# Introducing nemo

## High performance, open-source energy system optimization

Jason Veysey and Eric Kemp-Benedict (presenter)

ETSAP Conference 2019, Paris

SEI Stockholm Environment Institute

6-7 June 2019

# |nemo: the Next Energy Modeling system for Optimization

**Implemented**

- Least-cost optimization of energy supply and demand
- Flexible specification of technologies, fuels, time periods, and constraints
- Modeling of
    - Energy storage
    - Emissions and emission constraints
    - Renewable energy targets
- Support for multiple regions and regional trade
- Straightforward integration of user-defined constraints and (via Julia JuMP) other modeling tools

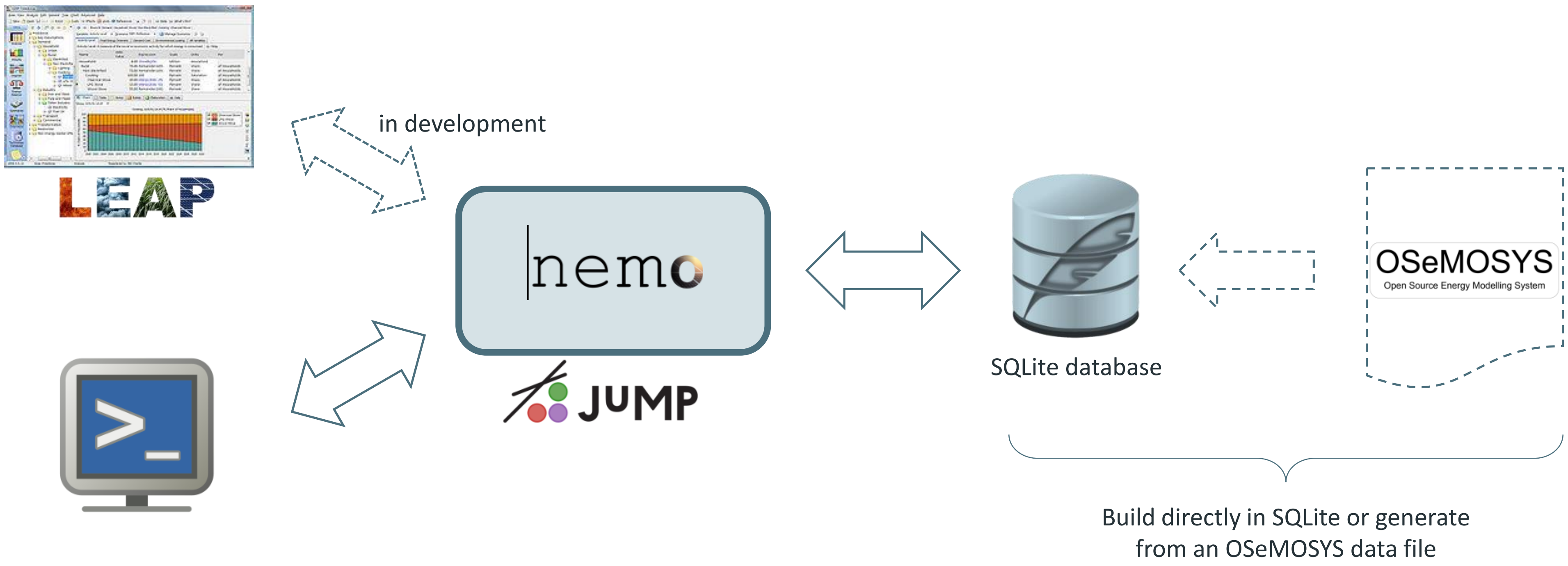
**In development**

- DC optimized power flow simulation
- Support for geospatially explicit energy demand and supply analyses

# Why |nemo?

- Evolving needs of energy planners in developing countries (the core audience for the LEAP energy planning software)
  - Accelerating technological change
  - Growing economies and population
  - Rapidly changing agendas on climate and sustainability
- Existing tools present challenges: cost, learning curve, or performance
- |nemo will be open-source, but is SEI led, so we can rapidly address partners' needs

# Architecture



# Development

- Initially built to replicate OsEMOSYS functionality
- Cut runtime of large OSeMOSYS files by over 90% (Julia/JuMP + SQLite + refactoring)
- Confirmed that it replicated OSeMOSYS results
- No longer compatible: |nemo now has a different storage algorithm than OSeMOSYS

The system is under active development: Not yet open source

# Distribution

Delivered as a Julia package named “NemoMod” (currently in a private GitHub repository)

An all-in-one Windows installer is included:

- Installs Julia if needed
- Installs the NemoMod package
- Optionally installs a custom version of the Julia system image that contains a compiled copy of |nemo: greatly improves performance

Julia dependencies are set up automatically when NemoMod is installed

# Dimensions

| Abbreviation | Dimension |
|---|---|
| **e** | emission |
| **f** | fuel |
| **l** | time slice |
| **m** | mode of operation |
| **r** or **rr** | region |
| **s** | storage |
| **t** | technology |
| **tg1** | time slice group 1 |
| **tg2** | time slice group 2 |
| **y** | year |

# Time slicing

Years are subdivided using three sets and two parameters:

## Sets

- **TIMESLICE**: Slices that divide the hours of the year (each hour must belong to exactly one time slice)
- **TSGROUP1** (time slice group 1): Higher-level groups of time slices that divide the year
- **TSGROUP2** (time slice group 2): Lower-level groups of time slices that divide a time slice group 1

## Parameters

**LTsGroup**: A parameter mapping time slices to time slice groups 1 and 2.

**YearSplit**: A parameter that specifies fractions of the year for each time slice.

# Storage sets

**MODE_OF_OPERATION**: e.g., “generate” and “store”

**STORAGE** – One element for each storage unit (e.g., “lithium-ion batteries”, “pumped hydro”, “Northfield Mountain pumped hydro”, etc.)

