# Harnessing Reduced Order Models for Efficient Burnup Calculations

**Jonathan Pilgram**
**August 16, 2024**

TU Delft

# *Going Nuclear*
# Harnessing Reduced Order Models for Efficient Burnup Calculations

by

Jonathan Pilgram

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday August 16, 2024 at 14:00.

| | | |
|---|---|---|
| Student number: | 4969839 | |
| Project duration: | December 1, 2023 – August 9, 2024 | |
| Thesis committee: | Dr. ir. D. Lathouwers, | TU Delft, supervisor |
| | Dr. Z. Perkó, | TU Delft |
| | Prof. dr. ir. M. B. van Gijzen, | TU Delft |

Cover: Reactor Institute Delft by Reg Lindemans

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

# Abstract

The goal of the thesis is to find a Reduced Order Modelling method that speeds up burnup simulations—as encountered in the core of a Molten Salt Fast Reactor—while keeping accuracy loss minimal. Four different methods have been tested: Proper Orthogonal Decomposition (POD), heuristically corrected POD, Balanced Truncation (BT), and Balanced Proper Orthogonal Decomposition (BPOD). The first two methods are inadequate, because of stability issues. The third is stable, but fails to execute for burnup simulations. The fourth, a midway method between the first and third, does work for burnup simulations.

Using BPOD with 4 orders for a 10 year burnup simulation of 1650 nuclides with 1000 simulation steps, we find a normalised relative error of $10^{-5}$ both for the total model and each nuclide individually. The execution time per simulation step is reduced to $10^{-5}$ s. These results are a factor 1000 better than known alternatives such as the ORIGEN burnup program.

The conclusion contains recommendations for incorporating different fuel mixtures and non-linearity of the burnup equation in the BPOD. The method could be generalised to handle arbitrary burnup problems with a single Reduced Order Model.

# Treatise on the Significance of Technical Style

During my bachelor studies of astronomy and physics, I made a life-changing discovery. The discovery was that, although smart people get far, the people able to communicate their ideas well get further. As a result, I studied rhetorics for a year. Without any real empirical backing (except, maybe some of the more cited papers in the list of references) I make the claim that there is a correlation between the amount of citations and the style of a paper.

One might ask: what exactly is meant by style?

Style exists only in dichotomy with content. For example, in the context of a scientific paper, the content would be the idea that the writer means to communicate and the style is everything else. Every word can be chosen differently, there are a thousand ways to structure a paper and every visual aid can be modified indefinitely. Interesting is that although there is are near infinite ways to change style, there are some which simply work better than others. The effect of good style is that readers will grasp the idea that is being conveyed correctly without losing attention. Conversly, bad style confuses or even infuriates the reader. I could even argue that content is not merely the idea that the writer tries to convey, but rather the idea the reader gets. Style and content thus live in duality to each other. Content cannot be conveyed without style and style defines the content.

So, then, what makes technical style "technical" compared to common style?

I identify three aspects that differ: the addition of mathematical language, accurate visual aids, and presupposed expertise. Mathematics allows us to express ideas about the natural world in a very precise and concise language at the cost of having the strictest possible grammar. Any mistake and the mathematical sentence loses its intended meaning. Besides, it is easy to obfuscate concepts with mathematics—even a technical expert may have difficulty understanding complex mathematics during a first reading. In my opinion, mathematics should be used sparingly, and when used, it should be both effective and correct. Whereas large tables and graphs are in most fields of studies eschewed, technicians love them. Not everyone can instantly visualise $y = (x - 2)(x + 2)$, but a graph would easily reveal a parabola with a minimum at $(0, -4)$. I'm of the opinion that it is nigh impossible to have too many graphical aids, but each one of them should be kept as clean and simple as possible. Lastly, writing for technical experts gives the unique opportunity to assume a huge body of shared concepts, which can be used to your advantage. There are generally two types of technical experts: those who understand the world through the abstract and those who understand the abstract by looking at the world. The first thrive by mathematics and exact, short sentences, while the second thrive by visual aids and more prose-like explanations.

Lastly, I would like to refer to the beautiful definition for style of Willem Elsschot in his 1933 novel 'Kaas', which conceptualizes style as writing the minimal necessary to clearly communicate, while keeping it interesting by surprising the reader with an increasing rhythm.

# Acknowledgement

During my work, I was reminded of the following quote regularly:

> If I have seen further, it is by standing on the shoulders of giants.
> — *Isaac Newton*

I now want to thank some of the giants who let me climb on their shoulders to allow me to peer further.

First and foremost, I would like to acknowledge Danny Lathouwers, who not only got me interested during a course of nuclear reactor physics, but also happily supervised my project. Besides being brilliantly smart, he also gave me space to work in a unique style that suits only me. I'm going to miss our discussions about the nuclear reactor physics, which we both find wonderfully interesting. Thank you.

I want to express my sincere gratitude to Sridhar Chellappa, who responded to my email with questions about a paper of his research group and proceeded to give me guidance through mail, exchanged ideas during a meeting and was genuinely interested in my work.

Furthermore, I would like to thank Jort Bouma for the flying start. During the most stressful period of his thesis, he took a full day to help me get started.

To the Reactor Physics group of Delft: thanks for including me. I'm looking forward to playing another match of volleyball against you all.

Then there are the giants of my personal life. You are all wonderful people, whom I'm extremely grateful to have around. I know I can always depend on you for fun, letting off steam and helping me with my troubles.

To all my family, those still around and those whom I had to leave behind, without you I would not have been able to get a master's degree, not only because of the support, but also because there would not have been a noteworthy 'me' to speak of. I hope you share in the pride of this and future achievements.

To my wide range of friends, whom I had to miss during the final stretch of my thesis, I look forward to seeing you all again. I would like to name some of you for going above and beyond expectations regarding my thesis:

Benjamin van Ommen, my ever helpful dear friend: I was happy to learn physics along with you, am pleased to have your support, and will happily see you long into the future.

Julius Hendrix, my dear (not always helpful) friend: Thank you for showing me the way in project management and help me find the necessary distraction.

Margo Witjes, my dear (ex) housemate and good friend: You helped me stay motivated and were there for me when I could not go further.

Cornel Marck, my dear old friend: my visit to you in London was not only very exciting, but also very fruitful. My programming would not have been so clean without your guidance.

Finally, I would like extend my gratitude to you, the reader, for your interest in my work!

# Contents

# Acronyms

**BPOD** Balanced Proper Orthogonal Decomposition. ii, 3, 43, 45, 46, 47, 48, 49, 50, 52, 68

**BT** Balanced Truncation. ii, 3, 35, 36, 38, 39, 40, 41, 42, 43, 45, 46, 51, 52

**FOM** Full Order Model. 2, 17, 20, 22, 23, 24, 25, 27, 29, 30, 32, 35, 38, 39, 40, 41, 42, 45, 46, 49, 51

**FP** Fission Products. 2, 6, 10, 16, 46, 47, 50

**MSFR** Molten Salt Fast Reactor. 3, 5, 6, 7, 8, 11, 15, 46, 48, 51, 52

**POD** Proper Orthogonal Decomposition. ii, 2, 3, 15, 16, 17, 18, 19, 20, 22, 23, 24, 25, 26, 27, 29, 30, 32, 33, 35, 38, 39, 40, 41, 42, 43, 44, 46, 49, 51, 52

**ROM** Reduced Order Modelling. 2, 3, 7, 16, 29, 38, 51

**ROM** Reduced Order Model. 8, 11, 14, 15, 17, 18, 19, 20, 22, 23, 24, 25, 26, 29, 30, 33, 35, 36, 38, 40, 41, 45, 46, 48, 49, 50, 51, 52

**ROMs** Reduced Order Models. 3, 15, 16, 17, 20, 22, 26, 27, 29, 30, 32, 33, 39, 40, 41, 42, 52

**SVD** Singular Value Decomposition. 17, 18, 19, 20, 25, 44, 46

# Chapter 1

# Introduction

In the landscape of renewable energy there is one that is most controversial: nuclear energy. Proponents argue that nuclear plants generate power without any carbon emission (Igini, 2023) on a small amount of space. For example, the small Dutch nuclear power plant in Borssele generates 0.48GW net (N.V. EPZ, 2023)—day and night, wind and no wind—on less than 10 hectares. A similar amount of solar energy, as generated by the Núñez de Balbao Photovoltaic Plant in Spain under ideal conditions during the day, requires 1000 hectares of space (Iberdrola corp., 2019).

So why did we not go nuclear? Critics argue nuclear plants do not solve the problems of a sustainable future. Although there is no carbon emission, a nuclear reactor produces radioactive waste, which can remain a problem for thousands of years (World Nuclear Association, 2022). Furthermore, there are legitimate safety concerns nuclear energy. Accidents can lead to disasters when mishandled, as the historical Chernobyl disaster showed (World Nuclear Association, 2024).

A more nuanced view is that nuclear energy is a potential solution for our future energy needs, but more research into the subject is required to mitigate the current safety and waste issues.

Nuclear reactor physicists around the world are working on improving nuclear reactor designs and waste management schemes, such as in the European MIMOSA project (MIMOSA, 2024). The MIMOSA project tries to incorporate molten salt reactors into a multiple cycle nuclear reactor scheme to optimise fuel efficiency and reduce the amount of nuclear waste (figure 1.1). Molten salt reactors—integral to MIMOSA—are heavily investigated, but are not yet in use. Challenges include mitigating corrosion of the reactor vessel, numerically simulating safety measures and modelling fuel burnup (Serp et al., 2014).

Fuel burnup simulations, numerically tracking the amount of each nuclide for days to years, are central to predicting requirements of the recycling schemes; we want to know how much of a particular isotope we need to process years into the future. SERPENT, a Monte Carlo neutron transport code (Leppänen et al., 2015), tracks 1726 different nuclides, with decay constants ranging from $10^{-20}$ seconds to $10^{20}$ seconds. The governing equation for the burnup process is a coupled linear set of ordinary differential equations

$$\frac{\mathrm{d}\mathbf{N}(t)}{\mathrm{d}t} = A\mathbf{N}(t), \tag{1.1}$$

where $\mathbf{N}(t)$ is the nuclide vector and $A$ is a square burnup matrix. Keeping track of a of a vector with 1650 nuclides is expensive, so schemes exist to speed up the simulation. Some of the schemes rely on eliminating short-lived nuclides, whilst others rely on mathematical

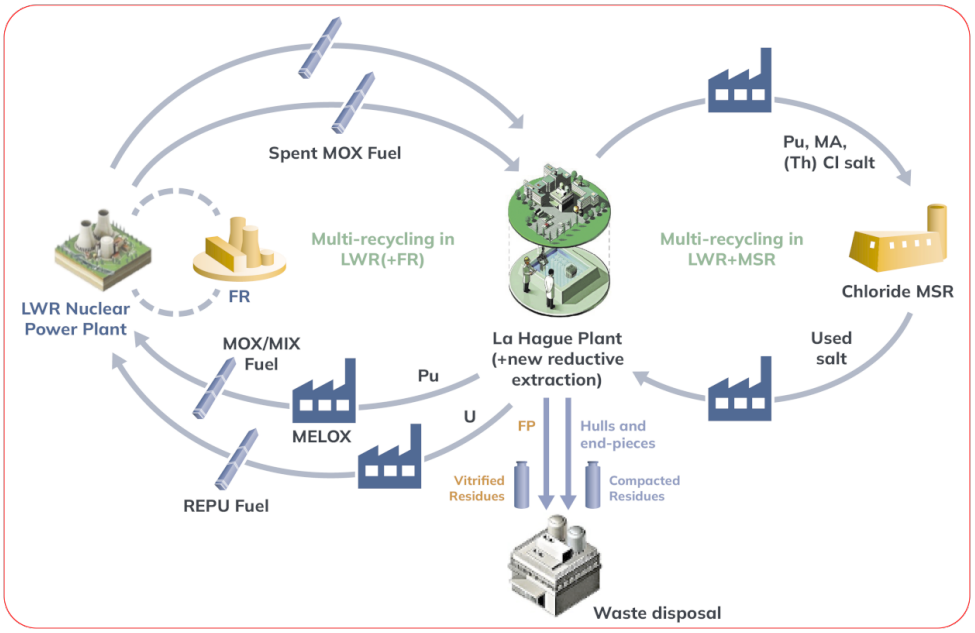



**Figure 1.1: General overview of the MIMOSA project**
Light water reactors (LWR)—with the possible addition of fast reactors (FR)—produce spent mixed oxide (MOX) fuel, which is sent to the La Hague reprocessing plant. The reprocessing plant has many output products. Plutionium can be processed in the MELOX plant with fresh uranium to new MOX/MIX fuel (MIX meaning mixed fuel) for the light water reactors. From Uranium reprocessed uranium (REPU) fuel can be made. To minimise the amount of Fission Products (FP) and other radioactive material that needs to be vitrified or stored, the MIMOSA scheme adds a chloride based molten salt reactor (MSR), which can use plutonium-, minor actinides- (MA) and thorium-chloride salt to create more energy (MIMOSA, 2024).

approximations of the burnup equations. Two schemes achieve the best performance. The fastest scheme is the ORIGEN program (Croff, 1983), which takes around 0.01 seconds per simulation step, with about 1% error and over 10% differences for specific nuclides (Isotalo & Aarnio, 2011). The most accurate is the CRAM (Chebyshev rational approximation method) which takes around 0.05 seconds for a single burnup step, with less than 1% error for individual nuclides (Isotalo & Aarnio, 2011).

The current thesis takes a novel approach to achieve speedup of the burnup equations, namely Reduced Order Modelling (ROM). ROM is a (data-driven) technique to transform the model to a lower-dimensional model which accurately approximates the original Full Order Model (FOM) at a lower cost. ROM techniques trace their origin to a paper by Karl Pearson in 1901, who introduced the principle component analysis (Pearson, 1901). In the second half of the twentieth century the fields of aerospace engineering, fluid dynamics and electrical engineering started developing more and better ROM methods. Castagna et al. (2020) attempted to apply ROM to nuclear fuel burnup, with moderate success; the authors used a simple Proper Orthogonal Decomposition (POD) technique and only published memory-saving results. The question thus remains: how much speedup can we achieve?

## Structure of the thesis

Due to comparative nature of the thesis, the structure is unorthodox. Although the first numbered chapter is comparable to a background, chapter two through five are more alike a scientific paper on their own. Each chapter introduces a method, gives the conceptual background, explains the method, shows results, concludes, and discusses recommendations

for further research.

The first numbered chapter, the Molten Salt Fast Reactor model, introduces a simplified version of the Molten Salt Fast Reactor (MSFR) as proposed by MIMOSA, whilst focusing on the numerical essentials: the properties of the burnup matrix. As will become abundantly clear throughout the thesis, the 40 orders of difference between the largest and smallest decay constant—called stiffness of the problem—challenges the stability of the Reduced Order Modelling (ROM) methods. To test the methods, we introduce a simplified decay chain model at the end of the chapter, of which we can tune the stiffness. The chapter provides tools to test the limits of each ROM method.

Chapters two to five all follow a similar structure, but deal with a different ROM technique.

Chapter two introduces the simplest technique, being Proper Orthogonal Decomposition (POD). Chapter three introduces a heuristically corrected POD, which aims to increase stability, a feature lacking in the simple POD. Chapter four introduces the more advanced Balanced Truncation (BT) from control theory, which can guarantee stability under the right circumstances. Chapter five presents the midway solution, Balanced Proper Orthogonal Decomposition (BPOD), which combines the stability of BT with the computational power of the POD.

While the chapters can be read individually and in any order, they do build on each other. References within each chapter help the reader find relevant sections of previous chapters.

The conclusion, reiterates the results of each method, discusses the strengths and weaknesses of the four methods, and provides a recommendation in which to conduct further research.

## Reproducibility

Most of Reduced Order Models (ROMs) were run on a ASUS zenbook UX310U laptop, with a 2.30 GHz base clockspeed processor and 8 GB memory. More demanding calculations ran on a computer cluster.

To maximize the reproducibility of the work, the code is accessible online, divided into two repositories. The first is a 600 lines custom-build python package that simulates decay chains and performs Proper Orthogonal Decomposition (POD): https://github.com/JonathanPilgram/BurnupROM-PythonPackage. The second is a collection of scripts—Python, Julia and Serpent—that perform numerical experiments, with a collection of Jupyter notebook files to generate visual aids and plots for the thesis:
https://github.com/JonathanPilgram/BurnupROM-ScriptsAndData.

You are free to use the code of each repository for any legal purpose, preferably with acknowledgement.

# Chapter 2

# Molten Salt Fast Reactor

Molten salt reactors are one of the six candidates for the fourth generation of nuclear reactors (Serp et al., 2014), and are an essential part of the European developed MIMOSA scheme. A molten salt reactor replaces the solid nuclear fuel pellets with a liquefied fuel salt mixture containing nuclear fuel. The fuel salt melts at 565°C for Lithium based salts (CNRS, 2013, p.4) and circulates around in the core. The molten salt serves two purposes: it cools the reactor by carrying away the heat, and it also acts as the fuel to keep the nuclear reaction going. Figure 2.1 shows a general overview of the Molten Salt Fast Reactor (MSFR) reactor design.

Advantages of the MSFR over current pressurised water reactors include, but are not limited to the four listed below.

First, an MSFR tweaks the design of a molten salt reactor by eliminating the moderator, resulting in a fast neutron spectrum. Fast neutrons increase the number of neutron produced per fission event. A so-called better neutron economy converts more nuclear waste products into usable fuel (Duderstadt & Hamilton, 1976, p.86–88).

Second, the increased operating temperature leads to a higher efficiency. The steam turbines can convert heat more efficiently to electricity at a higher temperature.

Third, the MSFR can use online recycling. Current pressurised water reactors must shut down during refuelling for an average of 30 days once every 12, 18, or 24 months (Foro Nuclear, 2024). During a shut down, other power plants must compensate for the loss of power, which is costly. MSFRs overcome the refuelling shutdown with online recycling: a recycling plant continuously filters waste products from the reactor core and replaces them with new fuel during operation, potentially allowing the reactor to run indefinitely.

Fourth, the MSFR includes a novel safety feature called a freeze plug; a frozen plug of reactor salt that seals the bottom of the reactor core. The nuclear reaction generates heat, which means that the plug must be actively cooled; otherwise, the plug melts, draining the reactor core into safety tanks, which would happen in case of a total power failure (Tiberga et al., 2019).

In summary, the MSFR offers potential improvements over current nuclear reactors, including better neutron economy, higher thermal efficiency, continuous recycling, and new fail-safes such as the freeze plug.

On the other side, there are still technical problems to be solved, such as the ever-changing liquid fuel salt composition corrodes common steels used for the reactor vessel (Guo et al., 2018). Besides, the ever changing salt composition affects neutron physics of the reactor. Keeping track of the fuel salt composition is necessary to ensure safe and efficient operation of the reactor.

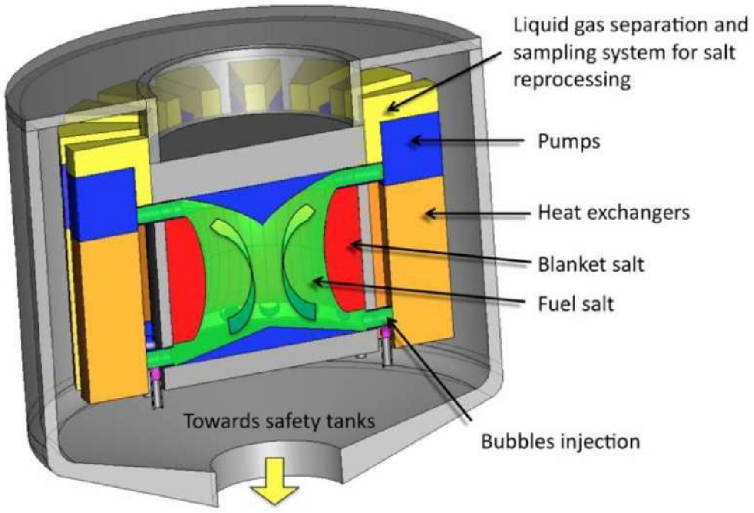



**Figure 2.1: Conceptual overview of the MSFR**
The cylindrical core of liquid fuel salts is green, while the solid blanket salts used for breeding are in red. Blue areas are empty space, possibly for pumps to force flow through the orange heat exchangers. Bubble injectors filter out Fission Products (FP) and neutron sponges, such as xenon gas, by forcing the FP and xenon to the separation system on top, indicated in yellow. A novel safety measure of the MSFR is visible at the bottom in purple; in case of an overheating accident, the core automatically drains through the so-called freeze plugs to safety tanks (CNRS, 2013, p.2–5).

## 2.1 Fuel salt composition

The initial fuel salt mixture put into a reactor is one of two types: fluoride-based and chloride-based salts (Serp et al., 2014), referring to the anions of the salt mixture. To keep the scope of this research limited, we focus on chloride-based fuel salts. The corresponding cation is natrium. A percentage of the cations are replaced with fissile, fissionable and fertile materials.

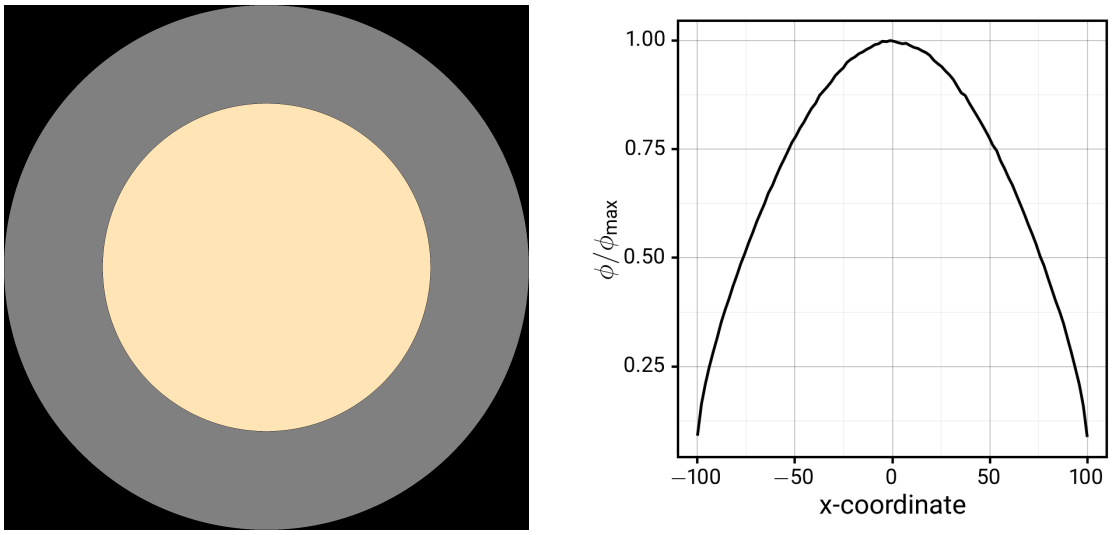
**Table 2.1: Plutonium fractions of spent UOX fuel**
The percentages are the molar fractions of each isotope. Last column reports the total percentage of fissile plutonium isotopes in UOX fuel. Adapted from table 4.1 in the work of Mesthiviers (2022, p.87).

| Pu-238 | Pu-239 | Pu-240 | Pu-241 | Pu-242 | fissile |
|---|---|---|---|---|---|
| 3.10% | 52.14% | 25.20% | 11.80% | 7.76% | 63.94% |

The neutron economy of the MSFR make it an good tool for burning through spent fuel, such as UOX fuel. The plutonium composition of UOX fuel is shown in Table 2.1. Burning spend fuel has two advantages: we get more energy out of the same resources and we can burn the problematic radioactive transuranics.

The MIMOSA scheme, as seen in Figure 1.1, uses the MSFR as a crucial link in the long-term processing of reactor fuel.

When the MSFR starts operation the neutrons start interacting with the initial fuel mixture. After minutes there are more than 1000 different nuclides present in the core. As the nuclide mixture determines the chemical and physical properties of the reactor core, keeping track of them is important, but computationally expensive. The current chapter details the accurate, but expensive method of a full order simulation. The next chapters present ways to cut down on the computational costs using reduced order models.




**(a)** Radial geometry

**(b)** Neutron flux

**Figure 2.2: The geometry and neutron flux of the molten salt fast reactor model**
Figure **(a)** depicts the XY-plane of the reactor geometry. The 60 cm iron reflector is grey, the 1 meter radius cylindrical core with chloride based fuel salt is yellow/beige, and black indicates the outside. Figure **(b)** depicts the normalised neutron flux of the fuel for the x-coordinate, which is expected from the radially symmetric simple geometry.

Section 2.2 presents a simplified model of the MSFR simulated using SERPENT. The simplifications allow us to study the burnup properties of the MSFR.

Section 2.3 details the burnup calculations and properties of the burnup matrix, which holds all the numerical information relevant to the burnup of a reactor. As the full burnup of the MSFR is both computationally expensive and challenging due to the large number of nuclides with timescales varying over 40 orders of we need a simpler model for testing purposes.

