ 0, & otherwise \end{cases} \qquad [\%] \qquad (4.14)$$

4.4 Results

4.4.1 Climate-based A-P model coefficients

The A-P model coefficients for each climate zone and month, which were obtained through a least squares regression analysis, are displayed in Tab. 4.1.

Table 4.1: A-P model coefficients for each climate zone.

Zone	$a_{z,1}$	$a_{z,2}$	$a_{z,3}$	$a_{z,4}$	$a_{z,5}$	$a_{z,6}$	$a_{z,7}$	$a_{z,8}$	$a_{z,9}$	$a_{z,10}$	$a_{z,11}$	$a_{z,12}$
Af	0.41	0.29	0.07	0.20	0.60	0.80	0.93	0.79	0.36	0.42	0.17	0.62
Cwa	-0.08	-0.00	-0.42	0.15	0.89	1.02	0.62	0.61	0.61	-0.05	-0.21	3.81
Am	1.65	-0.19	0.04	-0.54	0.14	0.26	0.13	0.83	1.78	1.52	1.36	1.12
Aw	0.66	0.63	0.04	0.74	0.70	0.63	0.85	0.96	0.75	0.74	0.70	1.37
Af/Am	1.03	0.05	0.06	-0.17	0.37	0.53	0.53	0.81	1.07	0.97	0.76	0.87
Af/Aw	0.54	0.46	0.06	0.47	0.65	0.71	0.89	0.88	0.56	0.58	0.43	1.00
Cwa/Am	0.79	-0.10	-0.19	-0.20	0.52	0.64	0.37	0.72	1.19	0.73	0.58	2.46
Cwa/Aw	0.29	0.31	-0.19	0.44	0.80	0.83	0.74	0.78	0.68	0.34	0.25	2.59
Am/Aw	1.16	0.22	0.04	0.10	0.42	0.44	0.49	0.90	1.26	1.13	1.03	1.24
Af/Am/Aw	0.91	0.24	0.05	0.13	0.48	0.56	0.63	0.86	0.96	0.89	0.74	1.04
Cwa/Am/Aw	0.75	0.15	-0.11	0.12	0.58	0.64	0.53	0.80	1.05	0.74	0.62	2.10
Zone	$b_{z,1}$	$b_{z,2}$	$b_{z,3}$	$b_{z,4}$	$b_{z,5}$	$b_{z,6}$	$b_{z,7}$	$b_{z,8}$	$b_{z,9}$	$b_{z,10}$	$b_{z,11}$	$b_{z,12}$
Af	0.04	0.25	0.67	0.48	-0.18	-0.58	-0.88	-0.66	0.15	0.04	0.57	-0.38
Cwa	1.79	1.51	2.62	0.82	-0.86	-1.30	-0.22	0.05	0.12	1.70	2.76	-9.92
Am	-2.39	1.27	0.89	1.64	0.66	0.34	0.65	-0.36	-2.04	-1.82	-1.78	-1.48
Aw	-0.36	-0.26	0.71	-0.40	-0.34	-0.20	-0.45	-0.58	-0.26	-0.34	-0.34	-1.89
Af/Am	-1.18	0.76	0.78	1.06	0.24	-0.12	-0.11	-0.51	-0.95	-0.89	-0.61	-0.93
Af/Aw	-0.16	-0.00	0.69	0.04	-0.26	-0.39	-0.67	-0.62	-0.05	-0.15	0.11	-1.13
Cwa/Am	-0.30	1.39	1.75	1.23	-0.10	-0.48	0.22	-0.15	-0.96	-0.06	0.49	-5.70
Cwa/Aw	0.72	0.63	1.66	0.21	-0.60	-0.75	-0.33	-0.26	-0.07	0.68	1.21	-5.91
Am/Aw	-1.38	0.51	0.80	0.62	0.16	0.07	0.10	-0.47	-1.15	-1.08	-1.06	1.68
Af/Am/Aw	-0.91	0.42	0.76	0.57	0.05	-0.14	-0.23	-0.53	-0.72	-0.70	-0.52	-1.25
Cwa/Am/Aw	-0.32	0.84	1.40	0.69	-0.18	-0.38	-0.00	-0.29	-0.73	-0.15	0.21	-4.43

Although A-P coefficients found in the literature generally lie in the range [0,1], many values listed in Tab. 4.1 lie outside this range. One reason therefor is that no upper or lower conditions were imposed on the regression analysis. Besides, the prevailing climate conditions in ASEAN may justify the presence of negative coefficients.

By definition, sunshine duration is the period for which direct solar irradiance exceeds a certain threshold (typically 120 W m^{-2}). Consequently, $b_{z,m}$ reflects the influence of direct radiation, and $a_{z,m}$ the impact of diffuse radiation. A combination of a positive $a_{z,m}$ and a negative $b_{z,m}$ means that a higher sunshine fraction leads to a lower clearness index, and that the share of diffuse radiation in the global radiation increases. Practically, this can be explained by the presence of high cloud layers during daytime that do not cover the sky completely. The amount of solar irradiance that reaches the ground is higher than the threshold value of the measuring instruments, so that these clouds do not prevent the meteorological instruments from measuring sunshine hours, while causing more diffusion in the sky. In most parts of ASEAN, the presence of clouds over a long period of time throughout the year is common due to the tropical conditions. On the other hand, a combination of a negative $a_{z,m}$ and a positive $b_{z,m}$ means that the clearness index is mostly driven by the direct radiation. This case has only occurred in the climate zones Cwa and Am (and their buffer zones) during the months with low clearness indices and low sunshine hours. One possible explanation is that the measuring instruments tend to underestimate the amount of sunshine hours during "winter", when the sun elevation angle is low. Negative $a_{z,m}$ values penalise the diffuse radiation to compensate for the underestimated impact of direct radiation. All in all, the A-P model coefficients are within an acceptable range and can be used to estimate the clearness indices for each climate zone.

4.4.2 Validation of the A-P model

The A-P model was validated in two steps. Firstly, the RMSE values using the clearness indices generated by the A-P model were compared to the RMSE values of the MERRA data at the twenty solar measurement stations. This comparison is necessary to check whether the A-P model presents any improvement in accuracy compared to the MERRA data. Secondly, the robustness of the model is tested by running the optimisation after removing each individual meteorological station separately. The results are displayed in Fig. 4.3.

According to Fig. 4.3, the A-P model using all the available radiation measurements provides more accurate results than MERRA data for 14 stations. At these stations, the improvements are considerable, especially in Hanoi, Vietnam (RMSE reduced by 0.31), Mengzi, China (-0.20), Jinghong, China (-0.19), and Jakarta, Indonesia (-0.14). For the other six stations, the RMSE values are in both cases low and do not vary a lot, with the worst deterioration occurring in Penang, Malaysia (0.07 for MERRA, versus 0.15 for the A-P model). On average, the RMSE for all the stations is reduced by almost 0.07 compared to MERRA. Using the A-P model, the RMSE values of 17 stations lie below 0.15, whereas there are only 10 stations that fulfil this condition for MERRA. All in all, applying monthly corrections using the climate-based A-P model allows a better estimation of the radiation potential of ASEAN than using unmodified MERRA data.

However, the errors remain higher than 0.15 in three stations, namely Jakarta (Indonesia), Bandung (Indonesia), and Chiang Mai (Thailand), with RMSEs of 0.31, 0.25, and 0.19, respectively. This is probably due to the quality of the measurements themselves, which were made in a short period of time in the late 1960s and early 1970s. The selected years may not fit to the long-term sunshine hours averages provided by CLIMWAT. Besides, the accuracy of the measuring instruments at that time and their compliance with the standards of the World Meteorological Organization (WMO) cannot be attested with certitude.

Testing the model without the data from one station shows that the RMSE values (blue crosses in Fig. 4.3) remain close to the previously obtained values (red dots). Using all the data available results in a lower average RMSE than any model with one missing station. Thus, it is better to use all the data available for this study, even though the quality of the measurements at some stations is questionable.

4.4.3 Clearness indices

The A-P model coefficients allow the calculation of the clearness index for any location where the sunshine hours data is available. This is particularly the case for the twenty stations with radiation data. It is possible to compare the monthly fluctuation according to measurements, to the A-P model, and to MERRA within these stations. For the sake of clarity, only one station for each climate zone was selected to draw the curves in Fig. 4.4.

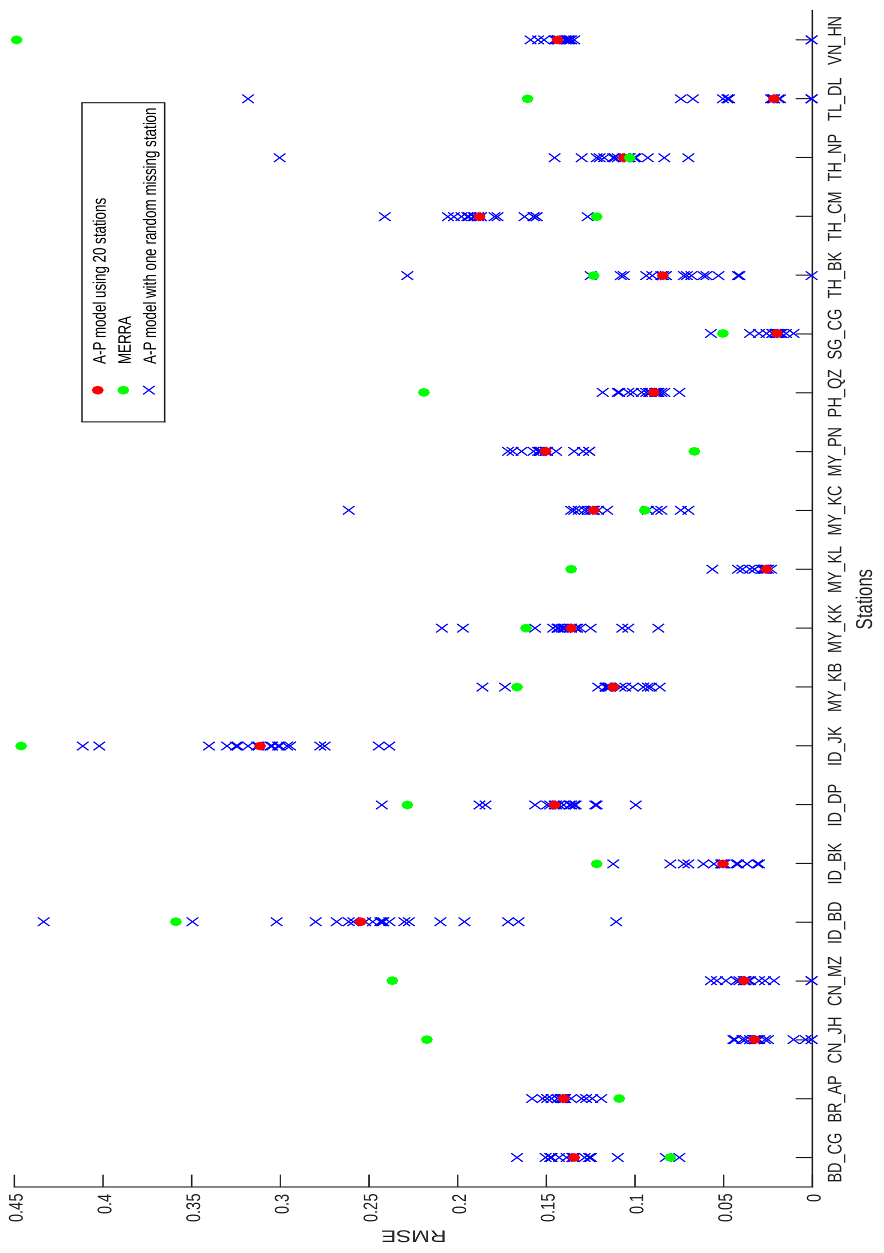


Figure 4.3: RMSE values for the A-P model generated data with 20 stations, for MERRA data, and for the A-P model with one random missing station.

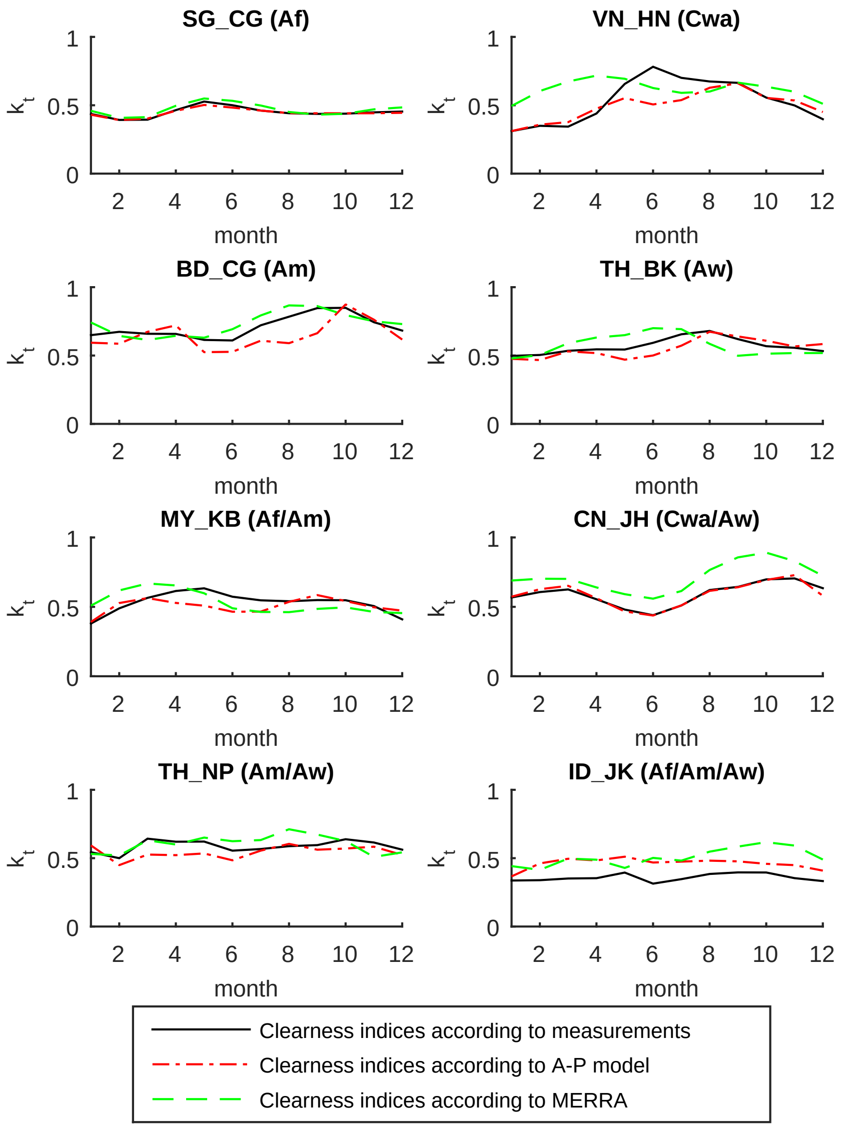


Figure 4.4: Comparison between monthly averages according to measurements, to the climate-based A-P model, and to MERRA for eight stations in different climate zones.

Fig. 4.4 confirms the advantages of the A-P model over MERRA. In fact, the A-P model can reproduce the monthly fluctuations in a better way than MERRA, which shows a non-negligible time lag for some locations, e.g. Hanoi, Vietnam. However, considerable errors persist for certain locations and months, which are either due to the quality of the input data or to the limitations of the climate-based model itself.

4.4.4 Solar resource potential in ASEAN

The measurements of sunshine hours are available at 249 stations, which are geographically well-distributed over ASEAN and its bordering regions. This data will be first interpolated using Empirical Bayesian Kriging with a resolution of 15 arcsec, then used to estimate the solar radiation in ASEAN. The resulting map of average daily GHI values across ASEAN is shown in Fig. 4.5.

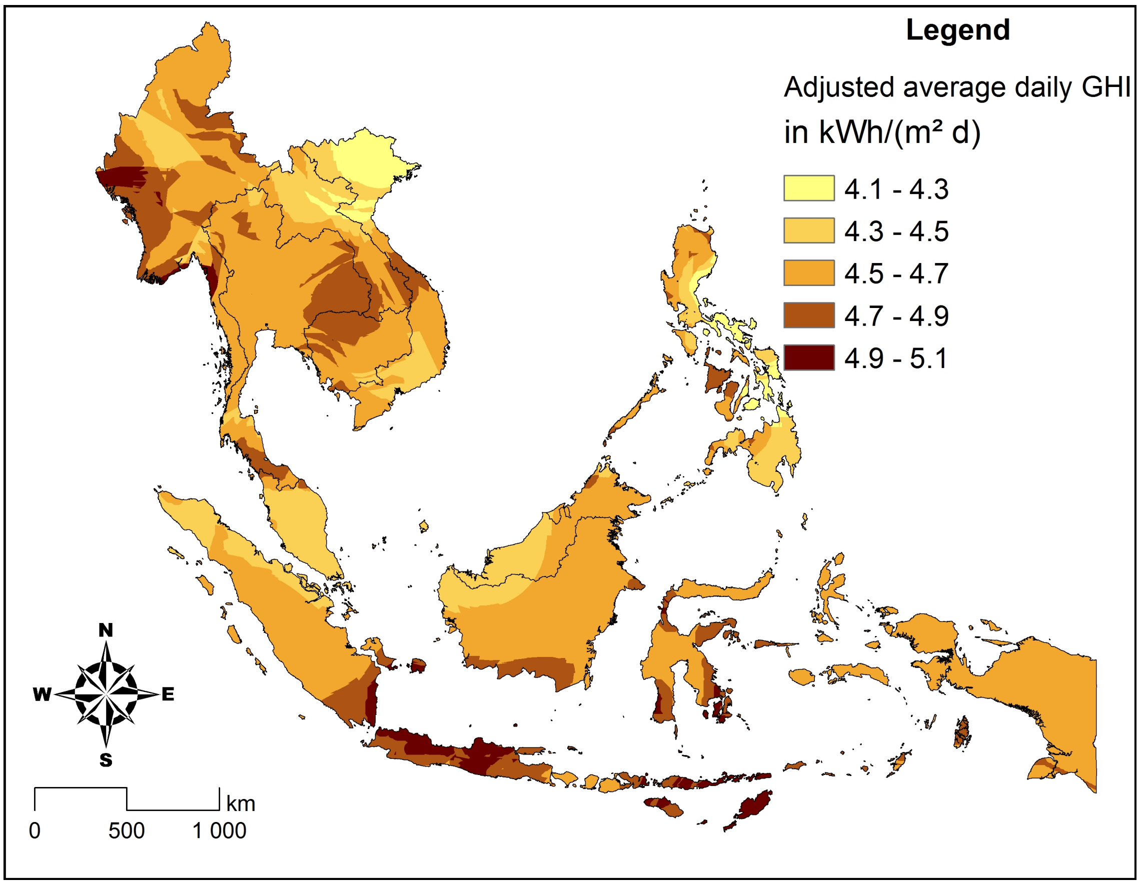


Figure 4.5: Average daily GHI in ASEAN.

According to Fig. 4.5, the solar radiation potential in ASEAN lies within a narrow range. The lowest GHI averages ($4.1\,\mathrm{kW\,h\,m^{-2}}$) are observed in northern Vietnam and eastern Visayas, Philippines. The areas with the highest potential experience a GHI of $5.1\,\mathrm{kW\,h\,m^{-2}}$, and are located in Timor, Java, Sulawesi, and southern Sumatra (Indonesia), and western Myanmar. Eastern Thailand and central Vietnam also have a high solar potential, with GHI values above $4.7\,\mathrm{kW\,h\,m^{-2}}$.

4.4.5 Hourly global-to-diffuse ratio

Based on the hourly measurements of global, diffuse, and direct irradiance in Bukit Kototabang, Indonesia, a relationship between the clearness indices, the elevation angles, and the diffuse fractions was derived. In Fig. 4.6, the scatter corresponds to the input data for Bukit Kototabang, whereas the planes correspond to the pieceweise linear functions of Eq. (4.15). The colour changes gradually from blue to yellow with increasing value of $sin(\alpha)$.

The equations for the planes were derived by means of polynomial fitting and are expressed as follows:

$$k_{d,h} = \begin{cases} -0.418k_{t,h} - 0.072sin(\alpha) + 1, & 0 < sin(\alpha) \leq 0.2 \\ -1.219k_{t,h} - 0.909sin(\alpha) + 1.520, & 0.2 < sin(\alpha) \leq 0.4 \\ -1.346k_{t,h} - 0.605sin(\alpha) + 1.601, & 0.4 < sin(\alpha) \leq 0.6 \\ -1.358k_{t,h} - 0.201sin(\alpha) + 1.432, & 0.6 < sin(\alpha) \leq 0.8 \\ -1.365k_{t,h} + 0.108sin(\alpha) + 1.216, & 0.8 < sin(\alpha) \leq 1 \end{cases} \quad [-] \quad (4.15)$$

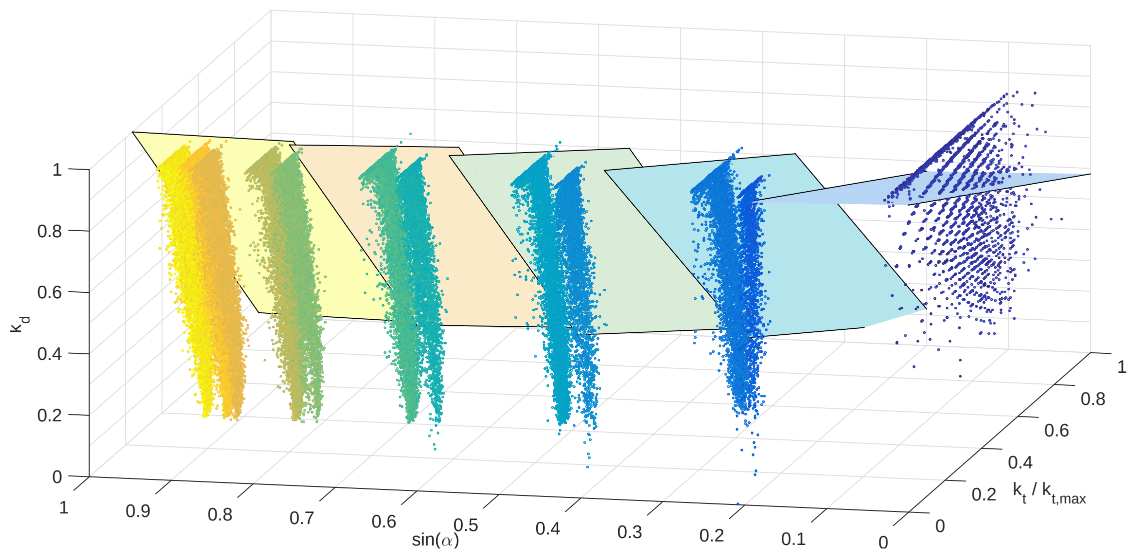


Figure 4.6: Relationship between the clearness indices, the elevation angles, and the diffuse fractions for Bukit Kototabang.

with $k_{d,h}$ the hourly diffuse fraction defined within the range [0.05,1], i.e. values delivered by Eq. (4.15) outside this range will be replaced by the closest possible value.

For Bukit Kototabang, the clearness indices are generally very low (< 0.3), except in the early morning or late afternoon. Despite the limited range for $k_{t,h}$ during the day, all values for the diffuse fractions are possible according to the scatter cloud in Fig. 4.6. Nevertheless, the diffuse fraction is assumed to be close to 1 for low clearness indices, which means that the PV model in this study will overpredict the diffuse fraction in locations similar to Bukit Kototabang.

4.4.6 PV technical potential in ASEAN

First, a map of the theoretical full-load hours (FLH) of PV modules is created, assuming that they are installed in every suitable location in ASEAN, as defined by the technical constraints. This map is shown in Fig. 4.7. Uncoloured areas are considered unsuitable for both urban and rural PV projects and cover mainly mountainous areas and water bodies. Similarly to Fig. 4.5, the best locations for PV projects are in Java, southern Sumatra (Indonesia), central Myanmar, and eastern Thailand, with values exceeding 1350 h, equivalent to an average capacity factor of 15.4 %. A correlation between the climate zone classification and the PV output is visible, by superposing Fig. 4.2 and Fig. 4.7. For instance, the FLH in the climate zone Cwa are particularly low. On the one hand, this could be due to the geographical location of this climate zone in the very north of ASEAN which leads to more intense seasonal fluctuations and hence less solar radiation during winter time. On the other hand, the climate zone Cwa is by definition characterised by humid summers which in turn could lead to increased scattering and therefore a lower PV output. Besides, the climate zone Am offers high FLH values which could be explained by the dry summers which are characteristic for this zone and lead to reduced scattering of solar radiation.

Second, the energy yield (E_{out})of the PV module was calculated for the possibly available land area in ASEAN. The results are aggregated for every country and land-use type and summarised in Tab. 4.2. Instead of converting all the possibly suitable terrains into PV farms, an availability factor was included. For the conservative scenario, only 10 % of barren lands, grasslands, and shrublands are considered available for rural PV projects, and 5 % of urban areas can be covered with rooftop PV panels. In reality, increasing power demand and high fossil fuel costs will lead to the conversion of more terrains into PV farms. For the realistic scenario, 1 % of forests, savannas, croplands, and natural vegetation terrains can be used for PV projects, whereas 50 % of barren lands, grasslands, and shrublands are considered available.

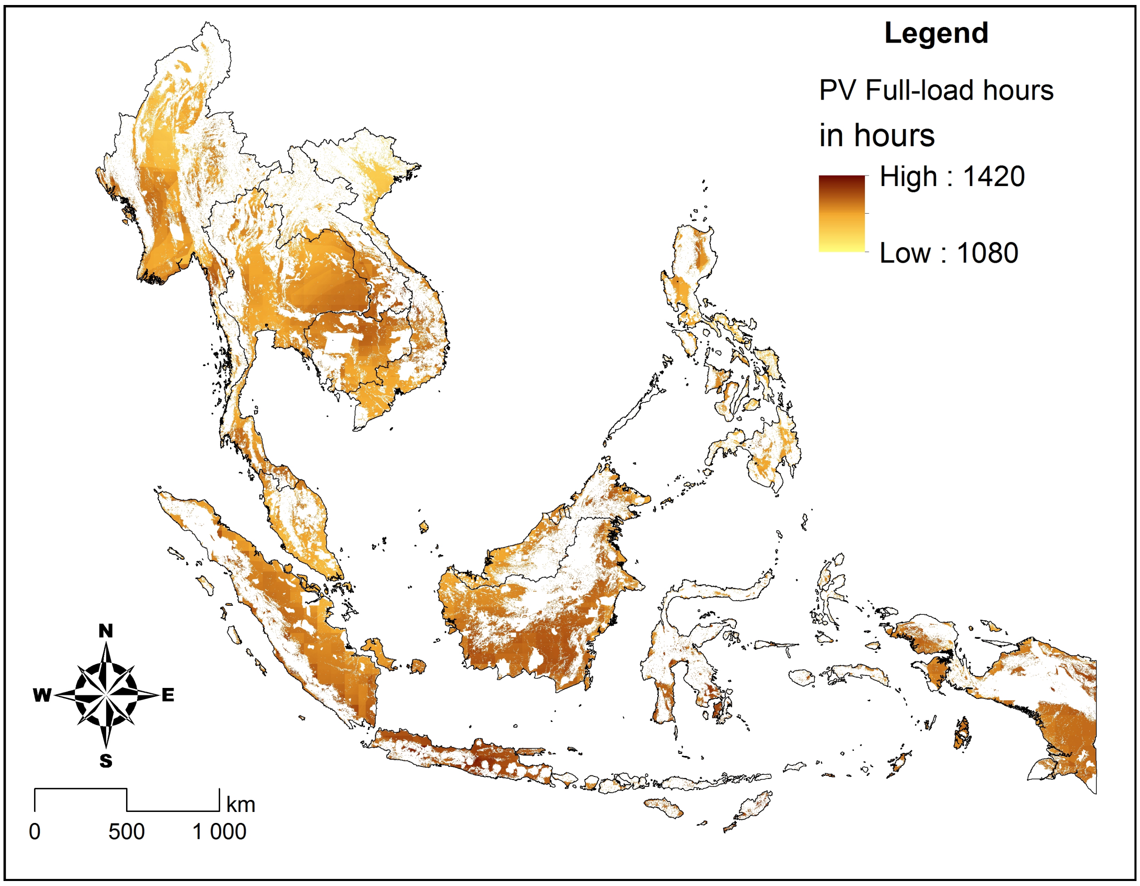


Figure 4.7: Theoretical FLH of PV modules installed in every available location in ASEAN.

In terms of the PV generation potential, the country size plays a determinant role. In fact, the ASEAN countries with the largest land area (Indonesia, Myanmar, Thailand) have the highest technical potential in the conservative scenario (144.4 TW h, 118.2 TW h, and 86.4 TW h, respectively). However, there is no obvious linear correlation between the country size and the energy yield. For example, this trend does not apply for smaller countries, because Cambodia (52.8 TW h, 8th area-wise) has a higher potential than Malaysia (24.2 TW h, 5th area-wise). Thus, the variations between the countries regarding the technical potential are not only caused by the differences in total area, but also by other factors.

As stated previously, the resource potential in ASEAN lies within a narrow range. The similar capacity factor averages between ASEAN countries confirm this claim. Consequently, the small variations of the resource potential between the countries do not justify the differences in their technical potential.

By elimination, the disparities in the technical potential are mainly due to the technical constraints adopted in this study. The impact of the terrain constraints is shown in Tab. 4.2. In fact, the share of suitable terrains varies considerably between ASEAN countries. The city-state of Singapore, with its flat, urban landscapes has by far the highest share of suitable areas (83.3 %), followed by Thailand (59.4 %) and Cambodia (58.2 %). On the other hand, mountainous countries like Laos (23.2 %) and the Philippines (29.9 %) have the lowest suitability ratios. The average within ASEAN lies at 43.0 %. Additionally, the results are very sensitive to the land-use types observed in every country. In the conservative scenario, countries with a high share of barren lands, grasslands, shrublands, and urban areas such as Singapore, Brunei, Cambodia, and Vietnam are favoured. In the realistic scenario, the share of forests, savannas, croplands, and natural vegetation mosaics has a more important impact for most countries, even though their availability factor is low (1 %). All in all, 0.08 % of ASEAN's land area ($\approx 3500\,\mathrm{km}^2$) could generate 430 TW h in the conservative scenario. With more allocated areas for PV projects (ninefold, $\approx 31\,000\,\mathrm{km}^2$), it is possible to reach 3964 TW h in the realistic scenario, with Indonesia, Myanmar, and Thailand generating about 70 % of it.

The spacing of the PV panels influences the Ground Cover Ratio (GCR), and thus the power capacity density. In this study, the highest GCR (88.9 %) is observed at the Equator and decreases gradually north and south of it. In southern Indonesia, it reaches 81.9 %, whereas in northern Myanmar, it declines to 57.7 %. Consequently, a PV farm at the Equator may be 1.5 times denser than another one in northern Myanmar or Vietnam. Obviously, this factor also has an impact on the technical PV potential of ASEAN countries.

The influence of all the physical and technical constraints on the results is summarised in Fig. 4.8. The average solar irradiations over ASEAN, the energy losses, and the total theoretical energy yield within a year are displayed in a Sankey diagram.

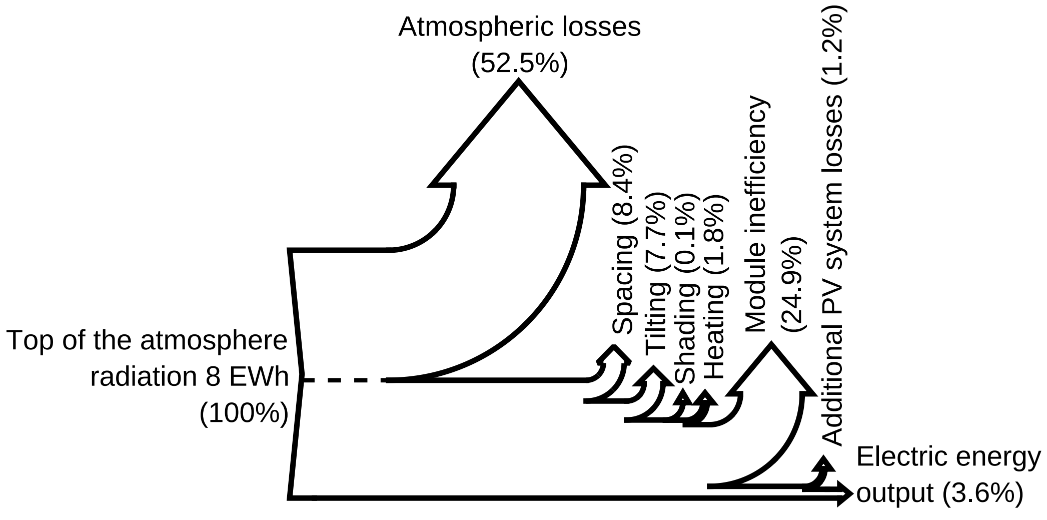

Atmospheric losses (52.5%)
Spacing (8.4%)
Tilting (7.7%)
Shading (0.1%)
Heating (1.8%)
Module inefficiency (24.9%)
Additional PV system losses (1.2%)
Top of the atmosphere radiation 8 EWh (100%)
Electric energy output (3.6%)

Figure 4.8: Sankey diagram of the average solar irradiations over ASEAN, the energy losses, and total theoretical energy yield within a year.

More than 50 % of the solar radiation at the top of the atmosphere above ASEAN is either reflected, absorbed, or scattered by the atmospheric particles. Comparatively, all the other losses are negligible. Since the PV panels are spaced according to the GCR, only 82.3 % of the irradiation that reaches the surface actually hits the PV panels. The tilting and orientation of the PV panels in ASEAN are disadvantageous in terms of power generation, but necessary to avoid maintenance costs. They account for 7.7 % of the losses (19.7 % less energy than on horizontal panels). The presence of mountains and hills causes topographic shading in the early morning or late afternoon, but has little incidence on the power losses which do not exceed 0.1 % in ASEAN. The hot, tropical climate conditions cause the cell temperature of the modules to exceed the STC temperature, so that heat losses reach 1.8 % of the top-of-the-atmosphere irradiation, or 5.8 % of the incident irradiation. Other losses caused by the inefficiency of the PV cells account for 24.9 % of losses based on the input energy, and another 1.2 % are due to additional system losses (e.g. converter

Table 4.2: Theoretical PV generation potential in ASEAN by country and land-use type.

	Land-use	% A_{total}	Avail. [%]	$\overline{CF}$ [%]	E_{out} [TW h]		Land-use	% A_{total}	Avail. [%]	$\overline{CF}$ [%]	E_{out} [TW h]
Brunei	44% of F/S	38.8%	0 – 1	14.7	0 – 3.0	Myanmar	23% of F/S	17.0%	0 – 1	14.3	0 – 139.8
	87% of C/NV	4.1%	0 – 1	14.6	0 – 0.3		91% of C/NV	21.0%	0 – 1	14.3	0 – 178.1
	88% of B/G/SH	0.1%	10 – 50	14.6	0.1 – 0.2		69% of B/G/SH	0.9%	10 – 50	14.3	77.8 – 389.2
	85% of U	3.0%	5	14.5	1.2		91% of U	0.2%	5	14.3	8.6
	Total	**45.9%**	**0 – 1**	**14.6**	**1.2 – 4.7**		**Total**	**39.2%**	**0 – 2**	**14.3**	**86.5 – 715.7**
Cambodia	49% of F/S	29.2%	0 – 1	14.8	0 – 71.5	Philippines	15% of F/S	7.1%	0 – 1	14.1	0 – 27.0
	84% of C/NV	27.3%	0 – 1	14.5	0 – 65.4		47% of C/NV	21.7%	0 – 1	14.3	0 – 83
	84% of B/G/SH	1.6%	10 – 50	14.6	38.1 – 190.6		76% of B/G/SH	0.1	10 – 50	14.4	3.4 – 16.8
	87% of U	0.2%	5	14.4	1.8		90% of U	0.9%	5	14.2	17.4
	Total	**58.2%**	**0 – 2**	**14.6**	**39.9 – 329.4**		**Total**	**29.9%**	**0 – 1**	**14.2**	**20.8 – 144.2**
Indonesia	42% of F/S	32.3%	0 – 1	14.8	0 – 838.4	Singapore	75% of F/S	9.0%	0 – 1	13.8	0 – 0.1
	70% of C/NV	12.0%	0 – 1	14.9	0 – 313.2		98% of C/NV	16.9%	0 – 1	13.8	0 – 0.1
	45% of B/G/SH	0.2%	10 – 50	15.0	39.2 – 196.1		100% of B/G/SH	1.1%	10 – 50	13.8	0.1 – 0.4
	89% of U	0.5%	5	15.1	69.1		99% of U	56.3%	5	13.8	2.3
	Total	**45.0%**	**0 – 1**	**15.0**	**108.3 – 1416.8**		**Total**	**83.3%**	**4**	**13.8**	**2.4 – 2.9**
Laos	16% of F/S	14.2%	0 – 1	14.6	0 – 43.0	Thailand	12% of F/S	4.3%	0 – 1	14.5	0 – 28.4
	75% of C/NV	8.2%	0 – 1	14.6	0 – 24.8		88% of C/NV	53.7%	0 – 1	14.5	0 – 362.6
	90% of B/G/SH	0.8%	10 – 50	14.7	24.6 – 123.1		92% of B/G/SH	0.5%	10 – 50	14.5	36.1 – 180.7
	93% of U	0.1%	5	14.4	1.1		96% of U	0.8%	5	14.4	28.4
	Total	**23.2%**	**0 – 3**	**14.6**	**25.7 – 192.0**		**Total**	**59.4%**	**0 – 2**	**14.4**	**64.5 – 600.0**
Malaysia	34% of F/S	28%	0 – 1	14.4	0 – 122.6	Vietnam	15% of F/S	8.0%	0 – 1	14.3	0 – 34.1
	78% of C/NV	10.2%	0 – 1	14.3	0 – 44.3		67% of C/NV	25.3%	0 – 1	14.1	0 – 106.2
	76% of B/G/SH	0.1%	10 – 50	14.1	2.7 – 13.7		80% of B/G/SH	0.9%	10 – 50	14.3	40.0 – 200.0
	86% of U	0.7%	5	14.1	16.1		92% of U	1.0%	5	14.0	21.4
	Total	**39.0%**	**0 – 1**	**14.2**	**18.8 – 196.6**		**Total**	**35.3%**	**0 – 2**	**14.2**	**61.4 – 361.7**

A_{total}: country total area, $\overline{CF}$: countrywide average capacity factor, F/S: Forest/Savanna, C/NV: Cropland/Natural Vegetation, B/G/SH: Barren/Grassland/Shrubland, U: Urban and transformer losses). All in all, from the 8 EW h hitting the atmosphere above the suitable terrains in ASEAN initially, only 3.6 % can be converted into electricity flowing into the grid. Less than 1 % come from reflected radiation, the rest is almost evenly split into direct and diffuse radiations.

4.5 Conclusions

In this paper, the resource and the technical potential of solar PV in South East Asia was determined. Therefore, two empirical models, which focus specifically on the ASEAN region and rely exclusively on publicly available input data with global coverage, were proposed. On the one hand, a climate-based A-P model was developed in order to receive monthly clearness indices with a high spatial resolution. On the other hand, a decomposition model was presented to estimate the diffuse fraction of the solar radiation, depending on clearness index and solar elevation angle. The model results were validated by comparison with real measurements. The two models were developed on a modular basis and can be used separately in further studies.

The solar irradiation at the top of the atmosphere is very high in South East Asia due to its location at the Equator. However, because of the cloudy tropical climate, the yearly average surface irradiation varies between 4.1 $\mathrm{kW\,h\,m^{-2}\,d^{-1}}$ and 5.1 $\mathrm{kW\,h\,m^{-2}\,d^{-1}}$, which is low compared to neighbouring regions in India and Australia. Nevertheless, due to the large land area which is suitable for PV installations, ASEAN offers a large potential of power generation from PV. This paper confirms the existence of a geographic correlation between Köppen-Geiger climate classification and the PV technical potential.

Applying conservative restrictions on the available terrain, 0.08 % of ASEAN's land area could be used for PV, generating 430 TW h annually. Annual power generation could be increased to 3964 TW h if 0.69 % of the total land area were used for PV installations. Therefore, solar PV could significantly contribute to supplying the strongly increasing power demand in ASEAN, especially for countries with very limited fossil resources and reserves.

Furthermore, the impact of technical constraints on the resulting PV potential was evaluated in order to find possible model simplifications with low influence on the final result. It was found that the share of energy output caused by ground-reflected radiation is relatively low and hence could be neglected. Besides, in the absence of high-resolution elevation data, the shading losses cannot be estimated accurately and may also be neglected. On the other hand, heat losses have a high influence on the final result and therefore should be analysed in detail. Further module types with lower temperature coefficients might be considered. Finally, the optimal GCR and tilting angle should be determined in relationship with terrain acquisition costs and maintenance costs, in order to maximise profitability.

Acknowledgements

This work was financially supported by the Singapore National Research Foundation under its Campus for Research Excellence And Technological Enterprise (CREATE) programme.

4.A Solar radiation measurement stations

Table 4.A.1: Weather stations with GHI measurements in ASEAN and neighbouring regions.

N	Country	Station	Code	Latitude	Longitude	Elevation [m]	Period	Climate
1	Bangladesh[a]	Chittagong	BD_CG	22.35°N	91.49°E	33	1966 - 1971[b]	Am
2	Brunei	Airport	BN_AP	4.93°N	114.93°E	22	1987 - 1995	Af
3	China[a]	Jinghong	CN_JH	21.97°N	100.76°E	550	1979 - 2000	Cwa/Aw
4	China[a]	Menzi	CN_MZ	23.38°N	103.38°E	1302	1979 - 2000	Cwa
5	Indonesia	Bandung	ID_BD	6.83°S	107.61°E	1310	1970 - 1972[b]	Af/Am
6	Indonesia	Bukit Kototabang	ID_BK	0.20°S	100.31°E	864	1996 - 2013	Af
7	Indonesia	Denpasar	ID_DP	8.66°S	115.21°E	6	1970 - 1973[b]	Af/Am/Aw
8	Indonesia	Jakarta	ID_JK	6.18°S	106.83°E	8	1970 - 1973[b]	Af/Am/Aw
9	Malaysia	Kota Bharu	MY_KB	6.16°N	102.28°E	5	1979 - 2009	Af/Am
10	Malaysia	Kota Kinabalu	MY_KK	5.93°N	116.05°E	2	1989 - 2007	Af/Am
11	Malaysia	Kuala Lumpur	MY_KL	3.11°N	101.55°E	17	1979 - 2009	Af
12	Malaysia	Kuching	MY_KC	1.48°N	110.35°E	20	1989 - 2009	Af
13	Malaysia	Penang	MY_PN	5.30°N	100.26°E	3	1979 - 2009	Af
14	Philippines	Quezon	PH_QZ	14.63°N	121.06°E	51	1979 - 1996	Af/Am/Aw
15	Singapore	Changi	SG_CG	1.36°N	103.98°E	15	1979 - 2004	Af
16	Thailand	Bangkok	TH_BK	13.66°N	100.61°E	60	1979 - 2013	Aw
17	Thailand	Chiang Mai	TH_CM	18.78°N	98.98°E	313	1966 - 1973[b]	Aw
18	Thailand	Nakhon Phanom	TH_NP	17.41°N	104.78°E	145	1966 - 1970[b]	Am/Aw
19	Timor-Leste[a]	Dili	TL_DL	8.58°S	125.58°E	4	1964 - 1974[b]	Aw
20	Vietnam	Hanoi	VN_HN	21.03°N	105.85°E	5	1990 - 2001	Cwa

[a] These stations lie in countries outside ASEAN, but have similar climatic conditions to neighbouring ASEAN regions.

[b] MERRA data is unavailable prior to 1979. Measurements prior to 1979 are treated as if they were taken in the following solar cycles, i.e. after 11 or 22 years.

Chapter 5

Decreasing the spatial resolution: Application of geoclustering

The second publication introduces a novel clustering method for high resolution data in order to obtain regions with specific characteristics. It applies data-driven techniques to decrease the resolution by ensuring that the loss of information is less critical than conventional aggregation methods. Hence, it provides examples for the techniques that are described in Section 3.3. Moreover, it shows how the choice of the model regions affects the results of an energy system optimization. While the underlying data is the same, and the number of regions is kept constant, the mere variation of the shapes of the regions leads to large differences in the cost-optimal energy mix, particularly in the future under stringent CO_2 constraints.

As a first author of the publication, I was responsible for the idea, structure and content of the paper. I analyzed the clustering results qualitatively and quantitatively, and also run the *urbs* optimization and compared the results. Besides, I took care of the communication with the journal and the reviewers. My co-author, Mohammad Youssef Mahfouz, wrote the first version of the clustering code and contributed to the introduction and to the description of the clustering method. He also created Figure 5.2 and the pseudocodes in the annex.

Since the publication of the article, I have continued developing the code with the help of my students. It is now available as an open-source tool written in Python and published in GitHub. Some of the new features include the possibility to cluster multiple rasters at the same time, the flexibility in choosing the compression ratio at each stage of the algorithm, and the addition of hierarchical clustering based on transmission line capacities. The latest documentation of the tool is included in Annex B.

Title: Impact of the choice of regions on energy system models

Authors: Kais Siala, Mohammad Youssef Mahfouz

Journal: Energy Strategy Reviews (Elsevier)

Publication date: August 2019

Abstract

Many existing energy system models rely on input data available at country-level, or at the level of administrative divisions. However, there is usually no correlation between the distribution of data such as solar radiation, wind speed, and electrical load on one hand, and the administrative divisions on the other hand. The goal of the research is to measure the impact of the shape of model regions on the results of system optimization models. A novel clustering methodology for high-resolution data is presented and applied to define new regions for an energy system model which optimizes expansion planning and unit commitment. The new model regions take into account the bottlenecks in the transmission system and their effect on the expansion of renewable energy sources. We compare the obtained energy mixes, new capacities, and curtailment levels against a model using administrative divisions. The results show discrepancies between the models in the case of a high share of variable renewable energy, and quantify the impact of the distribution of load, wind and solar resources on energy system models. Possible applications of the new model regions are discussed to emphasize their utility for modelers and policy-makers.

Keywords: Spatial clustering; Energy system model; Expansion planning; Optimization; GIS

5.1 Introduction

The electricity system in Europe is increasingly relying on decentralized generation from variable renewable energy sources. The conventional power plants that used to be conveniently located next to load centers are being replaced with wind and solar power plants in areas with high renewable potential. This trend is set to continue if the decarbonization targets are to be achieved without relying heavily on nuclear and hydro power generation.

The increased reliance on geographically distributed, time-dependent renewable energy generation requires a deep understanding of the spatial and temporal distribution of wind and solar resources, and their eventual correlation with load patterns (Pfenninger et al., 2014). This is helpful to determine the system flexibility requirements, such as the need for new transmission lines or for storage devices. Thanks to improvements in the processing power, it is now possible to solve multi-regional optimization problems with various wind and solar potentials. The advantage of using a high number of model regions has been quantified in previous studies (Frew and Jacobson, 2016). However, the definitions of the model regions has been mostly dictated by the political boundaries of countries and their administrative subdivisions, especially if the models span over many states or support policy-makers on an international level (Frew and Jacobson, 2016; Gago Da Camara Simoes et al., 2013). The lack of diversity in the shape of model regions might be partly due to the fact that the input data of energy system models are usually available on the level of administrative divisions, and are seldom readily available to be used in another spatial configuration.

One way of achieving a higher flexibility in the definition of regions is to first obtain or generate high resolution data using geographic information systems (GIS), and them cluster them into categories or groups, depending on certain trends in the data set. The generation of high resolution data is a critical research topic that has been addressed by previous studies. In this analysis, we assume that high resolution data is readily available and we focus therefore on the clustering step.

Since Lloyd (1982) published the *k-means* algorithm as a clustering technique, other variants have been developed to accommodate a wide range of user requirements, as explained by Jain and Dubes (1988). For spatial clustering, it is possible to use such algorithms and add contiguity constraints to them, as done by Adam et al. (2012) in their analysis of the pan-European electricity system for 2050. But as data analysis developed and GIS became more common, new algorithms specific to spatial clustering emerged, such as the *max-p regions* algorithm by Duque et al. (2012).

Despite these developments, the ability of spatial clustering algorithms like *max-p regions* is still limited to small data sets with hundreds of data points. The novelty of our work resides in the clustering of high resolution spatial data (tested with $\sim 10^8$ data points) into contiguous regions that can be used in energy system optimization models. We apply the method in the case of Europe to compare the results of the models in 2015 and in 2050 under the constraint of a 95% CO_2 emissions reduction.

Hence, we structured our analysis in the following way. First, we go through the clustering method used to obtain the new regions in Section 5.2. The obtained clusters are described qualitatively and quantitatively in Section 5.3. Then, in Section 5.4, we introduce *urbs*, the open-source model framework used for the optimization of the energy system of Europe. The input data and the assumptions for the models are described in the same section. Finally, we analyze the results of the optimization in Section 5.5, followed by a conclusion and a discussion of possible applications. An overview of the paper workflow is shown in Figure 5.1.

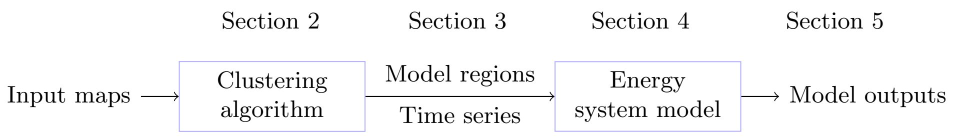


Figure 5.1: Paper workflow.

5.2 Spatial clustering

The goal of the spatial clustering is to create contiguous clusters with maximum data homogeneity within each one of them. These clusters are used afterwards as model regions in an energy system optimization model.

Since Fischer (1980) laid the groundwork for the taxonomy of spatial clustering problems, several algorithms were developed to either decrease the running time or to optimize the clustering process. One of the main problems with the existing spatial clustering algorithms is their inability to handle data with huge resolution. Therefore, a new approach is adopted in this paper, which applies two existing algorithms (*k-means++* and *max-p regions*) in a three-step process. A short description of the process is provided below. The inputs and outputs of the clustering are also presented for the case of Europe.

5.2.1 Clustering method

The clustering method introduced in this section is a multi-stage process involving the *k-means++* and *max-p regions* algorithms. A formal description of the process is provided in Annex 5.A. The code is also available as open source (Mahfouz et al., 2019).

The core of *k-means++* is the standard *k-means* algorithm developed by Lloyd (1982) which assigns each data point to the nearest centroid of k initial clusters. Arthur and Vassilvitskii (2007) enhanced the *k-means* algorithm by choosing the initial centroids such that they are as far away from each other as possible. The algorithm is fast and capable of handling a big amount of data, but it does not produce spatially contiguous regions. If a contiguity constraint is added in the cost function of the distance minimization, the algorithm generates compact clusters with the shape of Voronoi polygons with the respective centroids at their centers. Enforcing a strict contiguity constraint would lessen the importance of the data homogeneity within each cluster. Hence, *k-means++* cannot be applied solely to achieve the desired outcome of the study.

The spatial contiguity and the data homogeneity within clusters are, on the other hand, the strengths of the *max-p regions* clustering algorithm (Duque et al., 2018). The algorithm is one of the first mixed-integer programming (MIP) spatial clustering algorithms developed for the p-regions problem (Duque et al., 2012). It clusters data of n regions into p contiguous clusters where every cluster satisfies a minimum threshold value, such as the minimum solar energy potential that should exist in every cluster. The information lost due to clustering is minimal because the algorithm produces the maximum number of clusters that can be achieved in regard to the set threshold value. That number is unknown, but it increases if the threshold value decreases. With proper understanding of the data variance, the number of output clusters can be known to be within a certain range. The contiguous shapes of the clusters depend only on the input data and are not necessarily compact. Moreover, it is well implemented in python in the pySAL library (Rey et al., 2018) and in GeoDa (Anselin et al., 2006), which makes it user-friendly. The main drawback for *max-p regions* is the inability to handle huge data. According to Duque et al. (2012), the max-p regions problem algorithm has computational complexity of $(n-1)n^2 + \frac{n^2-n}{2}$ variables and $3n + (n-1)n^2 + n\frac{n^2-n}{2}$ constraints, where n is the number of data points to be clustered. This corresponds to a complexity of $O(n^3)$ according to the Bachmann—Landau notation. Hence, if the number of areas increases, the problem becomes intractable (Duque et al., 2012). Therefore, *max-p regions* cannot handle our input data without decreasing its resolution drastically.

Our approach combines the strengths of both algorithms to achieve the desired outcome:

1. Starting with rasters of $\sim 10^8$ data points, we apply the "divide and conquer" principle to split the data into 100 equally-sized rasters that can be processed in parallel (see Figures 5.2a and 5.2b).

2. We then cluster the data of each raster part (at most $\lesssim 1.27 \cdot 10^6$ numerical data points) using *k-means++*, as exemplified in Figures 5.2c and 5.2d. In order to determine the number of clusters k_i for every map tile a_i, we first search for a reference area a_r which has the highest product of relative standard deviation σ_r/σ_{max} and relative size (number of valid data points) n_r/n_{max}. The elbow method (Thorndike, 1953) was used to determine the number of clusters k_r for a_r. Second, for each tile a_i, the number of clusters k_i was calculated using the following expression:

$$k_i = k_r \cdot (0.7\frac{n_i}{n_{max}} + 0.3\frac{\sigma_i}{\sigma_{max}}). \tag{5.1}$$

   We end up with $\lesssim$ 100 clusters for each map part. This step is fast ($\sim$ 3h for all tiles) and leads to the largest data compression (down to 1 cluster for $\sim$ 15000 data points in the case of load).

3. The output of *k-means++* is a raster for each tile, which we polygonize into a shapefile.

4. For each tile a_i, we run the *max-p* algorithm. We define the threshold of the *max-p* algorithm as a function of relative standard deviation and relative number of valid data points, as described in the following equation:

$$thr_i = A \cdot e^{-B(\frac{n_i}{n_{max}} + C\frac{\sigma_i}{\sigma_{max}})}, \tag{5.2}$$

   where *thr*$_i$ is the threshold for tile a_i and A, B, and C are three global parameters for the complete data set. We determine A, B, and C through regression so that:

   - if a tile has the smallest n_i and the smallest σ_i, it will be split into at most two clusters;
   - if a tile has the largest n_i and the largest σ_i, it will have a very low threshold so that it may retain all of its parts from *k-means++*;
   - if a tile has the smallest size but the highest standard deviation, its threshold will be average.

Despite a limited data compression (the total number of clusters is almost halved), this step lasts $\sim$ 3h for all the tiles.

