CyBORG: Employing the Origen API in a Fuel Cycle Simulator

Steven E. Skutnik
University of Tennessee-Knoxville

2017 SCALE Users' Group Meeting
Oak Ridge National Laboratory

THE UNIVERSITY OF TENNESSEE KNOXVILLE

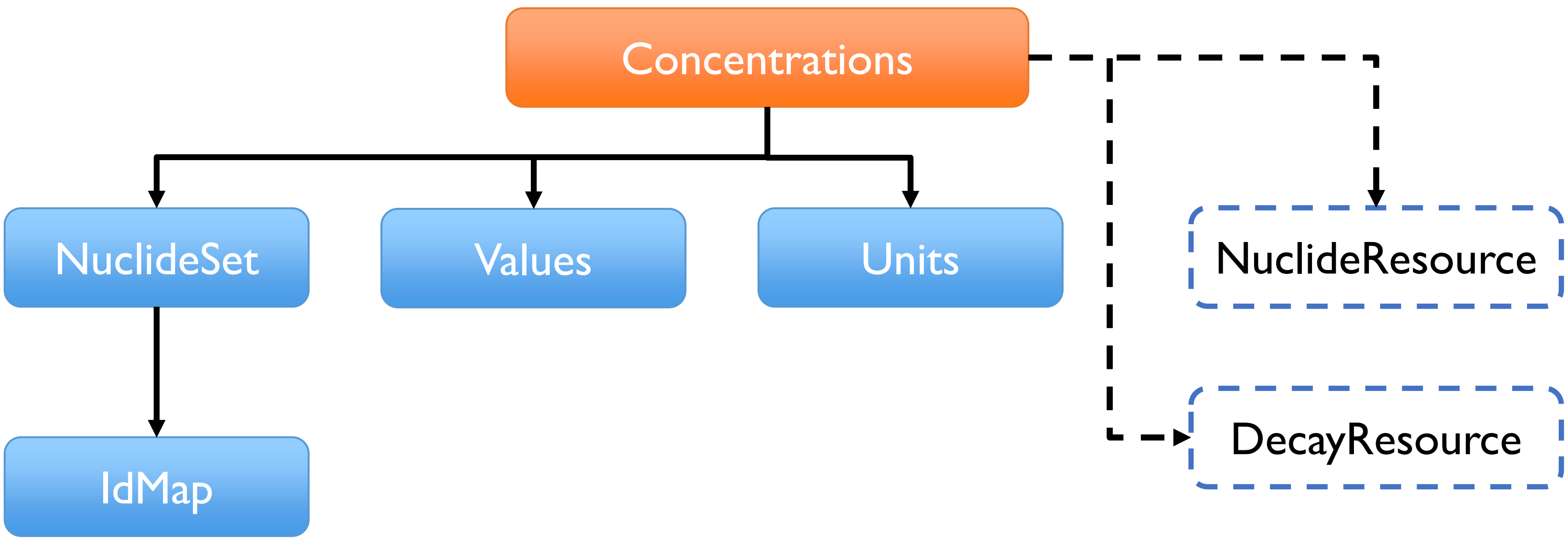
Agenda

- Introduction to the modernized Origen API in SCALE 6.2
- The need for physics-based depletion in fuel cycle assessment
- **CyBORG**: Integrating Origen into the Cyclus nuclear fuel cycle simulator
- Handling generalized cross-section interpolation

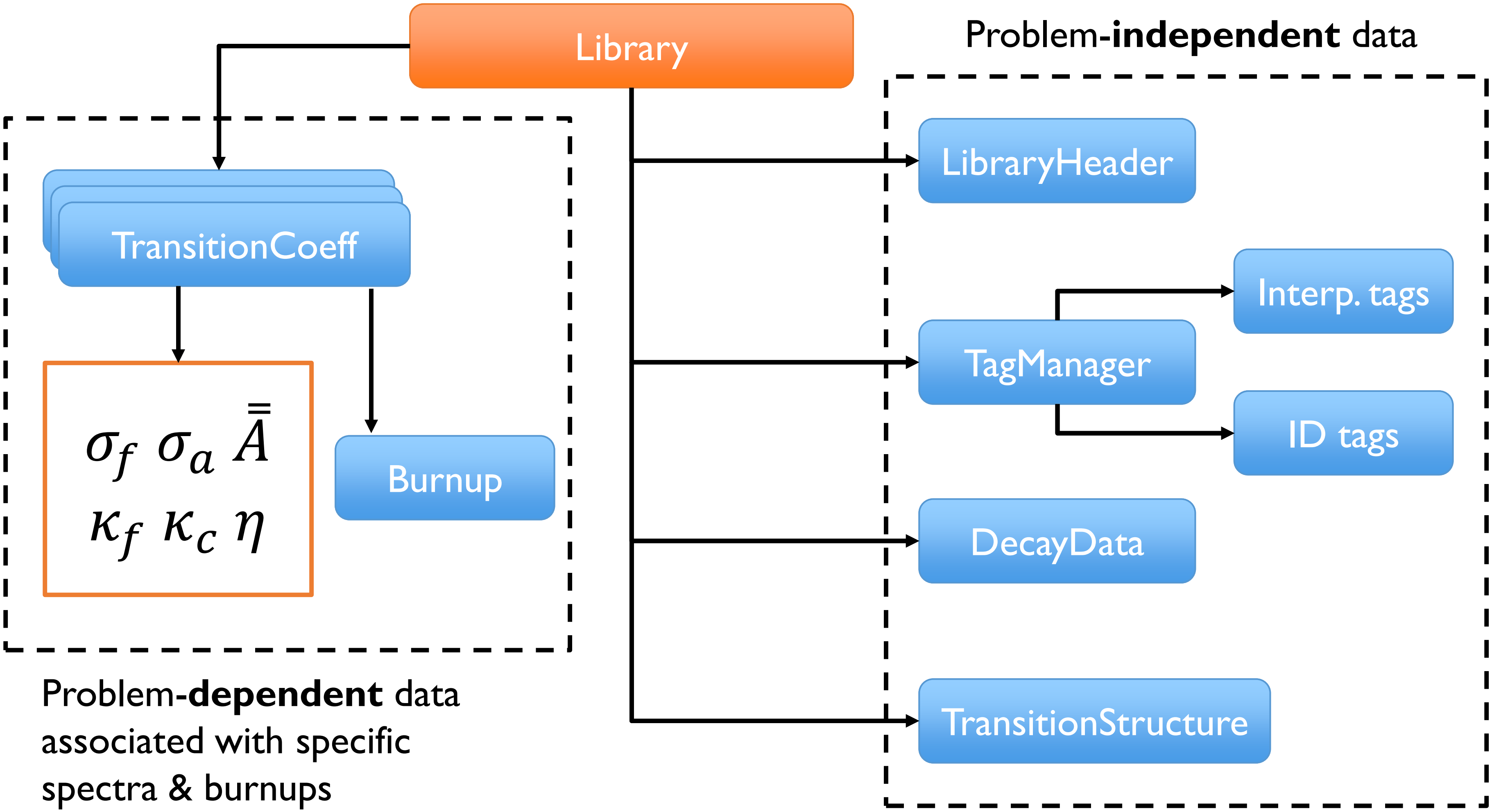
Introduction to the modernized Origen API in SCALE

