Performance Modeling at a Discount

An extended evaluation for the IPDPS 2020 paper submission

This document provides an extended version of the evaluation of the IPDPS 2020 paper submission 'Performance Modeling at a Discount'. In order to ensure the general reproducibility of our results this document provides a detailed explanation of the complete evaluation process.

1 Methodology

To evaluate our new sparse modeling technique and quantify it's accuracy and cost in comparison to our old method we conducted an extensive synthetic data analysis using different levels of noise, parameter-value selection strategies and modeler configurations. Therefore, we ran many different experiments considering 1, 2 and 3 model parameters. We chose to use upto 3 model parameters for the evaluation as this is the maximum number of parameters supported by the old modeling approach due to the high costs it requires. In order to find the optimal trade-off between cost reduction and model accuracy we varied the number of measurement points per parameter $P = \{5, 6, 7\}$ and the number of repetitions per measurement point $R = \{1, 2, 3, 4, 5, 6, 7\}$, while applying $\pm 5\%$ of noise to the generated measurements. For models with 3 configuration parameters we additionally experimented with different levels of noise $n = \{0\%, \pm 1\%, \pm 2\% \pm 5\%\}$ and parameter-value selection strategies, which determine the measurement points used for modeling. Apart from that, we investigated if there is a sweet spot for the number of repetitions per measurement point that counters the effects of noise and if there is an optimal number of measurement points per parameter that increases the model accuracy. Each experiment we conducted was evaluated over 100.000 synthethic test functions, were we analyzed and compared the results of the two model generators. However, for performance models with only one parameter this comparison is not necessary. In this special case the modeling procedure used by the old and the new sparse modeler are absolutely identical and provide the exact same results. Therefore, we simply searched for the optimal number of repetitions per measurement point and the optimal number of measurement points.

2 Environment

In order to conduct all of these experiments under the same conditions we used the following evaluation environment and general definitions.

Synthethic performance functions: For the evaluation of each experiment we generated a set of 100.000 unique synthethic test functions, which we created by instantiating our performance model normal form (PMNF) from Eq. 1 with random coefficients $c_l \in (0.01, 1000)$ and random exponents i_l and j_l, selected from the sets $I = \{0, 1, 2, 3\}$ and $J = \{0, 1, 2\}$. We used a seed for the generation of the coefficients and selection of the exponents, so that we could evaluate different experiment configurations with the same test functions and compare the results.

$$f(x_1, \ldots, x_m) = \sum_{k=1}^{n} c_k \cdot \prod_{l=1}^{m} x_l^{i_{k_l}} \cdot log_2^{j_{k_l}}(x_l) \tag{1}$$

When instantiating the PMNF we created synthetic performance functions such as the one represented by Eq. 2. Each function consists of m terms (m is the number of model parameters), which in turn can consist of a polynomial term, a logarithmic term or both. The behaviour of these different terms can either be additive or multiplicative, as shown by Eq. 2 and Eq. 3. The coefficients $c_0, c_1, \ldots, c_k$ are randomly generated from the interval $(0.01, 1000)$. The equations 4 and 5 show examples for such performance functions with 2 model parameters, while the equations 6 and 7 show examples for functions with 3 model parameters.

$$f(x) = c_0 + c_1(x^i)^{0|1} \cdot (log_2^j(x))^{0|1} \tag{2}$$

$$f(x) = c_0 + c_1(x^i)^{0|1} + (log_2^j(x))^{0|1} \tag{3}$$

$$f(x, y) = c_0 + c_1(x^i)^{0|1} \cdot (log_2^j(x))^{0|1} \cdot (y^i)^{0|1} \cdot (log_2^j(y))^{0|1} \tag{4}$$

$$f(x, y) = c_0 + c_1(x^i)^{0|1} \cdot (log_2^j(x))^{0|1} + c_2(y^i)^{0|1} \cdot (log_2^j(y))^{0|1} \tag{5}$$

$$f(x, y, z) = c_0 + c_1(x^i)^{0|1} \cdot (log_2^j(x))^{0|1} \cdot (y^i)^{0|1} \cdot (log_2^j(y))^{0|1} \cdot (z^i)^{0|1} \cdot (log_2^j(z))^{0|1} \tag{6}$$

$$f(x, y, z) = c_0 + c_1(x^i)^{0|1} \cdot (log_2^j(x))^{0|1} + c_2(y^i)^{0|1} \cdot (log_2^j(y))^{0|1} + c_3(z^i)^{0|1} \cdot (log_2^j(z))^{0|1} \tag{7}$$

Theoretically it is possible to add more than one term per parameter, however while this can increase the model accuracy, it also requires a much larger number of measurements for modeling. Since our work focused on the cases where a cost reduction is actually possible, such functions were out of scope of the evaluation.