5. After applying the *max-p* algorithm to every tile a_i, we merge all the tiles together again to get the full map of Europe (see Figure 5.2f). The total number of clusters at this stage is $\lesssim$ 1800 in all three maps, which can be clustered at the next step.
6. *max-p* algorithm is applied on the whole map to obtain the final map shown in Figure 5.2g. In this study, we chose a target number of 28 regions to match the number of countries in Europe that are usually used as model regions. In order to obtain exactly 28 regions at the end, we varied the threshold through trial and error. The algorithm required 5 to 8 hours for the clustering of $\sim$ 1800 data points.

5.2.2 Clustering input

As mentioned before, the method used in this analysis is applicable on raster data sets with a high resolution, which cannot be clustered with standard methods. In the following we will use three rasters of Europe in 15 arcsec resolution: one for the wind potential (expressed in $\mathrm{kWh/kW_p}$), one for the photovoltaic (PV) potential (in $\mathrm{kWh/kW_p}$), and one for the load density (in MWh/pixel/a) (Siala and Mahfouz, 2018). The rasters for the solar and wind potentials and their corresponding model regions after the clustering are displayed in Figure 5.3. For the load distribution map, please refer to the Figures 5.2a and 5.2g.

5.3 Results of the spatial clustering

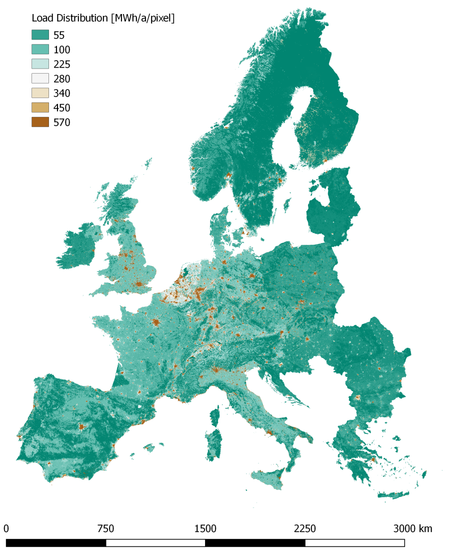
Before using the clusters to create energy system models, we analyze them qualitatively and quantitatively by describing them and comparing them to each other and to regions delimited by national borders.

5.3.1 Qualitative description

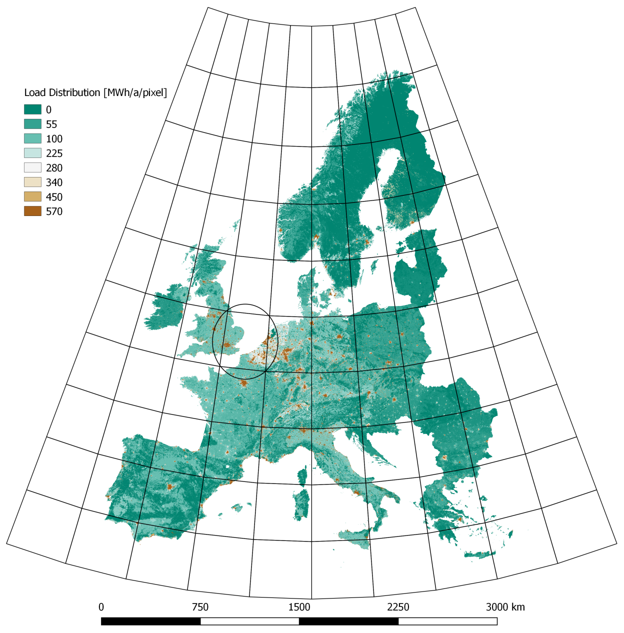
The outputs of the clustering process have different characteristics that might impact the results of the energy system optimization. The *Load* map is characterized by small, stretched out regions in the densely populated areas of Central Europe, between Northern Italy and the Netherlands, and by large regions in the periphery (Scandinavia, Iberian Peninsula, Eastern Europe). The *Solar* map is mostly made of equally sized regions that have the shape of horizontal bands, since the solar potential highly correlates with the latitude. Regarding the *Wind* map, we observe that the shapes of the clusters are affected by the elevation (Northern Spain, Norway) and by the distance to the shore, with many distinct clusters along the coast and others that are mostly continental.

5.3.2 Information loss

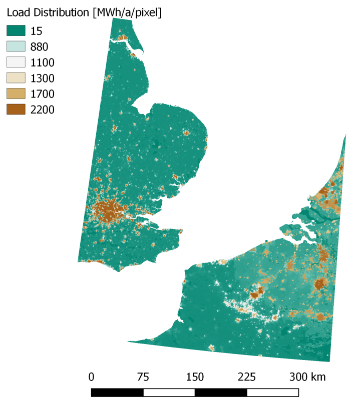
Clustering is used in this study to summarize the large amount of information contained in the high resolution maps into homogeneous groups. The compression of data leads inevitably to loss of information at the different stages of the algorithm, as displayed in Table 5.1. The largest data compression occurs after running *k-means++*. The compression ratios at that stage are affected by the maximum number of clusters k_r, which was determined using the elbow method for the largest and most diverse region, and by the sizes and standard deviation of the tiles. Choosing a higher k_r will lead to more clusters. Despite the high ratios, the information loss at this stage is not critical for the solar and wind potential maps, because the full-load hours do not variate a



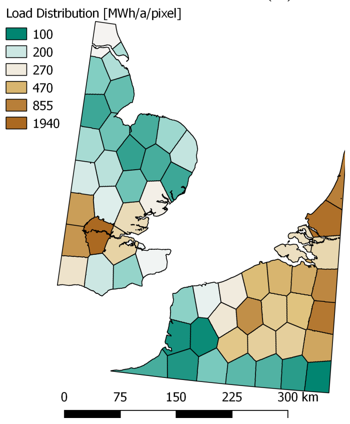
(a) The initial load raster.



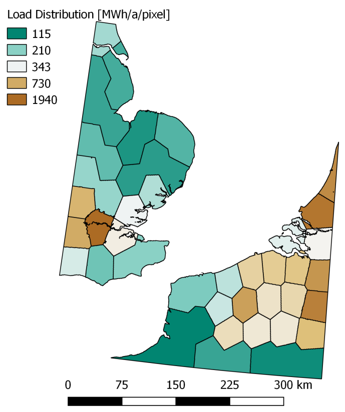
(b) Division into smaller parts.



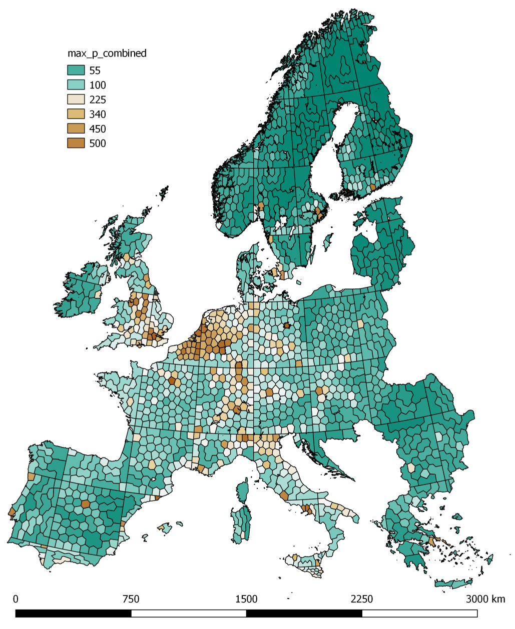
(c) Exemplary raster part before clustering.



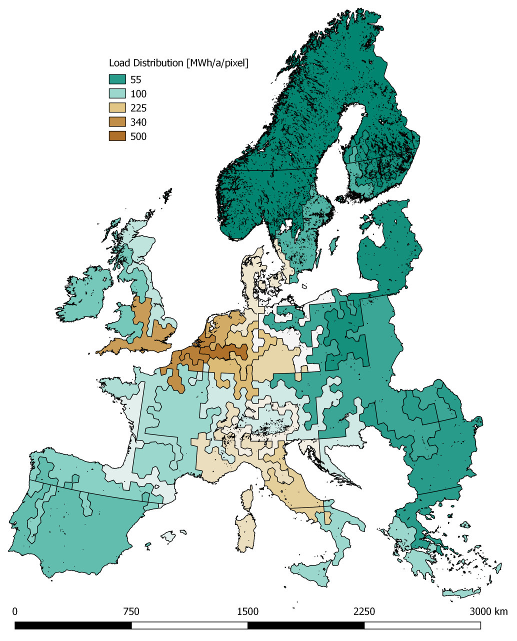
(d) Result of k-means clustering.



(e) Result of max-p clustering.

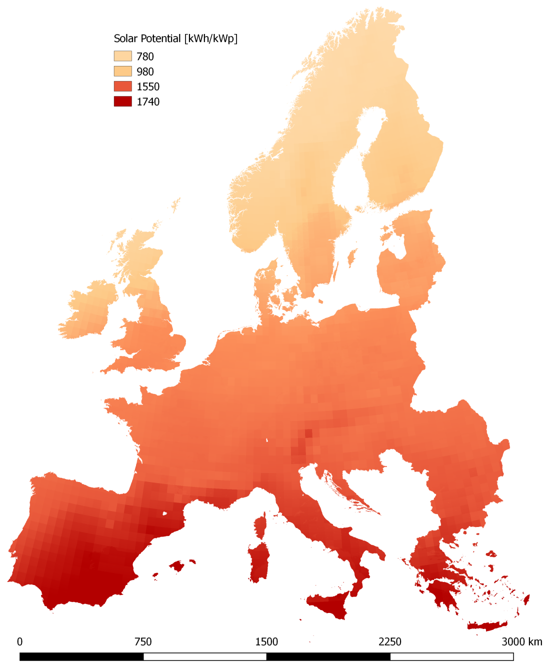


(f) Recomposition.

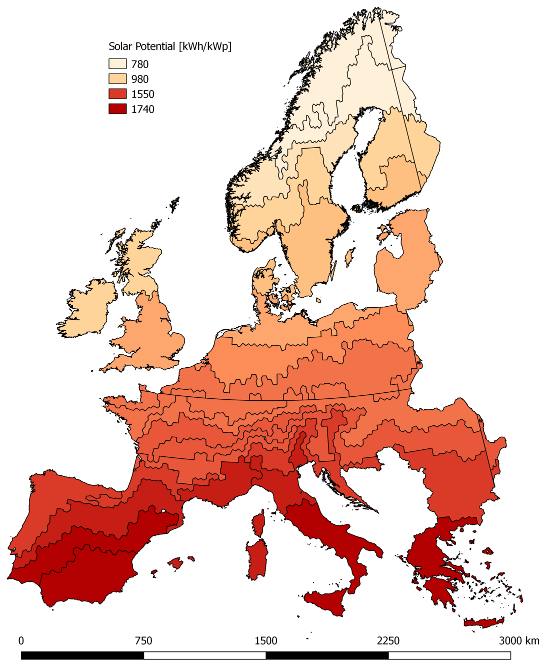


(g) Result of 2nd max-p clustering.

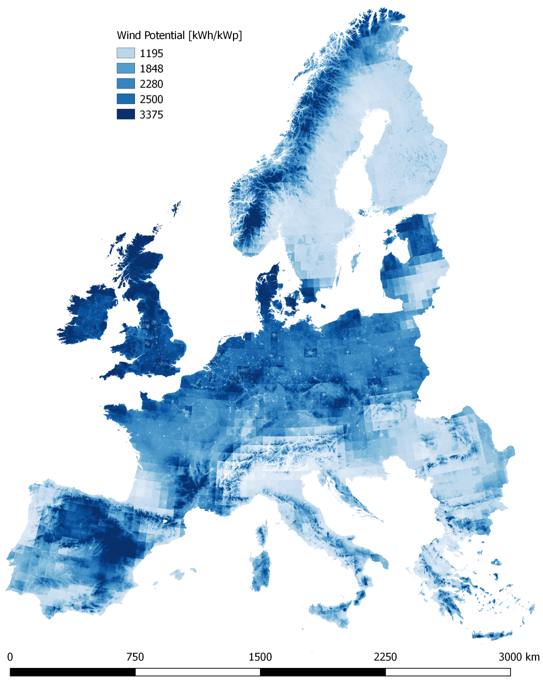
Figure 5.2: Clustering methodology steps as applied to the load raster map.



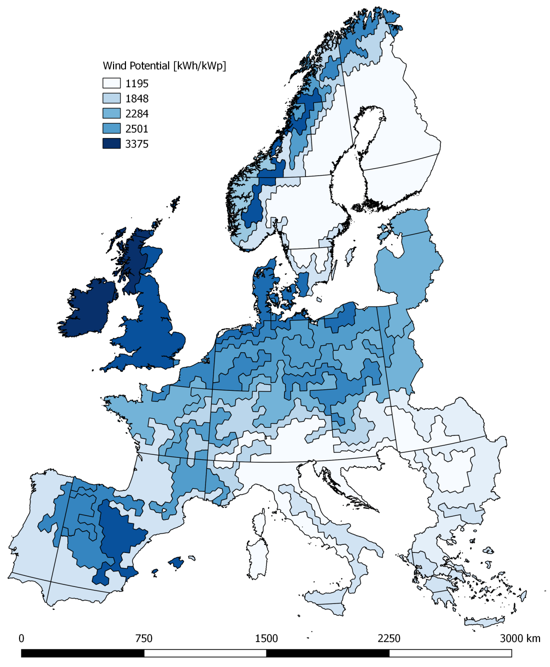
(a) The initial solar potential raster.



(b) Clusters based on solar potentials.



(c) The initial wind potential raster.



(d) Clusters based on wind potentials.

Figure 5.3: Input rasters and output regions produced by the clustering process.

lot within a small radius, as reflected in the low coefficients of variation (below 0.04 for the solar potential map). However, for the load map, small urban settlements with high load densities are dissolved with their surrounding areas, and the coefficients of variation may exceed 20 in northern Scandinavia. Increasing the number of *k-means* clusters (above the optimum number derived from the elbow method) might preserve the contrast between urban and rural areas until the next stage, but it would lead to longer computation times for the *max-p* algorithm running on all tiles.

Table 5.1: Evolution of the number of valid data points over the clustering process.

Map	Input	*k-means++*	*max-p* 1	*max-p* 2
Solar	$\sim 36.7 \cdot 10^6$	3490	1853	28
Wind	$\sim 36.7 \cdot 10^6$	2792	1780	28
Load	$\sim 36.7 \cdot 10^6$	2347	1340	28

Currently, the first run of the *max-p* algorithm has the lowest data compression ratios. It is possible to alter the threshold values by setting different constraints for the regression that determines the parameters A, B, and C. Higher thresholds lead to less clusters for each tile, which can speed up the second run of *max-p* for the whole map. However, the quality of the final maps will be lower in this case. Hence, we chose the constraints so that we obtain as many clusters after the first run of the *max-p* algorithm as we can process in a reasonable amount of time (below 8 hours).

5.3.3 Homogeneity and contrast

By design, the final solar and wind clusters should be overall more homogeneous with respect to the solar and wind potentials than individual countries. This is for instance visible in Figure 5.4, which depicts the PV full-load hours of a sample of data points for each region (blue lines, sorted by the median of the regions) and for all Europe (black line in the background). The median values for each region, whether a country (top) or a solar cluster (bottom), are shown in red. Figure 5.4 shows that the new clusters have smaller value ranges, are of similar sizes, and their medians lie almost always on the black curve of the sorted FLH values in Europe. Hence, the clustering method splits Europe in homogeneous regions of equal sizes, that can be represented by their median values without sacrificing too much accuracy. This is not the case for the map using country borders, where countries with very good locations for solar projects are misrepresented by their low median values.

In all three maps, the coefficients of variation are usually smaller than for the map of countries (with respect to the data type chosen for the clustering), both for the extremes and on average. It varies between 1.14 and 3.56 for the load clusters (countries: 1.23 – 6.05), between 0.01 and 0.09 for the solar clusters (countries: 0.01 – 0.15), and between 0.11 and 0.50 for the wind clusters (countries: 0.12 – 0.59). The improved homogeneity may have a positive impact on the robustness of the modeling of energy systems with high shares of solar and wind power supply. In fact, model regions are usually represented by one or a few points with particular FLH values, for which time series are generated. Therefore, it is crucial that these points (best, upper 10%, median, etc.) summarize the quality of the region without distortion.

In addition to homogeneity, higher contrast between the regions may be desired for some applications. By clustering the load map so that the regions satisfy a minimum total load, we created clusters of high load densities in Central Europe and low load density on the periphery. The load densities are plotted against the area of the regions in the case of countries (crosses) and load clusters (filled circles) in Figure 5.5. Compared to countries, the load clusters have more distinguishable regions on the extremes (large area and low density, or small area with high

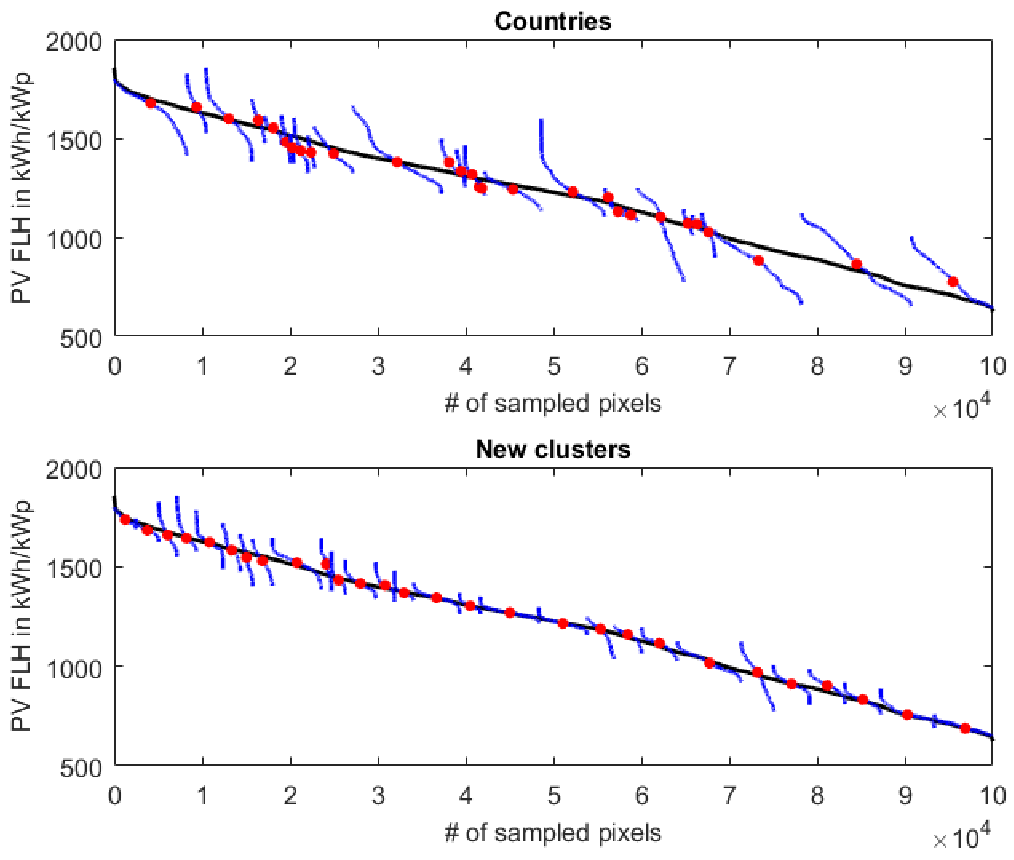


Figure 5.4: Solar FLH values within regions (blue lines), sorted in descending order by their median value (red circles), plotted against the sorted values for Europe (black line in the background). The top figure corresponds to the countries, the bottom one to the solar clusters.

density), which can be useful for studies that investigate the energy system of high-demand regions (e.g. cities) surrounded by low-demand regions (e.g. countryside).

Whether these topological differences have a measurable impact on energy system optimizations can be determined by running an experiment using energy system models. The procedure of the experiment used in this study is explained in the next section.

5.4 Energy system optimization

In order to quantify the impact of the choice of regions on energy system optimizations, we build models using the regions of Figures 5.2g, 5.3b and 5.3d. We first describe the modeling framework, then provide information about the data and the assumptions used in this experiment.

5.4.1 Expansion planning models with *urbs*

We use the open-source modeling framework *urbs* to generate the models for our analysis. The created models co-optimize capacity expansion as well as hourly dispatch of generation, transmission, and storage from a social planner perspective. The optimization goal is to minimize the costs of expanding and operating the energy system including the annualized capital costs, the

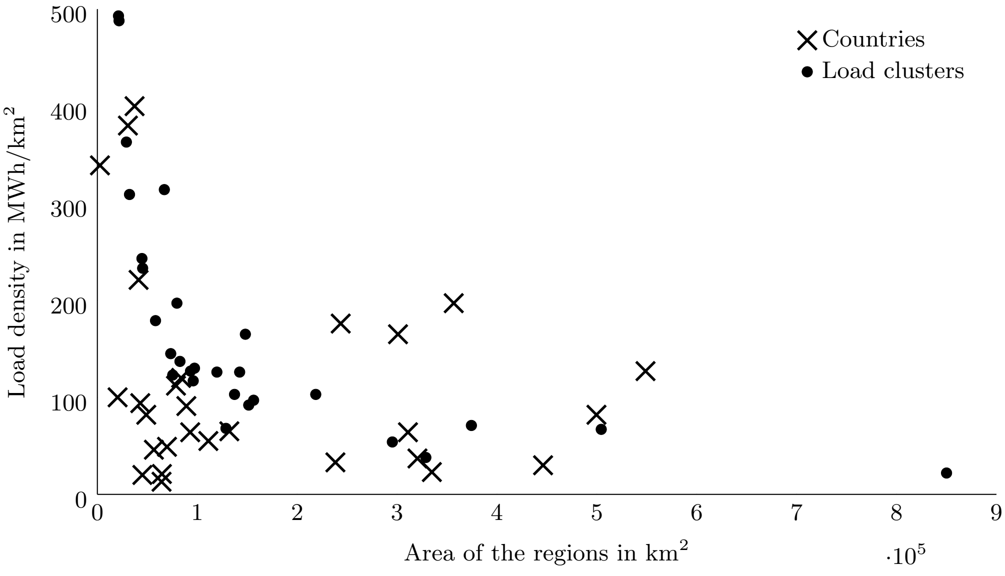


Figure 5.5: The load density in relation to the area of each region in the case of countries (crosses) and load clusters (filled circles).

fuel costs, as well as other fix and variable operational costs. Major inputs are the hourly time series for the load and the capacity factors of renewable energy sources. Other important input data include the existing infrastructure (grid, power plants, storage) which can eventually be built in the models. Techno-economic parameters cover costs for investments, maintenance, fuels, and emissions. Finally, restrictions can be set such as maximal capacities for grid/generation expansion or a limit on the CO_2 emissions.

Each model solves a linear optimization problem that is written in Pyomo using gurobi. Major outputs include the installed capacities (generation, grid, storage) and the hourly operation of the system. The models also provide the emissions, the costs, and the marginal costs at each region. The source code for *urbs* and an extensive description can be found on GitHub (Dorfner et al., 2018).

5.4.2 Data and assumptions

The main assumptions and data sources used to conduct this analysis are described below. The data pre-processing is automated in order to ensure a uniform model generation process.

Geographic coverage The analysis covers the 28 countries of the European Union, excluding Malta and Cyprus and adding Norway and Switzerland.

Representation of time We use the second weeks of January, April, July, and October to represent the full year of 2015. The time resolution is 1 hour.

Model regions We use four models based on: country borders (later referred to as *Countries*), regions with homogeneous wind potential in terms of full-load hours (*Wind*), regions with homogeneous solar photovoltaic potential (*Solar*), and regions with similar total electricity demand (*Load*).

Load time series Hourly time series for each country (ENTSO-E, 2015) are disaggregated into sectoral load time series based on typical sectoral load profiles (BDEW, 2017). Sectors are distributed geographically based on land use types (Broxton et al., 2014). Load is aggregated again for the new model regions. The same load time series are used in 2015 and 2050.

Renewable time series Wind and solar hourly capacity factor time series are generated by combining MERRA-2 radiation, temperature and wind speed data (Gelaro et al., 2017) with maps of land use, elevation, and protected areas. Among the suitable locations in each region, we pick the time series of the pixel with the median full-load hours for onshore wind and solar PV, and at the top 10% for offshore. We use a uniform correction factor lower than 1 for each technology in order to match the wind and solar generation of Europe in 2015 approximately. Identical time series were used in 2050. A new version of the code for generating the time series and the input rasters for wind and solar is available open source (Siala et al., 2019).

Commodity prices We assume the same prices in 2015 and 2050. The prices are constant over the year and respect the merit order of power plants observed in 2015. For more information, see the inputs files for the *urbs* models (Siala and Mahfouz, 2018).

Conventional power plants Power plants in operation in 2015 (Hofmann et al., 2018) were allocated to the regions based on their coordinates, then aggregated based on their types. In 2050, none are still existing. New capacities for nuclear are not allowed to exceed the levels of 2015. Missing power plant characteristics were filled with own assumptions.

Renewable power plants Installed capacities in each country in 2015 were collected from IRENA (Lavagne d'Ortigue et al., 2016) then distributed geographically based on technical potential maps (wind and solar), land use types (others), and a randomness factor. There are no pre-existing capacities in 2050. New capacities for hydro and biomass are not allowed to exceed 2015 levels.

Transmission lines Transmission lines are extracted from GridKit (Wiegmans, 2016). Based on their lengths and voltage levels, a transmission capacity is assigned to them. Only connections between different regions are considered.

Economic parameters Investment, fix and variable costs for most power plant types are derived from ETRI (Lacal Arantegui et al., 2014).

CO_2 emissions There are no limits for 2015. For 2050, we assume a 95% reduction compared to 2015. No CO_2 certificate price are used.

5.5 Results of the optimization

In this section, we discuss the results of the energy system optimization for 2015 and for 2050 using different regions within Europe.

We first compare the results of all models (*Countries*, *Wind*, *Solar*, *Load*) to ENTSO-E statistics for the year 2015. Despite using only four weeks instead of a full year, the models manage to match the energy mix of Europe with minor discrepancies, as shown in Figure 6.2a. The models overestimate the share of nuclear power, but the error is compensated by an underestimation of coal, gas, and other mixed fuels. Almost no energy is curtailed (less than 0.1% in all models).

Comparing the model *Countries* to the three others, we observe negligible differences in the solar PV generation (relative error between 1% and 5%). However, the wind generation is overestimated in all three models, with a relative error ranging from 5% for *Solar* to 10% for *Wind*. The reason for this lies in the choice of the representative time slices and in the time series for renewable generation. First, although the model *Countries* was calibrated to match the energy mix of ENTSO-E using a full-year optimization[1], this calibration is lost when running a four-week optimization. The time series happen to be representative for the solar generation in that time frame, but they are not for wind generation. Second, as mentioned in section 5.4.2, the renewable generation time series were determined for a pixel at the median locations for onshore wind and solar PV, and at the top 10% for offshore wind. Since the shape of the regions are not the same in the models, the locations of the medians and the top 10% are also different, which leads to different time series.

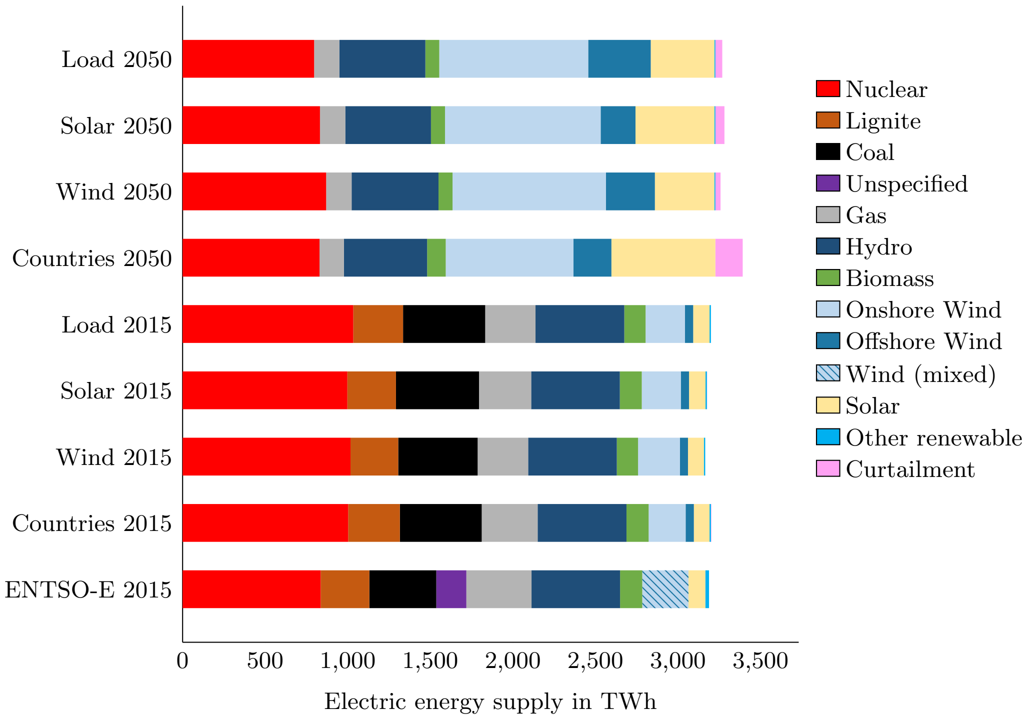


Figure 5.6: Electric energy supply in Europe in TWh according to the *urbs* model and to ENTSO-E statistics of 2015.

The discrepancies between the models widen for the year 2050. Due to the stringent CO_2 emissions constraint, lignite and coal power plants disappear from the power mixes. The only CO_2 emitting technology which is still allowed is gas, due to its lower specific emissions. The combined share of hydro power and biomass remains almost constant in all four cases. On the other hand, the shares of solar PV, onshore and offshore wind, nuclear, and the amount of curtailment vary considerably between the models.

In the model *Countries*, the ratio of solar to wind is 1:1.6, the lowest in all four models. It is also coupled with a higher amount of curtailment (166 TWh compared to 31 TWh for *Wind*, the lowest value). All in all, a positive correlation exists between the amount of curtailment and the ratio of solar to wind. Also, the higher the capacity of solar PV, the bigger is the installed

[1] A full-year optimization was conducted using the model *Countries* for the purpose of calibration, but its results are not discussed in this paper, because the focus lies in the relative differences between the models.

capacity of battery storage, as shown in Figure 5.7. One possible explanation for this could be the low wind FLH values of countries in Southern Europe, which have high FLH of solar PV. Yet this explanation is not sufficient, because an even stronger effect would have been visible in the model *Solar*. However, we observe that the cost-optimal solution for the model regions in *Solar* is to build less PV and more onshore wind than in *Countries*. This counter-intuitive result could be explained with the shape of the regions: the stretched horizontal bands of the solar cluster have more transmission line connections to their neighbors than the compact model regions in *Countries*. With less transmission bottlenecks, the model *Solar* can integrate wind and solar generation better in the North-South direction.

As expected, the model *Solar* has a higher share of solar PV than *Wind*, which has on the other hand a higher share of onshore and offshore wind, combined. This is most probably due to the existence of very favorable locations with high full-load hours hours of solar PV (in the case of *Solar*) and onshore wind (in the case of *Wind*). The model *Load* is characterized by the highest share of offshore wind generation (379 TWh). Here again, the shape of the regions and their geography provide a possible explanation. Many of the regions with high load density lie in Northern Europe, close to the North Sea, an area with very good conditions for offshore wind power plants due to its shallowness and high wind speeds. Hence, offshore wind power plants located in that area operate with high capacity factors and at competitive costs to cover a high proportion of the electricity demand of the model regions of *Load*.

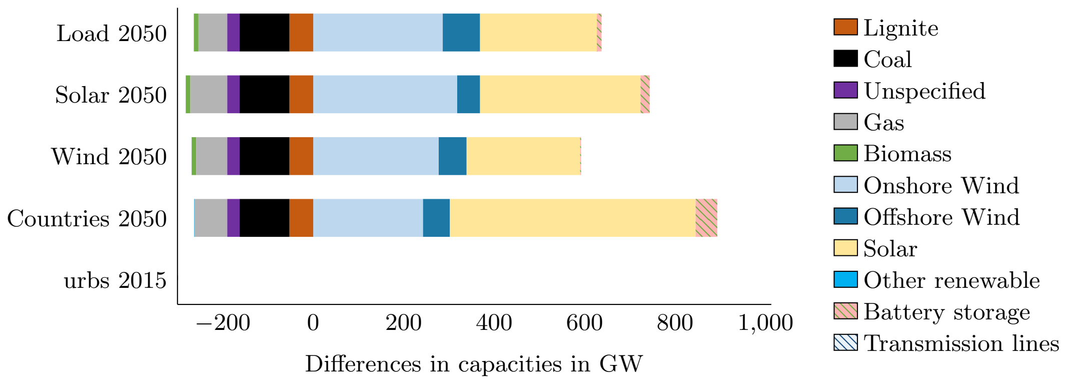


Figure 5.7: Differences in capacities of power plants, storage units, and transmission lines in GW compared to the capacities in 2015.

Last but not least, Figure 5.7 shows no investment in transmission lines and a little investment in battery storage, when compared to the large investments in solar PV and wind. This is due to the way the transmission grid is simplified: There are no transmission constraints within each region, and interconnections between model regions limit the amount of electricity that can be traded between them. With only 28 model regions, none of which are designed to reflect transmission bottlenecks, the limits for electricity transport are usually high and renewable power generation seems to be easy to integrate with no grid expansion nor large amounts of storage.

The discrepancy between the models is not limited to the aggregated capacities and energy production values, but affects the geographic distribution of the power plants as well. Although the *urbs* models do not deliver exact locations for the new power plants, it is possible to draw approximate geographic distribution maps based on the potential map and a randomness factor. In Figure 5.8 we plot possible distributions of onshore wind power plants based on the results of the four models, as an example.

The darker the color in Figure 5.8, the more agreement there is between the models that wind power plants should be built in the area. This seems to be the case for the coastal areas in Northern Europe and Norway, the United Kingdom, Estonia and Latvia. There is also a consensus to avoid

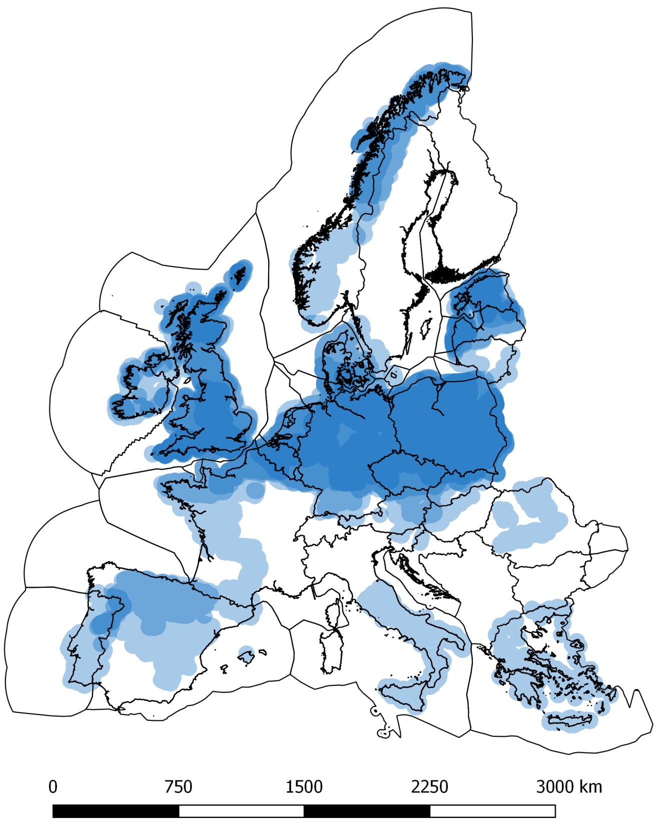


Figure 5.8: Possible geographic distribution of new wind capacities in 2050. Each shade of blue corresponds to the distribution of one of the models, so that dark blues areas are those identified by all models as most likely project locations.

Northern Italy, Southern Spain, Bulgaria and Central Sweden. Despite the differences between the models, these results are considered robust.

5.6 Conclusion and Discussion

In this paper, we argue that creating energy system models with different shapes for the regions has several advantages. The focus in the first part was on the clustering methodology that we used to obtain regions with homogeneous characteristics out of high resolution data. The method is scalable and we were able to apply it on data sets with roughly 10^8 data points to obtain 28 regions. We explained how the number of clusters at the end of each stage of the algorithm can be varied, and which impacts this would have on the quality of the results and on the computation time. For the sake of clarity, the clustering was conducted using one single characteristic at a time (either similar total electricity demand, or wind potential, or solar potential). However, it is possible to use two or more parameters to obtain regions with a combination of characteristics. This could be done in a future study.

The regions created through clustering have inherent properties that are crucial for certain analyses. For instance, the model based on load density clusters is suitable for studying the interplay of cities and countryside, particularly in the power sector. Using regions with different renewable potentials ensures their homogeneity, which is usually implicitly assumed in energy system models that rely on one or a few time series per region. We showed that this assumption does not hold for models using countries as model region, and that clusters based on wind or solar full-load hours have lower coefficients of variation. Homogeneous regions are adequately represented through single time series, which leads to more robust results in energy system optimizations.

Using countries as model regions is appropriate in many situations, such as the analysis of policies on a national level. However, energy system modelers should be able to assess the magnitude of the errors caused by the choice of the regions. This study provided an example of an electricity system optimization for Europe for 2015 and 2050. Whereas the errors in 2015 were minimal, due to the limited share of renewable power supply, they were higher for 2050, causing discrepancies of 5-10% in the energy mix. This example showed some advantages of using other model regions for a complementary analysis. For modelers, the variation of the regions works as a sensitivity analysis. It puts the robustness of the model results to the test and helps visualize their actual impacts on the geography of the energy system. It can also reveal the weaknesses of the model assumptions. In our analysis, this was particularly the case of the misrepresentativity of some renewable time series and of the lack of transmission line congestion within regions. The clustering algorithm can be used to solve the former problem, by creating homogeneous areas (either independently or within political or administrative divisions), for which representative time series can be generated. As of the modeling of transmission lines, all our cluster models and the one based on national borders neglected bottlenecks within the regions, which is not a valid assumption in many instances. We recommend clustering transmission networks based on line contingencies, yet we were not able to achieve this due to the lack of topologically valid open grid data for Europe.

For policy-makers, trying various shapes of regions for the same study can be a powerful argument in a time characterized by a lack of acceptance for infrastructure projects, such as transmission lines and onshore wind power plants. It shows willingness to search for alternatives, and overcomes certain modeling weaknesses (e.g. abstraction, unsubstantiated assumptions, etc.) through a visually appealing sensitivity analysis.

Acknowledgments

The authors acknowledge the financial support by the Federal Ministry for Economic Affairs and Energy of Germany (BMWi) in the project “4NEMO - Research Network for the Development of New Methods in Energy System Modeling; research project: 0324008A”.

5.A Pseudocodes

Pseudocode 1: Main Code

```
$D$: High resolution data raster
$m$: Number of tiles in the vertical dimension
$n$: Number of tiles in the horizontal dimension
$F$: Output shapefile
$\alpha$: Range of clustering values for elbow method
$L$: Limit of accepted distance in Call-k-means++

$A$= Split ($D$, $m$, $n$), set of split areas
$k_r$ $n_{max}$ $n_{min}$ $\sigma_{max}$ $\sigma_{min}$ = Elbow Method ($A$, $\alpha$, $L$)
for every $a_i \in A$ do
    $\varphi_i$ = Call-k-means++ ($a_i$, $k_r$, $\sigma_{max}$, $n_{max}$, $L$)
$E$ = Call-max-p$_1$ ($n_{min}$, $n_{max}$, $\sigma_{min}$, $\sigma_{max}$, $\varphi$)
$F$ = Call-max-p$_2$ (E, $thr_2$, $N_2$)
```

Pseudocode 2: Split

```
Inputs: $D$, $m$, $n$
Read number of rows of input raster $r_D$ and the number of columns $c_D$
$r = r_D/m$, number of rows in each tile
$c = c_D/n$, number of columns in each tile
$A$ is empty
for $v = 1 \cdots m$ do
    for $h = 1 \cdots n$ do
        Cut the data set $D$ between $c \cdot (h-1)$ and $c \cdot h$ in the horizontal
          dimension, and $r \cdot (v-1)$ and $r \cdot v$ in the vertical dimension
        Save sub-data set in $a_i$
        Add $a_i$ to $A$
Outputs: $A = a_1, a_2, \cdots, a_i, \cdots, a_n$
```

Pseudocode 3: Elbow Method

```
Inputs: $A$, $\alpha$, $L$

$e_h$: Inertia of k-means++ clusters.

for every $a_i \in A$ do
    Calculate standard deviation $\sigma_i$ and number of non-null points $n_i$ of every
     area $a_i$
    Calculate the product $X_i$ of $\sigma_i$ and $n_i$ of every area $a_i$
Pick area $a_r$ with maximum value of $X_i$
Save $n_{max}$, the maximum value of number of non-null points in all areas
Save $n_{min}$, the maximum value of number of non-null points in all areas
Save $\sigma_{max}$, the maximum value of standard deviation in all areas
Save $\sigma_{min}$, the maximum value of standard deviation in all areas

$G$ is an empty plot
for every element $h$ of $\alpha$ do
    $\sim, e_h =$ k-means++($a_r, h, L$), Euclidian square distance from k-means++
     output
    Plot the point with the coordinates $(h, e_h)$ in graph $G$
Apply the elbow method to $G$ as described in Thorndike (1953)
Choose the point $P_\alpha$ at which the gain of adding more clusters drastically
 decreases
$k_r$ = x-coordinate of $P_\alpha$, number of clusters of reference area $a_r$

Outputs: $k_r$ $n_{max}$, $n_{min}$, $\sigma_{max}$, $\sigma_{min}$
```

Pseudocode 4: Call-k-means++

```
Inputs: $a_i$, $k_r$, $\sigma_{max}$, $n_{max}$, $L$

$k_i = k_r \cdot (0.7\frac{n_i}{n_{max}} + 0.3\frac{\sigma_i}{\sigma_{max}})$, Number of clusters for area $a_i$
$\varphi_i, \sim=$ k-means++($k_i, a_i, L, \sigma_{max}, n_{max}$) as described in Arthur and Vassilvitskii
 (2007), clustered data set for area $a_i$

Output: $\varphi_i$
```

Pseudocode 5: Call-max-p$_1$

```
Inputs: $n_{min}$, $n_{max}$, $\sigma_{min}$, $\sigma_{max}$, $\varphi$

$A, B, C$: Regression constants used in calculating $thr_1$

$Islands$: all $\varphi_i$ which do not have common borders with other areas
$thr_1$: Threshold value for max-p$_1$
$N_1$: Neighbor matrix for max-p$_1$
$d$: one data feature in $a_i$ picked according to the clustering analysis priority
  (Solar, Wind, etc.)
Create $N_1$ matrix

if $islands \neq 0$ then
    Assign every island to the nearest neighboring area

Calculate the regression equation constants $A, B, C$.
$A \cdot e^{-B(\frac{n_{min}}{n_{max}}+C\frac{\sigma_{min}}{\sigma_{max}})} = 0.5$
$A \cdot e^{-B(\frac{n_{min}}{n_{max}}+C\frac{\sigma_{max}}{\sigma_{max}})} = 0.1$
$A \cdot e^{-B(\frac{n_{max}}{n_{max}}+C\frac{\sigma_{max}}{\sigma_{max}})} = 0.01$

Calculate the threshold value $thr_1$ for area $a_i$.
$thr_1 = A \cdot e^{-B(\frac{n_i}{n_{max}}+C\frac{\sigma_i}{\sigma_{max}})}$
for $h = 1 \cdots n$ do
    $e_i$: max-p$_1$ output as described in Duque et al. (2012)
    $e_i$ = max-p$_1$($\varphi_i, thr_1, N_1$)
Merge $e_i$ zones to get $E$
$E$: Map of all areas A after first step of max-p

Output: $E = e_1, e_2, \cdots e_i, \cdots, e_n$
```

Pseudocode 6: Call-max-p$_2$

```
Inputs: $E$

$Islands$: clusters in $E$ which do not have common borders with other clusters
$d$: one data feature picked from $E$ according to the clustering analysis priority
  (Solar, Wind, etc.)
$thr_2$: Threshold value for max-p$_2$
$N_2$: Neighbor matrix for max-p$_2$

Create $N_2$ matrix
**if** $islands \neq 0$ **then**
    Assign every island to the nearest neighboring area
$thr_2 = \frac{\sum_{d \in E} d}{40}$
$F = \texttt{max-p}_2(E, thr_2, N_2)$

Output: $F$
```

Chapter 6

Combining spatial resolutions: Application of model coupling

The third publication describes a model coupling exercise involving an optimization model for expansion planning and economic dispatch for Europe and a global Input-Output analysis. The models have different geographic scopes and different purposes. Through the model coupling, the implications of the decarbonization of the European power system on the economy and the environment in and outside of Europe are investigated in a high technological detail by combining the capabilities of the two models. Some of the lessons learned from the combination of spatial resolutions are listed in Section 3.4, namely how to clearly identify the shared scope, how to avoid syntax and semantic errors, and how to setup a data sharing interface.

As a first author of the publication, I was responsible for the structure and the submission process. I contributed to the method section, focusing on the expansion planning with the modeling framework *urbs* and on the linking to the multi-regional input-output analysis. I was also the main author of the results section, and I created the plots. I run the *urbs* optimization and established the data sharing interface with my co-author, Cristina de la Rúa, who was responsible for the input-output analysis. She contributed mainly to the introduction, to the method section (describing the multi-regional input-output analysis), and to the results and conclusions. Yolanda Lechón contributed mainly to the introduction and the literature review, and provided a critical review to other sections of the paper. Thomas Hamacher participated in concretizing the idea of the paper and provided a critical review.

Title: Towards a Sustainable European Energy System: Linking Optimization Models with Multi-Regional Input-Output Analysis

Authors: Kais Siala, Cristina de la Rúa, Yolanda Lechón, Thomas Hamacher

Journal: Energy Strategy Reviews (Elsevier)

Publication date: August 2019

Permission to use: Elsevier publishing guidelines allow authors to “include their articles in full or in part in a thesis or dissertation for non-commercial purposes”.

Abstract

During the Paris COP21, the European Union set ambitious goals regarding renewable energy and greenhouse gas (GHG) emissions. This study uses two models to analyze the implications of these goals on the European energy system and on the rest of the world, based on several scenarios for the mid and long term. First, using a linear programming optimization model for capacity expansion and unit commitment, we obtain the optimal design of the European power system in 2030 and 2050. This design is then used as an input for a Multi-Regional Input-Output Analysis, in order to analyze the environmental and socio-economic effects derived from the new energy system. The results will focus on the impact on Germany, but also on a European and global scale. Linking these tools, it will be possible to understand under which conditions the transition will succeed and its consequences on the global supply chain in terms of GHG emissions, cumulative energy demand, value added, and job creation.

Keywords: Capacity expansion optimization; input-output analysis; socio-economic impacts; environmental impacts

6.1 Introduction

The European Union has set a long-term target of reducing greenhouse gas emissions (GHG) in the energy sector by 80-95% by 2050 compared to 1990 levels, in order to make the European economy climate-friendly (European Commission, 2012). Although all sectors should contribute to the decarbonisation of the economy, the electricity sector deserves special attention because of its biggest potential for reducing direct GHG emissions through the implementation of efficiency measures and the expansion of clean technologies[1]. In this sense, the use of electricity system models allows for a better understanding of how the system might evolve in the future, so that the demand is satisfied at the minimum cost and under specific constraints. These constraints can be technical, such as resource availability, but also political, such as nuclear phase-out or renewable energy quotas.

The expansion of clean technologies will contribute to the reduction of direct GHG emissions but will require new investments, and therefore, many sectors of the economy will respond to the expansion by extending their production. The increase in the production of goods and services derived from the future new configuration will mean additional energy use and GHG emissions, caused by other non-energy sectors. Considering that the final objective of the European Union is to achieve a zero emissions economy, it is highly important to account for the indirect effects associated with the new energy scenarios. Besides the potential environmental benefits associated to a new energy system, other benefits could be expected. The stimulation of the economy could bring an increase in the value added as well as the creation of new employment, both directly and indirectly. All these effects can be examined through an extended Input-Output Analysis.

In this work we combine two quantitative tools in order to better understand the future developments of the electricity system in Europe as a whole, and in Germany in particular. We link the outputs of an expansion optimization model to an extended Multiregional Input-Output model in order to assess the environmental and socioeconomic impacts of the electricity sector. Our study is structured in the following way: In the next section, we introduce the models and explain the workflow of the model coupling. In section 6.3, the results of the models are described. In the final section, we discuss the outcomes and their possible policy implications.

[1] According to the scenarios of the Energy Roadmap 2050, the electricity sector should be able to cut emissions by 57–65% in 2030 and 96–99% in 2050 compared to 1990 levels (European Commission, 2012, p. 7).

6.2 Method

This section briefly describes the workflow of the paper with references to the relevant sections for further details.

In the first part of the analysis, we use an expansion planning optimization tool to model the European electricity system in 2015, 2030, and 2050 (blue, left side in Figure 6.1). A description of the model is provided in section 6.2.1. The results of the optimization, including the calibration and the sensitivity analysis, are given in sections 6.3.1 and 6.3.2.

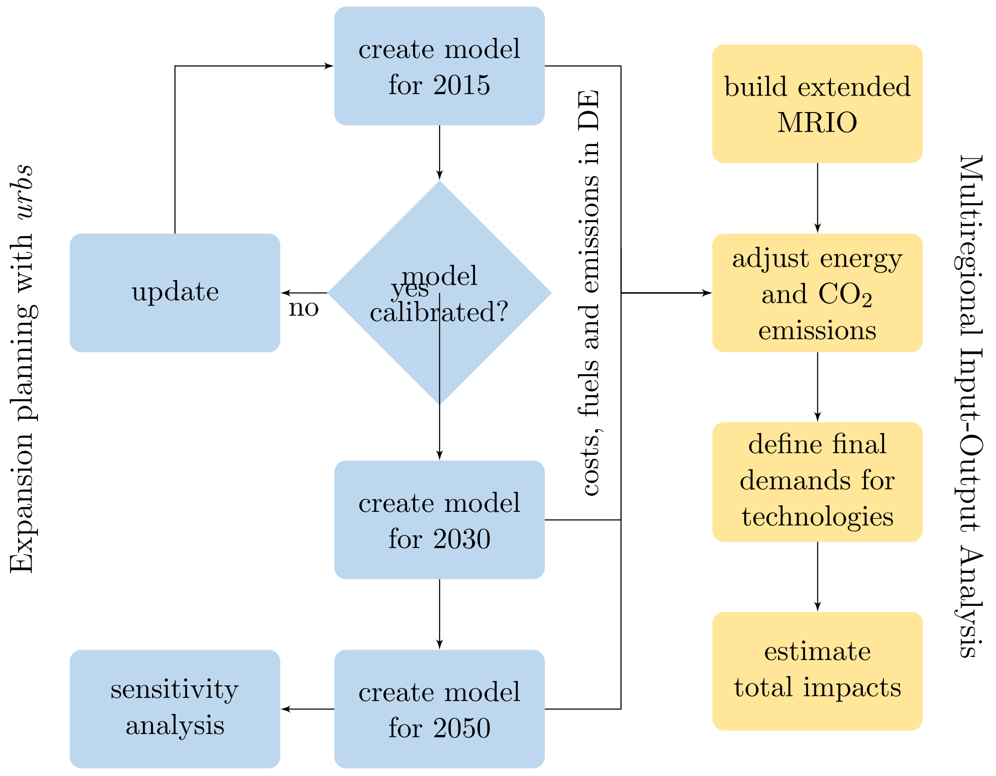

Expansion planning with *urbs*
create model for 2015
model calibrated?
no
yes
update
create model for 2030
create model for 2050
sensitivity analysis
costs, fuels and emissions in DE
build extended MRIO
adjust energy and CO_2 emissions
define final demands for technologies
estimate total impacts
Multiregional Input-Output Analysis

Figure 6.1: Workflow of the study.

The different costs, fuels quantities and direct CO_2 emission are then used as an input for the second part, the Multiregional Input-Output Analysis (MRIO). Section 6.2.2 provides a theoretical background in Input-Output Analysis. The core of this study, the linking of the optimization model with MRIO analysis, is explained in section 6.2.3. The results of the analysis for the case of Germany are presented in section 6.3.3.

6.2.1 Expansion planning with *urbs*

We use the open-source model framework *urbs* to generate the models for our analysis. The created models co-optimize capacity expansion, hourly dispatch of generation, transmission, and storage within Europe, using countries as model regions. The optimization goal is to minimize the costs of expanding and operating the European energy system. Major inputs are the hourly time series for the load (ENTSO-E, 2015) and the capacity factors of renewable energy sources (Gelaro et al., 2017), the existing infrastructure (grid (Wiegmans, 2016), power plants (Hofmann et al., 2018; Lavagne d'Ortigue et al., 2016), storage (Hofmann et al., 2018)), and techno-economic parameters such as investment and maintenance costs (Lacal Arantegui et al., 2014), fuels costs, and specific emissions. Each model solves a linear optimization problem that is written in Python/Pyomo using the gurobi solver. Major outputs include the installed capacities (generation, grid, storage) and the hourly operation of the system. The models also provide the direct emissions, the total costs, and the marginal costs in each region. The source code for *urbs* and an extensive description can be found on GitHub (Dorfner et al., 2018).

In this analysis, we do not allow for the expansion of transmission lines. New biomass, hydro power plants and pumped hydroelectric storage units can be built as long as their total installed capacity does not exceed their capacity in 2015. Retrofitting applies also for nuclear power plants, with the exception of countries that do not intend to operate nuclear power plants in the future[2]. Most importantly, we set restrictions on the CO_2 emissions, which should decrease by 50% in 2030 and by 95% in 2050 compared to their level in 2015. The use of 2015 instead of 1990 as a reference year is justified by the general consensus that the electricity sector could play a key role in the decarbonization of the energy sector, hence it should be able to achieve more ambitious targets than the energy sector as a whole (European Commission, 2012, p. 7).

6.2.2 Multiregional Input-Output Analysis

The second tool presented in this paper is based on the Input-Output Analysis (IOA), developed by Leontief (1951). It measures how the different economic activity sectors respond to a change in the final demand of goods and services within a national economy. The core of the IOA are the Input-Output Tables, which describe the trading relationships among the different economic sectors and with the final demand users in monetary units. Based on the National Accounts, these tables have two main components: the *inter-industry flows*, also known as *transaction matrix*, and the *final demand*.

