LA-UR-22-22699

Approved for public release; distribution is unlimited.

Title:	Spiner-EOSPAC Comparison: performance and accuracy on Power9 CPU and GPU
Author(s):	Pietarila Graham, Anna Mataleena Holladay, Daniel Alphin Miller, Jonah Maxwell Peterson, Jeffrey Hammett
Intended for:	Report
Issued:	2022-03-23

Spiner-EOSPAC Comparison: performance and accuracy on Power9 CPU and GPU

Anna Pietarila Graham, Daniel Holladay, Jonah Miller, Jeffrey Peterson

February 2022

1 Introduction

The equations of conservation of mass and momentum form the Euler equations of fluid dynamics. When closed with an equation of state (EOS)—a constitutive model for pressure as a function of density and energy—these equations can be solved as an initial-value problem, allowing one to predict the evolution of a material. Physics and engineering problems at all scales, from climate science, to race car design, to astrophysics use this methodology to make predictions. Arguably the most important information contained in the equation of state is the *sound speed* or *bulk modulus*, describing how the pressure changes with respect to other thermodynamic quantities. For a variety of reasons, thermodynamic quantities are often conveniently represented in terms of density ρ and temperature T. This means that when a fluid dynamics code requires, say, pressure as a function of density and energy, the temperature must first be computed by "inverting" the equation for energy from a function of density and temperature to one that is a function of density and energy.

A typical Sesame [1] equation of state is constructed as a semi-empirical model where tunable parameters are adjusted to ensure that the EOS reproduces available data. However, the available data is subject to varying degrees of uncertainty and only represents a small fraction of the overall domain over which a Sesame EOS is expected to operate. Moreover, the details of solving the Euler equations—including solving the Riemann problem at cell boundaries as part of determining fluxes as well as finding the mixed cell volume fractions through a closure rule such a pressure-temperature equilibrium—place additional demands on the EOS that can exacerbate errors in the interpolation. The result is a balance between performance and the ability to faithfully represent the required physics of the material within the uncertainty of the underlying data and models that went into the creation of the EOS.

EOSPAC [2] is a utility library written in C for accessing, manipulating, and interpolating Sesame EOS and transport coefficients (e.g., opacity data). It offers a well-documented interface for multiple languages and platforms, including CPUs and GPUs. EOSPAC is used by several of the ASC physics codes in addition to being publicly available via, e.g., spack. In this work only a very small subset of EOSPAC capabilities are evaluated. The EOSPAC manual [2] provides a thorough documentation of the code.

Spiner EOS is a tabulated equation of state reader for continuum dynamics codes, packaged as part of the open-source Singularity EOS library [3]. It is purpose-built specifically for providing exactly the information required for continuum dynamics codes in a way that is both performant on both CPUs and GPUs and easy for them to use. To enable performance-portability, Spiner EOS is built on top of the Spiner [4] interpolation library. While any data can be converted to Spiner table format, Spiner EOS is designed to leverage EOSPAC to populate the Spiner table with Sesame data. In particular, a separate program `sesame2spiner` uses EOSPAC to interpolate not only Sesame data but inverted quantities and thermodynamic derivatives onto a new grid used by Spiner EOS. This is a core design feature: EOSPAC's rational-function interpolation and inversion utilities provide high-accuracy reconstructions of the sparsely populated Sesame tables which Spiner EOS uses. Both codes support a number of options that can be tuned to optimize a trade-off between performance, accuracy, and memory cost. For example, the density of the grid, as well as whether or not to pre-invert, whether to run on CPU or GPU can all be controlled at run

time or compile time. Additionally, Spiner supports caching of initial guesses for inversions and performance tricks for floating point operations.

The goal of this report is to concisely compare EOSPAC to Spiner EOS for a selection of commonly-used materials over as much of their domain as is possible. Importantly, this comparison does not address the way that the Euler equations may amplify interpolation errors and we take as given that the rational function interpolator gives the best possible representation of the EOS represented by the Sesame data. Neither does the report address all, or even majority of, the capabilities of each code nor does it explore performance or accuracy over multiple compilers and platforms.

