31st International Conference Nuclear Energy for New Europe
NENE 2022 September 12-15 PORTOROŽ SLOVENIA

Resampling a Detailed Reactor Power History With the Cycle Power History Script

Vid Merljak
Krško NPP
Vrbina 12, SI-8270 Krško, Slovenia
vid.merljak@nek.si

ABSTRACT

For various applications a reactor power history dataset might be needed. One example is the determination (calculation) of isotopic composition of spent nuclear fuel. Sometimes we need data with great detail (i.e. raw input data), sometimes with fewer detail. This is why the Cycle Power History Python script was developed. It allows the user to specify the number of intervals with which the input data is to be approximated. Furthermore, a few different methods are available for automatic plateau- and step-change detection. Among these, a relatively simple iterative interval merging – based on distance from the average of two neighbouring values – has proven to be both accurate and robust. This article explains the methodology, and provides both quantitative and qualitative assessment of the results.

1 INTRODUCTION

When calculating the isotopic composition of a spent nuclear fuel, we usually treat the reactor as zero-dimensional point or one-dimensional cylinder of infinite height (i.e. where the fuel assemblies are points subjected to different neutron fluence, depending on their radial position in the reactor core). In both cases the main input parameters are time and power at which the fuel was depleting ("burning") [1]. Although the amount of reactor power history details reportedly does not change the resulting isotopic composition by much [2], [3], this project was born out of curiosity to calculate this effect on a real-life case.

Using the ORIGAMI Automator tool of the SCALE package [4], [5] to perform fuel depletion calculation of several fuel assemblies (FAs) with detailed power history proved to be unsuccessful. The exact cause remains unknown, but our investigation suggests that there might be an internal constraint on how many depletion intervals can be used for any single FA. This number is approximately hundred. Since some FAs are used in nuclear reactor for up to 5 times, a single cycle power history should be given in ≤ 25 intervals. Therefore, our problem arises. It can be defined as approximating a detailed timeseries data with N coarse sectors of constant values ($N_{\text{input data}} \approx 10^4 \gg N > 3$).

The problem itself is far from being new. Moreover, since it is quite general, there is a multitude of different scientific terms that might all describe it adequately: automatic step-change detection; plateau detection; edge detection; binary segmentation; down-sampling; binning with non-equal bin widths; change-point methods; … In various scientific fields there even exists ready-made solutions, e.g. Refs. [6]-[8] in DNA-related studies. However, the DNA problem is slightly different in that there can only be discrete plateaus, whereas reactor power history (input data) also includes (linear) power-change ramps. All these will nevertheless be represented as plateaus at some constant power.

2 METHOD

2.1 Definitions and visualisation of the problem

The problem is characterised by contradictory requests to average-out relatively small but fast fluctuations and to retain detail in important transients, such as reactor trip. In other words, we need to capture both slow and fast changes (see Figure 1). As follows from the intended application – i.e. using coarse power history as depletion intervals to compute spent fuel isotopic composition – possible suitable criteria could be in units of energy released in a certain interval, $\Delta E = P\Delta t$.

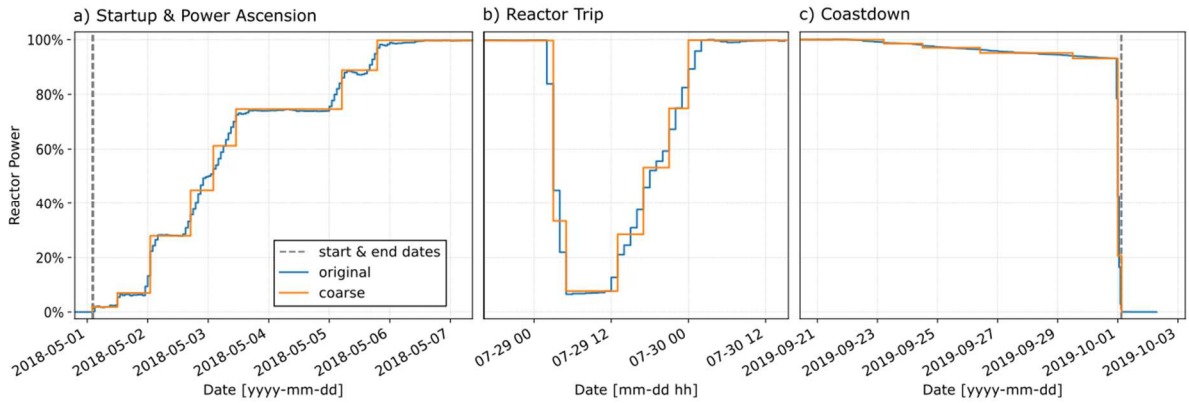

Figure 1: Typical features of a reactor fuel cycle history: a) startup and power ascension programme; b) possible reactor trip; and c) power coastdown and reactor shutdown. Mind the time scale changes. Both original input data and coarse resampled output data are shown.