The transaction matrix describes the production process by industry or activity in columns, and the use of goods and services in rows. In the transaction matrix, the goods and services being interchanged correspond to intermediate goods, which will be further processed by other activity sectors (Miller and Blair, 2009). Thus, the units included in the final demand component refer only to goods and services already processed.

The Input-Output Tables can be represented using the technical coefficients, which describe the normalized cost requirements by sector and are denoted by:

$$a_{ij} = z_{ij}/x_j \tag{6.1}$$

where a_{ij} is the technical coefficient, z_{ij} is the amount of goods and services from sector i consumed by sector j, and x_j is the total output or production of sector j.

All the productions by sector x_{ij} can be expressed as a whole in a matrix equation:

$$x = (I - A)^{-1}y \tag{6.2}$$

where $(I - A)^{-1}$ is the Leontief inverse matrix, or the multiplier matrix, that expresses the direct and indirect requirements by sector per unit of final demand. Using this analysis, it is possible to estimate the potential economic impacts of new infrastructure projects or policies, in terms of total gross production and value added.

The IOA was initially applied in national or regional economies. However, the current global situation cannot be analyzed from a domestic perspective only, because the supply chains have

[2] As of 2018, these countries include Austria, Belgium, and Germany. Other countries did not have nuclear power plants in 2015 and are thus *de facto* excluded from having new ones. This assumption might be valid for most cases, but is in contradiction with the policy of some countries such as Poland. We do not think that the exact modeling of all the national policies is necessary for this study, since we focus mostly on Germany.

been fragmented in the last decades across countries, modifying the domestic economies and the international trade structure. These changes have affected not only the economic structure, but also other aspects such as pollution or job creation, due to the differences among the countries in terms of environmental legislation, industrial automation and employment structures. The Multiregional Input-Output Analysis gets beyond this limitation by including the interregional and intraregional transactions (Wiedmann et al., 2011).

Besides the economic impacts, it is possible to estimate other effects by expanding the Input-Output tables with other relevant sectorial information such as environmental aspect, which leads to an Environmentally Extended Input-Output Analysis (Chen et al., 2010; Davis and Caldeira, 2010; Su et al., 2010). To do that, an additional matrix or vector shall be included in equation 6.2, which defines how much energy for instance is used by each activity sector to produce one unit of its output (Kitzes, 2013; Tukker et al., 2009).

6.2.3 Linking the optimization model with MRIO

In this study, we have used the World Input-Output Database (WIOD) to build the multiregional model. The database was developed under the $7th$ Framework Program, and it contains Input-Output Tables for 41 regions of the world (Timmer et al., 2015). Additionally, the project also produced satellite accounts containing environmental and social data, which allow us to develop an extended multiregional model.

The use of economic models relies on many assumptions that simplify the complex reality. Among others, input-output models describe the economy in a particular year, in which the relation between inputs, in terms of required goods and services, and outputs of sectors remains constant. This assumption becomes stronger in the case of the environmental performance of each sector. If the emissions in one sector would change in the short term, the IOA would not be able to capture this new situation. When dealing with future scenarios, this becomes a limiting factor, and even more so when the electricity sector is involved. By using the current static information, any intermediate production of the electricity sector in the future scenarios would result in CO_2 emissions higher than what would correspond to that sector. In order to solve this, we have generated new emission factors for the electricity sector at each future scenario, which convert the model into a hybrid model. The new figures have been obtained from the optimization results from the *urbs* model, which provides the direct CO_2 emissions associated to the German electric energy mix at each scenario.

Having solved this limitation, we need to face another important problem of input-output models. These models rely on aggregated sector data, which might not precisely represent a particular good or service. This means that the model is not able to distinguish between the different energy technologies within the electricity sector. Based on this, producing 1€ of electricity from solar energy would result in the same figures as producing it from nuclear energy. To overcome this constraint, the most appropriate approach would be to break down the current electricity sector by technology, readjusting the supply and use tables, and then generating new technical coefficients for each technology. However, the large amount of assumptions and data required make this solution impractical.

As Zafrilla et al. (2014) have done, instead of disaggregating or creating a new sector for each technology, we have defined new final demand vectors for them, so that the output from each technology will be treated as an exogenous demand. Each final demand vector represents the cost function of the technology, which is linked to the activity sector in the input-output table which will supply the required good or service. Once the final demand vector is defined by each technology, we calculate the total amount of goods and services required by technology and scenario. To do that, we use the results obtained from the electricity system model. In fact, *urbs* provides the investment, fix, variable, and fuel costs by technology, all of which are necessary so that the demand of electricity is fulfilled in a cost-optimal way. Hence, *urbs* outputs will be the exogenous final demands for the MRIO analysis. Although *urbs* has been used to model the European electricity system, the analysis will focus on the German context. Therefore, only the environmental and socioeconomic impacts associated to the German electricity system are estimated, although the effects will be located not only in Germany but in the 40 remaining regions including in the WIOD.

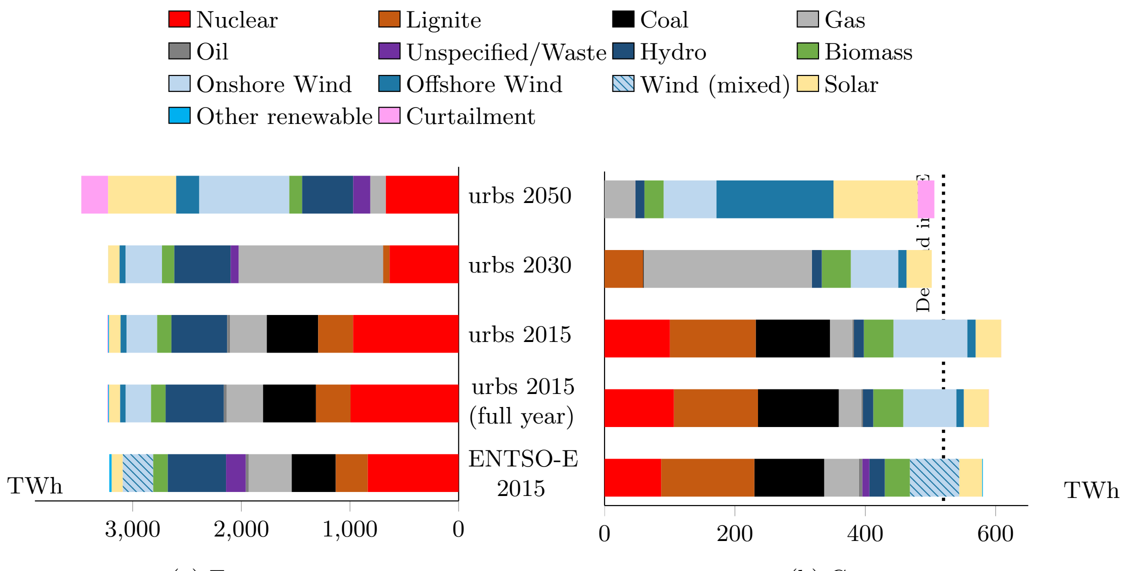
6.3 Results

In this section, the results of the electricity system optimization are discussed first, followed by a short sensitivity analysis. The main part provides an overview of results that are obtained by linking the optimization model to the MRIO.

6.3.1 Energy mixes of Europe and Germany

As stated in section 6.2.1, the optimization model delivers several outputs including the new capacities to be added, the hourly dispatch of power plants, and the costs and emissions caused directly by the operation of the electricity system. We aggregate the hourly dispatch of the power plants to obtain the energy mixes of Europe and Germany, which are shown in Figure 6.2.

The *urbs* model of Europe for 2015 does not allow any new capacities to be built. Hence, it is used for calibration[3] in order to match the energy mix according to ENTSO-E statistics (ENTSO-E, 2015). The result shows that the model overestimates the share of nuclear power in the energy mix, but underestimates the shares of coal, gas, and other mixed fuels, so that the ratio between conventional and renewable generation is the same. In terms of CO_2 emissions, the energy mix of the *urbs* model leads to slightly lower emissions.



(a) Europe (b) Germany

Figure 6.2: Electric energy supply in TWh according to the *urbs* model and to ENTSO-E statistics of 2015.

[3] In the calibration process, we use the full year model with 8760 time steps, scale down the hourly capacity factors for solar and wind power plants to match the yearly generation in Europe, and vary the prices of fuels within a certain range while preserving the merit-order of power plants.

In order to reduce the computation time of the optimization, we run all the models for four weeks ($4 \times 7 \times 24$ time steps) instead of a full year. In 2015, we only observe slight discrepancies due to an overestimation of the wind generation in Europe as a whole and in Germany in particular. Nevertheless, the error is tolerated considering the reduced computation time. Despite the differences between the reduced *urbs* model for 2015 and the ENTSO-E statistics, we can use the 2015 model as a basis to build the models of 2030 and 2050, as long as we conduct an analysis based on qualitative trends and relative quantitative changes. The model does not forecast the future of the electricity system.

When comparing the results for 2015 with the models for 2030 and 2050, we observe that the decarbonization of the electricity system occurs in two steps. In the mid-term, assuming a 50% reduction in CO_2 emissions, all coal power plants and most lignite power plants are phased out in Europe. They are mainly replaced by gas-fired power plants, and, to a lesser extent, by wind farms. In Germany, due to the lack of baseload nuclear capacity by 2030, lignite power plants still contribute to the electricity mix, even though their share shrinks to less than half what it used to be in 2015. According to the model, wind generation also decreases. The reason is that some wind parks would have reached the end of their lifetime by 2030, so they are decommissioned and replaced with gas-fired plants. This development is due to two limitations of the model. First, power plant retrofitting at lower costs than the costs of new installations is not implemented in the model. Second, the wind generation profile that is used in the model is an aggregate that does not reflect the diversity of the regions within Germany, which includes many profitable, cost-competitive onshore wind locations.

Furthermore, based on the model results, Germany becomes a net electricity importer in 2030, and the trend will intensify in 2050. One possible explanation lies in the nuclear phase-out. Since nuclear fission is a cheap way of generating electricity (if only the fuel costs are considered), other countries in Europe which still use nuclear power plants would be able to generate electricity in a more cost-competitive way than Germany.

The second step of the decarbonization occurs in the long-term and is characterized by a large-scale deployment of renewable power plants. Together, solar, onshore wind and offshore wind power plants may provide half of the energy demand in Europe. In Germany, their share could exceed 75%. Lignite is completely phased out of the system, and gas-fired power plants are the only technology that emits CO_2 directly during operation. However, the renewable power plants are not completely integrated into the system. Despite the investment in battery storage and pumped hydroelectric storage, a considerable amount of the power generated is curtailed.

In order to check the robustness of the results in spite of high future uncertainties, a sensitivity analysis is typically conducted for critical parameters. In the next section, we vary the investment costs of solar power plants, batteries, and wind power plants and observe their impacts on the *urbs* model of 2050.

6.3.2 Sensitivity analysis

A thorough sensitivity analysis covers all critical parameter variations, as well as combinations of these. In this section, we only show the results of the sensitivity analysis for two cases. In the first one, denoted by "PV-30%", we reduce the investment costs of PV modules and batteries by 30%. In the second one, "Wind-30%", onshore and offshore wind power plants are 30% cheaper than in the base case. The results for Europe and Germany are displayed in Figure 6.3.

Cheaper PV modules and batteries would lead to an increase in the share of solar in the energy mixes of Europe and Germany, but also to more onshore wind, less offshore wind, and more curtailment. This could be explained by the fact that a combination of solar PV and batteries can cover part of the base load, which would have required quasi-constant offshore wind generation otherwise. Onshore wind power plants, which are cheaper than offshore ones, would fill the gap,

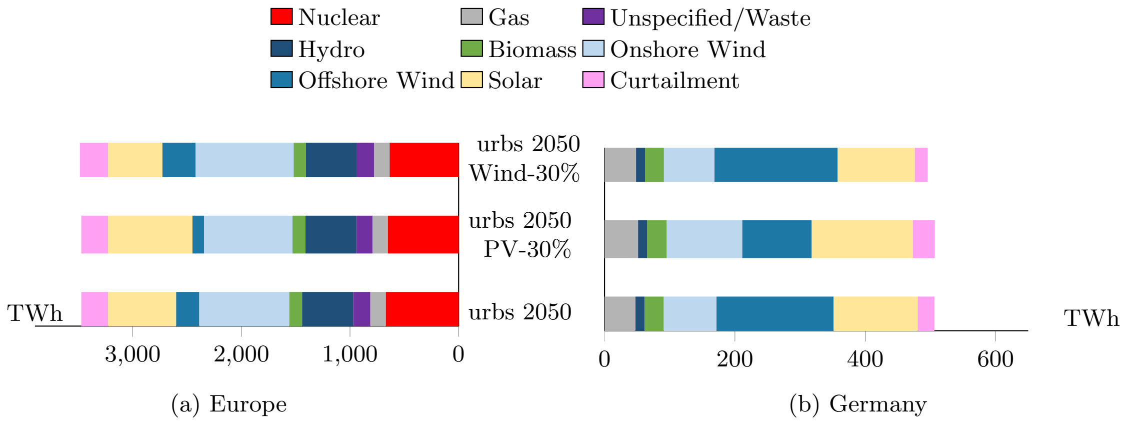


Figure 6.3: Electric energy supply in TWh according to the *urbs* model for various cost assumptions.

even though they cannot be completely integrated in the system, hence the additional curtailment. In the case of "Wind-30%", no big variations are noticeable. Thus, the model results are robust regarding the costs of wind power plants, but would vary if PV modules and batteries become cheaper than expected in 2050.

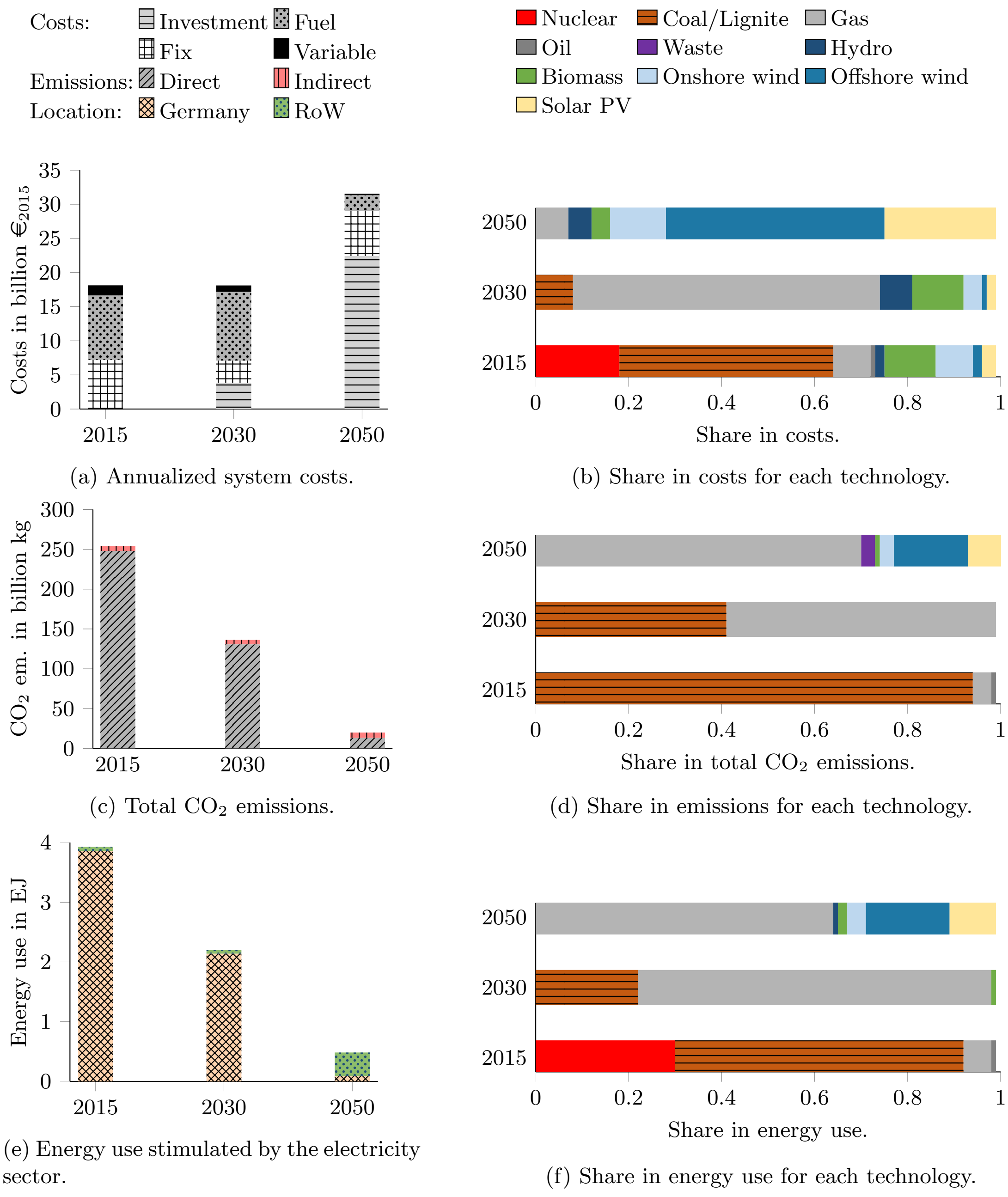
6.3.3 Results of the model coupling

As stated in section 6.2.3, the costs of investing in and operating the power system, and its direct CO_2 emissions, are delivered by the *urbs* models and used as inputs to define the final demand vectors for the MRIO analysis. In this section, we restrict the scope of the study to the case of Germnay and discuss the economic and environmental impacts of its electricity sector on a national and global scale. Major results are displayed in Figures 6.4 and 6.5.

Looking at the annualized system costs in Figure 6.4a, we observe that they almost stagnate between 2015 and 2030, before doubling in 2050. The stagnation of the absolute costs in 2030 coincides with a decline in the total electricity generation, which leads to higher specific costs per unit of electricity generated. The costs are driven by investments, with fuel and variable costs declining in the future. Up until 2030, all the renewable power plants made less than 30% of the system costs. In 2050, offshore wind plants are responsible for almost half the costs, even though they cover only one third the total electricity demand.

CO_2 emissions form the main constraints of the *urbs* models. However, the constraints only affected the direct emissions caused by the combustion of fuels. Other CO_2 emissions due to the increased economic activity, which was stimulated by the electricity sector, are aggregated together and displayed in Figure 6.4c as indirect emissions. In 2015 and 2030, the indirect emissions are negligible compared to the direct emissions, yet in 2050 they actually reach one third of the total emissions. Most of the indirect emissions can be traced back to the investment in offshore wind power plants and solar modules. In fact, when comparing the shares in indirect emissions to the shares in the electricity mix in 2050, it emerges that offshore wind plants cause the most indirect CO_2 emissions per unit of electricity produced, followed closely by PV, then by onshore wind.

Additionally, we compare the energy use stimulated by the electricity sector, which can give us an idea about the efficiency of the system. The total energy use decreases with stricter CO_2 constraints on direct emissions, but its decline is slower, with a decrease of 44% by 2030 (direct CO_2 emissions: -50%) and only 88% in 2050 (-95% for direct CO_2 emissions). The location of the energy use varies considerably, since most of it will happen outside of Germany by 2050. According to Figure 6.4f, the energy use in 2050 will be due to the gas-fired power plants (in Germany) and to offshore wind and solar power plants (in the rest of the world).



(a) Annualized system costs.

(b) Share in costs for each technology.

(c) Total CO$_2$ emissions.

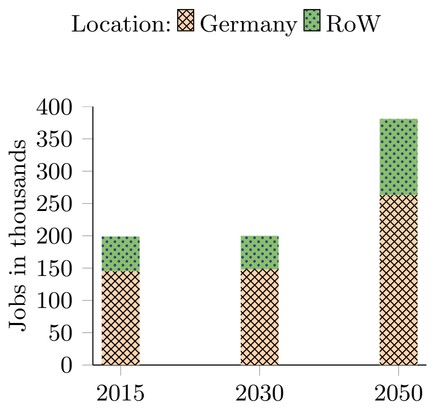
(d) Share in emissions for each technology.

(e) Energy use stimulated by the electricity sector.

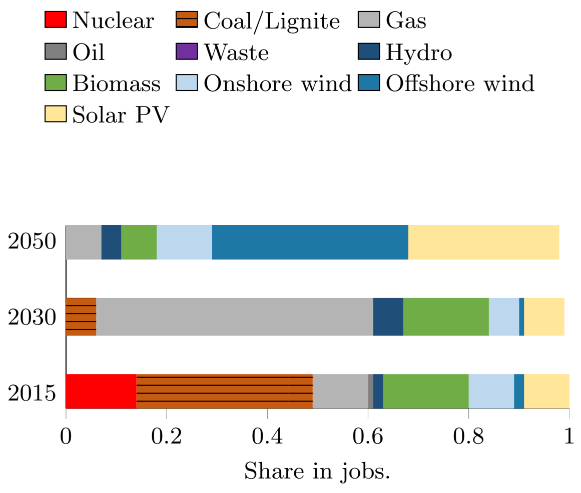
(f) Share in energy use for each technology.

Figure 6.4: Results of the *urbs* models and the MRIO analysis for Germany.

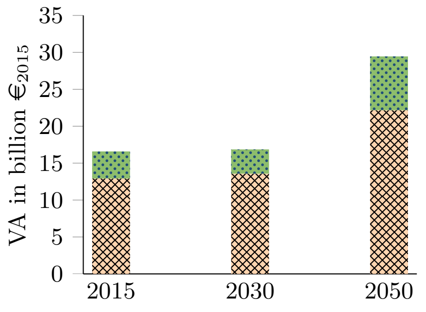
There is an obvious correlation between the evolution of the system costs on one side, and the total value added on the other side, which is visible in Figures 6.4a and 6.4i. However, it is interesting to see *where* the value added is created. Whereas the share of value added that is created outside Germany lies between 20% and 22% until 2030, it increases to 25% by 2050. Thus, the doubling of the value added, which is stimulated by the German electricity sector, will be more beneficial to other economies. The shares of the different technologies in the value added



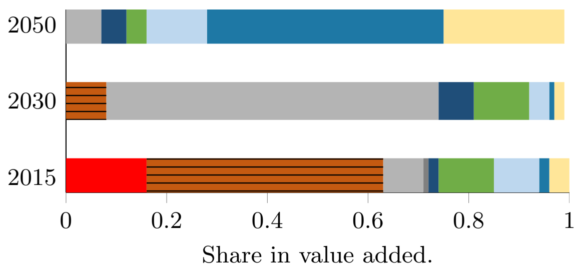
(g) Jobs stimulated by the electricity sector.



(h) Share in jobs for each technology.



(i) Total value added.



(j) Share in value added for each technology.

Figure 6.4: Results of the *urbs* models and the MRIO analysis for Germany (continued).

are similar to their shares in the costs.

The evolution of the number of jobs created, displayed in Figure 6.4g, follows a similar trend. In fact, we observe that the share of jobs created within Germany would decrease from 72% in 2015 to 69% in 2050. Interestingly, the highest share of jobs created within Germany occurs in 2030 (74%), and even though the absolute number of jobs remains stable between 2015 and 2030, the specific jobs created per unit of power generated are higher in 2030. Consequently, these findings highlight the fact that it is possible to decarbonize the German electricity sector in the midterm while preserving jobs and increasing the efficiency of the system. In terms of the shares in job creations for each technology, Figure 6.4h shows that biomass and PV feature among the most job-intensive technologies, especially if we consider their specific job creations per unit of power generated.

One of the strengths of a multiregional analysis is the ability to identify the regions that will be stimulated or that will contribute the most to certain impacts. For instance, we could analyze in more depth the socio-economic and environmental impacts derived from the energy system in 2030. As previously shown, Germany would keep 80% of the value added and 74% of the employment. 65% of the indirect CO_2 emissions associated to the new configuration of the energy system would occur in Germany. Europe, excluding Germany, would also benefit from the new energy system by keeping 12% of the value added generated and 9% of the job creation. Around 13% of the emissions would be located along other European regions. The case of other regions and countries, such China, deserves also some attention. China is nowadays one of the main primary suppliers, and a multiregional model should be able to track its relevance. As we could expect, any increase in the final demand of goods and services in Europe might have some consequences in China, and this is also the case for the energy system in 2030. Around 8% of the total employment generated will be located in China. However, the value added remaining in the country is very low, only 2%. This gives an idea about the economic structure of the country, meaning that the sectors being more stimulated have low productivity. In terms of CO_2 emissions, 7% of the indirect emissions would be located in China. This effect is known as *carbon leakage.*

The effects of decarbonization can be analysed by comparing the specific CO_2 emissions and the specific value added, even if we include carbon leakage. Decoupling economy growth and CO_2 emissions seems to be feasible through an economically optimal energy system. The reduction of CO_2 emissions would not slow down the growth of the economy, since it creates new opportunities that contribute to the generation of value added and employment as can be seen in Figure 6.5.

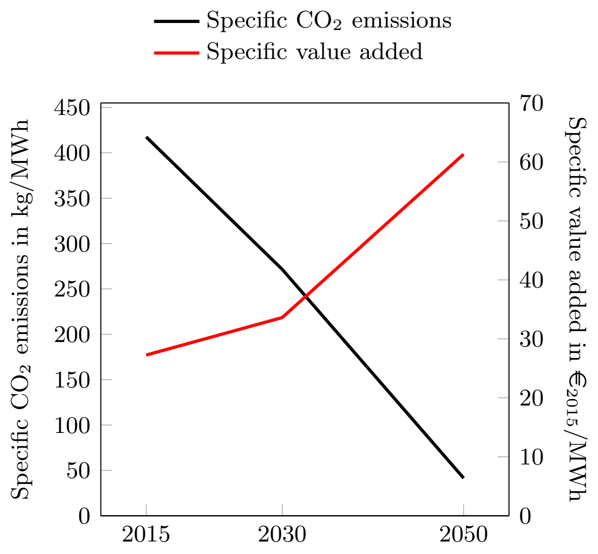


Figure 6.5: Evolution of the specific CO_2 emissions and the specific value added per unit of power generated between 2015 and 2050.

6.4 Conclusions and Discussion

This study has two main achievements: the first on the methodological level, and the second on the application level. Regarding the methodology, we successfully linked the outputs of an optimization model of the electricity system of Europe to an extended Multiregional Input-Output model. The combination takes advantage of the strengths of both models (high technical resolution for the optimization model, and geographic and sectoral coverage for the MRIO) to provide different perspectives on the electricity system of Germany in the mid and long term. The results go beyond the technical aspects to also address environmental and economic repercussions on other sectors, whether in Germany or in other countries. The method is faster and less data-intensive than detailed computable general equilibrium models with high technical and temporal resolutions, and complements the techno-economic optimizations with the environmental and socio-economic dimensions. It is also flexible, because the list of modeled technologies can be easily adapted to the region of interest. Even though the MRIO analysis has only been carried out for the electricity sector in Germany, the method is reproducible and can be applied to all the European countries which are modeled.

There are at least three limitations when applying this method. First, the model linking is only valid if the electricity system optimization is conducted on an almost autarkic system, i.e. if the electricity trade with other regions is considerably lower than the energy generated locally. This is because electricity imports are not included in the final demand in MRIO. Second, one major limitation from the side of the MRIO is related to the staticity of the database. In this study, we alleviated this problem by adjusting the specific CO_2 emissions and energy use for each year. However, other parts of the global economy were not changed, even though they will not remain constant until 2050. Limitations also arise from the fact that the linking is conducted as an *ex-post* analysis, where the results of the optimization cannot be altered anymore. If a feedback effect is desired, where the MRIO analysis adjusts the optimization targets and is affected by it, we obtain a multi-objective analysis that relies on a hard-linking of the models. This type of analysis was out of the scope of the study due to its higher complexity and might be tested in the future.

The second achievement concerns the outcomes of the analysis of the German electricity system. The study provides a detailed impact assessment of the mid and long-term decarbonization targets. Hence, the contribution of each technology to the energy mix, system costs, total CO_2 emissions, energy use, job creation, and value added are assessed to obtain a more holistic and nuanced view. The obtained results validate the idea of decoupling the economy growth from CO_2 emissions through the deployment of an optimized energy system. The results also show that, especially in the mid-term, the decarbonization can be achieved by phasing out coal and lignite power plants and replacing them with gas-fired power plants, without a dramatic increase of system costs and without net job losses. We believe that this analysis can support policy-makers in assessing the consequences of their policies from a national, regional and global perspective, and it can provide substantiated arguments for them when they reach out to a wider audience.

Since the focus of the study was the implementation of the method and displaying its capabilities, the future scenarios were defined in a simple way by only taking into account the direct CO_2 emissions constraints. However, there is a lot of uncertainty regarding the future electricity demand, which may increase significantly due to sector-coupling. Nevertheless, the method can be applied on other scenarios to conduct a thorough analysis of a future electricity system characterized by a high degree of sector-coupling.

Acknowledgements

The authors acknowledge the financial support by the Federal Ministry for Economic Affairs and Energy of Germany (BMWi) in the project “4NEMO - Research Network for the Development of New Methods in Energy System Modeling; research project: 0324008A”.

Chapter 7

Discussion

In this chapter, I will first discuss the overall approach towards modeling the spatial complexity in energy systems, then provide a critical review of the three publications from that perspective. Finally, I describe the measures I have undertaken to ensure its applicability.

New approach for modeling space

The comparison of the state of the art in modeling space in geography and in energy system modeling has shown that energy system modelers usually adopt the concept of *space as a container*, and divide it according to political and administrative spatial units. The approach that I described in this work expands the range of possibilities when defining the model regions. For most applications, space will still be modeled *as a container*. However, the spatial divisions can deviate from the administrative boundaries: they can be based on natural units or functional units, as defined by geographers, or they can even be unconventional and user-defined. This is critical since the optimal resolution is usually the outcome of a compromise between all the relevant aspects for the research question, and does not necessarily follow administrative divisions. One possible criticism of this approach is that the user-defined spatial units are not relatable to existing literature or available databases. Thus, the results cannot be easily interpreted or validated at the end of the processing phase. There are however ways to overcome this limitation. The simplest solution is to validate the results at the aggregate level, if the same scope is used in other studies and sources, as done in Chapter 5 for whole Europe. For a thorough validation, I recommend disaggregating the results into high resolution data, then aggregating them using conventional administrative divisions, for which the validation and the interpretation are more convenient. In other words, this is equivalent to replicating the steps of disaggregating then aggregating the data, using the results instead of the input data.

The concept of space as a *system of relations*, i.e. where the spatial organization of the components and the distances between them matters, is also used in energy system modeling. For research questions involving such a concept, the main contribution of this approach is the additional flexibility in setting the scale of the spatial units. By choosing smaller spatial units, the modeler preserves more information regarding the locations of the system components.

Analyzing space as a *category of perception*, i.e. observing the spatial structure of human behavior and how the perception of space affects it, is also found in energy system modeling literature, specifically in urban and regional planning. Although this dissertion does not focus on this aspect, the proposed approach can take into account the preferences of the stakeholders that occupy the space using a multi-criteria decision analysis, as in Chapter 4. For example, regarding the future expansion of renewable technologies, the preferences of the stakeholders can be represented using binary values for land suitability or weighting factors for land availability.

The suitability and availability coefficients can be arranged into layers, which are superposed to the resource pontential maps and are taken into considereation when defining the spatial units for potential projects.

The concept of *space as a social construct* is used in this thesis to reflect on how space is viewed within the field of energy system modeling, and from the outside. Due to the widespread use of political and administrative divisions, we would be tempted to think that policy-makers on different administrative levels hold a lot of power in shaping the energy system landscape through their policies. This might have been true in a world where conventional power plants can be built almost everywhere, and where the supply adapts to the demand. However, with the increasing share of intermittent renewable energy sources and the ambitious goals of carbon emission reductions, we are witnessing the return of determinism: the design of the energy system within a region will be mostly dictated by the expansion potential of the renewable energy technologies. Demand-side management will ensure that demand follows the supply patterns. Transmission lines and storage will guarantee higher spatial and temporal flexibility.

If we push the idea about the relationship between energy policy and space a bit further, we could envision a world where energy geography shapes the political boundaries, not the other way around. However, intermittent renewable energy sources, in particular wind and solar, have inherent limitations which go against this idea. On the one hand, their power densities (power per unit of area) are lower than those of conventional power plants. Hence, they would not be significant enough to justify a redefinition of political or administrative boundaries. On the other hand, the power density tends to not vary a lot within a short distance, i. e. the difference between neighboring regions would also hardly justify a redrawing of the administrative divisions.

Ultimately, the proposed definition of the optimal spatial resolution in Chapter 2 breaks with the convention that the political and administrative aspects are by default the most dominant aspects for any research question. In the light of the transformation of the energy system, with the increasing shares of renewable energy that is tied to the local geography, and the growing number of stakeholders operating in different sectors and at different scales, it is clear that the way energy modelers model space should evolve and adjust to the research questions they are answering. Nevertheless, the political aspect should still be considered when designing the model regions if the models are informing decision-makers on the impact of their energy-related policies.

High resolution: opportunities and drawbacks

The general approach that I suggest in Section 3.1 requires transforming the data into high resolution before aggregating it again into the desired resolution. While this improves the flexibility in modeling, it comes at a cost in terms of uncertainty. In fact, in addition to the uncertainty affecting the raw input data, another layer of uncertainty is now added through the disaggregation methods. My answer to this critique is threefold. First, if disaggregation can be avoided, then it should be. I encourage modelers to seek high resolution data that is already validated, instead of devising their own method of generating it. Fortunately, there is a growing trend towards making energy system data in high resolution publicly available, so this issue might become obsolete in the future. The use of disaggregation techniques can be seen as a bridging solution until open energy data becomes ubiquitous. Second, the modeler should be aware of the trade-off and be able to judge whether having a higher uncertainty is better than using a sub-optimal spatial resolution. Third, the modeler should bare in mind that part of the uncertainty is alleviated when the data is aggregated again based on the desired resolution. If the resolution remains high, then so does the uncertainty.

Another drawback of this approach is the additional computation that is needed to pre-process the data and convert it into the desired resolution. Here I would argue that the ease or difficulty of obtaining the data is part of the trade-off, because finding the data with the appropriate resolution (if possible) also costs resources (time, money, usage rights, etc.). Besides, the computational penalty depends on the pre-processing tools and the way they have been designed. I have personally shared the source code of the pre-processing tools that I have created, along with their documentation, hoping to make this step less challenging for other modelers.

Disaggregating the data is, in certain cases, not an option but a necessity. For research questions revolving around renewable potentials and the optimal siting of energy infrastructure (power plants, transmission lines, charging stations, etc.), the spatial units need to be small enough to capture land restrictions adequately. The techniques that are described in this thesis enable the modelers to answer such research questions, even if their original input data is coarse.

In general, the disaggregation techniques vary in their complexity, and in whether they include data correction processes. In the case of the first publication on the solar potential in South-East Asia, I performed a correction based on weather station data. Two possible criticisms are that such a correction might lead to overfitting with past data, and that it would not have been necessary if the reanalysis data was reliable in the first place. My answer to the first point is that reanalysis data sets, despite becoming the standard in such models for potential estimations, were not designed to be used for energy system applications, and so contain large biases that need to be corrected. The risk of overfitting with past data exists, irrespective of whether that data is measured weather parameters or power plant generation. In both cases, future developments and the impacts of climate change are not considered, but the advantage of the correction using weather stations is that the obtained data is technology-neutral, so it can be used with future technology assumptions. Regarding the reliability of reanalysis data, there are currently few alternatives on a global scale, and the growing research on disaggregation techniques and downscaling will eventually compensate for their drawbacks.

All in all, taking into account the trends in computational power and open data availability, I consider that moving towards high resolution data is a move in the right direction. The complexity of the pre-processing and post-processing steps will recede with increased automation, higher computational power, and better data availability. Meanwhile, better understanding of local physical and social phenomena will improve the quality of the downscaling techniques, and consequently, of the high resolution data they generate. The sharing of the algorithms and the scripts for disaggregation as open-source codes might speed up the quality improvement within the scientific community.

Clustering of geographic data

In order to generate model inputs for the desired resolution, I presented some aggregation techniques, with a focus on data-driven clustering of geographic data in Chapter 5. The analysis of the obtained regions has shown that they are indeed more homogeneous in terms of the properties used for clustering (solar PV full-load hours, for example), so that the information loss is mindered if the regions are represented by only a few parameters (such as a single PV time series, in the same example). Despite using the same underlying high-resolution data, differences have emerged when using the various spatial units in energy system models, especially in the long-term future where renewable energy plays a major role in the energy system.

The paper challenges two misconceptions. The first one is that the model regions do not matter, and that any regions can be used, which has so far led to political and administrative divisions becoming the norm in energy system modeling. The second is more subtle: whereas there are modelers who have shown that the spatial resolution matters, their key message was that the *number* of regions matters, with no statement regarding the *shape*. Surely, whenever the number of regions changes, the shapes also change. But does choosing different shapes for the same number of regions make a difference? Although former studies were silent on this issue, I argue that it does make a difference, and that the shape matters. If the computational power is constrained, the number of regions cannot be varied, but their shape and, consequently, the quality of the results can still be different.

However, the experiment in Chapter 5 has its limitations. First, it used a linear optimization model, so the observed variation in the final results could be due to the sensitivity of linear models and how they tend to "all-or-nothing" solutions. This is indeed possible, yet it does not make the sensitivity to model regions boundaries less relevant than other aspects, such as technology cost assumptions. The paper is a proof of concept which shows that the choice of regions impacts the results in linear optimization models, but further testing is still needed on other types of models. Second, the reasons for the discrepancies should be investigated further. The study presented some clues, but the individual impacts of each aspect could not be measured. Proving the causality was not possible on such a complex system as the electricity system of Europe. For such an analysis, I recommend a smaller simplistic model, where the link between cause and effect could be established.

Regarding the clustering method in general, a common criticism was that only one aspect (solar potential, wind potential, energy demand) was used at a time. This was actually done with the purpose of facilitating the interpretation of the results. The current version of the tool, whose documentation is enclosed in Appendix B, allows the clustering based on multiple aspects at once, with the possibility of giving weights to the different data layers. It also includes an algorithm for hierarchical clustering of transmission lines, to identify regions that preserve grid bottlenecks (see example in Figure 1.2). In all these cases, it is assumed that the spatial data is static, and does not change over time. The algorithm cannot be applied for example on rasters of geothermal water flow or on networks of varying power flows, because the rate at which these rasters change is probably faster than the algorithm in its current form. Such a dynamic clustering could happen over bigger time scales, like years, but not hours. For small time scales, the alternative would be to cluster 3-D data, where time is the third dimension. This is useful for optimization problems as well, because the spatio-temporal correlation between load patterns and renewable energy generation matters. However, this is hardly possible using the current tool because of the max-p algorithm which cannot deal with large data sets. One viable alternative would be to reduce the temporal dimension to critical hours, days, or seasons, assuming that storage devices could flatten intraday or interseasonal variations.

Model coupling and the spatial complexity

By providing an example of a model coupling experiment using an optimization model for the electricity system of Europe and a global, economy-wide Input-Output analysis, I showed how model coupling can expand the geographic scope and provide answers that individual models are not capable of delivering. In another publication, which is not part of this thesis but is included in the list of publications (Boiarchuk et al., 2018), I coupled a unit commitment optimization model with an AC power flow model of Europe. In that case, the models shared the same geographic scope but used different scales (administrative divisions versus individual nodes in the transmission grid). Thus, model coupling may affect the spatial complexity in two different ways: either by enlarging the study area, or by zooming into smaller scales. However, as I have mentioned in Section 3.4, there is a lack of standardized methods to perform model coupling due to the variety of combinations of models and scopes. In order to overcome this, I provided some guidance to modelers on critical aspects of model coupling. This can be built upon to establish standard methods in the future.

In Chapter 6, the impacts of the decarbonization of the European electricity system on indirect emissions, energy use, job creation, and value added in the global economy was investigated. Although the model coupling makes it possible to see the impacts of each country within Europe on the rest of the world, the results were not given in that detail, and were limited to the implications of the decarbonization in Germany. Since the emphasis of the study was on the model coupling, only a subset of the results was shown as a proof of concept. Further results can be shared in a future analysis with a focus on the application itself.

One fundamental criticism of the method is that the coupling was performed in one direction, from the optimization model to the Input-Output tables, with no feedback loop. In general, feedback loops are possible in model coupling, but in the case of the study of Chapter 6 they were not implemented because Input-Output tables do not generate endogenous commodity prices, unlike CGE models. Also, the outcome could have been different if the effects on the global economy and the environment were internalized in a single optimization model. In fact, "hard linking" could deliver an optimum to an objective function that takes those externalities into account. However, the resulting model would be very complex, and the problem probably intractable if the technical, temporal, and spatial resolutions are high. Besides, it could be difficult to establish causalities using a complex model with many features and capabilities. Modelers could avoid these issues if they use good, simple, and highly specialized models and couple them using "soft linking".

Applicability of the proposed approach and methods

The use of the optimal spatial resolution challenges the status quo in energy system modeling and requires additional efforts, particularly in the pre-processing and post-processing phases of modeling. Hence, I have proceeded on three different fronts in order to ensure the applicability of the proposed approach in this thesis.

First, I have derived the approach and the related methods from my work on different research projects, in particular the 4NEMO project (Research Network for the Development of New Methods in Energy System Modeling) which is supported by the Federal Ministry for Economic Affairs and Energy of Germany (BMWi). One of the objectives of that project was to develop and apply new methods for increasing the spatial resolution and to improve the depiction of regional characteristics. Therefore, I developed and implemented the techniques which are described in the thesis to fill that research gap. Through the inclusion of my research findings in the project report, they might reach a broad audience.

Second, I have made my scripts available as open-source codes with permissive licenses, and included a detailed description of the modules and a users' guide for installation and use. This reduces the adoption barrier for prospective users and could lead to improved quality and higher trust, since the codes are open to public scrutiny. Other users can suggest improvements or describe their issues. Based on their requirements, the tools can be continuously expanded to include more features and cover more applications.

Third, I have organized a few workshops to introduce the tools to colleagues and prospective users. Although this measure is on a local level, it has a high impact since it ensures active knowledge transfer and establishes a pool of maintainers for the tools, which is critical in the long-term. The workshops also offered the possibility to the participants to describe their use cases and learn how to apply the tools to answer their research questions.

Chapter 8

Conclusion

Energy modeling frameworks are widely used to provide policy recommendations about different energy system designs, taking into account technical, economic, social and environmental aspects. The use of mathematical frameworks and computer software requires the simplification, abstraction and discretization of the physical world, in particular of space.

Different concepts exist for modeling space. In the field of geography, where space is the main focus, four major concepts have been in use at different times over the last century: *space as a container*, *space as a system of relations*, *space as a category of perception*, and *space as a construct*. A literature review has revealed that there are applications of these concepts in energy system modeling. However, the use of political and administrative boundaries to define model regions is predominant. This could be explained by the ease of obtaining data on that level of detail, by the ease of interpreting, validating, and communicating the results, and/or by the common belief that such spatial divisions are adequate for most energy system analyses.

New trends in energy systems are challenging the way space is modeled. The increasing share of decentralized generation from renewable energy sources requires a high level of detail to capture their technical potential on a regional level. The use of different spatial scales is necessary to capture the interactions between the different energy sectors, due to increased sector coupling. The improvements in data availability and computational power offer the chance to explore alternative spatial divisions to political and administrative boundaries.

The aim of this thesis was to address those challenges and exploit the opportunities by providing more flexibility to modelers in the process of defining the spatial units of their models. Since there is a large number of ways of dividing space into discrete spatial units for energy modeling, there must be at least one optimal spatial division for a given research question under certain constraints. The following is a short summary of the central questions in this thesis:

What is the optimal spatial resolution? The second chapter provided a working definition of the optimal spatial resolution, which is mainly dictated by the limits of technical observability, the scope of the research question, the required accuracy, and the computational complexity. It proposed an algorithm for energy system modelers to ensure that all relevant aspects for the research question are respected. The definition acknowledges the trade-off between higher resolution and accuracy, recommending the use of validated input data, or of data that has been transformed with validated methods. It also recognizes that the maximum number of model regions is constrained by the computation time, which depends on the technical and temporal complexities of the model as well.

How to obtain the optimal resolution? (Theoretical approach) In Chapter 3, a systematic approach for transforming data into the optimal spatial resolution is proposed. For research questions which can be answered with one model, it is possible to first disaggregate the data, increase its resolution in an intermediate step, and finally aggregate it again according to the desired resolution. If more than one model is needed, model coupling can be used to expand the modeling capabilities and/or the geographic scope. The chapter also includes an overview of modeling techniques that enable the transformation of the data. Value repetition, interpolation, and the combination of different data sources can be used to increase the spatial resolution. Data aggregation, whether it is user-defined or data-driven, could be achieved through statistical summaries or sampling. To compensate for the lack of standards for model coupling, some guidance on how to define the shared scope, how to avoid syntax and semantic errors, and how to manage data sharing between the models is provided.

How to reach the optimal resolution? (Applications) Chapters 4, 5, and 6 are first-author publications in scientific journals that applied the techniques from Chapter 3. The first publication was about the estimation of the photovoltaic potential in South East Asia. It illustrated how to combine information from different data sources, some of it in low resolution, in order to obtain a dataset in high spatial resolution for the technical potential. The second publication introduced a novel clustering method for high resolution data in order to obtain regions with specific characteristics, based on the homogeneity of the solar potential, wind potential, and total energy demand, respectively. It applied data-driven techniques to decrease the resolution by ensuring that the loss of information is less critical than conventional aggregation methods. The third publication described a model coupling exercise involving an optimization model for expansion planning and economic dispatch for Europe and a global Input-Output analysis. The models had different geographic scopes and different purposes. Through the model coupling, the implications of the decarbonization of the European power system on the economy and the environment in and outside of Europe were investigated in a high technological detail by combining the capabilities of the two models.

How to reach the optimal resolution? (Tools) A further contribution of this work is the distribution of three open-source tools in GitHub, each one with a detailed documentation included in the annex of this thesis. The first one, pyGRETA, generates high-resolution potential maps and time series for user-defined regions within the globe. The second, pyCLARA, clusters high-resolution spatial data (rasters or polylines connecting Voronoi polygons) into contiguous, homogeneous regions. The third, pyPRIMA, automates the creation of energy system models using a common database. The tools were developed to facilitate the flexible modeling of space in energy system models.

Why does the spatial resolution matter? Existing literature has already established that varying the number of model regions has an impact on the model results, in particular on the share of renewable energy and on the transmission grid expansion. The novelty of the study in Chapter 5 lies in the fact that the number of model regions was constant. While the underlying data is the same and the number of regions is kept constant, the mere variation of the shapes of the regions leads to large differences in the cost-optimal energy mix (shares of solar/wind can be 10% higher or lower in Europe), particularly in the future under stringent CO_2 constraints. Hence, the shape of the spatial units should be chosen carefully. By testing multiple spatial units, a spatial sensitivity can identify robust results which are independent of the boundaries of the regions.

Outlook

This thesis opens up the opportunity to answer further research questions. Future work should expand the theoretical framework while providing practical solutions for energy system modelers.

The definition of the optimal resolution could be formalized and generalized for other types of models. Currently, the limitations of the computational infrastructure are considered in the proposed definition, which makes the optimality relative to the resource-constrained reality of each modeler. An absolute, model-independent optimality could be defined for each research question. It would serve as a benchmark, where each deviation from the optimum could be quantified and its impact on the results estimated.

Regarding the modeling techniques, it would be interesting to have an assessment of the aggregation or disaggregation techniques that modelers can use in a given use case, in terms of computation requirements and accuracy of results. Besides, the aggregation functions should be varied and their impact estimated. In the analysis in Chapter 5 for example, representative solar photovoltaic and onshore wind time series were generated for the locations with median full-load hour (FLH) values. Instead of the FLH values, other aggregation functions for time series could be used, and instead of picking the median value in the regions, alternative spatial aggregation functions could be investigated.

The multi-criteria potential assessment from Chapter 4 used binary parameters to exclude unsuitable areas, and coefficients to weight available areas. In light of the increasing share of renewable energy sources, and the increasing resistance to new wind and transmission line projects that go with it, there is an urgent demand for improving the modeling of social acceptance and reflecting its uncertainty in potential assessment maps.

The clustering technique in Chapter 5 was useful in proving that the choice of the model regions matters, even when using the same underlying data. There is a huge potential of applications that can be derived out of the study. Using a simplified model with generic data could help establish the causality between the regional aggregation and the level of accuracy. For real case applications, it would be interesting to see the impact of combining different input maps in order to generate model regions with complex features.

As mentioned before in this thesis, there is a lack of a standardized approach for model coupling. Based on the study from Chapter 6, some guidelines for modelers were derived, focusing on the spatial aspect. Future research could build on that to create a standard procedure, or even a "plug-and-play" model coupling framework. Due to the emergence of highly specialized models, there will be a growing demand for such solutions, which should have a solid theoretical background from a system perspective.

In the spirit of open science, other researchers could access the shared open source tools, use them and develop them further. Potential improvements could be categorized in three groups:

- Software engineering perspective: make the computation faster, reduce the memory usage, improve the coding style, improve accessibility by creating downloadable stand-alone packages, etc.
- Power engineering perspective: improve the existing technical models, add new model features, etc.
- System perspective: connect the tools to each other, expand the interface between the tools and other existing models and data sources, etc.

Appendix A

Documentation of tum-ens/pyGRETA

This is the documentation of *pyGRETA: python Generator of REnewable Time series and mAps*, which generates high-resolution potential maps and time series for user-defined regions within the globe.

Features

- Generation of potential maps and time series for user-defined regions within the globe;
- Modeled technologies: onshore wind, offshore wind, PV, CSP (user-defined characteristics);
- Use of MERRA-2 reanalysis data, with the option to detect and correct outliers;
- High resolution potential taking into account the land use suitability/availability, topography, bathymetry, slope, distance to urban areas, etc.;
- Statistical reports with summaries (available area, maximum capacity, maximum energy output, etc.) for each user-defined region;
- Generation of several time series for each technology and region, based on user's preferences;
- Possibility to combine the time series into one using linear regression to match given full-load hours and temporal fluctuations.

Applications

This code is useful if you want to:

- estimate the theoretical and/or technical potential of an area, which you can define through a shapefile, in high resolution;
- define your own technology characteristics;
- generate time series for an area after excluding unsuitable parts for renewable power plants;
- generate multiple time series for the same area (best site, upper 10%, median, etc.);
- match historical capacity factors of countries from the IRENA database.

Contributors

I created the first version of this code in Matlab. Houssame Houmy translated it into Python, and we both added new modules and edited the code structure. The individual contributions can be traced back in GitHub (Siala and Houmy, 2020).

Contents

Chapter 1

User manual

1.1 Installation

Note: We assume that you are familiar with git and conda.

First, clone the git repository in a directory of your choice using a Command Prompt window:

```
$ ~\directory-of-my-choice> git clone https://github.com/tum-ens/pyGRETA.git
```

We recommend using conda and installing the environment from the file `ren_ts.yml` that you can find in the repository. In the Command Prompt window, type:

```
$ cd pyGRETA\env\
$ conda env create -f ren_ts.yml
```

Then activate the environment:

```
$ conda activate ren_ts
```

In the folder `code`, you will find multiple files:

File	Description
config.py	used for configuration, see below.
runme.py	main file, which will be run later using `python runme.py`.
lib\initialization.py	used for initialization.
lib\input_maps.py	used to generate input maps for the scope.
lib\potential.py	contains functions related to the potential estimation.
lib\time_series.py	contains functions related to the generation of time series.
lib\regression.py	contains functions related to the regression.
lib\spatial_functions.py	contains helping functions related to maps, coordinates and indices.
lib\physical_models.py	contains helping functions for the physical/technological modeling.
lib\correction_functions.py	contains helping functions for data correction/cleaning.
lib\util.py	contains minor helping functions and the necessary python libraries to imported.

1.2 config.py

This file contains the user preferences, the links to the input files, and the paths where the outputs should be saved. The paths are initialized in a way that follows a particular folder hierarchy. However, you can change the hierarchy as you wish.

1.2.1 Main configuration function

`config.`**`configuration`**`()`

This function is the main configuration function that calls all the other modules in the code.

Return (paths, param) The dictionary paths containing all the paths to inputs and outputs, and the dictionary param containing all the user preferences.

Return type tuple(dict, dict)

`config.`**`general_settings`**`()`

This function creates and initializes the dictionaries param and paths. It also creates global variables for the root folder `root`, and the system-dependent file separator `fs`.

Return (paths, param) The empty dictionary paths, and the dictionary param including some general information.

Return type tuple(dict, dict)

Note: Both *param* and *paths* will be updated in the code after running the function `config.configuration`.

Note: `root` points to the directory that contains all the inputs and outputs. All the paths will be defined relatively to the root, which is located in a relative position to the current folder.

The code differentiates between the geographic scope and the subregions of interest. You can run the first part of the script `runme.py` once and save results for the whole scope, and then repeat the second part using different subregions within the scope.

`config.`**`scope_paths_and_parameters`**`(`*paths*, *param*`)`

This function defines the path of the geographic scope of the output *spatial_scope* and of the subregions of interest *subregions*. Both paths should point to shapefiles of polygons or multipolygons. It also associates two name tags for them, respectively *region_name* and *subregions_name*, which define the names of output folders.

- For *spatial_scope*, only the bounding box around all the features matters. Example: In case of Europe, whether a shapefile of Europe as one multipolygon, or as a set of multiple features (countries, states, etc.) is used, does not make a difference. Potential maps (theoretical and technical) will be later generated for the whole scope of the bounding box.
- For *subregions*, the shapes of the individual features matter, but not their scope. For each individual feature that lies within the scope, you can later generate a summary report and time series. The shapefile of *subregions* does not have to have the same bounding box as *spatial_scope*. In case it is larger, features that lie completely outside the scope will be ignored, whereas those that lie partly inside it will be cropped using the bounding box of *spatial_scope*. In case it is smaller, all features are used with no modification.
- *year* defines the year of the input data.
- *technology* defines the list of technologies that you are interested in. Currently, four technologies are defined: onshore wind `'WindOn'`, offshore wind `'WindOff'`, photovoltaics `'PV'`, concentrated solar power `'CSP'`.

Parameters

- **`paths`** (*`dict`*) – Dictionary including the paths.
- **`param`** (*`dict`*) – Dictionary including the user preferences.

Return (paths, param) The updated dictionaries paths and param.

Return type tuple(dict, dict)

Note: We recommend using a name tag that describes the scope of the bounding box of the regions of interest. For example, `'Europe'` and `'Europe_without_Switzerland'` will actually lead to the same output for the first part of the code.

Note: As of version 1.0.1, it is possible to use different technologies in the same run, but not the same technology with different settings.

Warning: If you intend to use the wind correction feature relying on the Global Wind Atlas, it is recommended that *spatial_scope* covers **all** the countries that you are interested in, because the correction is done on a country-level. Also, you have to download the data from the Global Wind Atlas for each country that lies within the scope, even partially, and put it in the corresponding location.