2 Comparison setup

The scope of the comparison is limited due to the vast number of options, table types and materials used in EOS look-ups. For this comparison we have chosen five materials (3720 aluminium, 4272 stainless steel, 2140 iron, 5031 dry air and 5267 deuterium) and four table types: two tables, $p(\rho, e)$ and $T(\rho, e)$ require inversion and $e(\rho, T)$ and $p(\rho, T)$ do not [1]. These materials represent a variety of EOS behaviors and are important materials for LANL hydro code users. The 4272 stainless steel table is a commonly-used EOS that contains Van der Waal loops (i.e. the pressure is not monotonic in density) and has a fairly dense density-temperature grid. By contrast, the 2140 iron table has a very sparse grid that sometime poses a challenge for the rational function interpolator and has Maxwell constructions to ensure that the pressure is monotonic. The remaining two tables are both gasses that are monotonic in pressure, but the 5267 deuterium pressure table contains Maxwell constructions and sharp discontinuities while 5031 air is fairly smooth. The look-ups are limited to within table bounds (i.e., we do not consider extrapolation) and sample the full dynamic range of the table. We consider two batch sizes (number of points for interpolation), 512^2 and 2048^2. For performance we test both ordered and random input. We do not evaluate the accuracy of derivatives but note that they are often essential outputs for host codes.

The baseline for comparisons of accuracy and performance for this analysis is EOSPAC with its default settings: rational interpolation, no pre-inversion and no insertion of additional points to the table prior to inversion and interpolation. We did not include any tests based on analytical expressions for the EOS, instead assuming that the default settings for EOSPAC provide the most accurate representation of the EOS data. This assumption is expected to fail at, for example, sharp gradients. However an analysis of these phenomena is beyond the scope of this report. All the tests and baseline are run on a Power9 node using the NVHPC compiler. We measure performance both on CPU and GPU. The Spiner GPU implementation is achieved via Kokkos [5] while EOSPAC uses OpenMP [6]. For CPU only serial performance is considered.

For Spiner two setups are used for tables requiring inversion: *RhoT* in which the inversion is done iteratively during interpolation by Spiner and *RhoSie* where EOSPAC is used to create a pre-inverted table at preset density and energy points. For all tests 100 points per decade are used to sample the originally much coarser Sesame tables.

For EOSPAC we investigate linear and rational interpolation, pre-inversion, and the effect of grid density (adding a preset number of points between the original Sesame grid points). It is worth noting the very different approaches the two codes utilize for enhancing grid resolution: while Spiner samples the original data uniformly in log-space EOSPAC retains the original data's sampling pattern.

3 Comparison results

For the results we use the following naming convention:

S=Spiner, *RhoSie*=pre-inversion

E=EOSPAC, P=pre-inversion, R rational interpolation, number=number of new data points inserted in the Sesame table prior to inversion and interpolation

[1] T=temperature, ρ=rho=density, p=pressure, e=internal energy

We found that EOSPAC linear interpolation errors are almost universally an order of magnitude higher than errors for any of the other options. The performance benefit when using linear instead of rational interpolation is also not large enough to justify the increased errors, so our results from linear interpolation will not be included in the results.

3.1 Memory usage

On the GPU the memory usage of Spiner (without pre-inversion) and EOSPAC (PR6=pre-inverted, rational, data insert=6) are very similar. The total memory used in all tests (i.e., 5 materials, 4 table types and 2 batch sizes) is $1.1x$ larger for EOSPAC than for Spiner. For these tests, the memory usage is dominated by the interpolation batch size and not the amount of memory required to store the tables so these findings align with our expectations.