Section 2.4 provides that further simplified model: the decay chain model. Although decay chains exist in burnup matrices, we mainly use the decay chain model as a tunable version of our reactor to benchmark Reduced Order Modelling (ROM) techniques.

## 2.2 SERPENT reactor model

We use SERPENT (Leppänen et al., 2015), a continuous-energy Monte Carlo and photon transport code, to simulate the Molten Salt Fast Reactor (MSFR) model and generate a burnup matrix using the JEFF-3.3 nuclear dataset (Nuclear Energy Agency, 2017).

Based on the work of Bouma (2024, p.19–20), we simulate a cylindrical reactor core with a radius of 100 cm and a height of 200 cm surrounded by a 60 cm thick iron reflector, as shown in Figure 2.2a. The fuel mixture consists of $NaCl - PuCl_3 - ThCl_4$, with molar fractions of 46.5% NaCl, 18.5% $PuCl_3$, and 35% $ThCl_4$. The isotope mixture is their naturally occurring fractions, except for plutonium; we used the fractions found in spent nuclear fuel from pressurised water reactors, as shown in Table 2.1. Figure 2.2b displays the resulting radial flux. We used version 2.1.29 of SERPENT for all reactor physics simulations (Leppänen et al., 2015). The full input script is available in the repository https://github.com/JonathanPilgram/BurnupROM-ScriptsAndData.

## 2.3 Burnup calculations

Before delving into the mechanisms of the burnup calculations, we need to clarify some assumptions. We focus on simulating the burnup of the salt mixture inside the Molten Salt Fast Reactor (MSFR) over timescales from days to years, so we assume a mixed core; we disregard spatial dependence in the nuclide composition, reducing the reactor to a single point in space. The most questionable assumption is that we are dealing with linear time-invariant models, meaning that the reactor physics can be approximated by a single, time-independent burnup matrix, $A$. Although this assumption is not entirely accurate, as $A$ both affects and depends on the nuclide composition, it provides a reasonable approximation. Besides, if the neutron flux increases, the nuclear reaction occur more frequently, but the decay rates remain constant, modifying the burnup matrix.

One of the main challenges of burnup calculations for nuclear reactors is the problem's stiffness; see Section 2.3.2. The timescales of nuclides in a burnup calculation span over 40 orders. We could try to introduce cut-offs to ignore values far outside the timescale of interest, but the coupling of nuclides in a nuclear reactor prevents simple implementations; a short-lived, seemingly insignificant nuclide can decay into a significant long-lived nuclide. Ignoring nuclides with short timescales without corrections is innefective. The extreme stiffness of the burnup problem presents a major challenge when applying reduction techniques, as the effects of stiffness on Reduced Order Model (ROM) are not well understood.

### 2.3.1 The burnup matrix

The burnup matrix $A(\phi(t), N(t)) \approx A$ is a square matrix encapsulating all the time scales of the different processes occurring in a nuclear reactor. Each element of the matrix represents a rate of change of the nuclide vector $N(t)$; positive values an increase of a certain nuclide, while negative values a decrease. Mathematically, this is expressed as:

$$\frac{\mathrm{d}\mathbf{N}(t)}{\mathrm{d}t} = A\mathbf{N}(t). \tag{2.1}$$

Equation 2.1 eexpressesents a system of coupled linear ordinary differential equations. Numerically, we solve the burnup equation for the full system with Radau methods, an L-stable fully implicit method that efficiently handles stiff problems—refer to Hairer and Wanner (1996, Sec. IV.8) for more details. The SERPENT simulation program, used to generate burnup matrices for nuclear reactors, orders the nuclide vector in increasing ZAI (Z = proton number, A = mass number, I = isomeric state) notation. ZAI notation numbers nuclides as

$$10000 \cdot \mathrm{Z} + 10 \cdot \mathrm{A} + \mathrm{I},$$

where Z is the proton number, A the mass number, and I the isomeric state (Serpent Wiki, 2020); the ZAI notation increases with nuclide mass. In this notation, interactions that produce nuclides with a higher proton number appear in the upper half of the burnup matrix, while interactions producing nuclides with a lower proton number appear in the lower half.

Fuel in a reactor core undergoes two main processes: nuclear decay and nuclear reactions. Specific to MSFR is the optional implementation of online recycling, which SERPENT can model through an extension made by Aufiero et al. (2013). Although the current work does not include recycling, Appendix A provides information about the recycling matrix.

Decay and reactions have a different effect upon the burnup matrix. Decay rates are always the same, independent of neutron flux or fuel composition. The nuclear reactions are dependent on both neutron flux and fuel composition. Neutron flux multiplies the reaction part of the burnup matrix with a scalar, whilst nuclide composition changes the elements of the matrix relative to each other.

Figure 2.3 shows the sparsity pattern of the burnup matrix when considering decay and Figure 2.4 the sparsity pattern of solely nuclear reactions. Although, only the decay rates are time-independent, for the proceeding work we also assume the nuclear reactions to be time-independent.

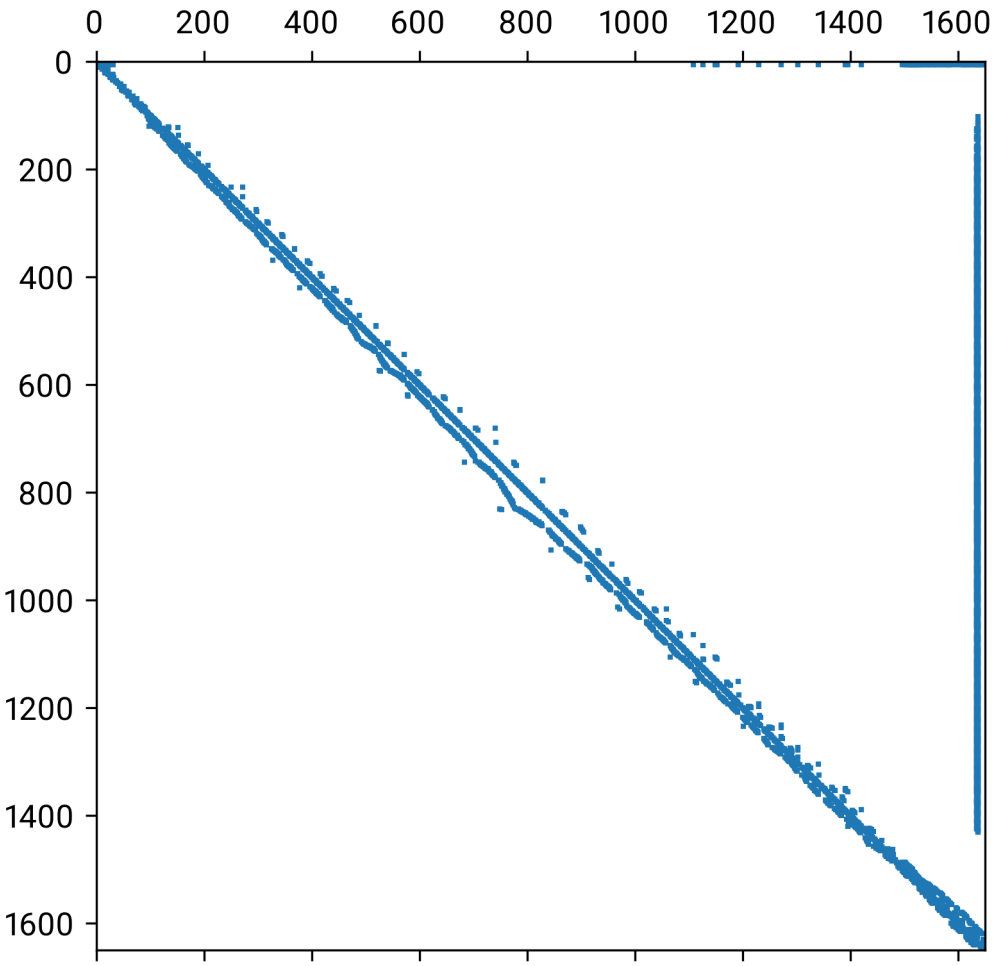



**Figure 2.3: Sparsity pattern burnup matrix with solely decay**
The matrix is ordered according to ZAI numbering, shown on the axis. Negative values on the diagonal indicate unstable nuclides. Beta decay lines are visible near the diagonal. These are positive values, which indicate creation of daughter nuclides. The rarer alpha decay is visible as dots in the helium row on top of the matrix. Less visible are the positive values of the daughter nuclides above the diagonal. Alpha decay happens for light and heavy nuclides. Some of the heaviest nuclides undergo spontaneous fission, which creates the populated columns on the right. The sum of the diagonal and off-diagonal elements is positive, corresponding to the extra nuclei created during alpha decay, spontaneous fission and some rare decay modes emitting particles.

### 2.3.2 Numerical imprecision and condition numbers

Computers use bits to represent numbers, using floating point formats for real numbers. The number of bits used to store the floating point limits the precision of the representation.

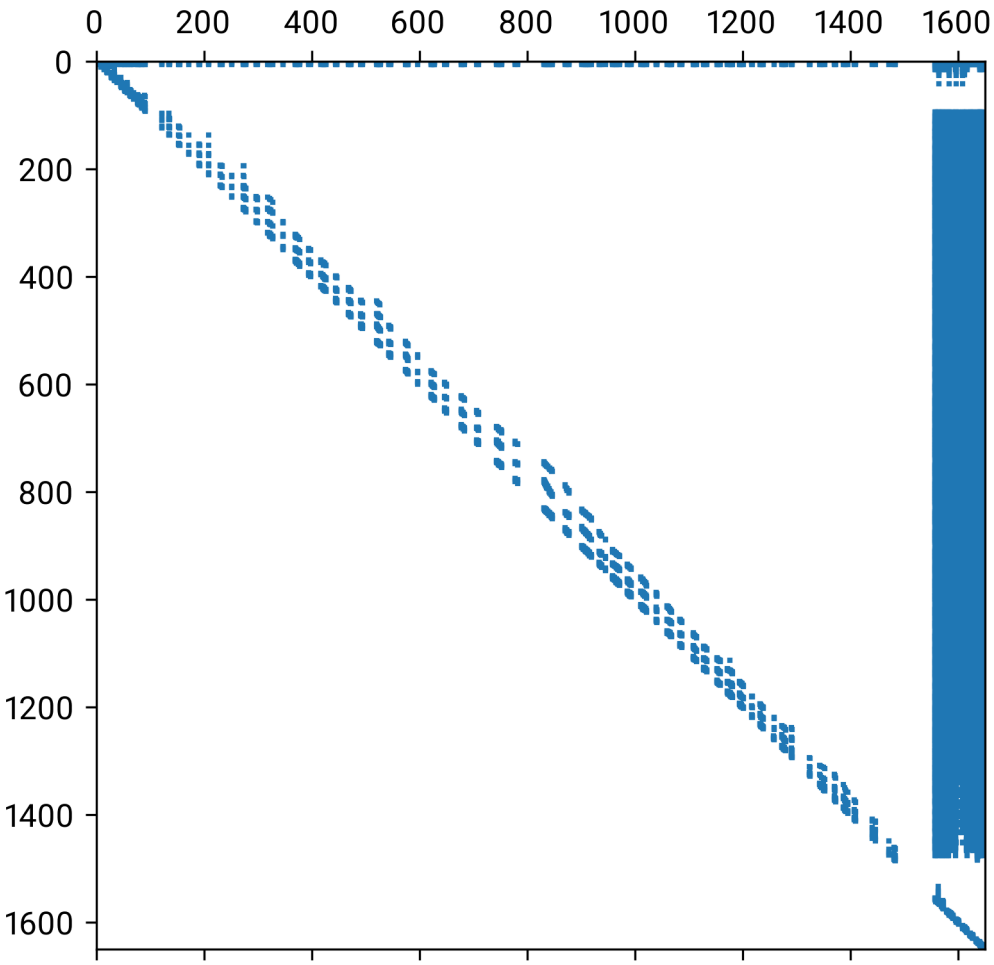



**Figure 2.4: Sparsity pattern burnup matrix with solely reactions**
The matrix is ordered according to ZAI numbering, shown on the axis. The values on the diagonal are negative, indicating reaction rates of nuclides. The reaction products are positives on the off-diagonal. We discern three regions. First, the upper left corner (ZAI<100) where capture and all transfer reactions occur, thus the matrix is densely populated. Second, the middle area (100<ZAI<1500), where capture, limited transfer and $(n, 2n)$ reactions occur, creating the sparse pattern. Third, the right columns (1500<ZAI), where fission can also occur. The columns represent the Fission Products (FP). The off-diagonal elements of the matrix are larger than the negative values on the diagonal, because fission and the $(n, \alpha)$ transfer reaction creates extra nuclides. The columns represent the FP.

Python, for example, uses 64 bits floating point numbers by default. The IEEE Standard for Floating-Point Arithmetic defines the binary64 format as a floating point number with 15.95 decimal significant digits.

So, why care about numerical precision up until almost sixteen digits of precision? Because operations like multiplication and taking an inverse propagate numerical errors; the condition number is the maximum amplification factor of the error. The condition number of a matrix, subject to the Euclidean norm, is defined as the ratio between the largest and smallest singular value

$$\kappa(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)}. \tag{2.2}$$

In a square matrix, the singular values correspond to the eigenvalues.

If the base ten logarithm of the condition number approaches the significant digits of the floating point format used to store the number, then the error amplification can be significant—the matrix is ill-conditioned (Cheney & Kincaid, 2007, p. 321). The values in the SERPENT burnup matrix range from $\lambda_{\min} = 10^{-20}$ s$^{-1}$ to $\lambda_{\max} = 10^{20}$ s$^{-1}$, representing a stiffness of $10^{40}$. We estimate the condition number to be $\kappa \approx 10^{40}$. We conclude that amplification of numerical errors poses a problem when manipulating the burnup matrix.

A high condition number also indicates a model that is very sensitive to small changes in input, which can mean instability. Besides, iterative solving methods need more iterations when solving stiff/ill-conditioned problems, which increase computational demands.

## 2.4 Decay chain test model

The Molten Salt Fast Reactor (MSFR) burnup problem stiffness of $10^{40}$ creates an ill-conditioned burnup matrix. If we test different methods for creating a more efficient Reduced Order Model (ROM), then we need a way to circumvent the ill-conditioned burnup matrix if the method struggles with precision. For this we introduce the decay chain test model, which can be used as a stepping stone to test and compare different techniques of making a ROM.

A decay chain consists of a series of nuclides that decay sequentially. Figure 2.5 shows an example of a real decay chain. The decay chain model can generate arbitrary decay chains, varying in length of nuclides and the distribution of decay constants. The primary purpose is to create a simplified, yet physically inspired, model that we can adjust. We use the model throughout the thesis as a first stepping stone for testing the ROM methods.

The Bateman equations, governing radioactive decay, describe the rate of change of the number of atoms for each nuclide $N_i$

$$\frac{\mathrm{d}N_i}{\mathrm{d}t} = \sum_j N_j \lambda_j b_{j\to i} - N_i \lambda_i. \tag{2.3}$$

The first term is a source term, summing over all nuclide amounts $\sum_j N_j$, factored in the decay rate of each nuclide $\lambda_j$ and the branching ratio from nuclide $j$ to $i$: $b_{j\to i}$. The second term represents decay of nuclide $i$, given by its density and decay rate. A bit of rewriting reveals many similarities with Equation 2.1; in fact, they state the same.

The decay rate constant $\lambda$ expresses different rates of decay. The decay rate relates to the average lifetime of nuclides as $\lambda = 1/\tau$, where $\tau$ indicates the time after which the number of parent atoms reduces to $1/e$. The simplest decay chain model assumes the same decay constant for each nuclide. In reality, not all nuclides are equal; for example, Bismut-209 has a half-life ($6.3 \cdot 10^{26}$ s) longer than the current known age of the universe (De Marcillac et al.,

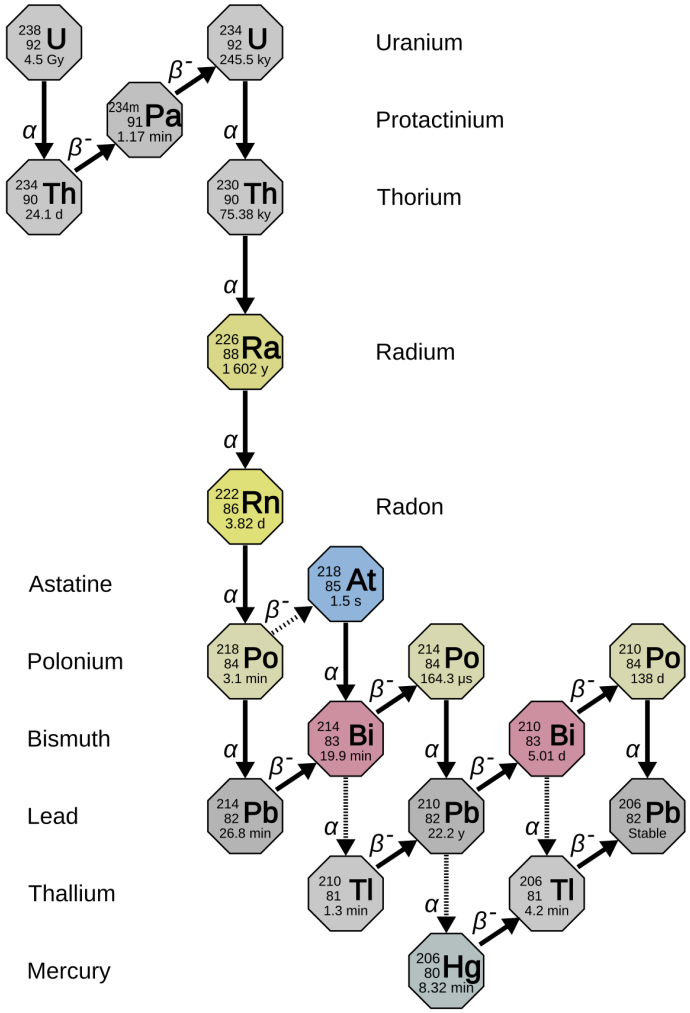



**Figure 2.5: Example decay chain of Uranium-238**
The decay chain terminates when it reaches the stable lead nuclide after 14 decay steps. Note that there are alternative decay paths; some nuclides have more than one possible decay products. Data provided by the National Nuclear Data Center (2023) and figure by Tosoka (2014).

2003), while Hydrogen-5 has a half-life less than $100 \cdot 10^{-24}$ s (Kondev et al., 2021, p.20). The ratio between the smallest and largest decay constant in a chain—referred to as the stiffness of the problem—can theoretically be $10^{50}$ . To simulate interactions between nuclides with decay constants spanning orders of magnitude, we use a logarithmic distribution

$$\lambda_n(\lambda) = \begin{cases} 10^{\log(\lambda_{\min}) + \log(\lambda_{\max}/\lambda_{\min})} & \text{for } \lambda_{\min} \leq \lambda \leq \lambda_{\max} \\ 0 & \text{otherwise.} \end{cases} \tag{2.4}$$

Here $\lambda_n$ denotes the randomly sampled decay constant for nuclide $n$, while $\lambda_{\min}$ and $\lambda_{\max}$ denote the sample range. By creating opposite signed pairs in the decay matrix, we can form a chain. The decay chain model terminates with a stable nuclide. The result is a burnup matrix, as shown in Figure 2.6.

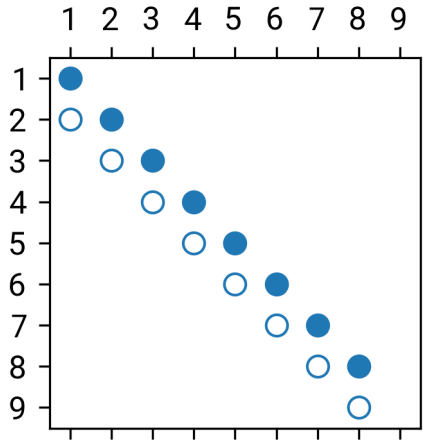



**Figure 2.6: Sparsity pattern decay chain model of 9 nuclides**
Filled circles represent negative values, whilst empty circles represent positive values. Each pair (in columns) has the same magnitude, but opposite signs. The last nuclide has no negative value associated with it, indicating a stable nuclide.

The Bateman equations (Equation 2.3) for a decay chain are simpler than a full burnup system, so we solve them with the implicit Euler method: a first-order backward difference solver. Although the Radau method offers higher precision, the implicit Euler method is quicker. A full description of the method is available in Butcher (2003, p. 57).

Figure 2.7 gives two examples of the decay chain model: one with constant decay constants and one with decay constants taken according to Equation 2.4.

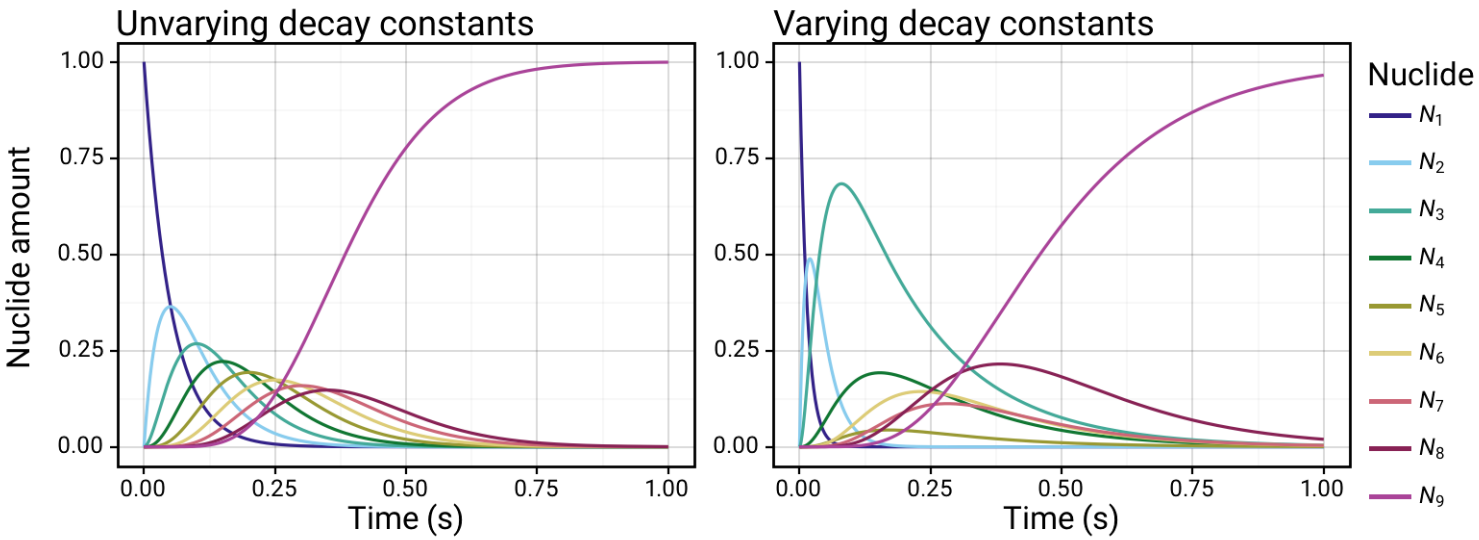



**Figure 2.7: Two simulation examples of decay chain models with 9 nuclides**
The initial condition is $N_1(t_0) = 1$, whilst the rest is zero. The decay chains terminate with a stable nuclide. Both models are simulated using the Radau method with 1000 integration steps over 1 second. The left plot shows a decay chain with decay constant $\lambda = 20\ \mathrm{s}^{-1}$ for every nuclide. The right plot depicts the decay constants sampled from the distribution described in Equation 2.4, with $\lambda_{\min} = 5\ \mathrm{s}^{-1}$ and $\lambda_{\max} = 75\ \mathrm{s}^{-1}$.

The Python class used for the decay chain model can be found in Appendix B.
The decay chain model presented is not meant to simulate real nuclides, although it

can do so by setting the decay constants to their physical values. The model is primarily used for testing purposes, such as comparing different ROM methods and finding the limit of the methods. In theory, the model can be modified to include more complex processes, such as recycling. Throughout the next chapters, we frequently use the decay chain model. Appendix C summarises the default settings of the decay chain simulations.

# Chapter 3

# Proper Orthogonal Decomposition

Burnup calculations of a full reactor system are expensive; the calculation tracks 1650 nuclides with decay constants spanning 40 orders. For example, solving the burnup equations of the Molten Salt Fast Reactor (MSFR) model for 500 time steps using an efficient Radau solver takes over an hour, with a processor clock speed of 2.30 GHz. This is where Reduced Order Model (ROM) can offer a solution. ROM is essentially a collection of techniques that perform a basis transformation to a lower-dimensional (lower-order) basis, while retaining as much information as possible. Previous work in ROM has shown promising results in the field of nuclear reactor physics. Castagna et al. (2020) shows the possibility of saving memory by a factor 500 with intrusive Proper Orthogonal Decomposition (POD) techniques and Bouma (2024, p.11–28) shows similar time savings with the technique.

