

An Optimization Framework for Nuclear Fuel Cycle Simulation

Antoine Mouchamps^a, Damien Ernst^a

^aMontefiore Institute, University of Liège, Liège, Belgium

ARTICLE INFO

Keywords:

Nuclear fuel cycle simulation
Advanced nuclear fuel cycle

ABSTRACT

Advanced Nuclear Fuel Cycles (NFCs) involving fuel recycling and advanced reactor and dedicated system technologies can help address several of the challenges facing nuclear energy, such as uranium availability and the production of long-lived, highly radiotoxic radioactive waste. Determining how such cycles should be deployed, however, requires jointly determining facility deployment, material routing, and operations over long time horizons under various constraints. Existing NFC simulation tools provide increasingly detailed representations of fuel-cycle facilities and material transfers, but remain focused on evaluating predefined scenarios, with deployment specified by the user and material routing determined sequentially. Assessment is therefore limited to comparing a handful of hand-built scenarios through sensitivity and uncertainty analyses, rather than searching the deployment space for optimal fuel-cycle configurations.

This work introduces an optimization framework for NFC design and operation. The NFC problem is first represented as a directed multigraph, where nodes correspond to generic NFC facilities such as reactors or fuel fabrication plants, and edges define the connections enabling mass transfer between them. In this framework, the decision variables are the timing and commissioning of the facilities, as well as the routing of mass packages both within and between nodes, and various constraints can be added in order to limit the decision space. We then show that under this formalism, a fuel-cycle scenario is a solution of the resulting constrained decision problem, and state-of-the-art simulation models can be recovered as special cases in which the decision rules uniquely determine the system evolution.

1. Introduction

The growing need for low-carbon, dispatchable electricity and increasing concerns over energy supply security have recently renewed interest in nuclear energy in many countries [1]. While nuclear power offers clear advantages in terms of carbon intensity and baseload reliability, traditional open fuel cycles, in which fuel is used once and disposed of as high-level waste, lead to inefficient uranium utilization, large inventories of long-lived radioactive waste, and high long-term radiotoxicity [2]. These limitations, in addition to recent projections of uranium supply constraints as more and more reactors are commissioned globally [3, 4], motivate the development of advanced Nuclear Fuel Cycles (NFCs) capable of multi-recycling spent fuel, separating and transmuting minor actinides, and supporting advanced fast reactor technologies [5, 6].

Planning the deployment of such heterogeneous reactor fleets demands accurate tracking of material flows (nuclide compositions and facility inventories) across timescales ranging from months to centuries and beyond. NFC simulation tools must model the entire fuel chain from uranium mining and enrichment through reprocessing and waste conditioning and nuclear reactors themselves, while also handling composition-dependent constraints such as reprocessed uranium enrichment penalties [7] or plutonium-related timing constraints in plutonium-based fuel cycles [8]. Closed fuel cycles in particular introduce recycling loops and complex multi-facility network topologies that are significantly harder to simulate than the predominantly linear flows of open cycles.

Current simulation codes solve NFC problems using a discrete-time simulation-based approach, in which decisions

are taken one time step at a time. These tools allow to analyze prescribed scenarios (simulations of pre-determined reactor fleets and supporting facilities), but do not allow for exhaustively searching the space of possible scenarios and determine an optimal planning given a set of constraints. Answering the latter requires decisions to be determined jointly over the entire time horizon, under demand, capacity, and composition constraints, and possibly optimizing objectives such as minor actinide inventories or natural uranium consumption. This motivates the need for planning-based tools, in which a given scenario is not an input to be evaluated but rather the solution of a constrained decision problem.

The contributions of this paper are therefore twofold: first, a planning-and-control-oriented formalization of the NFC problem, in which existing simulation-based software represents special cases; second, an optimization software built on this formalization, which determines optimal scenarios over the whole time horizon rather than evaluating prescribed ones.