3.2 Performance and accuracy

For each test we compute the absolute relative error,

$$error = \frac{|baseline - test|}{\max(|baseline|, 1.0 \times 10^{-10})}$$

and then report the 95^{th} percentile over all the sampled points. In other words, the error for a randomly sampled point in the domain of the EOS would be expected to be less than the reported error 95% of the time. We then combine these results of the different materials and batch sizes for each table type and report the min, max, mean, and median for the distribution. The results are summarized in figures: 1, 2, 3 and 4 as well as table 1. Note that both Spiner and EOSPAC CPU and GPU results are internally identical, so any errors presented will apply to both.

Spiner interpolation error is $\approx 1 \times 10^{-4}$ both for *RhoT* and *RhoSie*. The speedup on the CPU for tables not requiring inversion is slightly larger than 2 and on the GPU Spiner is $\approx 25x$ faster than the GPU implementation of EOSPAC. For tables requiring inversion it is evident and not surprising that *RhoSie*, which relies on pre-computed inversions by EOSPAC, is more performant. Both *RhoT* and *RhoSie* have similar errors indicating that the pre-inversion error in the tables created in EOSPAC is smaller or comparable to the Spiner interpolation error.

EOSPAC pre-inversion errors decrease as the number of points inserted in the table increases. For the highest density grid considered (inserting 6 data points) the error is comparable to Spiner. The choice of using 100 points per decade for the Spiner tables leads to the tables being on average roughly $100x$ larger than the original Sesame tables (note the inserted points are not distributed based on original Sesame points as is the case for EOSPAC but based on the data values). The EOSPAC PR6 inserts fewer points ($36\times$ original number of points).

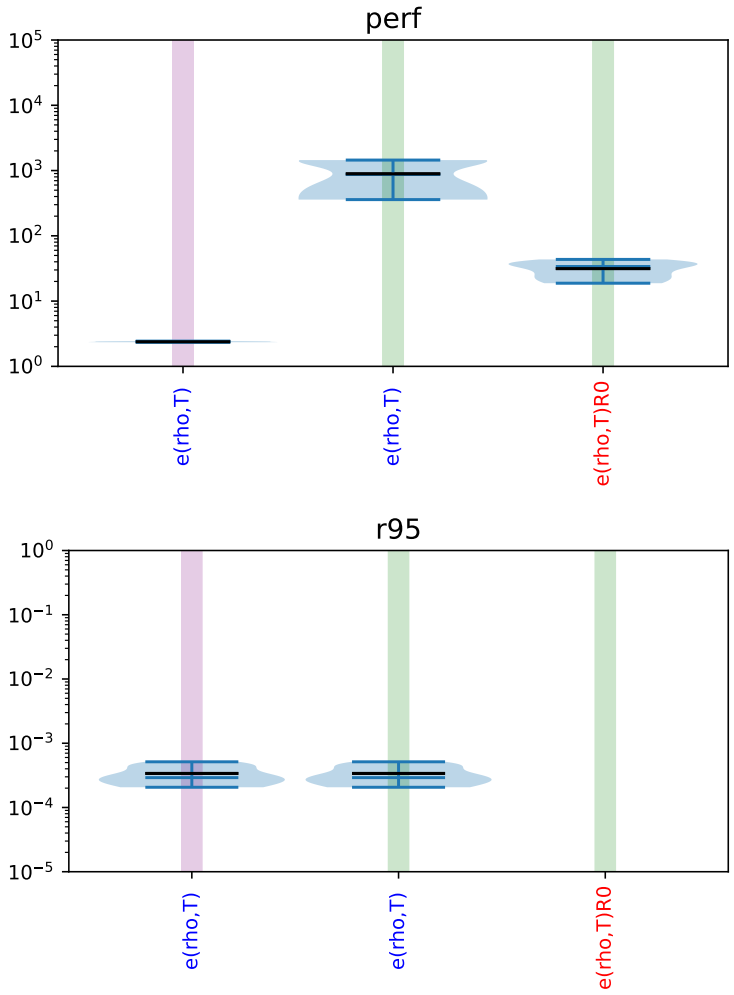

Figure 1: Comparison of $e(\rho, T)$ performance and accuracy. Top figure shows the performance relative to baseline and bottom figure shows the 95^{th} percentile error. The blue and red fonts denote Spiner and EOSPAC, while the light purple and green backgrounds represent CPU and GPU results respectively. The bulbous blue "violin plots" represent the distribution of results over all materials and sample sizes. The overlaid blue bars present the minimum, median and maximum values in the distribution and the black bars are the means. Note that in the error plots the CPU and GPU results for a given table are identical.