Because the original input intervals can be of varying lengths, we need to incorporate weights into our equations. Note that throughout this paper, symbols for power, P, and for a general ordinate axis variable, x, will be used interchangeably to retain more general formula looks. Let us first declare notation convention for a simple weighted mean,

$$\bar{x} = \overline{x_{(1,\ldots,N)}} = \bar{x}|_{i=1}^{N} = \frac{\sum_{i=1}^{N} \Delta t_i x_i}{\sum_{i=1}^{N} \Delta t_i}, \tag{1}$$

where Δt_i is the i-th interval duration. Weighted root mean squared error (RMSE) is a scalar quantity which can be defined either for deviation of a single dataset from its mean value, or for deviation between two datasets. We will use them both, as appropriate:

$$\mathrm{wRMSE}_1 = \sqrt{\frac{\sum_{i=1}^{N} \Delta t_i (\bar{x} - x_i)^2}{\sum_{i=1}^{N} \Delta t_i}}, \qquad \mathrm{wRMSE}_2 = \sqrt{\frac{\sum_{i=1}^{N} \Delta t_i \left(\mathrm{P}_i^{\mathrm{fine}} - \mathrm{P}_i^{\mathrm{coarse}}\right)^2}{\sum_{i=1}^{N} \Delta t_i}}. \tag{2}$$

Of course, the second formula will only work if both fine and coarse datasets have same number of intervals, $N = N_{\mathrm{input}}$. This is achieved by padding the coarse dataset, i.e. solely for comparison purposes temporary restoring all initial fine intervals and populating them by the average values of each corresponding coarse interval.

2.2 Implementation

The Power Cycle History script is coded in Python programming language and is designed for interactive execution in an (Anaconda) command prompt. Besides actual intervals determination and inevitable parsing of input arguments, the program also does the following:

- checks the input data for possible inconsistencies or bad measurements (available if two complementary reactor power datasets are available, such as calorimetric reactor power and neutron flux);
- automatically detects the starting and ending point of the fuel cycle (based on supplied official start and end timestamps [it is generally recommended to ensure there are a few extra measurements before start and after the end of a fuel cycle]);
- calculates the total released energy and (referring again to the official records) normalises the output accordingly;
- displays the results and writes them onto a file in the format of ORIGAMI Automator.

2.3 Possible Methods

For automatic detection of interval boundaries, one could use several criteria. Of course, some are better than other which can be assessed qualitatively by their robustness to input data peculiarities, and quantitatively by RMSE deviation from original data (see Eq. 2 and Sec. 3, Results). Some criteria/methods with comments on their applicability to our field of usage are listed in Table 1. For the sake of simplicity, the input data is assumed to be in units of percent of reactor Rated Thermal Power (% RTP).

Table 1: Possible criteria and/or methods with remarks on their applicability.

No.	Criteria based on	Unit	Comment
1	Absolute magnitude of signal change	% RTP	Key parameter: tolerance on the magnitude.
2	Signal change rate	% RTP/h	Discriminate the lower 99-th percentile of signal change rate absolute values.
3	Interval change rate (frequency)	interval/h	Not discerning important intervals from totally omittable ones.
4	Released energy	EPFD	Merge neighbouring intervals, starting from those with the most similar power level (see Section 2.4).
5	Cumulative RMSE	% RTP	Sum of forward and backward cumulative RMSE has a minimum at step-change.
6	Rolling average	/	Signal processing method. Not applicable as it smudges sharp corners at reactor trip.
7	FFT + low/high pass filter	/	Signal processing method. Not applicable as we need to capture both slow and fast changes. Generality is difficult to achieve.

Among these criteria, No. 3, 6 and 7 are not implemented, No. 1 and 2 are implemented but their usage is discouraged (i.e. they are neither optimal nor robust but are retained for comparison), while No. 4 and 5 are implemented and their performance is discussed more deeply in the following subsections.