The remainder of this paper is organized as follows. Section 2 reviews existing NFC simulation codes and identifies the methodological gap addressed by this work. Section 3 then formalizes the NFC problem as a finite-horizon discrete-time control problem. Section 4 shows how existing simulation-based models are recovered as special cases of this formalization, using ANICCAv2 and Cyclus as illustrations. Finally, Section 5 presents an implementation of the framework as a mixed-integer quadratic program, together with a case study.

TODO: add conclusion

2. State of the art

Over the past decades, several simulation codes for nuclear fuel cycle analysis have been developed by various research institutions and countries, each tailored to the specific scenarios, hypotheses, and computational environments of its home institution. ANICCA, CLASS, COSI, DYMOND, FAMILY21, and NMB4.0 [9] are among the codes with the most active and recent research records.

A recurring pattern can be identified across most of these historical codes: the scenarios are structured around nuclear-engineering-specific facility categories rather than a generic network representation. COSI [10], CLASS [11, 12], FAMILY21 [13], and NMB4.0 [14] all organize their solvers around fixed front-end, reactor, and back-end components, and ANICCA [15] likewise hardcodes a fixed categorization of facility and process types. In all cases, the topology is inseparable from the physical interpretation of each facility, which makes these architectures inherently non-modular: adding a facility type or a new topology requires modifying the framework itself. These limitations motivated the development of a new generation of modular, facility-agnostic codes. Cyclus[16] and the upcoming DYMOND v7 [17] pursue this goal through agent-based architectures, in which facilities are autonomous agents exchanging materials and information through connections. The upcoming ANICCA v2 [18] instead adopts a graph-based architecture, representing the fuel cycle as a network of black-box modules and applies a sequential-modular-like approach from chemical process simulation to a centralized solver.

These modern codes are considerably more versatile and modular than their predecessors. Yet they remain, in essence, simulation-based: they evaluate a scenario that is predefined by the user, whose facility deployment is a fixed input rather than optimization results and whose material dispatch is, at best, the result of a per-step optimization. None of them addresses NFC optimization on the whole time horizon. This reflects a deeper, methodological limitation: existing NFC software have always been presented by detailing their implementations and features, without ever defining the general problem that their implementation actually models and tackles.

The present paper adopts the opposite approach and formalizes the NFC problem as a discrete-time control problem, in which dispatch and deployment are decision variables and a given scenario is determined based on a solution of the resulting constrained problem. In this formalization, classical simulation-based codes are recovered as the particular case in which the implemented constraints yield a single admissible decision at every time step.

3. Formalization

A Nuclear Fuel Cycle (NFC) system is described by a directed multigraph $G = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ represents a set of nodes, each node representing a block that can either represent a facility or a directed multigraph, and $\mathcal{E}$ represents

a set of edges connecting the nodes to one-another. In this graph, nodes represent facilities that can process and store masses in an internal inventory, while edges represent connections through which masses can be exchanged between nodes.

The system is modeled as a finite-horizon discrete-time mass-dispatch problem. In this system, masses are represented inside individual *packages* that are transferred between nodes. At every time step, package transfers through each edge are computed, inventories are updated, and packages are processed. Although the graph itself and its topology are fixed, i.e. the $\mathcal{N}$ and $\mathcal{E}$ sets are fixed, the activation of the nodes is to be determined (i.e. the commissioning/decommissioning of every facility). The simulation time horizon is denoted as T .

The decision problem consists of determining, at every time step $t \in [0, T)$, two quantities for each node: its activation status, i.e. whether the node is commissioned or not, and the allocation of the packages, i.e. where packages should move, both between nodes (as package transfers), and inside of every node (should packages be processed, stored, or sent out). From now-on, we define the solver as the module responsible to determine the solution of the decision process.