Directly accessing Origen depletion methods & embedding depletion capabilities into other code frameworks

Basic topology of the Origen API: Concentrations

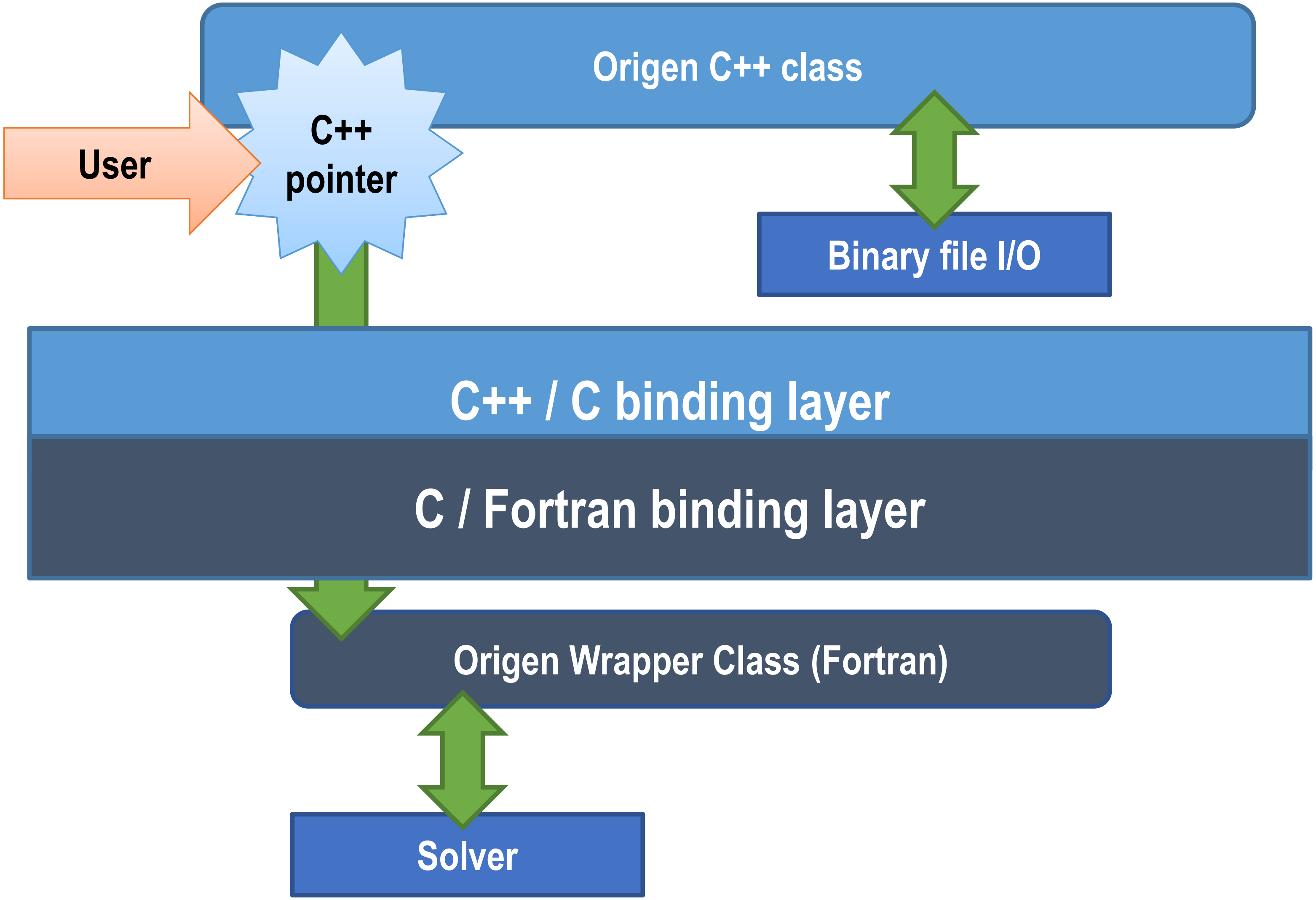


Basic topology of the Origen API: Library

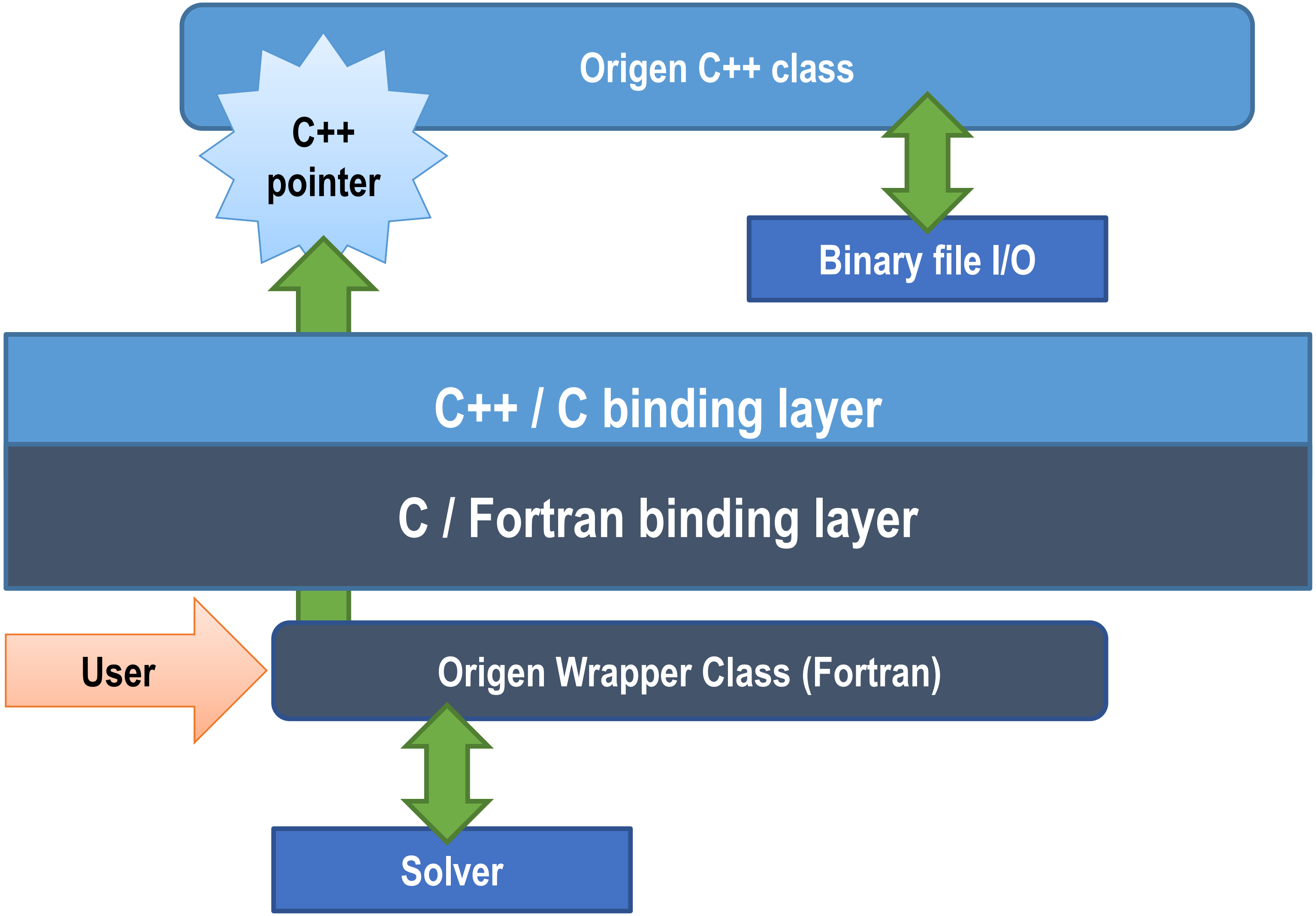


Problem-**dependent** data associated with specific spectra & burnups

Interfacing with the Origen solver (C++)



Fortran wrapper interface

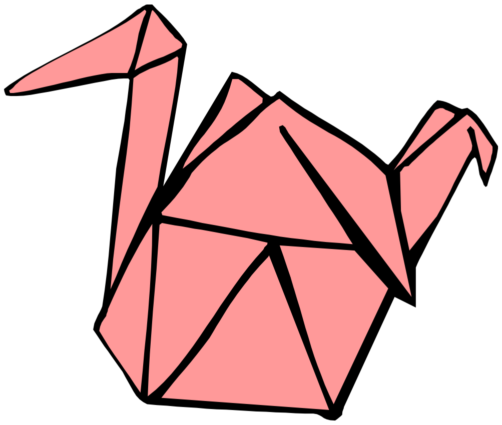