Measurements: After generating the syntethic performance functions we then generated the corresponding measurements. Therefore, we evaluated each function in our test set for 5^m parameter value combinations, where $m \in \{1, 2, 3\}$ is the number of parameters, drawn from predefined parameter values for $x = \{4, 8, 16, 32, 64\}$, $y = \{10, 20, 30, 40, 50\}$, and $z = \{2, 4, 6, 8, 10\}$. For functions with 3 parameters, consequently we created 125 measurements points and 625 measurements, when considering 5 repetitions per point. In order to generate the measurements for the selected metric we simply inserted the defined parameter values into the generated functions.

Noise: To simulate the noise one would experience taking performance measurements on a large computing cluster, we ran several identical experiments with different levels of noise $n = \{0\%, \pm 1\%, \pm 2\%, \pm 5\%\}$. Therefore, for each measurement we generated an additional random value $v \in [-1, 1]$. This value represents the relative divergence from the actual measured value. A value of $v = 1$ represents a divergence of the maximum allowed noise level e.g. $+5\%$. A $v = -1$ represents a divergence of the minimum allowed noise level e.g. -5%. By performing this calculation for all measurements we were able immitate realistic system noise that is not uniformly distributed.

Term contribution: Before we used one of the generated functions for the evaluation, the contribution of each term tc in regard to the overall function value was checked. When modeling with noisy data it is important that the contribution of a term is higher than the amount of noise in the system. Otherwise the modeler can not distinguish between system noise and actual application behaviour (signal). Therefore, we only allowed terms contributing at least 3 times the amount of noise, resulting in the following configuration for $tc = \{1\%, 3\%, 6\%, 15\%\}$.

Sparse modeler configuration: The generated measurements are the inputs for the two model generators. While the state of the art modeler requires all of these measurements for modeling, the sparse modeler only uses a small subset of points depending on its configuration. In principle there are 3 major configuration parameters that influence the point selection of the sparse modeler.

- *The single parameter point selection strategy:* determines which points are used to model the single parameter functions for x, y and z (e.g. cheapest, most expensive points).
- *The number of additional measurement points:* specifies the number of additionally used measurement points for modeling the multiparameter function besides the baseline points that were used for creating the single parameter functions (e.g. 0, 1, 2).
- *The multi parameter point selection strategy:* defines which additional points are used to model the multi parameter function (e.g. cheapest, most expensive points).

Based on these configuration parameters the sparse modeler produces different results. For each experiment and all of its functions we saved the respective

result of the two modelers, so that we can analyze them using different evaluation metrics.

3 Metrics

To evaluate the results of the synthetic analysis, we considered two aspects, accuracy and cost.

Cost: To define cost, we try to replicate the scenario of the tool being used to determine the scalability of an application, which is the one most often encountered in practice. We imagine that one parameter, x, represents the number of processes and that the metric we measure is the total runtime per process. The cost of the measurement (e.g., the total core hours to run the experiment) is therefore not the same for all samples. Given that the exhaustive modeler requires all measurements, we consider their accumulated cost to be 100%, and determine which percentage of it the sparse modeler requires.

Accuracy: To define accuracy, we consider the next parameter values in every series (e.g., $x = 128$, $y = 60$, and $z = 12$) and evaluate the resulting model for this combination. Then, we verify whether the predicted value is within $\{\pm 5, \pm 10, \pm 15, \pm 20\}$ % of the actual value.

Additionally, we took a look at a different accuracy measure, the term identification of the modeler. Depending on the number of functions terms one can evaluate if the modeler is able to correctly predict the entire function (we call this an identical model), only the lead orderm term (the major term that influences the function behaviour), or neither of them (we call this an incorrect model).

4 Results

The following sections discusses the results of the synthetic evaluation of the sparse modeling approach. It is structured as the folders in this Git repository. Each section will discuss the findings for the different numbers of model parameters. Then the evaluation is further devided in experiments with different amounts of measurement points per parameter and the repetitions per measurement point. For the evaluation with 3 parameters we additionally provide experiments with different noise levels and parameter-value selection strategies. Each of the sub-folders in the Git repository contains a README file that further describes the organization of the files and how to interpret them. For a graphical representation of the presented data please see the plots in the described sub-folders.

