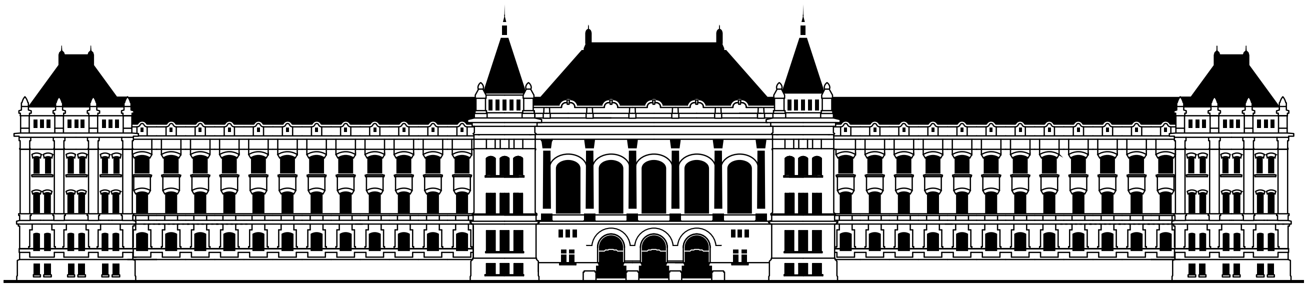
Proceedings of the 25th Minisymposium

of the

Department of Measurement and Information Systems
Budapest University of Technology and Economics

(Minisy@DMIS 2018)

January 29, 2018
Budapest University of Technology and Economics
Building I



Budapest University of Technology and Economics
Department of Measurement and Information Systems

Head of the Department: Tamás Dabóczi

Conference chairman:
Béla Pataki

Organizers:
Vince Molnár
Dávid Honfi
Szilárd Bozóki

Homepage of the Conference:
`http://minisy.mit.bme.hu/`

Sponsored by:

Schnell László Foundation

FOREWORD

This proceedings is a collection of the lectures of the 25th Mini-Symposium held at the Department of Measurement and Information Systems of the Budapest University of Technology and Economics. At the beginning, starting 25 years ago, the main purpose of these symposiums was to give an opportunity to the PhD students of our department to present a summary of their work done in the preceding year. It was an interesting additional benefit that the students got some experience: how to organize such events. Beyond this goal, it turned out that the proceedings of our symposiums give an interesting overview of the research and PhD education carried out in our department. Some years ago, the scope of the Minisymposium had been widened, foreign partners and some of the best MSc students were also involved. This was a real benefit, therefore, this year we have kept this widened scope. By this year, the event turned to be a small conference with an open call for papers instead of a PhD students' symposium, but the traditional name is kept to represent the origins.

The lectures reflect parts of the scientific fields and work of our department, but we think that an insight to the research and development activity of ours and our partner departments is also given by these contributions. Traditionally, the symposium was focused on measurement and instrumentation. The field has slowly changed, the scope widened during the last few years. New fields mainly connected to embedded information systems, new aspects e.g. dependability and security are now in our scope of interest as well. Both theoretical and practical aspects are dealt with.

During this twenty-five-year period there have been shorter or longer cooperation between our department and some universities, research institutes, organizations and firms. Some research works gained a lot from these connections. In the last year the cooperation was especially fruitful with the European Organization for Nuclear Research (CERN), Geneva, Switzerland; Vrije Universiteit Brussel Dienst ELEC, Brussels, Belgium; Robert Bosch GmbH., Stuttgart, Germany; Department of Engineering, Università degli Studi di Perugia, Italy; National Instruments Hungary Kft., Budapest; University of Innsbruck, Austria; University of Geneva, Italy; and University of Florence, Italy.

We hope that similarly to the previous years, this Minisymposium will also be useful for the lecturers, for the audience and for everyone who reads the proceedings.

Budapest, January, 2018

Béla Pataki
Chairman of the
Minisymposium

Papers of the Minisymposium

Author	Title	Page
Borkó, Máté and Bolgár, Bence and Sarkozy, Peter	Basecalling Raw Nanopore DNA Sequencing Reads using Neural Networks	4
Bozóki, Szilárd and Pataricza, András	Use of Resource Leveling in Peak Workload Scheduling	8
Bruncsics, Bence and Gézsi, András and Antal, Péter	Unknown, Uncertain, Untrue: Challenges in Inference Using Semantic Life Science Data	12
Csurcsia, Péter Zoltán and Schoukens, Johan and Peeters, Bart	An Application Example of Regularization: Time-Varying Operational Modal Analysis	16
Farkas, Rebeka and Bergmann, Gábor	Towards Reliable Benchmarks of Timed Automata	20
Graics, Bence and Molnár, Vince	Mix-and-Match Composition in the Gamma Framework	24
Guenfoud, Zeyneb and Antal, Péter	Adaptive Sequential Laboratory Diagnostic Tests: Joint Bayesian Analysis for Optimality	28
Hajdu, Ákos and Micskei, Zoltán	A Preliminary Analysis on the Effect of Randomness in a CEGAR Framework	32
Honfi, Dávid and Micskei, Zoltán	Towards Supporting Dynamic Symbolic Execution via Multi-Domain Metrics	36
Jagyugya, Erik and Sarkozy, Peter	Comparison of Nanopore DNA Sequencing Basecallers on Whole Human Data	40
Klenik, Attila and Pataricza, András	Preliminary Performance Assessment of Hyperledger Fabric	44
Nagy, Péter and Jobbágy, Ákos	Oscillometric Blood Pressure Measurement Using Constant Cuff Pressure Intervals	49
Rabatin, Gábor and Vörös, András	Towards the Verification of Neural Networks for Critical Cyber-Physical Systems	53
Šeketa, Goran and Dzaja, Dominik and Vugrin, Jurica and Lackovic, Igor and Magjarevic, Ratko	Signal Acquisition for Accelerometer-Based Fall Detection	57
Varga, Balázs and Orosz, György	High Frequency Active Distortion Cancellation	61
Verbeke, Dieter and Schoukens, Johan	Improved Frequency Response Measurements Using Local Parametric Models	65

Basecalling Raw Nanopore DNA Sequencing Reads using Neural Networks

Máté Borkó, Bence Bolgár, Peter Sarkozy
Department of Measurement and Information Systems
Budapest University of Technology and Economics
Budapest, Hungary
borkomate@gmail.com

***Abstract*—The single-molecule real-time (SMRT) DNA sequencer developed by Oxford Nanopore Technologies offers breakthrough read lengths in a handheld device, while greatly simplifying input DNA library preparation procedures. The standard method of identifying the individual bases passing through each pore relies on a Hidden Markov Model (HMM), mapping raw current levels to individual bases in a nonlinear, multi-staged fashion. Recent advancements in artificial neural networks (ANN) and related natural language processing (NLP) techniques allow novel neural architectures that may improve the accuracy of the basecalling. We developed and examined a novel deep neural network based method to perform basecalling on raw current level measurements, as well as an efficient method of selecting and curating a training database from a set of real measurements.**

***Index Terms*—DNA sequencing, neural network, LSTM, Nanopore**

I. Introduction

Oxford Nanopore Technologies's (ONT) MinION device uses a new single-molecule real-time technique to read individual DNA sequencing. The greatest achievement of this effort is the ability to sequence a single molecule of DNA without resorting to enzymatic amplification of the strand. ONT makes unprecedented read lengths possible, up to tens of thousands of bases long, compared to several hundred bases long provided by current technologies. One disadvantage of this new platform is that the basecalling accuracy (the ratio of correctly identified nucleotides in a sequence) is an order of magnitude lower than other competing platforms. Sequencing is performed by passing each individual single stranded DNA molecule through an engineered nano-scale pore, along with a ratcheting helicase enzyme to slow the passing of the molecule to levels where the characteristic current of the nucleotides inside each pore can be captured. The current is sampled at 4 kHz, while the average traversal rate of the DNA strand is approximately 300-500 nucleotides per second. The current level is representative of multiple nucleotides (up to 6), thus currently Hidden Markov Models are used to perform the basecalling.

The use of neural networks in the field of DNA sequencing is a recent phenomenon, with only a few existing solutions [3]. The performance of deep neural networks in natural language processing (NLP) gave us a good starting point, as they share a strikingly similar underlying model from a signal processing aspect. We aim to establish a high quality training set for future neural network basecallers, and to identify an efficient and NN architecture for basecalling. A successful neural network basecaller architecture requires two vital components:

- The accuracy of any neural network is inherently limited by the amount and quality of available training samples.
- An appropriate neural network structure must be declared. The number of layers and neurons must be selected according to the required result, and is governed by a suitable cost function which allows the optimization of the structure with respect to training time and accuracy.

II. Creating Training Samples

A large, publicly available dataset, in the form of the whole genome shotgun sequencing of the NA12878 human sample [4] was used as a training set. This multi-TB dataset contains the entire human genome at approximately 30x coverage, and provides a wide variety of training data. We used the GRCh38 reference human genome as a gold standard, as its deviation from the true NA12878 is orders of magnitude lower than our expected error rates.

A. Original Mapping Procedure

Initially, we attempted to implement a method similar to how Metrichor [5], the vendor-released basecaller maps input signals to DNA sequences. This 3-step method functions as follows:

- The raw data files containing the time-current measurements are separated into contiguous segments with no changes in current called events.
- The basecaller assigns a k-mer probability to each event, using a HMM.
- The kmer probabilities along with the HMM step and stay probabilities are calculated to assign a final basecalled sequence and a PHRED score

The CIGAR string describes the alignment between the read and the reference sequence, noting soft clipped (S), matched (M), inserted (I) and deleted (D) bases. The process is summarized on Fig. 1.

B. Sorting Training Data

Using the previously described method we were able to assign raw data samples to reference bases. Initially, we

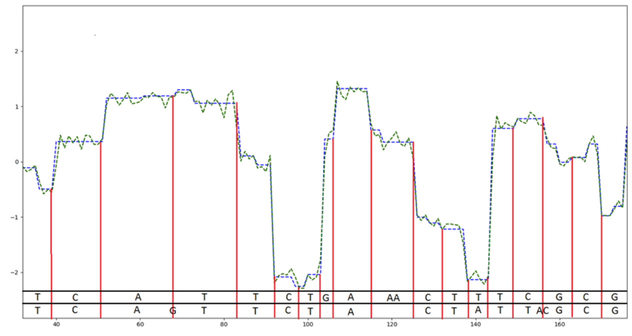


Fig. 1. *Abstraction levels during basecalling. Raw current levels are shown in green, event current levels in blue. Red columns denote each event start. The vertical axis shows raw normalized current, and the horizontal axis shows the reference sequence (not time scaled).*

attempted training the neural network directly with these segmented signals, but further research suggested an architectural element, Connectionist Temporal Classification (CTC) [10] that made data segmentation unnecessary.