1.2.2 User preferences

`config.`**`computation_parameters`**(*param*)

This function defines parameters related to the processing:

- *nproc* is an integer that limits the number of parallel processes (some modules in `potential.py` and `time_series.py` allow parallel processing).
- *CPU_limit* is a boolean parameter that sets the level of priority for all processes in the multiprocessesing. Leave `True` if you plan on using the computer while FLH and TS are being computed, `False` for fastest computation time.

Parameters **`param`** (*`dict`*) – Dictionary including the user preferences.

Return param The updated dictionary param.

Return type dict

`config.`**`resolution_parameters`**(*param*)

This function defines the resolution of weather data (low resolution), and the desired resolution of output rasters (high resolution). Both are numpy arrays with two numbers. The first number is the resolution in the vertical dimension (in degrees of latitude), the second is for the horizontal dimension (in degrees of longitude).

Parameters **`param`** (*`dict`*) – Dictionary including the user preferences.

Return param The updated dictionary param.

Return type dict

Note: As of version 1.0.1, these settings should not be changed. Only MERRA-2 data can be used in the tool. Its spatial resolution is 0.5° of latitudes and 0.625° of longitudes. The high resolution is 15 arcsec in both directions.

`config.`**`weather_data_parameters`**(*param*)

This function defines the coverage of the weather data *MERRA_coverage*, and how outliers should be corrected using *MERRA_correction*:

- *MERRA_coverage*: If you have downloaded the MERRA-2 data for the world, enter the name tag `'World'`. The code will later search for the data in the corresponding folder. It is possible to download the MERRA-2 just for the geographic scope of the analysis. In that case, enter another name tag (we recommend using the same one as the spatial scope).
- *MERRA_correction*: MERRA-2 contains some outliers, especially in the wind data. *MERRA_correction* sets the threshold of the relative distance between the yearly mean of the data point to the yearly mean of its neighbors.

Parameters **`param`** (*`dict`*) – Dictionary including the user preferences.

Return param The updated dictionary param.

Return type dict

`config.`**`file_saving_options`**(*param*)

This function sets some options for saving files.

- *savetiff* is a boolean that determines whether tif rasters for the potentials are saved (`True`), or whether only mat files are saved (`False`). The latter are saved in any case.
- *report_sampling* is an integer that sets the sample size for the sorted FLH values per region (relevant for `potential.report_potentials`).

Parameters **`param`** (*`dict`*) – Dictionary including the user preferences.

Return param The updated dictionary param.

Return type dict

`config.`**`time_series_parameters`**(*param*)

This function determines the time series that will be created.

- *quantiles* is a list of floats between 100 and 0. Within each subregion, the FLH values will be sorted, and points with FLH values at a certain quantile will be later selected. The time series will be created for these points. The value 100 corresponds to the maximum, 50 to the median, and 0 to the minimum.
- *regression* is a dictionary of options for `regression.regression_coefficients`:
  - *solver* is the name of the solver for the regression.
  - *WindOn* is a dictionary containing a list of hub heights that will be considered in the regression, with a name tag for the list.
  - *WindOff* is a dictionary containing a list of hub heights that will be considered in the regression, with a name tag for the list.
  - *PV* is a dictionary containing a list of orientations that will be considered in the regression, with a name tag for the list.
  - *CSP* is a dictionary containing a list of settings that will be considered in the regression, with a name tag for the list.

  If all the available settings should be used, you can leave an empty list.
- *modes* is a dictionary that groups the quantiles and assigns names for each subgroup. You can define the groups as you wish. If you want to use all the quantiles in one group without splitting them in subgroups, you can write:

```
param["modes"] = {"all": param["quantiles"]}
```

- *combo* is a dictionary of options for `time_series.generate_stratified_timeseries`:
  - *WindOn* is a dictionary containing the different combinations of hub heights for which stratified time series should be generated, with a name tag for each list.
  - *WindOff* is a dictionary containing the different combinations of hub heights for which stratified time series should be generated, with a name tag for each list.

  - *PV* is a dictionary containing the different combinations of orientations for which stratified time series should be generated, with a name tag for each list.
  - *CSP* is a dictionary containing the different combinations of settings for which stratified time series should be generated, with a name tag for each list.

If all the available settings should be used, you can leave an empty list.

Parameters **`param`** (`dict`) – Dictionary including the user preferences.

Return param The updated dictionary param.

Return type dict

`config.`**`landuse_parameters`**(*param*)

This function sets the land use parameters in the dictionary *landuse* inside param:

- *type* is a numpy array of integers that associates a number to each land use type.
- *type_urban* is the number associated to urban areas (useful for `input_maps.generate_buffered_population`).
- *Ross_coeff* is a numpy array of Ross coefficients associated to each land use type (relevant for `physical_models.loss`).
- *albedo* is a numpy array of albedo coefficients between 0 and 1 associated to each land use type (relevant for reflected irradiation, see `physical_models.calc_CF_solar`).
- *hellmann* is a numpy array of Hellmann coefficients associated to each land use type (relevant for `correction_functions.generate_wind_correction`).
- *height* is a numpy array of gradient heights in meter associated to each land use type (relevant for `correction_functions.generate_wind_correction`).

Parameters **`param`** (`dict`) – Dictionary including the user preferences.

Return param The updated dictionary param.

Return type dict

Land use reclassification:

```
# 0   -- Water
# 1   -- Evergreen needle leaf forest
# 2   -- Evergreen broad leaf forest
# 3   -- Deciduous needle leaf forest
# 4   -- deciduous broad leaf forest
# 5   -- Mixed forests
# 6   -- Closed shrublands
# 7   -- Open shrublands
# 8   -- Woody savannas
# 9   -- Savannas
# 10  -- Grasslands
# 11  -- Permanent wetland
# 12  -- Croplands
# 13  -- Urban and built-up
# 14  -- Croplands / natural vegetation mosaic
# 15  -- Snow and ice
# 16  -- Barren or sparsely vegetated
```

`config.`**`protected_areas_parameters`**(*param*)

This function sets the parameters for protected areas in the dictionary *protected_areas* inside param:

- *type* is a numpy array of integers that associates a number to each protection type.
- *IUCN_Category* is an array of strings with names associated to each protection type (for your information).

Parameters **`param`** (*`dict`*) – Dictionary including the user preferences.

Return param The updated dictionary param.

Return type dict

`config.`**`pv_parameters`**(*param*)

This function sets the parameters for photovoltaics in the dictionary *pv* inside param:

- *resource* is a dictionary including the parameters related to the resource potential:
  - *clearness_correction* is a factor that will be multiplied with the clearness index matrix to correct it. If no correction is required, leave it equal to 1.
- *technical* is a dictionary including the parameters related to the module:
  - *T_r* is the rated temperature in °C.
  - *loss_coeff* is the loss coefficient (relevant for `physical_models.loss`).
  - *tracking* is either 0 for no tracking, 1 for one-axis tracking, or 2 for two-axes tracking.
  - *orientation* is the azimuth orientation of the module in degrees relative to the equator.
  - The tilt angle from the horizontal is chosen optimally in the code, see `physical_models.angles`.
- *mask* is a dictionary including the parameters related to the masking:
  - *slope* is the threshold slope in percent. Areas with a larger slope are excluded.
  - *lu_suitability* is a numpy array of values 0 (unsuitable) or 1 (suitable). It has the same size as the array of land use types.
  - *pa_suitability* is a numpy array of values 0 (unsuitable) or 1 (suitable). It has the same size as the array of protected area categories.
- *weight* is a dictionary including the parameters related to the weighting:
  - *GCR* is a dictionary of design settings for the ground cover ratio:
    - *shadefree_period* is the number of shadefree hours in the design day.
    - *day_north* is the design day for the northern hemisphere.
    - *day_south* is the design day for the southern hemisphere.
  - *lu_availability* is a numpy array of values between 0 (completely not available) and 1 (completely available). It has the same size as the array of land use types.
  - *pa_availability* is a numpy array of values between 0 (completely not available) and 1 (completely available). It has the same size as the array of protected area categories.
  - *power_density* is the power density of PV projects, assuming a GCR = 1, in MW/m^2.
  - *f_performance* is a number smaller than 1, taking into account all the other losses from the module until the AC substation.

Parameters **`param`** (*`dict`*) – Dictionary including the user preferences.

Return param The updated dictionary param.

Return type dict

Raises **`Tracking Warning`** – If *tracking* is not set to 0 and *orientation* is given as a value other than 0 or 180 (South or North), the orientation is ignored.

`config.`**`csp_parameters`**(*param*)

This function sets the parameters for concentrated solar power in the dictionary *csp* inside param:

- *resource* is a dictionary including the parameters related to the resource potential:

  - *clearness_correction* is a factor that will be multiplied with the clearness index matrix to correct it. If no correction is required, leave it equal to 1.

- *technical* is a dictionary including the parameters related to the module:
  - *T_avg_HTF* is the average temperature in °C of the heat transfer fluid between the inlet and outlet of the solar field.
  - *loss_coeff* is the the heat loss coefficient in W/(m^{2}K), which does not depend on wind speed (relevant for `physical_models.calc_CF_solar`).
  - *loss_coeff_wind* is the the heat loss coefficient in W/(m^{2}K(m/s)^0.6), which depends on wind speed (relevant for `physical_models.calc_CF_solar`).
  - *Flow_coeff* is a factor smaller than 1 for the heat transfer to the HTF (Flow or heat removal factor).
  - *AbRe_ratio* is the ratio between the receiver area and the concentrator aperture.
  - *Wind_cutoff* is the maximum wind speed for effective tracking in m/s.
- *mask* is a dictionary including the parameters related to the masking:
  - *slope* is the threshold slope in percent. Areas with a larger slope are excluded.
  - *lu_suitability* is a numpy array of values 0 (unsuitable) or 1 (suitable). It has the same size as the array of land use types.
  - *pa_suitability* is a numpy array of values 0 (unsuitable) or 1 (suitable). It has the same size as the array of protected area categories.
- *weight* is a dictionary including the parameters related to the weighting:
  - *lu_availability* is a numpy array of values between 0 (completely not available) and 1 (completely available). It has the same size as the array of land use types.
  - *pa_availability* is a numpy array of values between 0 (completely not available) and 1 (completely available). It has the same size as the array of protected area categories.
  - *power_density* is the power density of CSP projects in MW/m^2.
  - *f_performance* is a number smaller than 1, taking into account all the other losses from the CSP module until the AC substation.

Parameters **`param`** (*`dict`*) – Dictionary including the user preferences.

Return param The updated dictionary param.

Return type dict

`config.`**`onshore_wind_parameters`**(*param*)

This function sets the parameters for onshore wind in the dictionary *windon* inside param:

- *resource* is a dictionary including the parameters related to the resource potential:
  - *res_correction* is either 1 (perform a redistribution of wind speed when increasing the resolution) or 0 (repeat the same value from the low resolution data). It is relevant for `correction_functions.generate_wind_correction`.
  - *topo_correction* is either 1 (perform a correction of wind speed based on the altitude and the Global Wind Atlas) or 0 (no correction based on altitude).
  - *topo_weight* is only relevant if *topo_correction* = 1. It defines how to weight the correction factors of each country. There are three options: `'none'` (all countries have the same weight), `'size'` (larger countries have a higher weight), or `'capacity'` (countries with a higher installed capacity according to IRENA have a higher weight).
- *technical* is a dictionary including the parameters related to the wind turbine:
  - *w_in* is the cut-in speed in m/s.

- *w_r* is the rated wind speed in m/s.
- *w_off* is the cut-off wind speed in m/s.
- *P_r* is the rated power output in MW.
- *hub_height* is the hub height in m.

- *mask* is a dictionary including the parameters related to the masking:
  - *slope* is the threshold slope in percent. Areas with a larger slope are excluded.
  - *lu_suitability* is a numpy array of values 0 (unsuitable) or 1 (suitable). It has the same size as the array of land use types.
  - *pa_suitability* is a numpy array of values 0 (unsuitable) or 1 (suitable). It has the same size as the array of protected area categories.
  - *buffer_pixel_amount* is an integer that defines the number of pixels making a buffer of exclusion around urban areas.
- *weight* is a dictionary including the parameters related to the weighting:
  - *lu_availability* is a numpy array of values between 0 (completely not available) and 1 (completely available). It has the same size as the array of land use types.
  - *pa_availability* is a numpy array of values between 0 (completely not available) and 1 (completely available). It has the same size as the array of protected area categories.
  - *power_density* is the power density of onshore wind projects in MW/m^2.
  - *f_performance* is a number smaller than 1, taking into account all the other losses from the turbine generator until the AC substation.

Parameters **`param`** (*`dict`*) – Dictionary including the user preferences.

Return param The updated dictionary param.

Return type dict

`config.`**`offshore_wind_paramters`**(*param*)

This function sets the parameters for offshore wind in the dictionary *windoff* inside param:

- *resource* is a dictionary including the parameters related to the resource potential:
  - *res_correction* is either 1 (perform a redistribution of wind speed when increasing the resolution) or 0 (repeat the same value from the low resolution data). It is relevant for `correction_functions.generate_wind_correction`.
- *technical* is a dictionary including the parameters related to the wind turbine:
  - *w_in* is the cut-in speed in m/s.
  - *w_r* is the rated wind speed in m/s.
  - *w_off* is the cut-off wind speed in m/s.
  - *P_r* is the rated power output in MW.
  - *hub_height* is the hub height in m.
- *mask* is a dictionary including the parameters related to the masking:
  - *depth* is the threshold depth in meter (negative number). Areas that are deeper are excluded.
  - *pa_suitability* is a numpy array of values 0 (unsuitable) or 1 (suitable). It has the same size as the array of protected area categories.
- *weight* is a dictionary including the parameters related to the weighting:
  - *lu_availability* is a numpy array of values between 0 (completely not available) and 1 (completely available). It has the same size as the array of land use types.

- *pa_availability* is a numpy array of values between 0 (completely not available) and 1 (completely available). It has the same size as the array of protected area categories.
- *power_density* is the power density of offshore wind projects in MW/m^2.
- *f_performance* is a number smaller than 1, taking into account all the other losses from the turbine generator until the AC substation.

Parameters **`param`** (*`dict`*) – Dictionary including the user preferences.

Return param The updated dictionary param.

Return type dict

1.2.3 Paths

`config.`**`weather_input_folder`**(*paths*, *param*)

This function defines the path *MERRA_IN* where the MERRA-2 data is saved. It depends on the coverage of the data and the year.

Parameters

- **`paths`** (*`dict`*) – Dictionary including the paths.
- **`param`** (*`dict`*) – Dictionary including the user preferences.

Return paths The updated dictionary paths.

Return type dict

`config.`**`global_maps_input_paths`**(*paths*)

This function defines the paths where the global maps are saved:

- *LU_global* for the land use raster
- *Topo_tiles* for the topography tiles (rasters)
- *Pop_global* for the global population raster
- *Bathym_global* for the bathymetry raster
- *Protected* for the shapefile of protected areas
- *GWA* for the country data retrieved from the Global Wind Atlas (missing the country code, which will be filled in a for-loop in :mod:correction_functions.calc_gwa_correction)
- *Countries* for the shapefiles of countries
- *EEZ_global* for the shapefile of exclusive economic zones of countries

Parameters **`paths`** (*`dict`*) – Dictionary including the paths.

Return paths The updated dictionary paths.

Return type dict

`config.`**`output_folders`**(*paths*, *param*)

This function defines the paths to multiple output folders:

- *region* is the main output folder.
- *weather_data* is the output folder for the weather data of the spatial scope.
- *local_maps* is the output folder for the local maps of the spatial scope.
- *potential* is the output folder for the ressource and technical potential maps.
- *regional_analysis* is the output folder for the time series and the report of the subregions.
- *regression_in* is the folder where the regression parameters (FLH, fitting time series) are saved.

- *regression_out* is the output folder for the regression results.

All the folders are created at the beginning of the calculation, if they do not already exist,

Parameters

- **paths** (*dict*) – Dictionary including the paths.
- **param** (*dict*) – Dictionary including the user preferences.

Return paths The updated dictionary paths.

Return type dict

`config.`**`weather_output_paths`**(*paths*, *param*)

This function defines the paths to weather filesfor a specific *year*:

- *W50M* is the file for the wind speed at 50m in m/s.
- *CLEARNESS* is the file for the clearness index, e.g. the ratio between total ground horizontal radiation and total top-of-the-atmosphere horizontal radiation.
- *T2M* is the file for the temperature at 2m in Kelvin.

Parameters

- **paths** (*dict*) – Dictionary including the paths.
- **param** (*dict*) – Dictionary including the user preferences.

Return paths The updated dictionary paths.

Return type dict

`config.`**`local_maps_paths`**(*paths*, *param*)

This function defines the paths where the local maps will be saved:

- *LAND* for the raster of land areas within the scope
- *EEZ* for the raster of sea areas within the scope
- *SUB* for the raster of areas covered by subregions (both land and sea) within the scope
- *LU* for the land use raster within the scope
- *BATH* for the bathymetry raster within the scope
- *TOPO* for the topography raster within the scope
- *SLOPE* for the slope raster within the scope
- *PA* for the raster of protected areas within the scope
- *POP* for the population raster within the scope
- *BUFFER* for the raster of population buffer areas within the scope
- *CORR_GWA* for correction factors based on the Global Wind Atlas (mat file)
- *CORR_ON* for the onshore wind correction factors (raster)
- *CORR_OFF* for the offshore wind correction factors (raster)
- *AREA* for the area per pixel in m^2 (mat file)

Parameters

- **paths** (*dict*) – Dictionary including the paths.
- **param** (*dict*) – Dictionary including the user preferences.

Return paths The updated dictionary paths.

Return type dict

`config.`**`irena_paths`**(*paths*, *param*)

This function defines the paths for the IRENA inputs and outputs:

- *IRENA* is a csv file containing statistics for all countries and technologies for a specific *year*, created using a query tool of IRENA.
- *IRENA_dict* is a csv file to convert the code names of countries from the IRENA database to the database of the shapefile of countries.
- *IRENA_summary* is a csv file with a summary of renewable energy statistics for the countries within the scope.

Parameters

- **`paths`** (*`dict`*) – Dictionary including the paths.
- **`param`** (*`dict`*) – Dictionary including the user preferences.

Return paths The updated dictionary paths.

Return type dict

`config.`**`regression_paths`**(*paths*, *param*, *tech*)

This function defines the paths for the regression parameters:

- *FLH_regression* is a csv file containing FLH statistics for the subregions and the four technologies for a specific *year*, based on the previously created *IRENA_summary*.
- *TS_regression* is a csv file containing time series to be match for each subregion and technology, based on EMHIRES time series if available.

Parameters **`paths`** (*`dict`*) – Dictionary including the paths.

Return paths The updated dictionary paths.

Return type dict

`config.`**`emhires_input_paths`**(*paths*, *param*, *tech*)

This function defines the path to the EMHIRES input file for each technology (only `'WindOn'`, `'WindOff'`, and `'PV'` are supported by EMHIRES).

Parameters

- **`paths`** (*`dict`*) – Dictionary including the paths.
- **`param`** (*`dict`*) – Dictionary including the user preferences.
- **`tech`** (*`string`*) – Name of the technology.

Return paths The updated dictionary paths.

Return type dict

`config.`**`potential_output_paths`**(*paths*, *param*, *tech*)

This function defines the paths of the files that will be saved in the folder for the potential outputs:

- *FLH* is the file with the full-load hours for all pixels within the scope (mat file).
- *mask* is the file with the suitable pixels within the scope (mat file).
- *FLH_mask* is the file with the full-load hours for the suitable pixels within the scope (mat file).
- *weight* is the power density for all the pixels in the scope (mat file).
- *FLH_weight* is the potential energy output for all the pixels in the scope (mat file).

Parameters

- **`paths`** (*`dict`*) – Dictionary including the paths.
- **`param`** (*`dict`*) – Dictionary including the user preferences.

- **tech** (*string*) – Name of the technology.

Return paths The updated dictionary paths.

Return type dict

`config.`**`regional_analysis_output_paths`**(*paths*, *param*, *tech*)

This function defines the paths of the files that will be saved in the folder for the regional analysis outputs:

- *Locations* is the shapefile of points that correspond to the selected quantiles in each subregion, for which the time series will be generated.
- *TS* is the csv file with the time series for all subregions and quantiles.
- *Region_Stats* is the csv file with the summary report for all subregions.
- *Sorted_FLH* is the mat file with the sorted samples of FLH for each subregion.
- *Regression_coefficients* is the path format for a csv files containing the regression coefficients found by the solver
- *Regression_TS* is the path format for a csv files with the regression resulting timeseries for the tech and settings

Parameters

- **paths** (*dict*) – Dictionary including the paths.
- **param** (*dict*) – Dictionary including the user preferences.
- **tech** (*string*) – Name of the technology.

Return paths The updated dictionary paths.

Return type dict

1.3 runme.py

`runme.py` calls the main functions of the code:

```python
from lib.correction_functions import generate_wind_correction
from lib.initialization import initialization
from lib.input_maps import generate_maps_for_scope
from lib.potential import calculate_full_load_hours, mask_potential_maps, weight_
↪potential_maps, report_potentials
from lib.regression import get_regression_coefficients
from lib.time_series import (
    find_representative_locations,
    generate_time_series_for_representative_locations,
    generate_time_series_for_regions,
    generate_time_series_for_specific_locations,
)

if __name__ == "__main__":

    paths, param = initialization()

    # Generate input raster maps
    generate_maps_for_scope(paths, param)

    # Wind speed correction
    # if "WindOn" in param["technology"] or "WindOff" in param["technology"]:
    #     generate_wind_correction(paths, param)

    for tech in param["technology"]:
```

(continues on next page)

(continued from previous page)

```
        print("Tech: " + tech)

        # Generate potential maps and reports
        calculate_full_load_hours(paths, param, tech)
        mask_potential_maps(paths, param, tech)
        weight_potential_maps(paths, param, tech)
        report_potentials(paths, param, tech)

        # Generate time series
        find_representative_locations(paths, param, tech)
        generate_time_series_for_representative_locations(paths, param, tech)
        generate_time_series_for_specific_locations(paths, param, tech)

    for tech in param["technology"]:
        print("Tech: " + tech)

        # Generate regression coefficients for FLH and TS model matching
        # get_regression_coefficients(paths, param, tech)

        # Generate times series for combinations of technologies and locations
        # generate_time_series_for_regions(paths, param, tech)
```

1.4 Recommended input sources

For a list of GIS data sources, check this wikipedia article.

1.4.1 Weather data from MERRA-2

The most important inputs within this model are the weather time series. These are taken from the Modern-Era Retrospective Analysis for Research and Applications, version 2 (MERRA-2), which is the latest atmospheric reanalysis of the modern satellite era produced by NASA's Global Modeling and Assimilation Office (GMAO) [5]. The parameters taken from MERRA-2 are:

- Global Horizontal Irradiance (*GHI*): Downward shortwave radiation received by a surface horizontal to the ground (*SWGDN* in MERRA-2 nomenclature).
- Top of the Atmosphere Irradiance (*TOA*): Downward shortwave radiation at the top of the atmosphere (*SWTDN* in MERRA-2 nomenclature).
- Air temperature 2 meters above the ground (*T2M*).
- Northward wind velocity at 50 meters (*V50M*).
- Eastward wind velocity at 50 meters (*U50M*).

The *GHI* and *TOA* data are time-averaged hourly values given in W/m while *T2M* data are instantaneous values in Kelvin. *V50M* and *U50M* are instantaneous hourly values given in m/s.

The spatial arrangement of the data consists of a global horizontal grid structure with a resolution of 576 points in the longitudinal direction and 361 points in the latitudinal direction, resulting in pixels of 5/8° longitude and 1/2° latitude [1].

It is possible to download MERRA-2 dataset for the whole globe or just for a subset of your region of interest. Depending on the *MERRA_coverage* parameter in config.py, the script can accept both datasets. Note that downloading the coverage for the whole globe is easier but will require a significant amount of space on your drive (coverage of the whole globe requires 13.6 Gb for one year).

In both cases, please follow these instructions to download the MERRA-2 dataset:

1. In order to download MERRA-2 data using the FTP server, you first need to create an Eathdata account (more on that on their website).

2. Navigate to the link for the FTP sever here.
3. In *Data Product*, choose tavg1_2d_slv_NX and select the *Parameters* T2M, U50M, V50M to downaload the temperature and the wind speed datasets.
4. In *Spatial Search*, enter the coordinates of the bounding box around your region of interest or leave the default values for the whole globe. To avoid problems at the edge of the MERRA-2 cells, use the following set of formulas:

$$minLat = \left\lfloor \frac{s+0.25}{0.5} \right\rfloor \cdot 0.5 - \epsilon$$
$$maxLat = \left\lceil \frac{n-0.25}{0.5} \right\rceil \cdot 0.5 + \epsilon$$
$$minLon = \left\lfloor \frac{w+0.3125}{0.625} \right\rfloor \cdot 0.625 - \epsilon$$
$$maxLon = \left\lceil \frac{e-0.3125}{0.625} \right\rceil \cdot 0.625 + \epsilon$$

   where *[s n w e]* are the southern, northern, western, and eastern bounds of the region of interest, which you can read from the shapefile properties in a GIS software, and
   $epsilon$ a small number.
5. In *Temporal Order Option*, choose the year(s) of interest.
6. Leave the other fields unchanged (no time subsets, no regridding, and NetCDF4 for the output file format).
7. Repeat the steps 4-6 for the *Data Product* tavg1_2d_rad_Nx, for which you select the *Parameters* SWGDN and SWTDN, the surface incoming shortwave flux and the top of the atmosphere incoming shortwave flux.
8. Follow the instructions in the website to actually download the NetCDF4 files from the urls listed in the text files you obtained.

If you follow these steps to download the data for the year 2015, you will obtain 730 NetCDF files, one for each day of the year and for each data product.

1.4.2 Raster of land use

Another important input for this model is the land use type. A land use map is useful in the sense that other parameters can be associated with different landuse types, namely:

- Urban areas
- Ross coefficients
- Hellmann coefficients
- Albedo
- Suitability
- Availability
- Installation cost
- etc.

For each land use type, we can assign a value for these parameters which affect the calculations for solar power and wind speed correction. The global land use raster for which `lib.input_maps.generate_landuse` has been written cannot be downloaded anymore (broken link), but a newer version is available from the USGS website. However, this new version requires additional data processing. The spatial resolution of the land use raster, and therefore of the other geographic intermediate rasters used in this model, is 1/240° longitude and 1/240° latitude.

1.4.3 Shapefile of the region of interest

The strength of the tool relies on its versatility, since it can be used for any user-defined regions provided in a shapefile. If you are interested in administrative divisions, you may consider downloading the shapefiles from the website of the Global Administration Divisions (GADM). You can also create your own shapefiles using a GIS software.

Warning: In any case, you need to have at least one attribute called *NAME_SHORT* containing a string (array of characters) designating each sub-region.

1.4.4 Shapefile of countries

A shapefile of all the countries of the world is also needed. It can be downloaded again from GADM. The attribute "GID_0" contains the ISO 3166-1 Alpha-3 codes of the countries, and is currently hard coded in the script.

Warning: If you want to use another source or other code names, you need to edit the name of the attribute "GID_0" and the dictionary *dict_countries.csv*.

1.4.5 Shapefile of Exclusive Economic Zones (EEZ)

A shapefile of the maritime boundaries of all countries is available at the website of the Flanders Marine Institute (VLIZ). It is used to identify offshore areas.

1.4.6 Raster of topography / elevation data

A high resolution raster (15 arcsec = 1/240° longitude and 1/240° latitude) made of 24 tiles can be downloaded from viewfinder panoramas.

1.4.7 Raster of bathymetry

A high resolution raster (60 arcsec) of bathymetry can be downloaded from the website of the National Oceanic and Atmospheric Administration (NOAA). The one used in the database is ETOPO1 Ice Surface, cell-registered.

1.4.8 Raster of population density

A high resolution raster (30 arcsec) of population density can be downloaded from the website of SEDAC after registration.

1.4.9 Shapefile of protected areas

Any database for protected areas can be used with this tool, in particular the World Database on Protected Areas published by the International Union for Conservation of Nature (IUCN). The shapefile has many attributes, but only one is used in the tool: "IUCN_CAT". If another database is used, an equivalent attribute with the different categories of the protection has to be used and `config.py` has to be updated accordingly.

1.4.10 Wind frequencies from the Global Wind Atlas

In order to correct the data biases of MERRA-2, especially in high-altitude regions, this tool uses wind speed frequencies of each country that are obtained from the Global Wind Atlas. Select the country that you need, choose 50m height, and download the plot data for the wind speed in a folder with the name of the country (use the ISO 3166-1 Alpha-3 code).

Note: You need to download the data for all the countries that lie in the scope of the study.

1.5 Recommended workflow

The script is designed to be modular and split into four main modules: *`lib.input_maps`*, *`lib.potential`*, *`lib.time_series`*, and *`lib.regression`*.

Warning: The outputs of each module serve as inputs to the following module. Therefore, the user will have to run the script sequentially.

The recommended use cases of each module will be presented in the order in which the user will have to run them.

1. *Input raster maps*
2. *Potential maps and reports*
3. *Time series for quantiles and user-defined locations*
4. *Regression*
5. *Stratified time series*

The use cases associated with each module with examples of their outputs are presented below.

It is recommended to thoroughly read through the configuration file *config.py* and modify the input paths and computation parameters before starting the *runme.py* script. Once the configuration file is set, open the *runme.py* file to define what use case you will be using the script for.

1.5.1 Input raster maps

The *`lib.input_maps`* module is used to generate data (mostly raster maps, but also arrays in MAT files) for the spatial scope defined by the user. These data sets include:

- Weather data
- Land and sea masking
- Subregions masking
- Land use
- Bathymetry
- Topography
- Slope
- Population and the population buffer masking
- Area

All these maps are needed before the potential or time series modules can be used for a specific spatial scope.

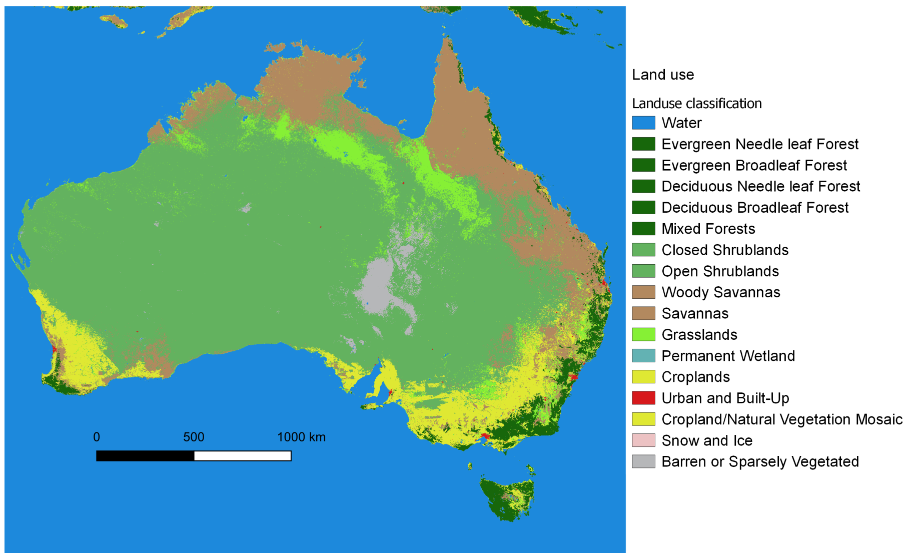


Fig. 1: Land Use Raster Map - Australia

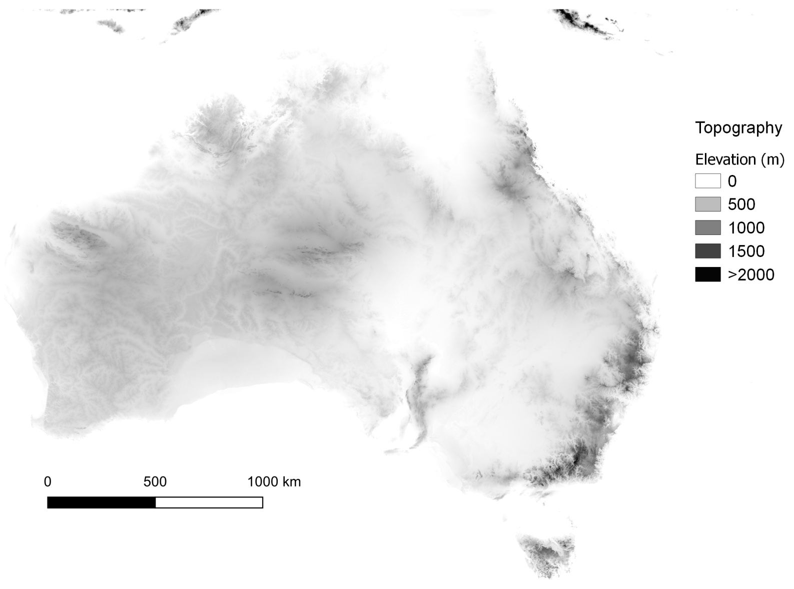


Fig. 2: Topography Raster Map - Australia

1.5.2 Potential maps and reports

The `lib.potential` module serves to generate potential raster maps for all four technologies supported by the script. This module generates a Full-Load Hour (FLH) raster map, masking and masked rasters for unsuitable and protected areas, and a weighting and weighted raster used for energy and power potential calculations. It also generates a CSV report containing metrics for each subregion:

- Available number of pixels, before and after masking
- Available area in in km^2
- FLH mean, median, max, min values, before and after masking
- FLH standard deviation after masking
- Power Potential in GW, before and after weighting
- Energy Potential in TWh in total, after weighting, and after masking and weighting
- Sorted sample of FLH values for each region

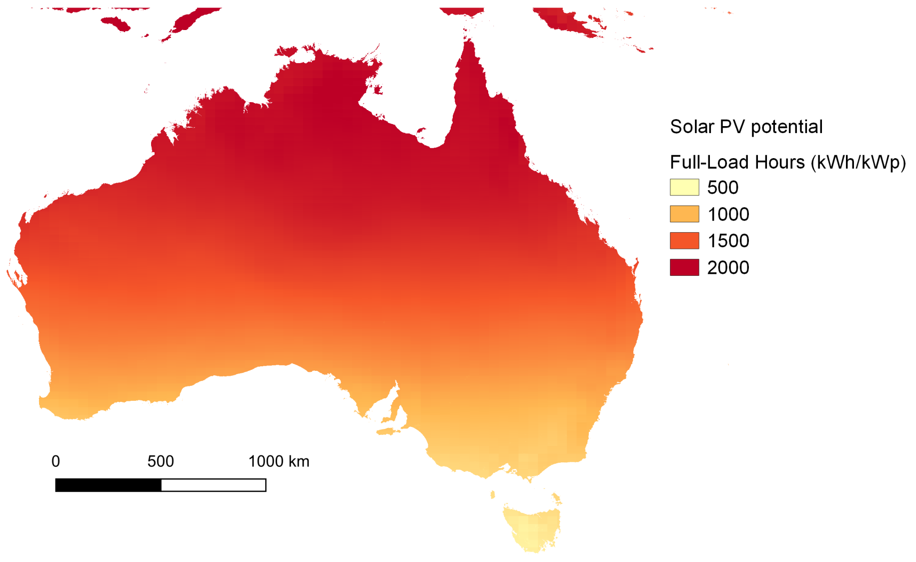


Fig. 3: FLH of solar PV - Australia

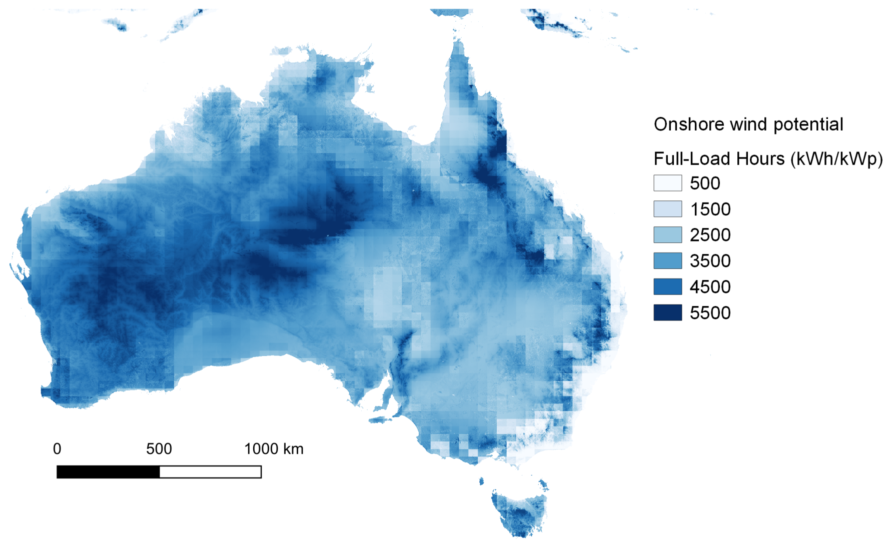


Fig. 4: FLH of onshore wind - Australia

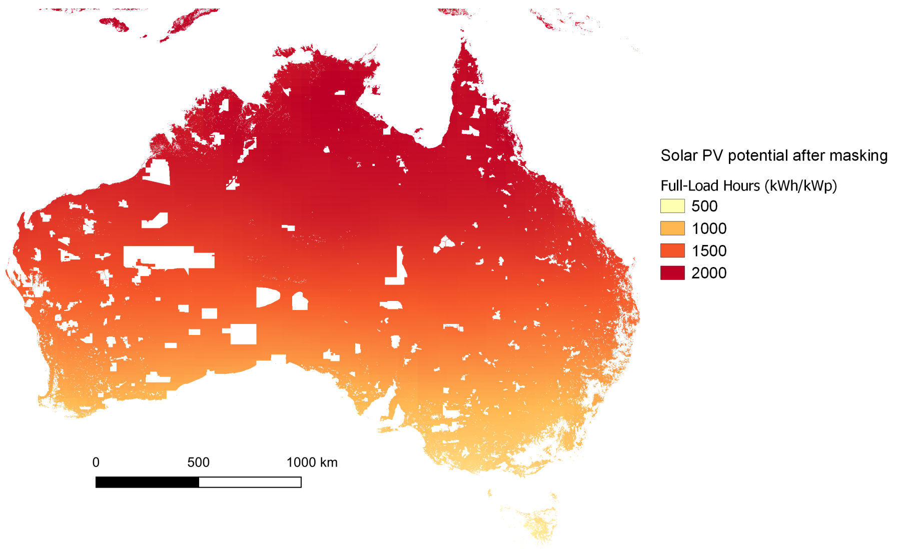


Fig. 5: FLH of solar PV after masking - Australia

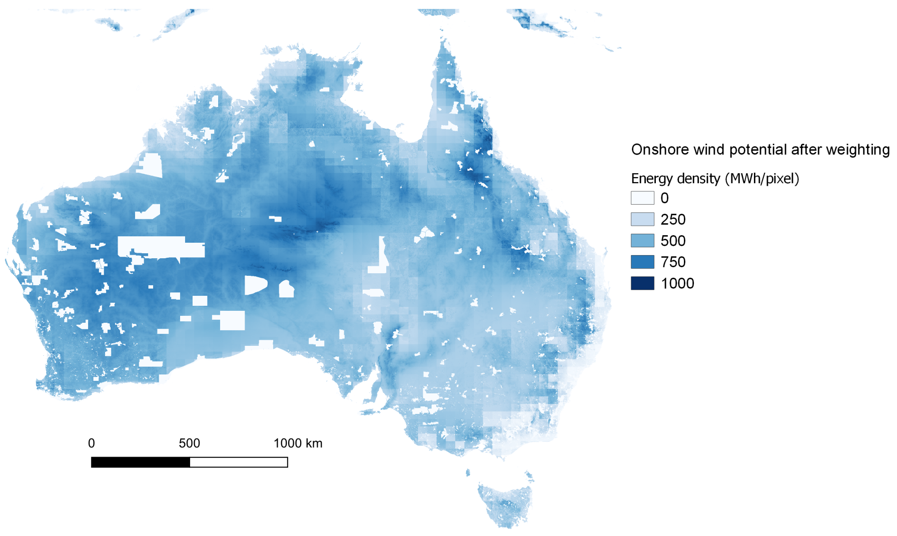

Onshore wind potential after weighting
Energy density (MWh/pixel)
0
250
500
750
1000
0
500
1000 km

Fig. 6: Energy output of onshore wind after weighting - Australia

Sample of potential report:

Region	Available area (km^2)
Region A	4315.7
Region B	2128.3
Region C	561.3
Region D	100953.1
Region E	10.2
Region F	2829.8

FLH mean	FLH max	FLH min	Masked FLH mean	Masked FLH max	Masked FLH min	Power potential (GW)	Energy potential (TWh)
1638.4	1686.3	1578.0	1644.3	1686.3	1589.7	6.5	10.8
1682.9	1699.7	1601.6	1684.2	1695.0	1613.7	1.4	2.4
1849.7	1853.4	1833.8	1849.6	1853.3	1840.8	0.9	1.7
2017.6	2090.5	1986.8	2018.0	2086.1	1986.8	183.7	369.8
1856.8	1857.1	1856.5	1856.8	1857.1	1856.5	0.0	0.0
1729.5	1772.2	1659.1	1731.4	1772.2	1659.1	4.8	8.3

1.5.3 Time series for quantiles and user-defined locations

The `lib.time_series` module allows to generate time series for quantiles as well as user-defined locations based on the FLH raster maps generated in the previously mentioned module. It is therefore important for the FLH raster maps to be generated first, in order to locate the quantiles. However, generating time series for user-defined locations does not require the potential maps to be generated beforehand.

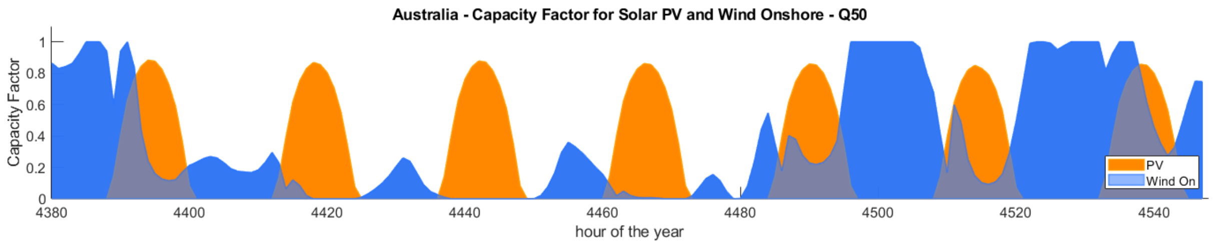


Fig. 7: Wind Onshore and Solar PV capacity factor time series for quantile 50 - Australia

1.5.4 Regression

Once a set of time series for different settings (hub heights for wind technologies, orientations for solar PV) is generated, the `lib.regression` module allows the user to find a combination of settings and quantiles in order to match a known FLH value and a given (typical) time series. The output is a set of regression coefficients that should be multiplied with the time series.

1.5.5 Stratified time series

Part of the `lib.time_series` module, the `lib.time_series.generate_time_series_for_regions` function reads the regression coefficients and the generated time series, and combines them into user-defined *modes* (combinations of quantiles) and *combos* (combinations hub height or orientations settings).

Example: - Graphic of Modes and Combos

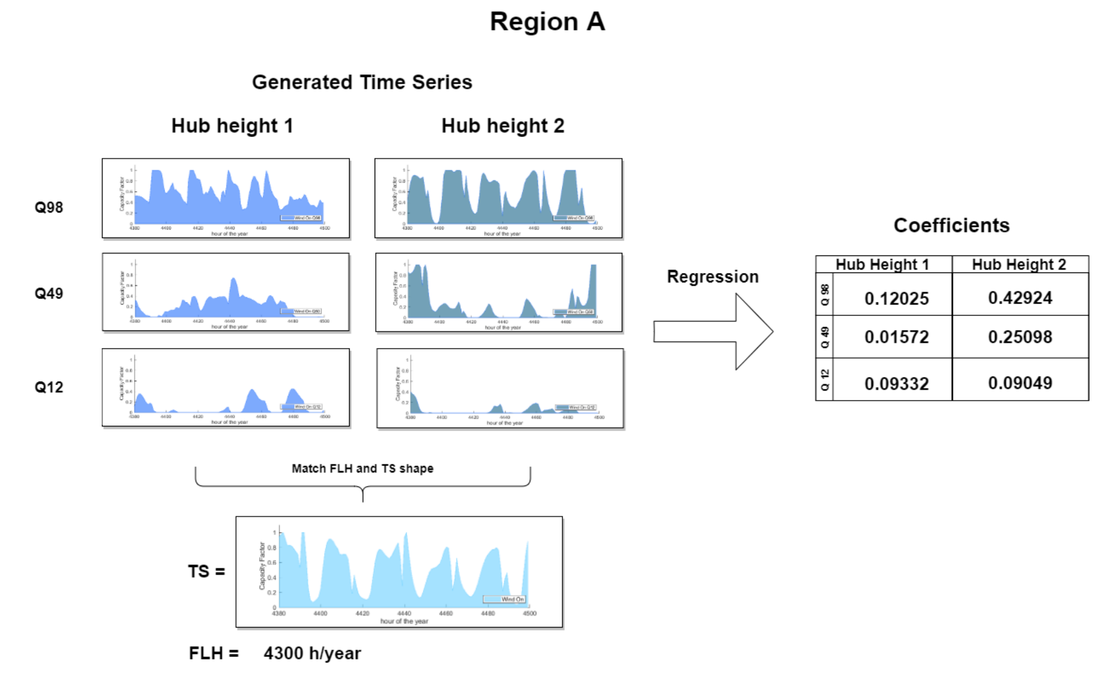


Fig. 8: Regression Coefficients - Process example Region A

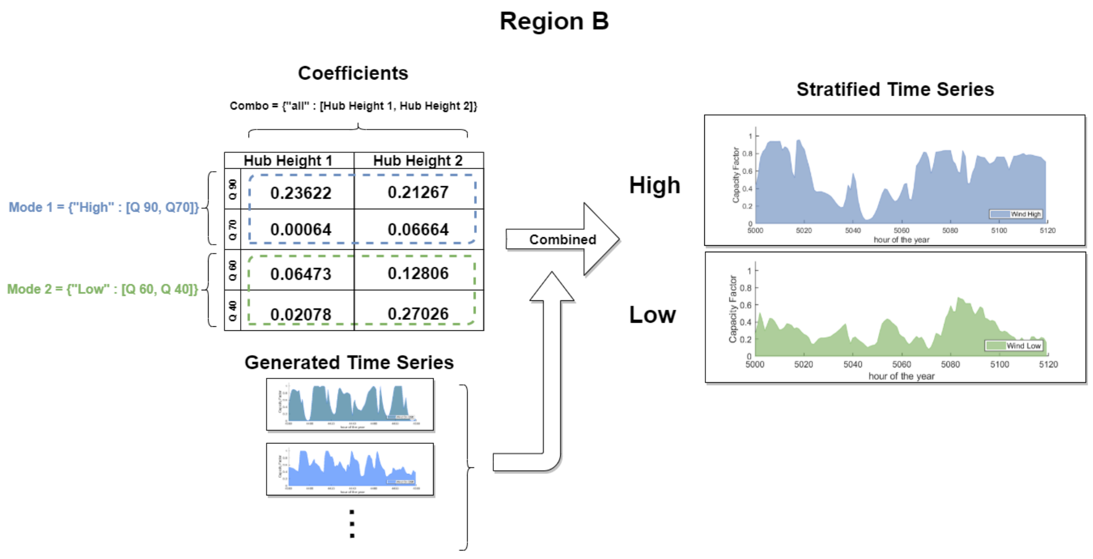


Fig. 9: Stratified Time Series - Process example Region A

Chapter 2

Theory

2.1 Solar

2.1.1 Solar Angles

While the output power of the Sun is usually considered as a constant, the amount of power arriving at the Earth's surface varies according to the time, location, weather, and relative position of the Earth with respect to the Sun. Besides, the available data needed for a solar power calculation is usually given for a horizontal surface and most of the PV systems are placed in a tilted position. Therefore, it is necessary to calculate a set of parameters describing the Sun's relative position with respect to the position of the system being irradiated. These parameters are calculated for points located at the center of every pixel (with high resolution) within the extension under analysis and for every hour of the year.

Declination Angle δ

This angle varies during the year due to the tilt of the Earth's axis, which is 23.45° tilted, so the declination ranges between -23.4° and 23.45° through the year. The declination could be interpreted as the latitude where the Sun's rays perpendicularly strike the Earth's surface at solar noon. This value is the same for all the locations within the globe for a given day and is calculated as follows [11]:

$$\delta = \arcsin\left(0.3978\sin\left(\frac{2\pi N}{365.25} - 1.4 + 0.0355\sin\left(\frac{2\pi N}{365.25} - 0.0489\right)\right)\right)$$

where N is the day of the year.

Solar Time

The time is important to define the position of the Sun in the sky. However, it is easier to use the time if it is converted into solar time. To do so, a few corrections are needed. The equation of time is an empirical equation which corrects the error caused by the axial tilt of the Earth and the eccentricity of its orbit [11]:

$$EOT = -0.128\sin\left(\frac{360}{365.25}N - 2.8\right) - 0.165\sin\left(\frac{720}{365.25}N + 19.7\right)$$

When the time is given in GMT, as it is for this model, it is also necessary to take into account the longitude of the location, hence the time correction:

$$TC = EOT + longitude/15$$

where the factor 15 accounts for the geographical span of each time zone (15° of longitude). With this correction, the local solar time is calculated as:

$$LST = T_{GMT} + TC$$

where LST is the local solar time. An even more apporpriate time measure for solar calculations is the hour angle ω This converts hours into degrees which indicate how the Sun moves in the sky relatively to the Earth, where the solar noon is 0°, the anlges after the noon positive, and before the noon negative.

$$\omega = 15(LST - 12)$$

Another important quantity is the duration of the day, which is delimited by the sunrise and the sunset. The sunrise and sunset have the same value, however, the sunrise is considered negative and the sunset positive. They depend on the day of the year and the location on the Earth (denoted by the declination and the latitude ϕ respectively) and they are calculated for a horizontal surface as follows [7]:

$$\omega_s = \arccos(-\tan\phi\tan\delta)$$

Due to self-shading, a tilted plane might be exposed to different sunrise and sunset's values. Also, if the plane is not facing the equator, the sunrise and sunset angles will be numerically different for such surfaces. The following equations consider this orientation changes for the sunrise and sunset values of a tilted plane [7].

$$a = \frac{\cos\phi}{\tan\beta} + \sin\phi$$

$$b = \tan\delta\cos\phi\cos\gamma - \frac{\sin\phi}{\tan\beta}$$

$$\omega_s' = \cos\left[\frac{ab \pm \sin\gamma\sqrt{a^2 - b^2 + \sin^2\gamma}}{a^2 + \sin^2\gamma}\right]$$

where γ is the azimuthal orientation of the panel and β is the tilt of the panel (for this model, chosen as the optimal tilt according to the latitude). These equations might give higher values than the real sunrise and sunset values. This would imply that the Sun rises over the tilted plane before it has risen over the horizon or that when the Sun sets, there is still light striking the plane. As this is wrong, the sunrise and sunset values for a horizontal plane must be compared with the values for a tilted plane and the lower values (for both sunrise and sunset) must be selected.

$$\omega_0 = \min(\omega_s, \omega_s')$$

Incidence Angle θ

As stated before, PV panels are not normally parallel to the Earth's surface, so it is necessary to calculate the incidence angle of the Sun's rays striking the surface of the panel. Nevertheless, a set of angles must be calculated in order to calculate the incidence angle.

The elevation angle α or altitude angle measures the angular distance between the Sun and the horizon. It ranges from 0° at the sunrise to 90° at the noon (the value at the noon varies depending on the day of the year) [12].

$$\alpha = \arcsin[\sin\delta\sin\phi + \cos\delta\cos\phi\cos\omega]$$

The azimuth angle A_z is an angular measurement of the horizontal position of the Sun. It could be seen as a compass direction with 0° to the North and 180° to the South. The range of values of the azimuth angle varies over the year, going from 90° at the sunrise to 270° at the sunset during the equinoxes. The equation for the azimuth depends on the time of the day. For the solar morning, it is [12]:

$$Az_{am} = \arccos\left(\frac{\sin\delta\cos\phi - \cos\delta\sin\phi\cos\omega}{\cos\alpha}\right)$$

and for the afternoon:

$$Az_{pm} = 360 - Az_{am}$$

With the already calculated angles, it is possible to calculate the incidence angle, which is the angle between the surface's normal and the Sun's beam radiation [10]:

$$\begin{aligned}\theta_i = &\arccos(\sin\delta\sin\phi\cos\beta \\ &- \sin\delta\cos\phi\sin\beta\cos\gamma \\ &+ \cos\delta\cos\phi\cos\beta\cos\omega \\ &+ \cos\delta\sin\phi\sin\beta\cos\gamma\cos\omega \\ &+ \cos\delta\sin\beta\sin\gamma\sin\omega)\end{aligned}$$

Tracking

When one-axis tracking is active, the tilt angle β and the azimuthal orientation γ of the panel change constantly as the panel follows the sun. In this model a tilted on-axis tracking with east-west tracking is considered. The rotation of the plane around the axis is deffned by the rotation angle R, it is calculated in order to achieve the smallest incidence angle for the plane by the following equations [13]:

$$X = \frac{-\cos\alpha\sin(A_z-\gamma_a)}{-\cos\alpha\cos(A_z-\gamma_a)\sin\beta_a+\sin\alpha\cos\beta_a}$$

$$\Psi = \begin{cases} 0, & \text{if } X=0, \text{ or if } X>0\wedge(A_z-\gamma_a)>0, \text{ or if } X<0\wedge(A_z-\gamma_a)<0 \\ 180, & \text{if } X<0\wedge(A_z-\gamma_a)>0 \\ -180, & \text{if } X>0\wedge(A_z-\gamma_a)<0 \end{cases}$$

for the previous equations, β_a and γ_a are considered as the tilt and azimuthal orientation of the tracking axis respectively. The variable Ψ places R in the correct trigonometric quadrant. For the selection of Ψ, the difference $(A_z-\gamma_a)$ must be considered as the angular displacement with the result within the range of -180°to 180°. Once the rotation angle is calculated, the tilt and azimuthal orientation of the panel are calculated as follows:

$$\beta = \arccos(\cos R\cos\beta_a)$$

$$\gamma = \begin{cases} \gamma_a + \arcsin\left(\frac{\sin R}{\sin\beta}\right), & \text{for } \beta_a\neq 0, -90\leq R\leq 90 \\ \gamma_a - 180 - \arcsin\left(\frac{\sin R}{\sin\beta}\right), & \text{for } -180\leq R<-90 \\ \gamma_a + 180 - \arcsin\left(\frac{\sin R}{\sin\beta}\right), & \text{for } 90<R\leq -90 \end{cases}$$

Then the incidence angle is calculated using the new β and γ angles. For two-axis tracking the beta angle is considered as the complementary angle to the altitude angle while the azimuthal orientation angle γ is considered as $A_z - 180$.