Building upon the Origen API as a foundation

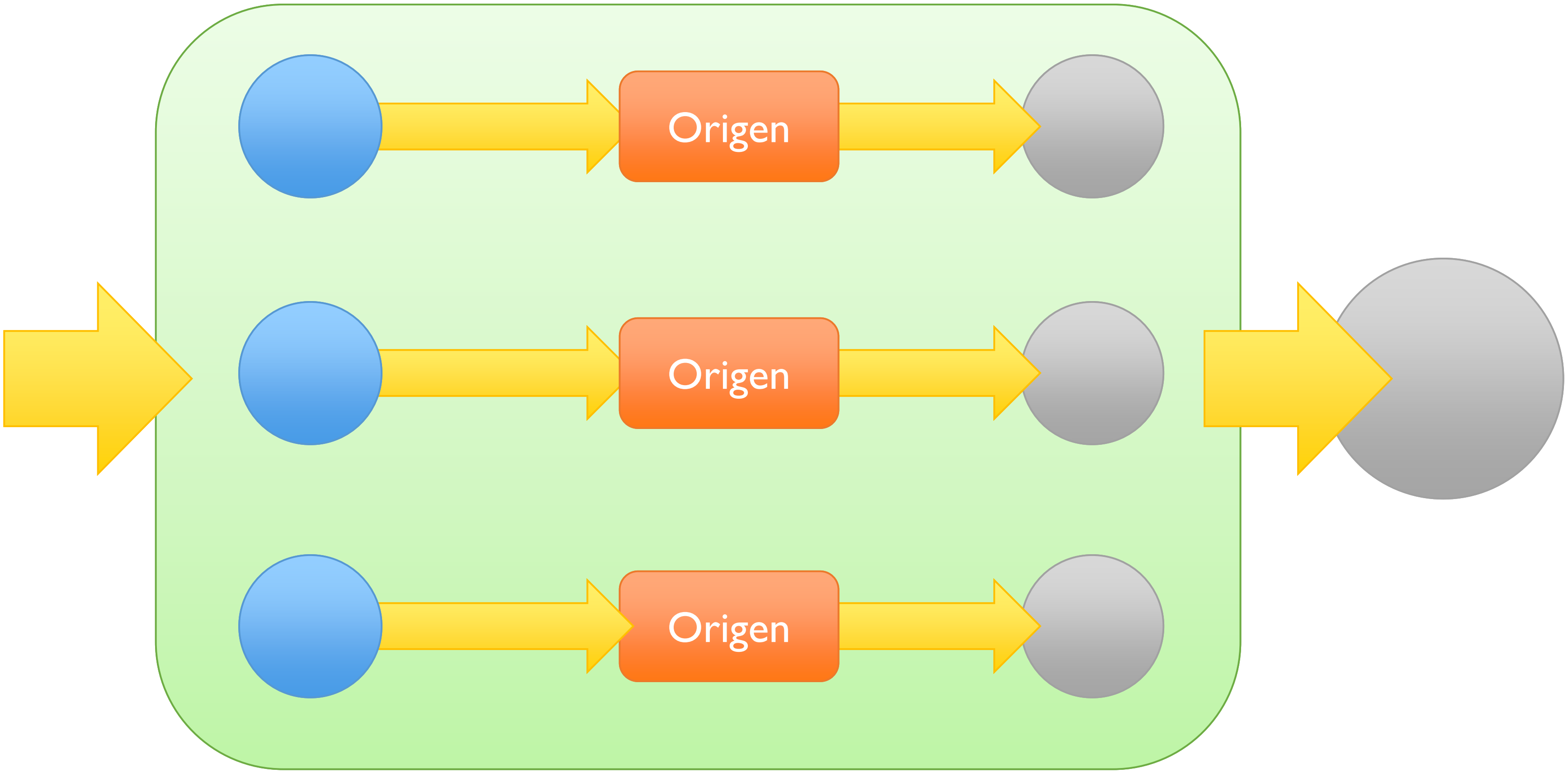
- With the ability to call Origen from a variety of in-memory interfaces, it becomes possible to develop new applications on top of the Origen solver interface
- **Examples of this include:**
  - The ORIGAMI interface for 3-D point depletion of fuel assemblies & source term generation
  - The CyBORG physics-based depletion reactor for the Cyclus fuel cycle simulator