All raw data files contained an offset to the start of the segmentation, but the duration of some events were not defined correctly, so the raw data-to-event mapping could not be executed trivially. We corrected the durations by calculating them from the start and stop times of each event. The results of each measurement are stored in $fast$ files. These files contain all relevant measurement related metadata, e.g. the event segmentation, offset, duration, temperature, and environmental variables that could be transferred into the neural network model. We extracted the $fastq$ sequences from the $fast5$ files, and mapped them to the reference sequence using the Burrows-Wheeler Aligner (BWA, [6]) and Samtools [7]. The aligner reports quality information about each alignment, which we used to identify candidate reads for use as training samples. Sequences that aligned to the positive strand of the genome with a mapping quality threshold above 30 were used as training samples. Additionally, if the CIGAR string of an alignment indicated the presence of deletion more than 5 bases long, the read was discarded from the training set [8].

C. *Final training set*

During the first phase of our research, we examined about 300 GB of raw measument data, originating from approximately 40000 reads. We initially selected 160 MB, or 1971 training sequences, to be representative of the entire set. The training data consists of raw current signals and reference nucleotide sequence pairs as inputs and outputs, respectively. These files were sorted by length, as zero padding of shorter sequences is required for efficient mini-batch training [12]. A neural network trained with these samples is capable of translating raw current signals into nucleotide sequences, as per the goal of our research. We normalized the samples before storage.

III. Selecting the Appropriate Architecture

Recent research has created new artificial neuron models, where the output of a neuron is the outcome of a more complex process. In this case the output depends on the last n (hidden)states of the neuron and on the last input sample, so an output at the time point t_i carries the effect of all previous inputs indirectly. A networks built from such elements are called Recurrent Neural Networks (RNN) [13]. The stored n states enables the handling of the time dependencies in the input sequence.

A. *LSTM*

During the backpropagation (training) RNN cells cannot prevent exponentially increasing gradients. The errors have to be backpropagated through all hidden neurons (states), which is not possible, if the gradients grow to the infinity. As a result, RNN structures allow only a finite number of hidden states, thus the time dependency that can be handled by RNNs is shorter than what is required for basecalling.

Long-Short Term Memory is an architecture that can handle such long distance time dependencies [9]. In this model the neurons are replaced with small memory cells, and these cells have control over the output. Through this control they can prevent the problem of exponentially increasing gradients.

The functionality of these cells can be described with 6 variables:

- g is the input
- i is the input gate. Its value is multiplied by the input, so it functions as a weight.
- f is the forget gate. It controls the influence of the hidden state.
- o is the output gate. It weights the value of the output
- s represents the state of the cell.
- h symbolizes the output.

Furthermore W are the related weight matrices and b the biases for every state variable. The activation function, represented by σ is a $tanh$ function. The equations that describe the functionality of an LSTM cell:

$$\begin{aligned}
g_t &= \sigma(W^{gx}x_t + W^{gh}h_{t-1} + b_g) \\
i_t &= \sigma(W^{ix}x_t + W^{ih}h_{t-1} + b_i) \\
f_t &= \sigma(W^{fx}x_t + W^{fh}h_{t-1} + b_f) \\
o_t &= \sigma(W^{ox}x_t + W^{oh}h_{t-1} + b_o) \\
s_t &= g_t \cdot i_t + s_{t-1} \cdot f_t \\
h_t &= \sigma(s_t) \cdot o_t
\end{aligned} \tag{1}$$

B. *Bidirectional Networks*

Alongside LSTM, bidirectional neural networks [11] have become the most important development of RNN structures, as these have been used to recognize handwritten letters. Bidirectional cells contain two hidden neurons that are in connection with the input cells and also with the output cells. One of them has a recurrent signal from the input layer and the other one from the output layer. This structure can learn the sequence from the both directions, and predict the output as the outcome of the previous and next time samples. Naturally, this functionality enables only offline processing.

C. CTC

The core of the processing model is the CTC cost function [10]. The usage of CTC makes input data segmentation unnecessary in the preprocessing phase. CTC cost functions with bidirectional LSTM structures have outperformed Hidden Markov Models in many applications [10].

The brief functional description of CTC is as follows: Alongside the standard output labels (here A, G, C, T) CTC introduces a "blank" label (b), and fills the desired output sequence with these blank labels. E.g. the sequence "$ACGT$" would be "$bAbCbGbTb$". This is the extended output. In next step, a two dimensional graph is declared, which contains a number of nodes equal to the input sequence length in the horizontal directions and extended output length number nodes in the vertical direction. The applied neural network predicts the probability of each label (A, C, G, T, b) in every time step. There are permitted and prohibited transitions regarding the output sequence. E.g. in the sequence "$bAbCbGbTb$" the prediction can start only with b or A and if a A has been predicted, the next predictions can be A or b or C, but G can not follow A, because it would cause loss (C would be eliminated from the output). Similar to Hidden Markov Models, two types of variables are used to model the state of the predicted probabilities. A forward and a backward variable for the forward and backward traversal of the graph. Finally, the blank labels are removed from the predicted output sequence, and depending on the decoder style, the repeated labels will be collapsed.

IV. Results

We initially started the formulation of the neural network architecture without any specific restraint, and created models 1-2 hidden layers with a number of neurons between 50 and 1000. We used a single test sample to configure the initial model structure, and the length of the input (raw current level) at 9000 samples caused two main issues: a single training iteration took longer than two minutes, and the CTC architecture required the entire input sequence, which taxed the memory subsystem of our hardware. We only used input sequences that were shorter than 4000 samples in the current domain to decrease training times. In order to select the optimal number of neurons in each layer, we analyzed the relationship between the raw current vector length and the output nucleotide sequence. We used the reference sequence to determine that each single nucleotide corresponds to approximately 11 raw current samples. However, the effect of a single nucleotide on the current inside of each pore extends further, to approximately 20 nucleotides [14]. This results in approximately 220 raw current samples, and thus 200 neurons were chosen to the output per time step. This network used a mixture of LSTM and bidirectional neural networks with 100-100 neurons in the forward and backward layers.

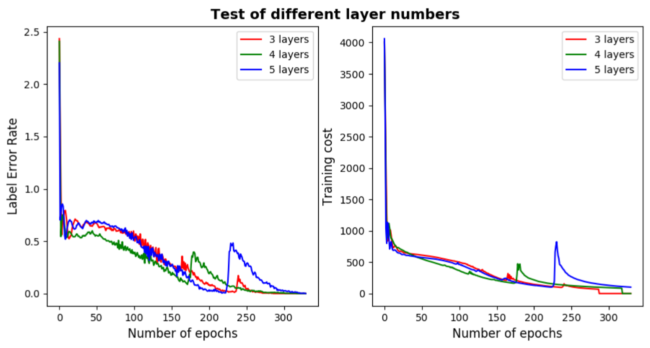


Fig. 2. *Training with different layer sizes*

Networks with only 1 or 2 layers were unable to learn the training data, so we increased the depth of our model. The results of 3, 4, 5 layers are shown in the Tab. I and Fig. 2.

TABLE I
LAYER TEST RESULTS

Number of layers	Epochs	Training time [s]
3	287	2348.37
4	318	3681.05
5	330	4743.70

We determined that using 3 layers, the accuracy of the network depended highly on the initially randomized weights, while with 5 layers, the training time was unnecessarily long. Thus a 4 layer model was chosen. Further results all used 4 layers with 200 neurons per layer structures with the ADAM optimizer and CTC loss function. The performance on unidirectional networks were also investigated, so we compared the reference network with a 4 layer feedforward neural network using 200 neurons per layer. As the Fig. 3 shows it performed markedly worse. It was unable to learn the reference sequence in 1000 iterations. Thus we concluded that unidirectional networks do not have sufficient performance for applying them in basecalling raw nanopore current levels.

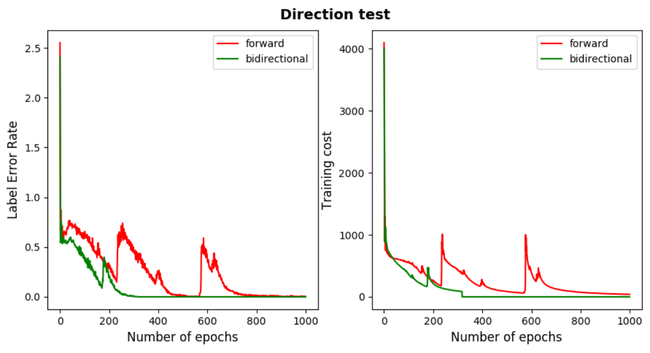


Fig. 3. *Comparison of forward and bidirectional networks*

To enhance the power of our network we implemented almost all of the optimization opportunities provided by the Tensorflow framework. We used features such as dropout training, using mini batches, shuffling the order of the training data and layer normalization. Dropout and shuffling the order of training data help avoid getting stuck in local minima and overfitting. Mini batches aid in more robust convergence. The optimal batch size is yet to be determined, as we currently use one sample per batch (batch size 1). Finally, layer normalization speeds up convergence and improves the achieved accuracy. All implementations used the shuffling of the order of training data. The Fig. 4 shows the result of dropout training.

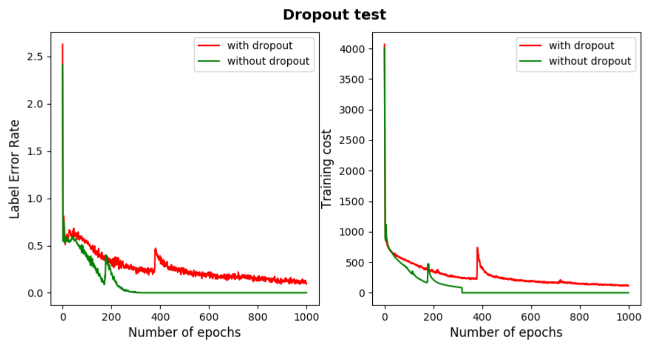


Fig. 4. *Dropout test*

As it is shown in the Fig. 4, we experienced, that the dropout stabilizes the performance of the network, even so we used it in further models. We initially had 34 training samples with the input lengths shorter than 4000. The first acceptable network was trained on these samples. The hyperparameters: 4 bidirectional layers with 100-100 neurons, dropout 0.5, learning rate 0.003. The training samples were divided into two parts, training samples (27) and test samples (7). The results are shown Tab. II. During the training we got 5% error rate on the training samples. We analyzed a larger subset of the entire raw dataset to collect more samples meeting our criteria. This increased our training set to 79 sequences shorter than 4000 samples, and were divided into training and test data by a ratio of 64/15. The structure of the network was not changed. The results in Tab. II show an accuracy of 4.8% on the updated training samples.