3.1. Mass transfer

A *package* $p \in \mathcal{P}$ represents a mass with a given composition. It is defined by the tuple

$$p[t] = (ct, m, \mathbf{w}[t]), \quad (1)$$

where $ct \in \mathcal{CT}$ is the commodity type of the package, $m \in \mathbb{R}_{>0}$ is its total mass, and $\mathbf{w}[t] \in [0, 1]^N$ is its mass-normalized composition vector, over N so-called *tracked* nuclides, such that

$$\mathbf{1}^\top \mathbf{w}[t] = 1 \quad \forall t \in [0, T). \quad (2)$$

The commodity type $ct \in \mathcal{CT}$ defines what the package represents, e.g. if it is a natural uranium package, a UOX fuel package, ..., and $\mathcal{CT}$ represents the space of all existing commodities. Since packages can represent radioactive nuclides that decay over time, their composition depend on t .

3.2. Node

A node n is a block that maintain a state and can receive, hold, process, and send packages. A node $n \in \mathcal{N}$ is formally defined by the tuple

$$n = (\mathcal{X}_n, \mathcal{U}_n, \mathcal{V}_n, \mathcal{Y}_n, g_n(\cdot), h_n(\cdot), \mathcal{W}_n, \Theta_n), \quad (3)$$

where

- $\mathcal{X}_n$ is the state space of the node;
- $\mathcal{U}_n$ is its decision space;
- $\mathcal{V}_n$ and $\mathcal{Y}_n$ are its input and output spaces, respectively;

- $g_n(\cdot)$ and $h_n(\cdot)$ are its state update and output functions, respectively;
- $\mathcal{W}_n$ and Θ_n are node parameters.

Each space is described in the remainder of this section.

3.2.1. State

The state of a node, denoted $x_n : x_n[t] \in \mathcal{X}_n, \forall t \in [0, T)$, is defined by a minimal tuple of four elements

$$x_n[t] = (I_n[t], H_n[t], \sigma_n[t], \eta_n[t]), \quad (4)$$

where:

- $I_n[t] \in \mathcal{P}_{n,I}^{P_{n,I}^{max}}$ is a list of maximum $P_{n,I}^{max}$ packages, representing the node inventory;
- $H_n[t] \in (\mathcal{P}^{|\mathcal{CT}|} \times \mathbb{N})_{n,H}^{P_{n,H}^{max}}$ is a list of maximum $P_{n,H}^{max}$ tuples; each tuple is composed of a tuple of maximum $|\mathcal{CT}|$ packages of distinct commodity type, and of a target time step $t' \in \mathbb{N} : t' > t$; $H_n[t]$ represents packages that are *in-processing* at time t and that will be fully processed into output packages at the corresponding target time step;
- $\sigma_n[t] \in \{0, 1\}$ is a boolean variable that represents if the node is commissioned ($\sigma_n[t] = 1$) or decommissioned ($\sigma_n[t] = 0$);
- $\eta_n[t] \in \{0, 1\}$ is a boolean variable that determines if the node has already been commissioned once ($\eta_n[t] = 1$) or no ($\eta_n[t] = 0$).

It should be noted that the state definition given above is a minimal set and that a node state could contain more information depending on the facility it represents, such as a reactor core for reactor facilities.

3.2.2. Input and output spaces

During a simulation timestep, a node can both receive and emit packages. The set of all possible package collections that can be received by a node defines its input space, while the set of all possible package collections that can be emitted defines its output space. Formally, the input space has the form

$$\mathcal{V}_n = \prod_{c \in C_{n,v}} \mathcal{P}_{n,c}^{P_{n,c}^{max}} \quad (5)$$

where $C_{n,v}$ are all the *input channels* of the node, and $P_{n,c}^{max}$ is the maximum number of packages that the node can accept through the channel c . Based on this definition, we can first define by $v_{n,c}$ the packages coming to the node n on its channel c . Then, the input of a node $v_n : v_n[t] \in \mathcal{V}_n, \forall t \in [0, T)$ is defined, at a time step t , by all the packages coming to the node through all of its input channels during t .