Questions remain: for instance, why did Castagna et al. (2020) not report any time savings? Bouma (2024, p.11–28) observes that the intrusive POD method can generate unstable Reduced Order Models (ROMs) and provides us with a clue; sometimes the eigenvalues of the ROM lie in a unstable region, but a detailed explanation of the observed instabilities is missing. The current chapter goes one step further by providing the underlying mechanics of the instabilities and mitigating them.

The first section of this chapter explains the POD in-depth. Essential concepts are introduced, followed by mathematical description of how to use POD to make ROMs, which leads to a summary of implementation steps. The second section presents results of POD ROMs of decay chain models for varying stiffness, number of reduced modes, and number of snapshots. A novel finding is the number of reduction failures when of the standard POD method; the failure mechanism is further investigated. The third section concludes the chapter: the POD is inadequate for reliably reducing burnup problems—recommendations for improvements are given.

## 3.1 Introduction to Proper Orthogonal Decomposition

Data produced by high dimensional coupled models have correlation between variables. Proper Orthogonal Decomposition (POD)-based Reduced Order Models (ROMs) leverage the correlation to find a more efficient ordered basis; the POD realigns and orders linear combinations of nuclides from most to least variation.

Although one does not expect POD to be very useful in two dimensions, it is a good example to illustrate the technique. Two dimensional data form an ellipse when plotted, as shown in Figure 3.1. The ratio between the major and minor axis corresponds to the degree of correlation. POD realigns basis in such a way, that the first basis vector is the direction of greatest variation; the major axis of the ellipse. We require each additional basis vector

of the POD to be orthogonal to all previous basis vectors, whilst maximising the variation. For the ellipse the second basis vector of the POD is the minor axis of the ellipse. The size of the first basis vector relative to the second increases if the ratio between the major and minor axis increases, i.e. more correlation in the data.

The ellipse example clarifies two important aspects of the POD:

- POD creates an ordered orthogonal basis, which is ordered from most to least variation in the data.
- The effectiveness of POD-based ROMs increases with increased correlation in the data.

To give an extreme example for the second aspect. If data is perfectly correlated, we can reduce the model to a single dimension without any loss of information—the data is described by a linear relation.

For two-dimensional data, the method is not of much use as we can reduce the model by one dimension. Besides, the accuracy loss of reducing this one dimension is probably significant. Looking at the ellipse we can see that the quality of the model depends on the relative length of the major to the minor axis. For a high-dimensional model POD transformation is more useful. POD produces a basis ordered by variation, such that each basis vector contains less information than the previous one; we expect that the information contained in last few basis vectors of the POD is insignificant compared to the first. POD essentially realigns and orders the variables from most to the least variation.

Physical systems exhibit regularities/correlation. For example, all fission reactions exhibit a recognizable camel curve for Fission Products (FP). Decay chains are correlated, as particles are conserved; decrease in one nuclide correlates with increase in another. Physicists catch the natural correlations in their formulas. The formulas for burnup are too unwieldy for solving large systems, which is where Reduced Order Modelling (ROM) comes in.

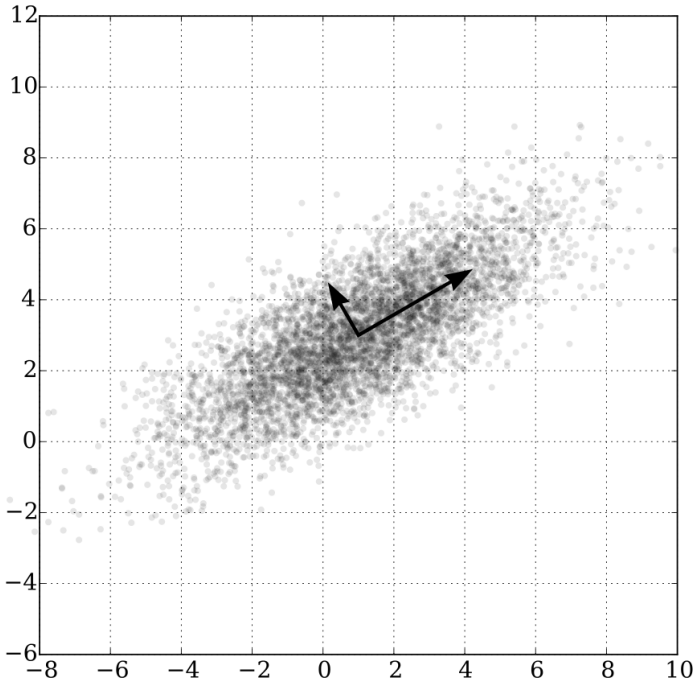



**Figure 3.1: Sample points of a covariate Gaussian distribution**
The arrows represent the directions of the greatest variance, which would be the basis vectors after a POD transformation. The arrows coincide with the eigenvectors of the covariance matrix of the distribution. Figure adapted from Nicoguaro (2016).

First, we generate snapshots of the system, which are solutions of the equations at certain time steps. POD converts the snapshots to an orthonormal basis of linear subspaces in a least squares optimal sense (Rathinam & Petzold, 2003, p1893), meaning that the basis vectors are ordered according to significance. The first basis vector predicts on average the greatest variance in the data, the next basis vector the next greatest variance and so on (Cordier & Bergmann, 2008, p.23–24). By truncating this basis to certain order, we obtain a transformation that can transform both the governing equations and nuclide vector of a Full Order Model (FOM) to a reduced space of quasinuclides: a quasinuclide represents a linear combination of real nuclides. Simulating the Reduced Order Model (ROM) with quasinuclides is cheaper, but is an approximation of the FOM; using more quasinuclides is computationally more expensive but approximate the FOM better.

POD allows both intrusive and non-intrusive approaches. Intrusive means that we transform both the nuclides and mathematical model to the reduced space, which allows the ROMs to use information about the governing equations. Although we use intrusive ROM in the thesis, there also exist non-intrusive ROMs. The POD can also be used non-intrusively, which means that the technique tries to rebuild the data of the snapshots in a reduced space without any description of the underlying mechanics. Scheepmaker (2024) used non-intrusive POD for making a predictive ROM of a high-temperature gas-cooled reactor and Alsayyari et al. (2021) used an adaptive non-intrusive POD method for nonlinear time-independent parameterized partial differential equations, such as neutron transport. Non-intrusive approaches excel when the underlying physics is hard to describe or simulation code is inaccessible.

In the fuel of a nuclear reactor we can approximate all the interactions with a system of coupled ordinary differential equations with a single burnup matrix; the complexity comes from the number of interactions—the burnup matrix includes 1650 different nuclides. Section 2.3 details the burnup matrix. The method of choice for burnup calculations is intrusive modelling, which can leverage the governing equations to reach a higher accuracy for the same amount of input data (Rathinam & Petzold, 2003).

The next subsections introduce five essential concepts: the snapshot matrix, determining the POD, Singular Value Decomposition (SVD), singular values and burnup model projection. The first four are part of the offline (preparatory) phase, which needs to be done once. The last is the online phase, which can be ran for as many cases as required.

### 3.1.1 Snapshot matrix

The snapshot matrix is essential for the data-driven approach of constructing the POD; data-driven means that we capture data from the FOM to perform the reduction. The data is collected in a snapshot matrix. Each snapshot (column) of the matrix represents the output of the FOM at a different time step (Cordier & Bergmann, 2008, p.29–36). The output of burnup calculations is the nuclide composition, which is represented by a vector. The nuclide vectors $\mathbf{N}(t)$ for different time steps are concatenated, forming the snapshot matrix $S$,

$$S = [\mathbf{N}(t_0), \mathbf{N}(t_1), \ldots, \mathbf{N}(t_m)]. \tag{3.1}$$

The snapshot matrix has dimensions $m_{\text{time steps}} \times n_{\text{nuclides}}$. Generally, generating more snapshots with the FOM increases the accuracy of the ROM, but requires more expensive simulations of the FOM. The so-called offline phase of the ROM takes longer.

### 3.1.2 Determining the Proper Orthogonal Decomposition

From the $m \times n$ snapshot matrix $S$, we can determine the POD. First calculate the covariance matrix

$$C = \frac{1}{m-1} S^T S. \tag{3.2}$$

Then the POD modes are the ordered eigenvectors of the covariance matrix

$$C = \Phi\Lambda\Phi^{-1} = \Phi\Lambda\Phi^{T}. \tag{3.3}$$

$C$ is symmetric, therefore $\Phi^{-1} = \Phi^{T}$. Each eigenvector is a mode of the POD and the corresponding of the eigenvalue give the significance of the POD mode (Weiss, 2019, p.4–6). Taking the eigenvalue decomposition is computationally an ill posed problem; numerical errors get larger. The stiffness of the burnup problem exceeds the numerical precision of the commonly used Float64 data type, hence numerical errors get significant.

### 3.1.3 The Singular Value Decomposition

Weiss (2019, p.16–17) shows that the eigenvalue decomposition of the covariance matrix is equivalent to the SVD of the $m \times n$ snapshot matrix.

$$S = U\Sigma V^{T}, \tag{3.4}$$

where $U$ is an $m \times m$ orthogonal matrix containing the temporal information and $V$ is a $n \times n$ containing spatial information, which we disregard. (For a non-intrusive approach, this information is used.) $\Sigma$ is a $m \times n$ diagonal matrix containing the ordered singular values $\sigma_i$, which relate to the eigenvalues $\lambda_i$ as

$$\lambda_i = \frac{\sigma_i^2}{m-1}. \tag{3.5}$$

Numerically, the SVD is more stable than the eigenvalue decomposition at a higher computational cost (Golub & Van Loan, 2013, Sec. 2.4). As the SVD is calculated during the offline phase, which we do once, the extra computational costs are acceptable.

### 3.1.4 Singular Values

Singular values can be used to calculate the expected precision of ROM, as they are ordered $\sigma_1 > \sigma_2 > \cdots > \sigma_n$. The relative information content of a lower order model is defined by (Cordier & Bergmann, 2008, p.11–16)

$$RIC(M) = \frac{\sum_{i=1}^{M} \sigma_i}{\sum_{i=1}^{N} \sigma_i}. \tag{3.6}$$

The numerator is a sum of all singular values $\sigma_i$ of the model reduced to order $M$. The denominator is the sum of all singular values of the full model of order $N$. The equation shows that the accuracy of the reduced order model is determined by the sum of the singular values included in ROM versus the total sum of the singular values. Whilst the equation can be useful, more often than not the order reduction is determined heuristically; one can plot the singular values and then choose the reduction order based on the speed of the singular value decrease. Figure 3.2 shows the singular values of an eight nuclide decay chain, which shows that a reduction to six orders would be accurate.

Either way, reducing the model increases simulation speed at the cost of accuracy; the expected results of the trade-off can be predicted by looking at the singular values.

Performing the dimensionality reduction involves truncating the matrix to column $r$. The result is a matrix $\Phi_R$ of $n \times r$ size—$n$ being the number of dimensions of the full order model. The matrix $\Phi_R$ is left invertible, meaning $\Phi_R^T\Phi_R = I$, but $\Phi_R\Phi_R^T \neq I$, with $I$ the identity matrix.

Truncating the basis concludes the offline phase of the POD process.

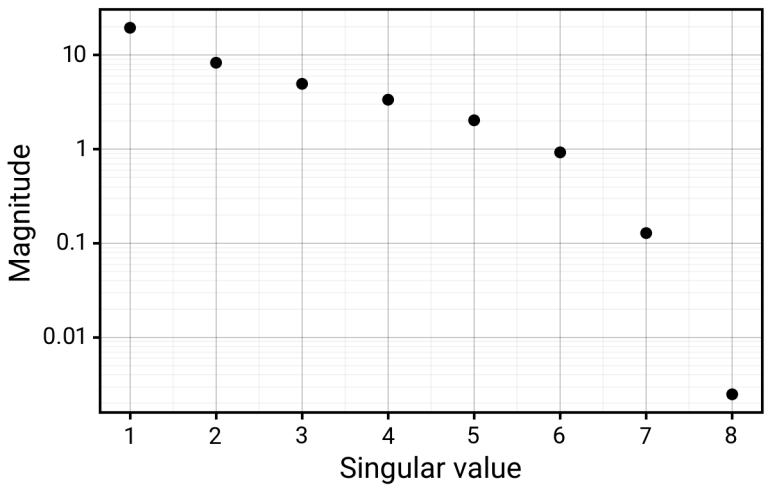



**Figure 3.2: The singular values of Figure 3.4 on a logarithmic scale**
The Full Order Model is an eight nuclide decay chain, so eight singular values. The plot shows that the singular values decay fast after the sixt singular value, which indicates that the relative error of the ROM is low when including six or more orders.

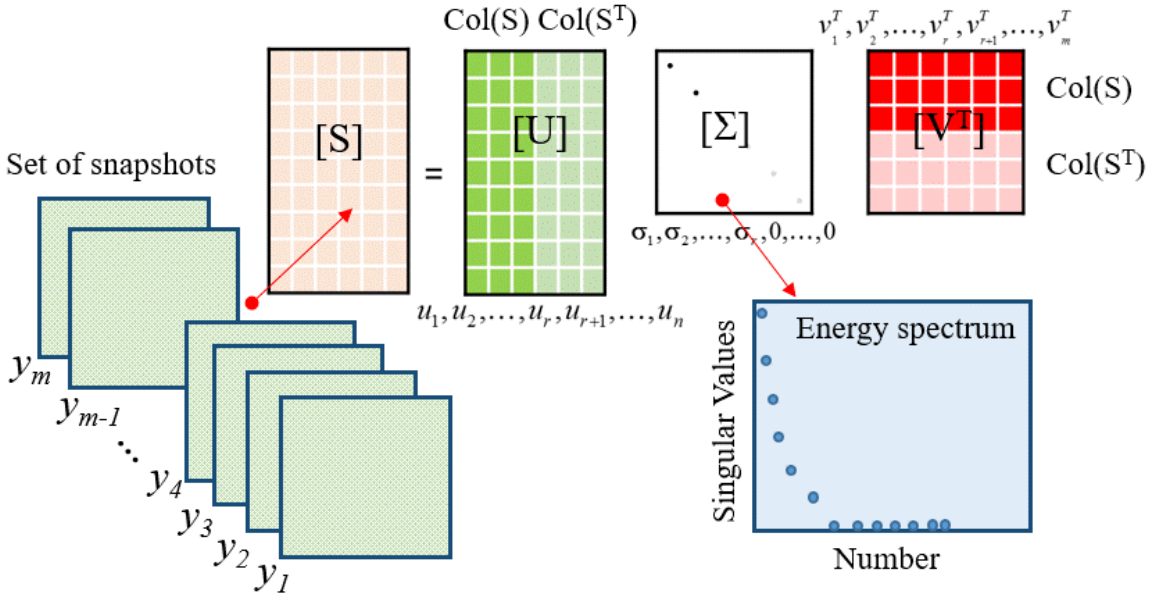



**Figure 3.3: Summary of the offline phase of the POD**
First a snapshot matrix is generated (Section 3.1.1), then the POD modes (Section 3.1.2) are determined using the SVD (Section 3.1.3). The singular values give a way to predict the relative information content of each mode (Section 3.1.4), which can be used to choose the order of the ROM. Figure from the work of Paul et al. (2018).

### 3.1.5 Projecting the burnup model

The POD essentially produces a basis transformation, so running the ROM is a matter of applying the basis transformation. We use a Petrov-Galerkin projection (Redmann & Duff, 2022). Given the burnup equation

$$\frac{\mathrm{d}\mathbf{N}(t)}{\mathrm{d}t} = A\mathbf{N}(t), \tag{3.7}$$

where $\mathbf{N}$ is the nuclide vector of length $n$. $A$ is a square matrix of size $n \times n$ containing constants.

To run the model in the reduced space, we project on the truncated POD modes $\Phi_R$

$$\mathbf{N} = \Phi_R \mathbf{N}_R \leftrightarrow \mathbf{N}_R = \Phi_R^T \mathbf{N}. \tag{3.8}$$

The resulting system of equations is

$$\begin{aligned} &\frac{\mathrm{d}}{\mathrm{d}t}\Phi_R \mathbf{N}_R = A\Phi_R \mathbf{N}_R \\ &\Rightarrow \frac{\mathrm{d}\mathbf{N}_R}{\mathrm{d}t} = \Phi_R^T A \Phi_R \mathbf{N}_R = A_R \mathbf{N}_R. \end{aligned} \tag{3.9}$$

$\mathbf{N}_R$ is the state vector in the reduced order space, determined by the initial condition of the state vector

$$\mathbf{N}_R(t_0) = \Phi_R^T \mathbf{N}(t_0). \tag{3.10}$$

The reduced order solution is transformed back to the full order system by applying the ROM basis to the state vector

$$N = \Phi_R N_R. \tag{3.11}$$

### 3.1.6 Summary

The first three steps are the offline phase, summarised in Figure 3.3. The last three steps are the online phase, where speed-up is acquired.

1. Acquire the snapshot matrix $S$ with results of the Full Order Model (FOM).
2. Calculate the Singular Value Decomposition (SVD) of S SVD$(S) = U\Sigma V^T$.
3. Truncate $L$ to acquire the Reduced Order Model (ROM) basis $\Phi_r$.
4. Project the FOM to the reduced space using Equation 3.8 and 3.10.
5. Simulate the ROM using Equation 3.9.
6. Transform back to the FOM using Equation 3.11.

Figure 3.4 shows an example of eight successive orders of Proper Orthogonal Decomposition (POD) based Reduced Order Models (ROMs) of a decay chain.

## 3.2 Reduction performance on decay chain models

Section 3.1 explains the methodology of the Proper Orthogonal Decomposition (POD) for Reduced Order Model (ROM); now we turn to the performance of the technique with the decay chain test model—introduced in Section 2.4. An excerpt of the script for the performance measurement is found in Appendix D. First, we study the effect of reduction orders and snapshots on the accuracy of the Reduced Order Models (ROMs). However, sometimes the reduced model 'fails'—the normalised relative error is larger than 95%. The failure mechanisms are investigated.

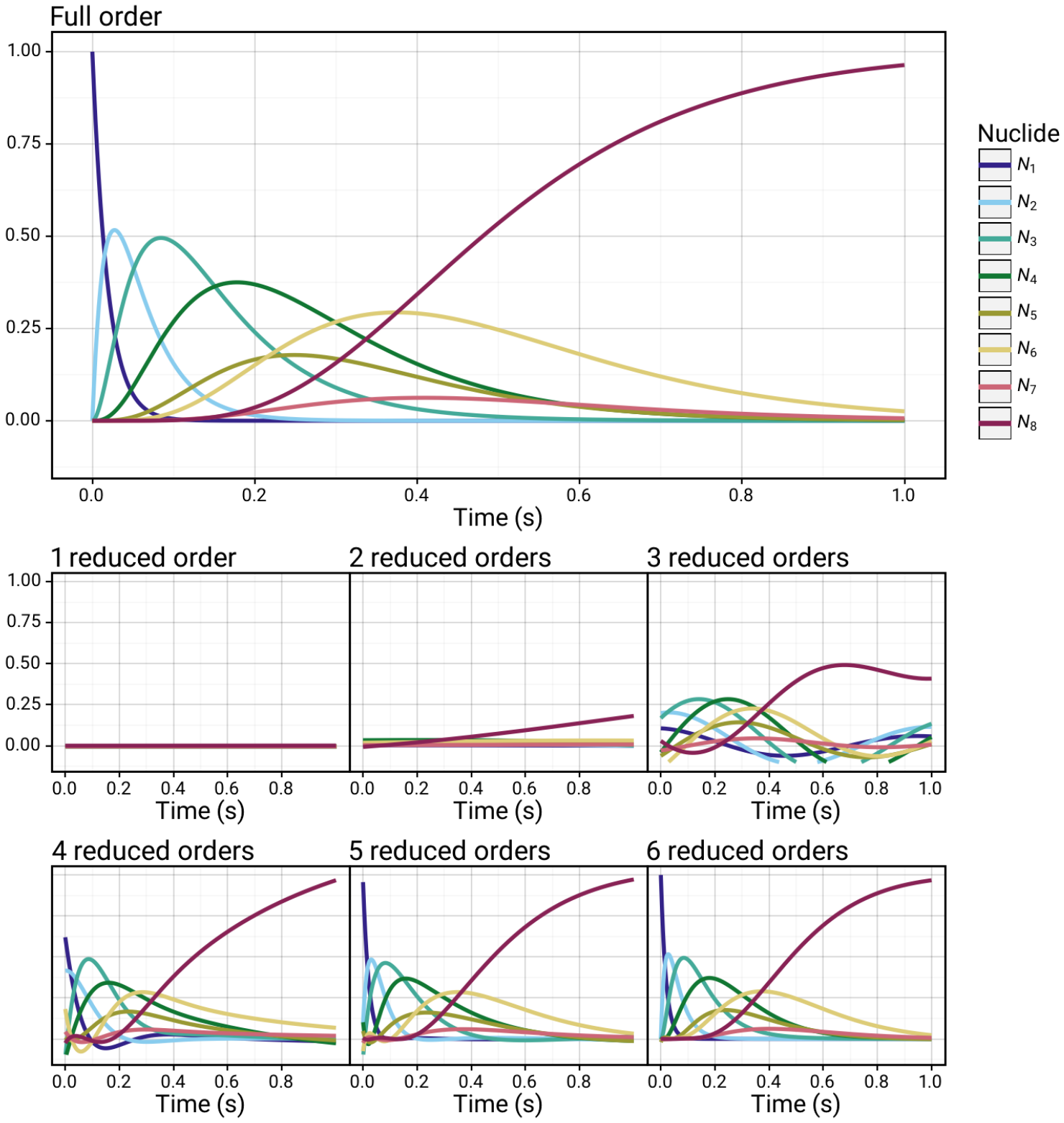



**Figure 3.4: Full order vs. reduced orders of a decay chain**
The first plot shows a decay chain model with 8 nuclides ($N_1(t_0) = 1$, the rest zero), simulated for 1 second. The decay constants are: $\lambda_1 = 52.7\ \mathrm{s}^{-1}, \lambda_2 = 24.2\ \mathrm{s}^{-1}, \lambda_3 = 10.6\ \mathrm{s}^{-1}, \lambda_4 = 8.3\ \mathrm{s}^{-1}, \lambda_5 = 15.2\ \mathrm{s}^{-1}, \lambda_6 = 6.9\ \mathrm{s}^{-1}, \lambda_7 = 32.0\ \mathrm{s}^{-1}$ and $N_8$ is stable. Figure 3.2 shows the singular values of the model. The decay chain has been reduced using 1000 snapshots. Each successive order approximates the full order model better, which is verified by the normalised relative error: $\epsilon_{r=1} = 1.00, \epsilon_{r=2} = 0.86, \epsilon_{r=3} = 0.51, \epsilon_{r=4} = 0.12, \epsilon_{r=5} = 4.6 \cdot 10^{-2}, \epsilon_{r=6} = 7.3 \cdot 10^{-3}, \epsilon_{r=7} = 2.1 \cdot 10^{-4}, \epsilon_{r=8} = 1.6 \cdot 10^{-5}$.

### 3.2.1 Modes vs snapshots

There are two input parameters for the POD reduction. The number of modes of the ROM and the number of data snapshots. Modes are the number of orders remaining in the ROM; modes represent quasinuclides. Similar to Bouma (2024, p.12–18) we make make a table that shows the effect of the input parameters on the normalised relative error. The normalised relative error is defined as

$$\epsilon = \frac{||\text{FOM} - \text{ROM}||^2}{||\text{FOM}||^2}, \tag{3.12}$$

where FOM is the full order solution and ROM is the reduced solution projected on the same full order space.

For two orders of stiffness stiffness ($\lambda \in [1, 100]$) 10 nuclide decay chain the results are shown in Table 3.1. For every set of input parameters, 100 different decay chains are reduced; The numbers reported are the average with the standard deviation.

The results indicate that more modes and snapshots increase the accuracy of the ROMs, albeit with diminishing returns. We just need enough modes and snapshots to capture the Full Order Model (FOM). More snapshots also decreases the variability in effectiveness of the ROM; the chance is higher that every characteristic is caught in the snapshots.

Interesting is the percentage of reduced models with $\epsilon > 0.95$, as seen in Table 3.2, which decrease with increasing snapshots/modes. Besides being worthless models, the so-called failures are statistically a different group. Only 16% of data should lie above one standard deviation and 2.5% above two. If we look in Table 3.1, then we find that the failure rate is higher than predicted by statistical fluctuations, suggesting a different mechanism in the case of failures.