TABLE II
TRAINING RESULTS

Number of training samples	27	64
Number of test samples	7	15
Average match rate	0.617	0.679
Average mismatch rate	0.112	0.084
Average insertion rate	0.198	0.134
Average deletion rate	0.072	0.104
Average label error rate	0.383	0.321
Training time	7 days 6 h	13 days

With respect to the label error rate, this nets an additional 5% improvement, resulting in a 68% final accuracy. While this accuracy is lower than the best currently available basecallers (up to 90%), it is still a promising result considering the small number of training samples.

V. Conclusion

Although our model has not yet reached the accuracy of Albacore yet, we have not found any reason why it could not be achieved. The reached 68% is not a record-breaker, but we must mention that the number of training samples used is extremely low. Our only major hurdle appears to be very long network training time, and we must explore further options to decrease it. To further increase the accuracy of our model, we plan to use longer training sequences. Adding longer sequences to the network directly has dire consequences in terms of memory usage and increased training times. Thus we plan to segment the data into overlapping partitions with a sliding window, allowing the use of arbitrary length raw sequences as training data. We also plan to investigate the addition of a convolutional layer into the model architecture, as such a mixture of layers often give excellent results in practice [10].

VI. Acknowledgements

The authors acknowledge that they are participants in the Oxford Nanopore Technologies's MinION Access Program, and have no other conflicts of interest. This research was supported by the OTKA-K-112915 Grant, and the Multipurpose Health Monitoring Platform bilateral Croatian-Hungarian grant. The Titan Xp used for this research was donated by the NVIDIA Corporation.

References

[1] Oxford Nanopore Technologies. Electronics for nanopore sensing. https://nanoporetech.com/how-it-works 05-20-2017
[2] Mikheyev AS., Tin MM., A first look at the Oxford Na- nopore MinION sequencer. Mol Ecol Resour 14(6): p. 1097-102. DOI 10.1111/1755-0998.12324 2014
[3] Wick, RR., Holt, KE., et al.: Comparison of Oxford Nanopore basecalling tools. https://github.com/rrwick Basecalling-comparison 11-09-2017
[4] Nanopore Whole Genome Sequencing Consortium, https://github.com/nanopore-wgs-consortium/NA12878 11-15-2017
[5] Oxford Nanopore Technologies, https://nanoporetech.com/ 2017-12-20
[6] Li H., Durbin R.: Fast and accurate long-read alignment with Burrows-Wheeler transform. Bioinformatics, 26, 589-595. 2010
[7] Li H., et al: 1000 Genome Project Data Processing Subgroup (2009) The Sequence alignment/map (SAM) format and SAMtools. Bioinformatics, 25, 2078-9. [PMID: 19505943]
[8] Sarkozy P., Antal, P., Jobbágy, Á.: Calling Homopolymer Stretches from Raw Nanopore Reads by Analyzing k-mer Dwell Times. 2016. DOI: 10.1007/978-981-10-5122-7-61.
[9] Schmidhuber, J., Hochriter, S.: Long Short-term Memory. 1997. http://www.bioinf.jku.at/publications/older/2604.pdf, doi: 10.1162/neco.1997.9.8.1735
[10] Graves, A.: Supervised Sequence Labelling with Recurrent Neural Networks, chapter 7. Connectionist Temporal Classification. 2012. doi: https://doi.org/10.1007/978-3-642-24797-2.
[11] Schuster M., Paliwal, KK., Bidirectional Recurrent Neural Networks , IEEE TRANSACTIONS ON SIGNAL PROCESSING, VOL. 45, NO. 11, NOVEMBER 1997, doi: 10.1109/78.650093
[12] Bottou, L.: Large-scale machine learning with stochastic gradient descent. In COMPSTAT2010 (pp. 177-186). Physica-Verlag HD. 2010
[13] Hopfield, J. J. (1982). Neural Networks and Physical Systems with Emergent Collective Computational abilities. Proc. Natl. Acad. Sci. USA,79, 2554-2558.
[14] Teng, H., et. al.: Chiron: Translating nanopore raw signal directly into nucleotide sequence using deep learning, doi: https://doi.org/10.1101/179531

Use of resource leveling in peak workload scheduling

Szilárd Bozóki
Department of Measurement and Information Systems
Budapest University of Technology and Economics
Budapest, Hungary
bozoki@mit.bme.hu

András Pataricza
Department of Measurement and Information Systems
Budapest University of Technology and Economics
Budapest, Hungary
pataric@mit.bme.hu

***Abstract*— High availability is not anymore a designated property of critical applications, but an important quality of service requirement even for common applications. This way, minimizing downtime, even under extremely high workload, has been a fundamental requirement. Capacity planning is the art of allocating sufficient and necessary resources to the applications. Achieving both is even more complex when the workload is extremely bursty.**

Overestimating the workload in the case of peaks leads to inefficient resource utilization and all the financial drawbacks originating in it. Frequently, the designer has an influence on the end-to-end process, including the shaping of the workload.

This paper investigates basic workload-resource leveling mechanisms in the context of a bursty workload environment.

Keywords—Extreme Value Analysis, peak workload management, burstiness, workload shaping, system dimensioning

1 INTRODUCTION

The phenomenon of system failure due to overloading has been around since humanity started using tools and machines. Consequently, over time, multiple different solutions have been developed to manage this issue (e.g. over-provisioning, queueing, auto-scaling, etc.) [1] [2].
Among others, estimating the characteristics of the system load (using models) and using the estimated maximum value as a basis for system dimensioning has been a very basic idea. However, modeling the maximum (peak) workload is far from trivial in many cases.
Dimensioning a system to an overestimated peak workload could lead to a waste of resources due to underutilization. This is especially true in the case of a workload ridden with rare but highly bursts. Here the deviation from the average workload is extremely large, necessitating a number of resources multiple times larger than in the average case. However, as the peak bursts are rare, their worst case values serving as a basic input for dimensioning is extremely hard to predict.
This paper analyses two basic peak workload mitigation strategies: (1) 5-day rolling average peak mitigation, (2) *first-in-first-out* (FIFO) queueing with a constant maximum capacity and infinite queue length, using R with several specialized packages [3] [11] [12] [13] [14].

In order to highlight the differences between low variance and high variance (bursty) workload, we evaluated two approaches to dimensioning: Facebook Prophet, a sophisticated approach aiming at dimensioning by focusing on the average load, and Extreme Value Analysis (EVA) targeting the worst case (peak workload).

1.1 Prophet

Facebook is a non-mission critical application with over 2 billion users across the globe served by a world-spanning extremely large infrastructure. The geographical distribution of its user base acts as a natural load balancer and statistical multiplexing can be assumed, owing to the large number and heterogeneity of its users.
Consequently, Facebook Prophet was designed for efficient resource utilization in focus, which in effect suppresses the outliers. Due to the nature of Facebook, this can be acceptable because availability or timeliness guarantees of the service under a peak workload are highly relaxed.
The heart of Facebook Prophet is a decomposable model aimed at modeling the following mutually relatively independent factors [9] [10]:

(1) Trends: increasing market penetration (e.g. shops, restaurants etc.) and growth in the user base
(2) Seasonality: weekly, monthly, yearly (e.g. going out with friends or family during the weekend, summer vacations etc.)
(3) Holidays and events (e.g. Christmas, Black Friday, Presidential election in the USA)

CI	Max / average load	Availability
95,00%	8,5	98,39%
96,00%	8,7	98,53%
97,00%	8,9	98,59%
98,00%	9,5	98,93%
99,00%	9,5	98,93%
99,90%	11,1	99,20%
99,99%	11,8	99,20%

1. Table Dimensioning with Facebook Prophet

Dimensioning the system based on the worst case scenario of Facebook Prophet requires around 12 times the average capacity and can achieve 99% availability (Table 1).
Considering critical applications requiring four nines or more availability, the 99% availability of Facebook Prophet is insufficient, as the non-peak periods over dominate the sample from the point of view of the algorithm estimating the necessary resources.

1.2 EVA

EVA is a branch of statistics aimed at modeling the extremely deviant values (the values usually labeled as outliers in traditional model fitting context) [4] [5] [6].
The classic question of EVA, originating from hydrology, is as follows: *"What is the maximal expected level of a flood in the next 100 years based on historical data?"* When using EVA, a usual answer is the *return level* $N\ l(N)$, which is the maximal level that is expected to occur once every N time intervals (100 years in the case of the example), along with the respective *confidence intervals* (CI).
Let $\chi = \langle X_1, X_2, \dots, X_n \rangle$ be a sequence of independent and identically distributed random variables with cumulative distribution function F and let $M_n = \max(X_1, X_2, \dots X_n)$ denote their maximum. Essentially, EVA aims at analyzing M_n.

$$P(M_n \leq y) = P(X_1 \leq y, X_2 \leq y, \dots X_n \leq y) = \prod_{i=1}^{n} P(X_i < y) = [F(y)]^n$$

Equation 1. Computing the exact distribution of the maximum if F is known

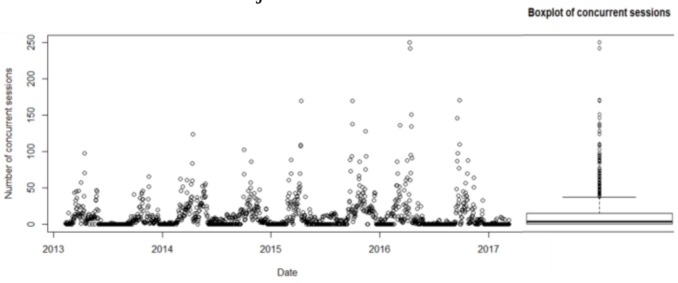
Provided F is known, the exact distribution function of M_n could be computed, due to the independence of the random variables X_i (Eq. 1).
However, when F is not exactly known (estimated, or approximated), computing $[F(y)]^n$ potentially ends in useless results because the otherwise relative small estimation errors are multiplied. In contrast, EVA takes a different approach by approximating $[F(y)]^n$ instead of F.
Essentially, there are two EVA models: the *block maxima* model backed by the *Fisher–Tippett–Gnedenko* theory, and the *threshold exceedance* model backed by the *Pickands–Balkema–de Haan theory* [4].
In our analysis, the threshold exceedance model was used, because it makes better use of a limited dataset. Threshold selection was based on visual exploratory analysis. [7] [8]

2 VCL

The first analyzed dataset corresponds to the real-life workload of a shared resource: a Virtual Computing Lab (VCL) operated by the department between 2013 and early 2017. The workload is computed by summarizing the concurrent virtual machine reservations in the system reservation log.
The VCL has two distinct purposes:

a) IaaS platform for the various student labs and research projects
b) Homework submission platform for a core course with a large number of students

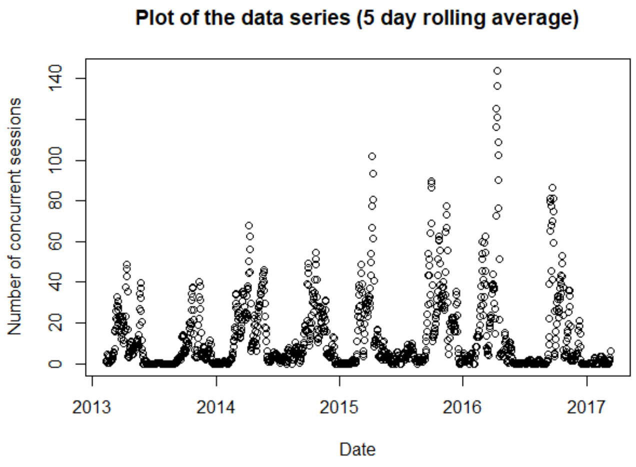
The workload originating from the IaaS offering has moderate variance in the workload due to the managed scheduling of the laboratories and the computationally heavy research projects.
On the other hand, the homework submission system is prone to extreme workload spikes because the majority of the students submit their homework just before the deadline.