Similarly, the output space has the form

$$\mathcal{Y}_n = \prod_{c \in C_{n,y}} \mathcal{P}_{n,c}^{P_{n,c}^{max}} \quad (6)$$

with $C_{n,y}$ as all the output channels through which the node can emit packages. We can then define the output of the node, $y_n : y_n[t] \in \mathcal{Y}_n, \forall t \in [0, T)$, by all the packages that are leaving the node through all of its output channels, and $y_{n,c}[t]$ by the packages going out of the node on a channel c at a time step t .

3.2.3. Parameters

A node can be parametrized by two sets of parameters. The first set are constant parameters $\theta_n \in \Theta_n$ that define the node over the time horizon. This parameter set typically contains the physical characteristics of a facility, such as its capacities, processing times, or process parameters. The second set are time-dependent exogenous parameters $w_n : w_n[t] \in \mathcal{W}_n, \forall t \in [0, T)$ that condition the dynamics of the node over time. It typically represents quantities such as an electricity demand to be met or a time-dependent external constraint.

3.2.4. Decision variables

The decision variable of a node $u_n[t]$ is taken in the decision space $\mathcal{U}_n$. The main decisions taken by $u_n[t]$ are the activation of the node and its package allocation. The package allocation inside a node n consists in assigning mass fractions from a set of source packages to a set of destinations. The source packages are composed of all the packages entering the node through the input channels $C_{n,v}[t]$, all the packages stored in the inventory $I_n[t]$, and the all packages currently undergoing processing in $H_n[t]$. The destinations correspond to either storage in $I_n[t]$, storage in $H_n[t]$, or transfer to the output function $h_{n,\theta_n}(\cdot)$. Regarding the activation of the node, if a node is not commissioned, it should not receive, nor send, nor process any packages.

3.2.5. Node dynamics

The dynamics of the node are modeled by two functions: the state update function $g_{n,\theta_n}(\cdot)$ that updates the state of the node between each time step, and the output function $h_{n,\theta_n}(\cdot)$ that computes its output. Formally, they are defined by

$$x_n[t+1] = g_{n,\theta_n}(t, x_n[t], u_n[t], v_n[t], w_n[t]) \quad (7)$$

$$y_n[t] = h_{n,\theta_n}(t, x_n[t], u_n[t], v_n[t], w_n[t]), \quad (8)$$

where $x_n[t+1]$ is the updated state and $y_n[t]$ contains all the output packages. Together, $h_n(\cdot)$ and $g_n(\cdot)$ are responsible for processing packages, routing packages to the output channels, adding/removing packages from the inventory and the in-processing list, updating commissioning-related variables, and handling any other feature modeled in the node.

It should be noted that in many cases, the output function does not depend on the current input $v_n[t]$, which is typically the case for nodes that process packages and whose processing spans multiple time steps. Nodes whose output function depends on their input packages are referred to as direct-feedthrough nodes, and non-direct-feedthrough nodes are grouped in the set $\mathcal{N}^* \subseteq \mathcal{N}$. Additionally, since the output of non-direct-feedthrough nodes does not depend on $v_n[t]$, the output function of such nodes will be denoted as

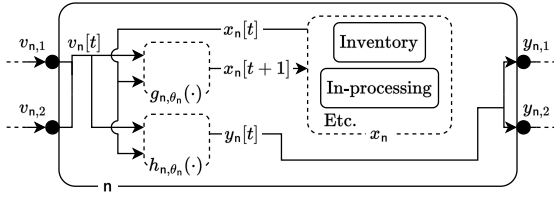


Figure 1: Structure of a direct-feedthrough node n with two input and two output channels. Each arrow represents a dependency link.

$h_{n,\theta_n}^*(\cdot)$, such that

$$y_n[t] = h_{n,\theta_n}^*(t, x_n[t], u_n[t], w_n[t]). \quad (9)$$

Beyond these requirements, no hypothesis is imposed on the structure of $g_{n,\theta_n}(\cdot)$ and $h_{n,\theta_n}(\cdot)$ in the general definition of a node. This expressiveness allows a node to model a wide range of behaviors, such as package processing, throughput limitations on each channel, or inventory limits, each expressed through the state update and output functions, the admissible set $\mathcal{U}_n(t, x_n[t])$, and the node parameters.