**Table 3.1: Relative normalised error for extremely low stiffness 10 nuclide decay chain**
A 10 nuclide long long decay chain with stiffness of 2 orders is reduced to varying orders and with varying snapshots. Each number represents an average (with standard deviation) of the Proper Orthogonal Decomposition (POD) reduction of 100 different decay chains. Relative normalised errors higher than $\epsilon > 0.95$ are disregarded as reduction failures. The table shows that more modes increase accuracy; more snapshots also increase accuracy. More snapshots decreases the variability of reduction accuracy.

| | Snapshots | | | | |
|---|---|---|---|---|---|
| Reduction | 5 | 10 | 20 | 40 | 80 |
| 3 Modes | $0.65 \pm 0.25$ | $0.50 \pm 0.28$ | $0.43 \pm 0.25$ | $0.40 \pm 0.24$ | $0.36 \pm 0.27$ |
| 5 Modes | $0.37 \pm 0.28$ | $0.27 \pm 0.28$ | $0.23 \pm 0.28$ | $0.14 \pm 0.22$ | $0.09 \pm 0.14$ |
| 7 Modes | $0.42 \pm 0.31$ | $0.13 \pm 0.20$ | $0.07 \pm 0.15$ | $(2.5 \pm 5.3)\cdot10^{-2}$ | $(0.9 \pm 2.9)\cdot10^{-2}$ |
| 9 Modes | $0.41 \pm 0.31$ | $(1.2 \pm 3.6)\cdot10^{-2}$ | $(0.8 \pm 2.9)\cdot10^{-2}$ | $(0.7 \pm 4.3)\cdot10^{-2}$ | $(0.4 \pm 1.9)\cdot10^{-3}$ |

**Table 3.2: Percentage of failed reductions of Table 3.1**
All POD-based ROMs with a relative error higher than $\epsilon > 0.95$ are deemed a failure. Lower order models with less snapshots fail more often.

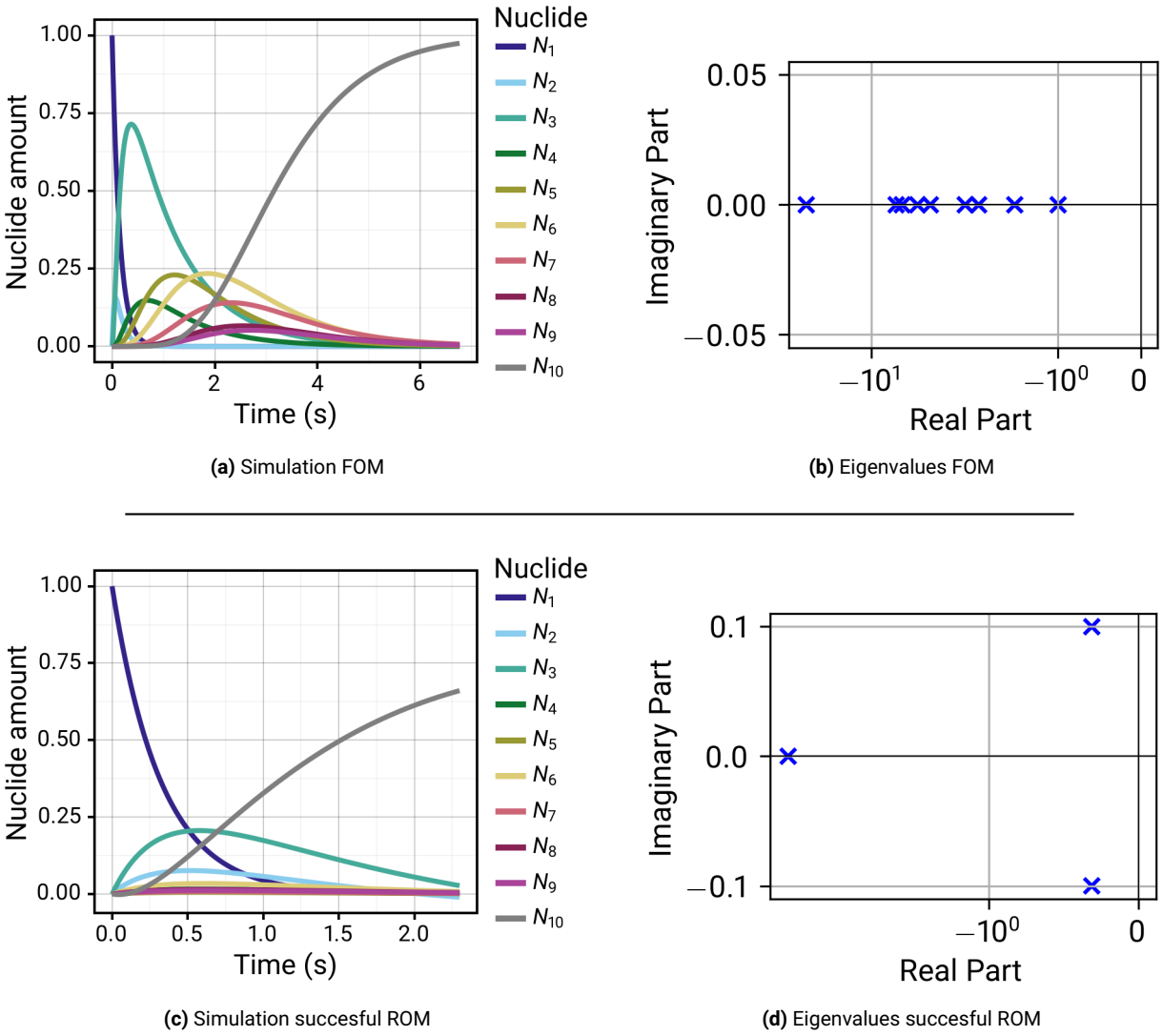
| | Snapshots | | | | |
|---|---|---|---|---|---|
| Reduction | 5 | 10 | 20 | 40 | 80 |
| 3 Modes | 37% | 21% | 10% | 7% | 11% |
| 5 Modes | 31% | 19% | 5% | 4% | 0% |
| 7 Modes | 34% | 11% | 5% | 2% | 0% |
| 9 Modes | 26% | 1% | 1% | 0% | 0% |

### 3.2.2 The cause of failure

To understand failure, we must first recognize succes, so we investigate a successful POD ROM of a decay chain. The eigendecomposition of the burnup equation (Equation 3.7) gives

$$\frac{\mathrm{d}\mathbf{N}(t)}{\mathrm{d}t} = \sum_{i=1}^{N} C_i \phi_i(t) = \sum_{i=1}^{N} C_i e^{\lambda_i t}. \tag{3.13}$$

$C_i$ are constants, determined by the initial composition of $\mathbf{N}(t_0)$, $\phi_i(t) = \epsilon^{\lambda_i t}$ are the eigenfunctions and $\lambda_i$ are the eigenvalues of the decay matrix. The eigenfunctions show that a single positive eigenvalues give an exponentially growing solution. For the negative eigenvalues, the speed of decay corresponds to the magnitude of the eigenvalue.



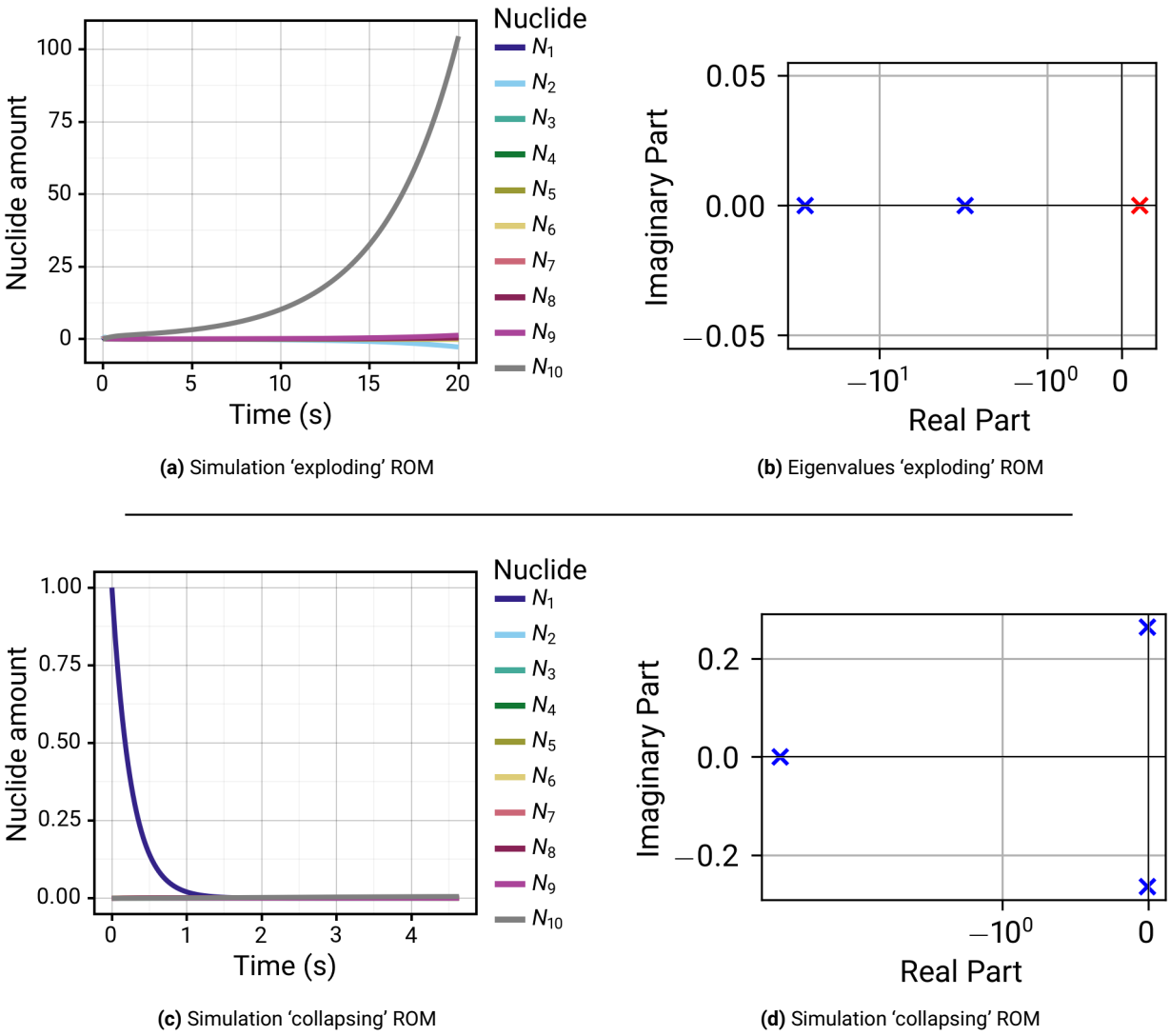

**(a)** Simulation FOM

**(b)** Eigenvalues FOM

**(c)** Simulation succesful ROM

**(d)** Eigenvalues succesful ROM

**Figure 3.5: Full Order Model and succesful Reduced Order model**
A 10 nuclide decay chain with $\lambda \in [1, 100]$ has been reduced using five snapshots. Figure (a) shows a the stable FOM. The eigenvalues of the nine unstable nuclides, shown in Figure (b) explain the stabililty: they are all real, negative and are spread logarithmically uniform. A succesful 3-order POD ROM, shown in Figure (c) shows similair eigenvalue structure. The three eigenvalues, show in Figure (d), are negative, but not all real. The two eigenvalues with an imaginary part are complex conjugates. The ratio between the real part of the largest and second largest eigenvalue of the reduced decay matrix is 10, which is smaller than the stiffness of the chain.




**(a)** Simulation 'exploding' ROM

**(b)** Eigenvalues 'exploding' ROM

**(c)** Simulation 'collapsing' ROM

**(d)** Simulation 'collapsing' ROM

**Figure 3.6: Reduction failures of a 10 nuclide decay chain using five snapshots.**
The stiffness is similar to Figure 3.5, which is $\lambda \in [1, 100]$. Figure (a) shows an 'exploding' type of unstable ROM. This type of failure is caused by one or more positive eigenvalues (indicated with a red marker), which is the case as seen in Figure (b). Figure (c) shows another type of failure: 'collapse'. The eigenvalues, shown in Figure (d), are strictly negative, but the ratio of the real part of the largest and second largest eigenvalue(s) is 525—larger than the possible stiffness of the chain. The large gap in the eigenvalues indicates a 'collapse'.

Figures 3.5 (a,b) show a decay chain with $\lambda \in [1, 100]$ with its respective eigenvalues. As expected, the eigenvalues are negative, real and within the range set by the limits of the constants in the decay matrix.

A successful POD reduction, as Figures 3.5 (c,d) show, has a similar structure in the eigenvalues of the reduced decay matrix. The eigenvalues are strictly negative and do not differ more than the stiffness of the FOM. Eigenvalues of the reduced decay matrix are not necessarily real, but can be complex conjugate pairs. Equation 3.13 allows complex conjugate pairs, as the imaginary components cancel each other out.

Reduction failures with an normalised relative error $\epsilon > 0.95$ are caused by two types of eigenvalue structures.

Any positive eigenvalue creates the 'exploding' failure, such as Figures 3.6 (a,b) show.

It does not matter how small the positive part is; if enough time passes, the exponentially growing eigenfunction dominates. The inevitable moment where the exponentially growing solution dominates is determined by the particular initial composition and magnitude of the positive eigenvalue. A stable ROM must exclude positive eigenvalues from the reduced decay matrix.

A large gap, larger than the stiffness of the FOM, between the largest and second largest eigenvalue(s) of the reduced decay matrix, causes the 'collapsing' structure, visible in Figures 3.6 (c,d). The first mode decays much more quickly than the others; and relatively contains more information as the gap in eigenvalues increases. The longer lived modes contain relatively little information, so the solution flatlines after the decay of the initial condition. The eigenvalues of the reduced decay matrix must be comparable in magnitude.

What causes the change in the eigenvalues? For a square matrix $A$ transformed by a rectangular matrix $\Phi_R$, such as a truncated POD transformation,

$$A_R = \Phi_R^T A \Phi_R, \tag{3.14}$$

we define the error matrix as the difference between the full matrix $A$ and the reduced matrix $A_R$

$$E = A - A_R = U\Sigma V^T - U_R \Sigma_R V_R^T, \tag{3.15}$$

where we have used the Singular Value Decomposition (SVD) for the second equality. Split $A$ into the part used for the reduction and the truncated part

$$E = U \begin{pmatrix} \Sigma_R & 0 \\ 0 & \Sigma_T \end{pmatrix} V^T - U_R \Sigma_R V_R^T = U_T \Sigma_T V_T^T = A_T. \tag{3.16}$$

Equation 3.16 shows we can rewrite $A + E = A_R + 2A_T$, which means that the singular values (root of eigenvalues) of $A_R$ and $A_T$ are independent. Apply the Bauer-Fike theorem with a spectral norm (Horn & Johnson, 2012, p.405–406)

$$|\hat{\lambda} - \lambda| \leq \kappa(S)||E||_2. \tag{3.17}$$

if $\hat{\lambda}$ is any eigenvalue of $A + E$, there exist an eigenvalue $\lambda$ of $A$, such that the inequality holds. $\kappa(S)$ is the condition number of $S$, which is the eigenvector basis of $A$; $\kappa_{\min}(S) = 1$, but for decay chain matrices it is comparable to the stiffness of $A$. $||E||_2 = \max(\sigma_i),\ \sigma_i \in \Sigma_T$ is the Frobenius norm of $E$.

Equation 3.17 provides an upper limit on the shift of eigenvalues of the reduced burnup matrix compared to the original matrix. A stiffer matrix $A$ increases the value $\kappa(S)$ and reducing the model to less modes increases $||E||_2$, both allowing more movement of the eigenvalues. So generally, we can assume that higher stiffness and less reduced orders leads to more reduction failures.

We reduce 1200 different decay chains with varying stiffness and reduction orders to quantify the failure rate, which is shown in table 3.3. The table confirms the assumption that for stiffness below the floating point precision ($\sim 10^{16}$) the failure rate increases with higher stiffness or less reduced modes. The number of failures also decreases with increasing data snapshots, which can be explained with a similar reasoning as above, but include extra terms in the error matrix $E$, caused by taking a subset of the data.

Above floating point precision the number of failures first decreases with increasing reduction orders, but then increases again, which suggest a different mechanism. Floating point errors are introduced when performing the transformation to the reduced basis $A_R = \Phi_R^T A \Phi_R$. Each multiplication amplifies numerical errors with a maximum of the condition number of $A$, which is determined by the stiffness. The numerical errors cause every eigenvalue to

move slighty from its exact position. The eigenvalues close to the positive plane thus have a chance to hop from the stable to unstable region. With more reduced orders there are more eigenvalues that have a chance to hop over the line, thus increasing the chance of a failure with increasing orders.

**Table 3.3: Unstabilized Proper Orthogonal Decomposition reduction performance**
200 nuclides long decay chains with decay constants logarithmically uniformly picked with varying stiffness ($\lambda_{\min} = 1$, $\lambda_{\max}$ varies) are reduced to 10, 30 or 50 orders based on 50 or 500 evenly-spaced snapshots. Each number represents an average (with standard deviation) of 100 different decay chains. Relative normalised errors higher than $\epsilon > 0.95$ are disregarded as reduction failures. More snapshots and modes perform better at the cost of speedup. The number of failures tends to increase with stiffness, whilst decreasing with more modes and snapshots, except for the extremely stiff ($10^{40}$) models, which behave rather erratically.

| | | 50 Snapshots | | | 500 Snapshots | | |
|---|---|---|---|---|---|---|---|
| | Reduction | $\epsilon_{\text{total}}$ | Speed-up | Fails | $\epsilon_{\text{total}}$ | Speed-up | Fails |
| Stiffness = $10^5$ | 10 Modes | $0.40 \pm 0.26$ | $21.5 \pm 1.4$ | 77% | $0.27 \pm 0.30$ | $21.8 \pm 2.4$ | 54% |
| | 30 Modes | $0.48 \pm 0.31$ | $9.89 \pm 0.29$ | 70% | $0.12 \pm 0.21$ | $9.98 \pm 0.19$ | 35% |
| | 50 Modes | $0.39 \pm 0.30$ | $6.46 \pm 0.18$ | 44% | $0.08 \pm 0.17$ | $6.49 \pm 0.20$ | 16% |
| Stiffness = $10^{12}$ | 10 Modes | $0.43 \pm 0.35$ | $21.4 \pm 1.2$ | 89% | $0.19 \pm 0.26$ | $21.0 \pm 2.2$ | 70% |
| | 30 Modes | $0.42 \pm 0.35$ | $9.84 \pm 0.21$ | 69% | $0.27 \pm 0.35$ | $9.87 \pm 0.19$ | 56% |
| | 50 Modes | $0.18 \pm 0.24$ | $6.39 \pm 0.18$ | 56% | $0.10 \pm 0.20$ | $6.37 \pm 0.23$ | 44% |
| Stiffness = $10^{25}$ | 10 Modes | $0.60 \pm 0.10$ | $22.5 \pm 1.3$ | 97% | $0.29 \pm 0.38$ | $21.3 \pm 1.4$ | 89% |
| | 30 Modes | $0.25 \pm 0.34$ | $9.66 \pm 0.25$ | 63% | $0.13 \pm 0.23$ | $9.62 \pm 0.20$ | 66% |
| | 50 Modes | $0.36 \pm 0.35$ | $6.30 \pm 0.15$ | 85% | $0.18 \pm 0.28$ | $6.21 \pm 0.27$ | 82% |
| Stiffness = $10^{40}$ | 10 Modes | 0.4 | 23.0 | 99% | $0.26 \pm 0.26$ | $21.30 \pm 0.53$ | 94% |
| | 30 Modes | $0.58 \pm 0.29$ | $9.62 \pm 0.14$ | 92% | $0.14 \pm 0.18$ | $9.66 \pm 0.17$ | 97% |
| | 50 Modes | 0.3 | 6.1 | 99% | | | 100% |

## 3.3 Chapter review

The current chapter discusses the Proper Orthogonal Decomposition (POD) technique of making Reduced Order Models (ROMs). The method is summarised in Section 3.1.6. Section 3.2 shows results of the method when tested with the decay chain test model of Section 2.4. The results show that standard POD Reduced Order Model (ROM) is unreliable; the ROM produces 'collapsing' and 'exploding' models as seen in Section 3.2.2. The first is caused by excessive spread in the eigenvalues of the reduced modes, which happens when the number of reduced modes is low. The quickest decaying mode contains too much information. The second is caused by positive, unstable eigenvalues—the solution exponentially grows. The POD projection and numerical errors can displace eigenvalues to the positive plane.

Table 3.3 gives an overview of the reduction performance on varying stiffness decay chain models with varying number of snapshots and reduction modes; we can draw seven conclusions about POD ROMs:

1. More reduction modes increase accuracy.
2. More snapshots increases accuracy.
3. Speed-up is only dependent on the number of reduced modes; fewer modes increase speed.
4. Stiffer models are harder to reduce; accuracy decreases and the failure rate increases.
5. More snapshots and reduced modes decrease the failure rate.

6. If the stiffness of the Full Order Model (FOM) exceeds the floating point precision ($\sim 10^{16}$), then failure rate increases.
7. Due to over a 90% failure rate for stiffness of $10^{40}$, POD ROMs are inadequate to reduce full burnup problems.

The first three conclusions are readily inferred from the theory of POD ROMs. We need to strike a balance between minimising the number of reduced modes and maintaining sufficient accuracy; more snapshots can help, but increase the expense in the offline phase, as the FOM needs to be simulated more. The fourth and fifth is not obvious; for example, a POD review study of Lassila et al. (2014) does not mention stiffness of the FOM when discussing stability. The sixth shows that both numerical precision and the POD projection influence the failure rate, albeit in a different ways. The standard POD performs best for low stiffness, highly stable problems; burnup analysis are neither. One advantage is the simplicity of the technique and possible reduction effectiveness. The technique easily extends to a broader input range by concetatenating data for different inputs to the snapshot matrix. For burnup anylisis the seventh conclusion holds true, which is consistend with previous reseach by Bouma (2024, p.25–28). To reduce the SERPENT burnup matrix (stiffness $\sim 10^{40}$), we must use a more robust technique; the standard POD technique fails too often at this level of stiffness.

A possible solution is to use symmetric inproduct, which theoretically guarantees that the eigenvalues do not cross from the negative to the positive plane when projecting the POD on the full order space (Tabandeh et al., 2016)(Rezaian & Wei, 2020). Currently, it is unclear wheter it would work when the stiffness of the problem exceeds the numerical precision. Besides, implementing such methods require rewriting large codebases, as standard packages are lacking.

The next chapter presents an alternative, easier to implement, strategy, which detects and moves the unstable eigenvalues.

# Chapter 4

# Heuristically Corrected Proper Orthogonal Decomposition

The standard POD technique produces ROMs, which efficiently simulate the burnup equation. The ROMs, if successful, depending on the number of reduction modes and snapshots are accurate, while achieving significant speedup. Nevertheless, standard POD Reduced Order Modelling (ROM) is not recommended for the extremely stiff burnup problem; the ROMs are unstable for 90% of the decay chains with a stiffness of $10^{40}$ with no a priori way to determine success.

The current chapter mitigates unstable ROMs by eigenvalue shift and reassignment. The idea is to detect the eigenvalues that cause the ROM to fail and modify them. It is a straightforward approach that is easy to implement.

The first section explains the three step stabilisation method for the POD. The second section shows results of the corrected POD by showcasing specific decay chains and providing a table with many repeated reductions. Finally, the last section reviews the corrected POD and gives recommendations for further improvements.

## 4.1 Search and reassign

Section 3.2.2 shows that there are two causes of failure. Either a positive value causing an 'exploding' solution, or too much spread in the negative eigenvalues causing a 'collapsing' solution. There is no way to determine the outcome before performing the reduction process. However, we can implement mitigation strategies. This research implements three mitigation strategies, one for 'collapsing' and two for 'exploding' solutions. The eigenvalue structure of a 'collapsing' Reduced Order Model (ROM) has a huge gap between the largest and second largest eigenvalue of the reduced decay matrix. An 'exploding' ROM has at least one positive eigenvalue. Given that the Full Order Model (FOM) has strictly negative eigenvalues, which is true for stable models, there are two mechanisms that move eigenvalues into the positive plane. Numerical imprecision, caused by the mathematical operations of the reduction and projection, cause small movements of all eigenvalues. The second involves large movements of some eigenvalues caused by the Proper Orthogonal Decomposition (POD) method itself.

### 4.1.1 Preventing eigenvalue gaps: use enough reduced modes

As the POD Reduced Order Modelling (ROM) method randomly displaces eigenvalues, the mitigation for 'collapse' is to add more modes. A large gap between the first and second most negative eigenvalue causes the 'collapse'. More modes means a greater number of different

eigenvalues of the reduced decay matrix, increasing the likelihood of filling the gap. Above a certain number of reduced modes, unfilled gaps become unlikely. Stiffer problems create a wider range of eigenvalues, increasing the gap size, thus generally require more modes to counteract gap formation. We determine the minimal number of modes to counteract 'collapse' by testing varying reduction modes. In other words, use more modes until we have success.