1. Figure Visualizations of the VCL data series

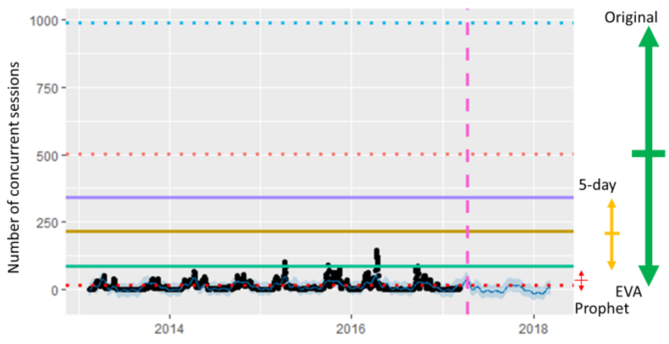
To sum up, the overall workload consists of a background load and spikes recurring every semester, with the largest spike being around 250 concurrent sessions. (Fig 1.)

2.1 5-day smoothing



2. Figure Visualization of the VCL data series with 5-days rolling average applied

Applying a 5-day rolling average to the workload reduced the amplitude of spikes as expected, with the largest peak reduced from 250 to 140 concurrent sessions. (Fig 2.)
With the original and the smoothed dataset at hand, Prophet and EVA were conducted. For demonstration, the $N = 100$ days return level is presented. Figure 3 represents the predictions of Prophet based on the smoothed dataset. With Prophet repressing the outliers, its predictions are clearly off the marks regarding peaks.
The 100 days return level of the original dataset expects a workload around 502 concurrent sessions with 16 concurrent sessions at 95% lower CI bound and 988 concurrent sessions at 95% upper CI bound. On the other hand, the smoothed dataset resulted in a 100 days return level of 215 concurrent sessions with 88 concurrent sessions 95% lower CI bound and 342 concurrent sessions upper 95% CI bound. (Fig. 3)

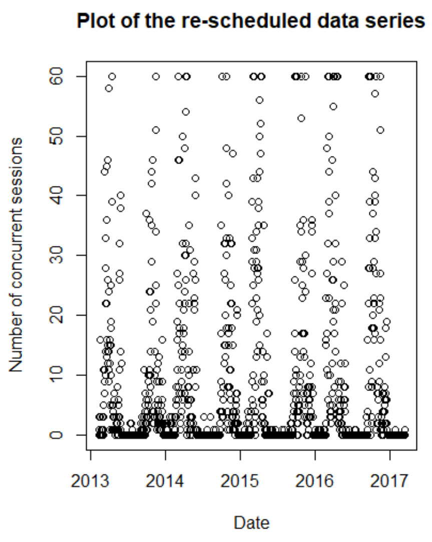


3. Figure Comparing the different cases

Comparing the EVA 100 days return levels results, the required static capacity can be reduced by a factor around 2x-3x by using a simple 5-day rolling average peak workload mechanism.

2.2 Queuing

The queueing mechanism was simulated by linearly parsing the system reservation log, and putting all the tasks in a FIFO that would exceed the predefined maximum capacity.
Essentially, the reservations where re-scheduled and the delay was computed from the difference between the original start time and the new start time.



4. Figure Plot of the re-scheduled data series with maximum capacity set to 60 concurrent sessions using FIFO queuing

The effects of queuing become clearly visible when the new data series is plotted. (Fig 4.)

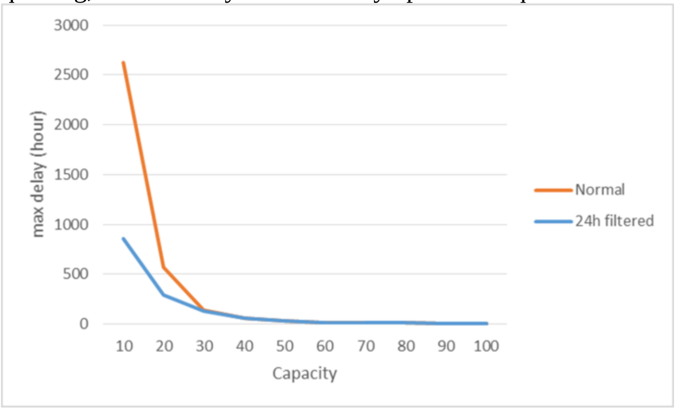
max. capacity	max. delay(h)	total delay(h)	average delay(h)
10	862,03	3976586,14	125,058
20	291,66	785925,72	24,716
30	136,11	324575,07	10,207
40	58,95	149764,04	4,710
50	35,44	70412,73	2,214
60	19,64	39209,23	1,233
70	13,24	21169,35	0,666
80	10,76	12923,49	0,406
90	8,75	7851,06	0,247
100	7,25	5052,87	0,159
110	6,13	3284,56	0,103
120	5,00	2141,47	0,067
130	4,25	1393,62	0,044
140	3,49	892,36	0,028
150	2,75	546,29	0,017
160	2,45	351,98	0,011
170	2,00	240,20	0,008
180	1,50	166,26	0,005
190	1,25	116,17	0,004
200	1,00	80,42	0,003
210	0,77	52,14	0,001
220	0,75	30,14	0,001
230	0,50	13,81	0,000
240	0,25	2,09	0,000
250	0,00	0,00	0,000

2. Table

In our case, queuing is a trade-off between the maximum capacity needed and the delay induced by waiting in the queue (Table 2).

2.2.1 Queueing with priority

It is well known that long tasks can dramatically increase maximum and the average queueing time when using FIFO queuing, because they unreasonably uphold the queue.



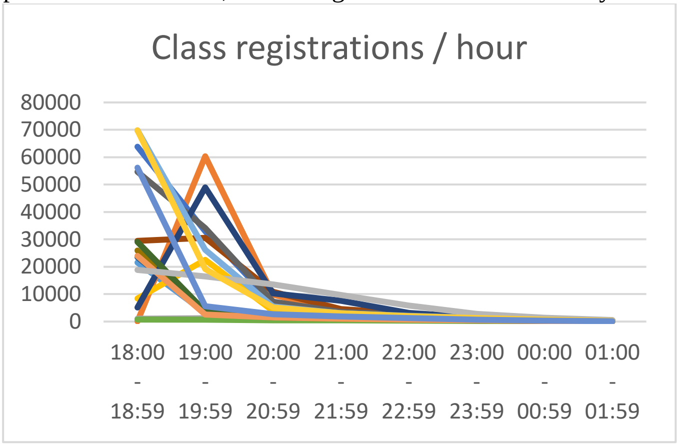
5. Figure The effect of prioritizing workload by length

Clearly, in our case, when the capacity is scarce (less than 30 concurrent sessions), the maximum and average delay is significantly higher compared to the other cases.
As an idea, we filtered out the reservations longer than 24-hours to see the effects. (Fig. 5) Comparing the maximum delay, the result was that filtering out long-term reservations can really help in resource-scarce scenarios.
As a conclusion, when managing the peak workload, redirecting the background load to a separate infrastructure can really help, thus workload classification, prioritization are both viable alternatives.

3 NEPTUNE

The second dataset corresponds to the workload of a university-wide student portal sampled at the start of a class registration for a semester. Similar to the VCL load, the average load is small compared to the peaks occurring when class registration starts. Consequently, in our case, the load is measured in the number of class registrations per hour.

Traditionally, class registration starts at 18:00, and the effect of peaks end in 8 hours, narrowing the window of our analysis.



6. Figure Workload distribution during peak hours

Fig 6. depicts the different proceedings of class registrations. Visibly, the first three hours are critical. Due to the extremely limited dataset, EVA was not applied, only queueing was investigated.

The dataset consists of hourly numbers ("there are buckets for each hour"), thus, in this case, capacity means the maximum amount of class registrations the system can process in an hour ("the bucket size"). Moreover, FIFO queuing means, moving class registrations from one bucket to another bucket (re-scheduling them), which means that re-scheduling is delaying class registration by integer hours.

capacity	max. delay(h)	total delay(h)	average delay(h)
15000	7	169	3
20000	5	89	2
25000	4	49	2
30000	3	34	1
35000	2	17	1
40000	2	15	1
45000	2	11	1
50000	1	6	1
55000	1	5	1
60000	1	4	1
65000	1	2	1
70000	0	0	0

3. Table

Looking at the results, by embracing 3 hours of maximum delay, the required capacity could be halved.

4 CONCLUSION AND FUTURE WORK

The basic dataset used for the EVA is quite narrow in the statistical sense as it contains only a very limited number of semesters. Moreover, each semester has its own dynamics in the terms of the behavior of the students before the submission deadline. Early analysis indicates a high level of similarity between the individual semesters (the psychology of students seems to be invariant…). In these terms, a combined methodology of aggregating the data similar to the block maxima principle before using the threshold exceedance algorithm seems the most promising one.

However, note that the same phenomena appear in many technical systems as well, where the operating time periods and accordingly the number of log data collected go beyond the strict limitations of the pilot example presented.

Based on our analysis, as expected, even the simplest peak management mechanisms can greatly reduce the required capacity in bursty workload environments, which could significantly increase the overall utilization of a system.

During the analysis of the VCL dataset analysis, only a single dimension was used: the number of concurrent virtual machines. Multidimensional analysis, where the CPU and memory usage are investigated, seems an evident and interesting area to research.

When queuing was analyzed, only a simple FIFO queue was implemented. However, considering the many queueing algorithms available, there is much to be researched here.

Moreover, a basic workload classification and prioritization idea (filtering larger than 24 hours reservations) seemed viable in peak workload management (especially when the capacity is limited), thus further investigations seem promising.

REFERENCES

[1] Robert Hanmer "Patterns for Fault Tolerant Software", Wiley Publishing, 2007

[2] Marcus Carvalho, Daniel A. Menasc, and Francisco Brasileiro. 2017. Capacity planning for IaaS cloud providers offering multiple service classes. Future Gener. Comput. Syst. 77, C (December 2017), 97-111.