2.1.2 Solar Power

To calculate the solar power we must start with the solar constant. However, the irradiance striking the top of the Earth's atmosphere (TOA) varies over the year. This is due to the eccentricity of the Earth's orbit and its tilted axis. The TOA is calculated according to the following equation [10]:

$$TOA = G_{sc}\left[1+0.03344\cos\left(\frac{2\pi N}{365.25}-0.048869\right)\right]\sin\alpha$$

where G_{sc} is the solar constant (1367 W/m2) and N is the day of the year. This equation escalates the solar constant by multiplying it with a factor related to the eccentricity of the Earth's orbit and the sinus of the altitude angle of the Sun to finally get the extraterrestrial horizontal radiation.

To calculate the amount of horizontal radiation at the surface of the Earth, the attenuation caused by the atmosphere must be considered. A way to measure this attenuation is by using the clearness index k_t. This value is the ratio between the extraterrestrial horizontal radiation and the radiation striking the Earth's surface:

$$k_t = \frac{GHI_{M2}}{TOA_{M2}}$$

where GHI_{M2} and TOA_{M2} are the global horizontal irradiance and the top of the atmosphere radiation extracted from MERRA-2 data. Furthermore, the GHI is made-up by diffuse and beam radiation:

$$GHI = G_b + G_d$$

where G_b is the beam radiation, which is the solar radiation that travels directly to the Earth's surface without any scattering in the atmosphere, and G_d stands forfor the diffuse radiation, the radiation that comes to a surface from all directions as its trajectory is changed by the atmosphere. These two components have different contributions to the total irradiance on a tilted surface, so it is necessary to distinguish between them. This can be done using the correlation of Erbs et al [4], which calculates the ratio R of the beam and diffuse radiation as a function of the clearness index.

$$R = \begin{cases} 1 - 0.09k_t, & \text{for } k_t \leq 0.22 \\ 0.9511 - 0.1604k_t + 4.388k_t^2 - 16.638k_t^3 + 12.336k_t^4, & \text{for } 0.22 > k_t \leq 0.8 \\ 0.165, & \text{for } k_t > 0.8 \end{cases}$$

Furthermore, diffuse radiation could be divided into more components. The HDKR model [3][10], developed by Hay, Davies, Klucher, and Reindl in 1979 assumes isotropic diffuse radiation, which means that the diffuse radiation is uniformly distributed across the sky. However, it also considers a higher radiation intensity around the Sun, the circumsolar diffuse radiation, and a horizontal brightening correction. To use the HDKR model, some factors must be defined first [10]. The ratio of incident beam to horizontal beam:

$$R_b = \frac{\cos\theta_i}{\sin\alpha}$$

The anisotropy index for forward scattering circumsolar diffuse irradiance:

$$A_i = (1 - R)k_t$$

The modulating factor for horizontal brightening correction:

$$f = \sqrt{1 - R}$$

Then the total radiation incident on the surface is calculated with the next equations:

$$GHI = k_t TOA$$

$$G_T = GHI\Big[(1 - R + RA_i)R_b + R(1 - A_i)\Big(\frac{1+\cos\beta}{2}\Big)\Big(1 + f\sin^3\frac{\beta}{2}\Big) + \rho_g\Big(\frac{1-\cos\beta}{2}\Big)\Big]$$

Where ρ_g is the ground reflectance or albedo and it is related to the land use type of the location under analysis. The first term of this equation corresponds to the beam and circumsolar diffuse radiation, the second to the isotropic and horizon brightening radiation, and the last one to the incident ground-reflected radiation.

2.1.3 PV

Temperature losses

In a PV panel, not all the radiation absorbed is converted into current. Some of this radiation is dissipated into heat. Solar cells, like all other semiconductors, are sensitive to temperature. An increase of temperature results in a reduction of the band gap of the solar cell which is translated into a reduction of the open circuit voltage. The overall effect is a reduction of the power output of the PV system. To calculate the power loss of a solar cell it is necessary to know its temperature. This can be expressed as a function of the incident radiation and the ambient temperature [9]:

$$T_{cell} = T_{amb} + kG_T$$

where T_{amb} is the ambient temperature and k is the Ross coefficient, which depends on the characteristics related to the module and its environment. It is defined based on the land use type of the region where the panel is located. With the temperature of the panel, the fraction of the irradiated power which is lost can be calculated as:

$$Loss_T = (T_{cell} - T_r)T_k$$

where T_r is the rated temperature of the module according to standard test conditions and T_k is the heat loss coefficient. Both values are usually given on the data sheets of the PV modules.

PV Capacity Factor Calculation

It is the ratio of the actual power output to the theoretical maximum output which is normally considered as $1000W/m^2$. The temperature loss is also considered for this calculation:

$$CF_{PV} = \frac{G_T(1 - Loss_T)}{1000}$$

Ground coverage ratio (GCR)

It is also important to consider the area lost due to the space between the modules or due to the modules shading adjacent modules. This is done with the GCR which is the ratio of the module area to the total ground area.

$$GCR = \frac{1}{\cos\beta + |\cos A_z| \cdot \left(\frac{\sin\beta}{\tan\alpha}\right)}$$

2.1.4 CSP

For its popularity and long development history, the parabolic trough technology was chosen to model Concentrated Solar power.

Convection Losses

The receiver of parabolic troughs are kept in a vacuum glass tube to prevent convection as much as possible. Radiative heat losses are still present and ultimatly results in convective losses between the glass tube and the air. These heat losses are increased when wind is blowing around the receiver. The typical heat losses for a receiver can be estimated through the following empirical equation [3]:

$$Q_{Loss} = A_r(U_{L_{cst}} + U_{L_{Wind}} \cdot V_{Wind}^{0.6})(T_i - T_a)$$

where A_r is the outer area of the receiver, $U_{L_{cst}}$ correspond to a loss coefficient at zero wind speed, $U_{L_{Wind}}$ is a loss coefficient dependent on the wind speed V_{Wind}, T_i is the average heat transfer fluid temperature, and T_a is the ambient temperature.

Typical values for the $U_{L_{cst}}$ and $U_{L_{Wind}}$ are 1.06 kW/m^2K and 1.19 $kW/(m^2K(m/s)^{0.6})$ respectively

Flow Losses

Flow loss coefficient or heat removal factor F_r is the ratio between the actual heat transfer to the maximum heat transfer possible between the receiver and the heat transfer fluid (HTF). These losses result from the difference between the temperature of the receiver and the temperature of the HTF and are dependent on the heat capacity and the flow rate of the HTF. A typical value for parabolic toughts is 95%.

CSP Capacity Factor Calculation

The capacity factor of a solar field is the ratio of the actual useful heat collected to the theoretical maximum heat output of 1000 W/m^2. It is given by the formula:

$$CF_{csp} = \frac{F_r(S - Q_{Loss})}{1000}$$

Where S is the component of the DNI captured by the collector at an angle (based on one axis traking), Q_{Loss} is the heat convection losses, and F_r is the heat removal factor.

2.2 Wind

2.2.1 Wind Speed

In order to use a single value for the following wind power calculations, a norm is calculated as if both variables were vectors, so the absolute velocity is:

$$W_{50M} = \sqrt{V50M^2 + U50M^2}$$

However, the wind velocities extracted from MERRA-2 are given for a height of 50 meters, which does not correspond to the hub height of the wind turbines; therefore, they must be extrapolated.

2.2.2 Wind Shear

While the wind is hardly affected by the Earth's surface at a height of about one kilometer, at lower heights in the atmosphere the friction of the Earth's surface reduces the speed of the wind [2]. One of the most common expressions describing this phenomenon is the Hellmann exponential law, which correlates the wind speed at two different heights [6].

$$v = v_0 \left(\frac{H}{H_0} \right)^{\alpha}$$

Where v is the wind speed at a height H, v_0 is the wind speed at a height H_0 and α is the Hellmann coefficient, which is a function of the topography and air stability at a specific location.

2.2.3 Wind Power

The wind turbines convert the kinetic energy of the air into torque. The power that a turbine can extract from the wind is described by the following expression [8]:

$$P = \frac{1}{2} \rho A v^3 C_p$$

where ρ is the density of the air, v is the speed of the wind and C_p is the power coefficient. As it is shown in the previous equation, the energy in the wind varies proportionally to the cube of the wind's speed. Therefore, the power output of wind turbines is normally described with cubic power curves. However, there are some regions within those curves which have special considerations.

Cut-in wind speed

The wind turbines start running at a wind speed between 3 and 5 m/s to promote torque and acceleration. Therefore, there is no power generation before this velocity.

Rated Wind Speed

The power output of a wind turbine rises with the wind speed until the power output reaches a limit defined by the characteristics of the electric generator. Beyond this wind speed, the design and the controllers of the wind turbine limit the power output so this does not increase with further increases of wind speed.

Cut-out wind speed

The wind turbines are programmed to stop at wind velocities above their rated maximum wind speed(usually around 25 m/s) to avoid damage to the turbine and its surroundings, so there is no power generation after this velocity.

Wind Onshore and Offshore Capacity factor

Finally, the capacity factors, which are the ratios of the actual power output to the theoretical maximum output (rated wind speed), are calculated according to the previously presented regions:

$$CF = \begin{cases} \dfrac{W_{hub}^3 - W_{in}^3}{W_r^3 - W_{in}^3}, & \text{for } W_{in} < W_{hub} < W_r \\ 1, & \text{for } W_r \leq W_{hub} \leq W_{out} \\ 0, & \text{for } W_{hub} < W_{in} | W_{hub} > W_{out} \end{cases}$$

where W_{in} is the cut-out wind speed, and W_r is the rated wind speed. The area is not included in the previous equations as it does not change in both generation states (actual and theoretical maximum power). While the density could vary for both states, the overall impact of a change in density is negligible compared to the wind speed and therefore is not included in the calculation.

Chapter 3

Implementation

You can run the code by typing:

```
$ python runme.py
```

The script `runme.py` calls the main functions of the code, which are explained in the following sections.

3.1 initialization.py

`lib.initialization.`**`initialization`**`()`

This function reads the user-defined parameters and paths from `config.py`, then adds additional parameters related to the shapefiles. First, it saves the spatial scope of the problem. Then, it distinguishes between countries, exclusive economic zones and subregions. For each one of them, it saves the geodataframes, the number of features, and the coordinates of the bounding boxes of each feature. Finally, it saves the number of rows and columns in the low and righ resolution, and a georeference dictionary used for saving tif files.

Returns The updated dictionaries param and paths.

Return type tuple(dict, dict)

3.2 input_maps.py

`lib.input_maps.`**`generate_area`**`(`*paths*, *param*`)`

This function retreives the coordinates of the spatial scope and computes the pixel area gradient of the corresponding raster.

Parameters

- **`paths`** (*`dict`*) – Dictionary of dictionaries containing the path to the output file.
- **`param`** (*`dict`*) – Dictionary of dictionaries containing spatial scope coordinates and desired resolution.

Returns The mat file for AREA is saved in its respective path, along with its metadata in a JSON file.

Return type None

`lib.input_maps.`**`generate_bathymetry`**`(`*paths*, *param*`)`

This function reads the global map of bathymetry, resizes it, and creates a raster out of it for the desired scope. The values are in meter (negative in the sea).

Parameters

- **paths** (*dict*) – Dictionary including the paths to the global bathymetry raster *Bathym_global* and to the output path *BATH*.
- **param** (*dict*) – Dictionary including the desired resolution, the coordinates of the bounding box of the spatial scope, and the georeference dictionary.

Returns The tif file for *BATH* is saved in its respective path, along with its metadata in a JSON file.

Return type None

`lib.input_maps.`**`generate_buffered_population`**(*paths*, *param*)

This function reads the land use raster, identifies urban areas, and excludes pixels around them based on a user-defined buffer *buffer_pixel_amount*. It creates a masking raster of boolean values (0 or 1) for the scope. Zero means the pixel is excluded, one means it is suitable. The function is useful in case there is a policy to exclude renewable energy projects next to urban settlements.

Parameters

- **paths** (*dict*) – Dictionary including the path to the land use raster for the scope, and to the output path BUFFER.
- **param** (*dict*) – Dictionary including the user-defined buffer (buffer_pixel_amount), the urban type within the land use map (type_urban), and the georeference dictionary.

Returns The tif file for BUFFER is saved in its respective path, along with its metadata in a JSON file.

Return type None

`lib.input_maps.`**`generate_landsea`**(*paths*, *param*)

This function reads the shapefiles of the countries (land areas) and of the exclusive economic zones (sea areas) within the scope, and creates two rasters out of them.

Parameters

- **paths** (*dict*) – Dictionary including the paths *LAND* and *EEZ*.
- **param** (*dict*) – Dictionary including the geodataframes of the shapefiles, the number of features, the coordinates of the bounding box of the spatial scope, and the number of rows and columns.

Returns The tif files for *LAND* and *EEZ* are saved in their respective paths, along with their metadata in JSON files.

Return type None

`lib.input_maps.`**`generate_landuse`**(*paths*, *param*)

This function reads the global map of land use, and creates a raster out of it for the desired scope. There are 17 discrete possible values from 0 to 16, corresponding to different land use classes. See `config.py` for more information on the land use map.

Parameters

- **paths** (*dict*) – Dictionary including the paths to the global land use raster *LU_global* and to the output path *LU*.
- **param** (*dict*) – Dictionary including the desired resolution, the coordinates of the bounding box of the spatial scope, and the georeference dictionary.

Returns The tif file for *LU* is saved in its respective path, along with its metadata in a JSON file.

Return type None

`lib.input_maps.`**`generate_maps_for_scope`**(*paths*, *param*)

This function calls the individual functions that generate the maps for the geographic scope.

Parameters

- **paths** (*dict*) – Dictionary including the paths.
- **param** (*dict*) – Dictionary including the user preferences.

Returns The maps are saved directly in the desired paths.

Return type None

`lib.input_maps.`**`generate_population`**(*paths*, *param*)

This function reads the global map of population density, resizes it, and creates a raster out of it for the desired scope. The values are in population per pixel.

Parameters

- **paths** (*dict*) – Dictionary including the paths to the global population raster *Pop_global* and to the output path *POP*.
- **param** (*dict*) – Dictionary including the desired resolution, the coordinates of the bounding box of the spatial scope, and the georeference dictionary.

Returns The tif file for *POP* is saved in its respective path, along with its metadata in a JSON file.

Return type None

`lib.input_maps.`**`generate_protected_areas`**(*paths*, *param*)

This function reads the shapefile of the globally protected areas, adds an attribute whose values are based on the dictionary of conversion (protected_areas) to identify the protection category, then converts the shapefile into a raster for the scope. The values are integers from 0 to 10.

Parameters

- **paths** (*dict*) – Dictionary including the paths to the shapefile of the globally protected areas, to the landuse raster of the scope, and to the output path PA.
- **param** (*dict*) – Dictionary including the dictionary of conversion of protection categories (protected_areas).

Returns The tif file for PA is saved in its respective path, along with its metadata in a JSON file.

Return type None

`lib.input_maps.`**`generate_slope`**(*paths*, *param*)

This function reads the topography raster for the scope, and creates a raster of slope out of it. The slope is calculated in percentage, although this can be changed easily at the end of the code.

Parameters

- **paths** (*dict*) – Dictionary including the paths to the topography map of the scope *TOPO* and to the output path *SLOPE*.
- **param** (*dict*) – Dictionary including the desired resolution, the coordinates of the bounding box of the spatial scope, and the georeference dictionary.

Returns The tif file for SLOPE is saved in its respective path, along with its metadata in a JSON file.

Return type None

`lib.input_maps.`**`generate_subregions`**(*paths*, *param*)

This function reads the shapefile of the subregions within the scope, and creates a raster out of it.

Parameters

- **paths** (*dict*) – Dictionary including the paths *SUB*, *LAND*, *EEZ*.
- **param** (*dict*) – Dictionary including the geodataframe of the shapefile, the number of features, the coordinates of the bounding box of the spatial scope, and the number of rows and columns.

Returns The tif file for *SUB* is saved in its respective path, along with its metadata in a JSON file.

Return type None

`lib.input_maps.`**`generate_topography`**(*paths*, *param*)

This function reads the tiles that make the global map of topography, picks those that lie completely or partially in the scope, and creates a raster out of them for the desired scope. The values are in meter.

Parameters

- **paths** (*dict*) – Dictionary including the paths to the tiles of the global topography raster *Topo_tiles* and to the output path *TOPO*.
- **param** (*dict*) – Dictionary including the desired resolution, the coordinates of the bounding box of the spatial scope, and the georeference dictionary.

Returns The tif file for *TOPO* is saved in its respective path, along with its metadata in a JSON file.

Return type None

`lib.input_maps.`**`generate_weather_files`**(*paths*, *param*)

This function reads the daily NetCDF data (from MERRA-2) for SWGDN, SWTDN, T2M, U50m, and V50m, and saves them in matrices with yearly time series with low spatial resolution. Depending on the *MERRA_correction* parameter this function will also call clean_weather_data() to remove data outliers. This function has to be run only once.

Parameters

- **paths** (*dict*) – Dictionary including the paths to the MERRA-2 input files *MERRA_IN*, and to the desired output locations for *T2M*, *W50M* and *CLEARNESS*.
- **param** (*dict*) – Dictionary including the year, the spatial scope, and the MERRA_correction parameter.

Returns The files T2M.mat, W50M.mat, and CLEARNESS.mat are saved directly in the defined paths, along with their metadata in JSON files.

Return type None

3.3 potential.py

`lib.potential.`**`calc_FLH_solar`**(*hours*, *args*)

This function computes the full-load hours for all valid pixels specified in *ind_nz* in *param*. Due to parallel processing, most of the inputs are collected in the list *args*.

Parameters

- **hours** (*numpy array*) – Filtered day hour ranks in a year (from 0 to 8759).
- **args** (*list*) – List of arguments: * *param* (dict): Dictionary including multiple parameters such as the status bar limit, the name of the region, and others for calculating the hourly capacity factors. * *tech* (str): Name of the technology. * *rasterData* (dict): Dictionary of numpy arrays containing land use types, Ross coefficients, albedo coefficients, and wind speed correction for every point in *reg_ind*. * *merraData* (dict): Dictionary of numpy arrays containing the weather data for every point in *reg_ind*.

Return FLH Full-load hours over the year for the technology.

Return type numpy array

`lib.potential.`**`calc_FLH_wind`**(*hours*, *args*)

This function computes the full-load hours for all valid pixels specified in *ind_nz* in *param*. Due to parallel processing, most of the inputs are collected in the list *args*.

Parameters

- **hours** (*numpy array*) – Hour ranks in a year (from 0 to 8759).
- **args** (*list*) – List of arguments: * *param* (dict): Dictionary including multiple parameters such as the status bar limit, the name of the region, and others for calculating the hourly capacity factors. * *tech* (str): Name of the technology. * *rasterData* (dict): Dictionary of numpy arrays containing land use types, Ross coefficients, albedo coefficients, and wind speed correction for every point in *reg_ind*. * *merraData* (dict): Dictionary of numpy arrays containing the weather data for every point in *reg_ind*.

Return FLH Full-load hours over the year for the technology.

Return type numpy array

`lib.potential.`**`calc_gcr`**(*Crd_all*, *m_high*, *n_high*, *res_desired*, *GCR*)

This function creates a GCR weighting matrix for the desired geographic extent. The sizing of the PV system is conducted on a user-defined day for a shade-free exposure to the sun during a given number of hours.

Parameters

- **Crd_all** (*list*) – Desired geographic extent of the whole region (north, east, south, west).
- **m_high** (*int*) – Number of rows.
- **n_high** (*int*) – Number of columns.
- **res_desired** (*list*) – Resolution of the high resolution map.
- **GCR** (*dict*) – Dictionary that includes the user-defined day and the duration of the shade-free period.

Return A_GCR GCR weighting raster.

Return type numpy array

`lib.potential.`**`calculate_full_load_hours`**(*paths*, *param*, *tech*)

This function calculates the yearly FLH for a technology for all valid pixels in a spatial scope. Valid pixels are land pixels for WindOn, PV and CSP, and sea pixels for WindOff. The FLH values are calculated by summing up hourly capacity factors.

Parameters

- **paths** (*dict*) – Dictionary of dictionaries containing the paths to the input weather data, land, sea and land use rasters, and correction rasters.
- **param** (*dict*) – Dictionary of dictionaries containing the spatial scope, and technology and computation parameters.
- **tech** (*str*) – Technology under study.

Returns The raster of FLH potential is saved as mat and tif files, along with the json metadata file.

Return type None

`lib.potential.`**`get_merra_raster_data`**(*paths*, *param*, *tech*)

This function returns a tuple of two dictionaries containing weather and correction rasters for specified technology.

Parameters

- **paths** (*dict*) – Dictionary of dictionaries containing the paths to the input weather and raster data.
- **param** (*dict*) – Dictionary of dictionaries containing land use, Ross coefficients, albedo, and Hellmann coefficients.
- **tech** (*str*) – Technology under study.

Return (merraData, rasterData) Dictionaries for the weather data and for the correction data.

Return type tuple (dict, dict)

`lib.potential.`**`mask_potential_maps`**(*paths*, *param*, *tech*)

This function first reads the rasters for land use, slope, bathymetry, and protected areas for the scope. Based on user-defined assumptions on their suitabilities, it generates a masking raster to exclude the unsuitable pixels. Both the mask itself and the masked potential rasters can be saved.

Parameters

- **paths** (*dict*) – Dictionary of dictionaries containing user-defined parameters for masking, protected areas, and landuse.
- **param** (*dict*) – Dictionary of dictionaries containing the paths to the land use, protected areas, slope and bathymetry, in addition to output paths.
- **tech** (*str*) – Technology under study.

Returns The files for the mask and the masked FLH are saved as tif and mat files, along with their metadata json files.

Return type None

`lib.potential.`**`report_potentials`**(*paths*, *param*, *tech*)

This function reads the FLH files and the subregion shapefile, and creates a CSV file containing various statistics:

- Available number of pixels, before and after masking
- Available area in in km^2
- FLH mean, median, max, min values, before and after masking
- FLH standard deviation after masking
- Power Potential in GW, before and after weighting
- Energy Potential in TWh in total, after weighting, and after masking and weighting
- Sorted sample of FLH values for each region

Parameters

- **paths** (*dict*) – Dictionary of dictionaries containing the paths to FLH, Masking, Weighting, and Area rasters.
- **param** (*dict*) – Dictionary of dictionaries containing technology parameters and sampling parameters.
- **tech** (*str*) – Technology under study.

Returns The CSV files with the report and the sorted FLH are saved directly in the desired paths, along with the corresponding metadata in JSON files.

Return type None

`lib.potential.`**`sampled_sorting`**(*Raster*, *sampling*)

This function returns a list with a defined length of sorted values sampled from a numpy array.

Parameters

- **Raster** (*numpy array*) – Input raster to be sorted.
- **sampling** (*int*) – Number of values to be sampled from the raster, defines length of output list.

Return s List of sorted values sampled from *Raster*.

Return type list

`lib.potential.`**`weight_potential_maps`**(*paths*, *param*, *tech*)

This function weights the power potential by including assumptions on the power density and the available area. Therefore, it reads the rasters for land use and protected areas for the scope. Based on user-defined assumptions on their availabilities, it generates a weighting raster to exclude the unsuitable pixels. Both the weight itself and the weighted potential rasters can be saved.

Parameters

- **`paths`** (*`dict`*) – Dictionary of dictionaries containing user-defined parameters for weighting, protected areas, and landuse.
- **`param`** (*`dict`*) – Dictionary of dictionaries containing the paths to the land use, protected areas, area, in addition to output paths.
- **`tech`** (*`str`*) – Technology under study.

Returns The files for the weight and the weighted FLH are saved as tif and mat files, along with their metadata json files.

Return type None

3.4 time_series.py

`lib.time_series.`**`calc_TS_solar`**(*hours*, *args*)

This function computes the hourly PV and CSP capacity factor for the desired quantiles.

Parameters

- **`hours`** (*`numpy array`*) – Hour ranks of the year (from 0 to 8759).
- **`args`** (*`list`*) – List of arguments:
  - *param* (dict): Dictionary including multiple parameters such as the status bar limit, the name of the region, and others for calculating the hourly capacity factors.
  - *tech* (str): Name of the technology.
  - *rasterData* (dict): Dictionary of numpy arrays containing land use types, Ross coefficients, albedo coefficients, and wind speed correction for every point in *reg_ind*.
  - *merraData* (dict): Dictionary of numpy arrays containing the weather data for every point in *reg_ind*.

Return TS Array of time series for the desired quantiles for each subregion.

Return type numpy array

`lib.time_series.`**`calc_TS_wind`**(*hours*, *args*)

This function computes the hourly onshore and offshore wind capacity factor for desired quantiles.

Parameters

- **`hours`** (*`numpy array`*) – Hour ranks of the year (from 0 to 8759).
- **`args`** (*`list`*) – List of arguments:
  - *param* (dict): Dictionary including multiple parameters such as the status bar limit, the name of the region, and others for calculating the hourly capacity factors.
  - *tech* (str): Name of the technology.
  - *rasterData* (dict): Dictionary of numpy arrays containing the wind speed correction for every point in *reg_ind*.
  - *merraData* (dict): Dictionary of numpy arrays containing the weather data for every point in *reg_ind*.

Return TS Array of time series for the desired quantiles for each subregion.

Return type numpy array

`lib.time_series.`**`combinations_for_time_series`**(*paths*, *param*, *tech*)

This function reads the list of generated regression coefficients for different hub heights and orientations, compares it to the user-defined modes and combos and returns a list of lists containing all the available combinations. The function will return a warning if the user input and the available time series are not congruent.

Parameters

- **`paths`** (*`dict`*) – Dictionary of dictionaries containing the paths to the regression output folder.
- **`param`** (*`dict`*) – Dictionary of dictionaries containing the year, the user defined combos, and subregions name.
- **`tech`** (*`str`*) – Technology under study.

Return combinations List of combinations of settings to be used in stratified time series.

Return inputfiles List of regression outputs to be used in generating the stratified time series.

Return type tuple (list, list)

Raises

- **`No coefficients`** – If regression coefficients are not available, a warning is raised.
- **`Missing coefficients`** – If regression coefficients are missing based on user-defined combos and mode, a warning is raised.

`lib.time_series.`**`find_representative_locations`**(*paths*, *param*, *tech*)

This function reads the masked FLH raster and finds the coordinates and indices of the pixels for the user-defined quantiles for each region. It creates a shapefile containing the position of those points for each region, and two MAT files with their coordinates and indices.

Parameters

- **`paths`** (*`dict`*) – Dictionary of dictionaries containing path values for FLH MAT files, region statistics, and output paths.
- **`param`** (*`dict`*) – Dictionary of dictionaries containing the user-defined quantiles, FLH resolution, and spatial scope.
- **`tech`** (*`str`*) – Technology under study.

Returns The shapefile with the locations and the two MAT files for the coordinates and the indices are saved directly in the given paths, along with their corresponding metadata in JSON files.

Return type None

`lib.time_series.`**`generate_time_series_for_regions`**(*paths*, *param*, *tech*)

This function reads the coefficients obtained from the regression function as well as the generated time series for the combinations of hub heights / orientations and quantiles, to combine them according to user-defined *modes* (quantile combination) and *combos* (hub heights / orientation combinations) and saves the results (time series) in a CSV file.

Parameters

- **`paths`** (*`dict`*) – Dictionary of dictionaries containing the paths to the regression coefficients and the time series.
- **`param`** (*`dict`*) – Dictionary of dictionaries containing the list of subregions, the modes, and the combos.
- **`tech`** (*`str`*) – Technology under study.

Returns The stratified time series for each region, mode, and combo are saved directly in the given path, along with the metadata in a JSON file.

Return type None

`lib.time_series.`**`generate_time_series_for_representative_locations`**(*paths*, *param*, *tech*)

This function generates yearly capacity factor time-series for the technology of choice at quantile locations generated in find_locations_quantiles. The timeseries are saved in CSV files.

Parameters

- **`paths`** (*`dict`*) – Dictionary of dictionaries containing paths to coordinate and indices of the quantile locations.
- **`param`** (*`dict`*) – Dictionary of dictionaries containing processing parameters.
- **`tech`** (*`str`*) – Technology under study.

Returns The CSV file with the time series for all subregions and quantiles is saved directly in the given path, along with the corresponding metadata in a JSON file.

Return type None

`lib.time_series.`**`generate_time_series_for_specific_locations`**(*paths*, *param*, *tech*)

This function generates yearly capacity factor time-series for the technology of choice at user defined locations. The timeseries are saved in CSV files.

Parameters

- **`paths`** (*`dict`*) – Dictionary of dictionaries containing paths output desired locations.
- **`param`** (*`dict`*) – Dictionary of dictionaries containing processing parameters, and user-defined locations.
- **`tech`** (*`str`*) – Technology under study.

Returns The CSV file with the time series for all subregions and quantiles is saved directly in the given path, along with the corresponding metadata in a JSON file.

Return type None

Raises

- **`Point locations not found`** – Is raised when the dictionary containing the points names and locations is empty.
- **`Points outside spatial scope`** – Some points are not located inside of the spatial scope, therefore no input maps are available for the calculations

3.5 regression.py

`lib.regression.`**`check_regression_model`**(*paths*, *tech*)

This function checks the regression model parameters for nan values, and returns the FLH and TS model dataframes. If missing values are present in the input CSV files, the users are prompted if they wish to continue or can modify the corresponding files.

Parameters

- **`paths`** (*`dict`*) – Dictionary of dictionaries containing the paths to the FLH and TS model regression CSV files.
- **`tech`** (*`str`*) – Technology under study.

Return (FLH, TS_reg) Tuple of pandas dataframes for FLH and TS.

Return type Tuple of pandas dataframes `lib.regression.`**`clean_FLH_regression`**(*paths*, *param*)

This function creates a CSV file containing the model FLH used for regression. If the region is present in the IRENA database, then the FLH are extracted directly from there. In case it is not present, a place holder for the regions is written in the csv file and it is the user's responsibility to fill in an appropriate value. The function will warn the user, and print all regions that are left blank.

Parameters

- **`param`** (*`dict`*) – Dictionary of dictionaries containing the list of regions.
- **`paths`** (*`dict`*) – Dictionary of dictionaries containing the paths to *IRENA_summary*, *IRENA_dict*.

Return missing List of string of the missing regions. The CSV file for the the FLH needed for the regression is saved directly in the given path, along with the corresponding metadata in a JSON file.

Return type list of str

Raises **`Missing Regions`** – No FLH values exist for certain regions.

`lib.regression.`**`clean_TS_regression`**(*paths*, *param*, *tech*)

This function creates a CSV file containing the model time series used for regression. If the region is present in the EMHIRES text files then the TS is extracted directly from it. If the region is not present in the EMHIRES text files, the highest FLH generated TS is used instead and is scaled to match IRENA FLH if the IRENA FLH are available.

Parameters

- **`paths`** (*`dict`*) – Dictionary containing paths to EMHIRES text files.
- **`param`** (*`dict`*) – Dictionary containing the *FLH_regression* dataframe, list of subregions contained in shapefile, and year.

Returns The time series used for the regression are saved directly in the given path, along with the corresponding metadata in a JSON file.

Return type None

Raises

- **`Missing FLH`** – FLH values are missing for at least one region. No scaling is applied to the time series for those regions.
- **`Missing EMHIRES`** – EMHIRES database is missing, generated timeseries will be used as model for all regions.

`lib.regression.`**`combinations_for_regression`**(*paths*, *param*, *tech*)

This function reads the list of generated time series for different hub heights and orientations, compares it to the user-defined combinations and returns a list of lists containing all the available combinations. The function will return a warning if the user input and the available time series are not congruent.

Parameters

- **`paths`** (*`dict`*) – Dictionary of dictionaries containing the paths to the regional analysis output folder.
- **`param`** (*`dict`*) – Dictionary of dictionaries containing the subregions name, year, and user-defined combinations.
- **`tech`** (*`str`*) – Technology under study.

Return combinations List of combinations for regression.

Return type list

Raises

- **`missing data`** – If no time series are available for this technology, a warning is raised.

- **missing combination** – If a hub height or orientation is missing based on user-defined combinations, a warning is raised.

lib.regression.**get_regression_coefficients**(*paths*, *param*, *tech*)

This function solves the following optimization problem: A combination of quantiles, hub heights or orientations is to be found, so that the error to a given historical time series (e.g. from EMHIRES for European countries) is minimized, while constraining the FLH to match a given value (for example from IRENA). The settings of the combinations can be defined by the user.

The function starts by identifying the existing settings (hub heights, orientations) and quantiles. If the combinations of time series requested by the user cannot be found, a warning is raised.

It later runs the optimization and identifies the subregions for which a solution was found. If the optimization is infeasible (too high or too low FLH values compared to the reference to be matched), the time series with the closest FLH to the reference value is used in the final output.

The output consists of coefficients between 0 and 1 that could be multiplied later with the individual time series in time_series.generate_stratified_timeseries. The sum of the coefficients for each combination is equal to 1.

Parameters

- **paths** (*dict*) – Dictionary including the paths to the time series for each subregion, technology setting, and quantile, to the output paths for the coefficients.
- **param** (*dict*) – Dictionary including the dictionary of regression parameters, quantiles, and year.
- **tech** (*str*) – Technology under study.

Returns The regression parameters (e.g. IRENA FLH and EMHIRES TS) are copied under *regression_in* folder, and the regression coefficients are saved in a CSV file under *regression_out* folder, along with the metadata in a JSON file.

Return type None

Raises

- **Missing Data** – No time series present for technology *tech*.
- **Missing Data for Setting** – Missing time series for desired settings (hub heights / orientations).

lib.regression.**pyomo_regression_model**()

This function returns an abstract pyomo model of a constrained least square problem for time series fitting to match model FLHs and minimize difference error with model time series.

Return model Abstract pyomo model.

Return type pyomo object

lib.regression.**read_generated_TS**(*paths*, *param*, *tech*, *settings*, *subregion*)

This function returns a dictionary containing the available time series generated by the script based on the desired technology and settings.

Parameters

- **paths** (*dict*) – Dictionary including output folder for regional analysis.
- **param** (*dict*) – Dictionary including list of subregions and year.
- **tech** (*str*) – Technology under study.
- **settings** – List of lists containing setting combinations.
- **subregion** (*str*) – Name of the subregion.

Return GenTS Dictionary of time series indexed by setting and quantile.

Return type dict

`lib.regression.`**`regmodel_load_data`**(*paths*, *param*, *tech*, *settings*, *subregion*)

This function returns a dictionary used to initialize a pyomo abstract model for the regression analysis of each region.

Parameters

- **`paths`** (*`dict`*) – Dictionary of dictionaries containing the paths to the CSV time series files.
- **`param`** (*`dict`*) – Dictionary of dictionaries contating IRENA's region list, FLHs and EMHIRES model timeseries.
- **`tech`** (*`str`*) – Technology under study.
- **`settings`** (*`list`*) – List of all the settings (hub heights/orientations) to be used in the regression.
- **`subregion`** (*`str`*) – Name of subregion.

Return data Dictionary containing regression parameters.

Return type dict

Helping functions for the models are included in `correction_functions.py`, `spatial_functions.py`, and `physical_models.py`.

3.6 correction_functions.py

`lib.correction_functions.`**`calc_gwa_correction`**(*paths*, *param*)

This function creates a correction matrix for onshore wind based on the topography and the frequency distribution of wind speed in each country in the Global Wind Atlas. We first read the MERRA-2 data for wind and increase its resolution without editing it. We also read the topographic data for the whole scope. For each country, we filter the two datasets based on valid pixels and obtain *w50m_reg* and *topo_reg*. We correct the wind data based on the following formula:

$$w50m_{corrected} = w50m_{reg} * min(exp(ai * topo_{reg} + bi), 3.5)$$

where *ai* and *bi* are two parameters that have to be determined, so that the error (difference to the sorted frequencies of wind speeds from the GWA) is minimalized for the whole scope. Instead of using a nonlinear optimization, we simply iterate over a range of discrete possibilities for *ai* and *bi*, save the errors, then pick the combinations that minimize the error. It is possible to weight the error of each country based on its area or on its installed onshore wind capacity, or to give the same weight to all the countries. Finally, the three possible correction matrices are saved.

Parameters

- **`paths`** (*`dict`*) – Dictionary that contains the paths to wind speed data, topography, IRENA summary, Global Wind Atlass folder, and to the output location.
- **`param`** (*`dict`*) – Dictionary that contains assumptions about the desired resolution, size of the output, shapefile of countries, and georeference dictionary.

Returns The correction matrices are saved in the same MAT file, directly in the given path, along with the metadata in the corresponding JSON file.

Return type None

`lib.correction_functions.`**`clean_IRENA_summary`**(*paths*, *param*)

This function reads the IRENA database, format the output for selected regions and computes the FLH based on the installed capacity and yearly energy production. The results are saved in CSV file.

Parameters

- **`param`** (*`dict`*) – Dictionary of dictionaries containing list of subregions, and year.

- **paths** (*dict*) – Dictionary of dictionaries containing the paths to the IRENA country name dictionary, and IRENA database.

Returns The CSV file containing the summary of IRENA data for the countries within the scope is saved directly in the desired path, along with the corresponding metadata in a JSON file.

Return type None

`lib.correction_functions.`**`clean_weather_data`**(*paths*, *param*)

This function detects data outliers in the weather input .mat files. An outlier is a data point, for which the absolute value of the difference between the yearly average value and the mean of the direct neighbors (Moore neighborhood) is higher than a user-defined threshold *MERRA_correction_factor*. It replaces the hourly values with the hourly values of the mean of the neighbors, and overwrites the original .mat file.

Parameters

- **paths** (*dict*) – Dictionary including the path to the weather .mat files.
- **param** (*dict*) – Dictionary including the threshold value *MERRA_correction_factor*.

Returns The file weather .mat files are overwritten after the correction.

Return type None

`lib.correction_functions.`**`generate_wind_correction`**(*paths*, *param*)

This function creates a matrix of correction factors for onshore and/or offshore wind. There are different types of correction:

- Gradient correction: Adjusts for the hub height of the wind turbines, based on the Hellmann coefficients of each land use type. This correction applies always.
- Resolution correction: Performs a redistribution of wind speed when increasing the resolution based on land use types, while ensuring that the average of each MERRA-2 cell at 50m is still the same. This correction is optional, and is activated if *res_correction* is 1. If not activated, the same value from the low resolution is repeated.
- Topographic/Orographic correction: Takes into account the elevation of the terrain, because MERRA-2 usually underestimates the wind speed in mountains. This correction is optional, uses data from the Global Wind Atlas for all countries in the scope, and is activated only for onshore wind if *topo_correction* is 1

Parameters

- **paths** (*dict*) – Dictionary of dictionaries containing the paths to the land, land use, and topography rasters, and to the output files CORR_ON and CORR_OFF.
- **param** (*dict*) – Dictionary of dictionaries containing user-preferences regarding the wind correction, landuse, hub height, weather and desired resolutions.

Returns The rasters for wind correction CORR_ON and/or CORR_OFF are saved directly in the user-defined paths, along with their metadata in JSON files.

Return type None

3.7 spatial_functions.py

`lib.spatial_functions.`**`array2raster`**(*newRasterfn*, *rasterOrigin*, *pixelWidth*, *pixelHeight*, *array*)

This function saves array to geotiff raster format based on EPSG 4326.

Parameters

- **newRasterfn** (*string*) – Output path of the raster.
- **rasterOrigin** (*list of two floats*) – Latitude and longitude of the North-western corner of the raster.

- **pixelWidth** (*integer*) – Pixel width (might be negative).
- **pixelHeight** (*integer*) – Pixel height (might be negative).
- **array** (*numpy array*) – Array to be converted into a raster.

Returns The raster file will be saved in the desired path *newRasterfn*.

Return type None

`lib.spatial_functions.`**`calc_geotiff`**(*Crd_all*, *res_desired*)

This function returns a dictionary containing the georeferencing parameters for geotiff creation, based on the desired extent and resolution.

Parameters

- **Crd_all** (*numpy array*) – Coordinates of the bounding box of the spatial scope.
- **res_desired** (*list*) – Desired data resolution in the vertical and horizontal dimensions.

Return GeoRef Georeference dictionary containing *RasterOrigin*, *RasterOrigin_alt*, *pixelWidth*, and *pixelHeight*.

Return type dict

`lib.spatial_functions.`**`calc_region`**(*region*, *Crd_reg*, *res_desired*, *GeoRef*)

This function reads the region geometry, and returns a masking raster equal to 1 for pixels within and 0 outside of the region.

Parameters

- **region** (*Geopandas series*) – Region geometry
- **Crd_reg** (*list*) – Coordinates of the region
- **res_desired** (*list*) – Desired high resolution of the output raster
- **GeoRef** (*dict*) – Georeference dictionary containing *RasterOrigin*, *RasterOrigin_alt*, *pixelWidth*, and *pixelHeight*.

Return A_region Masking raster of the region.

Return type numpy array

`lib.spatial_functions.`**`crd_exact_points`**(*Ind_points*, *Crd_all*, *res*)

This function converts indices of points in high resolution rasters into longitude and latitude coordinates.

Parameters

- **Ind_points** (*tuple of arrays*) – Tuple of arrays of indices in the vertical and horizontal axes.
- **Crd_all** (*numpy array*) – Array of coordinates of the bounding box of the spatial scope.
- **res** (*list*) – Data resolution in the vertical and horizontal dimensions.

Return Crd_points Coordinates of the points in the vertical and horizontal dimensions.

Return type list of arrays

`lib.spatial_functions.`**`crd_merra`**(*Crd_regions*, *res_weather*)

This function calculates coordinates of the bounding box covering MERRA-2 data.

Parameters

- **Crd_regions** (*numpy array*) – Coordinates of the bounding boxes of the regions.
- **res_weather** (*list*) – Weather data resolution.

Return Crd Coordinates of the bounding box covering MERRA-2 data for each region.

Return type numpy array

`lib.spatial_functions.`**`define_spatial_scope`**`(`*scope_shp*`)`
This function reads the spatial scope shapefile and returns its bounding box.

Parameters **`scope_shp`** (*`Geopandas dataframe`*) – Spatial scope shapefile.

Return box List of the bounding box coordinates.

Return type list

`lib.spatial_functions.`**`ind_exact_points`**`(`*Crd_points*, *Crd_all*, *res*`)`
This function converts latitude and longitude of points in high resolution rasters into indices.

Parameters

- **`Crd_points`** (*`tuple of arrays`*) – Coordinates of the points in the vertical and horizontal dimensions.
- **`Crd_all`** (*`numpy array`*) – Array of coordinates of the bounding box of the spatial scope.
- **`res`** (*`list`*) – Data resolution in the vertical and horizontal dimensions.

Return Ind_points Tuple of arrays of indices in the vertical and horizontal axes.

Return type list of arrays

`lib.spatial_functions.`**`ind_global`**`(`*Crd*, *res_desired*`)`
This function converts longitude and latitude coordinates into indices on a global data scope, where the origin is at (-90, -180).

Parameters

- **`Crd`** (*`numpy array`*) – Coordinates to be converted into indices.
- **`res_desired`** (*`list`*) – Desired resolution in the vertical and horizontal dimensions.

Return Ind Indices on a global data scope.

Return type numpy array

`lib.spatial_functions.`**`ind_merra`**`(`*Crd*, *Crd_all*, *res*`)`
This function converts longitude and latitude coordinates into indices within the spatial scope of MERRA-2 data.

Parameters

- **`Crd`** (*`numpy array`*) – Coordinates to be converted into indices.
- **`Crd_all`** (*`numpy array`*) – Coordinates of the bounding box of the spatial scope.
- **`res`** (*`list`*) – Resolution of the data, for which the indices are produced.

Return Ind Indices within the spatial scope of MERRA-2 data.

Return type numpy array

`lib.spatial_functions.`**`subset`**`(`*A*, *param*`)`
This function retrieves a subset of the global MERRA-2 coverage based on weather resolution and the bounding box coordinates of the spatial scope.

Parameters

- **`A`** (*`numpy array`*) – Weather data on a global scale.
- **`param`** (*`dict`*) – Dictionary of parameters containing MERRA-2 coverage and the name of the region.

Return subset The subset of the weather data contained in the bounding box of *spatial_scope*.

Return type numpy array

3.8 physical_models.py

`lib.physical_models.`**`angles`**(*hour*, *reg_ind*, *Crd_all*, *res_desired*, *orient*)

This function creates multiple matrices for the whole scope, that represent the incidence, hour angles, declination, elevation, tilt, azimuth and orientation angles of every pixel with the desired resolution.

Parameters

- **`hour`** (*`int`*) – Hour rank in a year (from 0 to 8759).
- **`reg_ind`** (*`tuple of arrays`*) – indices of valid pixels within the spatial scope (pixels on land).
- **`Crd_all`** (*`list`*) – Coordinates of the bounding box of the spatial scope.
- **`res_desired`** (*`list`*) – Desired high resolution in degrees.
- **`orient`** (*`int`*) – Azimuth orientation of the module in degrees.

Return (phi, omega, delta, alpha, beta, azi, orientation) Rasters of latitude, hour, declination, elevation, tilt, azimuth and orientation angles.

Return type tuple of arrays

`lib.physical_models.`**`calc_CF_solar`**(*hour*, *reg_ind*, *param*, *merraData*, *rasterData*, *tech*)

This function computes the hourly capacity factor for PV and CSP technologies for all valid pixels within the spatial scope for a given hour.

Parameters

- **`hour`** (*`integer`*) – Hour within the year (from 0 to 8759).
- **`reg_ind`** (*`tuple of arrays`*) – indices of valid pixels within the spatial scope (pixels on land).
- **`param`** (*`dict`*) – Dictionary including the desired resolution, the coordinates of the bounding box of the spatial scope, and technology parameters.
- **`merraData`** (*`dict`*) – Dictionary of numpy arrays containing the weather data for every point in *reg_ind*.
- **`rasterData`** (*`dict`*) – Dictionary of numpy arrays containing land use types, Ross coefficients, albedo coefficients, and wind speed correction for every point in *reg_ind*.
- **`tech`** (*`str`*) – Name of the technology (`'PV'` or `'CSP'`).

Return (CF_pv, CF_csp) the capacity factors for all the points during that hour for PV and CSP.

Return type tuple (numpy array, numpy array)

`lib.physical_models.`**`calc_CF_wind`**(*hour*, *reg_ind*, *turbine*, *m*, *n*, *merraData*, *rasterData*)

This function computes the hourly capacity factor for onshore and offshore wind for all valid pixels within the spatial scope for a given hour.

Parameters

- **`hour`** (*`integer`*) – Hour within the year (from 0 to 8759).
- **`reg_ind`** (*`tuple of arrays`*) – indices of valid pixels within the spatial scope (pixels on land for onshore wind, on sea for offshore wind).
- **`turbine`** (*`dict`*) – Dictionary including the turbine parameters (cut-in, cut-off and rated wind speed).
- **`m`** (*`int`*) – number of rows.
- **`n`** (*`int`*) – number of columns.

- **merraData** (*dict*) – Dictionary of numpy arrays containing the weather data for every point in *reg_ind*.
- **rasterData** (*dict*) – Dictionary of numpy arrays containing the wind speed correction for every point in *reg_ind*.

Return CF Capacity factors for all the valid points during that hour.

Return type numpy array

lib.physical_models.**coefficients**(*beta*, *ratio*, *R_b*, *A_i*, *f*)

This function creates three weighting matrices for the spatial scope with the desired resolution, that correspond to the gains/losses caused by tilting to each component of the incident irradiance (direct, diffuse, and reflected).

Parameters

- **beta** (*numpy array*) – Raster of tilt angles.
- **ratio** (*numpy array*) – Diffuse fraction of global horizontal solar radiation using the Erbs model.
- **R_b** (*numpy array*) – Ratio of incident beam to horizontal beam in the HDKR model.
- **A_i** (*numpy array*) – Anisotropy index for forward scattering circumsolar diffuse irradiance in the HDKR model.
- **f** (*numpy array*) – Modulating factor for horizontal brightening correction.

Return (F_direct, F_diffuse, F_reflected) Rasters of direct, diffuse and reflected ratios of irradiance.

Return type tuple of arrays

lib.physical_models.**global2diff**(*k_t*, *dims*)

This function estimates the global-to-diffuse irradiance ratio using the Erb model.

Parameters

- **k_t** (*numpy array*) – Raster of clearness indices.
- **dims** (*tuple*) – Dimensions of the output (similar to the dimension of the angles).

Return A_ratio Raster of global-to-diffuse irradiance ratios.

Return type numpy array

lib.physical_models.**loss**(*G_tilt_h*, *TEMP*, *A_Ross*, *pv*)

This function creates a temperature loss weighting matrix for the spatial scope.

Parameters

- **G_tilt_h** (*numpy array*) – Raster of incident irradiance on the tilted panel.
- **TEMP** (*numpy array*) – Raster of ambient temperatures in °C
- **A_Ross** (*numpy array*) – Raster of Ross coefficients for temperature sensitivity.
- **pv** (*dict*) – Dictionary containing PV-specific parameters for loss coefficient and rated temperature.

Return LOSS_TEMP raster of weighting temperature loss.

Return type numpy array

lib.physical_models.**toa_hourly**(*alpha*, *hour*)

This function returns the top of the atmosphere normal irradiance based on the solar constant, hour rank, and incidence angle.

Parameters

- **alpha** (*numpy array*) – Raster of elevation angles.

- **hour** (*int*) – Hour rank of the year (from 0 to 8759).

Return TOA_h Raster of the normal top of the atmosphere irradiance.

Return type numpy array

`lib.physical_models.`**`tracking`**(*axis*, *A_phi*, *A_alpha*, *A_beta*, *A_azimuth*)

This function computes the tilt angle and orientation based on the type of tracking, incidence, elevation tilt and azimuth angles.

Parameters

- **axis** (*int*) – Number of tracking axes (0, 1, 2). The value `0` means no tracking (fixed rack), `1` means single-axis tracking in east-west dimension, and `2` means double-axis tracking.
- **A_phi** (*numpy array*) – Raster of latitude angle.
- **A_alpha** (*numpy array*) – Raster of elevation angle.
- **A_beta** (*numpy array*) – Raster of tilt angle.
- **A_azimuth** (*numpy array*) – Raster of azimuth angle.

Return (A_orient, A_beta) Tuple of rasters for orientationa and tilt angles for specified tracking type.

Return type tuple of arrays

Utility functions as well as imported libraries are included in `util.py`.

3.9 util.py

`lib.util.`**`arccosd`**(*digit*)

This function calculates the inverse cosine of a number.

Parameters **digit** (*float*) – Number between -1 and 1.

Returns The inverse cosine of the number in degrees.

Return type float

`lib.util.`**`arcsind`**(*digit*)

This function calculates the inverse sine of a number.

Parameters **digit** (*float*) – Number between -1 and 1.

Returns The inverse sine of the number in degrees.

Return type float

`lib.util.`**`arctand`**(*digit*)

This function calculates the inverse tangent of a number.

Parameters **digit** (*float*) – Number.

Returns The inverse tangent of the number in degrees.

Return type float

`lib.util.`**`changeExt2tif`**(*filepath*)

This function changes the extension of a file path to .tif.

Parameters **filepath** (*str*) – Path to the file.

Returns New path with .tif as extension.

Return type str

`lib.util.`**`changem`**(*A*, *newval*, *oldval*)

This function replaces existing values *oldval* in a data array *A* by new values *newval*.

oldval and *newval* must have the same size.

Parameters

- **`A`** (*`numpy array`*) – Input matrix.
- **`newval`** (*`numpy array`*) – Vector of new values to be set.
- **`oldval`** (*`numpy array`*) – Vector of old values to be replaced.

Return Out The updated array.

Return type numpy array

`lib.util.`**`char_range`**(*c1*, *c2*)

This function creates a generator to iterate between the characters *c1* and *c2*, including the latter.

Parameters

- **`c1`** (*`char`*) – First character in the iteration.
- **`c2`** (*`char`*) – Last character in the iteration (included).

Returns Generator to iterate between the characters *c1* and *c2*.

Return type python generator

`lib.util.`**`cosd`**(*alpha*)

This function calculates the cosine of an angle in degrees.

Parameters **`alpha`** (*`float`*) – Angle in degrees.

Returns The cosine of the angle.

Return type float

`lib.util.`**`create_json`**(*filepath*, *param*, *param_keys*, *paths*, *paths_keys*)

Creates a metadata JSON file containing information about the file in filepath by storing the relevant keys from both the param and path dictionaries.

Parameters

- **`filepath`** (*`string`*) – Path to the file for which the JSON file will be created.
- **`param`** (*`dict`*) – Dictionary of dictionaries containing the user input parameters and intermediate outputs.
- **`param_keys`** (*`list of strings`*) – Keys of the parameters to be extracted from the *param* dictionary and saved into the JSON file.
- **`paths`** (*`dict`*) – Dictionary of dictionaries containing the paths for all files.
- **`paths_keys`** (*`list of strings`*) – Keys of the paths to be extracted from the *paths* dictionary and saved into the JSON file.

Returns The JSON file will be saved in the desired path *filepath*.

Return type None

`lib.util.`**`display_progress`**(*message*, *progress_stat*)

This function displays a progress bar for long computations. To be used as part of a loop or with multiprocessing.

Parameters

- **`message`** (*`string`*) – Message to be displayed with the progress bar.
- **`progress_stat`** (*`tuple(int, int)`*) – Tuple containing the total length of the calculation and the current status or progress.

Returns The status bar is printed.

Return type None

`lib.util.`**`field_exists`**(*field_name*, *shp_path*)

This function returns whether the specified field exists or not in the shapefile linked by a path.

Parameters

- **`field_name`** (*`str`*) – Name of the field to be checked for.
- **`shp_path`** (*`str`*) – Path to the shapefile.

Returns `True` if it exists or `False` if it doesn't exist.

Return type bool

`lib.util.`**`hourofmonth`**()

This function calculates the rank within a year of the first hour of each month.

Returns The rank of the first hour of each month.

Return type list

`lib.util.`**`ind2sub`**(*array_shape*, *ind*)

This function converts linear indices to subscripts.

Parameters

- **`array_shape`** (*`tuple (int, int)`*) – Dimensions of the array (# of rows, # of columns).
- **`ind`** – Linear index.

Returns Tuple of indices in each dimension (row index, column index).

Return type tuple(int, int)

`lib.util.`**`intersection`**(*lst1*, *lst2*)

This function calculates the intersection between two lists.