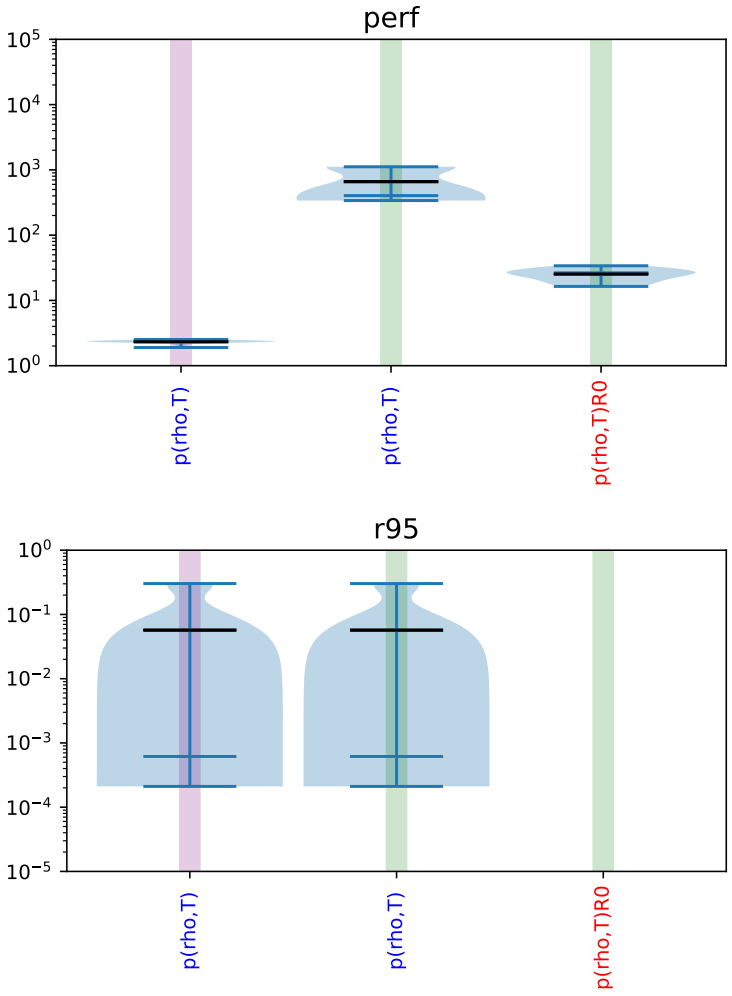

Figure 2: Comparison of $p(\rho, T)$ performance and accuracy. Figure interpretation is the same as for figure 1.