[3] The R Project for Statistical Computing: https://www.r-project.org/ (last check: 2017-12-01)

[4] Alexander J. McNeil, Rudiger Frey, and Paul Embrechts. 2015. Quantitative Risk Management: Concepts, Techniques, and Tools. Princeton University Press, Princeton, NJ, USA.

[5] P. Rakonczai: On Modeling and Prediction of Multivariate Extremes. Mathematical Statistics Centre for Mathematical Sciences, Lund University, 2009

[6] Pavel Cízek, Wolfgang Härdle, Rafal Weron "Statistical Tools for Finance and Insurance", Springer, 2011

[7] Caeiro, Frederico; Gomes, MIvette; 'Threshold Selection in Extreme Value Analysis: Methods and Applications', Extreme Value Modeling and Risk Analysis, pp.69-86, Chapman and Hall/CRC, 2015

[8] Scarrott, C. & MacDonald, A. (2012), 'A review of extreme value threshold estimation and uncertainty quantification', REVSTAT - Statistical Journal 10 (1), 33--60.

[9] Facebook Prophet: https://facebookincubator.github.io/prophet/ (last check: 2017-12-01)

[10] Taylor SJ, Letham B. (2017) Forecasting at scale. PeerJ Preprints 5:e3190

[11] extRemes: Extreme Value Analysis, 2.0-8

[12] ismev: An Introduction to Statistical Modeling of Extreme Values, 1.41

[13] prophet: Automatic Forecasting Procedure, 0.2.1

Unknown, Uncertain, Untrue: Challenges in Inference Using Semantic Life Science Data

Bence Bruncsics, Andras Gezsi, Peter Antal
Budapest University of Technology and Economics
Department of Measurement and Information Systems
Budapest, Hungary
Email: {bruncsics, gezsi, antal}@mit.bme.hu

Abstract—**The rapidly accumulating electronically available data and knowledge in life sciences currently cannot be integrated into a general, unified knowledge base with adequate inference. A unified knowledge representation is still missing, which could include large scale uncertain expert knowledge and *in silico* predictions from machine learning. Thus, the available information is fragmented and differently processed. In order to overcome this barrier and provide a common base for automatic data processing many databases became available in a semantic form. However, the current query languages for semantic linked open data have many limitations, therefore they are not sufficient for creating an integrated knowledge representation and reasoning. Overcoming this obstacle there are different solutions for knowledge representation but currently no such tool has the power to predict or find solutions for the most important biological questions. Combination of semantic web technologies and probabilistic techniques can result a novel powerful tool for life sciences.**

Index Terms—**graph databases, intelligent systems, probabilistic knowledge representation, semantic Web, uncertain databases**

I. Introduction

Besides to regular free-text publication an emerging trend is to make accessible data and knowledge in representations more suitable for automated processing. Amongst databases and information sources semantic technologies have the widest range covering most of the life science entities and properties with additional information about their relationships.

A. Semantic Solutions

Providing an efficient foundation for constructing, storing and interpreting life science data is challenging due to its size, complexity and the excessive need for an easily accessible way to process these data. A candidate solution originates from Word Wide Web Consortium in form of semantic technologies providing such scalable, efficient, accessible and simple standards as the RDF (Resource Description Framework) [1] and the OWL (Web Ontology Language) [2].

In practice the different biological entities like diseases, genes or compounds are handled in different databases and cross-domain integration is ensured by unique identifiers, URIs (Uniform Resource Identifier) of these entities. The URIs form a consistent system, because they are usually URLs pointing to locations maintained by the databases creators or by projects dedicated to RDFize multiple databases, technically they are formed by proper database specific prefixes and IDs for each entity.

There are numerous properties for most entities in the databases and in RDF these are connected to the entities using OWL based terms as a form of knowledge representation e.g. genes have a location property describing a position in the DNA where the gene can be found. In addition to properties from the databases, there is information about the relationship between entities from different databases like a gene is associated with a disease. The associations can have further properties like which mutation of a gene is associated with a disease, resulting in complex relationships between databases.

B. Linked Open Data and Semantic Databases

Data sharing became a central topic in life sciences, balancing aspects of privacy, proprietary, scientific advance and repeatability. To contribute to the semantic Web, the data needed to be RDFized. It is usually done by database providers such as PubChem [3], but more commonly separate organizations perform this conversation, such as the European Bioinformatics Institute (EMBL-EBI) [4], or collaborative projects such as Ontobee [5]. Semantic data has multiple locations, different origins and can even have duplicates, therefore its proper management is essential for its applicability. The Linked Open Data approach (Fig. 1) is a large scale community project aimed to set RDF links between the open RDF datasets and to encourage the community to RDFize the separate databases expanding the semantic Web [6].

Currently the most relevant semantic life science databases contain thousands ($10^3 - 10^4$) of diseases, over a million genes from different species, and millions ($10^6 - 10^8$) of compounds and many more databases and descriptions with billions of links between entities [7].

II. Background Knowledge of Life Sciences

In life sciences, many revolutionary measurement technologies have emerged in the last three decades, producing unprecedented amount of data and many databases were formed to store and organize these data. Further databases and ontologies were made to organize the already existing current information resulting in a large variety of information sources. Most of the databases are formed along various aspects of

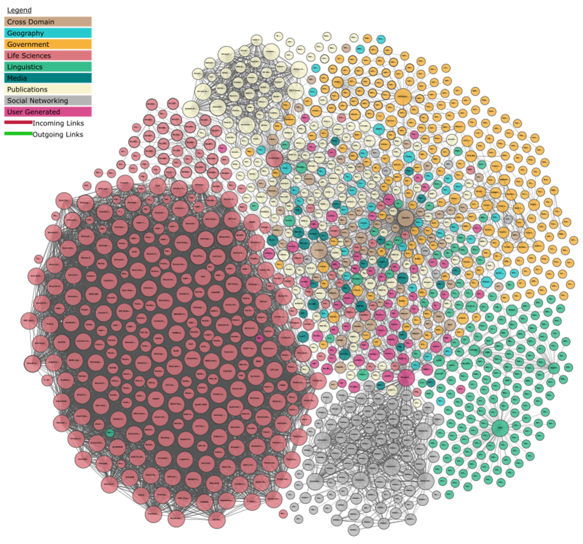


Fig. 1. Linked Open Data (LOD): Life sciences contribute the most data for LOD [8]

drugs, genes, and diseases, such as genetic, proteomic, phenotypic or chemical information. The most important databases are focusing on listing all the entities at a given level (such as all the genes) and usually contain additional descriptions of their entities via common properties. Other databases describe ontologies and refer to other entities affected by the ontologies; and there are databases focusing on linking databases based on specific techniques like gene expression profiles in different diseases (based on microarray technology).

However, inherent challenges of representing scientific information is still not addressed by the semantic framework, such as the (1) representation of large scale implicit, negative information (cf. closed world assumption), (2) representation of uncertain information (3) and representation of inconsistent information, partly as a consequence of untrue scientific reports from misconducted or fraudulent studies.

A. *Biases for known and unknown*

Life sciences always had biases due to scientific paradigms, trends, measurement constraints, budget limitations or publication and grant policies. For example, the publication bias for positive findings, is a serious challenge for text-mining, machine learning and automated discovery systems.

The need for publication and gathering citations creates trends in science resulting popular fields but also popular entities such as popular genes like CD4 gene which plays a role in HIV infections but less important for the general understanding of biology [9]. These trends result in disparate distributions and characteristics for the different entities and the popularity of a field or entity is not necessarily a consequence of the biological relevance.
The effect of the uneven mapping stands out more when only a small portion of field is discovered which is the case in most part of the life sciences. For an integrating model in drug discovery the chemical data should contain information about the targets (usually proteins) of the compounds. Knowing this information, a matrix can be created containing the compound target interactions, but in practice the coverage of these matrices can be as low as 0.1% (a standard bioactivity database ChEMBL contains 14 million values including duplicates for the 2 million x 10 thousand interactions of its compounds and targets, [10]). Knowing that the data in these sparse matrices are unevenly distributed rises further challenges for processing this information.

B. *Uncertain scientific knowledge*

Statistical data analysis provides the base for our empirical science and evidence-based medicine, however the limits and conditions of statistical test driven scientific publication policies are often overlooked. Even if we assume honesty for all researchers, it is easy to make mistakes (like excluding seemingly wrong parts from data) without a statistical mindset, not knowing the consequences of certain actions. And unfortunately, experiments with positive results will have higher chance to be accepted than those where the methods were rigorously followed, but had negative results.
Further problem is the mystified requirement and unconditional acceptance of p-values as ultimate proofs. In the field of biology, a result showing a p-value lower than 0.05 is accepted as significant effect, even without considering its power and the number of the corresponding experiments. Often the number of trials or the parallelly tested theories are not included in the statistics or sometimes these are not even published resulting in an overwhelmingly large number of false positive results. In summary, currently the representation of certainty and the context of a statistical evidence is severely restricted.

C. *Untrue, Non-repeatable Scientific Reports*

It is a suspected, but still astonishing finding that at least 50% of published studies of early-stage venture capital firms cannot be repeated with the same conclusions by an industrial lab [11]. Furthermore, a large-scale study by Bayer in 2011 showed that only 20-25% of their in-house findings were completely in line with the original publications [12]. Therefore, it is clear that the databases based on publications with overwhelming amount of false positive findings will have the same biases.

III. Reasoning using Life Science Data

The need for automation of reasoning using life science data is clear, because it could overcome many questions that the current state of science wouldn't be able to answer otherwise, but cooperative schemes with scientific discovery support systems can also accelerate research and increase efficiency.

A. *Challenges in Reasoning*

Despite rapid advancements in deep linguistic analysis, using solely free-text scientific publications is not a viable option currently for reasoning in life sciences, therefore the manually curated, computationally processable semantic data remains a default option. However, the usage of semantic data and technologies is hindered by many obstacles:

1) Vulnerable distributed inference: Life science databases are usually not integrated into a common datastore; therefore, distributed queries are strongly affected by the accessibility of the sources; even one missing endpoint could fail the query.

2) Intractable inference: The computational complexity of unrestricted queries is so high, that usual inference is intractable even in a problem-specific unified database using dedicated servers.

3) Expert queries: Query languages provide a wide range of logical and calculation methods, but these can be rather complicated, especially using negation for filtering.

4) Uncertain reasoning: There is no simple automated way to construct uncertain evidence over different entities and combine uncertain evidences, especially using further quantitative measures as well, such as similarity measures, which are popular in biomedicine.

5) Transparency: Checking the inference paths and calculation in case of complex queries requires deep understanding of the underlying techniques and it is time consuming. Hence, user-level interfaces are essential to translate the complex inference processes for a domain-expert scientist to be able to recognize the significance of a given inference.

There are numerous bioinformatics tools for the large-scale fusion of heterogeneous data and knowledge, but these tools either support the general, non-quantitative (logical) inference over semantic resources or focus on a dedicated task, such as gene prioritizers [13], [14]. Further problem in the current approaches is the lack of support for high-dimensional, partial, noisy evidences and deep control for the inference process.