3.3. Edge

An edge $e \in \mathcal{E}$ is responsible for linking a node output channel to another node input channel, in order to allow for package transfer between the nodes. Formally, each edge is characterized by the tuple

$$e = (n_{e,s}, n_{e,d}, c_{e,s}, c_{e,d}) \quad (10)$$

where:

- $n_{e,s} \in \mathcal{N}$ and $n_{e,d} \in \mathcal{N}$ represent the source and destination node, respectively;
- $c_{e,s} \in \mathcal{C}$ and $c_{e,d} \in \mathcal{C}$ represent the output and input channels of $n_{e,s}$ and $n_{e,d}$, respectively.

Edges directly link a single output channel of the source node to a single input channel of the destination node, effectively coupling all the decision problems of all nodes. As node channels all have their own space $\mathcal{P}_{n,c}^{P_{n,c}^{max}}$ an edge is only valid if it links two channels sharing the same space, i.e. if

$$P_{n_{e,s},c_{e,s}} = P_{n_{e,d},c_{e,d}} = P_e. \quad (11)$$

In addition, the following constraint is added for each edge:

$$y_{n_{e,s},c_{e,s}}[t] - v_{n_{e,d},c_{e,d}}[t] = \mathbf{0}. \quad (12)$$

3.4. Step evaluation

The information flow within a single node is depicted in Figure 1. At every time step, the state $x_n[t]$, the decision variable $u_n[t]$, and the input packages $v_n[t]$ are used to produce the output packages $y_n[t]$ and the state at the next time step $x_n[t+1]$.

In order to execute each node of the graph, one has to pay attention to the order of this execution, as the output of one

node has an influence on the input of subsequent nodes. This is especially the case regarding direct-feedthrough nodes, whose output directly depend on their input. For a graph G , an execution order can be determined under the condition that no cycle composed only of direct-feedthrough nodes exists in G . Otherwise, the output of every node of a given cycle would be required in order to evaluate the others nodes of this cycle. Formally, denoting by $\mathcal{Z}$ the set of directed cycles of G , each cycle $z \in \mathcal{Z}$ being identified with the set of nodes it contains, an NFC graph is valid if and only if

$$\forall z \in \mathcal{Z}, \exists n \in \mathcal{N}_z : n \in \mathcal{N} \setminus \mathcal{N}^*. \quad (13)$$

Equation 13 is not a strong limitation in practice, as most package transfer loop in an NFC graph contains at least one node whose processing spans multiple time steps, such as irradiation, cooling, or fuel fabrication, which usually makes sure that at least one non-direct-feedthrough node is present in each cycle.

Assuming that all the decision variables are determined, that the joint decision is admissible on the graph and that Equation 13 is respected, an execution order can be determined, such as the one explained below. First, one has to determine a topological ordering (denoted D) of the direct-feedthrough nodes of G . This ordering is based on the sub-graph obtained by removing every non-direct-feedthrough node of G and their connected edges. The topological order D exists as long as Equation 13 holds, and is fixed over the time horizon, as the topology of G is fixed. Second, for each non-direct-feedthrough node, its output can be computed using Equation 8 in any order, since the computation does not depend on any node input. Third, for each direct-feedthrough node iterated upon following the order given by D , its input $v_n[t]$ can be constructed by iterating over its input channels, and then its output $y_n[t]$ can be computed using Equation 9. Lastly, after collecting the remaining inputs $v_n[t]$ in any order, the state of every node can be updated according to Equation 7, in any order as well.