### 4.1.2 Preventing small numerical errors: shifting eigenvalues

We can shift all eigenvalues slightly towards the negative plane to counteract most instabilities caused by numerical errors. Numerical errors allow every eigenvalue to shift slightly with every mathematical operation. The magnitude of the shift scales with the condition number of the matrices involved in the mathematical operation. The movements are small, when compared to the maximum displacement the POD allows.

Given a matrix $A$, we can shift the eigenvalues by adding another similarly sized scalar matrix $B = cI$. The matrices commute: $AB = BA$. Commuting matrices are simultaneously diagonalizable, by virtue of the spectral theorem, thus

$$A + B = UD_AU^{-1} + UD_BU^{-1} = U(D_A + D_B)U^{-1}. \tag{4.1}$$

The diagonal matrices $D_A$ and $D_B$ contain the eigenvalues of the matrices $A$ and $B$ respectively; $D_A + D_B$ effectively shifts the eigenvalues. The amount by which the eigenvalues are shifted is determined heuristically.

### 4.1.3 Preventing eigenvalue displacement: reassign eigenvalues

The POD method can displace eigenvalues within the limit set by Equation 3.17; eigenvalues of the FOM can move to almost anywhere in the positive plane. We can make an eigenvalue decomposition of the reduced burnup matrix

$$A_R = U_R\Lambda_R U_R^{-1} \tag{4.2}$$

to find positive eigenvalues. We change the sign of the positive real component of the eigenvalue, reassigning the eigenvalue to the negative plane. The reduced burnup matrix is recomposed with the eigenvalue decomposition using the reassigned eigenvalues

$$\hat{A}_R = U_R\hat{\Lambda}_R U_R^{-1}. \tag{4.3}$$

### 4.1.4 The overall mitigation strategy

We stabilise the Proper Orthogonal Decomposition (POD) Reduced Order Model (ROM) with three steps:

1. Use enough modes to prevent 'collapsing' Reduced Order Models (ROMs).
2. If there are positive eigenvalues, shift all eigenvalues by a negative value with Equation 4.1 , mitigating numerical errors.
3. If positive eigenvalues remain in $\Lambda$ as found with Equation 4.2 , reassign them to the negative plane and rebuild the burnup matrix according to Equation 4.3 .

## 4.2 Performance

The stabilisation strategy should create Reduced Order Models (ROMs) that model the Full Order Model (FOM) accurately. We use a decay chain model with varying stiffness to check the performance. Not only do we check if the results are as intended, but we can also see

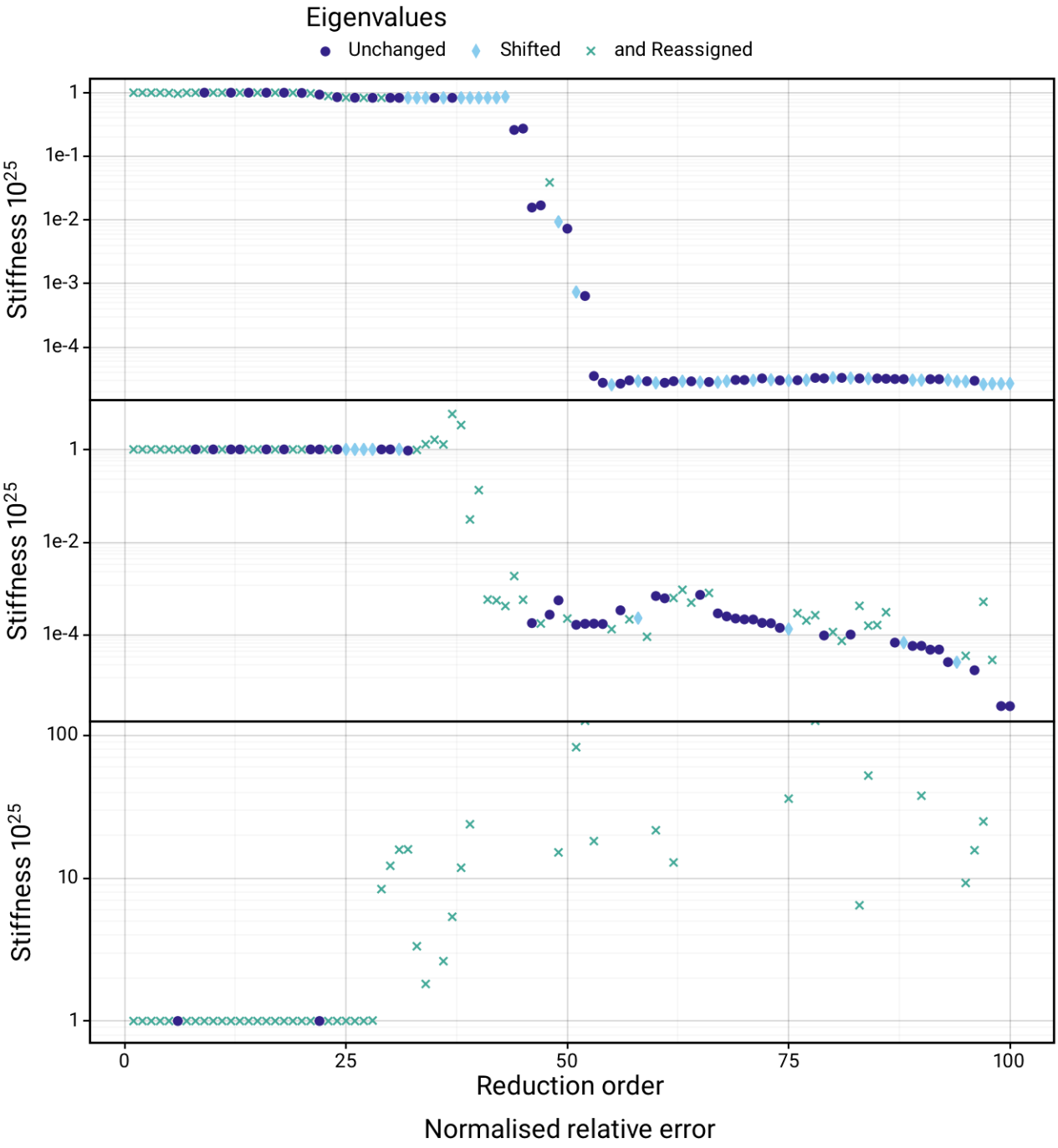



**Figure 4.1: Heuristically corrected POD strategy and performance**
For the three levels of stiffness we reduced a single 200 nuclide-long decay chain to incrementing modes, using 500 snapshots. If the reduced burnup matrix contains positive eigenvalues, indicating an unstable model, then the mitigation strategy of Section 4.1.4 is applied; first the eigenvalues are shifted by $-10^{-8}$ and remaining positives are reassigned to the negative plane. For a stiffness of $10^5$ (top figure) the stabilisation methods work as expected; there are no relative errors above one. The method also works for a stiffness of $10^{12}$ (middle figure), although the spread in error increases. The bottom figure shows the result for a stiffness of $10^{25}$ for which the stabilisation does not work. The errors become larger than one and reduced models failed to solve, shown as missing data points.

how often the eigenvalues are shifted and reassigned. An excerpt of the code is found in Appendix E.

Figure 4.1 shows the reduction strategy applied for a decay chain with stiffness of $10^5$, $10^{12}$, $10^{25}$, and $10^{40}$.

For low reduction orders, we see that the normalised relative error is close to 1, as expected; the model collapses very often due to the large difference between the most and second most significant mode.

For a stiffness of $10^5$, the stabilisation works as intended when enough reduced modes are used. Eigenvalue shift stabilises most unstable ROMs, which suggests that the unstable eigenvalues are within $10^8$ of the stable plane.

When the stiffness gets higher, such as to 12 orders of stiffness, the stabilisation method reassigns eigenvalues frequently, which is not very reliable. The normalised relative error can still exceed one after reassignment and the errors tend to be higher than without reassignment. Reassignment, however, does successfully prevent real 'exploding' models for a stiffness lower than the Float64 ($10^{16}$) precision.

If the stiffness exceed Float64 precision, such as at $10^{25}$ , then the reassignment strategy is triggered, but the reassigned eigenvalues do not stabilise the model. The problem is that the stabilisation strategy is dependent on correct detection of positive eigenvalues. The eigenvalue algorithm accuracy decreases with increasing condition number; the possible difference between the real eigenvalue and the numerically found eigenvalue increases with increasing condition number/stiffness. The numerical errors cause false negatives and false positives when determining the positive eigenvalues, so random eigenvalues get reassigned without the intended stabilisation.

Table 4.1 confirms the previous observations. For a stiffness of $10^5$ the stabilisation method prevents failure, especially when using more reduction modes, which suggests that the only failures remaining are 'collapsing' models due to insufficient reduced modes. For a stiffness of $10^{12}$, the number of failures increases, which is explained by the poorer performance of the stabilisation and the higher chance for the eigenvalues to contain a gap that causes a 'collapse'. For stiffness beyond numerical precision, such as $10^{25}$ and $10^{40}$ , the corrected POD performs worse than the uncorrected POD described in Chapter 3. The increase of failures could be explained by extra numerical errors introduced by the stabilisation methods, but a precise explanation is currently lacking.

## 4.3 Chapter review

A Proper Orthogonal Decomposition (POD) with enough reduced modes, which shifts and reassigns positive eigenvalues, guarantees stability when the Full Order Model (FOM) has a stiffness lower than the numerical precision. When the stiffness is larger than numerical precision, the strategy does not work as intended, which is confirmed by Table 4.1. We conclude about the stabilisation strategy of the current chapter:

1. Enough reduced order modes (around 50 for a 200 order FOM) counteract collapsing models effectively.

2. Shifting eigenvalues stabilises most exploding models for stiffness many orders beneath numerical precision.

3. Reassigning eigenvalues to the negative plane stabilises exploding models for stiffness smaller than numerical precision.

4. Both strategies are prone to false positive detection of unstable eigenvalues.

**Table 4.1: Performance of the heuristically corrected POD on decay chains**
The performance of the corrected Proper Orthogonal Decomposition (POD) has been tried for four levels of stiffness decay chains. Each decay chain is 200 nuclides long and reduced to varying orders using 500 snapshots. The values are the average and sigma deviation of 100 reruns. A fail is defined as $\epsilon_{\text{total}} > 0.95$. Compared to Table 3.3 the reduction fails less for stiffness of $10^5$ and $10^{12}$, but the number of times 'detection' of positive eigenvalues trigger shifting and reassignment of eigenvalues significantly exceeds the fails of the uncorrected POD, suggesting false positive detections of unstable eigenvalues. For a stiffness of $10^{25}$ and $10^{40}$, the stabilisation method seem to destabilise the Reduced Order Model (ROM) when compared to the uncorrected POD method.

| | Reduction | $\epsilon_{\text{total}}$ | Speed-up | Shifted | Reassigned | Fails |
|---|---|---|---|---|---|---|
| Stiffness = $10^5$ | 10 modes | $0.37 \pm 0.29$ | $22.2 \pm 1.3$ | 35% | 35% | 45% |
| | 30 modes | $0.19 \pm 0.31$ | $10.45 \pm 0.96$ | 41% | 28% | 34% |
| | 50 modes | $0.06 \pm 0.15$ | $6.63 \pm 0.19$ | 50% | 5% | 8% |
| | 60 modes | $0.02 \pm 0.07$ | $5.76 \pm 0.12$ | 59% | 4% | 1% |
| Stiffness = $10^{12}$ | 10 modes | $0.33 \pm 0.28$ | $22.0 \pm 2.5$ | 23% | 23% | 72% |
| | 30 modes | $0.28 \pm 0.35$ | $10.24 \pm 0.76$ | 40% | 33% | 47% |
| | 50 modes | $0.06 \pm 0.15$ | $6.52 \pm 0.24$ | 67% | 62% | 18% |
| | 60 modes | $0.10 \pm 0.20$ | $5.72 \pm 0.15$ | 71% | 67% | 14% |
| Stiffness = $10^{25}$ | 10 modes | $0.55 \pm 0.36$ | $22.2 \pm 2.1$ | 14% | 14% | 83% |
| | 30 modes | $0.67 \pm 0.30$ | $10.14 \pm 0.48$ | 6% | 6% | 94% |
| | 50 modes | | | 0% | 0% | 100% |
| | 60 modes | | | 0% | 0% | 100% |
| Stiffness = $10^{40}$ | 10 modes | 0.12 | 23.15 | 1% | 1% | 98% |
| | 30 modes | | | 0% | 0% | 100% |
| | 50 modes | 0.79 | 4.7 | 2% | 2% | 98% |
| | 60 modes | 0.8 | 2.6 | 1% | 1% | 99% |

5. Stiffness larger than the numerical precision cause too many false positive and false negative detection of positive eigenvalues for the stabilisation strategy to work correctly.

6. The current stabilisation strategy is not adequate to stabilise the POD method for application of burnup calculations.

There are possible improvements for the stabilisation strategy. Increasing the numerical precision could help to more accurately detect the positive eigenvalues. For example, the Julia language allows straightforward implementation of higher precision floating point formats. Rezaian and Wei (2018) presents a more complex and efficient eigenvalue reassignment strategy, which corrects the POD better.

Besides POD there are more methods to create Reduced Order Models (ROMs). Moment matching ROMs, beyond the scope of the current research, could be very useful for the burnup problem, as they guarantee stability. They can include parameters, such as neutron flux or fuel composition, in the Reduced Order Model (ROM) (Quarteroni & Rozza, 2014, p.159–185).

# Chapter 5

# Balanced Truncation

Moore (1981) realised that there is another way to reduce physical systems. Instead of minimizing the residual of the correlation inner product of data with the POD, we can optimize for the observability and controllability of the system. Observability and controllability are quantities that connect a system state to the measured output and chosen input respectively. Balancing these two quantities gives an ordered basis of the system from most likely state to least likely state given experimental conditions; Balanced Truncation (BT) truncates the least likely states, to produce a ROM.

Assume that we have a linear time invariant system, so we can write

$$\begin{aligned} \frac{\mathrm{d}\mathbf{N}(t)}{\mathrm{d}t} &= A\mathbf{N}(t) + B\mathbf{u}(t), \\ \mathbf{y}(t) &= C\mathbf{N}(t). \end{aligned} \tag{5.1}$$

$\mathbf{N}(t)$ is the state of the system; $A$ is the state evolution matrix; $B$ the input matrix with $\mathbf{u}(t)$ the input; $C$ is the measurement matrix and $\mathbf{y}(t)$ is the measurement. Each column of $B$ and each row of $C$ can be thought of as a different input or measurement device. The number of rows of $B$ is equal to the number of rows of $A$ and the number of columns of $C$ is equal to the number of columns of $A$.

Conceptually $A$ represents the 'invisible' internals of the system; the columns of $B$ represent external forces; $C$ represents the influence of the system on the environment. BT finds lower order matrices that represent the system als closely as possible considering interactions to and from the environment.

BT has four advantages over POD. First, the explicit split between the system state, measurement and input is that we can modify measurement and input settings without modifying the system itself. Second, a stable FOM guarantees a stable ROM with BT. Stability can be guaranteed. Third, BT requires no expensive snapshots to be generated, skipping the offline phase of the POD. Fourth, the upper error bound can be calculated a priori, so we can choose the reduction order according to our need.

A disadvantage is that BT requires the state matrices $A, B, C$ to be known explicitly, i.e. it is an intrusive modelling method. Additionally, it more computationally expensive compared to POD, which makes it unfeasible beyond a certain system size.

The first section introduces control theory. The second section tests the BT on decay chain models. The third and final section reviews the chapter and concludes that although BT performs better than POD for stiffness up to Float64 precision ($10^{16}$), we need another technique for reducing the burnup problem.

## 5.1 Introduction to Balanced Truncation

To understand what Balanced Truncation (BT) is, we need to introduce five concepts from control theory: controllability, observability, Lyapunov equations, balancing transformation, and truncation. The next five subsections explain each of these concepts. The final subsection provides a summary with concrete implementation steps, so we can use the method for constructing a Reduced Order Model (ROM).

If the reader would like to go more in-depth with control theory, the author recommends the Control Bootcamp video lecture series by Steve Brunton (Steve Brunton, 2017).

### 5.1.1 Controllability

Systems, belonging to a certain space, are not always able to reach every state in the space using the input of the system. Water in an open pan, dictated by temperature, cannot reach states above the boiling point because it evaporates, even if we input a lot of energy. Another example is a car, which, although described by six dimensions (three for place and three more for the direction), can hardly reach states far above the road, ridiculous amounts of speed or very steep angles. Certain states are unreachable, even by the best controller.

Formally, given a state evolution matrix $A$ with $\mathbb{R}^n$ dimensions and control matrix $B$, we define the controllability as

$$C = [B, AB, A^2B, \dots, A^{n-1}B]. \tag{5.2}$$

The span of the independent columns of $C$ forms a Krylov subspace, representing the directions in which the state can be manipulated with a certain control input $\mathbf{u}(t)$. If $C$ spans the full state space—the number of independent columns is equal to the that of the state space—then every state can be reached within a finite time with some input signal $\mathbf{u}(t)$ (Brunton & Kutz, 2022, p.322–324).

To quantify the controllability we can use the infinite controllability Gramian,

$$W_C = \int_0^\infty e^{A\tau} B B^* e^{A^*\tau} \, \mathrm{d}\tau, \tag{5.3}$$

which can be thought of as an eigenvector decomposition of the control space. The span of the controllability Gramian gives the directions which the system can reach, the eigenvalues the inversely correspond to the costs; directions with smaller eigenvalues cost more energy and vice versa (Antoulas, 2005, p. 67–74, 78–80).

### 5.1.2 Observability

Conceptually, observability refers to the group of system states $\mathbf{N}(t)$ that can be determined on the measurement history $\mathbf{y}(t)$. For example, for a thermometer that can only measure up to 80 degrees Celsius the state of the system is unobservable when put in boiling water. Another example would be the speedometer in a car. Although the full movement state of a car can be described using a three dimensional vector, we usually only care about the magnitude and thus not observe the full state, but only a subspace of it.

For a state evolution matrix $A$ with $\mathbb{R}^n$ dimensions and measurement matrix $C$, the observability is defined as

$$O = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix}. \tag{5.4}$$

The span of the independent rows of $O$ form a subspace of states that can be inferred from any measurement history $\mathbf{y}(t)$. If the observability matrix spans the full state space, then the system is fully observable in a finite time (Brunton & Kutz, 2022, p.324).

We can also define an infinite Gramian for the observability $W_O$

$$W_O = \int_0^\infty e^{A^*\tau} C^* C e^{A\tau} \, \mathrm{d}\tau, \tag{5.5}$$

which can be thought of as an eigenvector decomposition of the control space. The span of the observability Gramian and corresponding eigenvalues indicate which directions are costly to observe; directions corresponding to smaller eigenvalues are harder to observe and vice versa (Antoulas, 2005, p. 74–80).

### 5.1.3 Lyapunov equations

Equation 5.3 and 5.5 are in practice not very useful. Instead, we can calculate the Gramians via Lyapunov equations (Antoulas, 2005, p. 78–79). The controllability Gramian $W_C$ can be found by solving

$$AW_C + W_C A^* + BB^* = 0, \tag{5.6}$$

and the observability Gramian by

$$A^* W_O + W_O A + C^* C = 0. \tag{5.7}$$

The mathematical properties of the Lyapunov equations require that the matrices $W_C$, $B$, $W_O$, $C$ must be symmetric. The Gramians are thus symmetric matrices.

### 5.1.4 Balancing transformation

The controllability and observability Gramians quantify which states are hard to reach or observe, both of which are not interesting states, as the system is seldom in a hard to reach state or we do not know if it is in a hard to observe state. It would be beneficial to find a basis that spans directions that are both easy to control and observe. Finding this basis, which is done by balancing the two Gramians, is called a balancing transformation.

The balanced transformation is performed by a simultaneous diagonalization of the controllability and observability Gramians. Start with

$$W_C = UU^*, \tag{5.8}$$

where $U$ is the Cholesky factor of $W_C$. We then take an eigenvalue decomposition of

$$U^* W_O U = K\Sigma^2 K^*, \tag{5.9}$$

with $K$ the eigenvectors and $\Sigma^2$ the eigenvalues—square of the singular values $\Sigma$. The balancing transformation $T$ of the system is (Antoulas, 2005, p.207–211)

$$T = \Sigma^{1/2} K^* U^{-1} \leftrightarrow T^{-1} = UK\Sigma^{-1/2}. \tag{5.10}$$

### 5.1.5 Truncation

The system of Equation 5.1 can be projected to the balanced space similarly to Section Projecting the burnup model. Begin by balancing

$$\mathbf{N}(t) = T\hat{\mathbf{N}}(t), \tag{5.11}$$

where $\hat{\mathbf{N}}(t)$ is the system state in the balanced basis. Equation 5.1 can be expressed in this balanced coordinate as

$$\begin{aligned} \frac{\mathrm{d}\hat{\mathbf{N}}(t)}{\mathrm{d}t} = T^{-1}AT\hat{\mathbf{N}}(t) + T^{-1}B\mathbf{u}(t) = \hat{A}\hat{\mathbf{N}}(t) + \hat{B}\mathbf{u}(t) \\ \mathbf{y}(t) = CT\hat{\mathbf{N}}(t) = \hat{C}\hat{\mathbf{N}}(t). \end{aligned} \tag{5.12}$$

Making a ROM is then as easy as truncating the least reachable and observable dimensions by truncating the balanced transformation $T \to T_R$ in the equation above. We have achieved BT. If we determine the Hankel singular values with

$$\sigma_i = \sqrt{\lambda_i(W_C W_O)}, \tag{5.13}$$

where $\lambda_i(W_C W_O)$ are the eigenvalues of the product of the Gramian, then the error of the ROM is bounded by twice the sum of the truncated Hankel singular values

$$\epsilon \leq 2 \sum_{i=k+1}^{N} \sigma_i, \tag{5.14}$$

where $N$ is the dimension of the Full Order Model (FOM) and $k$ is the dimension of the ROM (Antoulas, 2005, p.211–215).

### 5.1.6 Summary

BT is a Reduced Order Modelling (ROM) method which truncates a basis that is balanced between the most controllable and observable states. The steps are:

1. Determine the system matrices A, B, C from Equation 5.1.
2. Solve for the controllability and reachability Gramians $W_C$ and $W_O$ with Lyapunov Equations 5.6 and 5.7.
3. Calculate the balancing basis $T$ and the inverse $T^{-1}$ with Equations 5.8 – 5.10.
4. Truncate the matrices $T$ and $T^{-1}$ to obtain $T_R$ and $T_R^{-1}$.
5. Simulate the reduced system using Equation 5.12.
6. Transform back to full order space with $\hat{\mathbf{N}}(t) = T_R^{-1}\mathbf{N}(t)$.

The Python Control Systems Library (https://python-control.readthedocs.io/en/0.10.0/) provides useful routines for using BT.

## 5.2 Reduction performance on decay chain models

Balanced Truncation (BT) should preserve stability of the original model and only requires a description of the system to make a Reduced Order Model (ROM). The current section implements BT for decay chain models, so we can compare the performance with that of the (heuristically corrected) Proper Orthogonal Decomposition (POD).

The general description of a linear time invariant system is given by equation 5.1. We see that the matrices $A, B$, and $C$ determine the system fully, so we need to define each of the three matrices for the decay chain.

The state evolution matrix $A$ is the decay matrix, as it describes how the system evolves through time. $A$ is a square matrix of size $N \times N$, where $N$ is the number of nuclides in the chain.

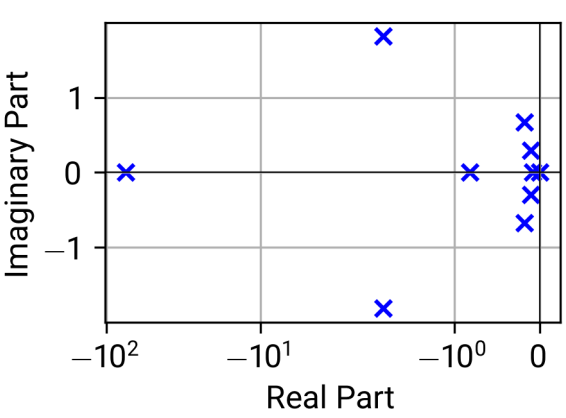
The system input matrix $B$ has some freedom of choice. For example, we could input

different initial fuel compositions. In the simplified case of the decay chain, however, the input is the first nuclide. $B$ is thus a single column vector of size $N$.