Parameters

- **`lst1`** (*`list`*) – First list of elements.
- **`lst2`** (*`list`*) – Second list of elements.

Return lst3 The unique elements that exist in both lists, without repetition.

Return type list

`lib.util.`**`limit_cpu`**(*check*)

This functions sets the priority of a process for CPU time and RAM allocation at two levels: average or below average.

Parameters **`check`** (*`boolean`*) – If `True`, the process is set a below average priority rating allowing other programs to run undisturbed. if `False`, the process is given the same priority as all other user processes currently running on the machine, leading to faster calculation times.

Returns The priority of the process is set.

Return type None

`lib.util.`**`resizem`**(*A_in*, *row_new*, *col_new*)

This function resizes regular data grid, by copying and pasting parts of the original array.

Parameters

- **`A_in`** (*`numpy array`*) – Input matrix.
- **`row_new`** (*`integer`*) – New number of rows.
- **`col_new`** (*`integer`*) – New number of columns.

Return A_out Resized matrix.

Return type numpy array

`lib.util.`**`sind`**(*alpha*)

This function calculates the sine of an angle in degrees.

Parameters **`alpha`** (*`float`*) – Angle in degrees.

Returns The sine of the angle.

Return type float

`lib.util.`**`sumnorm_MERRA2`**(*A*, *m*, *n*, *res_low*, *res_desired*)

This function calculates the average of high resolution data if it is aggregated into a lower resolution.

Parameters

- **`A`** (*`numpy array`*) – High-resolution data.
- **`m`** (*`int`*) – Number of rows in the low resolution.
- **`n`** (*`int`*) – Number of columns in the low resolution.
- **`res_low`** (*`numpy array`*) – Numpy array with with two numbers. The first number is the resolution in the vertical dimension (in degrees of latitude), the second is for the horizontal dimension (in degrees of longitude).
- **`res_desired`** (*`numpy array`*) – Numpy array with with two numbers. The first number is the resolution in the vertical dimension (in degrees of latitude), the second is for the horizontal dimension (in degrees of longitude).

Return s Aggregated average of *A* on the low resolution.

Return type numpy array

`lib.util.`**`tand`**(*alpha*)

This function calculates the tangent of an angle in degrees.

Parameters **`alpha`** (*`float`*) – Angle in degrees.

Returns The tangent of the angle.

Return type float

`lib.util.`**`timecheck`**(**args*)

This function prints information about the progress of the script by displaying the function currently running, and optionally an input message, with a corresponding timestamp. If more than one argument is passed to the function, it will raise an exception.

Parameters **`args`** (*`string`*) – Message to be displayed with the function name and the timestamp (optional).

Returns The time stamp is printed.

Return type None

Raise Too many arguments have been passed to the function, the maximum is only one string.

Bibliography

[1] MERRA-2: File Specification. Note No. 9. URL: http://gmao.gsfc.nasa.gov/pubs/office_notes.

[2] Danish Wind Industry Association. *Wind energy reference manual*. 2003.

[3] John A. Duffie and William A. Beckman. *Solar engineering of thermal processes*. Wiley, Hoboken, New Jersey, fourth edition edition, 2013. ISBN 9780470873663. URL: http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10683270, doi:10.1002/9781118671603.

[4] D. G. Erbs, S. A. Klein, and J. A. Duffie. Estimation of the diffuse radiation fraction for hourly, daily and monthly-average global radiation. *Solar Energy*, 28(4):293–302, 1982. doi:10.1016/0038-092X(82)90302-4.

[5] Ronald Gelaro, Will McCarty, Max J. Suárez, Ricardo Todling, Andrea Molod, Lawrence Takacs, Cynthia A. Randles, Anton Darmenov, Michael G. Bosilovich, Rolf Reichle, Krzysztof Wargan, Lawrence Coy, Richard Cullather, Clara Draper, Santha Akella, Virginie Buchard, Austin Conaty, Arlindo M. da Silva, Wei Gu, Gi-Kong Kim, Randal Koster, Robert Lucchesi, Dagmar Merkova, Jon Eric Nielsen, Gary Partyka, Steven Pawson, William Putman, Michele Rienecker, Siegfried D. Schubert, Meta Sienkiewicz, and Bin Zhao. The modern-era retrospective analysis for research and applications, version 2 (merra-2). *Journal of Climate*, 30(14):5419–5454, 2017. doi:10.1175/JCLI-D-16-0758.1.

[6] Martin Kaltschmitt, Wolfgang Streicher, and Andreas Wiese. *Renewable Energy: Technology, and Environment Economics*. Springer-Verlag Berlin Heidelberg, Berlin, Heidelberg, 2007. ISBN 9783540709473. URL: http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10229340, doi:10.1007/3-540-70949-5.

[7] S. A. Klein. Calculation of monthly average insolation on tilted surfaces. *Solar Energy*, 19(4):325–329, 1977. doi:10.1016/0038-092X(77)90001-9.

[8] Gilbert M. Masters. *Renewable and efficient electric power systems*. Wiley-Interscience, Hoboken, 2004. ISBN 9780471280606. doi:10.1002/0471668826.

[9] Laura Maturi, Giorgio Belluardo, David Moser, and Matteo Del Buono. Bipv system performance and efficiency drops: overview on pv module temperature conditions of different module types. *Energy Procedia*, 48:1311–1319, 2014. doi:10.1016/j.egypro.2014.02.148.

[10] Douglas T. Reindl. *Estimating Diffuse Radiation On Horizontal Surfaces And Total Radiation On Tilted Surfaces: Thesis*. PhD thesis, 1988. URL: https://minds.wisconsin.edu/bitstream/1793/47660/1/0001.pdf.

[11] K. Scharmer and J. Greif. *The European solar radiation atlas: Database and Exploitation Software*. Volume 2. Presses de l Ecole des Mines, Paris, 2000. ISBN 2911762223.

[12] B. Stine William and Geyer Michael. Powerfromthesun.net. 2014.

[13] F. Marion William and P. Dobos Aron. Rotation angle for the optimum tracking of one-axis trackers. 2013. URL: https://www.nrel.gov/docs/fy13osti/58891.pdf.

Python Module Index

Appendix B

Documentation of tum-ens/pyCLARA

This is the documentation of *pyCLARA: python Clustering of Lines And RAsters*, which clusters high-resolution spatial data (rasters or polylines connecting Voronoi polygons) into contiguous, homogeneous regions.

Features

- Clustering of one or multiple high-resolution rasters, such as wind resource maps or load density maps;
- Supported aggregation functions: average, sum, or density;
- Combination of k-means and max-p algorithms, to ensure contiguity;
- Clustering of grid data using a hierarchical algorithm;
- Flexibility in the number of polygons obtained.

Applications

This code is useful if you want to:

- obtain regions for energy system models with homogeneous characteristics (e.g. similar wind potential);
- cluster regions based on several characteristics simultaneously;
- take into account grid restrictions when defining regions for power system modeling.

Contributors

Mohammad Youssef Mahfouz created the first version of this code under my supervision. Waleed Sattar Khan improved the coding style and added the module for clustering transmission lines during his Master's Thesis that I supervised. Houssame Houmy edited the layout to reproduce the same repository structure as in the other tools. I improved the modularity and flexibility of the tool. The individual contributions can be traced back in GitHub (Siala et al., 2020).

Contents

Chapter 1

User manual

1.1 Installation

Note: We assume that you are familiar with git and conda.

First, clone the git repository in a directory of your choice using a Command Prompt window:

```
$ ~\directory-of-my-choice> git clone https://github.com/tum-ens/pyCLARA.git
```

We recommend using conda and installing the environment from the file `geoclustering.yml` that you can find in the repository. In the Command Prompt window, type:

```
$ cd pyCLARA\env\
$ conda env create -f geoclustering.yml
```

Then activate the environment:

```
$ conda activate geoclustering
```

In the folder `code`, you will find multiple files:

File	Description
config.py	used for configuration, see below.
runme.py	main file, which will be run later using `python runme.py`.
lib\initialization.py	used for initialization.
lib\create_subproblems.py	used to split the clustering problem into smaller subproblems.
lib\kmeans_functions.py	includes functions related to the k-means clustering algorithm.
lib\max_p_functions.py	includes functions related to the max-p clustering algorithm.
lib\lines_clustering_functions.py	includes functions for the hierarchical clustering of transmission lines.
lib\spatial_functions.py	contains helping functions for spatial operations.
lib\util.py	contains minor helping functions and the necessary python libraries to be imported.

1.2 config.py

This file contains the user preferences, the links to the input files, and the paths where the outputs should be saved. The paths are initialized in a way that follows a particular folder hierarchy. However, you can change the hierarchy as you wish.

1.2.1 Main configuration function

`config.`**`configuration`**`()`

This function is the main configuration function that calls all the other modules in the code.

Return (paths, param) The dictionary paths containing all the paths to inputs and outputs, and the dictionary param containing all the user preferences.

Return type tuple(dict, dict)

`config.`**`general_settings`**`()`

This function creates and initializes the dictionaries param and paths. It also creates global variables for the root folder `root`, and the system-dependent file separator `fs`.

Return (paths, param) The empty dictionary paths, and the dictionary param including some general information.

Return type tuple(dict, dict)

Note: Both *param* and *paths* will be updated in the code after running the function `config.configuration`.

Note: `root` points to the directory that contains all the inputs and outputs. All the paths will be defined relatively to the root, which is located in a relative position to the current folder.

`config.`**`scope_paths_and_parameters`**`(`*paths*, *param*`)`

This function assigns a name for the geographic scope, and collects information regarding the input rasters that will be clustered:

- *region_name* is the name of the geographic scope, which affects the paths where the results are saved.
- *spatial_scope* is the path to the geographic scope that will be used to clip the map of transmission lines. You can ignore it if you are only clustering rasters.
- *raster_names* are the name tags of the inputs. Preferably, they should not exceed ten (10) characters, as they will be used as attribute names in the created shapefiles. If the user chooses strings longer than that, they will be cut to ten characters, and no error is thrown. The name tags are entered as keys into the dictionary `inputs`.
- *inputs* are the paths to the input rasters (strings). They are given as the first element of a values tuple for each key in the dictionary `inputs`.
- *agg* are the aggregation methods for the input data (strings: either `'mean'` or `'sum'` or `'density'`). They are given as the second element of a values tuple for each key in the dictionary `inputs`.
- *weights* are the weights of the input data during the clustering (int or float). They are given as the third element of a values tuple for each key in the dictionary `inputs`.

Parameters

- **`paths`** (*`dict`*) – Dictionary including the paths.
- **`param`** (*`dict`*) – Dictionary including the user preferences.

Return (paths, param) The updated dictionaries paths and param.

Return type tuple(dict, dict)

1.2.2 User preferences

`config.`**`computation_parameters`**`(`*param*`)`

This function sets the limit to the number of processes *n_jobs* that can be used in k-means clustering.

Parameters **param** (*dict*) – Dictionary including the user preferences.

Return param The updated dictionary param.

Return type dict

`config.`**`raster_parameters`**(*param*)

This function sets the parameters for the input rasters.

- *minimum_valid* is the lowest valid value. Below it, the data is considered NaN.
- *CRS* is the coordinates reference system. It must be the same for all raster input maps.

Parameters **param** (*dict*) – Dictionary including the user preferences.

Return param The updated dictionary param.

Return type dict

`config.`**`raster_cutting_parameters`**(*paths*, *param*)

This function sets how the large input rasters are cut before starting the clustering. There are two options: the maps are either cut using a shapefile of (multi)polygons, or using rectangular boxes.

- *use_shapefile*: if 1, a shapefile is used, otherwise rectangular boxes.
- *subregions*: the path to the shapefile of (multi)polygons that will cut the large raster in smaller parts (only needed if *use_shapefile* is 1).
- *rows*: number of rows of boxes that the raster will be cut into (only needed if *use_shapefile* is 0).
- *cols*: number of columns of boxes that the raster will be cut into (only needed if *use_shapefile* is 0).

Parameters

- **paths** (*dict*) – Dictionary including the paths.
- **param** (*dict*) – Dictionary including the user preferences.

Return (paths, param) The updated dictionaries paths and param.

Return type tuple(dict, dict)

`config.`**`kmeans_parameters`**(*param*)

This function sets the parameters for the k-means clustering:

- *method*: Currently, two methods for setting the number of clusters are used. By choosing `'maximum_number'`, the user sets the total number of clusters for all parts. This number will be distributed over the parts depending on their size and the standard deviation of their data. If the user chooses `'reference_part'`, then the part with the highest product of relative size and standard deviation (relatively to the maximum) is chosen as a reference part. For this one, the maximum number of clusters is identified using the Elbow method. The optimum number of clusters for all the other parts is a function of that number, and of their relative size and standard deviation.

Warning: The `'maximum_number'` might be exceeded due to the rounding of the share of each part.

- *ratio_size_to_std*: This parameter decides about the weight of the relative size and the relative standard deviation (relatively to the maximum) in determining the optimal number of clusters for each part. A ratio of 7:3 means that 70% of the weight is on the relative size of the part, and 30% is on its standard deviation. Any number greater than zero is accepted.
- *reference_part*: This is a dictionary that contains the parameters for the Elbow method. Cluster sizes between *min* and *max* with a certain *step* will be tested in about for-loop, before the optimal number of clusters for the reference part can be identified. The dictionary is only needed it the method is `'reference_part'`.

- *maximum_number*: This integer sets the maximum number of kmeans clusters for the whole map. It is only used if the method is `'maximum_number'`.

  Parameters **`param`** (*`dict`*) – Dictionary including the user preferences.

  Return param The updated dictionary param.

  Return type dict

`config.`**`maxp_parameters`**(*param*)

This function sets the parameters for max-p clustering. Currently, one or two iterations of the max-p algorithm can be used, depending on the number of polygons after kmeans.

- *maximum_number*: This number (positive float or integer) defines the maximum number of clusters that the max-p algorithm can cluster in one go. For about 1800 polygons, the calculation takes about 8 hours. The problem has a complexity of O(n^3) in the Bachmann-Landau notation.
- *final_number*: This integer defines the number of clusters that the user wishes to obtain at the end. There is no way to force the algorithm to deliver exactly that number of regions. However, the threshold can be defined as a function of *final_number*, so that the result will be close to it.
- *use_results_of_maxp_parts*: This parameter should be set to zero, unless the user has already obtained results for the first run of the max-p algorithm, and want to skip it and just run the second round. In that case, the user should set the value at 1.

  Parameters **`param`** (*`dict`*) – Dictionary including the user preferences.

  Return param The updated dictionary param.

  Return type dict

`config.`**`transmission_parameters`**(*param*)

This function sets the parameters for transmission line clustering.

- *CRS_grid*: The coordinates reference system of the shapefile of transmission lines, in order to read it correctly.
- *default_cap_MVA*: Line capacity in MVA for added lines (to connect electric islands).
- *default_line_type*: Line type for added lines (to connect electric islands).
- *number_clusters*: Target number of regions after clustering, to be used as a condition to stop the algorithm.
- *intermediate_number*: List of numbers of clusters at which an intermediate shapefile will be saved. The values affect the path *grid_intermediate*.
- *debugging_number*: Number of clusters within an intermediate shapefile, that can be used as an input (for debugging). It affects the path *grid_debugging*.

  Parameters **`param`** (*`dict`*) – Dictionary including the user preferences.

  Return param The updated dictionary param.

  Return type dict

1.2.3 Paths

`config.`**`output_folders`**(*paths*, *param*)

This function defines the paths to multiple output folders:

- *region* is the main output folder. It contains the name of the scope, and the names of the layers used for clustering (as a subfloder).
- *sub_rasters* is a subfolder containing the parts of the input rasters after cutting them.
- *k_means* is a subfolder containing the results of the kmeans clustering (rasters).

- *polygons* is a subfolder containing the polygonized kmeans clusters.
- *parts_max_p* is a subfolder containing the results of the first round of max-p (if there is a second round).
- *final_output* is a subfolder containing the final shapefile.
- *lines_clustering* is a subfolder containing the intermediate and final results of the line clustering.

All the folders are created at the beginning of the calculation, if they do not already exist,

Parameters

- **`paths`** (*`dict`*) – Dictionary including the paths.
- **`param`** (*`dict`*) – Dictionary including the user preferences.

Return paths The updated dictionary paths.

Return type dict

`config.`**`output_paths`**(*paths*, *param*)

This function defines the paths of some of the files that will be saved:

- *input_stats* is the path to a CSV file with general information such as the number of parts, the maximal size and the maximal standard deviation in the parts, and the maximum number of clusters as set by the user / by the Elbow method.
- *non_empty_rasters* is the path to a CSV file with information on each subraster (relative size, standard deviation, etc.).
- *kmeans_stats* is the path to a CSV file that is only created if the Elbow method is used (i.e. if using a reference part). It contains statistics for kmeans while applying the Elbow method.
- *polygonized_clusters* is the path to the shapefile with the polygonized rasters for the whole scope.
- *max_p_combined* is the path to the shapefile that is generated after a first round of the max-p algorithm (if there is a second).
- *output* is the path to the shapefile that is generated at the end, i.e. after running max_p_whole_map in `lib.max_p_functions.py`.

For line clustering, the keys start with *grid_*:

- *grid_connected* is the path to the shapefile of lines after adding lines to connect island grids.
- *grid_clipped* is the path to the shapefile of lines after clipping it with the scope.
- *grid_voronoi* is the path to the shapefile of voronoi polygons made from the points at the start/end of the lines.
- *grid_debugging* is the path of an intermediate file during the clustering of regions based on their connectivity.
- *grid_regions* is the path to the final result of the clustering (shapefile of regions based on their connectivity).
- *grid_bottlenecks* is the path to the final result of the clustering (shapefile of transmission line bottlenecks).

Parameters **`paths`** (*`dict`*) – Dictionary including the paths.

Return paths The updated dictionary paths.

Return type dict

1.3 runme.py

`runme.py` calls the main functions of the code:

```
from lib.initialization import initialization
from lib.create_subproblems import cut_raster
from lib.kmeans_functions import *
from lib.max_p_functions import *
from lib.lines_clustering_functions import *
from lib.util import *

if __name__ == "__main__":

    paths, param = initialization()

    cut_raster(paths, param)

    # k-means functions
    calculate_stats_for_non_empty_rasters(paths, param)
    if param["kmeans"]["method"] == "reference_part":
        choose_ref_part(paths)
        identify_max_number_of_clusters_in_ref_part(paths, param)
    k_means_clustering(paths, param)
    polygonize_after_k_means(paths, param)

    # max-p functions
    max_p_clustering(paths, param)

    # lines clustering functions
    lines_clustering(paths, param)
```

1.4 Recommended input sources

For a list of GIS data sources, check this wikipedia article.

1.4.1 Shapefile of transmission lines

High-voltage power grid data for Europe and North America can be obtained from GridKit, which used OpenStreetMap as a primary data source.

In this repository, the minimum requirements are a shapefile of lines with the attributes ID, Cap_MVA and Type. Such a file can be obtained using the repository tum-ens/pyPRIMA to clean the GridKit data.

1.4.2 Shapefile of the region of interest (useful for lines clustering)

The shapefile allows to identify the lines that are inside the scope, and those that lie outside of it. Hence, only the outer borders of the polygons matter. If you are interested in administrative divisions, you may consider downloading the shapefiles from the website of the Global Administration Divisions (GADM). You can also create your own shapefiles using a GIS software.

1.4.3 High resolution rasters

Any raster can be used. If multiple rasters are used, ensure that they have the same resolution, the same projection, and the same geographic extent. Rasters for renewable potentials can be generated using the GitHub repository tum-ens/pyGRETA. A high resolution raster (30 arcsec) of population density can be downloaded from the website of SEDAC after registration. A high resolution raster (15 arcsec = 1/240° longitude and 1/240° latitude) made of 24 tiles can be downloaded from viewfinder panoramas.

1.5 Recommended workflow

The script is designed to be modular and split into four multiple modules: `lib.create_subproblems`, `lib.kmeans_functions`, `lib.max_p_functions`, and `lib.lines_clustering_functions`.

Warning: The outputs of some modules serve as inputs to the others (this applies to the first three modules). Therefore, the user will have to run the script sequentially.

There are two recommended use cases: either the clustering of rasters, or the clustering of transmission lines. The modules will be presented in the order in which the user will have to run them.

1. *Clustering of rasters*

a. *Creating subproblems*

b. *k-means clustering*

c. *max-p functions*

2. *Clustering of transmission lines*

It is recommended to thoroughly read through the configuration file *config.py* and modify the input paths and computation parameters before starting the *runme.py* script. Once the configuration file is set, open the *runme.py* file to define what use case you will be using the script for.

1.5.1 Clustering of rasters

The first use case deals with the clustering of high resolution rasters. Depending on the size of the rasters and the user preferences, all or some of these modules will be used:

Creating subproblems

Instead of applying the clustering algorithms directly on the original rasters, the module `lib.create_subproblems` is called to split the rasters into smaller ones. There are two options: either to split the rasters into rectangles of similar sizes, or according to polygon shapes provided by the user.

Note: If you would like to cluster data on a European level, but would like to do it for each country separately, provide a shapefile of European countries to define the subregions.

You can also skip this step altogether and cluster the whole dataset at once (not recommended for very large maps, because the quality of the results and the speed of the calculation are affected).

k-means clustering

This step is also optional. The purpose is to compress the amount of information so that the max-p algorithm can be applied. The max-p algorithm can cluster up to ~1800 polygons in about 8h. So the functions inside the `lib.kmeans_functions` module have two goals: reduce the number of data points while preserving the maximum amount of information that can be processed by the next module, and polygonizing the rasters (i.e. create shapefiles of polygons from clustered rasters).

The user can define the number of clusters *k*, or let the code decide depending on the standard deviation and the number of valid data points.

max-p functions

The max-p algorithm is the one that ensures the spatial contiguity of the clustered data points, but due to its computational complexity, the previous steps might be needed. Also, there is a possibility to run the max-p algorithm in two steps: first on each subproblem separately, then on the whole geographic scope after combining the results of the subproblems. If the number of polygons after k-means is manageable in one run, then the max-p algorithm is applied once. The result is a shapefile of contiguous polygons with similar characteristics.

Note: You can customize the properties, the weights of the data sets, and the aggregating functions to be used during the clustering, e.g. 20% solar FLH (averaged), 40% wind FLH (averaged), and 40% electricity demand (summed).

1.5.2 Clustering of transmission lines

This use case is currently independent of the previous one. Starting from the maximum number of regions surrounding each node in the grid, it uses a hierarchical algorithm to merge connected nodes that have a high transmission capacity flowing from/into them and a low area. The algorithm stops when the target number of clusters is reached.

Chapter 2

Theory

2.1 Introduction

A number of different algorithms have been developed by scientists for clustering geographic data. The most trivial and commonly used algorithm is the k-means algorithm but it has some limitations. These limitations urged the researchers to come up with more novel and sophisticated methods of clustering geographic data. Usually, the type of algorithm which is used depends on the type of data available and the associated time complexity and requirements for clustering quality. Furthermore, the clustering algorithms are based on different techniques and one of these techniques is hierarchical clustering.

2.2 Hierarchical Clustering

A hierarchical clustering algorithm forms a dendrogram which is basically a tree that splits the input data into small subsets using a recursive approach [1]. The dendrogram can be built using two different approaches which are “bottom-up” or “top-down” approach. As the name suggests, the bottom-up approach treats each object as a separate cluster and combines the two “closest” clusters into one. The algorithm continues until all objects are merged together to form one big object. On the other hand, the top-down approach starts with one big cluster in which all the objects exist. The big cluster is repeatedly broken down into smaller clusters in each iteration of the algorithm [1].

2.3 k-means Method

The k-means method is one of the oldest methods that has been used for clustering. In this method, the average value of data objects in a cluster is used as the cluster center [2]. The steps that the k-means method follows are:

# The number of clusters k are chosen. The choice of right k is important as small k can result in clusters in which distances to centroid are very long whereas a too large k means that the algorithm will take longer to converge with little to no improvement in average distances. # Each cluster is initialized by either of the two methods namely Forgy method or Random partition method. For standard k-means algorithm, Forgy method is preferable in which k random observations are chosen as initial means for the clusters [3]. # This step is known as the assignment step. A total of k number of clusters are created with each data object being assigned to the nearest mean. The figure so obtained is known as Voronoi diagram. # The mean of all the data points in each cluster is calculated and the newly calculated mean serves as the new centroid for each cluster. This step is called the update step. # Step (3) and (4) are repeated until convergence criteria has been attained. Usually, the convergence is achieved when the object assignments do not change any more.

Generally, the k-means algorithm has a time complexity equal to O (n2) [4]. Moreover, it is a heuristic method which means that convergence to a global optimum is not guaranteed. However, as k-means is a relatively fast-algorithm, it is a common practice to run k-means method of clustering with different starting conditions and choose the run that has the best results.

2.4 k-means++ Method

The k-means++ clustering algorithm can be thought of as an add-on to the standard k-means clustering algorithm. The k-means++ algorithm is designed to improve the quality of clusters that are formed as local optimum using the standard k-means method [4]. Moreover, this method also tends to lower the runtime of the subsequently used k-means algorithm by making it converge quickly. k-means++ achieves this by choosing "seeds" for k-means algorithm. The "seeds" are the initial k cluster centers which are chosen based on a certain criteria instead of randomly choosing the cluster centers as was the case in the standard k-means algorithm. The steps that k-means++ algorithm follows for initializing the cluster centers are as follows:

# Choose one data point at random from the given data set and consider it as the first center c¬i. # Calculate the distance of each of the data points in the set from the nearest cluster center with D(x) being the distance of each point. # Choose a new cluster center ci. # Repeat steps (2) and (3) until a k number of centers have been chosen. # Using the initialized centers, proceed with the standard k-means algorithm as outlined in previous section.

Although initializing cluster centers using k-means++ method is computationally expensive, the subsequent k-means method actually takes less time to converge and as a result, the overall time taken to converge is usually less as compared to the standard k-means method. Moreover, the k-means++ method is O (log k) competitive [4]. Furthermore, as mentioned above, the local optimum obtained is much more optimized and therefore, the k-means++ method is better than the standard k-means method for clustering of data. Fig. 3 [5] shows the results of a clustering performed on a set of typical data using k-means and k-means++ method. It is clear from the figure that error is smaller when k-means++ is used for clustering and the convergence time is also almost the same.

2.5 max-p Method

The max-p regions problem has been developed to satisfy spatial contiguity constraints while clustering. As opposed to other constraint-based methods, this methods models the total number of sectors as an internal parameter [6]. Moreover, this approach does not put restrictions on the geometry of regions. In fact, it lets the data sets dictate the shape. The heuristic solution that has been proposed in order to solve the max-p regions problem follows the following steps.

# The first is the construction phase which is subdivided into two more phases. The first sub-phase is named growing phase in which the algorithm selects an unassigned area at random. This is called the seed area. # The neighboring regions of seed area are continuously added to the first seed until a threshold is reached. # Then the algorithm chooses a new seed area and grows a region by following step (2). This step is repeated until no new seeds which can satisfy the threshold value can be found. # Areas which are not assigned to a region are called enclaves. Therefore, at the end of growing phase, a set of enclaves and growing regions are formed. # The number of growing regions and enclaves can change in successive iterations. Solutions in which maximum growing regions are equal to maximum growing regions in the previous iteration are only kept. # The next step is the process of enclave assignment and each solution found in previous steps is passed to this step. Each enclave is assigned to a neighboring region based on similarity. This marks the end of construction phase. # In the next step i.e. the local search phase, a set of neighboring solutions is obtained by modifying the found solutions and improving them. Neighboring solutions are created in a way that each solution is feasible. One way to create neighboring solutions is to move one region to another region [7]. # Finally, a local search algorithm such as Simulated Annealing, Greedy Algorithm or Tabu Search Algorithm is used to improve the solution.

The main purpose of max-p regions problem and the proposed solution is to aggregate small areas into homogeneous regions in a way that the value of a spatially extensive attribute is always more than the threshold value. It can be useful for clustering rasters in which constraints such as population density, energy per capita need to be met [6].

Chapter 3

Implementation

Start with the configuration:

You can run the code by typing:

```
$ python runme.py
```

The script `runme.py` calls the main functions of the code, which are explained in the following sections.

3.1 initialization.py

`lib.initialization.`**`initialization`**`()`

This function reads the user-defined parameters and paths from `config.py`, then checks the validity of the input files. If they are missing or are not rasters, a warning is thrown and the code is exited. The same applies if the rasters do not have the same resolution or scope.

If the input files are valid, the CSV file in *input_stats* is generated and filled with the information that is already available. It will be called by other functions and eventually filled by them.

Returns The updated dictionaries param and paths.

Return type tuple(dict, dict)

3.2 create_subproblems.py

`lib.create_subproblems.`**`cut_raster`**`(`*paths*, *param*`)`

This function decides, based on the user preference in *use_shapefile*, whether to call the module `cut_raster_using_shapefile` or `cut_raster_using_boxes`.

Parameters

- **`paths`** (*`dict`*) – Dictionary of paths to inputs and outputs.
- **`param`** (*`dict`*) – Dictionary of parameters and user preferences.

Returns The submodules `cut_raster_using_shapefile` and `cut_raster_using_boxes` have their own outputs.

Return type None

`lib.create_subproblems.`**`cut_raster_using_boxes`**`(`*paths*, *param*`)`

This function cuts the raster file into a m*n boxes with m rows and n columns.

Parameters

- **`paths`** (*`dict`*) – The paths to the input rasters *inputs*, to the output raster parts *sub_rasters*, and to the CSV file *input_stats* which will be updated.
- **`param`** (*`dict`*) – Parameters that include the number of *rows* and *cols*, and the *raster_names*.

Returns The raster parts are saved as GEOTIFF files, and the CSV file *input_stats* is updated.

Return type None

`lib.create_subproblems.`**`cut_raster_using_shapefile`**(*paths*, *param*)

This function cuts the raster file into parts using the features of a shapefile of (multi)polygons. Each feature is used as a mask, so that only the pixels lying on it are extracted and saved in a separate raster file.

Parameters

- **`paths`** (*`dict`*) – The paths to the input rasters *inputs*, to the shapefile of *subregions*, to the output raster parts *sub_rasters*, and to the CSV file *input_stats* which will be updated.
- **`param`** (*`dict`*) – Parameters that include the array *Crd_all*, the array *res_desired* and the dictionary *GeoRef* for georeferencing the output rasters, the minimum valid value *minimum_valid*, and the *raster_names*.

Returns The raster parts are saved as GEOTIFF files, and the CSV file *input_stats* is updated.

Return type None

3.3 kmeans_functions.py

`class` `lib.kmeans_functions.`**`OptimumPoint`**(*init_x*, *init_y*)

This class is used in the Elbow method to identify the maximum distance between the end point and the start point of the curve of inertia as a function of number of clusters.

`lib.kmeans_functions.`**`calculate_stats_for_non_empty_rasters`**(*paths*, *param*)

This function calculates statistics for all non empty subrasters. These statistics include the number of rows and columns, the size (number of valid points), the standard deviation, the relative size (to the maximum) and the relative standard deviation, the product of the latter two, and the values of four mapping functions using the relative size and relative standard deviation.

The product of the relative size and relative standard deviation is used to identify the reference part. As of the four mapping functions, they are used to identify the four parts that lie in the corners of the cloud of points, where each point represents a part and is plotted on a graph with the relative size in one axis, and relative standard deviation on the other.

Parameters

- **`paths`** (*`dict`*) – Dictionary containing the paths to the folder of *inputs*, to the CSV *input_stats*, to the folder of *sub_rasters*, and to the output CSV *non_empty_rasters*.
- **`param`** (*`dict`*) – Dictionary of parameters containing the *raster_names* and the minimum valid value in the rasters, *minimum_valid*.

Returns The results are directly saved in the desired CSV file *non_empty_rasters*, and the CSV file *input_stats* is also updated.

Return type None

`lib.kmeans_functions.`**`choose_ref_part`**(*paths*)

This function chooses the reference part for the function `identify_max_number_of_clusters_in_ref_part`. The reference part is chosen based on the product of relative size and relative standard deviation. The part with the largest product in all the input files is chosen.

Parameters **`paths`** (*`dict`*) – The paths to the CSV files *non_empty_rasters* and *input_stats*.

Returns The CSV file *input_stats* is updated.

Return type None

`lib.kmeans_functions.`**`identify_max_number_of_clusters_in_ref_part`**(*paths*, *param*)

This function identifies the maximum number of clusters for the reference part using the Elbow method. The number of clusters is varied between *min* and *max* by *step*, and in each case, the inertia (distances to the cluster centers) are calculated. If the slope of the change of the inertia goes below a certain threshold, the function is interrupted and the maximum number of clusters for the reference part is determined.

Parameters

- **paths** (*dict*) – Dictionary containing the paths to the folder of *inputs*, to the CSV *input_stats*, to the folder of *sub_rasters*, and to the output CSV *kmeans_stats*.
- **param** (*dict*) – Dictionary of parameters containing the *raster_names* and their *weights*, the minimum valid value in the rasters, *minimum_valid*, kmeans-related parameters for the iteration of the Elbow method, and the number of processes *n_job*.

Returns The results are directly saved in the desired CSV file *kmeans_stats*, and the CSV file *input_stats* is also updated.

Return type None

`lib.kmeans_functions.`**`identify_opt_number_of_clusters`**(*paths*, *param*, *part*, *size_of_raster*, *std_of_raster*)

This function identifies the optimal number of clusters which will be chosen for k-means in each part.

In case you are using a reference part, then the optimal number is a function of the number of clusters in the reference part, and of the relative size and relative standard deviation, which are weighted according to *ratio_size_to_std*.

In case you are using the maximum number for the whole map, then the optimal number in each part is a function of the total number, of the relative size and relative standard deviation, and the weights in *ratio_size_to_std*.

Parameters

- **paths** (*dict*) – Dictionary of paths pointing to the location of the input CSV file *non_empty_rasters* and to *input_stats*.
- **param** (*dict*) – Dictionary of parameters including the ratio between the relative size and the relative standard deviation *ratio_size_to_std* and the *method* for setting the number of clusters.
- **part** (*integer*) – Counter for the raster parts.
- **size_of_raster** (*integer*) – Number of valid data points in the raster part.
- **std_of_raster** (*float*) – Standard deviation of the data in the raster part.

Return optimum_no_of_clusters_for_raster Optimum number of clusters for the raster part according to the chosen method.

Return type integer

`lib.kmeans_functions.`**`k_means_clustering`**(*paths*, *param*)

This function does the k-means clustering for every part.

Parameters

- **paths** (*dict*) – Dictionary containing the paths to the folder of *inputs*, to the CSV *input_stats* and *non_empty_rasters*, to the folder of *sub_rasters*, and to the output folder *kmeans*.
- **param** (*dict*) – Dictionary of parameters containing the *raster_names* and their *weights* and aggregation methods *agg*, the minimum valid value in the rasters, *minimum_valid*, the *method* for finding the number of kmeans clusters, and the number of processes *n_job*.

Returns The results are directly saved in the desired CSV file *kmeans_stats*, and the CSV file *input_stats* is also updated.

Return type None

`lib.kmeans_functions.`**`polygonize_after_k_means`**(*paths*, *param*)

This function converts the rasters created after k-means clustering into shapefiles of (multi)polygons which are used in the max-p algorithm.

Parameters

- **`paths`** (`dict`) – Dictionary containing the paths to the folder of *kmeans* for retrieving inputs, to the CSV *non_empty_rasters*, and to the folder *polygons* for saving outputs.
- **`param`** (`dict`) – Dictionary of parameters containing the minimum valid value in the rasters, *minimum_valid*, and the *CRS* of the shapefiles.

Returns The results are directly saved in the desired paths for each part (folder *polygons*) and for the whole map (file *polygonized_clusters*).

Return type None

3.4 max_p_functions.py

`lib.max_p_functions.`**`correct_neighbors_in_shapefile`**(*param*, *combined_file*, *existing_neighbors*)

This function finds the neighbors in the shapefile. Somehow, max-p cannot figure out the correct neighbors and some clusters are physically neighbors but they are not considered as neighbors. This is where this function comes in.

It creates a small buffer around each polygon. If the enlarged polygons intersect, and the area of the intersection exceeds a threshold, then the polygons are considered neighbors, and the dictionary of neighbors is updated.

Parameters

- **`param`** (`dict`) – The dictionary of parameters including the coordinate reference system *CRS* and the resolution of input rasters *res_desired*.
- **`combined_file`** (`str`) – The path to the shapefile of polygons to be clustered.
- **`existing_neighbors`** (`dict`) – The dictionary of neighbors as extracted from the shapefile, before any eventual correction.

Return neighbors_corrected The dictionary of neighbors after correction (equivalent to an adjacency matrix).

Return type dict

`lib.max_p_functions.`**`eq_solver`**(*coef*, *ll_point*, *ul_point*, *ur_point*)

This function serves as the solver to find coefficient values A, B and C for our defined function which is used to calculate the threshold.

Parameters

- **`coef`** (`dict`) – The coefficients which are calculated.
- **`ll_point`** (`tuple(int, int)`) – Coordinates of lower left point.
- **`ul_point`** (`tuple(int, int)`) – Coordinates of upper left point.
- **`ur_point`** (`tuple(int, int)`) – Coordinates of upper right point.

Return f Coefficient values for A, B and C in a numpy array. A is f[0], B is f[1] and C is f[2].

Return type numpy array

`lib.max_p_functions.`**`get_coefficients`**(*paths*)
This function gets the coefficients A, B and C for solving the 3 equations which will lead to the calculation of the threshold in the max-p algorithm.

Parameters **`paths`** (*`str`*) – The dictionary of paths including the one to *non_empty_rasters*.

Return coef The coefficient values for A, B and C returned as a dictionary. The expected structure is similar to this dictionary: {'A': 0.55, 'B': 2.91, 'C': 0.61}.

Return type dict

`lib.max_p_functions.`**`max_p_clustering`**(*paths*, *param*)
This function applies the max-p algorithm to the obtained polygons. Depending on the number of clusters in the whole map after k-means, it decides whether to run max-p clustering multiple times (for each part, then for the whole map) or once (for the whole map). If you have already results for each part, you can skip that by setting *use_results_of_max_parts* to 1.

Parameters

- **`paths`** (*`dict`*) – Dictionary of paths pointing to *polygonized_clusters* and *max_p_combined*.
- **`param`** (*`dict`*) – Dictionary of parameters including max-p related parameters (*maximum_number* and *use_results_of_maxp_parts*), and eventually the *compression_ratio* for the first round of max-p clustering.

Returns The called functions `max_p_parts` and `max_p_whole_map` generate outputs.

Return type None

`lib.max_p_functions.`**`max_p_parts`**(*paths*, *param*)
This function applies the max-p algorithm on each part. It identifies the neighbors from the shapefile of polygons for that part. If there are disconnected parts, it assumes that they are neighbors with the closest polygon.

The max-p algorithm aggregates polygons to a maximum number of regions that fulfil a certain condition. That condition is formulated as a minimum share of the sum of data values, `thr`. The threshold is set differently for each part, so that the largest and most diverse part keeps a large number of polygons, and the smallest and least diverse is aggregated into one region. This is determined by the function `get_coefficients`.

After assigning the clusters to the polygons, they are dissolved according to them, and the values of each property are aggregated according to the aggregation functions of the inputs, saved in *agg*.

Parameters

- **`paths`** (*`dict`*) – Dictionary containing the paths to the folder of *inputs* and *polygons*, to the CSV *non_empty_rasters*, to the output folder *parts_max_p* and to the output file *max_p_combined*.
- **`param`** (*`dict`*) – Dictionary of parameters containing the *raster_names* and their *weights* and aggregation methods *agg*, the *compression_ratio* of the polygonized kmeans clusters, and the *CRS* to be used for the shapefiles.

Returns The results of the clustering are shapefiles for each part, saved in the folder *parts_max_p*, and a combination of these for the whole map *max_p_combined*.

Return type None

`lib.max_p_functions.`**`max_p_whole_map`**(*paths*, *param*, *combined_file*)
This function runs the max-p algorithm for the whole map, either on the results obtained from `max_p_parts`, or on those obtained from `polygonize_after_k_means`, depending on the number of polygons after kmeans clustering.

It identifies the neighbors from the shapefile of polygons. If there are disconnected components (an island of polygons), it assumes that they are neighbors with the closest polygon. It also verifies that the code identifies neighbors properly and corrects that eventually using `correct_neighbors_in_shapefile`.

The max-p algorithm aggregates polygons to a maximum number of regions that fulfil a certain condition. That condition is formulated as a minimum share of the sum of data values, `thr`. The threshold for the whole map is set as a function of the number of polygons before clustering, and the desired number of polygons at the end. However, that number may not be matched exactly. The user may wish to adjust the threshold manually until the desired number is reached (increase the threshold to reduce the number of regions, and vice versa).

After assigning the clusters to the polygons, they are dissolved according to them, and the values of each property are aggregated according to the aggregation functions of the inputs, saved in *agg*.

Parameters

- **`paths`** (*`dict`*) – Dictionary containing the paths to the folder of *inputs* and to the output file *output*.
- **`param`** (*`dict`*) – Dictionary of parameters containing the *raster_names* and their *weights* and aggregation methods *agg*, the desired number of features at the end *final_number*, and the *CRS* to be used for the shapefiles.
- **`combined_file`** (*`str`*) – Path to the shapefile to use as input. It is either the result obtained from `max_p_parts`, or the one obtained from `polygonize_after_k_means`.

Returns The result of the clustering is one shapefile for the whole map saved directly in *output*.

Return type None

For the hierarchical clustering of the transmission network, use the following script:

3.5 lines_clustering_functions.py

`lib.lines_clustering_functions.`**`clip_transmission_shapefile`**(*paths*, *param*)

This function clips the shapefile of the transmission lines using the shapefile of the scope. MULTILINESTRING instances are formed as a result of clipping. Hence, the script cleans the clipped transmission file by replacing the MULTILINESTRING instances with LINESTRING instances.

Parameters

- **`paths`** (*`dict`*) – Dictionary of paths including the path to the shapefile of the transmission network after connecting islands, *grid_connected*, and to the output *grid_clipped*.
- **`param`** – Dictionary of parameters including *CRS_grid*.

Returns The shapefile of clipped transmission lines is saved directly in the desired path *grid_clipped*.

Return type None

`lib.lines_clustering_functions.`**`cluster_transmission_shapefile`**(*paths*, *param*)

This function clusters the transmission network into a specified number of clusters. It first reads the shapefile of voronoi polygons, and initializes its attributes *elec_neighbors*, *trans_lines*, *Area*, *Cap*, and *Ratio*. Starting with the polygon with the highest ratio, it merges it with its electric neighbors with the highest ratio as well. It then updates the values and repeats the algorithm, until the target number of clusters is reached.

Parameters

- **`paths`** (*`dict`*) – Dictionary of paths pointing to *grid_clipped*, *grid_debugging*, *grid_voronoi*, and to the outputs *grid_intermediate* and *grid_regions*.
- **`param`** (*`dict`*) – Dictionary containing the parameter *CRS_grid* and the user preferences *number_clusters* and *intermediate_number*.

Returns The intermediate and final outputs are saved directly in the desired shapefiles.

Return type None

`lib.lines_clustering_functions.`**`connect_islands`**(*paths*, *param*)

This script takes a shapefile of a transmission network and uses graph theory to identify its components (electric islands). After the identification of islands, it creates connections between them using additional transmission lines with low capacities. This correction assumes that there are actually no electric islands, and that multiple graph components only exist because some transmission lines are missing in the data. The output is a shapefile of transmission lines with no electric islands.

Parameters

- **`paths`** (*`dict`*) – Dictionary of paths including the path to the input shapefile of the transmission network, *grid_input*.
- **`param`** – Dictionary of parameters including *CRS_grid*, *default_cap_MVA*, and *default_line_type*.

Returns The shapefile with connections between islands is saved directly in the desired path *grid_connected*.

Return type None

`lib.lines_clustering_functions.`**`create_voronoi_polygons`**(*paths*, *param*)

This function creates a shapefile of voronoi polygons based on the points at the start/end of the lines.

Parameters

- **`paths`** (*`dict`*) – Dictionary of paths including the path to the shapefile of the transmission network after clipping, *grid_clipped*, to the scope *spatial_scope*, and to the output *grid_voronoi*.
- **`param`** – Dictionary of parameters including *CRS_grid*.

Returns The shapefile of voronoi polygons is saved directly in the desired path *grid_voronoi*.

Return type None

`lib.lines_clustering_functions.`**`lines_clustering`**(*paths*, *param*)

This function applies the hierarchical clustering algorithm to the shapefile of transmission lines. It first ensures that the whole grid is one component (no electric islands), by eventually adding fake lines with low capacity. Then it clips the grid to the scope, and creates a shapefile of voronoi polygons based on the points at the start/end of the lines. Regions with a small area and high connectivity to their neighbors are aggregated together, until the target number of regions is reached.

Parameters

- **`paths`** (*`dict`*) – Dictionary of paths pointing to input files and output locations.
- **`param`** (*`dict`*) – Dictionary of parameters including transmission-line-related parameters.

Returns The called functions `connect_islands`, `clip_transmission_shapefile`, `create_voronoi_polygons`, and `cluster_transmission_shapefile` generate outputs.

Return type None

`lib.lines_clustering_functions.`**`update_values_in_geodataframes`**(*gdf_trans*, *gdf_voronoi*, *poly1*, *poly2*, *cluster_no*)

This function updates the values in the geodataframes *gdf_trans* and *gdf_voronoi* after dissolving the polygons and is called in the loops in `cluster_transmission_shapefile`.

Parameters

- **`gdf_trans`** (*`geopandas GeoDataFrame`*) – Geodataframe of transmission lines, containing the columns *Start_poly* and *End_poly*.
- **`gdf_voronoi`** (*`geopandas GeoDataFrame`*) – Geodataframe of polygons, containing the columns *trans_lines*, *elec_neighbors*, *Cluster*, *Cap*, *Area*, and *Ratio*.

- **poly1** (*geopandas GeoDataFrame*) – First polygon to be dissolved, containing the same columns as gdf_voronoi.
- **poly2** (*geopandas GeoDataFrame*) – Second polygon to be dissolved, containing the same columns as gdf_voronoi.
- **cluster_no** (*integer*) – Cluster number to be used for the dissolved polygons.

Return (gdf_trans, gdf_voronoi) Updated geodataframes after dissolving poly1 and poly2.

Return type tuple of geodataframes

Helping functions for the models are included in `spatial_functions.py`, and `util.py`.

3.6 spatial_functions.py

`lib.spatial_functions.`**`array2raster`**(*newRasterfn*, *array*, *rasterOrigin*, *param*)
This function saves array to geotiff raster format (used in cutting with shapefiles).

Parameters

- **newRasterfn** (*string*) – Output path of the raster.
- **array** (*numpy array*) – Array to be converted into a raster.
- **rasterOrigin** (*list of two floats*) – Latitude and longitude of the Northwestern corner of the raster.
- **param** (*dict*) – Dictionary of parameters including *GeoRef* and *CRS*.

Returns The raster file will be saved in the desired path *newRasterfn*.

Return type None

`lib.spatial_functions.`**`array_to_raster`**(*array*, *destination_file*, *input_raster_file*)
This function changes from array back to raster (used after kmeans algorithm).

Parameters

- **array** (*numpy array*) – The array which needs to be converted into a raster.
- **destination_file** (*string*) – The file name where the created raster file is saved.
- **input_raster_file** (*string*) – The original input raster file from which the original coordinates are taken to convert the array back to raster.

Returns The raster file will be saved in the desired path *destination_file*.

Return type None

`lib.spatial_functions.`**`assign_disconnected_components_to_nearest_neighbor`**(*data*, *w*)
This loop is used to force any disconnected group of polygons (graph component) to be assigned to the nearest neighbors.

Parameters

- **data** (*geodataframe*) – The geodataframe of polygons to be clustered.
- **w** (*pysal weights object*) – The pysal weights object of the graph (`w.neighbors` is similar to an adjacency matrix).

Return w The updated pysal weights objected is returned.

Return type pysal weights object

`lib.spatial_functions.`**`calc_geotiff`**(*Crd_all*, *res_desired*)
This function returns a dictionary containing the georeferencing parameters for geotiff creation, based on the desired extent and resolution.

Parameters

- **`Crd_all`** (*`numpy array`*) – Coordinates of the bounding box of the spatial scope.
- **`res_desired`** (*`list`*) – Desired data resolution in the vertical and horizontal dimensions.

Return GeoRef Georeference dictionary containing *RasterOrigin*, *RasterOrigin_alt*, *pixelWidth*, and *pixelHeight*.

Return type dict

`lib.spatial_functions.`**`calc_region`**(*region*, *Crd_reg*, *res_desired*, *GeoRef*)
This function reads the region geometry, and returns a masking raster equal to 1 for pixels within and 0 outside of the region.

Parameters

- **`region`** (*`Geopandas series`*) – Region geometry
- **`Crd_reg`** (*`list`*) – Coordinates of the region
- **`res_desired`** (*`list`*) – Desired high resolution of the output raster
- **`GeoRef`** (*`dict`*) – Georeference dictionary containing *RasterOrigin*, *RasterOrigin_alt*, *pixelWidth*, and *pixelHeight*.

Return A_region Masking raster of the region.

Return type numpy array

`lib.spatial_functions.`**`ckd_nearest`**(*gdf_a*, *gdf_b*, *bcol*)
This function finds the distance and the nearest points in gdf_b for every point in gdf_a.

Parameters

- **`gdf_a`** (*`geodataframe`*) – GeoDataFrame of points, forming a component that is disconnected from *gdf_b*.
- **`gdf_b`** (*`geodataframe`*) – GeoDataFrame of points, forming a component that is disconnected from *gdf_a*.
- **`bcol`** (*`string`*) – Name of column that should be listed in the resulting DataFrame.

Return df Dataframe with the combinations of pair of points as rows, and `'distance'` and `'bcol'` as columns.

Return type pandas dataframe

`lib.spatial_functions.`**`crd_bounding_box`**(*Crd_regions*, *resolution*)
This function calculates coordinates of the bounding box covering data in a given resolution.

Parameters

- **`Crd_regions`** (*`numpy array`*) – Coordinates of the bounding boxes of the regions.
- **`resolution`** (*`numpy array`*) – Data resolution.

Return Crd Coordinates of the bounding box covering the data for each region.

Return type numpy array

`lib.spatial_functions.`**`ind_from_crd`**(*Crd*, *Crd_all*, *res*)
This function converts longitude and latitude coordinates into indices within the spatial scope of the data.

Parameters

- **`Crd`** (*`numpy array`*) – Coordinates to be converted into indices.
- **`Crd_all`** (*`numpy array`*) – Coordinates of the bounding box of the spatial scope.
- **`res`** (*`list`*) – Resolution of the data, for which the indices are produced.

Return Ind Indices within the spatial scope of data.

Return type numpy array

`lib.spatial_functions.`**`polygonize_raster`**(*input_file*, *output_shapefile*, *column_name*)
This function is used to change from a raster to polygons as max-p algorithm only works with polygons.

Parameters

- **`input_file`** (*`string`*) – The path to the file which needs to be converted to a polygon from a raster.
- **`output_shapefile`** (*`string`*) – The path to the shapefile which is generated after polygonization.
- **`column_name`** (*`string`*) – The column name, the values from which are used for conversion.

Returns The shapefile of (multi)polygons is saved directly in the desired path *output_shapefile*.

Return type None

3.7 util.py

`lib.util.`**`create_json`**(*filepath*, *param*, *param_keys*, *paths*, *paths_keys*)
Creates a metadata JSON file containing information about the file in filepath by storing the relevant keys from both the param and path dictionaries.

Parameters

- **`filepath`** (*`string`*) – Path to the file for which the JSON file will be created.
- **`param`** (*`dict`*) – Dictionary of dictionaries containing the user input parameters and intermediate outputs.
- **`param_keys`** (*`list of strings`*) – Keys of the parameters to be extracted from the *param* dictionary and saved into the JSON file.
- **`paths`** (*`dict`*) – Dictionary of dictionaries containing the paths for all files.
- **`paths_keys`** (*`list of strings`*) – Keys of the paths to be extracted from the *paths* dictionary and saved into the JSON file.

Returns The JSON file will be saved in the desired path *filepath*.

Return type None

`lib.util.`**`display_progress`**(*message*, *progress_stat*)
This function displays a progress bar for long computations. To be used as part of a loop or with multiprocessing.

Parameters

- **`message`** (*`string`*) – Message to be displayed with the progress bar.
- **`progress_stat`** (*`tuple(int, int)`*) – Tuple containing the total length of the calculation and the current status or progress.

Returns The status bar is printed.

Return type None

`lib.util.`**`get_x_y_values`**(*paths*)
This function finds the rel_size and rel_std of the four corners of the x,y scatter plot between rel_size and rel_std.

Parameters **`paths`** (*`dict`*) – Dictionary of paths including the path to the CSV file *non_empty_rasters*.

Returns Coordinates of the upper left, upper right, lower left and lower right points of the x,y scatter plot between rel_size and rel_std.

Return type tuple(tuples(int, int))

`lib.util.`**`timecheck`**`(`**args*`)`

This function prints information about the progress of the script by displaying the function currently running, and optionally an input message, with a corresponding timestamp. If more than one argument is passed to the function, it will raise an exception.

Parameters **`args`** (*`string`*) – Message to be displayed with the function name and the timestamp (optional).

Returns The time stamp is printed.

Return type None

Raise Too many arguments have been passed to the function, the maximum is only one string.

Python Module Index

Appendix C

Documentation of tum-ens/pyPRIMA

This is the documentation of a pre-processing tool which automates the creation of energy system models using a common database.

Features

- Aggregation of input data for any user-defined regions that are provided as a shapefile;
- Automation of the pre-processing steps to document assumptions and avoid human errors;
- Cleaning of raw input data and creation of model-independent intermediate files;
- Adaptation of the intermediate files to the energy system models *urbs* and *evrys*.

Applications

This code is useful if:

- You want to create different models using the same input database, but different model regions;
- You want to harmonize the assumptions used in different model frameworks (for comparison and/or coupling);
- You want to generate many models in a short amount of time with fewer human errors.

Contributors

I created the first version of this code with the support of Houssame Houmy. The individual contributions can be traced back in GitHub (Siala and Houmy, 2020).

Contents

Chapter 1

User manual

1.1 Installation

Note: We assume that you are familiar with git and conda.