2.4 Interval Merging Methodology

Interval merging based on the released energy is in fact a very simple method, as illustrated on Figure 2.

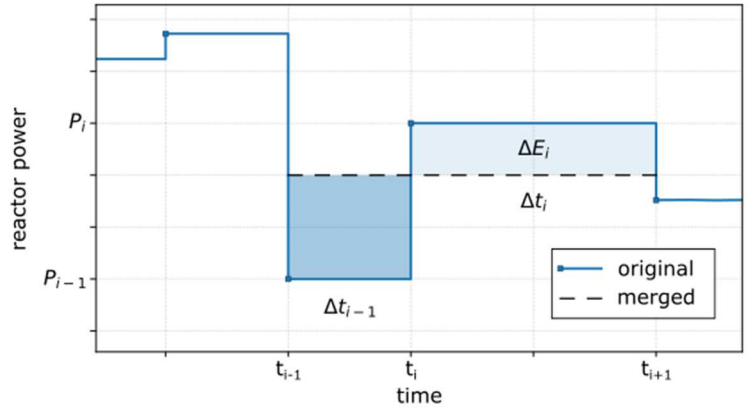

Figure 2: Interval merging scheme. Both shadowed areas correspond to same amount of energy, ΔE_i. The bigger this ΔE_i, more important is to *retain* the interval starting at t_i.

For each input interval i it calculates its *importance*, ΔE_i, i.e. the amount of surface (energy) that would "shift" between the interval in question (i) and its preceding interval ($i-1$), should the two be merged:

$$\Delta \mathrm{E_i} = |P_i - P_{i-1}| \left(\frac{\Delta t_i \cdot \Delta t_{i-1}}{\Delta t_i + \Delta t_{i-1}} \right) . \tag{3}$$

After calculating the importance of all initial intervals, we can set a cut-off value that would leave us with the N most important intervals. Off course, as the importance metric is calculated for merging of only two adjacent intervals, it has to be used that way, i.e. if the i-th interval was merged with its predecessor, then (the starting point of) the $(i+1)$-th interval must be left intact. This means that not all "unimportant" intervals can be merged in one pass – instead, the process is iterated. Starting with some 1.3×10^4 intervals, it takes about 15 iterations to get down to $N = 25$ intervals. With fewer intervals left, each iteration is faster than the previous one.

2.5 Improvements (?) to Interval Merging Method

A linearly decreasing safety factor may be employed, so that in first ten iterations more than $N = 25$ most important intervals are marked for preservation. The aim is to avoid prematurely merging intervals that would later (when some of their neighbours get merged) turn out to be important. Seeing it in action this seemed to produce step-changes at more appropriate times, however, according to the RMSE deviation between the initial and final (coarse) power history profile (Eq. 2), it actually leads to about 5 % (relative) worsening.

Another idea to improve the RMSE within the same integral merging method is to consider the order in which the intervals are merged. (Again, note that we can only merge two adjacent intervals at a time.) Implemented are the following three options:

a) "simple" – a simple sequential merging, time-ordered progression;

b) "lower of the two" – same time progression, but among the two neighbouring intervals with importance below cut-off, merge the one with lower importance;

c) "from lowest" – sort the intervals according to their importance and start merging from the lowest importance. Mind the "neighbour rule"!

Option c) makes the most sense, mathematically, and is used by default. However, its effect was overly anticipated – the final RMSE was affected (in fact, worsened) by about 2 % (relatively). As it will be seen in results (Table 2), such relative increase still yields satisfactorily small absolute RMSE deviation.

As the total released energy is preserved in any case, thus being acceptable for intended use of fuel depletion calculations, it remains open for debate which of the two criteria (visual vs. RMSE) should be prioritised.