ORIGAMI is an example of an Origen interface built upon the new Origen API

- The **ORIGAMI** interface for Origen treats the assembly as multiple axial and radial depletion "nodes"
- Each "node" has a power "shaping" factor to account for axial and radial differences in the burnup distribution
  - Radial nodes can also have different libraries – accounting for local differences in neutron spectrum



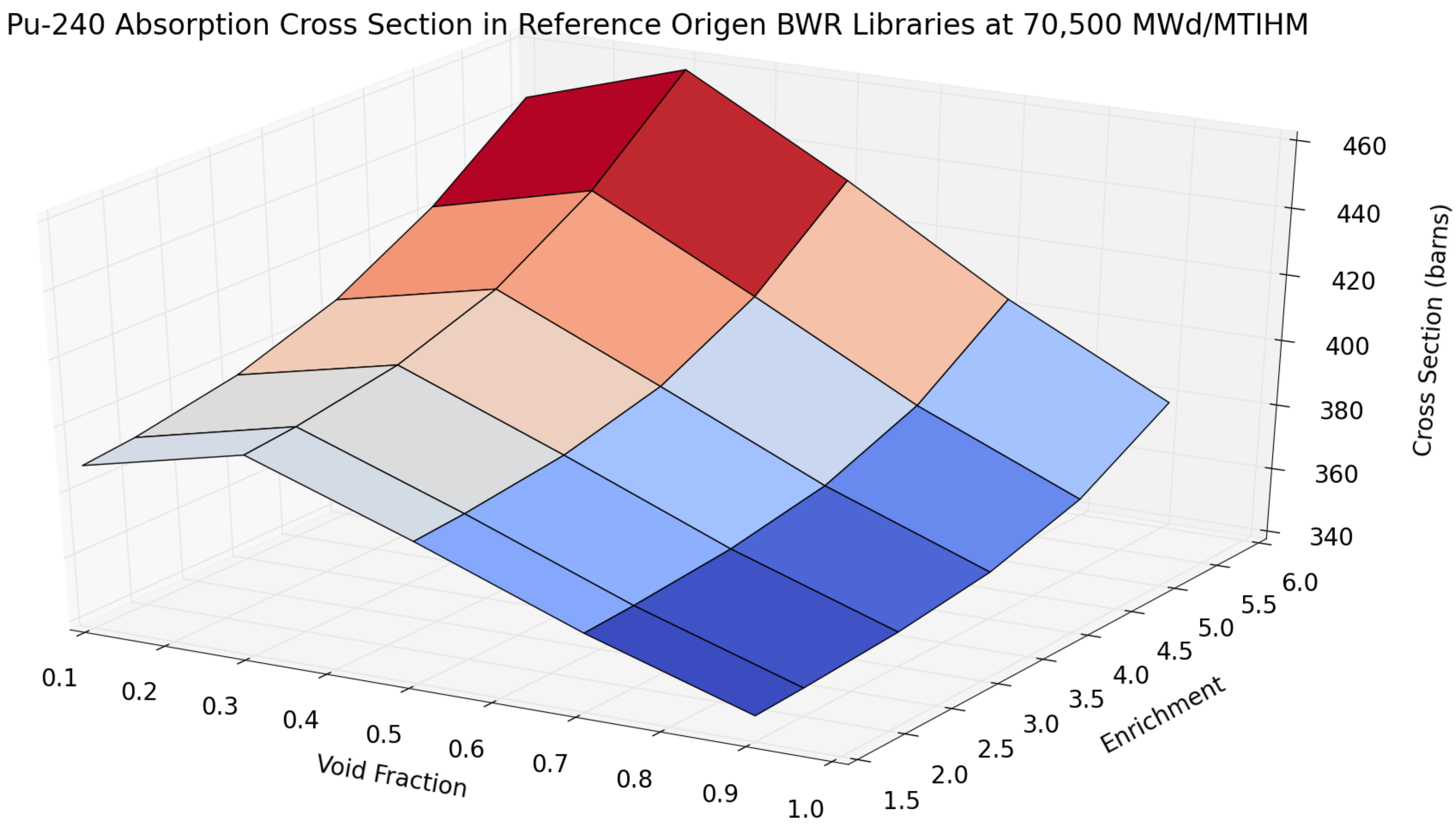
ORIGAMI: A new way to use Origen



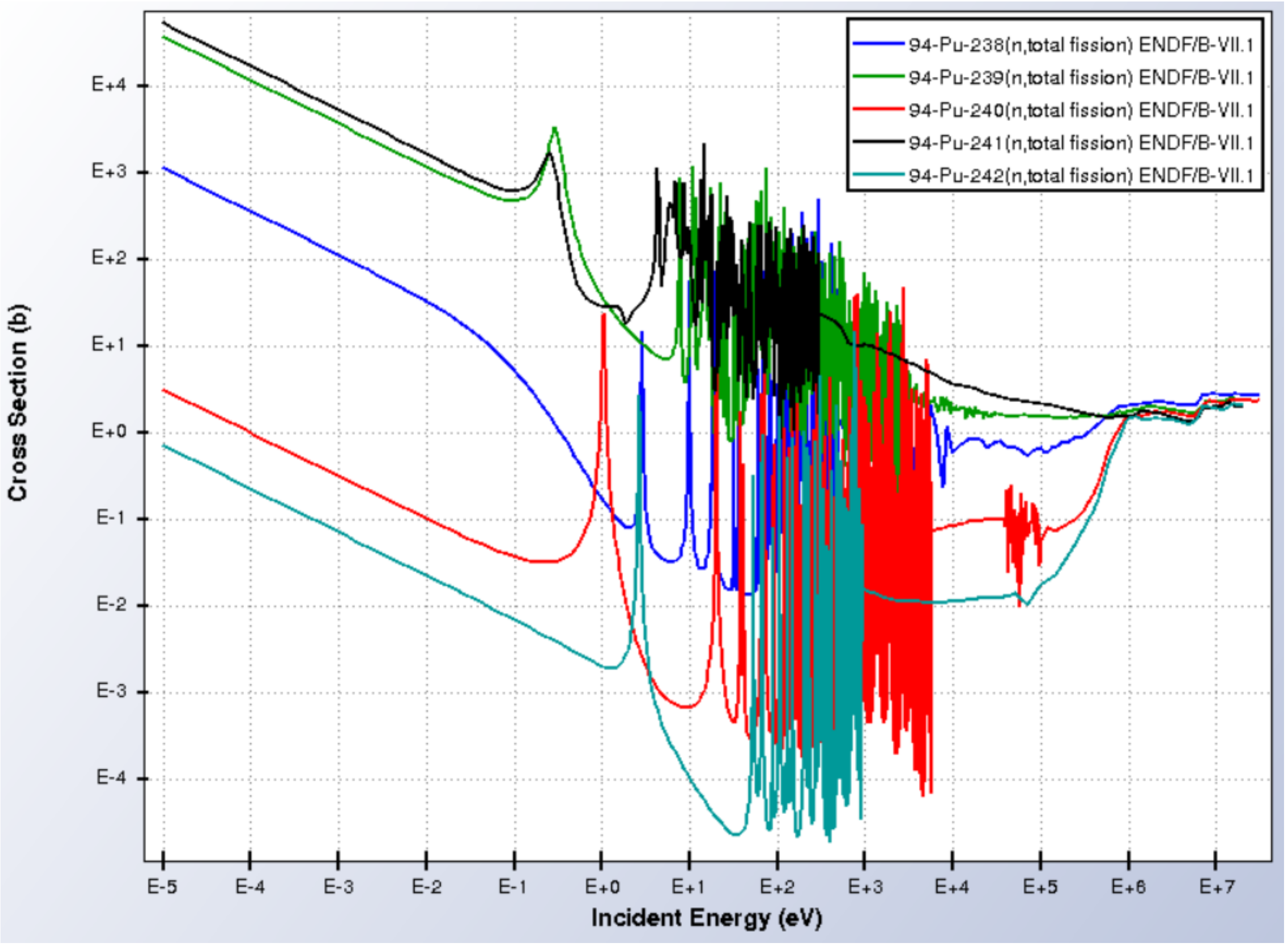
THE UNIVERSITY OF TENNESSEE KNOXVILLE

On the need for integrating physics-based depletion into fuel cycle assessment tools

Thermal-spectrum one-group cross-sections are highly sensitive to burnup & initial compositions



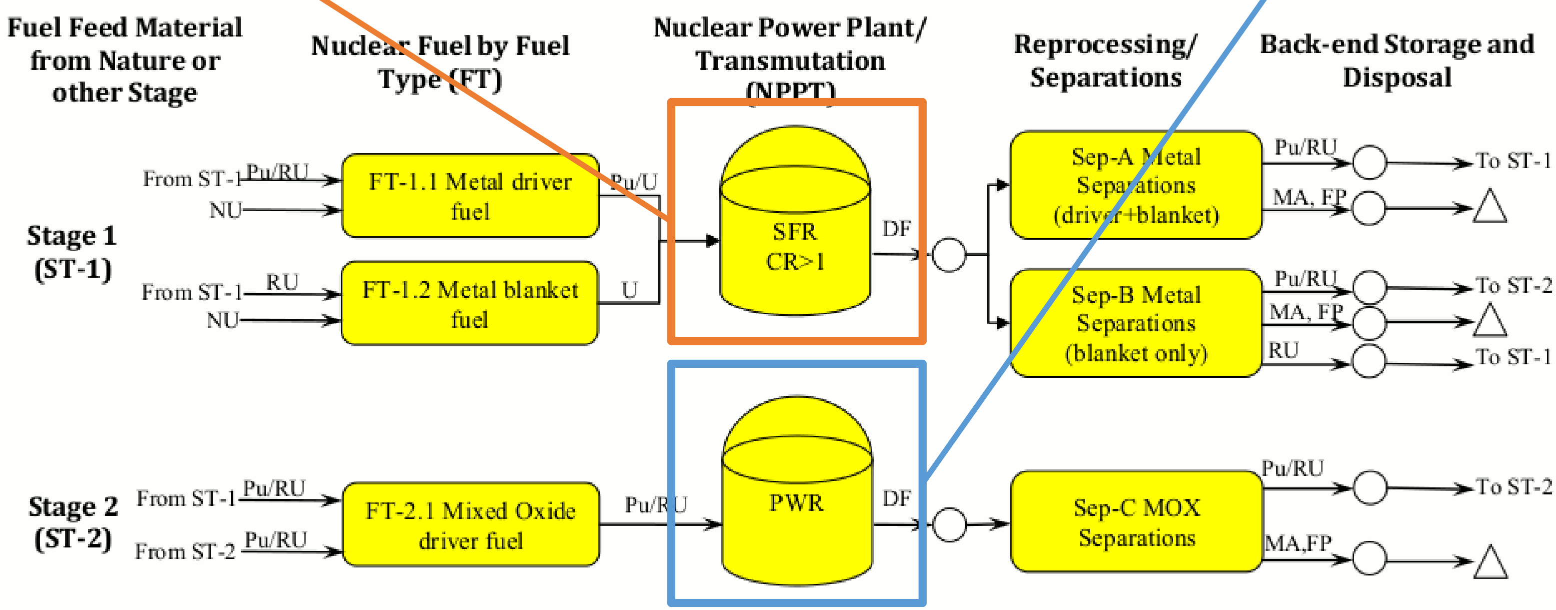
Reaction cross-sections show much higher sensitivity to perturbations in thermal energy region vs. fast