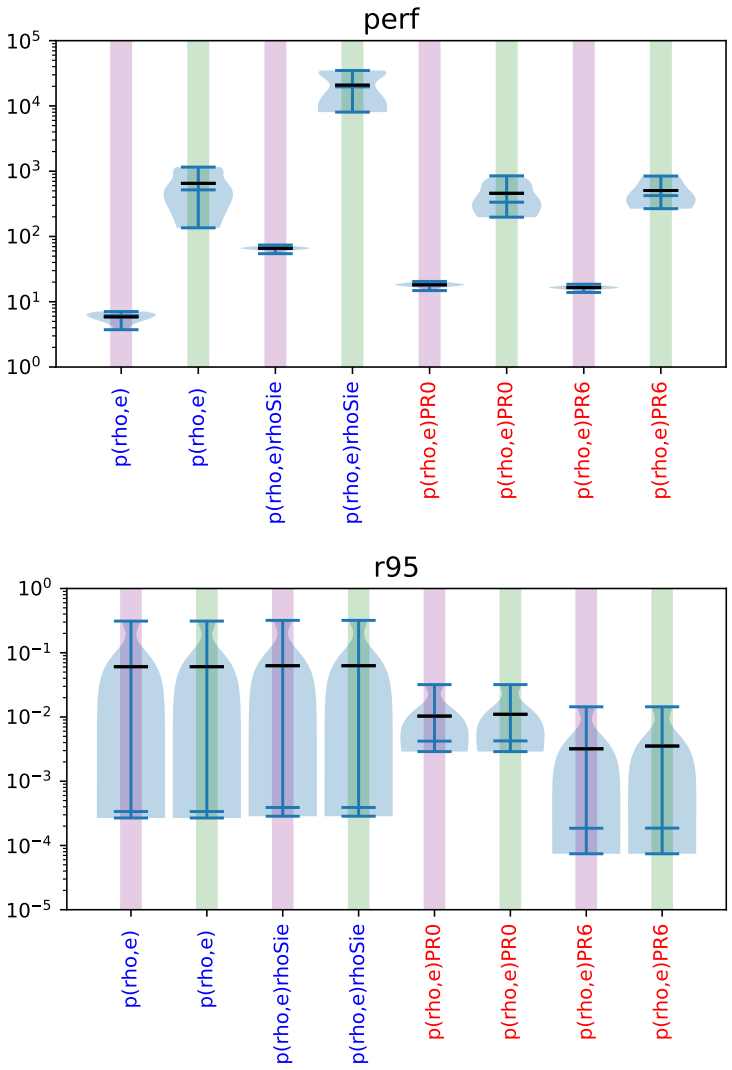

Figure 3: Comparison of $p(\rho, e)$ performance and accuracy. Figure interpretation is the same as for figure 1.

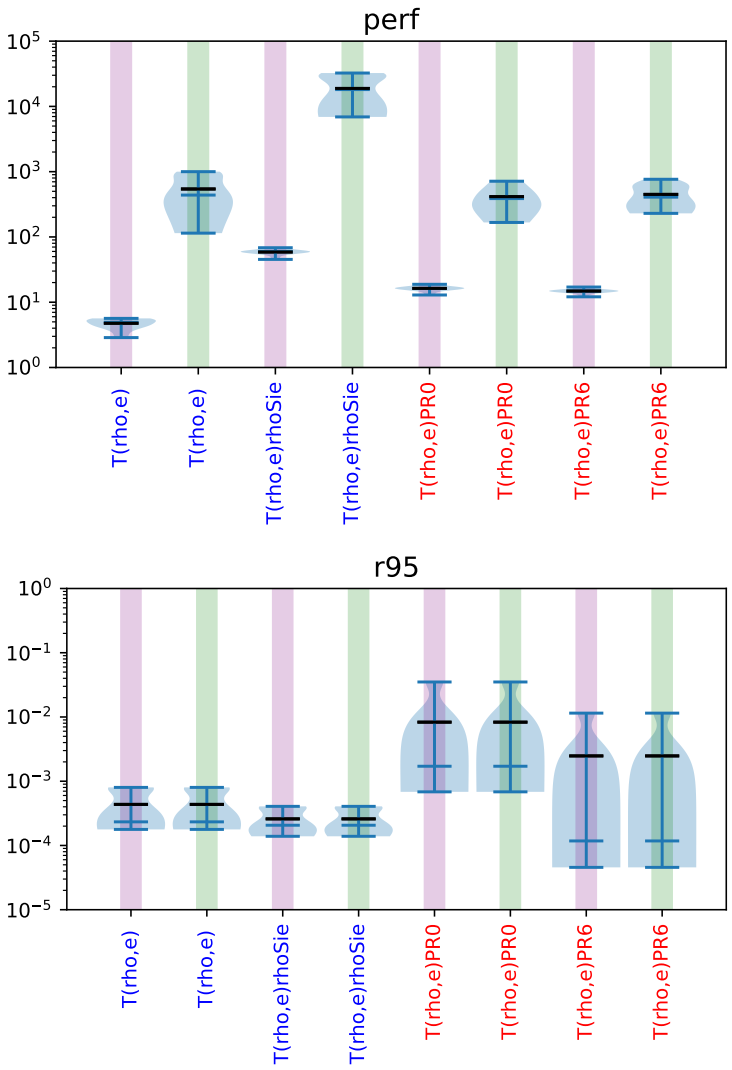

Figure 4: Comparison of $T(\rho, e)$ performance and accuracy. Figure interpretation is the same as for figure 1.

