# Richtlinie zum „Software-Sprint“

# EnergyModels

## *Schlussbericht*

Zuwendungsempfänger:

Jonas Hörsch

Das diesem Bericht zugrundeliegende Vorhaben wurde mit Mitteln des Bundesministeriums für Bildung und Forschung unter dem Förderkennzeichen 01IS18S53 gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt beim Autor.

## Kurze Darstellung der Aufgabenstellung und Motivation

Die Energiewende in Europa und auf der Welt ist in mancher Hinsicht noch ein technisches, aber zu großen Teilen inzwischen mehr ein gesellschaftliches Problem. Durch das komplexe Zusammenspiel zwischen variierender, erneuerbarer Erzeugung, Speichern und Übertragungsleitungen sind die vorgegebenen Referenzszenarien der EU und der Netzbetreiber weder von Bürgerinitiativen, noch von Umweltorganisationen prüfbar und dadurch auch nur schwer diskutierbar. Der undurchsichtige Prozess begünstigt Fehlinformationen und drückt die soziale Akzeptanz. Im Vorhaben EnergyModels soll ein modulares Framework geschaffen werden mit dem sowohl Investitionen als auch der Betrieb von Energiesystemen auf dem Weg zur Dekarbonisierung modelliert und analysiert werden können. Dabei sind Einschränkungen wie globale und sektorale Emissionsziele genauso wie gesellschaftlich gewünschte Ausbaubeschränkungen (bspw. der Übertragungsleitungen) flexibel wählbar.

EnergyModels ist ein Julia Paket, das den Datenfluß und die Struktur eines zu einem Graphen verbundenen Energiesystems mit variabler Zeitauflösung vorgibt. Es stellt die physikalischen Gleichungen für einen Satz von Standardkomponenten (Generatoren, Speichern, Übertragungsleitungen) bereit, erlaubt es aber neue Komponenten hinzuzufügen oder bereits bestehende zu ersetzen. Die Optimierungsvariablen und Zwangsbedingungen werden mit JuMP an quell-offene oder kommerzielle Optimierungssoftware kommuniziert und gelöst.

1. Entwurf und Dokumentation des zugrundeliegenden Komponentenmodells auf neuer Projektwebseite und Einrichtung und Verteilung der zugehörigen Mailingliste

2. Entwicklung der Kernfunktionalität des Datenmanagement und der Kommunikation mit der Optimierungssoftware

3. Erstellung eines Satzes nachvollziehbarer Tutorials in Form von Jupyter Notebooks zur Lösung einfacher Inselsysteme mit EnergyModels
4. Ausrichtung eines Do-a-thons auf dem Workshop der Open Energy Modelling Initiative
5. Integration der standardmäßigen Plotfunktionalität: Karten, Standardbalken und -kurvenplots
6. Vorbereitung und Lösung eines europäischen und eines deutschen Datensatzes auf Grundlage des PyPSA-Eur Modells
7. Entwicklung der Exportfunktionalität zu einer interaktiven, kartenbasierten Webseite
8. Ausrichtung von zwei Webinaren für zivilgesellschaftliche Organisationen (1) und für erfahrene Energiesystemmodellierer (2)

## Beitrag des Projektes zu den Zielen der Förderinitative „Software-Sprint“

Energiesystemmodellierer*innen in der Wissenschaft und in zivilgesellschaftlichen Organisationen erhalten eine flexibel erweiterbare Arbeitsgrundlage um Dekarbonisierungsszenarien zu entwerfen und Referenzszenarien auf Plausibilität zu testen. Das Projekt wurde mit den Entwicklern der derzeitigen offenen Frameworks zur linearen Energiesystemoptimierung, PyPSA, Calliope, urbs und oemof konzipiert.