Example: EG29 scenario

Fast spectrum: **low CX sensitivity**:
Physics **desirable**, not "**necessary**"

Thermal spectrum: **high CX sensitivity**:
CXs highly sensitive to (dynamic) initial Pu composition & burnup; **physics necessary**



Note: Only primary material flows are shown. Material flows from imperfect separations (losses), low-level waste, and other secondary streams that will be produced in performing various fuel cycle functions are not shown.

Legend:

NU = Natural Uranium
DU = Depleted Uranium
LEU = Low-enriched Uranium
RU = Recovered Uranium

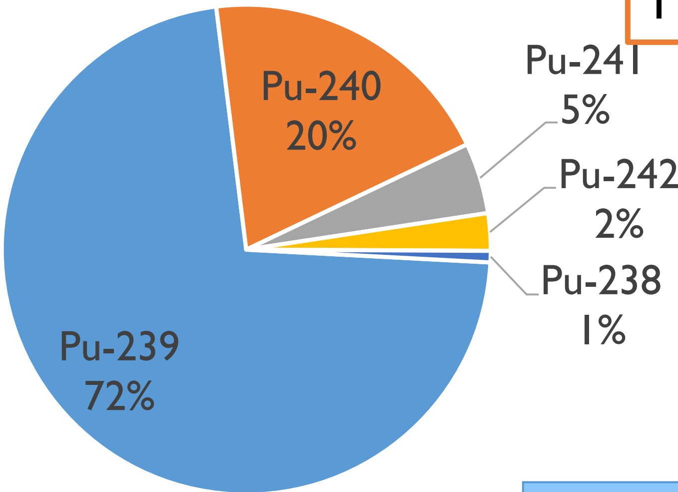
DF = Discharged Fuel
FP = Fission Products
TRU = Transuranics
MA = Minor Actinides

PWR = Pressurized Water Reactor
SFR = Sodium Fast Reactor
UOX = Uranium Oxide
MOX = Mixed Oxide

△ = Nuclear Waste Disposal
○ = Nuclear Material Storage
→ = Nuclear Material Transport
Pu/RU = Co-separated products