2.6 STDBA Methodology

Another option is the cumulative (or expanding) RMSE (method 5 from Table 1). As before, we must use weights which complicates the equations, thereby prolonging the calculation. The sum of *forward and backward cumulative standard deviation* (STD before-after or STDBA) is a vector quantity, computed as:

$$\mathrm{STDBA}_j = \sqrt{\frac{\sum_{i=1}^{j} \Delta t_i \left(\overline{x_{(1,\dots,j)}} - x_i\right)^2}{\sum_{i=1}^{j} \Delta t_i}} + \sqrt{\frac{\sum_{i=N}^{j} \Delta t_i \left(\overline{x_{(N,\dots,j)}} - x_i\right)^2}{\sum_{i=N}^{j} \Delta t_i}} \quad \text{for } j \in (1, 2, \dots, N). \tag{4}$$

Here, according to Eq. (1), $\overline{x_{(1,\dots,j)}}$ is a weighted mean from $i = 1$ up to j, and conversely $\overline{x_{(N,\dots,j)}}$ is a weighted mean from $i = N$ backwards to j.

STDBA has a local minimum at location of a step-change in input data. As early as in the power ascension program after refuelling outage there are several step-changes in the reactor power history. The STDBA signal is therefore rather complex (see Figure 3).

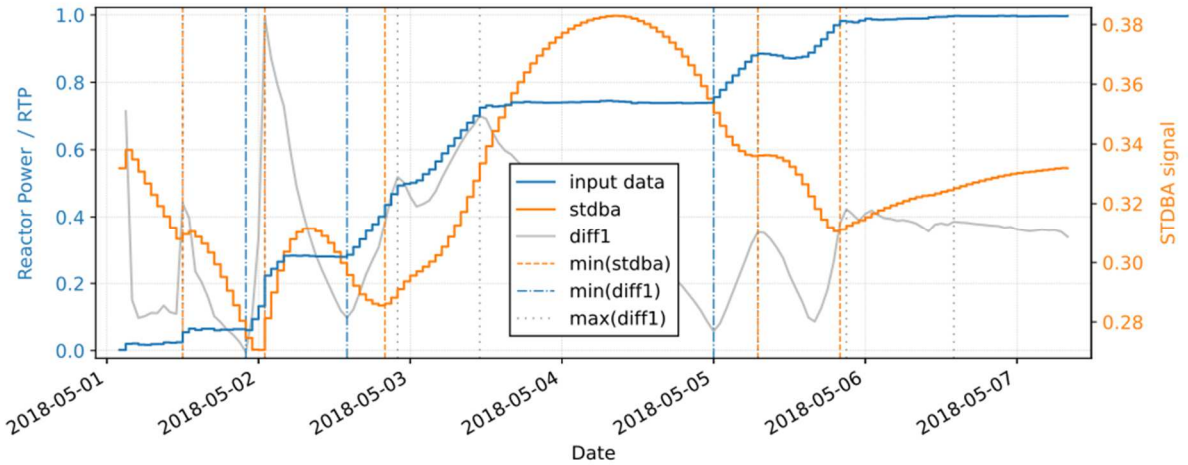

Figure 3: The STDBA methodology applied only on a small section reactor power history (startup & power ascension program). Note the complexity of the STDBA signal and how appropriate locations for new intervals can be extracted from local extremes of both the stdba signal and its time derivative. Some more could be added – e.g. see min(diff1) – but that would require hand-tuning of parameters for the local extreme search function.

Appropriate locations for starting a new power interval can be extracted by searching for local minima. Additionally, calculating the STDBA first order derivative, one can see that plateaus in the signal start at local maximum, and end at local minimum, which gives us more candidates for interval start points.

Unfortunately, it proved hard to control how many intervals are detected, and to ensure good detection throughout the cycle. This could be alleviated by computing the STDBA separately for several sub-intervals from input data (e.g. tenths of the total cycle duration), effectively increasing the edge-detection resolution. But how does one decide *where* to start a new section? And we would also need a means to compute absolute importance of intervals from different sections, so that they can be compared.

Despite the abovementioned (and other) effort(s), the STDBA approach remains an inferior option. Namely, a) it suffers from the fact that the outcome is highly dependent on sectioning and parameters for local extrema determination; b) would need help from some other method to be fully automatic; c) is computer time intensive (order $\mathcal{O}(N^2)$ as opposed to $\mathcal{O}(N)$ for interval merging); and d) cannot capture the linear power ramp at coastdown. The latter comes from the fact, that STDBA is *designed* to capture step-changes (what is more, only *one* step-change at a time). In a way, we were asking too much from this method. Taking all this into account, further refinements to the STDBA approach were abandoned.

3 RESULTS

In this paper it is not feasible to display a detailed side-by side visual comparison for all four implemented methods. In fact, the results for the interval merging method (based on the released energy criteria) are already displayed in Figure 1. The principle and difficulties of the STDBA approach are depicted in Figure 3. To establish overview, a quantitative comparison of the four implemented methods is given in Table 2.