4.1 One model parameter

As described previously, both modelers provide identical results when modeling functions with only one parameter. Since their implementation is the same we focused on finding the optimal number of repetitions and measurement points to increase the accuracy and noise robustness of models with one parameter.

Furthermore, it was not possible to reduce the modeling costs in this case. For one parameter we require a minimum of 5 points in order to be able to model different types of logarithmic and polynomial functions. Using less points for modeling would decrease the costs, however it would also drastically reduce model accuracy and we are not willing to make this trade.

From our previous research we had aquired empirical values for the number of repetitions ($R = 5$) and measurement points per parameter ($P = 5$). The results of our synthethic evaluation for one parameter show that these values were already very good. We found that for 5 points and 1 repetition 77% of models are within noise at the next measurement point with $x = 128$, when modeling with the points $p = \{4, 8, 16, 32, 64\}$. Furthermore, we find the sweet spot for the repetitions which is $R = 4$. With 4 repetitions we can achieve an accuracy of 92%, while 5, 6 and 7 repetitions only marginally increase the accuracy. When we increase the noise band i.e. the divergence limit for the accuracy check, the results are even better. At the same point the modeler predicts 92% of the models correctly within $2\times$ the noise level using only 2 repetitions. Additionally, we also checked scaling of our models at the point with $x = 256$. Again the sweet spot are 4 repetitions, which give us 81.9% correct models within $1\times$ noise. Again when increasing the divergence limit we see an increase in correct models for all number of repetitions.

Then we did the same analysis for $P = 6$ and $P = 7$. For $P = 6$ we check the scalability of the models at the point with $x = 256$. As with 5 measurement points 4 repetitions are the sweet spot and give us 92% correct models within $1\times$ noise. The additional measurement point gives us an overall small increase in correct models. The point where we check the scaling might be larger than the former one, however we also use one additional point for modeling, so that we can check the scalability for the same step size. As before more repetitions help to increase the number of correct models, as the increase of the divergence limit does. When checking the scaling at the next larger point with $x = 512$ we get slighty less accurate results. For 4 repetitions 85.8% of the models are within $1\times$ noise. Though, this are almost 4% more than when using only 5 measurement points.

For $P = 7$ we check the scalability of the models at the points with $x = 512$ and $x = 1024$. We see similar results for our experiments. 4 repetitions are the sweet spot and the second additional point also gives a slight increase in model accuracy. For example at the point with $x = 1024$ 89.8% of the models are within $1\times$ noise when using 4 repetitions. However, for the point with $x = 512$ there was no increase in accuracy, as we still get 92% correct models.

Apart from analyzing the scalability of the models we check if the function terms have been correctly predicted. Reflecting the results from the scalability analysis, the term analysis shows that 4 repetitions are ideal to increase the accuracy while we can drop one repetition to reduce cost. For 5 points we get 76.5% identical functions with 4 repetitions. For 6 points we get 85.1% identical functions and for 7 points 88.6% identical functions.

In conclusion this shows that more measurement points increase the accuracy in the single parameter model case. However, repetitions are preferable over new measurement points, as they cost much less than the next larger point and give a better trade-off between cost and accuracy increase. Also one need to consider that a minimum of 5 points is required in order to be able to correctly identify behaviours like log or $\log^2$. On the other hand it is often difficult to aquire more

than 5 points as additional points often already become very expensive. Kripke, one of the presented case studies provides such an example. In this case the number of processes x needs to be scaled as follows $x = \{8, 64, 512, 4096, 32.768\}$. On many systems it would already be difficult to aquire the 5th measurement point, not to think of a 6th or 7th point. Therefore, we conclude that the quality improvement does not warrant the increase in effort.

4.2 Two model parameters

For 2 model parameters we did the exact same experiments as for 1 parameter, however we also compared the results of the two different modelers. In the plots a 'B' stands for basline, the old modeler, and a 'S' stands for sparse modeler. The number behind the letter indicates the number of points that have been used for modeling. While the baseline modeler requires all points for modeling, the sparse modeler is executed with different configurations. The smalles of these is always the minimum number of points that is required for modeling and the largest the full matrix that is used by the baseline modeler. Furthermore, we varied the number of point per parameter $P = \{5, 6, 7\}$ and the repetitions $R = \{1, 2, 3, 4, 5, 6, 7\}$. For the scalability analysis we now not only scale one, but two axes that we also analyze independant of each other. Therefore, we scale the value of x without scaling y and vice versa. To see all results please take a look at the plots in the sub-folders. Here we only discuss the case when both axes are scaled at the same time, which is also the hardest one for the modeler, as the point it needs to predict is much further away from the data that was used for modeling.