First, clone the git repository in a directory of your choice using a Command Prompt window:

```
$ ~\directory-of-my-choice> git clone https://github.com/tum-ens/pyPRIMA.git
```

We recommend using conda and installing the environment from the file `gen_mod.yml` that you can find in the repository. In the Command Prompt window, type:

```
$ cd pyPRIMA\env\
$ conda env create -f gen_mod.yml
```

Then activate the environment:

```
$ conda activate gen_mod
```

In the folder `code`, you will find multiple files:

File	Description
config.py	used for configuration, see below.
runme.py	main file, which will be run later using `python runme.py`.
lib\initialization.py	used for initialization.
lib\input_maps.py	used to generate input maps for the scope.
lib\generate-models.py	used to generate the model files from intermediate files.
lib\generate_intermediate_files.py	used to generate intermediate files from raw data.
lib\spatial_functions.py	contains helping functions related to maps, coordinates and indices.
lib\correction_functions.py	contains helping functions for data correction/cleaning.
lib\util.py	contains minor helping functions and the necessary python libraries to be imported.

1.2 config.py

This file contains the user preferences, the links to the input files, and the paths where the outputs should be saved. The paths are initialized in a way that follows a particular folder hierarchy. However, you can change the hierarchy as you wish.

1.2.1 Main configuration function

`config.`**`configuration`**`()`

This function is the main configuration function that calls all the other modules in the code.

Return (paths, param) The dictionary param containing all the user preferences, and the dictionary path containing all the paths to inputs and outputs.

Return type tuple(dict, dict)

`config.`**`general_settings`**`()`

This function creates and initializes the dictionaries param and paths. It also creates global variables for the root folder `root` and the system-dependent file separator `fs`.

Return (paths, param) The empty dictionary paths, and the dictionary param including some general information.

Return type tuple(dict, dict)

Note: Both *param* and *paths* will be updated in the code after running the function `config.configuration`.

Note: `root` points to the directory that contains all the inputs and outputs. All the paths will be defined relatively to the root, which is located in a relative position to the current folder.

The code differentiates between the geographic scope and the subregions of interest. You can run the first part of the script `runme.py` once and save results for the whole scope, and then repeat the second part using different subregions within the scope.

`config.`**`scope_paths_and_parameters`**`(`*paths*, *param*`)`

This function defines the path of the geographic scope of the output *spatial_scope* and of the subregions of interest *subregions*. It also associates two name tags for them, respectively *region_name* and *subregions_name*, which define the names of output folders. Both paths should point to shapefiles of polygons or multipolygons.

For *spatial_scope*, only the bounding box around all the features matters. Example: In case of Europe, whether a shapefile of Europe as one multipolygon, or as a set of multiple features (countries, states, etc.) is used, does not make a difference. Potential maps (theoretical and technical) will be later generated for the whole scope of the bounding box.

For *subregions*, the shapes of the individual features matter, but not their scope. For each individual feature that lies within the scope, you can later generate a summary report and time series. The shapefile of *subregions* does not have to have the same bounding box as *spatial_scope*. In case it is larger, features that lie completely outside the scope will be ignored, whereas those that lie partly inside it will be cropped using the bounding box of *spatial_scope*. In case it is smaller, all features are used with no modification.

year defines the year of the weather/input data, and *model_year* refers to the year to be modeled (could be the same as *year*, or in the future).

technology is a dictionary of the technologies (*Storage*, *Process*) to be used in the model. The names of the technologies should match the names which are used in assumptions_flows.csv, assumptions_processes.csv and assumptions_storage.csv.

Parameters

- **`paths`** (*`dict`*) – Dictionary including the paths.
- **`param`** (*`dict`*) – Dictionary including the user preferences.

Return (paths, param) The updated dictionaries paths and param.

Return type tuple of dict

Note: We recommend using a name tag that describes the scope of the bounding box of the regions of interest. For example, `'Europe'` and `'Europe_without_Switzerland'` will actually lead to the same output for the first part of the code.

1.2.2 User preferences

`config.`**`resolution_parameters`**(*param*)

This function defines the resolution of weather data (low resolution), and the desired resolution of output rasters (high resolution). Both are numpy array with two numbers. The first number is the resolution in the vertical dimension (in degrees of latitude), the second is for the horizontal dimension (in degrees of longitude).

Parameters **`param`** (*`dict`*) – Dictionary including the user preferences.

Return param The updated dictionary param.

Return type dict

Note: As of version 1.0.0, these settings should not be changed. Only MERRA-2 data can be used in the tool. Its spatial resolution is 0.5° of latitudes and 0.625° of longitudes. The high resolution is 15 arcsec in both directions.

`config.`**`load_parameters`**(*param*)

This function defines the user preferences which are related to the load/demand. Currently, only one parameter is used, namely *default_sec_shares*, which sets the reference region to be used in case data for other regions is missing.

Parameters **`param`** (*`dict`*) – Dictionary including the user preferences.

Return param The updated dictionary param.

Return type dict

`config.`**`renewable_time_series_parameters`**(*param*)

This function defines parameters related to the renewable time series to be used in the models. In particular, the user can decide which *modes* to use from the files of the time series, provided they exist. See the repository tum-ens/renewable-timeseries for more information.

Parameters **`param`** (*`dict`*) – Dictionary including the user preferences.

Return param The updated dictionary param.

Return type dict

`config.`**`grid_parameters`**(*param*)

This function defines parameters related to the grid to be used while cleaning the data.

- *quality* is a user assessment of the quality of the data. If the data is trustworthy, use 1, if it is not trustworthy at all, use 0. You can use values inbetween.
- *default* is a collection of default values for voltage, wires, cables, and frequency, to use when these data are missing.

Parameters **`param`** (*`dict`*) – Dictionary including the user preferences.

Return param The updated dictionary param.

Return type dict

`config.`**`processes_parameters`**(*param*)

This function defines parameters related to the processes in general, and to distributed renewable capacities in particular.

For *process*, only the parameter *cohorts* is currently used. It defines how power plants should be grouped according to their construction period. If *cohorts* is 5, then you will have groups of coal power plants from 1960, then another from 1965, and so on. If you do not wish to group the power plants, use the value 1.

For distributed renewable capacities, *dist_ren*, the following parameters are needed:

- *units* is a dictionary defining the standard power plant size for each distributed renewable technology in MW.
- *randomness* is a value between 0 and 1, defining the randomness of the spatial distribution of renewable capacities. The complementary value (1 - randomness) is affected by the values of the potential raster used for the distribution. When using a high resolution map, set *randomness* at a high level (close to 1), otherwise all the power plants will be located in a small area of high potential, close to each other.
- *default_pa_type* and *default_pa_availability* are two arrays defining the availability for each type of protected land. These arrays are used as default, along with the protected areas raster, in case no potential map is available for a distributed renewable technology.

Parameters **`param`** (`dict`) – Dictionary including the user preferences.

Return param The updated dictionary param.

Return type dict

1.2.3 Paths

`config.`**`assumption_paths`**(*paths*)

This function defines the paths for the assumption files and the dictionaries.

- *assumptions_landuse* is a table with land use types as rows and sectors as columns. The table is filled with values between 0 and 1, so that each row has a total of 0 (no sectoral load there) or 1 (if there is a load, it will be distributed according to the shares of each sector).
- *assumptions_flows* is a table with the following columns:
  - *year*: data reference year.
  - *Process/Storage*: name of the process or storage type.
  - *Direction*: either `In` or `Out`. You can have multiple inputs and outputs, each one in a separate row.
  - *Commodity*: name of the input or output commodity.
  - *ratio*: ratio to the throughput, which is an intermediate level between input and output. It could be any positive value. The ratio of the output to the input corresponds to the efficiency.
  - *ratio-min* similar to *ratio*, but at partial load.
- *assumptions_processes* is a table with the following columns:
  - *year*: data reference year.
  - *Process*: name of the process usually given as the technology type.
  - *cap-lo*: minimum power capacity.
  - *cap-up*: maximum power capacity.
  - *max-grad*: maximum allowed power gradient (1/h) relative to power capacity.
  - *min-fraction*: minimum load fraction at which the process can run at.
  - *inv-cost*: total investment cost per power capacity (Euro/MW). It will be annualized in the model using an annuity factor derived from the wacc and depreciation period.
  - *fix-cost*: annual operation independent or fix cost (Euro/MW/a)
  - *var-cost*: variable cost per throughput energy unit(Euro/MWh) but excludes fuel costs.

  - *start-cost*: startup cost when the process is switch on from the off condition.
  - *wacc*: weighted average cost of capital. Percentage of cost of capital after taxes.
  - *depreciation*: deprecation period in years.
  - *lifetime*: lifetime of already installed capacity in years.
  - *area-per-cap*: area required per power capacity (m^2/MW).
  - *act-up*: maximal load (per unit).
  - *act-lo*: minimal load (per unit).
  - *on-off*: binary variable, 1 for controllable power plants, otherwise 0 (must-run).
  - *reserve-cost*: cost of power reserves (Euro/MW) (to be verified).
  - *ru*: ramp-up capacity (MW/MWp/min).
  - *rd*: ramp-down capacity (MW/MWp/min).
  - *rumax*: maximal ramp-up (MW/MWp/h).
  - *rdmax*: minimal ramp-up (MW/MWp/h).
  - *detail*: level of detail for modeling thermal power plants, modes 1-5.
  - *lambda*: cooling coefficient (to be verified).
  - *heatmax*: maximal heating capacity (to be verified).
  - *maxdeltaT*: maximal temperature gradient (to be verified).
  - *heatupcost*: costs of heating up (Euro/MWh_th) (to be verified).
  - *su*: ramp-up at start (to be verified).
  - *sd*: ramp-down at switch-off (to be verified).
  - *pdt*: (to be verified).
  - *hotstart*: (to be verified).
  - *pot*: (to be verified).
  - *pretemp*: temperature at the initial time step, per unit of the maximum operating temperature.
  - *preheat*: heat content at the initial time step, per unit of the maximum operating heat content.
  - *prestate*: operating state at the initial time step (binary).
  - *prepow*: available power at the initial time step (MW) (to be verified).
  - *precaponline*: online capacity at the initial time step (MW).
  - *year_mu*: average construction year for that type of power plants.
  - *year_stdev*: standard deviation from the average construction year for that type of power plants.

- *assumptions_storage* is a table with the following columns:
  - *year*: data reference year.
  - *Storage*: name of the storage usually given as the technology type.
  - *ep-ratio*: fixed energy to power ratio (hours).
  - *cap-up-c*: maximum allowed energy capacity (MWh)
  - *cap-up-p*: maximum allowed power capacity (MW)
  - *inv-cost-p*: total investment cost per power capacity (Euro/MW). It will be annualized in the model using an annuity factor derived from the wacc and depreciation period.
  - *inv-cost-c*: total investment cost per energy capacity (Euro/MWh). It will be annualized in the model using an annuity factor derived from the wacc and depreciation period.

  - *fix-cost-p*: annual operation independent or fix cost per power capacity (Euro/MW/a)
  - *fix-cost-c*: annual operation independent or fix cost per energy capacity (Euro/MWh/a)
  - *var-cost-p*: opertion dependent costs for input and output of energy per MWh_out stored or retreived (euro/MWh)
  - *var-cost-c*: operation dependent costs per MWh stored. This value can used to model technologies that have increased wear and tear proportional to the amount of stored energy.
  - *lifetime*: lifetime of already installed capacity in years.
  - *depreciation*: deprecation period in years.
  - *wacc*: weighted average cost of capital. Percentage of cost of capital after taxes.
  - *init*: initial storage content. Fraction of storage capacity that is full at the simulation start. This level has to be reached in the final timestep.
  - *var-cost-pi*: variable costs for charing (Euro/MW).
  - *var-cost-po*: variable costs for discharing (Euro/MW).
  - *act-lo-pi*: minimal share of active capacity for charging (per unit).
  - *act-up-pi*: maximal share of active capacity for charging (per unit).
  - *act-lo-po*: minimal share of active capacity for discharging (per unit).
  - *act-up-po*: maximal share of active capacity for discharging (per unit).
  - *act-lo-c*: minimal share of storage capacity (per unit).
  - *act-up-c*: maximal share of storage capacity (per unit).
  - *precont*: energy content of the storage unit at the initial time step (MWh) (to be verified).
  - *prepowin*: energy stored at the initial time step (MW).
  - *prepowout*: energy discharged at the initial time step (MW).
  - *ru*: ramp-up capacity (MW/MWp/min).
  - *rd*: ramp-down capacity (MW/MWp/min).
  - *rumax*: maximal ramp-up (MW/MWp/h).
  - *rdmax*: minimal ramp-up (MW/MWp/h).
  - *seasonal*: binary variable, 1 for seasonal storage.
  - *ctr*: binary variable, 1 if can be used for secondary reserve.
  - *discharge*: energy losses due to self-discharge per hour as a percentage of the energy capacity.
  - *year_mu*: average construction year for that type of storage.
  - *year_stdev*: standard deviation from the average construction year for that type of storage.

- *assumptions_commodities* is a table with the following columns:
  - *year*: data reference year.
  - *Commodity*: name of the commodity.
  - *Type_urbs*: type of the commodity according to urbs' terminology.
  - *Type_evrys*: type of the commodity according to evrys' terminology.
  - *price*: commodity price (euro/MWh).
  - *max*: maximum annual commodity use (MWh).
  - *maxperhour*: maximum commodity use per hour (MW).
  - *annual*: total value per year (MWh).

- *losses*: losses (to be verified).

- *assumptions_transmission* is a table with the following columns:
  - *Type*: type of transmission.
  - *length_limit_km*: maximum length of the transmission line in km, for which the assumptions are valid.
  - *year*: data reference year.
  - *Commodity*: name of the commodity to be transported along the transmission line.
  - *eff_per_1000km*: transmission efficiency after 1000km in percent.
  - *inv-cost-fix*: length independent investment cost (euro).
  - *inv-cost-length*: length dependent investment cost (euro/km).
  - *fix-cost-length*: fixed annual cost dependent on the length of the line (euro/km/a).
  - *var-cost*: variable costs per energy unit transmitted (euro/MWh)
  - *cap-lo*: minimum required power capacity (MW).
  - *cap-up*: maximum allowed power capacity (MW).
  - *wacc*: weighted average cost of capital. Percentage of cost of capital after taxes.
  - *depreciation*: deprecation period in years.
  - *act-lo*: minimum capacity (MW/MWp).
  - *act-up*: maximum capacity (MW/MWp).
  - *angle-up*: maximum phase angle ramp-up (to be verified).
  - *PSTmax*: maximum phase angle difference.

- *dict_season_north* is a table with the following columns:
  - *Month*: number of the month (1-12).
  - *Season*: corresponding season.

- *dict_daytype* is a table with the following columns:
  - *Weak day*: name of the weekday (Monday-Sunday).
  - *Type*: either *Working day*, *Saturday*, or *Sunday*.

- *dict_sectors* is a table with the following columns:
  - *EUROSTAT*: name of the entry in the EUROSTAT table.
  - *Model_sectors*: corresponding sector (leave empty if irrelevant).

- *dict_counties* is a table with the following columns:
  - *IRENA*: names of countries in IRENA database.
  - *Counties shapefile*: names of countries in the countries shapefile.
  - *NAME_SHORT*: code names for the countries as used by the code.
  - *ENTSO-E*: names of countries in the ENTSO-E dataset.
  - *EUROSTAT*: names of countries in the EUROSTAT table.

- *dict_line_voltage* is a table with the following columns:
  - *voltage_kV*: sorted values of possible line voltages.
  - *specific_impedance_Ohm_per_km*: specific impedance (leave empty if unknown).
  - *loadability*: loadability factor according to the St Clair's curve (leave empty if unknown).

  - *SIL_MWh*: corresponding surge impedance load (leave empty if unknown).
- *dict_technologies* is a table with the following columns:
  - *IRENA*: names of technologies in the IRENA database.
  - *FRESNA*: names of technologies in the FRESNA database.
  - *Model names*: names of technologies as used in the model.

Parameters **`paths`** (*`dict`*) – Dictionary including the paths.

Returns The updated dictionary paths.

Return type dict

`config.`**`load_input_paths`**(*paths*)

This function defines the paths where the load related inputs are saved:

- *sector_shares* for the sectoral shares in the annual electricity demand.
- *load_ts* for the load time series.
- *profiles* for the sectoral load profiles.

Parameters **`paths`** (*`dict`*) – Dictionary including the paths.

Return paths The updated dictionary paths.

Return type dict

`config.`**`renewable_time_series_paths`**(*paths*, *param*)

This function defines the paths where the renewable time series (inputs) are located. *TS_ren* is itself a dictionary with the keys *WindOn*, *WindOff*, *PV*, *CSP* pointing to the individual files for each technology.

Parameters

- **`paths`** (*`dict`*) – Dictionary including the paths.
- **`param`** (*`dict`*) – Dictionary including the parameters *region_name*, *subregions_name*, and *year*.

Return paths The updated dictionary paths.

Return type dict

`config.`**`grid_input_paths`**(*paths*)

This function defines the paths where the transmission lines (inputs) are located.

Parameters **`paths`** (*`dict`*) – Dictionary including the paths.

Return paths The updated dictionary paths.

Return type dict

`config.`**`processes_input_paths`**(*paths*, *param*)

This function defines the paths where the process-related inputs are located:

- *IRENA*: IRENA electricity statistics (useful to derive installed capacities of renewable energy technologies).
- *dist_ren*: dictionary of paths to rasters defining how the potential for the renewable energy is spatially distributed. The rasters have to be the same size as the spatial scope.
- *FRESNA*: path to the locally saved FRESNA database.

Parameters

- **`paths`** (*`dict`*) – Dictionary including the paths.
- **`param`** (*`dict`*) – Dictionary including the parameter *year*.

Return paths The updated dictionary paths.

Return type dict

config.**output_folders**(*paths*, *param*)

This function defines the paths to multiple output folders:

- *region* is the main output folder.
- *local_maps* is the output folder for the local maps of the spatial scope.
- *sites* is the output folder for the files related to the modeled sites.
- *load* is the output folder for the subregions-independent, load-related intermediate files.
- *load_sub* is the output folder for the subregions-dependent, load-related intermediate files.
- *grid* is the output folder for the subregions-independent, grid-related intermediate files.
- *grid_sub* is the output folder for the subregions-dependent, grid-related intermediate files.
- *regional_analysis* is the output folder for the regional analysis of renewable energy.
- *proc* is the output folder for the subregions-independent, process-related intermediate files.
- *proc_sub* is the output folder for the subregions-dependent, process-related intermediate files.
- *urbs* is the output folder for the urbs model input file.
- *evrys* is the output folder for the evrys model input files.

All the folders are created at the beginning of the calculation, if they do not already exist.

Parameters

- **paths** (*dict*) – Dictionary including the paths.
- **param** (*dict*) – Dictionary including the user preferences *region_name* and *subregions_name*.

Returns The updated dictionary paths.

Return type dict

config.**output_paths**(*paths*, *param*)

This function defines the paths to multiple output files.

Sites:

- *sites_sub* is the CSV output file listing the modeled sites and their attributes.

Load:

- *stats_countries* is the CSV output file listing some load statistics on a country level.
- *load_ts_clean* is the CSV output file with cleaned load time series on a country level.
- *cleaned_profiles* is a dictionary of paths to the CSV file with cleaned load profiles for each sector.
- *df_sector* is the CSV output file with load time series for each sector on a country level.
- *load_sector* is the CSV output file with yearly electricity demand for each sector and country.
- *load_landuse* is the CSV output file with load time series for each land use type on a country level.
- *intersection_subregions_countries* is a shapefile where the polygons are the outcome of the intersection between the countries and the subregions.
- *stats_country_parts* is the CSV output file listing some load statistics on the level of country parts.
- *load_ts_clean* is the CSV output file with load time series on the level of subregions.

Grid:

- *grid_expanded* is a CSV file including a reformatted table of transmission lines.
- *grid_filtered* is a CSV file obtained after filtering out erronous/useless data points.

- *grid_corrected* is a CSV file obtained after correcting erronous data points.
- *grid_filled* is a CSV file obtained after filling missing data with default values.
- *grid_cleaned* is a CSV file obtained after cleaning the data and reformatting the table.
- *grid_shp* is a shapefile of the transmission lines.
- *grid_completed* is a CSV file containing the aggregated transmission lines between the subregions and their attributes.

Renewable processes:

- *IRENA_summary* is a CSV file with a summary of renewable energy statistics for the countries within the scope.
- *locations_ren* is a dictionary of paths pointing to shapefiles of possible spatial distributions of renewable power plants.
- *potential_ren* is a CSV file with renewable potentials.

Other processes and storage:

- *process_raw* is a CSV file including aggregated information about the power plants before processing it.
- *process_filtered* is a CSV file obtained after filtering out erronous/useless data points.
- *process_joined* is a CSV file obtained after joining the table with default attribute assumptions (like costs).
- *process_completed* is a CSV file obtained after filling missing data with default values.
- *process_cleaned* is a CSV file obtained after cleaning the data and reformatting the table.
- *process_regions* is a CSV file containing the power plants for each subregion.
- *storage_regions* is a CSV file containing the storage devices for each subregion.
- *commodities_regions* is a CSV file containing the commodities for each subregion.

Framework models:

- *urbs_model* is the urbs model input file.
- *evrys_model* is the evrys model input file.

Parameters

- **`paths`** (`dict`) – Dictionary including the paths.
- **`param`** (`dict`) – Dictionary including the user preferences *region_name*, *subregions_name*, and *year*.

Returns The updated dictionary paths.

Return type dict

`config.`**`local_maps_paths`**(*paths*, *param*)

This function defines the paths where the local maps will be saved:

- *LAND* for the raster of land areas within the scope
- *EEZ* for the raster of sea areas within the scope
- *LU* for the land use raster within the scope
- *PA* for the raster of protected areas within the scope
- *POP* for the population raster within the scope

Parameters

- **`paths`** (`dict`) – Dictionary including the paths.

- **param** (*dict*) – Dictionary including the user preferences.

Return paths The updated dictionary paths.

Return type dict

1.3 runme.py

`runme.py` calls the main functions of the code:

```python
from lib.initialization import initialization
from lib.generate_intermediate_files import *
from lib.correction_functions import *
from lib.generate_models import *

if __name__ == "__main__":
    paths, param = initialization()

    ## Clean raw data
    clean_residential_load_profile(paths, param)
    clean_commercial_load_profile(paths, param)
    clean_industry_load_profile(paths, param)
    clean_agriculture_load_profile(paths, param)
    clean_streetlight_load_profile(paths, param)
    clean_GridKit_Europe(paths, param)
    clean_sector_shares_Eurostat(paths, param)
    clean_load_data_ENTSOE(paths, param)
    distribute_renewable_capacities_IRENA(paths, param)
    clean_processes_and_storage_FRESNA(paths, param)

    ## Generate intermediate files
    generate_sites_from_shapefile(paths, param)
    generate_load_timeseries(paths, param)
    generate_transmission(paths, param)
    generate_intermittent_supply_timeseries(paths, param)
    generate_processes(paths, param)
    generate_storage(paths, param)
    generate_commodities(paths, param)

    ## Generate model files
    generate_urbs_model(paths, param)
    generate_evrys_model(paths, param)
```

1.4 Recommended input sources

1.4.1 Load time series for countries

ENTSO-E publishes (or used to publish - the service has been discontinued as of November 2019) hourly load profiles for each country in Europe that is part of ENTSO-E.

1.4.2 Sectoral load profiles

The choice of the load profiles is not too critical, since the sectoral load profiles will be scaled according to their shares in the yearly demand, and their shapes edited to match the hourly load profile. Nevertheless, examples of load profiles for Germany can be obtained from the BDEW.

1.4.3 Sector shares (demand)

The sectoral shares of the annual electricity demand can be obtained from Eurostat. The table reference is *nrg_105a*. Follow these instructions to obtain the file as needed by the code:

- GEO: Choose all countries, but not EU
- INDIC_NRG: Choose all indices
- PRODUCT: Electrical energy (code 6000)
- TIME: Choose years
- UNIT: GWh
- Download in one single csv file

1.4.4 Power plants and storage units

The powerplantmatching package within FRESNA extracts a standardized power plant database that combines several other databases covering Europe. In this repository, all non-renewable power plants, all storage units, and some renewable power plants (e.g. geothermal) are obtained from this database. Since the capacities for most renewable technologies are inaccurate, they are obtained from another source (see below).

1.4.5 Renewable installed capacities

Renewable electricity capacity and generation statistics are obtained from the Query Tool of IRENA. The user has to create a query that includes all countries (but no groups of countries, such as continents), all technologies (but no groups of technology) for a particular year and name the file `IRENA_RE_electricity_statistics_allcountries_alltech_YEAR.csv`. This dataset has a global coverage, however it does not provide the exact location of each project. The code includes an algorithm to distribute the renewable capacities spatially.

1.4.6 Renewable potential maps

These maps are needed to distribute the renewable capacities spatially, since IRENA does not provide their exact locations. You can use any potential maps, provided that they have the same extent as the geographic scope. Adjust the resolution parameters in `config.py` accordingly. Such maps can be generated using the GitHub repository tum-ens/pyGRETA.

1.4.7 Renewable time series

Similarly, the renewable time series can be generated using the GitHub repository tum-ens/pyGRETA. This repository is particularly is the model regions are unconventional.

1.4.8 Transmission lines

High-voltage power grid data for Europe and North America can be obtained from GridKit, which used OpenStreetMap as a primary data source. In this repository, we only use the file with the lines (links.csv). In general, the minimum requirements for any data source are that the coordinates for the line vertices and the voltage are provided.

1.4.9 Other assumptions

Currently, other assumptions are provided in tables filled by the modelers. Ideally, machine-readable datasets providing the missing information are collected and new modules are written to read them and extract that information.

1.5 Recommended workflow

The script is designed to be modular and split into three main modules: `lib.correction_functions`, `lib.generate_intermediate_files`, and `lib.generate_models`.

Warning: The outputs of each module serve as inputs to the following module. Therefore, the user will have to run the script sequentially.

The recommended use cases of each module will be presented in the order in which the user will have to run them.

1. *Correction and cleaning of raw input data*
2. *Generation of intermediate files*
3. *Generation of model input files*

The use cases associated with each module are presented below.

It is recommended to thoroughly read through the configuration file *config.py* and modify the input paths and computation parameters before starting the *runme.py* script. Once the configuration file is set, open the *runme.py* file to define what use case you will be using the script for.

1.5.1 Correction and cleaning of raw input data

Each function in this module is designed for a specific data set (usually mentioned at the end of the function name). The pre-processing steps include filtering, filling in missing values, correcting/overwriting erronous values, aggregating and disaggregating entries, and deleting/converting/renaming the attributes.

At this stage, the obtained files are valid for the whole geographic scope, and do not depend on the model regions.

1.5.2 Generation of intermediate files

The functions in this module read the cleaned input data, and adapts it to the model regions. They also expand the attributes based on assumptions to cover all the data needs of all the supported models. The results are saved in individual CSV files that are model-independent. These files can be shared with modelers whose models are not supported, and they might be able to adjust them according to their model input requirements, and use them.

1.5.3 Generation of model input files

Here, the input files are adapted to the requirements of the supported model frameworks (currently urbs and evrys). Input files as needed by the scripts of urbs and evrys are generated at the end of this step.

Chapter 2

Theory

This chapters explains how the load time series are disaggregated spatially and according to sectors, then aggregated again according to the desired model regions.

2.1 Purpose

Load time series are widely available, but the published datasets are usually restricted to predefined spatial regions such as countries and their administrative subdivisions. The generate_load_timeseries() function takes the datasets which are available for these regions and disaggregate them according to a set of parameters, before aggregating them at a different spatial level. It is then possible to obtain time series for any region.

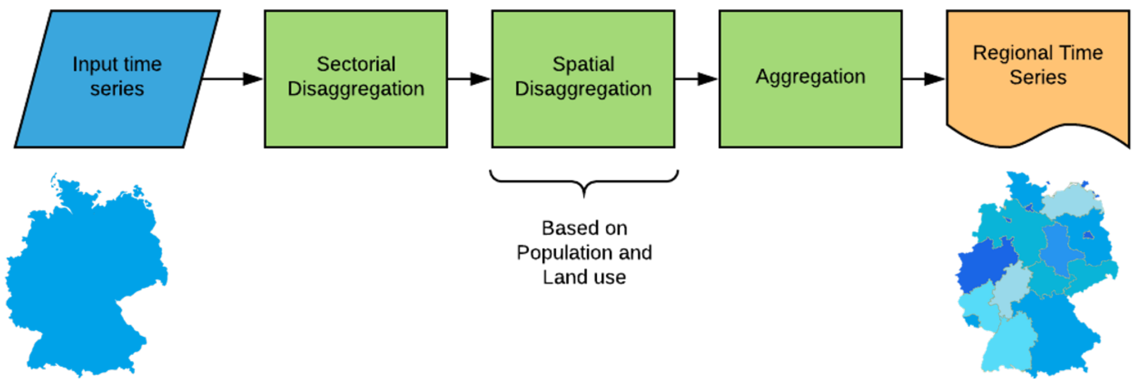


Fig. 1: Description of lib.generate_intermediate_files.generate_load_timeseries

2.2 Inputs

The main inputs of this script are:

- Load time series of the countries or regions to be disaggregated (hourly).
- Shapefiles of the countries or regions to be disaggregated
- Shapefiles of the regions of interest (subregions)
- Assumptions (land use and sector correspondence)
- Load profiles of the different sectors
- Raster of the population and land use correspondent to the country or region

2.3 Sectoral disaggregation

The load is assumed to be perfectly divided into four distinct sectors (load sources):

- Commercial
- Industrial
- Residential
- Agricultural

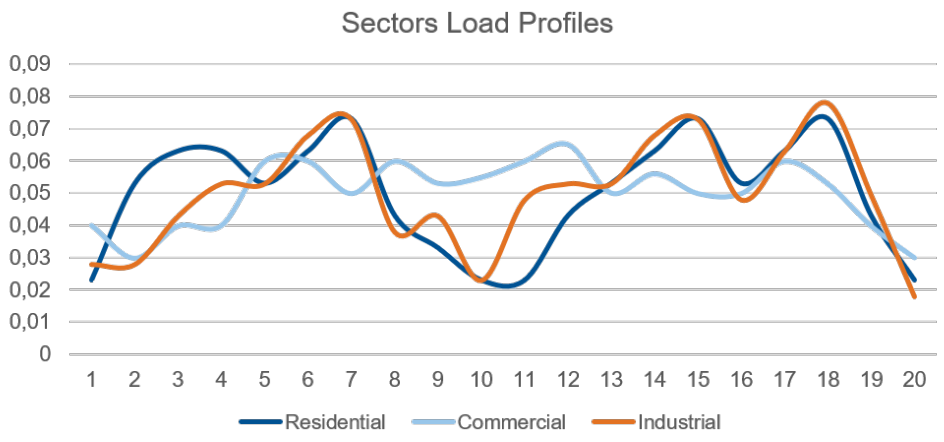


Fig. 2: Sectoral load profiles

Sectoral load shares:

Region	Industry	Commerce	Residential	Agriculture
A	0.41%	0.28%	0.29%	0.02%
B	0.31%	0.30%	0.38%	0.01%
C	0.44%	0.30%	0.25%	0.01%

An hourly load profile for each sector has been predefined for one week, the same load profile is assumed to repeat over the year. These load profiles are scaled and normalized based on sectoral load shares for each region(assumed to be constant throughout the spatial scope), by multiplying the load profiles by their corresponding share and normalizing their hourly sum to be equal to 1.

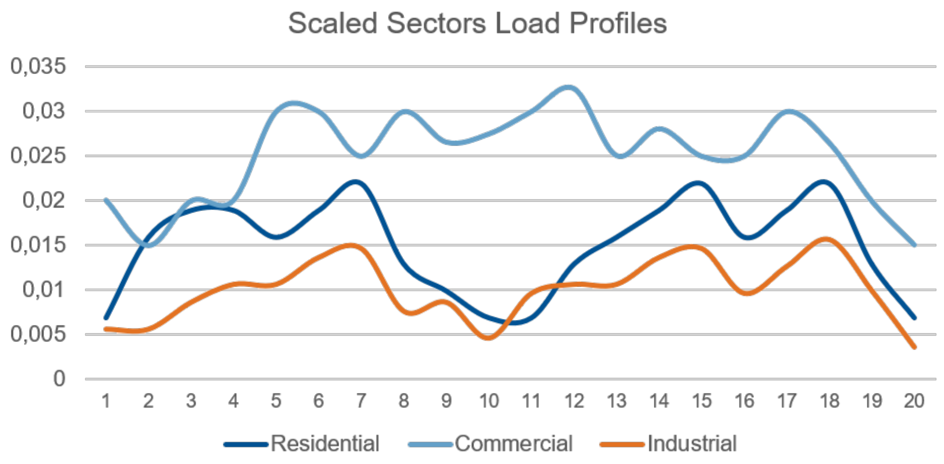


Fig. 3: Scaled sectoral load profiles

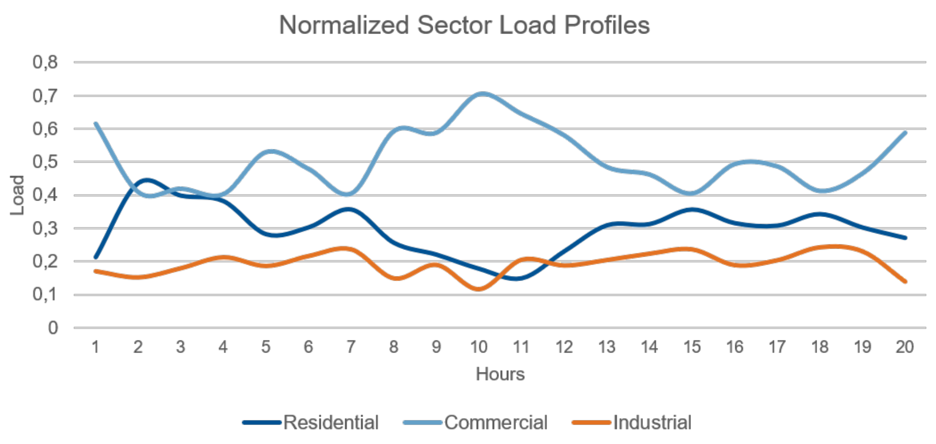


Fig. 4: Normalized sectoral load profiles

Once the load profiles are normalized, we can multiply them with the actual load time series to obtain the load timeseries for each sector.

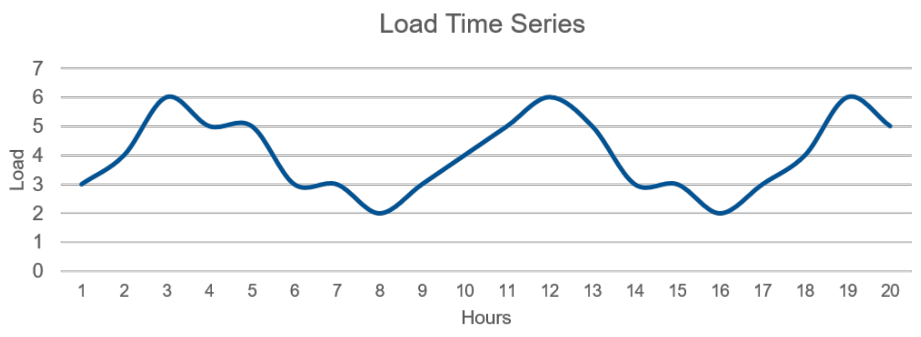


Fig. 5: Load time series

2.4 Spatial disaggregation

The next step is the spatial disaggregation, based on the land use and population concentration rasters. First, each land-use type is assigned a sectoral load percentage corresponding to the load components of the land use category. Then, the population concentration raster is used to calculate the population of each pixel.

Counting the pixels, and land use occurrences inside of region for which the sectoral load timeseries has been calculated, the sectoral load for the Industry, commercial, and agricultural can be retrieved for each pixel of that region. Similarly, the residential load timeseries can be assigned to each pixel based on the population contained in the said pixel. The spatial disaggregation results in the assignment to every pixel inside a given region to be assigned a specific sectoral load timeseries.

2.5 Re-aggregation

The result of the sectoral and spatial disaggregation, performed in the first two sections can be used to retrieve the sectoral load timeseries and, therefore, the general load time series of any desired region by summing up the loads of every pixel contained within the region. If a subregion spans more than one region or country, it is divided into subregions restrained to each of those countries.

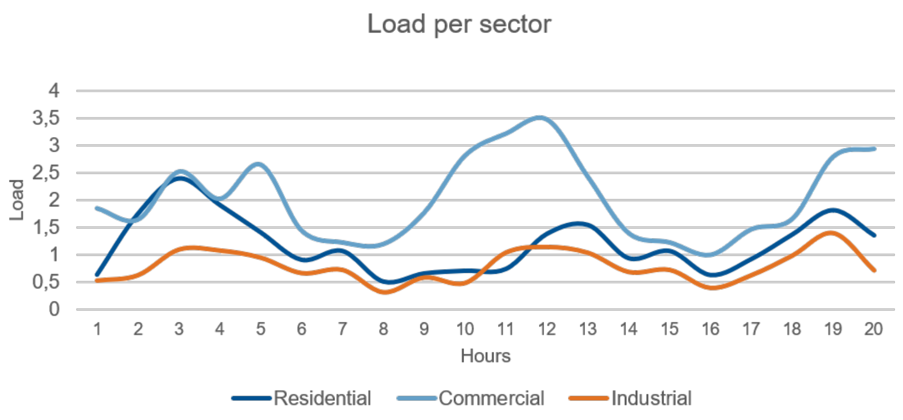


Fig. 6: Sectoral load time series

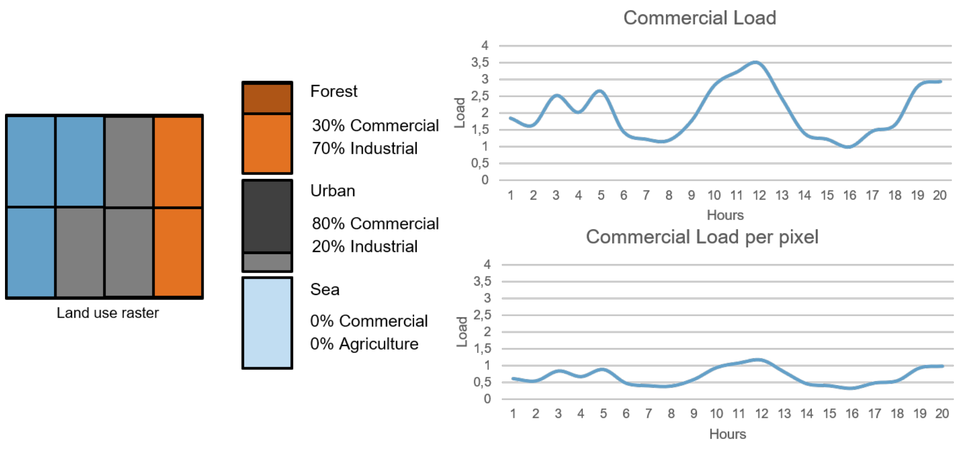


Fig. 7: Example - Commerce sector spatial disaggregation

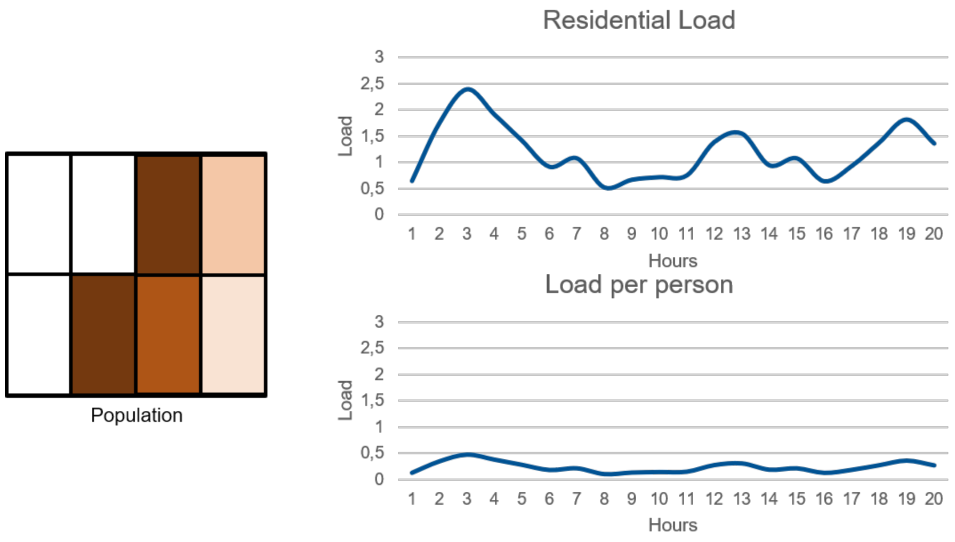


Fig. 8: Example - Residential sector spatial disaggregation

Chapter 3

Implementation

Start with the configuration:

You can run the code by typing:

```
$ python runme.py
```

`runme.py` calls the main functions of the code, which are explained in the following sections.

3.1 initialization.py

`lib.initialization.`**`initialization`**`()`

This function reads the user-defined parameters and paths from `config.py`, then adds additional parameters related to the shapefiles. First, it saves the spatial scope of the problem. Then, it distinguishes between countries, exclusive economic zones and subregions. For each one of them, it saves the geodataframes, the number of features, and the coordinates of the bounding boxes of each feature. Finally, it saves the number of rows and columns in the low and righ resolution, and a georeference dictionary used for saving tif files. It also creates the rasters *LAND* and *EEZ* to be used later.

Returns The updated dictionaries param and paths.

Return type tuple(dict, dict)

Helping functions for the models are included in `generate_intermediate_files.py`, `correction_functions.py`, `spatial_functions.py`, and `input_maps.py`.

3.2 generate_intermediate_files.py

`lib.generate_intermediate_files.`**`generate_commodities`**`(`*paths*, *param*`)`

This function reads the assumptions related to the flows and commodities and filters them based on the needs of the user. Then it applies them to all the combinations of sites and commodities which are used in the model. It also uses the output of the module `generate_load_timeseries` to derive the annual demand for each site. Finally, the output is saved in a model-neutral CSV file containing all the information about the commodities.

Parameters

- **`paths`** (*`dict`*) – Dictionary containing the paths to *assumptions_commodities*, *assumptions_flows", *sites_sub*, *load_regions*, as well as the output *commodities_regions*.
- **`param`** (*`dict`*) – Dictionary containing the user preferences *model_year* and *technology*.

Returns The CSV file with the commodities for each region is saved directly in the desired path, along with its metadata in a JSON file.

Return type None

`lib.generate_intermediate_files.`**`generate_intermittent_supply_timeseries`**(*paths*, *param*)

This function reads the time series from CSV files generated by the renewable-timeseries tool. It then checks whether the desired data is available, formats the data into a model-neutral format and saves it into a CSV file.

Parameters

- **`paths`** (*`dict`*) – Dictionary including the paths to the generated time series.
- **`param`** (*`dict`*) – Dictionary including the potential time series parameters.

Returns The time series for each region, and for all desired technologies and modes are saved directly in the given path, along with the metadata in a JSON file.

Return type None

Raises

- **`Missing TS`** – The time series file is missing.
- **`Missing mode`** – The desired mode is missing from the time series file.
- **`Subregions missing`** – Some subregions are not present in the time series file, and will be left zero in the output CSV file.

`lib.generate_intermediate_files.`**`generate_load_timeseries`**(*paths*, *param*)

This function reads the normalized sectoral standard load profiles, and the cleaned load time series for the countries in the scope. On one hand, it splits the time series into sectoral time series for each country. On the other hand, it determines the time series for each pixel of land use and for each person in the country, by assuming a relationship between sectors and land use types / population size. Finally, it aggregates the time series of the pixels that lie in the same subregions to obtain the time series for each desired subregion.

Parameters

- **`paths`** (*`dict`*) – Dictionary containing the paths to the cleaned input, to the intermediate files and to the outputs.
- **`param`** (*`dict`*) – Dictionary containing assumptions about the load, as well as the geodataframes for the countries and the subregions.

Returns All the outputs are saved in CSV or SHP files in the defined paths, along with their metadata in JSON files.

Return type None

`lib.generate_intermediate_files.`**`generate_processes`**(*paths*, *param*)

This function reads the assumptions related to the flows and processes and filters them based on the needs of the user. Then it reads the shapefile of processes and storages, and filters out the technologies which are not used in the model. Afterwards, it fills in the attributes of the processes based on the assumptions, removes processes that have exceeded their lifetime, and eventually groups the remaining entries into cohorts based on their construction year. Finally, it expands the list with possible site-power plant combinations.

Parameters

- **`paths`** (*`dict`*) – Dictionary containing the paths to *assumptions_processes*, *assumptions_flows", *sites_sub*, and *process_cleaned*, as well as the output *process_regions*.
- **`param`** (*`dict`*) – Dictionary containing the user preferences *model_year*, *year*, *process*, and *technology*.

Returns The CSV file with the power plants for each region is saved directly in the desired path, along with its metadata in a JSON file.

Return type None

`lib.generate_intermediate_files.`**`generate_sites_from_shapefile`**(*paths*, *param*)

This function reads the geodataframe of the sites and extracts their respective names and areas, as well as the coordinates of their centroids. It adds assumptions about other attributes and saves the output in a CSV file.

Parameters

- **paths** (*dict*) – Dictionary including the paths to the rasters *LAND*, *EEZ*, and to the output *sites_sub*.
- **param** (*dict*) – Dictionary including the geodataframe of regions, and parameters defining the resolution, the coordinates of the scope, and the georeference dictionary.

Returns The CSV file with the sites is saved directly in the desired path, along with the corresponding metadata in a JSON file.

Return type None

`lib.generate_intermediate_files.`**`generate_storage`**(*paths*, *param*)

This function reads the assumptions related to the flows and storages and filters them based on the needs of the user. Then it reads the shapefile of processes and storages, and filters out the technologies which are not used in the model. Afterwards, it fills in the attributes of the storages based on the assumptions, removes storages that have exceeded their lifetime, and eventually groups the remaining entries into cohorts based on their construction year. Finally, it expands the list with possible site-storage combinations.

Parameters

- **paths** (*dict*) – Dictionary containing the paths to *assumptions_storage*, *assumptions_flows", *sites_sub*, and *process_cleaned*, as well as the output *storage_regions*.
- **param** (*dict*) – Dictionary containing the user preferences *model_year*, *year*, *process*, and *technology*.

Returns The CSV file with the storage units for each region is saved directly in the desired path, along with its metadata in a JSON file.

Return type None

`lib.generate_intermediate_files.`**`generate_transmission`**(*paths*, *param*)

This function reads the cleaned grid data and the shapefile of the subregions. It first determines the names of the regions connected by each line, *Region_start* and *Region_end*, and only keeps those between two different subregions. Then it estimates the length between the centroids of the regions and uses it to estimate the efficiency of the lines and their costs. Finally, it completes the missing attributes with general assumptions and saves the result in a CSV file.

Parameters

- **paths** – Dictionary including the paths to *assumptions_transmission*, *grid_cleaned*, *sites_sub*, *subregions*, *dict_line_voltage*, and the output *grid_completed*.
- **param** (*dict*) – Dictionary including the geodataframe of the subregions and grid-related assumptions.

Returns The CSV file with the completed transmission data is saved directly in the desired path, along with its metadata in a JSON file.

Return type None

3.3 correction_functions.py

`lib.correction_functions.`**`clean_GridKit_Europe`**(*paths*, *param*)

This function reads the raw data from GridKit (Europe). First, it converts the column of locations into separate columns of longitude and latitude of starting and ending points. Then, it expands the dataframe, so that every row would only have one entry in the columns for *voltage*, *wires*, *cables*, and *frequency*. Based on the user judgement of the *quality* of the data, the dataframe is filtered and rows with missing data are filled with most common value. Based on *length_m* and *voltage*, the values for the impedance *X_ohm*, the *loadability* and the surge impedance loading *SIL_MW* are determined.

Parameters

- **`paths`** (*`dict`*) – Dictionary including the paths to the raw database *transmission_lines*, and to the desired intermediate and finally output locations.
- **`param`** (*`dict`*) – Dictionary including the *grid* dictionary.

Returns The cleaned database is saved as a CSV in the path *grid_cleaned* and as a shapefile in the path *grid_shp*, along with the corresponding metadata in JSON files.

Return type None

`lib.correction_functions.`**`clean_IRENA_summary`**(*paths*, *param*)

This function reads the IRENA database, format the output for selected regions and computes the FLH based on the installed capacity and yearly energy production. The results are saved in CSV file.

Parameters

- **`param`** (*`dict`*) – Dictionary of dictionaries containing list of subregions, and year.
- **`paths`** (*`dict`*) – Dictionary of dictionaries containing the paths to the IRENA country name dictionary, and IRENA database.

Returns The CSV file containing the summary of IRENA data for the countries within the scope is saved directly in the desired path, along with the corresponding metadata in a JSON file.

Return type None

`lib.correction_functions.`**`clean_agriculture_load_profile`**(*paths*, *param*)

This function reads the raw standard agricultural profile, repeats it to obtain a full year, normalizes it so that the sum is equal to 1, and saves the obtained load profile in a CSV file.

Parameters

- **`paths`** (*`dict`*) – Dictionary containing the paths to *dict_daytype*, *dict_season*, and to the raw standard load profiles.
- **`param`** (*`dict`*) – Dictionary containing the year of the data.

Returns The outputs are saved in CSV in the defined paths, along with their metadata in JSON files.

Return type None

`lib.correction_functions.`**`clean_commercial_load_profile`**(*paths*, *param*)

This function reads the raw standard commercial profile, repeats it to obtain a full year, normalizes it so that the sum is equal to 1, and saves the obtained load profile in a CSV file.

Parameters

- **`paths`** (*`dict`*) – Dictionary containing the paths to *dict_daytype*, *dict_season*, and to the raw standard load profiles.
- **`param`** (*`dict`*) – Dictionary containing the year of the data.

Returns The outputs are saved in CSV in the defined paths, along with their metadata in JSON files.

Return type None

`lib.correction_functions.`**`clean_industry_load_profile`**(*paths*, *param*)

This function reads the raw standard industrial profile, repeats it to obtain a full year, normalizes it so that the sum is equal to 1, and saves the obtained load profile in a CSV file.

Parameters

- **`paths`** (*`dict`*) – Dictionary containing the paths to *dict_daytype*, *dict_season*, and to the raw standard load profiles.
- **`param`** (*`dict`*) – Dictionary containing the year of the data.

Returns The outputs are saved in CSV in the defined paths, along with their metadata in JSON files.

Return type None

`lib.correction_functions.`**`clean_load_data_ENTSOE`**(*paths*, *param*)
This function reads the raw load time series from ENTSO-E, filters them for the desired year, scales them based on their coverage ratio, renames the countries based on *dict_countries*, and fills missing data by values from the day before (the magnitude is adjusted based on the trend of the previous five hours).

Parameters

- **`paths`** (*`dict`*) – Dictionary containing the paths to the ENTSO-E input, to the dictionary of country names, and to the output.
- **`param`** (*`dict`*) – Dictionary containing information about the year of the data.

Returns The result is saved directly in a CSV file in the desired path, along with its corresponding metadata.

Return type None

`lib.correction_functions.`**`clean_names`**(*text*)
This functions reads a string, removes non-ASCII characters, and shortens it to 64 characters.

Parameters **`text`** (*`string`*) – Input string (e.g. power plant name).

Return text_short The shortened name without non-ASCII characters.

Return type string

`lib.correction_functions.`**`clean_processes_and_storage_FRESNA`**(*paths*, *param*)
This function reads the FRESNA database of power plants, filters it by leaving out the technologies that are not used in the models based on *dict_technologies*, and completes it by joining it with the distributed renewable capacities as provided by `distribute_renewable_capacities_IRENA`. It allocates a type for each power plant, a unique name, a construction year, and coordinates, if these pieces of information are missing. It then saves the result both as a CSV and a shapefile.

- Type: This is derived from the properties *Fueltype*, *Technology*, and *Set* that are provided in FRESNA, in combination with user preferences in *dict_technologies*.
- Name: If a name is missing, the power plant is named `'unnamed'`. A number is added as a suffix to distinguish power plants with the same name. Names do not contain spaces.
- Year: If provided, the year for retrofitting is used instead of the commissioning year. If both are missing, a year is chosen randomly based on a normal distribution for each power plant type. The average and the standard deviation of that distribution is provided by the user in *assumptions_processes* and *assumptions_storage*.
- Coordinates: Only a few power plants are lacking coordinates. The user has the possibility to allocate coordinates for these power plants, otherwise coordinates of a random power plant within the same country are chosen to fill in the missing information.

Parameters

- **`paths`** (*`dict`*) – Dictionary containing the paths to the database *FRESNA*, to user preferences in *dict_technologies*, *assumptions_processes*, *assumptions_storage*, to *locations_ren* for the shapefiles of distributed renewable capacities, and to all the intermediate and final outputs of the module.
- **`param`** (*`dict`*) – Dictionary including information about the reference year of the data, and assumptions related to processes.

Returns The intermediate and final outputs are saved directly as CSV files in the respective path. The final result is also saved as a shapefile of points. The metadata is saved in JSON files.

Return type None

`lib.correction_functions.`**`clean_residential_load_profile`**(*paths*, *param*)

This function reads the raw standard residential profile, repeats it to obtain a full year, normalizes it so that the sum is equal to 1, and saves the obtained load profile in a CSV file.

Parameters

- **`paths`** (*`dict`*) – Dictionary containing the paths to *dict_daytype*, *dict_season*, and to the raw standard load profiles.
- **`param`** (*`dict`*) – Dictionary containing the year of the data.

Returns The outputs are saved in CSV in the defined paths, along with their metadata in JSON files.

Return type None

`lib.correction_functions.`**`clean_sector_shares_Eurostat`**(*paths*, *param*)

This function reads the CSV file with the sector shares from Eurostat (instructions on downloading it are in the documentation), filters it for the desired year and countries, reclassifies the country names and sectors, and normalizes the result.

Parameters

- **`paths`** (*`dict`*) – Dictionary containing the paths to the Eurostat input, to the dictionary of country names, and to the output.
- **`param`** (*`dict`*) – Dictionary containing information about the year of the data.

Returns The result is saved directly in a CSV file in the desired path, along with its corresponding metadata.

Return type None

`lib.correction_functions.`**`clean_streetlight_load_profile`**(*paths*, *param*)

This function reads the raw standard street light profile, repeats it to obtain a full year, normalizes it so that the sum is equal to 1, and saves the obtained load profile in a CSV file.

Parameters

- **`paths`** (*`dict`*) – Dictionary containing the paths to *dict_daytype*, *dict_season*, and to the raw standard load profiles.
- **`param`** (*`dict`*) – Dictionary containing the year of the data.

Returns The outputs are saved in CSV in the defined paths, along with their metadata in JSON files.

Return type None

`lib.correction_functions.`**`distribute_renewable_capacities_IRENA`**(*paths*, *param*)

This function reads the installed capacities of renewable power in each country, and distributes that capacity spatially. In the first part, it generates a summary report of IRENA for the countries within the scope, if such a report does not already exist. It then matches the names of technologies in IRENA with those that are defined by the user. Afterwards how many units or projects exist per country and technology, using a user-defined unit size.