IV. Towards Quantitative Semantic Fusion

To cope with these deficiencies, the Quantitative Semantic Fusion (QSF) Framework was developed at the ComBineLab, Department of Measurement and Information Systems. Epistemological aspects of linked open life science data can be approached in QSF as follows:

1) Predicting the unknown: There are techniques to complete sparse matrices based on the available data creating a better foundation for reasoning [15]. Furthermore, the used similarities and properties could be used to approximate probabilities for the filled or even for the original data.

2) Probabilistic representation of uncertainty: Semantic life science databases often formed around techniques like chemical screening or microarrays providing access to the sources. The statistical parameters of the data are characteristic to the techniques, therefore it is possible to calculate or approximate Bayesian probabilities for these entities.

3) Representing trust: There are dedicated solutions for knowledge representation such as Evidence Code Ontology (ECO) [16] in bio-medicine, providing information about the data source (e.g. motility assay evidence) or HELO [17] which is more suitable for probabilistic knowledge representation. Furthermore, even textual information can be translated into probabilistic graphical models and using approximations for the most probable explanations creating link between the available data and probabilistic techniques.
The QSF system offers the following solutions for the specific challenges.

1) Integration For a general biological model the integrative functions are necessary, therefore the essential disease, genetic, protein, pathway and substance information must be included. Based on linked open data Chem2Bio2RDF [7], the QSF framework is able to integrate and perform inference using the relevant information sources. Furthermore, it is possible to expand the framework with further sources.
2) Accessibility and scaling To cope with computational complexity, the relevant data sources are stored and processed in one server. Due to RDF compression techniques and proper selection of the relevant properties, entities and links the overall size of a minimal, but representative data covering the a given problem in life sciences remains in the few GB range.
3) Filtering SPARQL cannot handle well negation and complex filtering, but in the RDF-based QSF framework complex logic can be applied to focus and control the inference process.
4) Transparency Large-scale data visualization techniques like Cytoscape [18] allow users to create and explore graphs, which are natural, transparent, intuitive way for representing biological data. The QSF allows the export of the results of evidence propagation as paths in knowledge graphs, which provides easy access and interpretation for biologists.
5) Bayesian inference Using probabilistic methods, quantitative evidences can be constructed for the input queries based on the sources. For example, for a drug candidate the acceptance rate based on the trial number is a possible approximation for a drug-disease association. The QSF allows the approximation of a Bayesian inference technique. Many bioinformatics tools are able to incorporate and provide p-values, but QSF provides a full-fledged approximation for a Bayesian inference.
6) Bayesian fusion Integrating multiple sources or using data from similar, repeated experiments have significant advantages, due to the cancellation effect of method or setup specific biases. Calculating pseudo-Bayesian posteriors for such data can result a significant improvement in case of such an uncertain space as the life sciences [19].

V. Application of Quantitative Semantic Fusion

Translation of questions with complex biomedical background to a formal query is often difficult and require special technical knowledge. To support this process, the graphical user interface of the QSF endowed with the following functionalities:

1) Selecting targets: Having computationally approachable targets (i.e. answers) is essential for constructing a query, which in itself can be seen as a modeling activity. In QSF, evidences are entered, then propagated throughout the network, thus selection of targets does not influence the results, but essentially influence interpretability.

2) Context-sensitive inference: Planning which sources and paths should be taken into account in the inference can be controlled by the query and the results can be affected by them.

3) Collecting inputs: The proper input is key for inference and it is recommended to collect as much data as possible from different sources to overcome the biases of the data by canceling out the random effects.

4) Converting evidences: Providing probabilities for the inputs is challenging in many cases, and recommendations or analytic solutions are needed for this step.

5) Applying conditions: Filtering is essential within the inference, e.g. via negation even complex questions can be asked.

6) Interpreting the results: It is essential to check the paths of the evidences to find possible anomalies and to get a better understanding of the dominating effects in behind the results.

VI. Conclusion

Automated reasoning using life science data is challenging due to the fragmentation of the data located in separate databases, but fortunately semantic web technologies and the Linked Open Data approach provide sufficient background to access these data. However, current approaches are very limited in reasoning efficiently based on this knowledge: (1) the native usage of semantic data as inference in graph database lacks the ability of to incorporate uncertainty, (2) network approaches using diffusion-based inference methods lack semantic control within the inference, (3) kernel fusion based prioritization methods cannot directly manage relational data. Additionally, biases of life science data complicate the problem further, because the sources are uncertain, incomplete and unevenly distributed. Our group developed a system, approximating a full-fledged probabilistic inference, which is grounded in an underlying semantic graph database. Furthermore, we developed a graph-based query language to support the translation of complex biomedical queries, allowing semantic influence over the inference process. I developed multiple models and manually evaluated various inference schemes in this system to refine existing applicability and explore new directions for its development [20]. Tools integrating different information sources, handling high-dimensional weak evidences, providing semantic control for the inference and supporting the interpretation of the results, such as the QSF framework, could provide new possibilities for cooperative reasoning of experts and machines.

Acknowledgment

The author would like to thank the advices and guidance of Peter Antal, and the continuous help of Andras Gezsi and Gabor Guta. The research has been supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.2-16-2017-00013).

References

[1] S. Decker, S. Melnik, F. Van Harmelen, D. Fensel, M. Klein, J. Broekstra, M. Erdmann, and I. Horrocks, "The semantic web: The roles of xml and rdf," *IEEE Internet computing*, vol. 4, no. 5, pp. 63–73, 2000.

[2] S. Bechhofer, "Owl: Web ontology language," in *Encyclopedia of Database Systems*. Springer, 2009, pp. 2008–2009.

[3] S. Kim, P. A. Thiessen, E. E. Bolton, J. Chen, G. Fu, A. Gindulyte, L. Han, J. He, S. He, B. A. Shoemaker *et al.*, "Pubchem substance and compound databases," *Nucleic acids research*, vol. 44, no. D1, pp. D1202–D1213, 2015.

[4] S. Jupp, J. Malone, J. Bolleman, M. Brandizi, M. Davies, L. Garcia, A. Gaulton, S. Gehant, C. Laibe, N. Redaschi *et al.*, "The ebi rdf platform: linked open data for the life sciences," *Bioinformatics*, vol. 30, no. 9, pp. 1338–1339, 2014.

[5] E. Ong, Z. Xiang, B. Zhao, Y. Liu, Y. Lin, J. Zheng, C. Mungall, M. Courtot, A. Ruttenberg, and Y. He, "Ontobee: A linked ontology data server to support ontology term dereferencing, linkage, query and integration," *Nucleic acids research*, vol. 45, no. D1, pp. D347–D352, 2016.

[6] L. Yu, "Linked open data," in *A Developers Guide to the Semantic Web*. Springer, 2011, pp. 409–466.

[7] B. Chen, X. Dong, D. Jiao, H. Wang, Q. Zhu, Y. Ding, and D. J. Wild, "Chem2bio2rdf: a semantic framework for linking and data mining chemogenomic and systems chemical biology data," *BMC bioinformatics*, vol. 11, no. 1, p. 255, 2010.

[8] A. Abele, J. McCrae, P. Buitelaar, A. Jentzsch, and R. Cyganiak, "Linking open data cloud diagram (2017)," 2017.

[9] R. Hoffmann and A. Valencia, "Life cycles of successful genes," *TRENDS in Genetics*, vol. 19, no. 2, pp. 79–81, 2003.

[10] A. Gaulton, A. Hersey, M. Nowotka, A. P. Bento, J. Chambers, D. Mendez, P. Mutowo, F. Atkinson, L. J. Bellis, E. Cibrián-Uhalte *et al.*, "The chembl database in 2017," *Nucleic acids research*, vol. 45, no. D1, pp. D945–D954, 2016.

[11] L. Osherovich, "Hedging against academic risk," *SciBX: Science-Business eXchange*, vol. 4, no. 15, 2011.

[12] F. Prinz, T. Schlange, and K. Asadullah, "Believe it or not: how much can we rely on published data on potential drug targets?" *Nature reviews Drug discovery*, vol. 10, no. 9, pp. 712–712, 2011.

[13] L.-C. Tranchevent, A. Ardeshirdavani, S. ElShal, D. Alcaide, J. Aerts, D. Auboeuf, and Y. Moreau, "Candidate gene prioritization with endeavour," *Nucleic acids research*, vol. 44, no. W1, pp. W117–W121, 2016.

[14] G. Valentini, A. Paccanaro, H. Caniza, A. E. Romero, and M. Re, "An extensive analysis of disease-gene associations using network integration and fast kernel-based gene prioritization methods," *Artificial Intelligence in Medicine*, vol. 61, no. 2, pp. 63–78, 2014.

[15] B. Bolgár and P. Antal, "Vb-mk-lmf: fusion of drugs, targets and interactions using variational bayesian multiple kernel logistic matrix factorization," *BMC bioinformatics*, vol. 18, no. 1, p. 440, 2017.

[16] M. C. Chibucos, C. J. Mungall, R. Balakrishnan, K. R. Christie, R. P. Huntley, O. White, J. A. Blake, S. E. Lewis, and M. Giglio, "Standardized description of scientific evidence using the evidence ontology (eco)," *Database*, vol. 2014, 2014.

[17] L. N. Soldatova, A. Rzhetsky, K. De Grave, and R. D. King, "Representation of probabilistic scientific knowledge," *Journal of biomedical semantics*, vol. 4, no. 1, p. S7, 2013.

[18] P. Shannon, A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker, "Cytoscape: a software environment for integrated models of biomolecular interaction networks," *Genome research*, vol. 13, no. 11, pp. 2498–2504, 2003.

[19] M. A. Province and I. B. Borecki, "Gathering the gold dust: methods for assessing the aggregate impact of small effect genes in genomic scans." in *Pacific Symposium on Biocomputing*, vol. 13, 2008, pp. 190–200.

[20] B. Bence, "Large-scale data and knowledge fusion in aging research," Scientific Student Conference, 2017, 2017.

An Application Example of Regularization: Time-varying Operational Modal Analysis

Péter Zoltán Csurcsia[a,b,*], Johan Schoukens[a], Bart Peeters[b]
[a]Vrije Universiteit Brussel, Department of Engineering Technology, Brussels, Belgium
[b]Siemens Industry Software NV., Leuven, Belgium

Abstract— **This paper presents an efficient nonparametric time-varying time domain system identification method for the Operational Modal Analysis (OMA) framework. OMA tackles industrial measurements of vibrating structures in real-life operating conditions without the exact knowledge of the excitation signal. In this work a method is provided to estimate the time-varying output autocorrelation function of vibrating structures and applied to a measurement of wind tunnel testing of an airplane. Using the proposed methodology, the estimated time-varying two-dimensional output autocorrelation model provides a good data-fit with respect to tracking of varying resonances.**

Keywords*— *Bayesian methods; Nonparametric methods; Time-varying system identification; Operational Modal Analysis

I. Introduction

The paper deals with the time variations of (vibrating) mechanical and civil structures described by the nonparametric Operational Modal Analysis (OMA) framework. OMA is a special identification technique for estimating the modal properties (e.g. resonance frequencies, damping, mode shapes) of structures based on vibration data collected when the structures are under real operating conditions without having access to the excitation signals. This technique can provide the engineers with useful information to understand the dynamic behavior of the underlying structure in real-life usage scenario, and it can be used to validate and update the numerical models developed in the design phase [1].