**TECHNOLOGY** – At least one charging/discharging technology for each storage unit

# Storage parameters

**CapitalCostStorage** (r, s, y) – The cost of constructing one unit of a STORAGE

**InputActivityRatio** (r, t, f, m, y) – Input fuel consumption when a technology operates in a particular mode

**MinStorageCharge** (r, s, y) – Minimum allowable energy in a STORAGE. A percent between 0 and 1

**OperationalLifeStorage** (r, s) – Lifetime of a STORAGE in years

**ResidualStorageCapacity** (r, s, y) – Exogenously specified STORAGE capacity in the scenario's energy unit (e.g., PJ)

**OutputActivityRatio** (r, t, f, m, y) – Output fuel production when a technology operates in a particular mode

**StorageLevelStart** (r, s) – Energy in a STORAGE at the start of the modeling period (again, in the scenario's energy unit).

**StorageMaxChargeRate** (r, s) – Maximum charging rate for a STORAGE in the scenario's energy unit/year (e.g., PJ/year)

**StorageMaxDischargeRate** (r, s) – Maximum discharging rate for a STORAGE in the scenario's energy unit/year (e.g., PJ/year)

**TechnologyFromStorage** (r, t, s, m) – Links a TECHNOLOGY to a STORAGE for discharging (= 1 if linked)

**TechnologyToStorage** (r, t, s, m) – Links a TECHNOLOGY to a STORAGE for charging (= 1 if linked)

**TotalAnnualMaxCapacityInvestmentStorage** (r, s, y) – Maximum endogenous STORAGE capacity that can be added in the given year

**TotalAnnualMinCapacityInvestmentStorage** (r, s, y) – Minimum endogenous STORAGE capacity that must be added in the given year

**TotalAnnualMaxCapacityStorage** (r, s, y) – Maximum total STORAGE capacity (endogenous + exogenous) allowed in the given year

**TotalAnnualMinCapacityStorage** (r, s, y) – Minimum total STORAGE capacity (endogenous + exogenous) required in the given year

# Storage outputs

**vstorageleveltsgroup1start** (r, s, tg1, y) – Energy in the STORAGE at the beginning of the TSGROUP1 in the given year

**vstorageleveltsgroup1end** (r, s, tg1, y) – Energy in the STORAGE at the end of the TSGROUP1 in the given year

**vstorageleveltsgroup2start** (r, s, tg1, tg2, y) – Energy in the STORAGE at the beginning of the TSGROUP2 in the given year and TSGROUP1

**vstorageleveltsgroup2end** (r, s, tg1, tg2, y) – Energy in the STORAGE at the end of the TSGROUP2 in the given year and TSGROUP1

**vstorageleveltsend** (r, s, l, y) – Energy in the STORAGE at the end of the first hour in the given year and time slice

**vrateofstoragecharge** (r, s, l, y) – Rate of charging of the STORAGE in the given year and time slice

**vrateofstoragedischarge** (r, s, l, y) – Rate of discharging of the STORAGE in the given year and time slice

**vstoragelowerlimit** (r, s, y) – Minimum energy that must be in the STORAGE throughout the given year

**vstorageupperlimit** (r, s, y) – Maximum energy that can be in the STORAGE throughout the given year

**vaccumulatednewstoragecapacity** (r, s, y) – Total endogenous capacity for the STORAGE existing in the given year

**vnewstoragecapacity** (r, s, y) – Endogenous capacity for the STORAGE added in the given year

**vcapitalinvestmentstorage** (r, s, y) – Undiscounted capital costs for the STORAGE incurred in the given year

**vdiscountedcapitalinvestmentstorage** (r, s, y) – Discounted capital costs for the STORAGE incurred in the given year

**vsalvagevaluestorage** (r, s, y) – Undiscounted salvage value for STORAGE added in the given year

**vdiscountedsalvagevaluestorage** (r, s, y) – Discounted salvage value for STORAGE added in the given year

**vtotaldiscountedstoragecost** (r, s, y) – vdiscountedcapitalinvestmentstorage minus vdiscountedsalvagevaluestorage

# Final remarks

- |nemo replicates and extends OSeMOSYS functionality
- Using Julia JuMP and SQLite (and refactoring) gives a substantial performance boost
- Currently developing a LEAP link for an existing developing country application
- Under active development
  - Documentation
  - DC optimized power flow simulation
  - Support for geospatially explicit energy demand and supply analyses
- Will be released open source via GitHub