## Ausführliche Darstellung der Ergebnisse

EnergyModels.jl wurde als Optimierungsframework auf Github unter einer offenen Lizenz (MIT) veröffentlicht. Die Bibliothek enthält Modelle für die gängigsten Technologien (Generatoren, Speichertechnologien, HVAC- und HVDC-Leitungen) um Elektrizitätsnetzwerke zu optimieren. Neue Technologien können vom Benutzer mit wenigen Funktionsdefinitionen ergänzt werden. Daten werden aus einer einzigen speziell angeordneten NetCDF Dateien on-demand geladen, um den Arbeitsspeicher-Verbrauch gering zu halten. Der Übergang von Julia 0.6 zu 1.0 und der damit verbundenen Umstellung des Optimierungsframeworks JuMP von 0.18 zu 0.19 führte zu erheblichem Programmiermehraufwand für die Fertigstellung des Komponentenmodells und machte es unter anderem notwendig Anpassungen in fremden Bibliotheken (der Datenbibliothek AxisArrays#160 und vor allem des Optimierungsframeworks JuMP#2003) vorzunehmen und einzupflegen. (Meilensteine 1 + 2).

Auf dem Workshop der Open Energy Modelling Initiative in Aarhus wurde ein Do-a-thon angeboten, um EnergyModels.jl in die anderen Optimierungsmodelle und -bibliotheken in der Programmiersprache Julia einzuordnen (Meilenstein 4).

Basierend auf dem Workshop-Feedback, dass vorhandene Frameworks für linearisierte Energiesystemoptimierungen generell ausreichen, konkrete Modelle inklusive Daten allerdings schwierig zu finden und überprüfen wären, wurden einige Zielanpassungen vorgenommen:

1. Der Modellgenerator PyPSA-Eur wurde überarbeitet, um beliebige Länder innerhalb Europas auswählen zu können (z.B. Modell für DE oder BE-NL-FR-DE-AT-CH) und eine ausführliche Dokumentation wurde erstellt (Meilenstein 6)

2. Die Datenkompatibilität für EnergyModels.jl wurde erweitert um direkt mit PyPSA, wie auch PowerSystems/PowerSimulations (und damit auch MATPOWER, PSS/E) Datensätzen arbeiten zu können.

3. Da Energiesystemoptimierungen mit nicht-linearen Flussmodellen von den verglichenen Frameworks nur unzureichend unterstützt werden, wurde das Komponentenmodell um reaktive Leistungseinspeisungen und eine feingliedrigere Modellwahl erweitert und eine Anbindung an die Flussmodellbibliothek PowerModels.jl begonnen.

Die feingliedrige Modellwahl bedeutet, dass die möglichen Einsatzplanungsmodelle von den Flussmodellen und Kapazitätserweiterungsmodellen getrennt wurden, so dass eine nicht lineare oder auch block-/leitungsscharfe Kapazitätserweiterung mit linearem und blockscharfem Einsatz mit einem linearen Flussmodell kombiniert werden kann (und in naher Zukunft voraussichtlich auch einem nicht-lineare second-order cone basierten Flussmodell).

Die Umkonzeptionierung wurde gut begleitet durch das Betreuungsgespräch mit dem PTF Team und Projektmanagement Coaching durch Simon Höher.

Die Neuausrichtung zu mehr Interframeworkkompatibilität und dem Fokus auf spezialisierten Fähigkeiten, wie der Kombination von PowerModels Flussmodellen mit den Komponenten in EnergyModels, ersetzt die Plotfunktionalität und den Kartenexport die der Benutzer mit anderen Bibliotheken (PowerSimulations.jl, PyPSA oder einfach StatsPlots) realisieren sollte (Meilensteine 5+7).

Am 17.06. fand das OpenMod/NGO bridge meeting in Berlin mit Vertretern von 5 verschiedenen NGOs statt (mehr waren eingeladen) mit dem Ziel eine kollaborative Modellierung des Energiesystems anzustoßen. NGO-Vertreter vertraten klar die Position, dass Sie keine Tools benötigen, die Sie eigener Modellierung unterstützen würden, sondern vor allem wissenschaftliche Ansprechpartner, die zu konkreten Fragestellungen Stellung beziehen und Analysen erstellen würden (statt Webinar 1 in Meilenstein 8).