3.5. Sub-graph

As stated at the beginning of the section, each node can represent a subgraph, while still correspond to the definition of a node given by Equation 3. Let us name the graph of nodes that are not contained in another node by the main graph $G = (\mathcal{N}, \mathcal{E})$. Denoting by $G_{n'} = (\mathcal{N}_{n'}, \mathcal{E}_{n'})$ the subgraph contained in a node $n' \in \mathcal{N}$, the spaces $\mathcal{X}_{n'}$, $\mathcal{W}_{n'}$ and $\Theta_{n'}$ of this node can be seen as the Cartesian product of all the respective spaces of every node in $\mathcal{N}_{n'}$. Regarding the input (resp. output) space $\mathcal{V}_{n'}$ (resp. $\mathcal{Y}_{n'}$) of n' , it is the concatenation of all the input (resp. output) channel spaces of the nodes of $G_{n'}$ that are connected by edges to nodes of G . Then, as will be explained in the next section, $\mathcal{U}_{n'}$ can be represented as a subspace of the Cartesian product of the decision space of each node. Lastly, the state update and output functions can be seen as the abstraction of the sub-graph update, as described in Section 3.4.

As a last remark, it should be noted that the main graph can then be seen as a single node as well, with no input (resp.

output) channel nor space, since no inbound (resp. outbound) edges are connected to nodes outside of G . Doing so, the step evaluation of this global node, which essentially corresponds to the step evaluation of a graph as detailed in Section 3.4, can be abstracted by a single state update function $g(\cdot)$. This function is such that

$$x[t+1] = g_\theta(t, x[t], u[t], w[t]) \quad (14)$$

corresponds to advancing the state of the graph by one time step. In Equation 14, $x[t] = (x_n[t])_{n \in \mathcal{N}}$, $u[t] = (u_n[t])_{n \in \mathcal{N}}$, $\theta = (\theta_n[t])_{n \in \mathcal{N}}$ and $w[t] = (w_n[t])_{n \in \mathcal{N}}$ correspond to the graph state, the joint decision on the graph, and both joint sets of the parameters, respectively.

3.6. Problem statement

Although each node defines a local feedforward decision problem at every time step, the edge constraints between node inputs and outputs of Equation 12 make the decision problem global on the graph. Consequently, the decisions of all nodes are coupled, through both the edge constraints and the state update and output functions.

This coupling implies that the joint decision space is not the mere Cartesian product of the nodal decision spaces. Denoting by $u[t] = (u_n[t])_{n \in \mathcal{N}}$ the joint decision and by $x[t] = (x_n[t])_{n \in \mathcal{N}}$ the joint state, the joint admissible set $\mathcal{U}$ gathers the joint decisions whose induced node inputs and outputs satisfy the edge constraints and node dynamics, and is therefore in general a strict subset of the Cartesian product of the local sets:

$$\mathcal{U}(t, x[t]) \subseteq \prod_{n \in \mathcal{N}} \mathcal{U}_n(t, x_n[t]). \quad (15)$$

The dimensions removed from the Cartesian product correspond to the degrees of freedom lost due to the coupling.

The general decision problem, which consists of finding the joint decision at every time step depending on the system constraints, dynamics, and parameters, can now be defined formally as

$$\text{find:} \quad (16a)$$

$$\left\{ u[t] \quad \forall t \in [0, T) \right. \quad (16b)$$

subject to , $\forall t \in [0, T)$:

$$\begin{cases} y_{n_{e,s}, b_{e,s}}[t] - v_{n_{e,d}, a_{e,d}}[t] = 0 & \forall e \in \mathcal{E} \\ x_n[t+1] = g_{n,\theta_n}(t, x_n[t], u_n[t], v_n[t], w_n[t]) & \forall n \in \mathcal{N} \\ y_n[t] = h_{n,\theta_n}(t, x_n[t], u_n[t], v_n[t], w_n[t]) & \forall n \notin \mathcal{N}^* \\ y_n[t] = h_{n,\theta_n}^*(t, x_n[t], u_n[t], w_n[t]) & \forall n \in \mathcal{N}^* \end{cases} \quad (16c)$$

given:

$$\begin{cases} G = (\mathcal{N}, \mathcal{E}) \\ \theta = \theta_n & \forall n \in \mathcal{N} \\ x[0] = x_n[0] & \forall n \in \mathcal{N} \\ w[t] = w_n[t] & \forall n \in \mathcal{N}, t \in [0, T) \end{cases} \quad (16d)$$