	95th % error (CPU and GPU)				CPU performance				GPU performance			
	Mean	Median	Min	Max	Mean	Median	Min	Max	Mean	Median	Min	Max
$e(\rho, T)$ S	3.39E-04	2.92E-04	2.06E-04	5.14E-04	2.4	2.4	2.3	2.4	889.5	870.3	358.8	1447.0
$e(\rho, T)$R0 E									31.5	33.9	18.8	43.5
S/E									28.3	25.7		
$p(\rho, T)$ S	5.70E-02	6.16E-04	2.11E-04	3.03E-01	2.3	2.3	1.9	2.5	659.8	403.4	338.7	1115.8
$p(\rho, T)$R0 E									25.4	26.5	16.4	33.9
S/E									26.0	15.2		
$p(\rho, e)$ S	6.06E-02	3.37E-04	2.68E-04	3.11E-01	5.9	6.1	3.7	7.1	651.6	517.9	135.7	1155.7
$p(\rho, e)$*RhoSie* S	6.28E-02	3.90E-04	2.85E-04	3.19E-01	65.8	66.3	54.5	74.0	20790.7	19621.3	8042.6	34892.6
$p(\rho, e)$PR6 E	3.53E-03	1.86E-04	7.42E-05	1.44E-02	16.5	16.4	13.9	18.5	506.2	422.6	266.7	843.6
S/E	17.2	1.8			0.4	0.4			1.3	1.2		
S *RhoSie*/E	17.8	2.1			4.0	4.0			41.1	46.4		
$T(\rho, e)$ S	4.36E-04	2.33E-04	1.78E-04	8.02E-04	4.8	4.9	2.9	5.7	542.9	438.1	114.3	1000.2
$T(\rho, e)$*RhoSie* S	2.59E-04	2.07E-04	1.39E-04	4.06E-04	58.8	59.6	45.3	68.4	18849.7	18120.0	6902.0	32459.5
$T(\rho, e)$PR6 E	2.48E-03	1.18E-04	4.56E-05	1.15E-02	14.8	14.8	12.1	17.1	448.7	407.3	229.8	764.7
S/E	0.2	2.0			0.3	0.3			1.2	1.1		
S *RhoSie*/E	0.1	1.8			4.0	4.0			42.0	44.5		

∞ Table 1: Summary of performance and error measures. For EOSPAC results we show the best performing (in terms of error and performance) options: for non-inverted tables the default rational interpolation and for inverted pre-inverted rational interpolation with $INSERT\_DATA$=6. In addition to the baseline comparison we also compare Spiner against the EOSPAC resuls shown in the table.