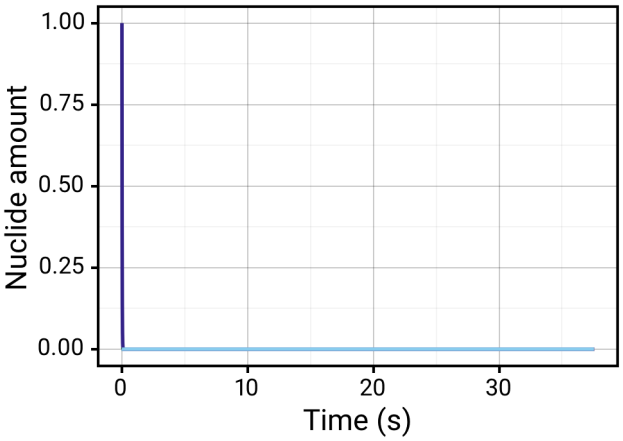
The measurement matrix $C$ represents all the states that can be measured. As we are running a simulation, we have access to all nuclide information independently, so the matrix $C$ is an identity matrix of size $N \times N$.

As initial state we use $\hat{\mathbf{x}}(t_0) = 0$ and for the input vector we use $\mathbf{u}(t_0) = N(t_0)$ and $\mathbf{u}(t > 0) = 0$. An excert of code executed for measuring the performance can be found in Appendix F.

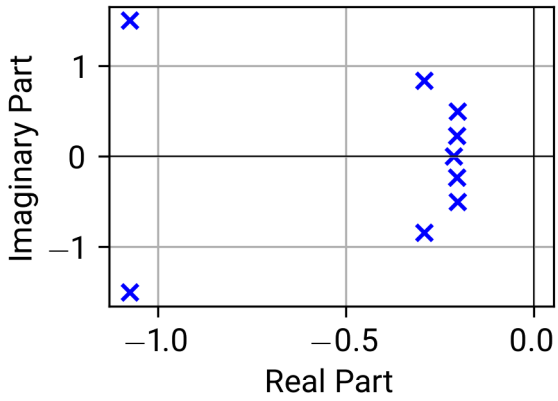
We now investigate two decay chain examples where POD Reduced Order Models (ROMs) fail and compare them with BT.



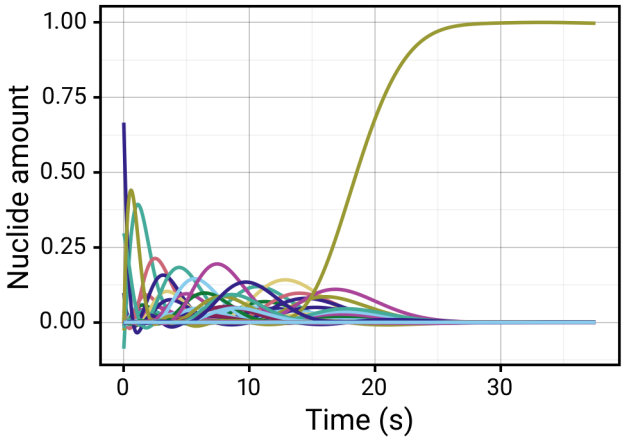

**(a)** Eigenvalues POD modes




**(b)** Simulation POD ROM

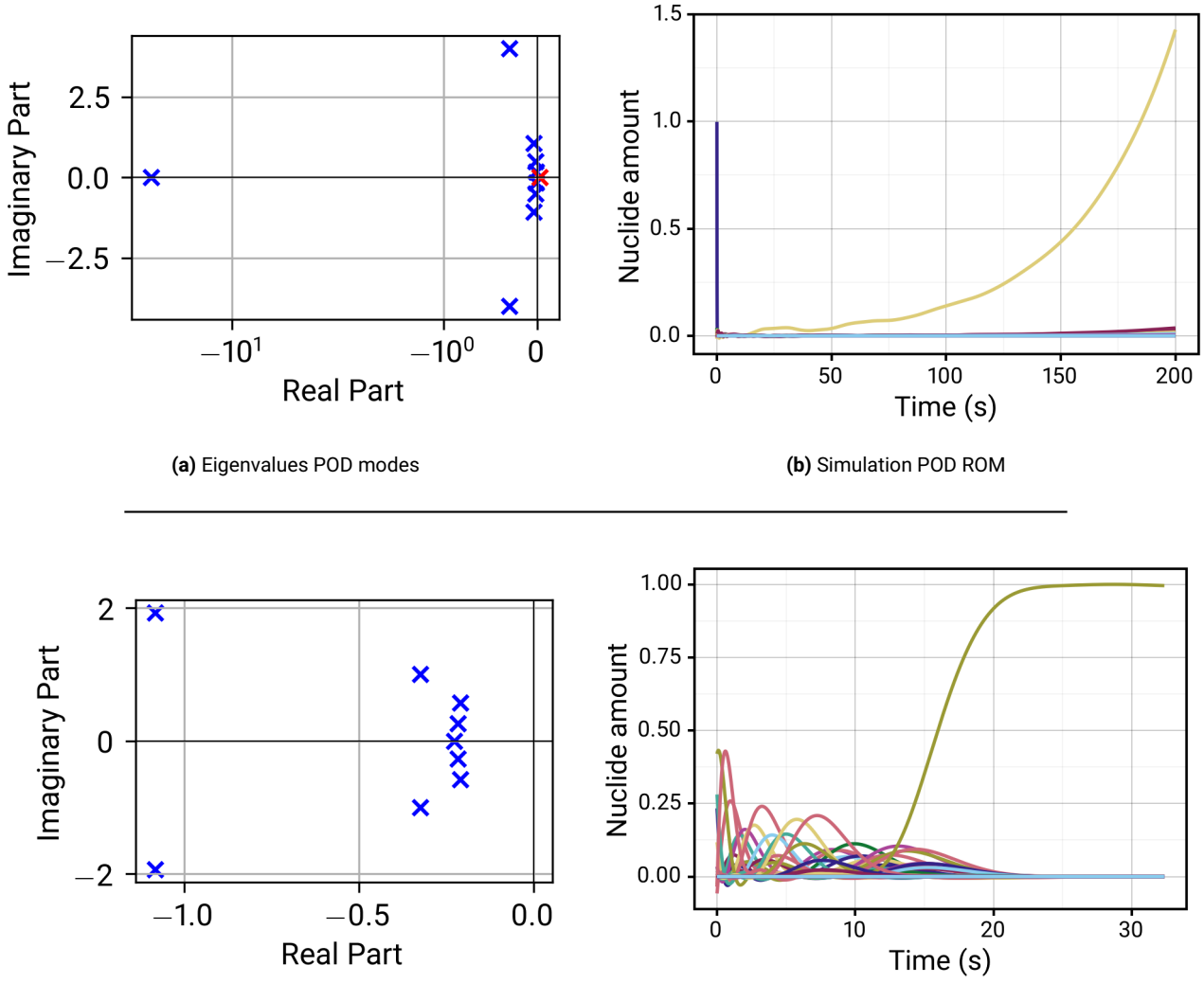



**(c)** Eigenvalues BT modes




**(d)** Simulation BT ROM

**Figure 5.1: Collapsing Proper Orthogonal Decomposition compared to Balanced Truncation**
A 200 nuclide decay chain ($\lambda \in [1, 10^5]$) has been reduced to 10 modes using POD with 500 snapshots. Figure **(a)** shows the eigenvalues of the reduced modes; the eigenvalues have a large gap indicating a collapsing model, which is confirmed by **(b)**. BT for the same decay chain locates the eigenvalue much closer together, as seen in **(c)**. The resulting simulation is shown in **(d)**, which has a normalised relative error of 4% with the Full Order Model (FOM).




**(a)** Eigenvalues POD modes

**(b)** Simulation POD ROM

**(c)** Eigenvalues BT modes

**(d)** Simulation BT ROM

**Figure 5.2: Exploding Proper Orthogonal Decomposition compared to Balanced Truncation**
A 200 nuclide decay chain ($\lambda \in [1, 10^5]$) has been reduced to 10 modes using POD with 500 snapshots. The particular decay constants cause the POD to create a positive eigenvalue (indicated with a red marker), as seen in **(a)**. The simulation has been extended to 200 seconds in **(b)** to show the exponential growth clearly. BT for the same decay chain guarantees stability; the eigenvalues are all in the stable region as seen in **(c)**. The BT ROM simulation, shown in **(d)**, has a 5% normalised relative error.

### 5.2.1 Stability comparison

Decay chains showed instabilities after reduction in the previous chapters about (heurstically corrected) POD, especially as stiffness of the decay constants increases. There were two types of unstable models: 'collapsing', caused by a large difference in the most and second most negative eigenvalue and 'exploding', caused by a single or more positive eigenvalues; both are described in detail in Section 3.2.2.

BT should produce only stable ROMs if the FOM is stable, ensuring reliability of the reduced model. To test, we compare a specific case of POD 'collapse' and 'explosion' with a BT ROM for the same chain.

Figure 5.1 shows a 200 nuclide decay chain with five magnitude orders of stiffness of

which the 10 order POD ROM collapses. BT on the same chain does produce a stable ROM, as the eigenvalues are placed much closer together. The relative normalised error is only 4%, which is on the low side of the error bound for what POD ROMs can achieve according to Table 3.3 and 4.1.

Figure 5.2 shows similar results for another five magnitude orders of stiffness decay chain, which explodes when using POD; the normalised relative error with the FOM is 5%.

BT does not only give a stable result where POD does not, but it also provides an efficient ROM.

Repeated BT reductions, as shown in Table 5.1, demonstrates that stability and efficiency of decay chains are reliable. None of the 800 decay chains failed to reduce with BT. The average and variation of the normalised total error with the FOM is also much lower than for (heurstically corrected) POD.

For higher stiffness, the variation increases. The average total error is lower for 10 reduced modes, probably because stiffer decay chains have more extremely short lived nuclides, which can be left out without loss of accuracy. For more modes, the variation in error dominates the average error, limiting the average error.

**Table 5.1: Reduction performance of Balanced Truncation**
For each stiffness and reduction setting, 100 different 200-nuclide-long decay chains have been reduced using BT. The average and standard deviation are reported in the table. For stiffness of $10^5$ and $10^{12}$, none of the reductions failed. With increasing reduced modes precision increases while losing speed-up compared to the FOM. Results for stiffness of $10^{25}$ and $10^{40}$ are not reported, as all reductions failed. The 64 bit implementation of BT did not work for stiffness beyond numerical precision.

| | Reduction | $\epsilon_{\text{total}}$ | Speedup | Fails |
|---|---|---|---|---|
| Stiffness = $10^5$ | 10 modes | $(3.7 \pm 1.1)\cdot10^{-2}$ | $20.88 \pm 0.36$ | 0% |
| | 30 modes | $(0.2 \pm 1.5)\cdot10^{-3}$ | $9.83 \pm 0.78$ | 0% |
| | 50 modes | $(0.1 \pm 1.2)\cdot10^{-9}$ | $7.20 \pm 0.55$ | 0% |
| | 60 modes | $(1.0 \pm 2.5)\cdot10^{-11}$ | $7.19 \pm 0.53$ | 0% |
| Stiffness = $10^{12}$ | 10 modes | $(20.66 \pm 7.9)\cdot10^{-3}$ | $20.70 \pm 0.91$ | 0% |
| | 30 modes | $(2.7 \pm 6.7)\cdot10^{-3}$ | $10.1 \pm 1.1$ | 0% |
| | 50 modes | $(0.6 \pm 4.6)\cdot10^{-6}$ | $9.00 \pm 0.78$ | 0% |
| | 60 modes | $(0.2 \pm 1.7)\cdot10^{-4}$ | $9.13 \pm 0.80$ | 0% |

### 5.2.2 Limit of Balanced Truncation

BT produces reliable and efficient ROMs for a stiffness of $10^5$ and $10^{12}$, but for stiffness higher than 64 bit numerical precision, such as $10^{25}$ and $10^{40}$, the method does not work. Although the real system is stable, the 64 bit approximation is not. The Lyapunov equations can not be solved precise enough within the numerical precision, so the controllability and observability Gramians are not stable. Without stable Gramians the balancing transformation can not be performed (Antoulas, 2005, p. 242); the code throws an stability error and exits.

We can call systems with stiffness beyond numerical precision 'pseudo-unstable': they are stable, but appear unstable within our observed precision.

## 5.3 Chapter review

This chapter recasts the burnup problem in terms of control theory, specifically Balanced Truncation (BT). Section 5.1 explains the concepts of BT. Section 5.2.1 compares the performance of BT with that of (heurstically corrected) Proper Orthogonal Decomposition (POD)

from previous chapters, as shown in Figure 5.1 and 5.2. Table 5.1 shows the performance of BT on decay chains of stiffness—$10^5$ and $10^{12}$—lower than numerical precision. Higher stiffness does not work. Based on the findings we can make several conclusions:

1. Balanced Truncation guarantees stability if the Full Order Model (FOM) is stable and the stiffness of the FOM is within numerical precision.
2. The BT Reduced Order Models (ROMs) of 200 nuclide decay chains are at least 90% precise for 10 reduced modes.
3. The precision increases with more reduced modes.
4. The precision of the ROMs is reliable, as the variation is comparable to the error.
5. Less reduced modes increase speed-up, up to 20 times for 200 nuclide decay chains reduced to 10 modes.
6. BT with 64 bit floats is not fit to reduce burnup problems, as the method fails every time.

Compared to (heurstically corrected) POD, BT has advantageous properties. Under the right circumstances, the stability is guaranteed and the ROMs are reliably efficient. However, for burnup problems with a stiffness of $10^{40}$ , the method does not work because the lower numerical precision cause 'pseudo-instabilities'.

The method could work if the numerical precision is increased to a level where the Gramians of the burnup problem can be determined correctly. However, determining the Gramians with increased precision may become too expensive to compute within reasonable time. Additionally, it requires extensive tailoring of the available libraries.

Maybe, just maybe, we can combine the reliability and stability of BT with the computational advantages of POD to have the best of both worlds. The next chapter presents the Balanced Proper Orthogonal Decomposition, a hybrid of the BT and POD.

# Chapter 6

# Balanced Proper Orthogonal Decomposition

Hanging between POD and BT is the Balanced Proper Orthogonal Decomposition (BPOD). The idea is to find a balancing transformation similar to BT, but approximating controllability and reachability matrices with a singular value decomposition of data snapshots (Willcox & Peraire, 2002). BPOD is an intrusive method: full knowledge of the system matrices is required (Taira et al., 2017).

The BPOD guarantees conservation of stability, like BT; the reachability and observability subspaces are not explicitly calculated, so we can circumvent the computational issues of the BT found in the previous chapter.

Similar to BT, BPOD relies on a linear time-invariant system for the nuclide vector $\mathbf{N}(t)$

$$\begin{aligned} \frac{\mathrm{d}\mathbf{N}(t)}{\mathrm{d}t} &= A\mathbf{N}(t) + B\mathbf{u}(t) \\ \mathbf{y}(t) &= C\mathbf{N}(t). \end{aligned} \tag{6.1}$$

Chapter 5 provides the details of each of the system matrices $A, B, C$ and system input $\mathbf{u}(t)$. Contrary to BT, BPOD does not first balance the full system description and then truncate it, but the other way around: the system is truncated and then balanced, which makes it less demanding.

The chapter first introduces the method, ending with a summary of the concrete steps. The second section tests the reduction performance of BPOD on decay chains of stiffness $10^5$, $10^{12}$, $10^{25}$, and $10^{40}$. The third section shows the performance of a 1650 nuclide burnup problem. Finally, the fourth section reviews the chapter and gives recommendations for improvements.

## 6.1 Introduction to Balanced Proper Orthogonal Decomposition

Balanced Proper Orthogonal Decomposition (BPOD) is unique in the way that it uses a limited amount of data to directly generate a truncated balancing transformation. This section explains the differences with previous methods and concludes with a summary on how to perform the BPOD. This section is based on Rowley (2005), which provides a concise and clear explanation of the BPOD.

### 6.1.1 Input impulse response

A system, as described by Equation 6.1, can be characterized by the impulse response to the possible inputs (Brunton & Kutz, 2022, p.370–371). If we limit our inputs, then we need to simulate only a few impulse responses. The relevant inputs are defined by $B$. Therefore, set

$$\mathbf{N}_k(t_0) = \mathbf{b}_k \tag{6.2}$$

into the burnup equation

$$\frac{\mathrm{d}\mathbf{N}_k(t)}{\mathrm{d}t} = A\mathbf{N}_k(t). \tag{6.3}$$

Here, $b_k$ is the $k$th column of the $B$ matrix. In other words, we capture the impulse response of the possible inputs.

For the decay chain, this is a single vector with the initial composition of the chain. For more complex simulations, we could choose for example $B$ to represent different initial fuel mixtures.

### 6.1.2 Output impulse response

Similar to the input impulse response, we can simulate the output impulse response, however, this process is more involved.

First, determine the Proper Orthogonal Decomposition (POD) modes $\phi_m$ of $\{C\mathbf{N}_1 \ \cdots \ C\mathbf{N}_k\}$. Chapter 3 explains the POD modes. Choose a suitable projection rank $r$ for the POD modes $\phi_m$. The output impulse response can be found by putting

$$z_r(0) = C^*\phi_r \tag{6.4}$$

into the adjoint system

$$\frac{\mathrm{d}\mathbf{Z}_r(t)}{\mathrm{d}t} = A^*\mathbf{Z}_r(t). \tag{6.5}$$

The adjoint system must be simulated $r$ times.

### 6.1.3 Data matrices

Define $t_{1:\to T}$ as a concatenation of all time steps between $[t_1,\ t_T]$ multiplied by the square root of the quadrature coefficients $\delta$, i.e. the time step. Using this notation we can write

$$\{\sqrt{\delta_1}\mathbf{N}(t_1) \ \cdots \sqrt{\delta_T}\ \mathbf{N}(t_T)\} = \mathbf{N}(t_{1:\to T}). \tag{6.6}$$

We collect the simulated impulse response data in the matrix

$$X = \{\mathbf{N}_1(t_{1:\to T}) \ \cdots \ \mathbf{N}_k(t_{1:\to T})\} \tag{6.7}$$

and the output response data in the matrix

$$Y = \{\mathbf{Z}_1(t_{1:\to T}) \ \cdots \ \mathbf{Z}_r(t_{1:\to T})\}. \tag{6.8}$$

### 6.1.4 Balanced Proper Orthogonal Decomposition Transformation

The controllability and observability Gramian can be approximated as (Moore, 1981),

$$W_C = XX^*, W_O = YY^*. \tag{6.9}$$

Now, supposen that we take the Singular Value Decomposition (SVD) of $Y^*X$ and truncate the singular values to $R$

$$Y^*X = U\Sigma V^* = \begin{pmatrix} U_R & U_T \end{pmatrix} \begin{pmatrix} \Sigma_R & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V_R^* \\ V_T^* \end{pmatrix} = U_R\Sigma_R V_R^* \tag{6.10}$$

then we define the matrices

$$T_R = XV_R\Sigma_R^{-1/2},\ S_R = \Sigma_R^{-1/2}U_R^*Y^*. \tag{6.11}$$

Rowley (2005) shows when the singular values are not truncated, that $T_R$ and $S_R$ are equal to the balancing transformation and the inverse. The matrices directly provide a truncated balanced transformation.

### 6.1.5 Projection

The projection to the Reduced Order Model (ROM) space is similar to Balanced Truncation (BT), shown in Section 5.1.5,

$$\mathbf{N}(t) = T_R\mathbf{N}_R(t),\ \mathbf{N}_R(t) = S_R\mathbf{N}(t), \tag{6.12}$$

where $\mathbf{N}_R(t)$ is the nuclide vector in the truncated balanced basis. Equation 6.1 can be expressed as (Brunton & Kutz, 2022, p.371–372)

$$\begin{aligned}\frac{\mathrm{d}\mathbf{N}_R(t)}{\mathrm{d}t} = S_RAT_R\mathbf{N}_R(t) + S_RB\mathbf{u}(t) = A_R\mathbf{N}_R(t) + B_R\mathbf{u}(t)\\ \mathbf{y}(t) = CT_R\mathbf{N}_R(t) = C_R\mathbf{N}_R(t).\end{aligned} \tag{6.13}$$

### 6.1.6 Summary

The Balanced Proper Orthogonal Decomposition (BPOD) method can be summarised by the following eight steps:

1. Define system matrices $A, B, C$.
2. Solve the system $\frac{\mathrm{d}\mathbf{N}(t)}{\mathrm{d}t} = A\mathbf{N}(t)$ for every input column of $B$.
3. Compute the Proper Orthogonal Decomposition modes of $\{C\mathbf{N_1}(t), \ldots, C\mathbf{N_k}(t)\}$.
4. Solve adjoint system $\frac{\mathrm{d}\mathbf{Z}(t)}{\mathrm{d}t} = A^*Z(t)$ $r$ times for $z_r(0) = C^*\phi_r$.
5. Form data matrices $X, Y$ with Equation 6.7 and 6.8.
6. Compute the Singular Value Decomposition of $Y^*X$.
7. Determine the BPOD transformation with Equation 6.11.
8. Project and run the model in the reduced space according to Equation 6.13.

Note that only step 6 and 7 perform operations with possibly ill-conditioned matrices, however, this should not pose problems. The balanced space is symmetric, so is the inner product. Symmetric inner products are known to be stable (Tabandeh et al., 2016).

## 6.2 Reduction performance on decay chains

The properties of the Balanced Proper Orthogonal Decomposition (BPOD) mean that the Reduced Order Model (ROM) should be stable, if the Full Order Model (FOM) is stable and the data snapshots $X, Y$ represent the model well. We want to choose a solver with minimal numerical errors for stiff problems. The solver of choice is the Rodas5P method implemented in Julia, which benchmarks as the best within the Rodas family for high accuracy requirements (Steinebach, 2023). Furthermore, Julia offers faster executing times when compared to Python and allows intuitive multithreading.

Choose $A$ to be the decay matrix of 200 nuclides, $B$ a 200 long column vector with the one on the first entry and zeros on all others, and $C$ to be a $200 \times 200$ identity matrix.

The initial state of the system is $x(t_0) = 0$, so we start the simulation by $u(t_0) = \mathbf{N}(t_0)$, after which we set $u(t > 0) = 0$. Due to a partial rewrite to Julia, the steps for measuring performance are

1. Generate a decay chain using Python.
2. Pass the system to Julia.
3. Find the BPOD transformations $T_R$ and $S_R$, using the Rodas5P solver.
4. Pass the transformations to Python.
5. Compare performance FOM with ROM using implicit Euler method in Python.

An excerpt of the Julia and Python script is added in Appendix G.

100 different decay chains with stiffness $10^5$, $10^{12}$, $10^{25}$, and $10^{40}$ have been reduced to 2, 4, 6, 8 and 10 reduction orders, for a total of 2000 reductions. The results are listed in Table 6.1.

The speedup decreases with more modes, but remains a bit above that of Balanced Truncation (BT), as seen in Table 5.1 and is similar to Proper Orthogonal Decomposition (POD), as seen in Table 3.3.

Interesting is the behaviour of the normalised relative total error and failure rate. Contrary to the stability guarantee predicted by the theory, the BPOD has some failures for all levels of stiffness. However, at stiffness lower than numerical precision, $10^{16}$, the failures stop for more modes, suggesting some type of 'collapse'. For stiffness above numerical precision the normalised total error first decreases with more modes before flattening out after 8 modes for stiffness $10^{25}$ and 6 modes for stiffness $10^{40}$ . At the same time the failure rate increases, until also flattening out for a stiffness of $10^{40}$.

The exact cause of the failures and flattening of error is unknown. A quick experiment with similar decay chains, but 256-bit precision for the multiplication and Singular Value Decomposition (SVD) of the data matrices $X, Y$, gives similar results, which suggests that the failures are caused by something else.

We suggest that the failure is caused by the numerical solvers of the decay chain, which is supported by the fact that error and failure rate flatten out. The BPOD method can not approximate the FOM better than allowed by the precision of the data. We could test this hypothesis by using preciser solvers with more snapshots.

For now, we conclude that the BPOD implementation for decay chains can be improved, but the BPOD method can be tried on a real burnup problem, as the failure rate is the lowest for the extreme stiffness of $10^{40}$ compared to (corrected) POD and BT reduction methods.

## 6.3 Reduction of Reactor Burnup

So far the Balanced Proper Orthogonal Decomposition (BPOD) shows promising results in regards to reliability and efficiency for reducing a full burnup problem. Using the Molten Salt Fast Reactor (MSFR) reactor model, described in Section 2.2, as an example, we test BPOD on a burnup problem. We can use the script for the decay chain performance of Appendix G with a Radau solver instead. We use the Radau solver to make a Full Order Model (FOM) simulation, shown in Figure 6.1. We took $A$(t=30 days) to ensure xenon saturation. The simulation shows the conversion of fuel to Fission Products (FP).

Surprisingly, the 64-bit eigenvector decomposition finds positive eigenvalues, although the simulation is stable up until 10 years. The positive eigenvalues are probably negative eigenvalues, which are displaced to the positive plane due to numerical errors. Increasing the timescale of the simulation, or using a eigenvector corresponding to a positive eigenvalue does

**Table 6.1: Decay chain reduction performance of Balanced Proper Orthogonal Decomposition**
For every stiffness and reduced modes, a 100 different decay chains have been reduced using BPOD with 500 snapshots. The values reported are the average, with the standard variation. Fails are defined as $\epsilon_{\text{total}} > 0.95$. Increasing reduced modes generally significantly increases accuracy, however, for a stiffness of $10^{25}$ accuracy increases only up to 8 modes and for stiffness of $10^{40}$ up until 6 modes. The speedup decreases with more modes, as expected. Currently, the failure rate for low orders and low stiffness remains unexplained. For stiffness $10^{25}$ the failure rate increases for more modes. For stiffness $10^{40}$ the failure rate also increases with the modes, but flattens out after 6 modes.