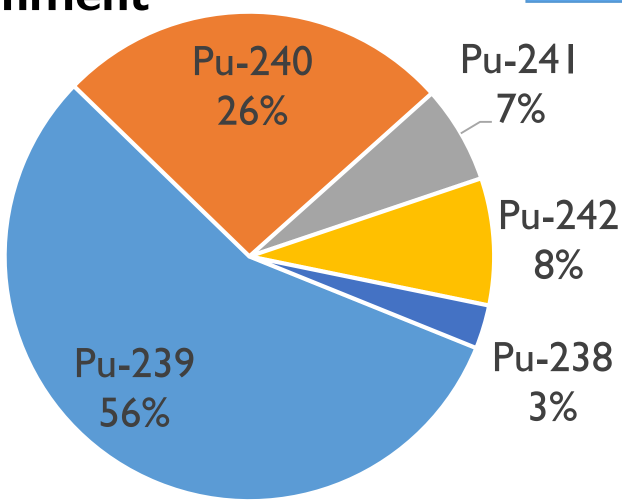
A tale of two MOXes

24.5 GWd/MTU



W17x17 (PWR)
4.0 % enrichment

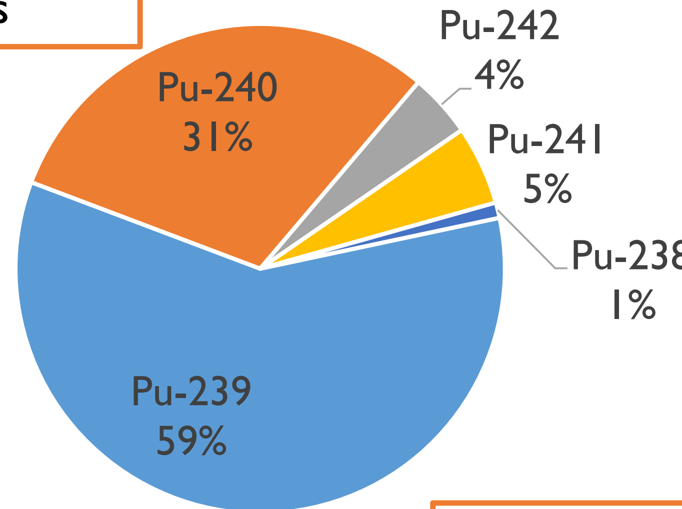
49.0 GWd/MTU



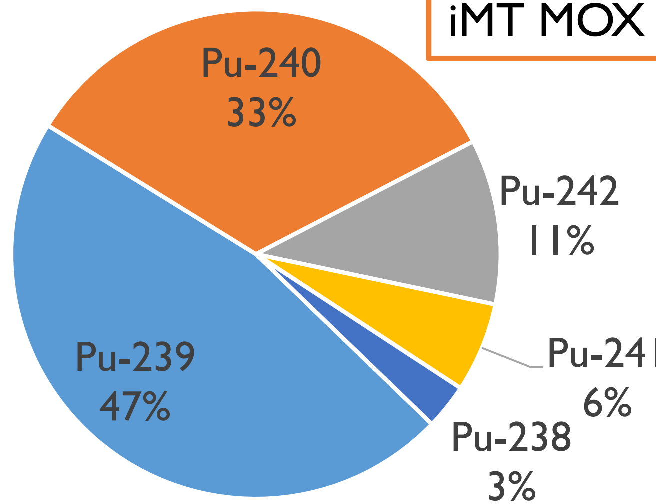
MOX 17x17 (PWR)
8% Pu / NU
33 GWd/MTHM
1 MTHM basis

Depletion

61.1 kgPu / iMT MOX



59.9 kgPu / iMT MOX



Development and function of an Origen-based Reactor Analysis module (CyBORG)

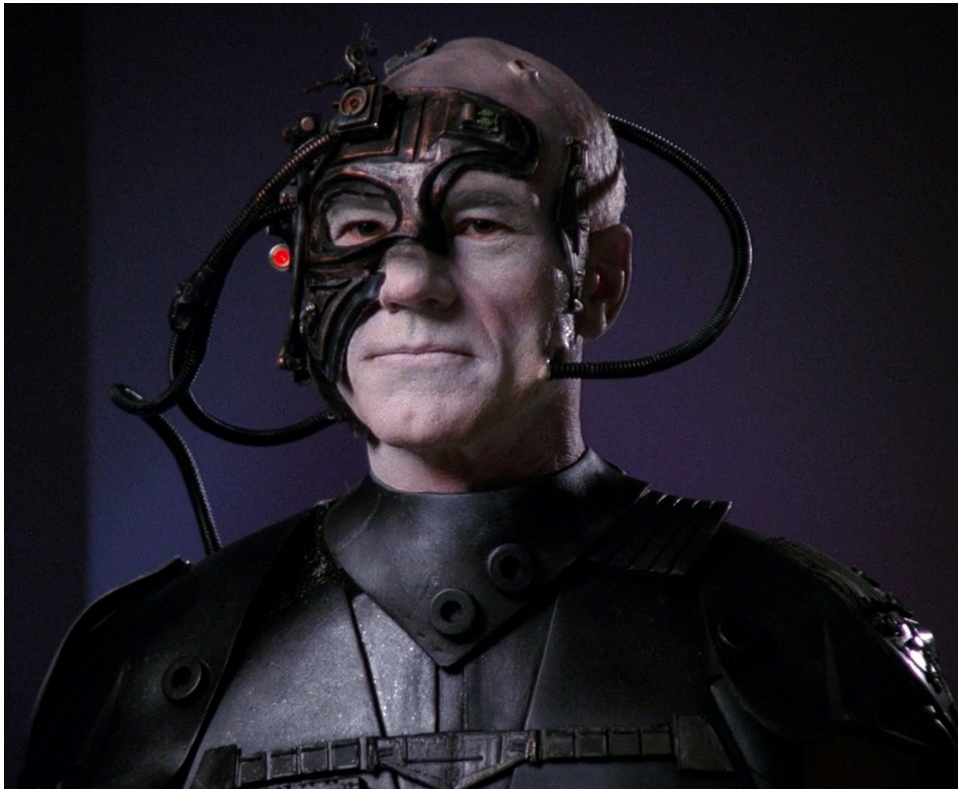
Cyclus is an agent-based nuclear fuel cycle simulator based upon a dynamic resource exchange engine