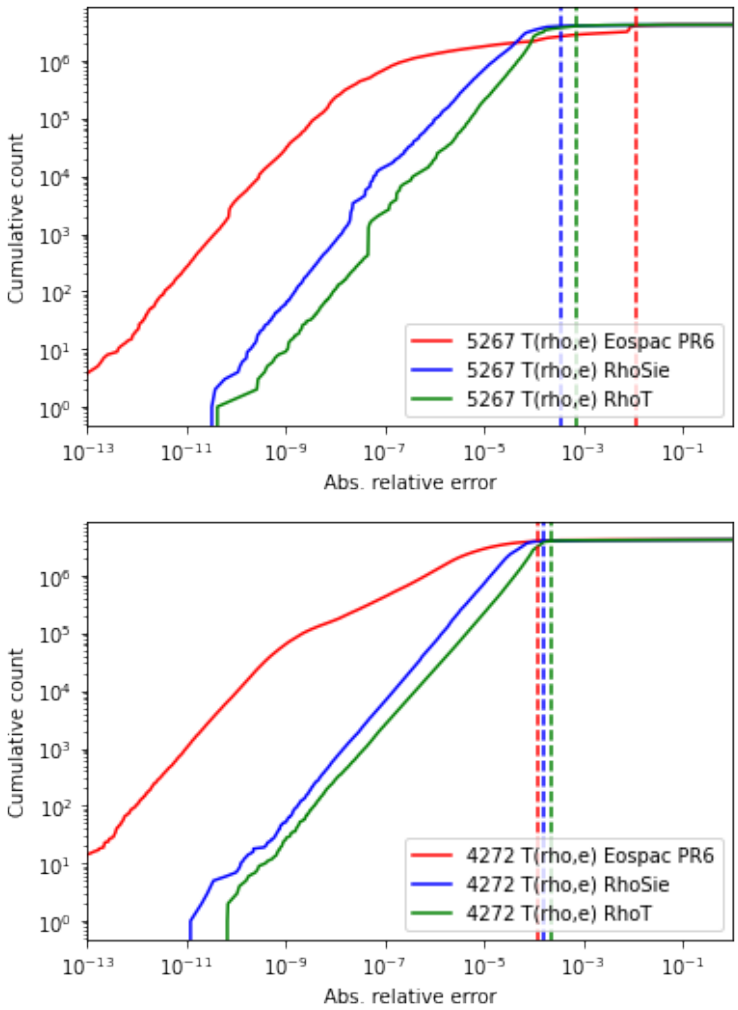

Figure 5: $T(\rho, e)$ cumulative distribution functions for materials 5267 (top) and 4272 bottom. The vertical lines indicate the 95^{th} percentile error for each distribution.

The 95thth percentile error does not capture the expected error for most of the lookups on a table but rather reports a measure of the "worst-case" scenario. The $T(\rho, e)$ violin plot (fig. 4) for EOSPAC PR6 has a wide spread: the mean is over an order of magnitude larger than the median. Figure 5 shows the cumulative distribution functions for material 5267 (which has the highest error in the EOSPAC PR6 distribution) and 4272 (a significantly lower overall error). The more the distribution lies to the left of these plots, the smaller the error will be for the vase majority of points. The two materials' functions are very similar for the bulk of the data points for all three setups. However, the 5267 EOSPAC distribution has a fairly small subset of points with noticeably higher errors. This results in the EOSPAC 95$^{\text{th}}$ percentile error being well over a magnitude higher than both the Spiner errors while the median error is roughly the same for all three.

The performance comparisons shown above all relied on ordered input. Table 2 compares the performance of using ordered input data with using randomly shuffled input data. Spiner *RhoT* and to a lesser degree EOSPAC on CPU are less performant if input data is randomized. Spiner *RhoSie* is barely affected. We note that *RhoT* offers caching machinery to leverage the order of the input data, which we did not modify for this test. Modifying this setting may recover performance for *RhoT*. We also note that, unlike the two other setups, *RhoSie* involves no inversion.

	EOSPAC (PR6)	Spiner *RhoSie*	Spiner *RhoT*
CPU	1.6x	1.1x	3.5x
GPU	1.0x	1.1x	2.2x

Table 2: Effect of ordered vs random input on performance. The values shown are the relative performance of ordered to random input. Values greater than one indicate a speedup when running with ordered input.

	EOSPAC PR6		Spiner		
	No inversion	Pre-inversion	*RhoSie*	*RhoT* no inversion	*RhoT* inversion
CPU		1.1x	1.1x	1.0x	1.3x
GPU	1.4x	2.2x	3.6x	3.4x	2.2x

Table 3: Effect of batch size on performance. The values shown are the relative performance of the larger (2048^2) to the smaller (512^2) batch size. Values greater than one indicate a speedup when running with a larger batch size.