The main issue is that the dynamics of underlying systems may vary significantly when operating in real-life conditions. In this case, advanced modelling is needed taking into account the time-varying (TV) behaviour because the unmodelled time variations might lead to instability and structural failures. Contrary to the classical identification frameworks, a further challenge with the OMA framework is that the excitation signal is not known exactly, but it is assumed to be white noise.

A good example of a time-varying mechanical structure can be, for instance, an airplane. The TV behavior originates from the decreasing weight due to the fuel consumption, and from different surface configurations during take-off, cruise and landing Moreover, the resonance frequency and damping of most vibrating parts (for instance the wings) of a plane vary as a function of the flight speed and height. In this paper the wind tunnel testing of an airplane is considered.

In this work the first results of a nonparametric TV OMA method are presented to estimate efficiently the linear time-varying (LTV) output autocorrelation function (ACF) of the observed system. In case of TV systems it is a common practice to use different types of short-time Fourier transform (SFT) techniques that describes the underlying system as a series of linear-time invariant (LTI) systems ([2]-[7]). In case of TV OMA, other nonparametric techniques have a limited applicability due to lack of knowledge of excitation. These SFT based models can describe the time-varying behavior quite well when the time-variations are very smooth. The drawback of these methods is that such an LTI model is not sufficient to describe the system's behavior, if the time variations are fast, or when a significant variation occurs.

The proposed method is intended to overcome the issues with the classical SFT based methods and it is based on the regularization methods that have been developed for impulse response function (IRF) modelling of LTI systems ([7]-[9]). The novelty of this work compared to the IRF regularization is to formulate the regularization for the special case of the estimation of LTV systems in an industrial OMA framework – i.e. unknown input– similarly to the regularized LTV framework with known excitation [3],[4].

The paper is organized as follows: in Section II the basic concepts are summarized and the exact problem formulation is given. Section III provides the model structure and the cost function of the problem. Section IV introduces the regularized LTV OMA estimation method and addresses the implementation procedure. Section V shows a measurement example where the results of the classical framework are shown to compare to the results proposed method. Finally, the conclusions are provided in Section VI.

II. Basics

A. Definitions and Assumptions

Several definitions and assumptions must be addressed prior to carrying out any system identification procedure.

A discrete LTV system is completely characterized by its 2D IRF $h[t,\tau]$. Its steady-state response to an arbitrary signal $u[t]$ is given by ([10],[11]):

$$y[t] = \sum_{\tau=-\infty}^{+\infty} h[t,\tau]u[t-\tau] \qquad (1)$$

where the parameter t is the global time (the time when the system is observed) and τ is the system time (the direction of the impulse responses/lags) [3].

The following assumptions are made in this work:

ASSUMPTION 1 The system output $y[t]$ is disturbed by additive, i.i.d. Gaussian stationary noise with a zero mean and a finite variance.

ASSUMPTION 2 The 2D IRF is smooth i.e. the spectral content of the underlying LTV system is highly concentrated at the low frequencies. A more precise definition can be found in [9].

ASSUMPTION 3 The considered discrete LTV system is causal (i.e. $h[t,\tau] = 0$, when $\tau < 0$) and stable

ASSUMPTION 4 The length of the IRFs ($L+1$) is shorter than the observation (measurement) window length (N).

ASSUMPTION 5 The excitation signal $u[t]$ is not known exactly but it is assumed to be white noise with a TV finite variance.

B. Problem formulation

In this section, a brief overview of the LTV OMA problem is presented where we explain 1) the issues related to the nonparametric identification of LTV systems in general 2) the issues related to the OMA framework w.r.t. the classical identification framework.

1) Nonparametric Identification of LTV Systems

The challenge with LTV systems is that the TV 2D IRF and FRF are not uniquely determined from a single set of input and output signals – unlike the case of LTI systems.

Consider the model defined in (1). Imposing *Assumptions* 1-4, the following measured output $y_m[t]$ is given at time t:

$$y_m[t] = y[t] + e[t] = \sum_{\tau=0}^{L} h[t,\tau]u[t-\tau] + e[t] \quad (2)$$

where $t = 0 \dots N-1$.

The problem lies in the fact that there is a nonuniqeness issue. The number of unknown in the TV 2D nonparametric model is NL but there are only N measured points. Hence, the model that relates the input to the output is not unique, because there are only N linear relations (measurement samples) for NL unknowns.

2) Operational Modal Analysis Framework

OMA is a very important tool because in the case of vibrating structures it is common that the real operating conditions differ significantly from dynamic measurements performed in laboratory conditions. In case of industrial measurements of large mechanical and civil structures (e.g. airplanes, bridges, wind tunnels) the excitation signal is most of the time not measurable, or it would be too difficult and complex to measure [1].

Because the excitation signal is not known (see *Assumption* 5) an IRF or FRF cannot be directly estimated. Instead, the output autocorrelation function (ACF) and/or its Fourier transform are used in practice.

III. THE NONPARAMETRIC IDENTIFICATION METHOD

A. The Model Structure

Using the OMA framework, the underlying time-varying systems can be represented by their 2D ACF when *Assumption* 1-5 are satisfied. In this case the 2D TV ACF would provide the convolved LTV IRF in time-domain ($\sum_{\tau=0}^{L} h[t,\tau]h[t+\tau]$). The proof is out of scope of this paper, details for the LTI case can be found in [1]. The TV ACF R_{yy} centered around t, at time lag τ, with a window length of (L+1), is estimated with the measured output signal y_m (see (2)) by smoothing over a window with a length of L+1:

$$\hat{R}_{y_m y_m}[t,\tau] = \frac{1}{L+1}\sum_{k=-\frac{L}{2}}^{\frac{L}{2}} y_m[t+\tau+k]y_m[t+k] \quad (3)$$

Next, the double time indices $[t,\tau]$ will be omitted in order to make the text more accessible. The key idea of this work is to extend the existing nonparametric regularization methods such that some advanced prior information can be taken into account.

B. The Cost Function

The basic idea of the regularization technique is that by using the prior knowledge (formalized later on) on the system dynamics, and by introducing some bias error, the variance can be significantly reduced resulting in a significantly reduced mean square error (MSE) [8]. In order to include the prior knowledge in the nonparametric representation, an extended cost function (V) must be defined.

This cost function (V) consists of the sum of the ordinary least squares cost function (V_{LS}) which is now defined for the 2D LTV ACF case as

$$V_{LS} = \left\| vect(R_{yy}) - vect(\hat{R}_{y_m y_m}) \right\|_2^2 \quad (4)$$

and the regularization cost function (V_r):

$$V_r = \sigma^2 vect(R_{yy}^T)P^{-1}vect(R_{yy}), \quad (5)$$

such that the new, combined cost function is given by

$$V = V_{LS} + V_r \quad (6)$$

where P is a – special covariance – matrix containing the prior information, and σ^2 is the amount of regularization (prior) applied – it is usually proportional to the noise variance of the observation. To simplify the model and computational complexity σ^2 is kept constant but in general TV σ^2 – and observation noise– may be considered.

Minimizing (6) with respect to R_{yy} gives the solution which estimates the 2D IRF of an LTV system:

$$vect(\hat{R}_{yy,reg}) = (I + \sigma^2 P^{-1})^{-1} vect(\hat{R}_{y_m y_m}) \quad (7)$$

where I denotes the identity matrix.

IV. REGULARIZATION

In the section the practical implementation of the inclusion of the prior knowledge into the covariance matrix is addressed.

A. Considered Kernel Functions

The covariance hypermatrix P is constructed by using kernel functions. The specific choice of the kernels has a major effect on the quality of the estimate. Next, the kernels that will be used in this paper are explained.

The smoother Radial Basis Functions (RBF) is defined by

$$P_{RBF}(t_1,t_2)=e^{-\frac{(t_1-t_2)^2}{\gamma}} \tag{8}$$

where γ is a parameter representing the length scale. The larger the length scale, the smoother the resulting estimated function is.

In case of Diagonal Correlated (DC) kernels next to the smoothness assumption, exponentially decaying envelope can be imposed and is defined as follows:

$$P_{DC}(t_1,t_2)=e^{-\alpha|t_1-t_2|}e^{-\frac{\beta(t_1+t_2)}{2}} \tag{9}$$

where α quantifies the correlation length between adjacent impulse response coefficients, or in other words it controls the smoothness, and β scales the exponential decaying.

Please note that depending upon the prior knowledge, different shorts of kernel functions may be used.

B. *The Covariance Matrix and the Prior Knowledge*

In order to overview the properties of the TV ACF an illustration is shown in Fig 1. The first prior knowledge is that the considered ACFs are smooth. This smoothing is applied once over the system time τ which refers to "classical" evolution of IRF/ACF, and once over the global time t which gives a support to handle the time-varying behavior.

In addition to smoothing properties, another assumption can be imposed and incorporated in P by applying a more strict definition of stability: the IRFs and therefore the ACFs are exponentially decaying. Note, that most of the stable mechanical systems exploit this behaviour. When this is not the case other kernels might be used.

In the proposed approach, every point of the 2D ACF surface is connected to each other. It means that all the elements in the covariance matrix are non-zero, therefore the number of constraints is high, and the degrees of freedom of the system of linear equations are significantly decreased resulting in a unique solution. The elements in the τ direction of the autocorrelations (horizontal blue direction) are linked by DC kernels. Elements in the global time t direction (vertical red direction) are linked by RBF kernels. This contraction is formulated as follows:

$$\mathrm{P}_{\{t_1,t_2\},\{\tau_1,\tau_2\}}=\mathrm{P}_{RBF}(t_1,t_2)\cdot\mathrm{P}_{DC}(\tau_1,\tau_2) \tag{10}$$

for every possible pair of t and τ.

C. *Tuning of the Model Complexity and Hyperparamters*

$\gamma,\alpha,\ \beta,\sigma^2$ parameters in (5),(8),(9) are the so called hyperparameters and their values are restricted as follows: $\gamma,\sigma^2>0$ and $\alpha,\beta\geq 0$. The nonparametric system identification method presented in this work consists essentially of two steps: 1) optimization of the hyperparameters to tune the matrix P, and 2) computation of the model parameters using (7).