When scaling the models created when using the points $x = \{4, 8, 16, 32, 64\}$ and $y = \{10, 20, 30, 40, 50\}$ to $x = 128$ and $y = 60$ 93.6% of the models are within $1\times$ noise when using 4 repetitions and the old modeling approach. Again 4 repetitions per points seems to be the sweet spot, though in contrast to the one parameter case, more repetitions can also decrease the model accuracy. We then did the same evaluation for the sparse modeler using different configurations. We start by using only the minimum requirement of points for modeling (9 points). With these we achieve 67.6% correct models within noise. This is off cause much less than with the old modeler, however we only used 12% of the cost of the old modeler. When we use 15 points for modeling i.e. the 9 base points and 6 additional points, we can increase the accuracy to 84.1% with 4 repetitions and even to 88.3% with 5 repetitions. Still this configuration of the sparse modeler uses only 17% of the cost of the old modeler. This corresponds to a cost reduction of 83% while we drop only $\approx$ 10% accuracy. When using all available points for modeling the sparse modeler achieves the same results as the old modeler, showcasing that is able to achieve the high accuracy results when given the same data.

We repeated this analysis for different divergence limits $n = \{1\times, 2\times, 3\times, 4\times, 5\times\}$ and the next bigger measurement point with $x = 256$ and $y = 70$. While an increased divergence limit again leeds to more correct models, our models are slighlty less accuracte at the next larger point, what is also to be expected.

The term analysis for models with 2 parameters shows that we need at least 4 repetitions when exposing the modeler to $\pm$5% noise. Otherwise it is not able to accurately predict the correct function terms. With 4 repetitions the old modeler achieves 56.5% identical models and 10.7% lead order terms correct.

This result shows that also the old modeler is not able to correctly predict all possible functions. Again the sparse modeler that is using only the 9 base points is the worst with only 44.4% of identical models. However, in contrast to the scaling analysis, here only one additional point is already enough to increase the number of identical models to 53.5%. More additional points only marginally increase the accuracy while increasing the cost.

The experiments with 6 and 7 points per parameter delivered similar results as for one parameter models. More points per parameter increase the accuracy of the models, however their cost can only be justified if the necessary measurements are either very cheap, or the cost of the modeling process is not critical. More important are the repetitions of each measurement point to counter the affects of noise. Ideally, one takes at least 4 repetitions per point, ideally when the modeling budget allows it even 5.

The accuracy of the sparse modeler increase proportionally with the number of additional measurement points it uses for modeling. In most cases even the minimum required number of points is enough to get a first impression of the application performance. This knowledge can be extended step by step, remodeling with an increasing number of additional measurement points.

4.3 Three model parameters

For models with 3 parameters we followed the same approach, however we also investigated how the results change if we expose the modeler to different levels of noise or use other parameter-value selection strategies to determine the point selection of the sparse modeler. Though, for simplicity we start by taking a look at the results where we varied the number of points per parameter and the number of repetitions as before considering 5% of noise on the measurements.

The baseline, our old modeler, now uses 125 points for modeling and achieves an accuracy of 92.9% with 4 repetitions. When using the same amount of points we also achieve this result with the sparse modeler. We ran the sparse modeler with 5 additional configurations, respectively using 13 (minimum requirement), 14, 14, 25 and 75 points for modeling. The results show that the first additional point gives the biggest increase in model accuracy. However, every additional point helps to further increase accuracy. There is no hard point where the benefit of adding points drops off. Therefore, we used the results of the different modeler configurations as an input for our modeler. By using the accuracy results for the different configurations for 4 repetitions we can model the function of the sparse modeler accuracy in regard to the number of measurement points it uses, and therefore also for its cost. This function is $A = f(p) = 111.59 - 103.01 \cdot p^{-1/3}$ where $f(p)$ is the accuracy and p the number of points used for modeling. Using this function we can determine the optimal configuration for the modeler to reduce cost and retain a high accuracy. The cost of the points used for modeling can be calculated in relation to the old modeler which required all 125 points. The cost of one point is calculated by $f(c) = p \cdot t$, where $f(c)$ is the cost, p is the number of processes and t the time that is the result of model $f(x, y, z) = t$. Since we know exactly which points a specific configuration of the sparse modeler used for modeling, we can calculate the relativ cost of all points in comparison to the full matrix. In conclusion we can achieve a cost reduction of about 85% while retaining 92% model accuracy on the synthetic evaluation data using this method.