Equation 16 defines, for the whole horizon, a single feasibility problem whose unknown is the complete decision sequence. Denoting this sequence by $\mathbf{u} = (u[0], \dots, u[T-1])$ and by Ω the set of sequences satisfying Equation 16c given Equation 16d, the general problem reads: find $\mathbf{u} \in \Omega$. It should however be noted that at this level, no hypothesis is made on the existence or uniqueness of $\mathbf{u}$.

When Ω contains more than one element, the remaining degrees of freedom can be exploited to select a decision sequence that is optimal with respect to one or several objectives. An objective is expressed as a function of the trajectory (sequence of generated states and decisions) generated by $\mathbf{u}$ over the whole horizon. Since a given commissioning and operational planning may be evaluated against several, possibly conflicting, objectives, such as minimizing an economic cost, a material inventory, or a given mass transfer, the general problem is posed in a multi-objective form. Given K objective functions gathered in the vector $\mathbf{J}(\mathbf{u}) = (J_1(\mathbf{u}), \dots, J_K(\mathbf{u}))$, the optimization problem reads

$$\mathbf{u}^* \in \arg \min_{\mathbf{u} \in \Omega} \mathbf{J}(\mathbf{u}), \quad (17)$$

understood in the Pareto sense: a solution $\mathbf{u}^*$ is sought such that no other feasible sequence improves one objective without degrading another. The single-objective case is recovered for $K = 1$.

It should be noted that a same quantity of interest may equally be used as an objective, as above, or as a constraint defining Ω through the state update or output functions, for instance by bounding a material inventory rather than minimizing it. The choice between the two is part of the problem specification.

4. Expressing simulation-based models

5. MIQP formulation and case study

CRediT authorship contribution statement

Antoine Mouchamps: Writing review & editing, Methodology, Formal analysis, Conceptualization. **Damien Ernst:** Writing review & editing, Validation, Project administration, Funding acquisition.

References

- [1] International Energy Agency, The path to a new era for nuclear energy, Technical Report, IEA, Paris, France, 2025. URL: <https://www.iea.org/reports/the-path-to-a-new-era-for-nuclear-energy>.
- [2] R. Taylor, W. Bodel, L. Stamford, G. Butler, A review of environmental and economic implications of closing the nuclear fuel cycle part one: wastes and environmental impacts, *Energies* 15 (2022) 1433. doi:10.3390/en15041433.
- [3] S. Shannak, L. Cochrane, D. Bobarykina, Global uranium market dynamics: analysis and future implications, *International Journal of Sustainable Energy* 44 (2025). doi:10.1080/14786451.2025.2457376.
- [4] T. Wang, Z. Wu, W. Xiong, X. Pan, X. Ye, X. Liu, The impact of uranium resource constraints on chinas nuclear power development, *Energies* 18 (2025) 1507. doi:10.3390/en18061507.