Table 2: Performance of and comments to the implemented methods.

No.	Criteria based on	N	RMSE	t_{CPU} [s]	Quick verdict
1	Change magnitude (tolerance)	26	0.0062	0.083	Surprisingly good (but where is the coastdown?)
2	Signal change rate	48	0.0065	0.004	Some discrepancy at TRIP but: fastest!
4	Released energy	25	0.0052	7.8	Most accurate; default
5	Cumulative RMSE	24	0.0230	15.5	Inappropriate (see Sec. 2.6)

Surprisingly, the CPU time needed for each of the methods varies by almost four decades. The interval merging method (No.4, released energy criteria) is actually quite slow, as it has to do several iterations each with more than a thousand intervals. It is, however, the most accurate one in terms of the lowest RMSE $\times N$, where N is the final number of coarse output intervals. The magnitude-of-change method (No.1) follows closely. However, with the tolerance hand-tuned to get to $N \approx 25$, most of the intervals were "spent" on power ascension programme and reactor trip, effectively ignoring coastdown. Rate sensitive method (No.2) captures the coastdown better, but produces slightly worse deviation at reactor trip and generally uses too many intervals to achieve comparable RMSE value. The many issues of the STDBA method (No.5, cumulative RMSE criteria) have been already described in detail. Thus, let us repeat the verdict of an over-engineered, inefficient and fundamentally inappropriate approach.

Last but not least, one of the main design requests for our resampling script was for user to be able to prescribe the number of output coarse intervals. This was in fact achieved only with the interval merging method. Relating to the last sentence from Sec. 2.3, it is now clear that the only recommended (working) method is the interval merging method, while the other three remain accessible for comparison purposes.

4 CONCLUSION

In a challenge of finding an optimal way to resample a power reactor history, we investigated four methods of varying complexity. Two simplest ones are not versatile enough, while one other method is too complicated and in fact fundamentally inappropriate.

The remaining method – interval merging method – performs flawlessly. It considers merging pairs of neighbouring intervals that have similar power level. Off course, interval time duration is accounted for. The method is iterative and quite CPU-time expensive, but this is not an issue since we only need to perform the task once for each reactor fuel cycle power history. The user is to provide the input data in a predetermined format and set the desired number of output intervals. Results are obtained in $< 20\ s$, with a normalised RMSE of about 5×10^{-3}.

REFERENCES

[1] S. Glasstone, A. Sesonske, Nuclear Reactor Engineering, Fourth edition, Volume one, Chapman & Hall, New York, 1994, pp. 221-224.

[2] U.S. NRC, Regulatory Guide 3.54, Revision 2, "Spent Fuel Heat Generation in an Independent Spent Fuel Storage Installation", 2018, Washington, DC.

[3] V. Merljak, "Comparing Different Approaches to Calculating Decay Heat Power of a Spent Fuel Dry Storage Cask for Krško NPP", Proc. Int. Conf. Nuclear Energy for New Europe, Bled, Slovenia, September 6-9, 2021, paper No. 304, Nuclear Society of Slovenia, 2021.

[4] W. A. Wieselquist, R. A. Lefebvre, M. A. Jessee (Editors), SCALE Code System, Version 6.2.4, ORNL/TM-2005/39, Oak Ridge National Laboratory, April 2020.

[5] W. A. Wieselquist et al., ORIGAMI Automator Primer: Automated ORIGEN Source Terms and Spent Fuel Storage Pool Analysis, ORNL/TM-2015/409, Oak Ridge National Laboratory, April 2016.

[6] Adam B. Olshen et al., "Circular binary segmentation for the analysis of array-based DNA copy number data", Biostatistics, Volume 5, Issue 4, October 2004, pp. 557–572, DOI: https://doi.org/10.1093/biostatistics/kxh008.

[7] Venkatraman E. Seshan, Adam B. Olshen, "A faster circular binary segmentation algorithm for the analysis of array CGH data", Bioinformatics, Volume 23, Issue 6, March 2007, pp. 657–663, DOI: https://doi.org/10.1093/bioinformatics/btl646.

[8] Venkatraman E. Seshan, Adam B. Olshen (2021). DNAcopy: DNA copy number data analysis. R package, version 1.68.0