As for 1 and 2 model parameters 4 repetitions are the optimum trade-off. For higher cost 6 repetitions offer the overall best results. Interestingly with one additional repetition the accuracy is decreasing again. This leads to the assumption that more repetitions acutally do not help analyzing the noise, rather it leads to modeling it. The other experiments that we conducted for 6 and 7 measurement points per parameter as well as the different points where we evaluated the scaling of the models, indicate the same trends as for 1 and 2 parameters. The scaling is slightly worse when checked at larger values of x, y, z. More measurement points per parameter improve accuracy but are difficult to measure and expensive. Therefore, more repetitions are prefarable.

However, there is one interesting observation we made. When scaling only one axis e.g. $x = 1024$ and leaving all other parameter values at their initial value $y = 10$ and $z = 2$, the sparse modeler achieves much higher accuracy when scaling its models to larger points than the old modeler. The most interesting fact is that the highest percentage of correct models is reached with the least amount of points (the 13 minimum required points for modeling). We explain this by the fact that in order to create the single parameter models the sparse modeler does not use all points. It just uses 5 points per parameter and then combines these points to different multiparameter hypotheses and selects the best of them. When modeling without any addtional points the modeler uses only the information separatly aquired for each parameter. This missing influence of points that entail information of how the parameters interact with each other seams to help to create models that scale very well for a specific parameter along one axis. We see this phenomen for all axis when scaled independantly of the others.

Depending on the values of the parameters and their contribution to the function we also reach different levels of accuracy. For our evaluation functions x is clearly more significant than z for the runtime $t = f(x, y, z)$. We can see this when taking a look at the results of the scaling analysis, where we separatly scaled the x-axis and z-axis. When scaling only the x-axis we reach over $\approx 90\%$ correct models with the modeler configuration S13. When scaling the z-axis we only reach $\approx 85\%$ correct models with S13. Therefore, we conclude that it is more or less difficult for the modeler to accurately model the behaviour of multiple parameters depending on the parameter values where we take the measurements and especially when the growth rate of the measurements is not monotonically increasing.

The term analysis of the evaluated 3 parameter models also delivered interesting results. They show that it is very difficult to correctly identify the terms of a performance function with an increasing amount of noise (with $\pm 5\%$ or above). Models created with less than 4 repetitions are completely inaccurate, receiving only 39% identical models and 19.6% models with correct lead order terms when using all 125 points for modeling. Even though we are able to predict the core function behaviour correctly in about 60% of the cases, this is still a suprisingly bad result. More repetitions do also only slightly increase accuracy by $\approx 1\%$. More measruement points per parameter, however can drasitically increase the number of correctly predicted functions. When using 7 points per parameter and the full matrix as before we can increase the percentage of correct models to $\approx 79\%$, with 68% identical models and 11% correct lead order terms. But despite this results being not optimal, the sparse modeler is able to achieve the same results as the old modeler, and as before it can reduce the cost

by about 85% while retaining 92% model accuracy when we use the priviously determined model accuracy function $A = f(p)$.

Noise levels: For 3 parameter models we also experimented with different levels of noise to see how our new modeler can handle it. We analyzed 4 distinct noise levels $n = \{0\%, \pm 1\%, \pm 2\%, \pm 5\%\}$ for 5 points per parameter. As we already clarified that 5 points per parameter are optimal in most scenarios we skipped the evaluation of different noise levels for 6 and 7 points.

When working with perfect data the sparse modeler shows very promising results. No matter of what number of repetitions we use the baseline modeler achieves an accuracy of 100% over all 100.000 synthetic functions. The sparse modeler with 13 points achieves 93.1% accuracy and with one additional point model accuracy goes up to 100% again. This means for 1.7% of the cost we get perfect models. Even for the next larger point we analyzed the scaling at the modelers still produce the same result, 100% accuracy independent of the number of repetitions used. When scaling only one specific axis all modeler configurations achieve an accuracy of 100%, also the minimum configuration of the sparse modeler S13. The term analysis shows similar results. Using only 14 points we are able to predict the exact models for all functions we generate. There is no point in using only 13 points, as the additional point only is an increase of $\approx 0.1\%$ in cost. However, in contrast S13 only predicts 93.1% exact models for all functions. For the term analysis the number of repetitions also did not matter.