- Cyclus is an open-source fuel cycle simulator developed & maintained by the University of Wisconsin-Madison
- Relies on an **agent-based** framework wherein fuel cycle facilities are represented as individual "agents"
  - Individual facility behaviors are described by "archetypes," which interact with the simulation by means of resource exchange
  - Agents can express ranked "preferences" for material types / flows
  - Cyclus' kernel, the **dynamic resource exchange**, seeks to satisfy all resource bids (buy/sell) within the system

http://fuelcycle.org

Introducing CyBORG: Cyclus-Based Origen

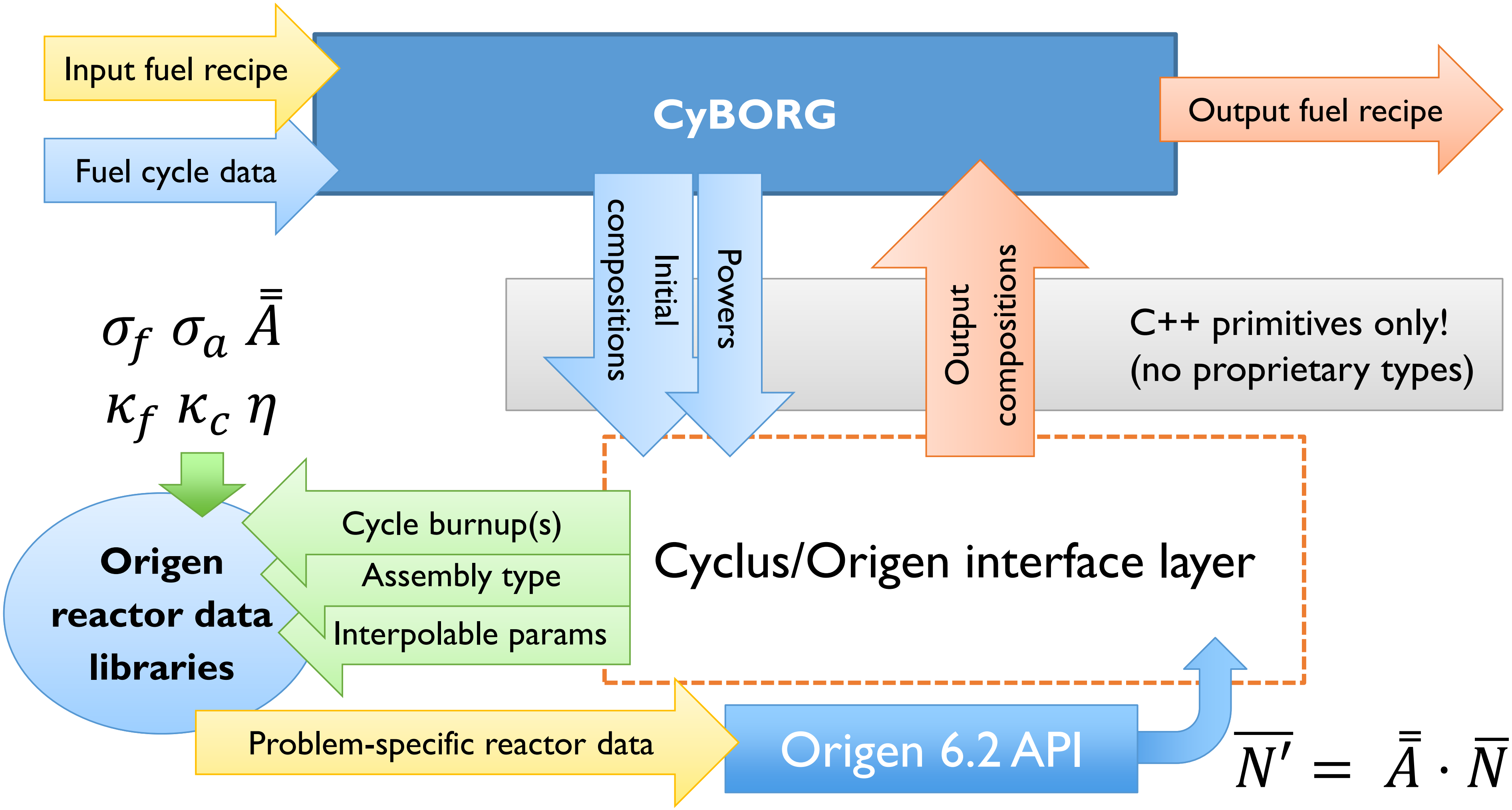
Assimilating physics-based depletion into fuel cycle scenario development



How do we incorporate Origen into Cyclus?

- Newest Origen API facilitates direct, in-memory calls to Origen solver
- By developing a portable, embeddable "**depletion engine**," Origen operations can be "wrapped" into a Cyclus-friendly format
  - "Depletion engine" builds from Origen API (from shared SCALE/Origen libraries)
  - **CMake-based configuration** allows for easy incorporation of required libraries & headers

CyBORG provides physics-based flexibility for reactor simulations within Cyclus via coupling with Origen



THE UNIVERSITY OF TENNESSEE KNOXVILLE

A CyBORG "shim layer" connects the Origen API to Cyclus while isolating Origen data types

- Goal of the "shim layer" is to connect Origen capabilities into an outside ecosystem without requiring awareness of Origen-specific types (and vice versa)
  - Data types passed through as C++ primitives & standard library types
- Shim layer performs "packaging" and "unpackaging" of Origen-specific data types (e.g., concentrations, interpolated reactor libraries, etc.)

CyBORG balances the performance cost of depletion via a "hash-and-cache" recipe generation strategy†

- Because of the relative cost of depletion (and reactor data library interpolation) relative to simulation time, we only to invoke physics-based depletion when necessary
- **Solution:** **Cache** output recipe (generated via depletion) to a **unique hash** based on relevant depletion conditions
  - e.g., initial enrichment, fuel type, discharge burnup

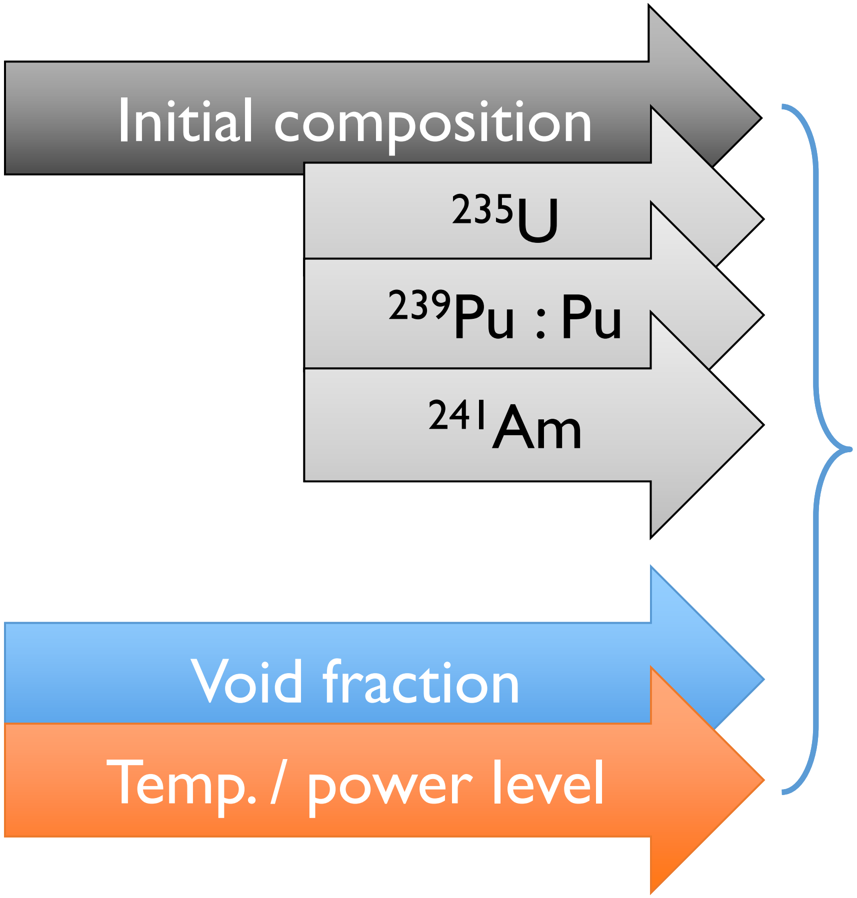
†Special thanks to Paul Wilson from UW-Madison for this idea