The effect of batch size (Table 3) is most evident for tests run on the GPU. All three, EOSPAC and Spiner *RhoT* and *RhoSie*, benefit performance-wise when the number of points to be interpolated, i.e., the amount of work, increases.

4 Conclusions and recommendations

This comparison work confirms already established recommendations for EOSPAC: when accuracy is the most important factor and performance is not as critical, one should use the default options for EOSPAC, namely no pre-inversion and rational interpolation. If performance is critical choosing rational interpolation and pre-inversion with a moderate number of inserted points is preferable. Since Spiner interpolation is found to be more performant than EOSPAC, especially on the GPU, Spiner *RhoSie* can be used to further accelerate EOSPAC's pre-inversion at negligible cost to accuracy.

We did not test how the EOSPAC PR6 error, in particular for $T(\rho, e)$, would change if the number of points was increased and/or the points were distributed in a different way, perhaps mimicking Spiner. The Spiner errors with *RhoT* and *RhoSie* indicate that with a higher density grid and/or better distributed points the error is $\leq 1 \times 10^{-4}$. The EOSPAC pre-inversion errors would likely be similarly reduced by increasing the grid density to be comparable to that used by Spiner.

When using EOSPAC with default settings (operating mode for many host codes) as the baseline for comparison, the best performance and smallest error is achieved by *RhoSie*, i.e., using EOSPAC to create the pre-inverted tables at a much higher grid density than the original Sesame data. While the *RhoT* errors are comparable for inverted tables there is no performance gain when compared to EOSPAC PR6. For pure interpolation with 100 points per decade the error is roughly 1×10^{-4}.

Both Spiner setups, *RhoT* and *RhoSie*, depend on EOSPAC: to create the densely sampled tables used for interpolation and in the case of *RhoSie* to invert the data. Because of this dependency any issues in the interpolation and inversion of data inside of EOSPAC will be inherited by Spiner. Any uncertainty in the underlying equation of state model is inherited by both codes.

EOSPAC is well documented and has an established API. In comparison implementing Spiner in a host code will require interaction with the Singularity EOS code development team—fortunately the team is available for this work. The scope of the current analysis is limited by the materials and table types considered, while also neglecting to study the accuracy of derivatives or extrapolation. We intend to study the accuracy of the derivatives through host code integration since that will provide the context necessary to properly interpret their effect. Further work will also be required to study the effect of grid resolution in the Spiner and pre-inverted EOSPAC tables to deliver suitable recommendations for users.

5 Acknowledgements

We would like to thank the XCAP team, especially Joann Campbell, Christoph Junghans, Christopher Mauney, and Karen Tsai. We are grateful to the EOSPAC team, including David Pimentel, Daniel Sheppard, and Scott Crockett. Finally, we thank The RIOT team, in particular, Josh Dolence, Chad Meyer, and Sriram Swaminarayan.

References

[1] Sesame: the los alamos national laboratory equation of state database. Technical report, 1992.

[2] David A Pimentel. *EOSPAC User's Manual: V.6.5.* United States, 2021.

[3] Jonah M Miller et al. Singularity eos: Performance portable equations of state. *In Prep. For the Journal of Open Source Software*, 2021.

[4] Jonah M Miller et al. Spiner: Performance portable tables and interpolation. *In Prep. For the Journal of Open Source Software*, 2021.

[5] Christian R. Trott, Damien Lebrun-Grandié, Daniel Arndt, Jan Ciesko, Vinh Dang, Nathan Ellingwood, Rahulkumar Gayatri, Evan Harvey, Daisy S. Hollman, Dan Ibanez, Nevin Liber, Jonathan Madsen, Jeff Miles, David Poliakoff, Amy Powell, Sivasankaran Rajamanickam, Mikael Simberg, Dan Sunderland, Bruno Turcksin, and Jeremiah Wilke. Kokkos 3: Programming model extensions for the exascale era. *IEEE Transactions on Parallel and Distributed Systems*, 33(4):805–817, 2022.

[6] OpenMP Architecture Review Board. OpenMP application program interface version 4.5, 2015.