For $\pm 1\%$ of noise we see similar effects as for 5% regarding the number of repetitions. The baseline modeler achieves an accuracy of 94.5% with 4 repetitions. The sparse modeler achieves an accuracy of 71.3% when using only the minimum number of required points for modeling and an accuracy of 89.9% when using only one additional point. In contrast to measurement with 5% noise here one additional point is enough to retain $\approx 95\%$ accuracy. At the same time S14 only uses $\approx 1\%$ of the cost of the baseline modeler. So for measurement with $\pm 1\%$ noise we can effectively reduce the cost of the modeling process by a factor of 100. Apart from that the term analysis also shows very good results. B125 predictes only 11% of all models incorrectly (when using $R = 4$). The sparse modeler with one additional point (S14) predictes only 11.2% of the models incorrectly. The decrease in accuracy is absolutely neglectable when compare to the amount of cost reduction we achieve.

For $\pm 2\%$ of noise we get similar results. While one additional point still offers the biggest increase in model accuracy, S14 is further away from the baseline. Therefore, we can conclude that with an increasing amount of noise we need more additional points to reach the accuracy of the baseline. For B125 with $R = 4$ we reach an accuracy of 93.5% and for S14 an accuracy of 83.3%. This are only 89% of the baseline accuracy a decrease of 6% compared to our experiments with $\pm 1\%$ noise. Similar changes we see in the term analysis. Where S14 now predicts 23.4% of the models incorrectly. However the sparse modeler is not far away from the baseline as B125 has an error rate of 22.3% (when using $R = 4$).

Despite the different amounts of noise the sparse modeler has to work with we still see the same phenomena as before when scaling only one axis. The sparse modeler with the least amount of points still produces the best scaling

models for one specific axis, no matter if we apply ±1%, ±2% or ±5% of noise.

Parameter-value selection strategies: As described the parameter-value selection strategy determines the points the sparse modeler uses for modeling. In total there are two configuration parameters for the modeler that determine which points are used for modeling. The number of additional points, the single parameter selection strategy and the multiparameter selection strategy. For a more detailed explanation of them please see Section 2. With the privious experiments we already covered the number of additional points. We also invetigated the mutli parameter selection strategies (for ±5% of noise). Therefore, we choose two different strategies. First, we added additional points for the creation of the multiparameter model with increasing cost (smallest points first). Second, with a decreasing cost (most expensivest points first). We expected that the more expensive additional points have a more positive affect on the scaling of the created models and this is also the case. When using the most expensive points for modeling with S14 we achieve an accuracy of 83.5%. With S14 and the cheapest points we achieved an accuracy of 71.8%. These 12% improvement in accuracy are very good, but the problem is that only the first additional point costs us ≈ 50% of the entire matrix. When filling up with the cheapest point in average we can use more than 70 additional points and still use only about 15% of the total matrix cost, also giving us better results. So what matters is the fact that we use additional points for modeling that are not part of the points used to explain the behaviour of the single parameter functions that are combined to the mutli parameter model. Therefore, selecting additional measurement points with decreasing cost is not an option.

5 Conclusion

The evaluation showed several important things.

First, we could refine our emperical expecation from our previous work for the optimal number of measurement points per parameter and the optimal number of repetitions per measurement point. The results of the evaluation show that repeating each measurement 4 times provides an optimal cost/benefit ratio in most scenarios. For models with only one parameter even 2 repetitions can be enough. Generally, the number of repetitions should be adjusted to the level of noise on the system where the measurements are taken. Furthermore, an anlysis of the repetitions for each point can identify outliers, and their exclusion from the modeling process can in turn increase accuracy.

Second, we found that more measurement points per parameter are benefitial for the model accuracy. However, in most cases these are much more expensive than additional repetitions per point or additional measurement points with smaller parameter-values. Both of the latter are preferable to larger additional measurement points to model the single parameter functions. One has to decide the window in which the models should be accuracte. Then one needs to choose the parameter-values for the measurement points accordingly. Furthermore, one can decide if the accuracy with 5 points per parameter is enough or the increased model cost justifies the gain in accuracy for the scaling for points that are not used for modeling. Though, in general the quality improvement does not warrant the increase in effort.