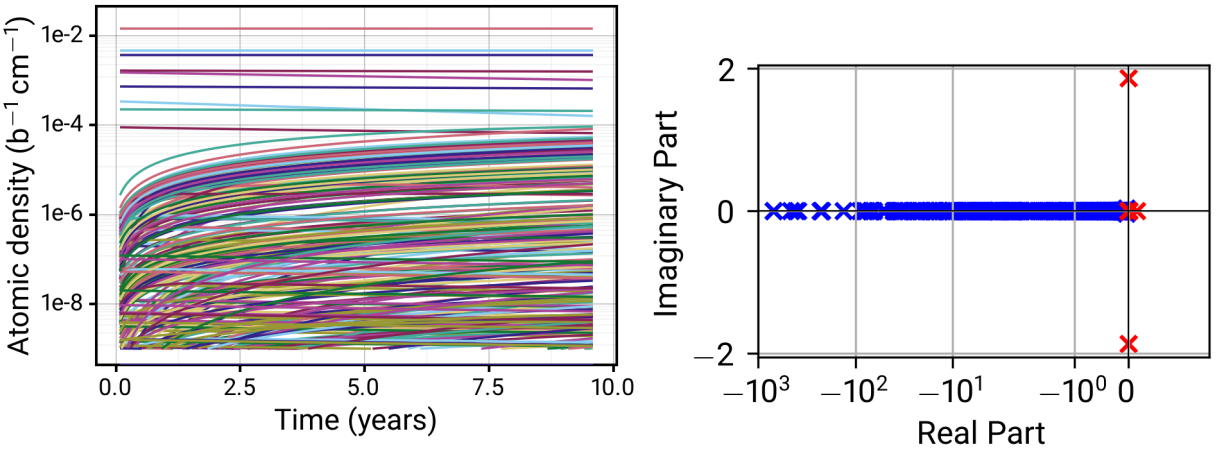
| | Reduction | $\epsilon_{\text{total}}$ | Speedup | Fails |
|---|---|---|---|---|
| Stiffness = $10^5$ | 2 modes | $(373.533 \pm 3.4)\cdot10^{-3}$ | $28.12 \pm 0.63$ | 2% |
| | 4 modes | $(19.28 \pm 2.4)\cdot10^{-2}$ | $28.05 \pm 0.40$ | 10% |
| | 6 modes | $(11.46 \pm 4.5)\cdot10^{-2}$ | $26.89 \pm 0.42$ | 17% |
| | 8 modes | $(7.6 \pm 2.2)\cdot10^{-2}$ | $25.3 \pm 2.2$ | 0% |
| | 10 modes | $(3.7 \pm 1.4)\cdot10^{-2}$ | $22.33 \pm 0.53$ | 0% |
| Stiffness = $10^{12}$ | 2 modes | $(36.05 \pm 5.2)\cdot10^{-2}$ | $28.29 \pm 0.39$ | 2% |
| | 4 modes | $(15.12 \pm 3.5)\cdot10^{-2}$ | $27.58 \pm 0.62$ | 4% |
| | 6 modes | $(8.7 \pm 3.1)\cdot10^{-2}$ | $26.80 \pm 0.54$ | 1% |
| | 8 modes | $(3.9 \pm 1.8)\cdot10^{-2}$ | $24.93 \pm 0.43$ | 0% |
| | 10 modes | $(2.3 \pm 1.1)\cdot10^{-2}$ | $22.24 \pm 0.53$ | 0% |
| Stiffness = $10^{25}$ | 2 modes | $0.38 \pm 0.13$ | $28.21 \pm 0.56$ | 4% |
| | 4 modes | $(12.08 \pm 5.8)\cdot10^{-2}$ | $27.6 \pm 1.2$ | 13% |
| | 6 modes | $0.08 \pm 0.11$ | $26.97 \pm 0.43$ | 23% |
| | 8 modes | $(2.5 \pm 1.6)\cdot10^{-2}$ | $24.98 \pm 0.45$ | 22% |
| | 10 modes | $(2.0 \pm 1.7)\cdot10^{-2}$ | $22.0 \pm 1.4$ | 32% |
| Stiffness = $10^{40}$ | 2 modes | $0.34 \pm 0.12$ | $28.25 \pm 0.45$ | 12% |
| | 4 modes | $(13.2 \pm 9.3)\cdot10^{-2}$ | $28.18 \pm 0.44$ | 20% |
| | 6 modes | $0.07 \pm 0.13$ | $26.81 \pm 0.46$ | 55% |
| | 8 modes | $0.06 \pm 0.13$ | $25.2 \pm 2.1$ | 51% |
| | 10 modes | $(3.3 \pm 9.2)\cdot10^{-2}$ | $22.21 \pm 0.39$ | 49% |

not cause instability. We can thus confidently ascribe the positive eigenvalues to numerical errors, caused by the condition number of the burnup matrix.

The preparatory phase of the BPOD, which is finding the transformation matrices to and from the reduced space, starts with collecting 500 snapshots of the system. The runtime is approximately 10 minutes per mode, as each mode requires collecting 500 more snapshots of the adjoint system. A run with 1000 snapshots did not show any significant improvement over 500 snapshots per simulation, suggesting that the number of snapshots could be lowered further than 500 to decrease data collection runtime.

The BPOD reduces burnup problems efficiently. With as few as 4 reduced modes based on 500 snapshots, the total normalised relative error and the maximum error for any nuclide which reaches an atomic density of $10^{-9}$ ($b^{-1}cm^{-1}$, is below $10^{-5}$, as Figure 6.2 shows. The figure also shows that the total error and maximum nuclide error are near identical, which is confirmed by inspecting the accuracy of the FP, as seen in Figure 6.3.

There were no failed reductions within the dataset, so BPOD seems reliable for burnup problems. Probably because the change in neutron density is generally much slower than for decay chains, which allows the Rodas5P to solve accurately.




**(a)** Simulation **(b)** Eigenvalues

**Figure 6.1: Full order simulation of MSFR burnup**
The burnup matrix $A$(t = 30 days) is simulated to $t = 3 \cdot 10^8$s $\approx$ 10 years. The simulation of Figure (a) shows the steady decrease of fissionable nuclides and the production of many Fission Products. A zoom of the eigenvalue plot is shown in Figure (b). Most eigenvalues are negative, but some are positive (indicated with red markers). These should also be negative, but the numerical error of the eigenvector decomposition moves the eigenvalues.

Compared to other methods that approximate the burnup matrix, the BPOD ROM performs extremely well. The maximum nuclide error is a factor 1000 lower and the runtime a 1000 times faster than alternative methods reported by Isotalo and Aarnio (2011)!

### 6.3.1 Caveats and improvements

The current implementation of BPOD has some caveats that show room for improvement. The implementation can be improved in reliability, applicability and accuracy.

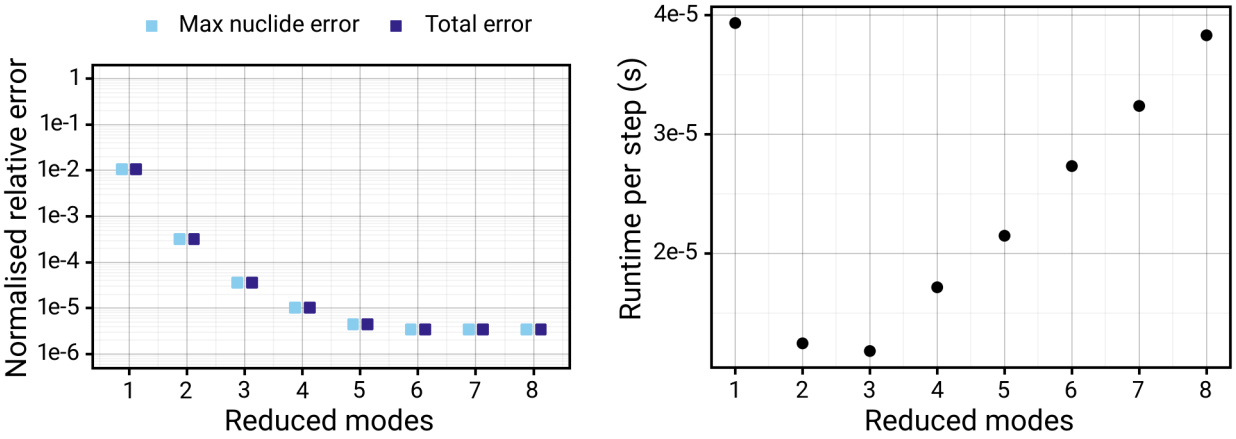
Although the method seems reliable for reactor burnup problems, this should be further investigated, as the decay chain model showed instabilities. We suggest implementing the method in one language, reducing the chance for unexpected behaviour of the code. Between Python and Julia, we would choose Julia, which offers more control over the numerical differential equation solver and executes at a higher speed.

The current model has been trained on a single initial fuel mixture. We could generalise the method by encoding information of different fuel mixtures in the system response matrix $B$, adding columns for different initial mixtures. Training $i$-mixtures would extend the preparatory phase of the BPOD by a factor $i$ as the whole preparatory phase up until the data matrices $X, Y$ needs to be repeated.

Furthermore, the current model assumes a linear time-invariant burnup matrix $A$, while in reality it depends on both the nuclide composition $\mathbf{N}(t)$ and the neutron flux $\phi(t)$. Similar to Bouma (2024, p.29–38) we could split the burnup matrix into a constant part and variable part. Leveraging the system description of Equation 6.1, we could write

$$\begin{aligned} \frac{\mathrm{d}\mathbf{N}(t)}{\mathrm{d}t} &= A_1\mathbf{N}(t) + A_2\mathbf{u}(\phi(t), \mathbf{N}(t)) \\ \mathbf{y}(t) &= C\mathbf{N}(t). \end{aligned} \tag{6.14}$$

Looking at the properties of the burnup matrix, as described in Section 2.3.1, we should choose $A_1$ as the decay part of the burnup matrix concerning decay. $A_2$ should then represent the response of the system to every nuclide, which approximately is the part of the burnup




**(a)** Accuracy BPOD

**(b)** Speed BPOD

**Figure 6.2: Performance of Balanced Proper Orthogonal Decomposition for t = $3 \cdot 10^8$**
The BPOD is based on 500 snapshots generated with Rodas5P. Figure (a) shows the decrease in error between the Reduced Order Model (ROM) and FOM, which decreases up until 6 modes. The maximum individual nuclide error—maximum nuclide error—for nuclides which reach an atomic density of more than $10^{-9}$ $b^{-1}cm^{-1}$ is compared to the total normalised relative error; the difference is minimal. The ROM ran for 1000 steps. The runtime per simulation step, shown in Figure (b), inexplicably first decreases, before increasing with more modes, but generally is in the order of $10^{-5}$ seconds.

matrix concerning reactions. The input vector $u(\phi(t), N(t)) \approx \phi(t)\mathbf{N}(t)$ can then be used to simulate changes the flux and nuclide composition. It could even be possible to choose $A_2$ in such a way that it encodes fuel mixture information. If so, then a single transformation can be found to reduce general burnup problems. There are two downsides: the preparatory phase could significantly increase, as many simulations are needed to get data on all possible inputs, and the ROM should probably include more modes than four to remain accurate for the full parameter space, which would come at the cost of speedup.

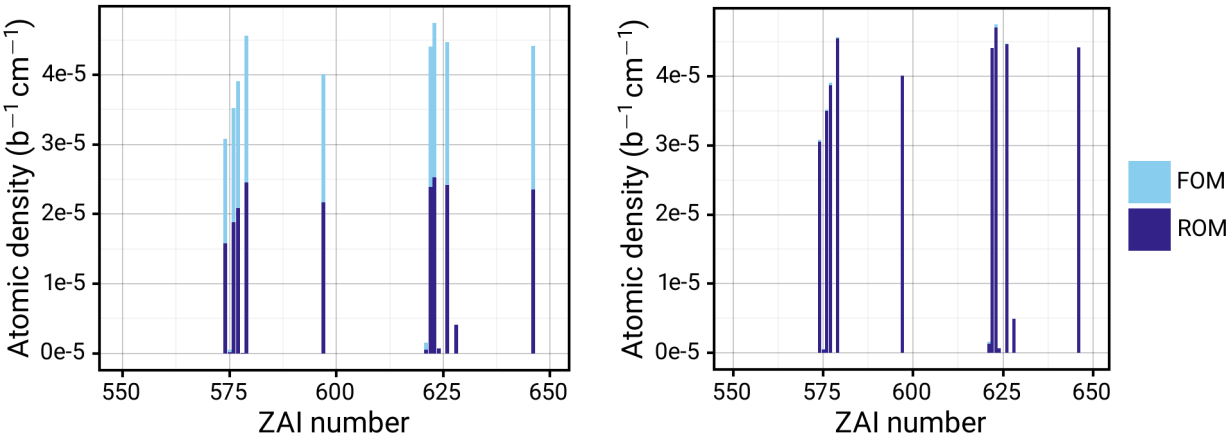
## 6.4 Chapter review

The chapter starts by introducing the Balanced Proper Orthogonal Decomposition (BPOD), which leverages simulated impulse responses on the (adjoint) system to gather data matrices. The data matrices can be used to directly find a truncated balancing transformation, skipping directly calculating the system Gramians. The method is summarised in Section 6.1.6.

Section 6.2 shows results of the BPOD on decay chains, which give mixed results. The speedup is similar to that of Proper Orthogonal Decomposition (POD), but accuracy and stability for decay chains are lacking; an explanation is lacking, especially when looking at the results for the full burnup problem of Section 6.3, which is stable and accurate. Currently, the hypothesis for the instability is that the BPOD stability depends on the accuracy of the dataset, which could be too low for the decay chains.

We conclude:

1. BPOD achieves similar speedup to POD.
2. A few BPOD modes create an accurate Reduced Order Model (ROM) of nuclear burnup—4 modes give a normalised relative error of $10^{-5}$.
3. BPOD can create a unstable ROM from a stable Full Order Model (FOM), although an explanation for this is lacking.




**(a)** One BPOD mode

**(b)** Two BPOD modes

**Figure 6.3: Zoom in of a subset of Fission Products at t = $3{\cdot}10^8$ for one and two modes**
A bar plot the FP atomic density confirms that no individual nuclide performs worse than others. The BPOD reduces all nuclides with the same precision. However, the error at the end of the simulation is clearly higher than the total normalised error reveals for one mode, as seen in Figure (a). A small error at the beginning can grow larger. Adding a second mode, as seen in Figure (b) fixes resolves the problem.

4. The right choice of solver for the burnup equation is crucial for BPOD.
5. Collecting the datasets for the BPOD takes a significant amount of simulation time, which can still be optimised.
6. The total error of the model is equivalent to the individual nuclide error; no nuclide is reduced worse than another.
7. BPOD is a promising technique for reducing nuclear burnup problems: the speedup and accuracy is a factor 1000 better than reported alternatives (Isotalo & Aarnio, 2011).

Section 6.3.1 discusses three possible improvements. The first is to make a pure Julia implementation for the burnup problem; the second is to modify the system matrix $B$ to include more fuel mixtures in the ROM; the third is to modify the system according to Equation 6.14 to include non-linear behaviour of burnup.

Furthermore, the differential solving method should be investigated. The solver can cause inaccuracies and takes up the most significant portion of determining the BPOD. The current implementation uses constant time-stepping, but variable time-stepping should increase the accuracy of the datasets, which is possible according to Equation 6.7. It could even be possible to implement another approximation method for generating the BPOD datasets. For example, the Chebyshev rational approximation method of Isotalo and Aarnio (2011).

# Chapter 7

# Conclusions

The fuel mixture in nuclear reactors is constantly changing, due to decay of unstable nuclides and nuclear reactions with the free neutrons in the reactor core. The burnup equation, governing the fuel mixture change, is a set of coupled ordinary first order differential equations.

Solving the burnup equation would not pose a problem if it were not for the size of the system. The fuel of the Molten Salt Fast Reactor (MSFR), a very promising new reactor type, such as described in Chapter 2, contains 1650 different nuclides during operation. The matrix $A$ is of size $1650 \times 1650$, with entries ranging from $10^{-20}$ $s^{-1}$ to $10^{20}$ $s^{-1}$. Directly solving the differential equation can take a long time and is not recommended.

Reduced Order Modelling (ROM), a technique that reduces the order of a system while keeping accuracy loss to a minimum, can speed up burnup simulations. Four different methods have been investigated and compared. The stiffness, determined by the fraction of the largest and smallest value in the burnup matrix, is $10^{40}$. Such extreme stiffness causes numerical issues, so we introduce a simplified verion of the burnup problem, that of an arbitrary decay chain: a chain of nuclides decaying into one another. The stiffness can be tuned.

Chapter 3 focusses on the Proper Orthogonal Decomposition (POD), which is easy to implement and can a huge speedup. The idea is to generate snapshots of the model and use the correlation in the snapshots to find a transformation to a lower order basis of the most significant correlations. However, testing the POD on 200 nuclide long decay chains of stiffness $10^5$, $10^{12}$, $10^{25}$, and $10^{40}$, shows increasing instabilities; the normalised relative error of the Reduced Order Model (ROM) compared with the Full Order Model (FOM) is $\epsilon > 0.95$ in these cases. The eigenvalues of the ROM are either spread too far from each other, causing quick decay of the initial condition, or the eigenvalues are moved into the possitive plane, causing exponentially growing solutions. As the amount of unstable models is above 90% for stiffness $10^{40}$, we conclude that POD is not suited for nuclear burnup.

Chapter 4 modifies the POD algorithm by heuristically searching and replacing the eigenvalues that cause the unstable models. Add sufficient modes to prevent the spread of eigenvalues and shift or reassign the positive eigenvalues to the negative plane to avoid exponential growth. This correction strategy proves too simplistic, as the stability issues are for stiffness of $10^{25}$ and $10^{40}$ are worse. About 98% of the models fail to achieve an normalised relative total error of $\epsilon < 0.95$.

Chapter 5 implements a method taken from control theory: Balanced Truncation (BT). The method aims to find a transformation that balances the controllability and observability of the system with using the system description, i.e. no data is required. The first is the system states that can be reached with the possible inputs and the second are the states that can be observed with the measuring devices. The balancing transformation transforms

the system to an ordered space according to the expected observed abundance of system states. Truncate the least abundant states to create a ROM. BT produces accurate results without any unstable models, but there is a shortcoming. The controllability and observability Gramians, crucial matrices for calculating the balancing transformation, cannot be determined if the stiffness of the problem goes beyond the $10^{16}$ Float 64 precision. None of the decay chains with stiffness of $10^{25}$ and $10^{40}$ could be reduced.

Chapter 6 introduces Balanced Proper Orthogonal Decomposition (BPOD), a midway technique between POD and BT. The idea is to perform a balancing transformation but use the POD to directly calculate the truncated basis transformation from data snapshots. The method therefore does not determine the controllability and observability Gramian, allowing application to any stiffness. Furthermore, it reduces the system in the symmetric balanced space, which is beneficial for numerical stability. BPOD inexplicably can fail for all levels of stiffness decay chains, it does performs better with regard to stability than the previous methods for the highest stiffness: creating stable Reduced Order Models (ROMs) for about 50% of decay chains with stiffness $10^{40}$.

Of the four techniques, only the BPOD, is applied to a full burnup problem of the MSFR. Within the limits of a single initial fuel mixture and the assumption $A(\mathbf{N}(t), \phi(t)) \approx A$, the ROMs perform well. For a ROM of 4 orders the runtime per burnup step is around $10^{-5}$ s with a total and nuclide specific normalised relative error below $10^{-5}$. Both results are by a factor 1000 better than the best burnup methods reported by Isotalo and Aarnio (2011)

## Recommendations

We advise against POD for reducing burnup equations. The technique does not work for stiff burnup problems. Even with some, although heuristic, corrections, the method cannot reliably produce ROMs. Besides, there are more efficient reduction methods.

BT is a useful technique, currently limited by the Float 64 precision of Python. A implementation with higher floating point precision, in for example Julia, should work for burnup problems. As BT does not require data snapshots, it could be quick to set up and create ROMs for different scenarios. This can be worth pursuing.

BPOD is the most promosing. The implementation of this thesis already reduces burnup problems with succes. Incorporating a range of initial fuel mixtures requires small modifications and the accuracy/stability can be improved with a more accurate solver using variable time stepping. Furthermore, it could be possible to modify the system according to Section 6.3.1, so it incorporates the non-linearity of the burnup matrix. If done correctly, this could open the door to using BPOD to speedup general burnup simulations, independent of intial fuel mixture and neutron flux.

# Bibliography


Alsayyari, F., Perkó, Z., Tiberga, M., Kloosterman, J. L., & Lathouwers, D. (2021). A fully adaptive nonintrusive reduced-order modelling approach for parametrized time-dependent problems. *Computer Methods in Applied Mechanics and Engineering*, *373*, 113483.

Antoulas, A. C. (2005). *Approximation of large-scale dynamical systems*. Society for Industrial and Applied Mathematics.
OCLC: ocm58789279.

Aufiero, M., Cammi, A., Fiorina, C., Leppänen, J., Luzzi, L., & Ricotti, M. (2013). An extended version of the SERPENT-2 code to investigate fuel burn-up and core material evolution of the Molten Salt Fast Reactor. *Journal of Nuclear Materials*, *441*(1-3), 473–486.

Bouma, J. F. (2024, January). *Reduced Order Modelling of the Burnup Analysis of a Chloride-based MSFR* [Doctoral dissertation, TU Delft]. https://repository.tudelft.nl/

Brunton, S. L., & Kutz, J. N. (2022). *Data-Driven Science and Engineering : Machine Learning, Dynamical Systems, and Control* (Second Edition). Cambridge University Press.

Butcher, J. C. (2003). *Numerical methods for ordinary differential equations* (1st Edition). Wiley.

Castagna, C., Aufiero, M., Lorenzi, S., Lomonaco, G., & Cammi, A. (2020). Development of a Reduced Order Model for Fuel Burnup Analysis. *Energies*, *13*(4), 890.

Cheney, E. W., & Kincaid, D. (2007). *Numerical mathematics and computing* (6th ed). Brooks/Cole.

CNRS. (2013, November). Final Report - EVOL (Evaluation and Viability of Liquid Fuel Fast Reactor System). Retrieved June 20, 2024, from https://cordis.europa.eu/docs/results/249/249696/final1-final-report-f.pdf

Cordier, L., & Bergmann, M. (2008). Proper Orthogonal Decomposition: An overview. In *Lecture series 2002-04, 2003-03 and 2008-01 on post-processing of experimental and numerical data, Von Karman Institute for Fluid Dynamics, 2008*. (46 pages). VKI. https://hal.science/hal-00417819

Croff, A. G. (1983). ORIGEN2: A Versatile Computer Code for Calculating the Nuclide Compositions and Characteristics of Nuclear Materials. *Nuclear Technology*, *62*(3), 335–352.

De Marcillac, P., Coron, N., Dambier, G., Leblanc, J., & Moalic, J.-P. (2003). Experimental detection of $\alpha$-particles from the radioactive decay of natural bismuth. *Nature*, *422*(6934), 876–878.

Duderstadt, J. J., & Hamilton, L. J. (1976). *Nuclear reactor analysis*. Wiley.

Foro Nuclear. (2024, March). Refueling outages in nuclear power plants. Retrieved June 20, 2024, from https://www.foronuclear.org/en/updates/in-depth/refueling-outages-in-nuclear-power-plants/

Golub, G. H., & Van Loan, C. F. (2013). *Matrix computations* (Fourth edition). The Johns Hopkins University Press.
OCLC: ocn824733531.

Guo, S., Zhang, J., Wu, W., & Zhou, W. (2018). Corrosion in the molten fluoride and chloride salts and materials development for nuclear applications. *Progress in Materials Science*, *97*, 448–487.

Hairer, E., & Wanner, G. (1996). *Solving Ordinary Differential Equations II* (Vol. 14). Springer Berlin Heidelberg.

Horn, R. A., & Johnson, C. R. (2012). *Matrix analysis* (2nd ed). Cambridge University Press.

Iberdrola corp. (2019, December). Núñez de Balboa completed: Iberdrola finalizes the construction of the largest photovoltaic plant in Europe within one year. Retrieved June 12, 2024, from https://www.iberdrola.com/press-room/news/detail/nunez-balboa-completed-iberdrola-finalizes-construction-largest-photovoltaic-plant-europe-within-year

Igini, M. (2023, January). The Advantages and Disadvantages of Nuclear Energy. Retrieved June 12, 2024, from https://earth.org/the-advantages-and-disadvantages-of-nuclear-energy/

Isotalo, A., & Aarnio, P. (2011). Comparison of depletion algorithms for large systems of nuclides. *Annals of Nuclear Energy*, *38*(2-3), 261–268.