## Zielgruppe, Nutzen und mögliche Weiterentwicklungen

Energiesystemmodellierer können nun auf ein klar strukturiertes Komponentenmodell und eine Verknüpfung mehrerer Optimierungsbibliotheken zu einem kohärenten Framework zurückgreifen, die im Programmiercode sehr übersichtlichen Einblick in die verwendeten Gleichungen gibt und auch ohne Veränderung des Hauptprogrammcodes von Benutzer um neue Komponenten erweitert werden kann, die ein oder mehrere Flussmodelle unterstützen.

Die Veröffentlichung des Codes unter der freien MIT-Lizenz ermöglicht die unbedenkliche Nutzung und Weiterentwicklung der Projektergebnisse oder auch Teile der Projektergebnisse in sowohl akademischen als auch kommerziellen Projekten.

Mögliche Weiterentwicklungen sind:

1. Die Fertigstellung der PowerModels-Anbindung (EnergyModels.jl#21) zur Verwendung nicht-linearer Flussmodelle (wahrscheinlich)
2. Die Bereitstellung eines tabellarischen Interfaces, um übersichtlich Zugriff auf statische und dynamische Systemparameter und die Lösungargumente zu bekommen (basierend auf Tables.jl)
3. Verbesserung der Plot- und Exportfunktionen
4. Erweiterung der eingebauten Komponenten und verfügbare Standardbibliothek an Parametern

## Kurze Darstellung der Arbeiten, die zu keiner Lösung geführt haben

Die Anbindung an PowerModels.jl um nicht-lineare Flussmodelle zu verwenden, wurde konzipiert, aber noch nicht ausreichend getestet, um in eine neue Version des Optimierungsframeworks übernommen zu werden. Der jetzige Stand ist als sogenannter Pull Request (EnergyModels.jl#21) diskutierbar veröffentlicht. Fabian Neumann, Doktorand der Energiesystemmodellierungsgruppe am Institute for Automation and Applied Informatics des Karlsruhe Institut für Technologie möchte diese Arbeit weiterführen.

## Kurze Angabe von Präsentationsmöglichkeiten für mögliche Nutzer

Die Github-Seite von EnergyModels.jl, https://github.com/PyPSA/EnergyModels.jl, gibt direkt eine kurze Einführung in das Projekt und Framework. Dort sind auch die bereits gehaltenen Präsentationen inklusive der enthaltenen Tutorials aufgeführt.

## Kurze Erläuterung zur Einhaltung der Arbeits- und Kostenplanung

Das Projekt schloss die in der „Ausführlichen Beschreibung der Ergebnisse“ dokumentierten Meilensteine in abgerechneten 535 Stunden von den beantragten 540 Stunden ab.

## Kurze Darstellung von etwaigen Ergebnissen bei anderen Stellen

Die Bibliothek PowerSimulations.jl von NREL, Berkeley, entwickelt und gepflegt, stellt eine sich schnell weiterentwickelnde Alternative zu EnergyModels.jl dar, die bereits nicht-lineare Flussmodelle inkorporiert und eine bessere Anbindung an gängige Datenformate wie PTI (PSS/E) und M (*MATPOWER case)* bietet. Zusätzlich wurden einige Plot Rezepte in der Bibliothek verankert.

EnergyModels.jl kann daher nun Optimierungen auf Basis des PowerSimulations.jl Datentyp ausführen.

Für die Python-basierte Bibliothek PyPSA wurde eine minimale Bibliothek mit dem Namen nomopyomo geschrieben, mit der das Kapazitätsexpansions- und Einsatzplanungsproblem mit minimal-möglichem Speicheraufwand ausgeführt werden kann. Eine Erweiterung auf nicht-lineare Flussmodelle ist allerdings aufgrund des gewählten Ansatzes nicht möglich.