Generalized reactor data interpolation and processing within CyBORG

Accurate depletion calculations require problem-dependent nuclear data

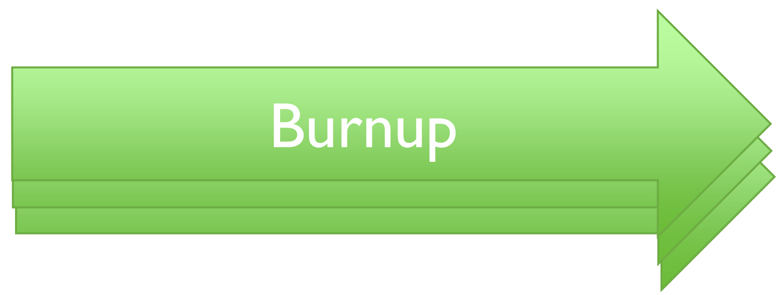
- **Problem-specific** nuclear data is required to dynamically produce accurate output inventories
  - Flux spectrum evolves with initial enrichment, void, burnup, etc.
- Origen handles this through interpolating pre-generated libraries to problem-specific conditions (e.g., initial enrichment, burnup, etc.)
  - Libraries developed transport calculations
  - "Assembly average" cross-sections

Example dimensions for N-D interpolation



Each parameter is treated as orthogonal, creating an <u>N-dimensional</u> interpolation (hyper-)space

Interpolation dimensions are <u>unspecified</u> until problem time (i.e., space is created dynamically)

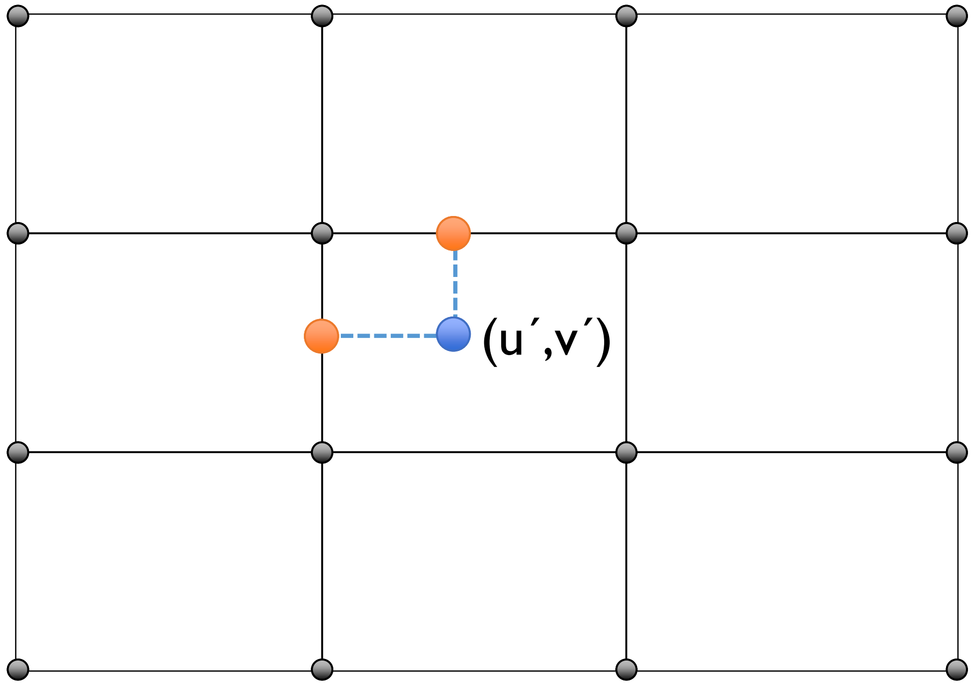


N-D space collapsed to 1-D and then interpolated along burnup dimension for problem-specific burnups

N-dimensional interpolation approach

$$\overline{\overline{A}}(u',v') = \sum_i c_i(u') \sum_j c_j(v') \overline{\overline{A}}^{S}_{(i,j)}$$

1. Determine adjacent 1-D "knots" for each dimension
2. Determine independent weight factors for each dimension
3. Apply appropriate weights to cross-section data across each dimension



Generalized interpolation allows new reactor types to easily be modeled using CyBORG

- Origen Library objects are correlated to interpolation data via the TagManager
  - Defines interpolable dimensions; e.g., initial enrichment, void fraction, etc.
- New libraries can easily be generated using the TRITON sequence in SCALE

Conclusions

The modern Origen API affords flexible depletion capabilities within larger frameworks

- CyBORG is but one example of how the Origen API can be applied outside of SCALE (building from the shared libraries & public API)
  - In the case of CyBORG, this allows for on-the-fly physics-based depletion given dynamic input fuel compositions
- Other frameworks may also benefit from embedding depletion, activation and decay capabilities via the Origen API

Current status of CyBORG & Origen API

- CyBORG is designed to work with the latest SCALE 6.2.2 release
- CyBORG v1.0 is now available for download and is fully compatible with Cyclus v1.5
- CyBORG source (including build configuration) & Origen API documentation available online

CyBORG repository: https://github.com/sskutnik/cyborg

Origen API docs: https://wawiesel.github.io/OrigenAPI-Demo/

Acknowledgements

- This work has been funded by a Nuclear Energy University Programs (NEUP) grant sponsored by the U.S. Department of Energy, Office of Nuclear Energy, award number DE-NE0000737
- Special thanks to Will Wieselquist (ORNL) who has collaborated on this development and leads development on the Origen API

Questions?

sskutnik@utk.edu

https://github.com/sskutnik/cyborg

THE UNIVERSITY OF
TENNESSEE
KNOXVILLE