Kondev, F., Wang, M., Huang, W., Naimi, S., & Audi, G. (2021). The NUBASE2020 evaluation of nuclear physics properties. *Chinese Physics C*, *45*(3), 030001.

Lassila, T., Manzoni, A., Quarteroni, A., & Rozza, G. (2014). Model Order Reduction in Fluid Dynamics: Challenges and Perspectives. In A. Quarteroni & G. Rozza (Eds.), *Reduced Order Methods for Modeling and Computational Reduction* (pp. 235–273). Springer International Publishing.

Leppänen, J., Pusa, M., Viitanen, T., Valtavirta, V., & Kaltiaisenaho, T. (2015). The Serpent Monte Carlo code: Status, development and applications in 2013. *Annals of Nuclear Energy*, *82*, 142–150.

Mesthiviers, L. (2022, December). *Capacités de conversion des transuraniens en Réacteurs à Sels Fondus (RSF)* [Theses]. Université Grenoble Alpes. https://theses.hal.science/tel-04048655

MIMOSA. (2024, January). MultI-recycling strategies of LWR SNF focusing on MOlten SAlt technology. Retrieved June 12, 2024, from https://www.mimosa-euratom.eu/

Moore, B. (1981). Principal component analysis in linear systems: Controllability, observability, and model reduction. *IEEE Transactions on Automatic Control*, *26*(1), 17–32.

National Nuclear Data Center. (2023, November). Evaluated Nuclear Structure Data File. Retrieved February 26, 2024, from https://www-nds.iaea.org/relnsd/NdsEnsdf/QueryForm.html

Nicoguaro. (2016, February). GaussianScatterPCA. Retrieved February 2, 2024, from https://commons.wikimedia.org/wiki/File:GaussianScatterPCA.svg

Nuclear Energy Agency. (2017, November). Joint Evaluated Fission and Fusion File (JEFF). https://www.oecd-nea.org/dbdata/jeff/jeff33/

N.V. EPZ. (2023). EPZ jaarverslag 2022. Retrieved June 12, 2024, from https://www.epz.nl/wie-wij-zijn/publicaties/

Paul, S., Rajan, A., Chang, J., Kuang, Y. C., & Ooi, M. P.-L. (2018). Parametric Design Analysis of Magnetic Sensor Based on Model Order Reduction and Reliability-Based Design Optimization. *IEEE Transactions on Magnetics*, *54*(3), 1–4.

Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, *2*(11), 559–572.

Quarteroni, A., & Rozza, G. (Eds.). (2014). *Reduced Order Methods for Modeling and Computational Reduction*. Springer International Publishing.

Rathinam, M., & Petzold, L. R. (2003). A New Look at Proper Orthogonal Decomposition. *SIAM Journal on Numerical Analysis*, *41*(5), 1893–1925.

Redmann, M., & Duff, I. P. (2022). Full state approximation by Galerkin projection reduced order models for stochastic and bilinear systems. *Applied Mathematics and Computation*, *420*, 126561.

Rezaian, E., & Wei, M. (2018). Eigenvalue Reassignment by Particle Swarm Optimization toward Stability and Accuracy in Nonlinear Reduced-Order Models. *2018 Fluid Dynamics Conference*.

Rezaian, E., & Wei, M. (2020). Impact of Symmetrization on the Robustness of POD-Galerkin ROMs for Compressible Flows. *AIAA Scitech 2020 Forum*.

Rowley, C. W. (2005). MODEL REDUCTION FOR FLUIDS, USING BALANCED PROPER ORTHOGONAL DECOMPOSITION. *International Journal of Bifurcation and Chaos*, *15*(3).

Scheepmaker, T. (2024). *A Predictive Non-Intrusive Reduced Order Model of a High-Temperature Gas-Cooled Reactor using Operator Inference* [Doctoral dissertation, TU Delft].

Serp, J., Allibert, M., Beneš, O., Delpech, S., Feynberg, O., Ghetta, V., Heuer, D., Holcomb, D., Ignatiev, V., Kloosterman, J. L., Luzzi, L., Merle-Lucotte, E., Uhlíř, J., Yoshioka, R., & Zhimin, D. (2014). The molten salt reactor (MSR) in generation IV: Overview and perspectives. *Progress in Nuclear Energy*, *77*, 308–319.

Serpent Wiki. (2020, December). Definitions, units and constants. Retrieved April 12, 2024, from https://serpent.vtt.fi/mediawiki/index.php/Definitions,_units_and_constants

Steinebach, G. (2023). Construction of Rosenbrock–Wanner method Rodas5P and numerical benchmarks within the Julia Differential Equations package. *BIT Numerical Mathematics*, *63*(2), 27.

Steve Brunton. (2017, January). Control Bootcamp: Overview. Retrieved July 21, 2024, from https://www.youtube.com/watch?v=Pi7l8mMjYVE

Tabandeh, M., Wei, M., & Collins, J. P. (2016). On the Symmetrization in POD-Galerkin Model for Linearized Compressible Flows. *54th AIAA Aerospace Sciences Meeting*.

Taira, K., Brunton, S. L., Dawson, S. T. M., Rowley, C. W., Colonius, T., McKeon, B. J., Schmidt, O. T., Gordeyev, S., Theofilis, V., & Ukeiley, L. S. (2017). Modal Analysis of Fluid Flows: An Overview. *AIAA Journal*, *55*(12), 4013–4041.

Tiberga, M., Shafer, D., Lathouwers, D., Rohde, M., & Kloosterman, J. L. (2019). Preliminary investigation on the melting behavior of a freeze-valve for the Molten Salt Fast Reactor. *Annals of Nuclear Energy*, *132*, 544–554.

Tosoka. (2014, June). Decay chain(4n+2, Uranium series) - Wikipedia. Retrieved February 26, 2024, from https://commons.wikimedia.org/wiki/File:Decay_chain(4n%2B2,_Uranium_series).svg

Weiss, J. (2019). A Tutorial on the Proper Orthogonal Decomposition. *AIAA Aviation 2019 Forum*.

Willcox, K., & Peraire, J. (2002). Balanced Model Reduction via the Proper Orthogonal Decomposition. *AIAA Journal*, *40*(11), 2323–2330.

World Nuclear Association. (2022, January). Radioactive Waste Management. Retrieved June 12, 2024, from https://world-nuclear.org/information-library/nuclear-fuel-cycle/nuclear-waste/radioactive-waste-management

World Nuclear Association. (2024, April). Chernobyl Accident 1986. Retrieved June 12, 2024, from https://world-nuclear.org/information-library/safety-and-security/safety-of-plants/chernobyl-accident

# Appendix A

# Recycling

A well designed recycling process conserves the amount of particles and chemical balance of the fuel mixture; there is a constant stream of old fuel being replaced with fresh fuel. The specifics of a real recycling process are still unclear, as no setup has yet been build. However, we can include a theoretical recycling process to our burnup calculations by adding terms to the burnup matrix.

Previous research by Aufiero et al. (2013) has defined an extended burnup code to include recycling. First, add a term representing the effective removal times of recycled chemical specie. Effective removal times are quite similar to decay, as chemical processes have different separation efficiencies for different specie; every cycle there is certain fraction separated. For example after a recycling for one year the fraction of a certain nuclide has halved (assuming no production of that nuclide), then we can define a effective 'half-removal' time of one year. Recycling can thus be modelled like decay with an effective removal times instead of decay times. Recycling adds terms to the burnup matrix for nuclide amount $N_i$

$$\frac{\mathrm{d}N_i}{\mathrm{d}t} = -N_i \lambda_{i,\mathrm{repro}}. \tag{A.1}$$

the $\lambda_{i,\mathrm{repro}}$ contains the effective reprocessing time for the recycled nuclides $N_i$.

To keep the chemical composition constant there are two additional processes. Each time a heavy metal nuclide (denoted as HM) fissions it needs te be replaced

$$\frac{\mathrm{d}N_i}{\mathrm{d}t} = c_i \sum_{k=\mathrm{HM}} N_k \phi \cdot \sigma_{k,f} \tag{A.2}$$

with fresh fuel feed, defined by the atomic fraction $c_i$ of the fuel feed. Furthermore the anions of the fuel salt (natrium) need to be taken out when new atoms are created by fission, but added again when fission products (denoted FP) are taken out by the recycling process

$$\frac{\mathrm{d}N_{\mathrm{Na}}}{\mathrm{d}t} = -\sum_{k=\mathrm{HM}} N_j \phi \cdot \sigma_{k,f} \cdot \left( \sum_{l=\mathrm{FP}} \mathrm{FY}_{k\to l} \right) + \sum_{l=\mathrm{FP}} N_l \lambda_{l,\mathrm{repro}} \tag{A.3}$$

$\mathrm{FY}_{k\to l}$ is the fission yield of fission product $l$ out of fissioning heavy metal nuclide $k$.

Recycling adds negative diagonal elements to the burnup matrix, associated with the recycled nuclides. Furthermore, the process adds some positives on the off diagonals associated with fresh fuel and fresh anions. For a well-designed recycling process the negatives elements should outweigh the positives. In an optimal scenario the sum of the recycling part of the burnup matrix would be as much negative as the decay and reaction part are positive; the amount of particles in the reactor core is conserved.

# Appendix B

# Decay chain class

The full code is available on https://github.com/JonathanPilgram/BurnupROM-PythonPackage.

```
import numpy as np
import time
import scipy
import matplotlib.pyplot as plt
import math

class DecayChain:
    """Class to simulate the decay process of a single chain of
       nuclides.

    Parameters
    ----------
    N_nuclides : int
        Amount of nuclides to simulate
    info : bool
        Controls whether to print extra information like plots and
            simulation time
    full_info : bool
        Shows even more information like snapshots and full
           solution

    Internal data
    -------------
    initial_composition : numpy zero array of size N_nuclides
        Used to set the initial composition of the nuclides
    decay_array : numpy zero array of size N_nuclides x N_nuclides
        The decay array used for the chain, starts as zeros array
           of size

    Initializes
    -----------
    solution : None
        The solution of the full model
    decay_array_red : None
```

```
    Comes from reduce_model.py
initial_composition_red : None
    Comes from reduce_model.py
rom_basis : None
    Comes from reduce_model.py
simulation_time_rom : None
    Added by reduce_model.py
decay_constants : None
    Added by build_decay_chain()
"""

def __init__(
    self,
    N_nuclides: int,
    info: bool = False,
    full_info: bool = False,
) -> None:
    self.N_nuclides = N_nuclides
    self.info = info
    self.full_info = full_info
    self.initial_composition = np.zeros(self.N_nuclides)
    self.decay_array = np.zeros((self.N_nuclides, self.
       N_nuclides))
    self.decay_array_red = None
    self.initial_composition_red = None
    self.rom_basis = None
    self.solut = None

def _bateman_equation(self, t, N) -> list:
    """Private function: Returns the coupled bateman equations
        for use by scipy"""
    dNdt = np.zeros_like(N)
    for varying_nuclide in range(self.decay_array_temp.shape
       [0]):
        for nuclide in range(self.decay_array_temp.shape[1]):
            dNdt[varying_nuclide] += (
                self.decay_array_temp[varying_nuclide, nuclide
                   ] * N[nuclide]
            )
    return dNdt

def _run_simulation(
    self, decay_array: np.array, initial_composition: np.array
       , solver: str
):
    """Private function: wrapper for the scipy integrator"""
    self.decay_array_temp = decay_array

    return scipy.integrate.solve_ivp(
        self._bateman_equation,
```

```
            [0, self.sim_time],
            initial_composition,
            t_eval=np.linspace(0, self.sim_time, self.sim_steps +
                1),
            method=solver,
        )

    def run_simulation(
        self, sim_time: int, sim_steps: int, solver: str = "Radau"
    ) -> None:
        """Runs the simulation using the solver of choice, default
            is Radau solver
        Other solvers are:
        'Euler_backward', 'Euler_forward', 'RK45'

        Saves the result in object as solution:
        solution.t - The list with evaluation times
        solution.y - 2D list with solutions for each nuclide

        paramters
        ---------
        sim_time : int
            The simulation time in seconds, saved to object
        sim_steps : int
            Amount of timesteps to simulate, saved to object
        solver : str (default: Radau)
            Solver of choice. All scipy solvers available and
                Euler_forward, Euler_backward, saved to object

        data saved
        ----------
        simulation_time_fom : float
            Simulation time for the full order solution in seconds
        """
        self.sim_time = sim_time
        self.sim_steps = sim_steps
        self.solver = solver
        if np.all(self.initial_composition == 0):
            print("Did you forget to set initial_composition?")

        start = time.time()

        if solver == "Euler_backward":
            solution = self._backward_euler(self.decay_array, self
                .initial_composition)
        else:
            solution = self._run_simulation(
                self.decay_array, self.initial_composition, solver
            )
```

```
        self.simulation_time_fom = time.time() - start
        if self.info == True:
            print("Simulation time:" + str(self.
                simulation_time_fom))
            self._plot_simulation(solution)
        if self.full_info == True:
            print(solution)
        self.solution = solution

    def _plot_simulation(self, solution) -> None:
        """Shows a plot of the decay simulation for debugging
           purposes"""
        N_nuclides = solution.y.shape[0]
        for i in range(N_nuclides):
            plt.plot(solution.t, solution.y[i], label=str(i))
        if N_nuclides < 15:
            plt.legend()
        plt.xlabel("Time (s)")
        plt.ylabel("Nuclide amount")
        plt.title("Full order decay chain with " + str(N_nuclides)
            + " nuclides")
        plt.show()

    def _backward_euler(self, decay_array: np.ndarray,
       initial_composition: np.ndarray):
        """Private function to run the simulation with first order
            backward euler

        parameters
        ----------
        Decay array: np.ndarray
            The decay array
        initial_composition: np.ndarray
            The initial composition

        Return
        ------
        solution
            Array containing the solution values"""
        nuclides_considered = np.shape(decay_array)[0]
        result = np.zeros(
            (nuclides_considered, self.sim_steps + 1)
        )
        result[:, 0] = initial_composition
        sim_times = np.linspace(0, self.sim_time, self.sim_steps +
            1)
        x_old = result[:, 0]
        K_iter = (
            -decay_array
```

```
        + np.diag(np.ones(nuclides_considered)) * self.
            sim_steps / self.sim_time
    )
    for step in range(self.sim_steps):
        rhs = x_old * self.sim_steps / self.sim_time
        result[:, step + 1] = np.linalg.solve(K_iter, rhs)
        x_old = result[:, step + 1]

    solution = scipy.optimize.OptimizeResult(t=sim_times, y=
        result)
    return solution
```

# Appendix C

# Default simulation paramaters decay chain models

Figures and tables in the thesis use roughly the same settings for the decay chain simulation. If not explicitely mentioned otherwise, the settings mentioned in this chapter are used.

### Decay constants

If a range of decay constants is mentioned, such as $\lambda \in [a, b]$, then it means that the model is initialized with logarithmically uniform random decay constants picked from that range, as described by Equation 2.4. The default decay chain terminates with a stable nuclide; the decay constant is set to zero.

Every rerun generates new random decay constants with the same settings, i.e. from the same range.

### Simulation parameters

The default end-time is determined by

$$T = \sum_{i=1}^{n} \frac{2}{\lambda_i} \approx 2\lambda_{\text{total}}, \tag{C.1}$$

which ensures that whole chain decayed. The default simulation is done with 10000 simulation steps using implicit Euler method.

# Appendix D

# Excerpt of Proper Orthogonal Decomposition code

The full code is available on: https://github.com/JonathanPilgram/BurnupROM-ScriptsAndData.

```
import rom
import numpy as np

# Function to compare FOM with POD reduction
def POD(N_rom, N_snaps, stiffness):
    model = rom.DecayChain(N_nuclides)
    rom.build_decay_chain(model, 1, 10**stiffness)
    model.initial_composition[0] = 1
    model.run_simulation(
        np.sum(2 / model.decay_constants[:-1]), N_steps, "
           Euler_backward"
    )
    rom.spatial_snapshots(model, N_snaps)
    rom.reduce_SVD(model, N_rom)
    rom.run_simulation_reduced(model)
    TE = rom.return_error(model)
    SE = rom.return_error(model, range(N_nuclides))
    speedup = model.simulation_time_fom / model.
       simulation_time_rom
    return TE, SE, speedup
```

# Appendix E

# Excerpt of Heuristically Corrected POD code

The full code is available on: https://github.com/JonathanPilgram/BurnupROM-ScriptsAndData.

```
import rom
import numpy as np

# Compares FOM with corrected POD
def POD(N_rom, N_snaps, stiffness):
    shift = 0
    reassign = 0
    model = rom.DecayChain(N_nuclides)
    rom.build_decay_chain(model, 1, 10**stiffness)
    model.initial_composition[0] = 1
    model.run_simulation(
        np.sum(2 / model.decay_constants[:-1]), N_steps, "
           Euler_backward"
    )
    rom.spatial_snapshots(model, N_snaps)
    rom.reduce_SVD(model, N_rom)
    positive_eigs, _ = rom.analyze_eigenvalues(model.
       decay_array_red)
    # Stabililastion method
    if len(positive_eigs) > 0:
        model.decay_array_red -= np.identity(N_rom) * 1e-8
        shift = 1
        eigval, eigvec = np.linalg.eig(model.decay_array_red)
        if len(eigval[eigval > 0]) > 0:
            eigval[eigval > 0] = (
                -eigval[eigval > 0].real + 1.0j * eigval[eigval >
                   0].imag
            )
            model.decay_array_red = np.real(
                eigvec @ np.diag(eigval) @ np.linalg.inv(eigvec)
            )
            reassign = 1
```

```
rom.run_simulation_reduced(model)
TE = rom.return_error(model)
SE = rom.return_error(model, range(N_nuclides))
speedup = model.simulation_time_fom / model.
   simulation_time_rom
return TE, SE, speedup, shift, reassign
```

# Appendix F

# Excerpt of Balanced Truncation code

The full code is available on: https://github.com/JonathanPilgram/BurnupROM-ScriptsAndData.

```
import rom
import numpy as np
import control as ct

# Compares FOM with BT
def BT(N_rom, N_snaps, stiffness):
    model = rom.DecayChain(N_nuclides)
    rom.build_decay_chain(model, 1, 10**stiffness)
    model.initial_composition[0] = 1
    model.run_simulation(
        np.sum(2 / model.decay_constants[:-1]), N_steps, "
           Euler_backward"
    )
    rom.reduce_abstract(model, N_rom)
    A = model.decay_array
    B = np.zeros((200, 1))
    B[0] = 1
    C = np.diag(np.ones(200))
    state = ct.ss(A, B, C, 0)
    state_red = ct.balred(state, N_rom)
    model.decay_array_red = state_red.A
    model.initial_composition_red = state_red.B[:, 0]
    model.rom_basis = state_red.C
    rom.run_simulation_reduced(model)
    TE = rom.return_error(model)
    SE = rom.return_error(model, range(N_nuclides))
    speedup = model.simulation_time_fom / model.
       simulation_time_rom
    return TE, SE, speedup
```

# Appendix G

# Excerpt of Balanced Proper Orthogonal Decomposition code

Two scripts have been used in the Balanced Proper Orthogonal Decomposition (BPOD) chapter. The first is a Julia script to determine the BPOD, which takes python data of the system as input. The second is a python script that measures performance for decay chains. General burnup requires some modification of the code for decay chains. The most important is to choose Radau as solver. The full code is available on: https://github.com/JonathanPilgram/BurnupROM-ScriptsAndData.

## Julia code to determine transformations

```
using HDF5
using LinearAlgebra, GenericLinearAlgebra
using DifferentialEquations
using SparseArrays

function balanced_POD(h5file, input, output)
    # Read the h5file
    save_file = joinpath(output, "OUT_" * h5file)
    python_data = h5open(joinpath(input, h5file), "r") do file
        data = read(file["A_data"])
        indices = read(file["A_indices"])
        indptr = read(file["A_indptr"])
        shape = read(file["A_shape"])
        intial_composition = read(file["N0"])
        N_rom = read(file["N_rom"])
        N_fom = read(file["N_fom"])
        N_snapshots = read(file["N_snapshots"])
        time = read(file["time"])
        A = Matrix(SparseMatrixCSC(shape[1], shape[2], indptr .+
            1, indices .+ 1, data))
        N_steps = read(file["N_steps"])
        stiffness = read(file["Stiffness"])
        run = read(file["Run"])
        # Write the burnup array to the output file
        h5write(save_file, "A_data", data)
```

```
    h5write(save_file, "A_indices", indices)
    h5write(save_file, "A_indptr", indptr)
    h5write(save_file, "A_shape", shape)
    # Return data
    (A, intial_composition, N_rom, N_fom, N_snapshots, time,
       N_steps, stiffness, run)
end

# Set parameters
N_rom = python_data[3] # reduced order modes
N_fom = python_data[4] # full order modes
N_snapshots = python_data[5]
time = python_data[6]
tspan = (0.0, time)
burnup_array = python_data[1]'
initial_composition = python_data[2]
solve_method = Rodas5P()
N_steps = python_data[7]
stiffness = python_data[8]
run = python_data[9]

# Bateman equation implementation
function burnup_equations!(dN, N, burnup, t)
    mul!(dN, burnup, N)
end

# Solve for the normal POD_modes
println("Starting")
system = ODEProblem(burnup_equations!, initial_composition,
   tspan, burnup_array)
solution = solve(system, solve_method, dt=time / N_steps,
   saveat=time / N_snapshots)
X = hcat(solution.u...)
U, _, _ = svd(X)
POD_modes = U[:, 1:N_rom]

# Solve for the adjoint solutions
Y = []
for POD_vector in 1:N_rom
    adjoint_initial = POD_modes[:, POD_vector]
    adjoint_system = ODEProblem(burnup_equations!,
       adjoint_initial, tspan, burnup_array')
    adjoint_solution = solve(adjoint_system, solve_method, dt=
       time / N_steps, saveat=time / N_snapshots)
    push!(Y, hcat(adjoint_solution.u...))
end
Y = hcat(Y...)
XY = Y' * X
# Determine balanced POD transformation
U, singular_values, Vh = svd(XY)
```

```
    U = U[:, 1:N_rom]
    Vh = Vh[:, 1:N_rom]
    singular_values = singular_values[1:N_rom]
    to_FOM = X * Vh * Diagonal(singular_values .^ (-1 / 2))
    to_ROM = Diagonal(singular_values .^ (-1 / 2)) * U' * Y'

    # Save result to hdf5 format
    A = sparse(burnup_array)
    h5write(save_file, "to_FOM", Float64.(to_FOM))
    h5write(save_file, "to_ROM", Float64.(to_ROM))
    h5write(save_file, "N_snapshots", N_snapshots)
    h5write(save_file, "N_rom", N_rom)
    h5write(save_file, "N_fom", N_fom)
    h5write(save_file, "time", time)
    h5write(save_file, "N_steps", N_steps)
    h5write(save_file, "Run", run)
    h5write(save_file, "Stiffness", stiffness)
    h5write(save_file, "initial_composition", initial_composition)
end
```

## Python code to measure performance on decay chains

```
import rom
import numpy as np
import h5py
import control as ct
from scipy.sparse import csr_matrix
import os

# Loops a folder of processed chains
for chain in sorted(os.listdir(processedChains)):
    # Reads data from Julia
    with h5py.File(processedChains + chain, "r") as file:
        t_sim = file["time"][()]
        N_rom = file["N_rom"][()]
        N_snapshots = file["N_snapshots"][()]
        to_FOM = file["to_FOM"][:].T
        to_ROM = file["to_ROM"][:].T
        N0 = file["initial_composition"][:]
        data = file["A_data"][:]
        indices = file["A_indices"][:]
        indptr = file["A_indptr"][:]
        shape = tuple(file["A_shape"][:])
        run = file["Run"][()]
        N_steps = file["N_steps"][()]
        stiffness = file["Stiffness"][()]
    burnupMatrix = csr_matrix((data, indices, indptr), shape=shape
        )

    # Runs the FOM and ROM
```

```
Bpod = rom.DecayChain(N_nuclides)
Bpod.decay_array = burnupMatrix
Bpod.initial_composition = N0
Bpod.run_simulation(t_sim, N_steps, "Euler_backward")
rom.reduce_abstract(Bpod, N_rom)
Bpod.decay_array_red = to_ROM @ Bpod.decay_array @ to_FOM
Bpod.initial_composition_red = to_ROM @ Bpod.
   initial_composition
Bpod.rom_basis = to_FOM
rom.run_simulation_reduced(Bpod)
TE = rom.return_error(Bpod)
SE = rom.return_error(Bpod, range(N_nuclides))
speedup = Bpod.simulation_time_fom / Bpod.simulation_time_rom
```