In the second part, it allocates coordinates for each unit, based on a potential raster map and on a random factor. This is done by calling the module `create_shapefiles_of_ren_power_plants` in *`lib.spatial_functions`*.

Parameters

- **paths** (*dict*) – Dictionary containing the paths to *IRENA_summary* and to *dict_technologies*, as well as other paths needed by `create_shapefiles_of_ren_power_plants`.
- **param** (*dict*) – Dictionary containing the dictionary of the size of units for each technology, and other parameters needed by `clean_IRENA_summary` and `create_shapefiles_of_ren_power_plants`.

Returns The submodules `clean_IRENA_summary` and `create_shapefiles_of_ren_power_plants`, which are called by this module, have outputs of their own.

Return type None

3.4 spatial_functions.py

`lib.spatial_functions.`**`array2raster`**(*newRasterfn*, *rasterOrigin*, *pixelWidth*, *pixelHeight*, *array*)

This function saves array to geotiff raster format based on EPSG 4326.

Parameters

- **newRasterfn** (*string*) – Output path of the raster.
- **rasterOrigin** (*list of two floats*) – Latitude and longitude of the North-western corner of the raster.
- **pixelWidth** (*integer*) – Pixel width (might be negative).
- **pixelHeight** (*integer*) – Pixel height (might be negative).
- **array** (*numpy array*) – Array to be converted into a raster.

Returns The raster file will be saved in the desired path *newRasterfn*.

Return type None

`lib.spatial_functions.`**`calc_geotiff`**(*Crd_all*, *res_desired*)

This function returns a dictionary containing the georeferencing parameters for geotiff creation, based on the desired extent and resolution.

Parameters

- **Crd_all** (*numpy array*) – Coordinates of the bounding box of the spatial scope.
- **res_desired** (*list*) – Desired data resolution in the vertical and horizontal dimensions.

Return GeoRef Georeference dictionary containing *RasterOrigin*, *RasterOrigin_alt*, *pixelWidth*, and *pixelHeight*.

Return type dict

`lib.spatial_functions.`**`calc_region`**(*region*, *Crd_reg*, *res_desired*, *GeoRef*)

This function reads the region geometry, and returns a masking raster equal to 1 for pixels within and 0 outside of the region.

Parameters

- **region** (*Geopandas series*) – Region geometry
- **Crd_reg** (*list*) – Coordinates of the region
- **res_desired** (*list*) – Desired high resolution of the output raster
- **GeoRef** (*dict*) – Georeference dictionary containing *RasterOrigin*, *RasterOrigin_alt*, *pixelWidth*, and *pixelHeight*.

Return A_region Masking raster of the region.

Return type numpy array

`lib.spatial_functions.`**`crd_exact_points`**(*Ind_points*, *Crd_all*, *res*)

This function converts indices of points in high resolution rasters into longitude and latitude coordinates.

Parameters

- **`Ind_points`** (*`tuple of arrays`*) – Tuple of arrays of indices in the vertical and horizontal axes.
- **`Crd_all`** (*`numpy array`*) – Array of coordinates of the bounding box of the spatial scope.
- **`res`** (*`list`*) – Data resolution in the vertical and horizontal dimensions.

Return Crd_points Coordinates of the points in the vertical and horizontal dimensions.

Return type list of arrays

`lib.spatial_functions.`**`crd_merra`**(*Crd_regions*, *res_weather*)

This function calculates coordinates of the bounding box covering MERRA-2 data.

Parameters

- **`Crd_regions`** (*`numpy array`*) – Coordinates of the bounding boxes of the regions.
- **`res_weather`** (*`list`*) – Weather data resolution.

Return Crd Coordinates of the bounding box covering MERRA-2 data for each region.

Return type numpy array

`lib.spatial_functions.`**`create_shapefiles_of_ren_power_plants`**(*paths*, *param*, *inst_cap*, *tech*)

This module iterates over the countries in the IRENA summary report, applies a mask of each country on a raster of potential of the technology *tech* that spans over the whole geographic scope, and calculates a probability distribution for that country which takes into account the potential but also a random factor. It selects the pixels with the highest probabilities, such that the number of pixels is equal to the number of units in that country and for that technology. After deriving the coordinates of those pixels, it saves them into a shapefile of points for each technology.

Parameters

- **`paths`** (*`dict`*) – Dictionary containing the paths for the potential maps of each technology. If a map is missing, the map of protected areas *PA* is used per default. It contains also the paths to the final shapefiles *locations_ren*.
- **`param`** (*`dict`*) – Dictionary containing information about the coordinates of the bounding box, the resolution, the georeferencing dictionary, geodataframes of land and sea areas, several parameters related to the distribution of renewable capacities.
- **`inst_cap`** (*`pandas dataframe`*) – Dataframe of the IRENA report after some processing in the module `distribute_renewable_capacities_IRENA`.
- **`tech`** (*`string`*) – Name of the renewable technology, as used in the dictionary *dist_ren* and in the dataframe *inst_cap*.

Returns The shapefile of points corresponding to the locations of power plants for the renewable energy technology *tech* is saved in the desired path, along with its corresponding metadata in a JSON file.

Return type None

`lib.spatial_functions.`**`define_spatial_scope`**(*scope_shp*)

This function reads the spatial scope shapefile and returns its bounding box.

Parameters **`scope_shp`** (*`Geopandas dataframe`*) – Spatial scope shapefile.

Return box List of the bounding box coordinates.

Return type list

lib.spatial_functions.**get_sites**(*points_shp*, *param*)
This function reads a shapefile of points, then performs a spatial join with a shapefile of regions to associate the names of the regions to the attributes of the points. It also removes duplicates and points that lie outside the subregions.

Parameters

- **points_shp** (*Geopandas dataframe*) – A shapefile of points (power plants, storage devices, etc.)
- **param** (*dict*) – Dictionary of user-defined parameters, including the shapefile *regions_sub*.

Return located The shapefile of points that are located within the subregions, with the name of the subregion as an attribute.

Return type Geopandas dataframe

lib.spatial_functions.**ind_global**(*Crd*, *res_desired*)
This function converts longitude and latitude coordinates into indices on a global data scope, where the origin is at (-90, -180).

Parameters

- **Crd** (*numpy array*) – Coordinates to be converted into indices.
- **res_desired** (*list*) – Desired resolution in the vertical and horizontal dimensions.

Return Ind Indices on a global data scope.

Return type numpy array

lib.spatial_functions.**ind_merra**(*Crd*, *Crd_all*, *res*)
This function converts longitude and latitude coordinates into indices within the spatial scope of MERRA-2 data.

Parameters

- **Crd** (*numpy array*) – Coordinates to be converted into indices.
- **Crd_all** (*numpy array*) – Coordinates of the bounding box of the spatial scope.
- **res** (*list*) – Resolution of the data, for which the indices are produced.

Return Ind Indices within the spatial scope of MERRA-2 data.

Return type numpy array

lib.spatial_functions.**intersection_subregions_countries**(*paths*, *param*)
This function reads two geodataframes, and creates a third one that is made of their intersection. The features are the different pieces that you would obtain by overlaying the features from the two input layers.

Parameters

- **paths** (*dict*) – Dictionary containing the path to the shapefile that is obtained.
- **param** (*dict*) – Dictionary containing the two input geodataframes (for countries, *regions_land*, and for subregions, *regions_sub*).

Return intersection Geodataframe with the features resulting from the intersection.

Return type geodataframe

lib.spatial_functions.**zonal_stats**(*regions_shp*, *raster_dict*, *param*)
This function calculates the zonal statistics for a given shapefile and a dictionary of rasters:

- Population: the sum is calculated.
- Landuse: the pixels for each land use type are counted.
- other_keys: the maximum is returned.

Parameters

- **`regions_shp`** (*`geodataframe`*) – Geodataframe containing the regions for which the statistics should be calculated.
- **`raster_dict`** (*`dict`*) – Dictionary with keys *Population*, *Landuse*, or any other string, and with paths of rasters as values.
- **`param`** (*`dict`*) – Dictionary containing *landuse_types*.

Return df Dataframe containing the zonal statistics, with the names of the regions as index and the keys of the statistics as columns.

Return type pandas dataframe

3.5 input_maps.py

`lib.input_maps.`**`generate_landsea`**(*paths*, *param*)

This function reads the shapefiles of the countries (land areas) and of the exclusive economic zones (sea areas) within the scope, and creates two rasters out of them.

Parameters

- **`paths`** (*`dict`*) – Dictionary including the paths *LAND* and *EEZ*.
- **`param`** (*`dict`*) – Dictionary including the geodataframes of the shapefiles, the number of features, the coordinates of the bounding box of the spatial scope, and the number of rows and columns.

Returns The tif files for *LAND* and *EEZ* are saved in their respective paths, along with their metadata in JSON files.

Return type None

`lib.input_maps.`**`generate_landuse`**(*paths*, *param*)

This function reads the global map of land use, and creates a raster out of it for the desired scope. There are 17 discrete possible values from 0 to 16, corresponding to different land use classes. See `config.py` for more information on the land use map.

Parameters

- **`paths`** (*`dict`*) – Dictionary including the paths to the global land use raster *LU_global* and to the output path *LU*.
- **`param`** (*`dict`*) – Dictionary including the desired resolution, the coordinates of the bounding box of the spatial scope, and the georeference dictionary.

Returns The tif file for *LU* is saved in its respective path, along with its metadata in a JSON file.

Return type None

`lib.input_maps.`**`generate_population`**(*paths*, *param*)

This function reads the global map of population density, resizes it, and creates a raster out of it for the desired scope. The values are in population per pixel.

Parameters

- **`paths`** (*`dict`*) – Dictionary including the paths to the global population raster *Pop_global* and to the output path *POP*.
- **`param`** (*`dict`*) – Dictionary including the desired resolution, the coordinates of the bounding box of the spatial scope, and the georeference dictionary.

Returns The tif file for *POP* is saved in its respective path, along with its metadata in a JSON file.

Return type None

`lib.input_maps.`**`generate_protected_areas`**(*paths*, *param*)

This function reads the shapefile of the globally protected areas, adds an attribute whose values are based on the dictionary of conversion (protected_areas) to identify the protection category, then converts the shapefile into a raster for the scope. The values are integers from 0 to 10.

Parameters

- **`paths`** (*`dict`*) – Dictionary including the paths to the shapefile of the globally protected areas, to the landuse raster of the scope, and to the output path PA.
- **`param`** (*`dict`*) – Dictionary including the dictionary of conversion of protection categories (protected_areas).

Returns The tif file for PA is saved in its respective path, along with its metadata in a JSON file.

Return type None

Utility functions as well as imported libraries are included in `util.py`.

3.6 util.py

`lib.util.`**`assign_values_based_on_series`**(*series*, *dict*)

This function fills a series based on the values of another series and a dictionary. The dictionary does not have to be sorted, it will be sorted before assigning the values. However, it must contain a key that is greater than any value in the series. It is equivalent to a function that maps ranges to discrete values.

Parameters

- **`series`** (*`pandas series`*) – Series with input values that will be mapped.
- **`dict`** (*`dictionary`*) – Dictionary defining the limits of the ranges that will be mapped.

Return result Series with the mapped discrete values.

Return type pandas series

`lib.util.`**`changem`**(*A*, *newval*, *oldval*)

This function replaces existing values *oldval* in a data array *A* by new values *newval*.

oldval and *newval* must have the same size.

Parameters

- **`A`** (*`numpy array`*) – Input matrix.
- **`newval`** (*`numpy array`*) – Vector of new values to be set.
- **`oldval`** (*`numpy array`*) – Vector of old values to be replaced.

Return Out The updated array.

Return type numpy array

`lib.util.`**`create_json`**(*filepath*, *param*, *param_keys*, *paths*, *paths_keys*)

This function creates a metadata JSON file containing information about the file in filepath by storing the relevant keys from both the param and path dictionaries.

Parameters

- **`filepath`** (*`string`*) – Path to the file for which the JSON file will be created.
- **`param`** (*`dict`*) – Dictionary of dictionaries containing the user input parameters and intermediate outputs.
- **`param_keys`** (*`list of strings`*) – Keys of the parameters to be extracted from the *param* dictionary and saved into the JSON file.

- **`paths`** (*`dict`*) – Dictionary of dictionaries containing the paths for all files.
- **`paths_keys`** (*`list of strings`*) – Keys of the paths to be extracted from the *paths* dictionary and saved into the JSON file.

Returns The JSON file will be saved in the desired path *filepath*.

Return type None

`lib.util.`**`display_progress`**(*message*, *progress_stat*)

This function displays a progress bar for long computations. To be used as part of a loop or with multiprocessing.

Parameters

- **`message`** (*`string`*) – Message to be displayed with the progress bar.
- **`progress_stat`** (*`tuple(int, int)`*) – Tuple containing the total length of the calculation and the current status or progress.

Returns The status bar is printed.

Return type None

`lib.util.`**`expand_dataframe`**(*df*, *column_names*)

This function reads a dataframe where columns with known *column_names* have multiple values separated by a semicolon in each entry. It expands the dataframe by creating a row for each value in each of these columns.

Parameters

- **`df`** (*`pandas dataframe`*) – The original dataframe, with multiple values in some entries.
- **`column_names`** (*`list`*) – Names of columns where multiple values have to be separated.

Return df_final The expanded dataframe, where each row contains only one value per column.

Return type pandas dataframe

`lib.util.`**`field_exists`**(*field_name*, *shp_path*)

This function returns whether the specified field exists or not in the shapefile linked by a path.

Parameters

- **`field_name`** (*`str`*) – Name of the field to be checked for.
- **`shp_path`** (*`str`*) – Path to the shapefile.

Returns `True` if it exists or `False` if it doesn't exist.

Return type bool

`lib.util.`**`get_sectoral_profiles`**(*paths*, *param*)

This function reads the raw standard load profiles, repeats them to obtain a full year, normalizes them so that the sum is equal to 1, and stores the obtained load profile for each sector in the dataframe *profiles*.

Parameters

- **`paths`** (*`dict`*) – Dictionary containing the paths to *dict_daytype*, *dict_season*, and to the raw standard load profiles.
- **`param`** (*`dict`*) – Dictionary containing the *year* and load-related assumptions.

Return profiles The normalized load profiles for the sectors.

Return type pandas dataframe

`lib.util.`**`resizem`**(*A_in*, *row_new*, *col_new*)

This function resizes regular data grid, by copying and pasting parts of the original array.

Parameters

- **`A_in`** (*`numpy array`*) – Input matrix.
- **`row_new`** (*`integer`*) – New number of rows.
- **`col_new`** (*`integer`*) – New number of columns.

Return A_out Resized matrix.

Return type numpy array

`lib.util.`**`reverse_lines`**(*df*)

This function reverses the line direction if the starting point is alphabetically after the end point.

Parameters **`df`** (*`pandas dataframe`*) – Dataframe with columns 'Region_start' and 'Region_end'.

Returns df_final The same dataframe after the line direction has been reversed.

Return type pandas dataframe

`lib.util.`**`timecheck`**(**args*)

This function prints information about the progress of the script by displaying the function currently running, and optionally an input message, with a corresponding timestamp. If more than one argument is passed to the function, it will raise an exception.

Parameters **`args`** (*`string`*) – Message to be displayed with the function name and the timestamp (optional).

Returns The time stamp is printed.

Return type None

Raise Too many arguments have been passed to the function, the maximum is only one string.

Finally, the module `generate_models.py` contains formating functions that create the input files for the urbs and evrys models.

3.7 generate_models module.py

`lib.generate_models.`**`generate_evrys_model`**(*paths*, *param*)

This function reads all the intermediate CSV files, adapts the formatting to the structure of the evrys Excel input file, and combines the datasets into one dataframe. It writes the dataframe into an evrys input Excel file. The function would still run even if some files have not been generated. They will simply be skipped.

Parameters

- **`paths`** (*`dict`*) – Dictionary including the paths to the intermediate files *sites_sub*, *commodities_regions*, *process_regions*, *grid_completed*, *storage_regions*, *load_regions*, *potential_ren*, and to the output *evrys_model*.
- **`param`** (*`dict`*) – Dictionary of user preferences, including *model_year* and *technology*.

Returns The XLSX model input file is saved directly in the desired path.

Return type None

`lib.generate_models.`**`generate_urbs_model`**(*paths*, *param*)

This function reads all the intermediate CSV files, adapts the formatting to the structure of the urbs Excel input file, and combines the datasets into one dataframe. It writes the dataframe into an urbs input Excel file. The function would still run even if some files have not been generated. They will simply be skipped.

Parameters

- **`paths`** (*`dict`*) – Dictionary including the paths to the intermediate files *sites_sub*, *commodities_regions*, *process_regions*, *assumptions_flows*, *grid_completed*, *storage_regions*, *load_regions*, *potential_ren*, and to the output *urbs_model*.
- **`param`** (*`dict`*) – Dictionary of user preferences, including *model_year* and *technology*.

Returns The XLSX model input file is saved directly in the desired path.

Return type None

Python Module Index

Bibliography

Bibliography for Chapter 1

M. Amado, F. Poggi, A. Ribeiro Amado, and S. Breu. A Cellular Approach to Net-Zero Energy Cities. *Energies*, 10(11):1826, 2017. doi: 10.3390/en10111826. 9

T. Anderski, Y. Surmann, S. Stemmer, N. Grisey, E. Momot, A.-C. Leger, B. Betraoui, and P. van Roy. e-HIGHWAY 2050: D 2.2 - European cluster model of the Pan-European transmission grid. Technical report, European Union, 2015. 9

N. Balta-Ozkan, T. Watson, and E. Mocca. Spatially uneven development and low carbon transitions: Insights from urban and regional planning. *Energy Policy*, 85:500–510, 2015. ISSN 03014215. doi: 10.1016/j.enpol.2015.05.013. 6

S. Batel. A critical discussion of research on the social acceptance of renewable energy generation and associated infrastructures and an agenda for the future. *Journal of Environmental Policy & Planning*, 20(3):356–369, 2018. ISSN 1523-908X. doi: 10.1080/1523908X.2017.1417120. 8

T. Benz, J. Dickert, M. Erbert, N. Erdmann, C. Johae, B. Katzenbach, W. Glaunsinger, H. Müller, P. Schegner, J. Schwarz, R. Speh, H. Stagge, and M. Zdrallek. Der zelluläre Ansatz: Grundlage einer erfolgreichen, regionenübergreifenden Energiewende [The cellular approach: Basis for a successful, cross-regional energy transition]. Technical report, ETG-VDE, 2015. URL `https://www.bund-naturschutz.de/fileadmin/Bilder_und_Dokumente/Presse_und_Aktuelles/Pressemitteilungen/2017/Energie_und_Klima/VDE_ST_ETG_ZellulareAnsatz_web.pdf`. Last viewed: May 31, 2020. 9

J. A. Berdegué, T. Hiller, J. M. Ramírez, S. Satizábal, I. Soloaga, J. Soto, M. Uribe, and O. Vargas. Delineating functional territories from outer space. *Latin American Economic Review*, 28(1): 534, 2019. ISSN 2198-3526. doi: 10.1186/s40503-019-0066-4. 2, 3

W. Biener and K. R. Garcia Rosas. Grid reduction for energy system analysis. *Electric Power Systems Research*, 185:106349, 2020. ISSN 03787796. doi: 10.1016/j.epsr.2020.106349. 6

K. Bódis, I. Kougias, A. Jäger-Waldau, N. Taylor, and S. Szabó. A high-resolution geospatial assessment of the rooftop solar photovoltaic potential in the European Union. *Renewable and Sustainable Energy Reviews*, 114:109309, 2019. ISSN 13640321. doi: 10.1016/j.rser.2019.109309. 8

J. Bosch, I. Staffell, and A. D. Hawkes. Temporally explicit and spatially resolved global offshore wind energy potentials. *Energy*, 163:766–781, 2018. ISSN 03605442. doi: 10.1016/j.energy.2018.08.153. 8

C. Candelise and P. Westacott. Can integration of PV within UK electricity network be improved? A GIS based assessment of storage. *Energy Policy*, 109:694–703, 2017. ISSN 03014215. doi: 10.1016/j.enpol.2017.07.054. 3

K.-K. Cao, K. von Krbek, M. Wetzel, F. Cebulla, and S. Schreck. Classification and Evaluation of Concepts for Improving the Performance of Applied Energy System Optimization Models. *Energies*, 12(24):4656, 2019. doi: 10.3390/en12244656. 6, 9

P. Díaz, O. van Vliet, and A. Patt. Do We Need Gas as a Bridging Fuel? A Case Study of the Electricity System of Switzerland. *Energies*, 10(7):861, 2017. doi: 10.3390/en10070861. 6

S. Dijks, M. Thylmann, and W. Peters. Regionale Auswirkungen des Windenergieausbaus auf die Vogelwelt [Regional effects of wind energy expansion on birds]. Technical report, Umweltstiftung WWF Deutschland, 2018. URL `https://mobil.wwf.de/fileadmin/fm-wwf/Publikationen-PDF/WWF_WEA_Vogelwelt.pdf`. Last viewed: May 31, 2020. 8

M. Drechsler, J. Egerer, M. Lange, F. Masurowski, J. Meyerhoff, and M. Oehlmann. Efficient and equitable spatial allocation of renewable power plants at the country scale. *Nature Energy*, 2 (9):17124, 2017. doi: 10.1038/nenergy.2017.124. 8

M. Düren. *Understanding the Bigger Energy Picture.* Springer International Publishing, Cham, 2017. ISBN 978-3-319-57965-8. doi: 10.1007/978-3-319-57966-5. 6

M. Eichhorn, F. Masurowski, R. Becker, and D. Thrän. Wind energy expansion scenarios – A spatial sustainability assessment. *Energy*, 180:367–375, 2019. ISSN 03605442. doi: 10.1016/j.energy.2019.05.054. 3

J. Fögele. Acquiring Powerful Thinking Through Geographical Key Concepts. In C. Brooks, G. Butt, and M. Fargher, editors, *The Power of Geographical Thinking*, volume 14 of *International Perspectives on Geographical Education*, pages 59–73. Springer International Publishing, Cham, 2017. ISBN 978-3-319-49985-7. doi: 10.1007/978-3-319-49986-4_5. 1, 2

F. Guo, J. Yang, and J. Lu. The battery charging station location problem: Impact of users' range anxiety and distance convenience. *Transportation Research Part E: Logistics and Transportation Review*, 114:1–18, 2018. ISSN 13665545. doi: 10.1016/j.tre.2018.03.014. 3

L. Hirth. Open data for electricity modeling: Legal aspects. *Energy Strategy Reviews*, 27:100433, 2020. ISSN 2211467X. doi: 10.1016/j.esr.2019.100433. 9

J. Hörsch and T. Brown. The role of spatial scale in joint optimisations of generation and transmission for European highly renewable scenarios. In *2017 14th International Conference on the European Energy Market (EEM)*, pages 1–7. IEEE, 2017. ISBN 978-1-5090-5499-2. doi: 10.1109/EEM.2017.7982024. 9

M. Howells, H. Rogner, N. Strachan, C. Heaps, H. Huntington, S. Kypreos, A. Hughes, S. Silveira, J. DeCarolis, M. Bazillian, and A. Roehrl. OSeMOSYS: The Open Source Energy Modeling System. *Energy Policy*, 39(10):5850–5870, 2011. ISSN 03014215. doi: 10.1016/j.enpol.2011.06.033. 2

International Renewable Energy Agency. Global Atlas, 2019. URL `http://www.irena.org/GlobalAtlas`. Last viewed: May 31, 2020. 8

K. Kanellopoulos, K. Kavvadias, F. Careri, M. De Felice, S. Busch, and I. Hidalgo Gonzalez. The operation of the European power system in 2016: A benchmarking study with METIS. *EU publications*, 2019. doi: 10.2760/434010. Reference number: KJ-NA-30008-EN-N. 2

S. Klein, S. W. Klein, T. Steinert, A. Fricke, and D. Peschel. Erneuerbare Gase - ein Systemupdate der Energiewende [Renewable gases - a system update of the energy transition]. Technical report, INES Initiative Erdgasspeicher e.V. / BWE Bundesverband Windenergie e.V., 2017. URL `https://www.wind-energie.de/fileadmin/redaktion/dokumente/publikationen-oeffentlich/themen/03-sektorenkopplung/20171212_studie_erneuerbare_gase.pdf`. Last viewed: May 31, 2020. 8

R. Knowles and J. Wareing. *Economic and Social Geography.* Elsevier, 1981. ISBN 9780750609227. doi: 10.1016/C2013-0-04558-6. 1, 2, 5, 6

V. Krishnan and W. Cole. Evaluating the value of high spatial resolution in national capacity expansion models using reeds. In *2016 IEEE Power and Energy Society General Meeting (PESGM)*, pages 1–5, July 2016. doi: 10.1109/PESGM.2016.7741996. 3

D. Majumdar and M. J. Pasqualetti. Dual use of agricultural land: Introducing 'agrivoltaics' in Phoenix Metropolitan Statistical Area, USA. *Landscape and Urban Planning*, 170:150–168, 2018. ISSN 01692046. doi: 10.1016/j.landurbplan.2017.10.011. 8

L. Mantzos, N. A. Matei, M. Rózsai, P. Russ, and A. S. Ramirez. POTEnCIA: A new EU-wide energy sector model. In *2017 14th International Conference on the European Energy Market (EEM)*, pages 1–5, June 2017. doi: 10.1109/EEM.2017.7982028. 2

F. Masurowski, M. Drechsler, and K. Frank. A spatially explicit assessment of the wind energy potential in response to an increased distance between wind turbines and settlements in Germany. *Energy Policy*, 97:343–350, 2016. ISSN 03014215. doi: 10.1016/j.enpol.2016.07.021. 8

P. Nijkamp. Energy problems and regional development. *Regional Science and Urban Economics*, 10(3):299–301, 1980. ISSN 01660462. doi: 10.1016/0166-0462(80)90034-4. 6

C. Perpiña Castillo, F. Batista e Silva, and C. Lavalle. An assessment of the regional potential for solar power generation in EU-28. *Energy Policy*, 88:86–99, 2016. ISSN 03014215. doi: 10.1016/j.enpol.2015.10.004. 3

S. Pfenninger and I. Staffell. Long-term patterns of European PV output using 30 years of validated hourly reanalysis and satellite data. *Energy*, 114:1251–1265, 2016. ISSN 03605442. doi: 10.1016/j.energy.2016.08.060. URL `https://www.renewables.ninja`. 8

S. Pfenninger, A. Hawkes, and J. Keirstead. Energy systems modeling for twenty-first century energy challenges. *Renewable and Sustainable Energy Reviews*, 33:74–86, 2014. ISSN 13640321. doi: 10.1016/j.rser.2014.02.003. 6

S. Rauner, M. Eichhorn, and D. Thrän. The spatial dimension of the power system: Investigating hot spots of Smart Renewable Power Provision. *Applied Energy*, 184:1038–1050, 2016. ISSN 03062619. doi: 10.1016/j.apenergy.2016.07.031. 3

L. Rey Los Santos, U. Temursho, M. Tamba, T. Vandyck, K. Wojtowicz, B. Saveyn, and M. Weitzel. Global macroeconomic balances for mid-century climate analyses. *EU publications*, 2018. doi: 10.2760/858469. Reference number: KJ-NA-29490-EN-N. 2

O. Ruhnau, S. Bannik, S. Otten, A. Praktiknjo, and M. Robinius. Direct or indirect electrification? A review of heat generation and road transport decarbonisation scenarios for Germany 2050. *Energy*, 166:989–999, 2019. ISSN 03605442. doi: 10.1016/j.energy.2018.10.114. 8

D. Ryberg, M. Robinius, and D. Stolten. Evaluating Land Eligibility Constraints of Renewable Energy Sources in Europe. *Energies*, 11(5):1246, 2018. doi: 10.3390/en11051246. 8

K. Siala and H. Houmy. tum-ens/pyPRIMA: python PReprocessing of Inputs for Model frAmeworks. doi: 10.5281/zenodo.3872023, June 2020. Version v1.0.0. 7

K. Siala, H. Houmy, W. S. Khan, and M. Y. Mahfouz. tum-ens/pyCLARA: python Clustering of Lines And RAsters. doi: 10.5281/zenodo.3872274, June 2020. Version v1.0.0. 7

S. Simoes, W. Nijs, P. Ruiz, A. Sgobbi, and C. Thiel. Decarbonised pathways for a low carbon eu28 power sector until 2050. In *11th International Conference on the European Energy Market (EEM14)*, pages 1–5, May 2014. doi: 10.1109/EEM.2014.6861232. 2

H.-W. Sinn. Buffering volatility: A study on the limits of Germany's energy revolution. *European Economic Review*, 99:130–150, 2017. ISSN 00142921. doi: 10.1016/j.euroecorev.2017.05.007. 8

Solargis. Global Solar Atlas, 2019. URL `https://globalsolaratlas.info`. Last viewed: May 31, 2020. 8

N. Strachan, B. Fais, and H. Daly. Reinventing the energy modelling–policy interface. *Nature Energy*, 1(3):16012, 2016. doi: 10.1038/nenergy.2016.12. 1

Technical University of Denmark. Global Wind Atlas, 2019. URL `https://globalwindatlas.info`. Last viewed: May 31, 2020. 8

T. Tröndle, S. Pfenninger, and J. Lilliestam. Home-made or imported: On the possibility for renewable electricity autarky on all scales in Europe. *Energy Strategy Reviews*, 26:100388, 2019. ISSN 2211467X. doi: 10.1016/j.esr.2019.100388. 8

A. Uslu, C. Coppens, H. Gordijn, M. Piek, P. Ruyssenaars, J.-j. Schrander, P. de Smet, R. Swart, M. Hoogwijk, M. Papalexandrou, E. de Visser, J. Horalek, P. Kurfürst, F. P. Jensen, B. S. Petersen, M. Harfoot, R. Milego, N.-E. Clausen, G. Giebel, and P. Kurfust. Europe's onshore and offshore wind energy potential. Technical report, European Environmental Agency, 2009. URL `https://www.energy.eu/publications/a07.pdf`. Last viewed: May 31, 2020. 8

U. Wardenga. Räume der geographie – zu raumbegriffen im geographieunterricht [spaces of geography - on spatial terms in geography lessons]. *Wissenschaftliche Nachrichten*, 120: 47–52, 2002. URL `https://homepage.univie.ac.at/Christian.Sitte/FD/artikel/ute_wardenga_raeume.htm`. Last viewed: May 31, 2020. 1, 2

J. J. Watson and M. D. Hudson. Regional Scale wind farm and solar farm suitability assessment using GIS-assisted multi-criteria evaluation. *Landscape and Urban Planning*, 138:20–31, 2015. ISSN 01692046. doi: 10.1016/j.landurbplan.2015.02.001. 8

B. Wiegmans. GridKit extract of ENTSO-E interactive map. doi: 10.5281/zenodo.55853, June 2016. 7

Bibliography for Chapter 2

P. Böhme. *Wärmeverbrauchsanalyse auf Basis einer raumbezogenen Zusammenführung von Gebäudedaten [Heat consumption analysis based on a spatially referenced conflation of building data]*. Dissertation, Technische Universität München, Munich, Germany, 2013. URL `http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20131219-1162472-0-3`. Last viewed: May 31, 2020. 18

K.-K. Cao, K. von Krbek, M. Wetzel, F. Cebulla, and S. Schreck. Classification and Evaluation of Concepts for Improving the Performance of Applied Energy System Optimization Models. *Energies*, 12(24):4656, 2019. doi: 10.3390/en12244656. 20, 21

S. M. Manson. Simplifying complexity: a review of complexity theory. *Geoforum*, 32(3):405–414, 2001. ISSN 00167185. doi: 10.1016/S0016-7185(00)00035-X. 15

J. Priesmann, L. Nolting, and A. Praktiknjo. Are complex energy system models more accurate? An intra-model comparison of power system optimization models. *Applied Energy*, 255:113783, 2019. ISSN 03062619. doi: 10.1016/j.apenergy.2019.113783. 15, 21

Bibliography for Chapter 3

P. D. Broxton, X. Zeng, D. Sulla-Menashe, and P. A. Troch. A global land cover climatology using modis data. *Journal of Applied Meteorology and Climatology*, 53(6):1593–1605, 2014. doi: 10.1175/JAMC-D-13-0270.1. 26

J. de Ferranti and C. Hormann. Viewfinder Panorama. `http://viewfinderpanoramas.org`, 2015. Last viewed: May 31, 2020. 26

R. Gelaro, W. McCarty, M. J. Suárez, R. Todling, A. Molod, L. Takacs, C. A. Randles, A. Darmenov, M. G. Bosilovich, R. Reichle, K. Wargan, L. Coy, R. Cullather, C. Draper, S. Akella, V. Buchard, A. Conaty, A. M. da Silva, W. Gu, G.-K. Kim, R. Koster, R. Lucchesi, D. Merkova, J. E. Nielsen, G. Partyka, S. Pawson, W. Putman, M. Rienecker, S. D. Schubert, M. Sienkiewicz, and B. Zhao. The Modern-Era Retrospective Analysis for Research and Applications, Version 2 (MERRA-2). *Journal of Climate*, 30(14):5419–5454, 2017. doi: 10.1175/JCLI-D-16-0758.1. 26

Technical University of Denmark. Global Wind Atlas, 2019. URL `https://globalwindatlas.info`. Last viewed: May 31, 2020. 26

W. R. Tobler. A Computer Movie Simulating Urban Growth in the Detroit Region. *Economic Geography*, 46:234–240, 1970. ISSN 00130095, 19448287. doi: 10.2307/143141. 26

Bibliography for Chapter 4

A. Ångström. Solar and terrestrial radiation. *Quarterly Journal of the Royal Meteorological Society*, 50:121–126, 1924. 32

K. Bakirci. Models of solar radiation with hours of bright sunshine: A review. *Renewable and Sustainable Energy Reviews*, 13(9):2580 – 2588, 2009. ISSN 1364-0321. doi: 10.1016/j.rser.2009.07.011. 32

A. Boilley and L. Wald. Comparison between meteorological re-analyses from ERA-Interim and MERRA and measurements of daily solar irradiation at surface. *Renewable Energy*, 75:135–143, 2015. doi: 10.1016/j.renene.2014.09.042. 32

P. D. Broxton, X. Zeng, D. Sulla-Menashe, and P. A. Troch. A global land cover climatology using modis data. *Journal of Applied Meteorology and Climatology*, 53(6):1593–1605, 2014. doi: 10.1175/JAMC-D-13-0270.1. 33

Institute for Climate and Atmospheric Sciences. Global Energy Balance Archive. Online database, 2015. URL `http://www.geba.ethz.ch/`. Last viewed: May 31, 2020. 33

IUCN and UNEP. The World Database on Protected Areas (WDPA). Online database, 2014. URL `www.protectedplanet.net`. Last viewed: May 31, 2020. 33

A. Jarvis, E. Guevara, H. Reuter, and A. Nelson. Hole-filled srtm for the globe : version 4 : data grid, 2008. URL `http://srtm.csi.cgiar.org`. Published by CGIAR-CSI on 19 August 2008. Last viewed: May 31, 2020. 33

A. D. Kennedy, X. Dong, B. Xi, S. Xie, Y. Zhang, and J. Chen. A Comparison of MERRA and NARR Reanalyses with the DOE ARM SGP Data. *Journal of Climate*, 24(17):4541–4557, 2011. doi: 10.1175/2011JCLI3978.1. 32

A. Lopez, B. Roberts, D. Heimiller, N. Blair, and G. Porro. U.S. Renewable Energy Technical Potentials: A GIS-Based Analysis. Technical report, NREL, July 2012. URL `http://www.nrel.gov/docs/fy12osti/51946.pdf`. Reference number: NREL/TP-6A20-51946. Prepared under Task Nos. SA10.1012 and SA10.20A4. Last viewed: May 31, 2020. 38

A. Olseth and A. Skartveit. Solar irradiance, sunshine duration and daylight illuminance derived from METEOSAT data for some European sites. *Theoretical and Applied Climatology*, 69: 239–252, 2001. doi: 10.1007/s007040170029. 32

J. Page. The Role of Solar Radiation Climatology in the Design of Photovoltaic Systems . In T. Markvart and L. Castañer, editors, *Practical Handbook of Photovoltaics*, chapter 1, pages 5 – 66. Elsevier Science, Amsterdam, 2003. ISBN 978-1-85617-390-2. doi: 10.1016/B978-185617390-2/50004-0. 33, 35, 36, 37, 38

M. C. Peel, B. L. Finlayson, and T. A. McMahon. Updated world map of the Köppen-Geiger climate classification. *Hydrology and Earth System Sciences*, 11:1633–1644, October 2007. doi: 10.5194/hess-11-1633-2007. 33, 34

R. Perez, P. Ineichen, K. Moore, C. Kmiecik, M. amd Chain, R. George, and F. Vignola. A new operational model for satellite-derived irradiances: description and validation. *Solar Energy*, 73 (5):307–317, 2002. doi: 10.1016/S0038-092X(02)00122-6. 32

J. Polo, M. Gastø'n, J. Vindel, and I. Pagola. Spatial variability and clustering of global solar irradiation in Vietnam from sunshine duration measurements. *Renewable and Sustainable Energy Reviews*, 42(0):1326 – 1334, 2015. ISSN 1364-0321. doi: 10.1016/j.rser.2014.11.014. 32

J. A. Prescott. Evaporation from a water surface in relation to solar radiation. *Transactions of The Royal Society of South Australia*, 64:114–118, 1940. 32

D. T. Reindl, W. A. Beckman, and J. A. Duffie. Diffuse fraction correlations. *Solar*, 45(1):1 – 7, 1990. doi: 10.1016/0038-092X(90)90060-P. 37

D. B. Richardson and R. W. Andrews. Validation of the MERRA dataset for solar PV applications. In *2014 IEEE 40th Photovoltaic Specialists Conference (PVSC)*, pages 0809–0814. IEEE, June 2014. doi: 10.1109/PVSC.2014.6925039. 32

M. M. Rienecker, M. J. Suarez, R. Gelaro, R. Todling, J. Bacmeister, E. Liu, M. G. Bosilovich, S. D. Schubert, L. Takacs, G.-K. Kim, S. Bloom, J. Chen, D. Collins, A. Conaty, A. da Silva, W. Gu, J. Joiner, R. D. Koster, R. Lucchesi, A. Molod, T. Owens, S. Pawson, P. Pegion, C. R. Redder, R. Reichle, F. R. Robertson, A. G. Ruddick, M. Sienkiewicz, and J. Woollen. MERRA: NASA's Modern-Era Retrospective Analysis for Research and Applications. *Journal of Climate*, 24(14):3624–3648, July 2011. doi: 10.1175/JCLI-D-11-00015.1. URL `http://disc.sci.gsfc.nasa.gov/daac-bin/FTPSubset.pl`. Last viewed: May 31, 2020. 32, 33

J. A. Ruiz-Arias, T. Cebecauer, J. Tovar-Pescador, and M. Šúri. Spatial disaggregation of satellite-derived irradiance using a high-resolution digital elevation model. *Solar Energy*, 84(9):1644 – 1657, 2010. ISSN 0038-092X. doi: 10.1016/j.solener.2010.06.002. 33

Z. Sen. Fuzzy algorithm for estimation of solar irradiation from sunshine duration. *Solar Energy*, 63(1):39–49, 1998. doi: 10.1016/S0038-092X(98)00043-7. 32

E. Skoplaki and J. Palyvos. Operating temperature of photovoltaic modules: A survey of pertinent correlations. *Renewable Energy*, 34(1):23 – 29, 2009. ISSN 0960-1481. doi: 10.1016/j.renene.2008.04.009. 33, 38

J. Stein, R. Perez, and A. Parkins. Validation of PV Performance Models Using Satellite-based Irradiance Measurements: a Case Study. In R. Campbell-Howe, editor, *39th ASES National Solar Conference 2010 (SOLAR 2010)*, pages 265–290. American Solar Energy Society (ASES), Curran Associates, Inc., 2010. URL `http://www.asrc.cestm.albany.edu/perez/2010/spp.pdf`. Last viewed: May 31, 2020. 32

Water Resources, Development and Management Service. CLIMWAT, A Climatic Database. Online database, 2015. URL `http://www.fao.org/land-water/databases-and-software/climwat-for-cropwat/en/`. Last viewed: May 31, 2020. 33, 35

L. T. Wong and W. K. Chow. Solar radiation model. *Applied Energy*, 69(3):191–224, 2001. doi: 10.1016/S0306-2619(01)00012-5. 33

World Radiation Data Centre. WRDC archive data. Online database, 2015. URL `http://wrdc.mgo.rssi.ru/wrdc_en_new.htm`. Last viewed: May 31, 2020. 33, 36

Yingli Solar. PANDA 60 CELL SERIES 2. Online product sheet, September 2014. URL `https://pdf.archiexpo.com/pdf/yingli-green-energy-europe-gmbh/panda-60-cell-series-2/84386-252133.html`. Last viewed: May 31, 2020. 33

Bibliography for Chapter 5

K. Adam, M. Müller-Mienack, M. Paun, G. Sanchis, and K. Strunz. e-HIGHWAY 2050 — The ENTSO-E facilitated study programme towards a Modular Development Plan on pan-European Electricity Highways System 2050. In *2012 IEEE Power and Energy Society General Meeting*, pages 1–6, July 2012. doi: 10.1109/PESGM.2012.6344807. 52

L. Anselin, I. Syabri, and Y. Kho. GeoDa : An Introduction to Spatial Data Analysis. *Geographical Analysis*, 38(1):5–22, 2006. doi: 10.1111/j.0016-7363.2005.00671.x. 54

D. Arthur and S. Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, page 1027–1035, USA, 2007. Society for Industrial and Applied Mathematics. ISBN 9780898716245. URL `https://dl.acm.org/doi/10.5555/1283383.1283494`. Last viewed: May 31, 2020. 53, 67

BDEW. Standardlastprofile Strom [Standard load profiles Eletricity], Jan. 2017. URL `https://www.bdew.de/energie/standardlastprofile-strom/`. Last viewed: May 31, 2020. 61

P. D. Broxton, X. Zeng, D. Sulla-Menashe, and P. A. Troch. A global land cover climatology using modis data. *Journal of Applied Meteorology and Climatology*, 53(6):1593–1605, 2014. doi: 10.1175/JAMC-D-13-0270.1. 61

J. Dorfner, M. Dorfner, K. Schönleber, S. Candas, smuellr, dogauzrek, wyaudi, yunusozsahin, adeeljsid, T. Zipperle, S. Herzog, L. Odersky, K. Siala, and O. Akca. *urbs: v0.7.3: A linear optimisation model for distributed energy systems.* Chair of Renewable and Sustainable Energy Systems, Technical University of Munich, 2018. doi: 10.5281/zenodo.1228851. 60

J. C. Duque, L. Anselin, and S. J. Rey. The max-p-regions problem. *Journal of Regional Science*, 52(3):397–419, 2012. ISSN 00224146. doi: 10.1111/j.1467-9787.2011.00743.x. 52, 54, 68

J. C. Duque, M. C. Vélez-Gallego, and L. C. Echeverri. On the performance of the subtour elimination constraints approach for the p -regions problem: A computational study. *Geographical Analysis*, 50(1):32–52, 2018. ISSN 00167363. doi: 10.1111/gean.12132. 53

ENTSO-E. Monthly Hourly Load Values, 2015. URL `https://www.entsoe.eu/data/power-stats/`. Last viewed: May 31, 2020. 61

M. M. Fischer. Regional taxonomy. *Regional Science and Urban Economics*, 10(4):503–537, 1980. ISSN 01660462. doi: 10.1016/0166-0462(80)90015-0. 53

B. A. Frew and M. Z. Jacobson. Temporal and spatial tradeoffs in power system modeling with assumptions about storage: An application of the power model. *Energy*, 117:198 – 213, 2016. ISSN 0360-5442. doi: 10.1016/j.energy.2016.10.074. 52

S. Gago Da Camara Simoes, W. Nus, P. Ruiz Castello, A. Sgobbi, D. Radu, P. Bolat, C. Thiel, and E. Peteves. The JRC-EU-TIMES model - Assessing the long-term role of the SET Plan Energy technologies. EUR - Scientific and Technical Research Reports, European Commission Joint Research Centre, Institute for Energy and Transport, 2013. 52

R. Gelaro, W. McCarty, M. J. Suárez, R. Todling, A. Molod, L. Takacs, C. A. Randles, A. Darmenov, M. G. Bosilovich, R. Reichle, K. Wargan, L. Coy, R. Cullather, C. Draper, S. Akella, V. Buchard, A. Conaty, A. M. da Silva, W. Gu, G.-K. Kim, R. Koster, R. Lucchesi, D. Merkova, J. E. Nielsen, G. Partyka, S. Pawson, W. Putman, M. Rienecker, S. D. Schubert, M. Sienkiewicz, and B. Zhao. The Modern-Era Retrospective Analysis for Research and Applications, Version 2 (MERRA-2). *Journal of Climate*, 30(14):5419–5454, 2017. doi: 10.1175/JCLI-D-16-0758.1. 61

F. Hofmann, J. Hörsch, and F. Gotzens. FRESNA/powerplantmatching: v0.2 (Version v0.2). Zenodo. doi: 10.5281/zenodo.1405595, Aug. 2018. 61

A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988. ISBN 0-13-022278-X. 52

R. Lacal Arantegui, A. Jaeger-Waldau, M. Vellei, B. Sigfusson, D. Magagna, M. Jakubcionis, M. d. M. Perez Fortes, S. Lazarou, J. Giuntoli, E. Weidner Ronnefeld, G. De Marco, A. Spisto, and C. Gutierrez Moles. ETRI 2014 - Energy Technology Reference Indicator projections for 2010-2050. EUR - Scientific and Technical Research Reports, Joint Research Center of the European Union, 2014. doi: 10.2790/057687. 61

O. Lavagne d'Ortigue, A. Whiteman, and S. Elsayed. Renewable Energy Capacity Statistics 2015. Technical report, IRENA, 2016. URL `http://www.irena.org/-/media/Files/IRENA/Agency/Publication/2015/IRENA_RE_Capacity_Statistics_2015.pdf`. Last viewed: May 31, 2020. 61

S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2): 129–137, 1982. ISSN 0018-9448. doi: 10.1109/TIT.1982.1056489. 52, 53

M. Y. Mahfouz, W. S. Khan, and K. Siala. geoclustering: v0.1 (beta), 2019. URL `https://github.com/tum-ens/geoclustering`. Last viewed: May 31, 2020. 53

S. Pfenninger, A. Hawkes, and J. Keirstead. Energy systems modeling for twenty-first century energy challenges. *Renewable and Sustainable Energy Reviews*, 33:74 – 86, 2014. ISSN 1364-0321. doi: 10.1016/j.rser.2014.02.003. 52

S. Rey, L. Anselin, P. Amaral, D. Arribas-Bel, M. Wigginton Conway, C. Famer, D. C. Folch, J. Gaboardi, M. Hwang, W. Kang, E. Knaap, M. Kolak, J. Laura, Z. Li, S. Lumnitz, T. Oshan, R. Pahle, C. Schmidt, H. Shao, P. Stephens, S. Wang, and L. J. Wolf. pySAL Documentation - Release 1.14.4, 2018. URL `http://pysal.org/`. Last viewed: May 31, 2020. 54

K. Siala and M. Y. Mahfouz. Supplementary materials for the spatial clustering of Europe, Sept. 2018. doi: 10.5281/zenodo.1439342. 55, 61

K. Siala, H. Houmy, and S. A. Huezo Rodriguez. renewable-timeseries (beta), 2019. URL `https://github.com/tum-ens/renewable-timeseries`. Last viewed: May 31, 2020. 61

R. L. Thorndike. Who belongs in the family? *Psychometrika*, 18(4):267–276, 1953. ISSN 0033-3123. doi: 10.1007/BF02289263. 54, 67

B. Wiegmans. GridKit extract of ENTSO-E interactive map. doi: 10.5281/zenodo.55853, June 2016. 61

Bibliography for Chapter 6

Z. M. Chen, G. Q. Chen, and B. Chen. Embodied Carbon Dioxide Emissions of the World Economy: A Systems Input-Output Simulation for 2004. *Procedia Environmental Sciences*, 2: 1827 – 1840, 2010. ISSN 1878-0296. doi: 10.1016/j.proenv.2010.10.194. International Conference on Ecological Informatics and Ecosystem Conservation (ISEIS 2010). 75

S. J. Davis and K. Caldeira. Consumption-based accounting of CO2 emissions. *Proceedings of the National Academy of Sciences*, 107(12):5687–5692, 2010. ISSN 0027-8424. doi: 10.1073/pnas.0906974107. 75

J. Dorfner, M. Dorfner, K. Schönleber, S. Candas, smuellr, dogauzrek, wyaudi, yunusozsahin, adeeljsid, T. Zipperle, S. Herzog, L. Odersky, K. Siala, and O. Akca. *urbs: v0.7.3: A linear optimisation model for distributed energy systems.* Chair of Renewable and Sustainable Energy Systems, Technical University of Munich, 2018. doi: 10.5281/zenodo.1228851. 74

ENTSO-E. Monthly Hourly Load Values, 2015. URL `https://www.entsoe.eu/data/power-stats/`. Last viewed: May 31, 2020. 73, 76

European Commission. Energy roadmap 2050. Technical report, European Union, 2012. doi: 10.2833/10759. 72, 74

R. Gelaro, W. McCarty, M. J. Suárez, R. Todling, A. Molod, L. Takacs, C. A. Randles, A. Darmenov, M. G. Bosilovich, R. Reichle, K. Wargan, L. Coy, R. Cullather, C. Draper, S. Akella, V. Buchard, A. Conaty, A. M. da Silva, W. Gu, G.-K. Kim, R. Koster, R. Lucchesi, D. Merkova, J. E. Nielsen, G. Partyka, S. Pawson, W. Putman, M. Rienecker, S. D. Schubert, M. Sienkiewicz, and B. Zhao. The Modern-Era Retrospective Analysis for Research and Applications, Version 2 (MERRA-2). *Journal of Climate*, 30(14):5419–5454, 2017. doi: 10.1175/JCLI-D-16-0758.1. 73

F. Hofmann, J. Hörsch, and F. Gotzens. FRESNA/powerplantmatching: v0.2 (Version v0.2). Zenodo. doi: 10.5281/zenodo.1405595, Aug. 2018. 73

J. Kitzes. An introduction to environmentally-extended input-output analysis. *Resources*, 2(4): 489–503, 2013. ISSN 2079-9276. doi: 10.3390/resources2040489. 75

R. Lacal Arantegui, A. Jaeger-Waldau, M. Vellei, B. Sigfusson, D. Magagna, M. Jakubcionis, M. d. M. Perez Fortes, S. Lazarou, J. Giuntoli, E. Weidner Ronnefeld, G. De Marco, A. Spisto, and C. Gutierrez Moles. ETRI 2014 - Energy Technology Reference Indicator projections for 2010-2050. EUR - Scientific and Technical Research Reports, Joint Research Center of the European Union, 2014. doi: 10.2790/057687. 73

O. Lavagne d'Ortigue, A. Whiteman, and S. Elsayed. Renewable Energy Capacity Statistics 2015. Technical report, IRENA, 2016. URL `http://www.irena.org/-/media/Files/IRENA/Agency/Publication/2015/IRENA_RE_Capacity_Statistics_2015.pdf`. Last viewed: May 31, 2020. 73

W. Leontief. The Structure of American Economy: 1919-1939. *New York, Oxford University Press*, 1951. 74

R. E. Miller and P. D. Blair. *Input-Output Analysis*. Cambridge University Press, 2009. ISBN 9780521739023. URL `https://EconPapers.repec.org/RePEc:cup:cbooks:9780521739023`. Last viewed: May 31, 2020. 74

B. Su, H. Huang, B. Ang, and P. Zhou. Input–output analysis of CO2 emissions embodied in trade: The effects of sector aggregation. *Energy Economics*, 32(1):166 – 175, 2010. ISSN 0140-9883. doi: 10.1016/j.eneco.2009.07.010. 75

M. P. Timmer, E. Dietzenbacher, B. Los, R. Stehrer, and G. J. Vries. An Illustrated User Guide to the World Input–Output Database: the Case of Global Automotive Production. *Review of International Economics*, 23(3):575–605, 2015. doi: 10.1111/roie.12178. 75

A. Tukker, E. Poliakov, R. Heijungs, T. Hawkins, F. Neuwahl, J. M. Rueda-Cantuche, S. Giljum, S. Moll, J. Oosterhaven, and M. Bouwmeester. Towards a global multi-regional environmentally extended input–output database. *Ecological Economics*, 68(7):1928 – 1937, 2009. ISSN 0921-8009. doi: 10.1016/j.ecolecon.2008.11.010. Methodological Advancements in the Footprint Analysis. 75

T. Wiedmann, H. C. Wilting, M. Lenzen, S. Lutter, and V. Palm. Quo Vadis MRIO? Methodological, data and institutional requirements for multi-region input–output analysis. *Ecological Economics*, 70(11):1937 – 1945, 2011. ISSN 0921-8009. doi: 10.1016/j.ecolecon.2011.06.014. Special Section - Earth System Governance: Accountability and Legitimacy. 75

B. Wiegmans. GridKit extract of ENTSO-E interactive map. doi: 10.5281/zenodo.55853, June 2016. 73

J. E. Zafrilla, M.-Á. Cadarso, F. Monsalve, and C. de la Rúa. How Carbon-Friendly Is Nuclear Energy? A Hybrid MRIO-LCA Model of a Spanish Facility. *Environmental Science & Technology*, 48(24):14103–14111, 2014. doi: 10.1021/es503352s. PMID: 25386802. 75

Bibliography for Chapter 7

I. Boiarchuk, K. Siala, D. Hewes, R. Witzmann, and T. Hamacher. Investigation of the Performance of Nodal and Zonal ENTSO-E Transmission System Models in the Context of Large-Scale Integration of Renewables. In *Conference on Sustainable Energy Supply and Energy Storage Systems*, pages 289–295, Hamburg, 2018. VDE Verlag. ISBN 978-3-8007-4445-9. 86

Bibliography for Appendix A

K. Siala and H. Houmy. tum-ens/pyGRETA: python Generator of REnewable Time series and mAps. doi: 10.5281/zenodo.3872068, June 2020. Version v1.0.1. 93

Bibliography for Appendix B

K. Siala, H. Houmy, W. S. Khan, and M. Y. Mahfouz. tum-ens/pyCLARA: python Clustering of Lines And RAsters. doi: 10.5281/zenodo.3872274, June 2020. Version v1.0.0. 147

Bibliography for Appendix C

K. Siala and H. Houmy. tum-ens/pyPRIMA: python PReprocessing of Inputs for Model frAmeworks. doi: 10.5281/zenodo.3872023, June 2020. Version v1.0.0. 171