- [5] R. Taylor, W. Bodel, G. Butler, A review of environmental and economic implications of closing the nuclear fuel cycle part two: economic impacts, *Energies* 15 (2022) 2472. doi:10.3390/en15072472.
- [6] W. I. Ko, F. Gao, Economic analysis of different nuclear fuel cycle options, *Science and Technology of Nuclear Installations* 2012 (2012) 1–10. doi:10.1155/2012/293467.
- [7] G. D. DelCul, L. D. Trowbridge, J.-P. Renier, R. J. Ellis, K. A. Williams, B. B. Spencer, E. D. Collins, Analysis of the reuse of uranium recovered from the reprocessing of commercial LWR spent fuel, Technical Report, Office of Scientific and Technical Information (OSTI), 2009. doi:10.2172/947143.
- [8] T. Maldague, S. Pilate, A. F. Renard, Impact of plutonium and americium recycling in pwr on mox fuel fabrication, in: *Actinide and Fission Product Partitioning and Transmutation, Proceedings of the 3rd NEA International Information Exchange Meeting*, OECD Nuclear Energy Agency, Cadarache, France, 1994, pp. 215–223. URL: <https://oecd-neo.org/pt/docs/iem/cadarache94/session-4/Session4Maldague.pdf>.
- [9] Nuclear Energy Agency, Nuclear Fuel Cycle Codes Catalogue — oecd-neo.org, https://www.oecd-neo.org/jcms/pl_100633/nuclear-fuel-cycle-codes-catalogue, 2025. [Accessed 25-03-2026].
- [10] C. Coquelet-Pascal, M. Tiphine, G. Krivtchik, D. Freynet, C. Cany, R. Eschbach, C. Chabert, Cosi6: A tool for nuclear transition scenario studies and application to sfr deployment scenarios with minor actinide transmutation, *Nuclear Technology* 192 (2015) 91110. doi:10.13182/nt15-20.
- [11] B. Mouginot, B. Leniau, N. Thiollère, M. Ernoult, S. David, X. Doligez, A. Bidaud, O. Meplan, R. Montesanto, G. Bellot, J.-B. Clavel, I. Duhamel, E. Letang, J. Miss, Core Library for Advanced Scenario Simulation (C.L.A.S.S.): Principles and Applications, Poster presented at the International Conference on the Role of Reactor Physics toward a Sustainable Future (PHYSOR 2014), 2014. URL: <https://in2p3.hal.science/in2p3-01119778>.
- [12] CLASS Development Team, Class: Core library for advanced scenario simulation, 2026. URL: <https://gitlab.in2p3.fr/sens/CLASS>.
- [13] OECD Nuclear Energy Agency, Benchmark Study on Nuclear Fuel Cycle Transition Scenarios Analysis Codes, Technical Report NEA/NSC/WPFC/DOC(2012)16, OECD Nuclear Energy Agency, Paris, France, 2012. URL: https://www.oecd-neo.org/jcms/pl_19182/benchmark-study-on-nuclear-fuel-cycle-transition-scenarios-analysis-codes, WPFC Expert Group on Advanced Fuel Cycle Scenarios (EGAFCS).
- [14] T. Okamura, R. Katano, A. Oizumi, K. Nishihara, M. Nakase, H. Asano, K. Takeshita, Nmb4.0: development of integrated nuclear fuel cycle simulator from the front to back-end, *EPJ Nuclear Sciences & Technologies* 7 (2021) 19. doi:10.1051/epjn/2021019.
- [15] I. M. Rodríguez, A. Hernández-Solís, N. Messaoudi, G. Van den Eynde, The nuclear fuel cycle code anicca: Verification and a case study for the phase out of belgian nuclear power with minor actinide transmutation, *Nuclear Engineering and Technology* 52 (2020) 2274–2284. doi:10.1016/j.net.2020.04.004.
- [16] K. D. Huff, M. J. Gidden, R. W. Carlsen, R. R. Flanagan, M. B. McGarry, A. C. Opatowsky, E. A. Schneider, A. M. Scopatz, P. P. Wilson, Fundamental concepts in the cyclus nuclear fuel cycle simulation framework, *Advances in Engineering Software* 94 (2016) 46–59. doi:10.1016/j.advengsoft.2016.01.014.
- [17] Argonne National Laboratory, Dymond: Dynamic model of nuclear deployment, 2026. URL: <https://www.anl.gov/nse/dymond>.
- [18] A. Mouchamps, Design and Implementation of a Modular Modeling Framework for the ANICCA Nuclear Fuel Cycle Simulation Code, Master's thesis, Université de Liège, Liège, Belgium, 2026. Internship carried out at the Belgian Nuclear Research Centre (SCK CEN).