All the hyperparameters are tuned with the use of measured output data only by maximizing the marginal likelihood of the observed output ([8],[9]).

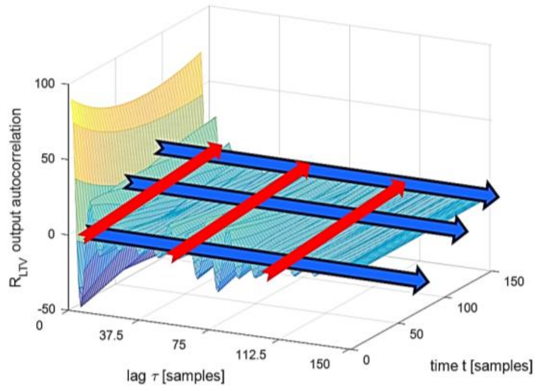


Fig. 1. The TV output ACF of a system. The arrows show the visualization of the possible regularization directions. The blue arrow refers to the τ system time where decaying and smoothness of the adjacent elements can be imposed. The red arrow refers to the direction of time variations (t axis) where the smoothness of the adjacent autocorrelation functions can be imposed.

V. Results

A. *Measurement Setup*

This section presents an industrial example of the in-line flutter assessment of the wind tunnel testing of a scaled airplane model. The measurement time is 424 sec, the sampling frequency (f_s) is 500 Hz, the number of data samples (N) is 212000. The segmented window size (L+1) is 500 samples (which corresponds to 1 sec, 1 Hz resolution). The Mach number (proportional to the airflow i.e. wind speed w.r.t. sound speed) is varying between 0.07 and 0.79. An illustration of the measurement setup is given in Fig 2.

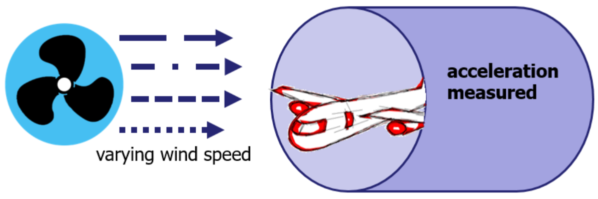


Fig. 2. The measurement setup is shown. The airflow (wind speed) is varying while the airplanes acceleration is measured in the wind tunnel.

In this type of testing, it is desired to carefully verify and track the vibration behavior, since during flutter appearance, system destruction can occur. In this paper, wind tunnel data at various flow conditions are used to validate the approach for tracking the evolution of the resonance frequencies, damping ratios.

Further details on the measurement procedure can be found in [12]. Exact specifications on the airplane are omitted due to the confidentiality of the industrial project.

B. *Results*

In this section the time and frequency domain results of the traditional SFT approach are compared to results of the 2D regularization method. In case of SFT the output measurement is split into short subrecords, and each time the output ACF is calculated. The result is a series of ACFs which represents the classical TV ACF estimate. The TV collerogram (power spectrum estimate) is then given by the Fourier transform of the TV ACF. When the proposed method is considered, the ACF estimates are normalized and regularized as it is detailed in Section III and IV.

Figure 3. and Figure 4. show the frequency domain results of the SFT and regularized approach. Since the most important part is the evolution the resonances, only the top view of the 2D collerograms is shown. Observe that the traditional approach is very noise where the regularized solution is smooth and more details can be captured.

It is important to remark that the classical SFT results can also be post filtered/smoothened – even with regularization or with other approaches (e.g. B-spline techniques) – but in this case the accuracy of the result will be between the classical and proposed approach.

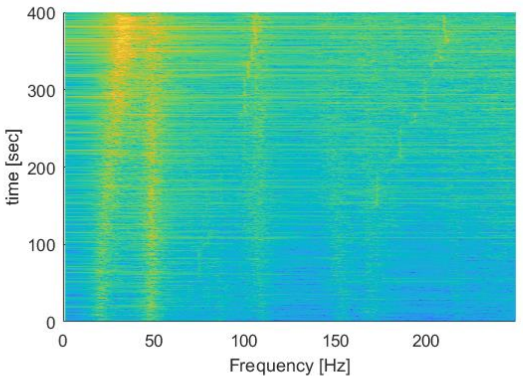


Fig. 3. The time-varying output power spectrum estimate of observed system is shown when the classical SFT method is considered.

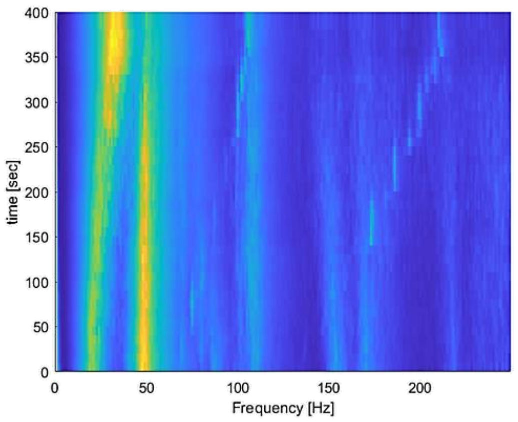


Fig. 4. The time-varying output power spectrum estimate of observed system is shown when the proposed method is considered.

VI. CONCLUSIONS

The main concept of the nonparametric TV OMA framework is to provide more accurate and smoother models which are suitable for simulation, design and – indirectly – control. This will improve the overall quality and safety of products, and speed up the design process. Using the proposed advanced 2D regularization techniques it is possible:

- to reduce the noise influence such that the measurement quality will be significantly better
- to gain a better insight into the details of the time-variations

Further, a nonparametric TV OMA model can be used:

- to capture and to track the time-varying resonance frequencies and damping ratios
- to simulate and validate during the design phases

When it is necessary, the nonparametric TV OMA model can be used to estimate a parametric OMA model by cleaning up the data and allowing direct access to the mode shapes and making the control possible.

The drawback of the proposed method is that the computational load and memory need are significantly higher but this is negligible in the applications where 1) the preparation of the measurement setup takes days, and 2) the safety of the product and its intended users is concerned.

ACKNOWLEDGMENTS

This work was funded by the Fund for Scientific Research (FWO-Vlaanderen), by the ERC advanced grant SNLSID, under contract 320378, and by the VLAIO Innovation Mandate project number HBC.2016.0235.

REFERENCES

[1] B. Peeters and G. De Roeck., “Stochastic system identification for operational modal analysis: a review”, ASME Journal of Dynamic Systems, Measurement, and Control, 123(4), pp. 659-667, 2001

[2] A. B. Jont, “Short Time Spectral Analysis, Synthesis, and Modification by Discrete Fourier Transform”, IEEE Transactions on Acoustics, Speech, and Signal Processing. Vol. 25, issue 3, pp. 235–238, 1977

[3] P. Z. Csurcsia, “Nonparametric identification of linear time-varying systems”. Phd thesis. University Press, Zelzate, 2015

[4] P. Z. Csurcsia. and J. Lataire, “Nonparametric Estimation of Time-varying Systems Using 2D Regularization”, IEEE Transactions on Instrumentation and Measurement, 2016

[5] P. Z. Csurcsia and J. Schoukens, “Nonparametric Estimation of a Time-variant System: An Experimental Study of B-splines and the Regularization Based Smoothing”, IEEE International Instrumentation and Measurement Technology Conference, pp. 216-221, Pisa, Italy, 2015

[6] M Freedman and G. Zames, “Logarithmic variation criteria for the stability of systems with time-varying gains”, In SIAM Journal on Control and Optimization, Pp. 487-507, vol. 6, no.3, 1968

[7] G. Pillonetto and A. Aravkin, “A new kernel-based approach for identification of time-varying linear systems”, IEEE International Workshop on Machine Learning for Signal Processing. pp. 1-6, Reims, 2014

[8] G- Pillonett, F. Dinuzzo, T. Chen, G. De Nicolao and L. Ljung, “Kernel methods in system identification, machine learning and function estimation: A survey”, In Automatica, 50(3), Pp. 657-682, 2014

[9] C.E. Rasmussen and C.K.I. Williams, “Gaussian Processes for Machine Learning”. MIT press, Cambridge, 2006

[10] L. A. Zadeh L.A. “A general theory of linear signal transmission systems”, Journal of the Franklin Institute. Vol. 253, no. 4, p. 293–312, 1952

[11] L.A. Zadeh, “Time-Varying Networks I”, Proceedings of the IRE, 1961

[12] B. Peeters, P. Karkle, M. Pronin and R. Van der Vorst R, “Operational Modal Analysis for in-line flutter assessment during wind tunnel testing”, 15th International Forum on Aeroelasticity and Structural Dynamic, 2011

Towards Reliable Benchmarks of Timed Automata

Rebeka Farkas, Gábor Bergmann
Budapest University of Technology and Economics, Department of Measurement and Information Systems
Email: {farkasr,bergmann}@mit.bme.hu
MTA-BME Lendület Cyber-Physical Systems Research Group

***Abstract*—The verification of the time-dependent behavior of safety-critical systems is important, as design problems often arise from complex timing conditions. One of the most common formalisms for modeling timed systems is the timed automaton, which introduces clock variables to represent the elapse of time. Various tools and algorithms have been developed for the verification of timed automata. However, it is hard to decide which one to use for a given problem as no exhaustive benchmark of their effectiveness and efficiency can be found in the literature. Moreover, there does not exist a public set of models that can be used as an appropriate benchmark suite. In our work we have collected publicly available timed automaton models and industrial case studies and we used them to compare the efficiency of the algorithms implemented in the Theta model checker. In this paper, we present our preliminary benchmark suite, and demonstrate the results of the performed measurements.**

I. Introduction

Since their introduction by Alur and Dill [1], timed automata has become one of the most common formalisms for modeling and verification of real time systems. There is a wide range of application areas, such as communication protocols [2] and digital circuits [3]. There are many extensions of the formalism, such as the *probabilistic timed automaton* that is able to represent stochastic behavior or the *parametric timed automaton* that can describe parametric timing properties.

The key challenge of the verification of timed automaton-based models is the same as in case of any formalism: developing efficient and scalable algorithms that can be applied in practice. Several algorithms and tools have been designed for this purpose, that may differ in the supported formalisms and queries. Since these algorithms are as diverse as the problems they address, a single best one can not be chosen: for each algorithm there are classes of models, that can be verified efficiently and other classes where other approaches would be more suitable to use. This raises the need for some guidelines to decide which tool to use for a given model.

A possible solution is to perform an exhaustive benchmark on a set of relevant problems that can later be used to determine which approach is the most suitable for a given problem, however, this would require a set of relevant case studies to use as inputs. Unfortunately such a benchmark suite is not available for timed automata.

Our goal is to provide a set of timed automaton models (and corresponding queries) that can be used as a benchmark suite for comparing the efficiency of tools and algorithms developed for the verification of timed automata. It is important that the benchmark suite should allow the performed measurements to fulfill the requirements of a reliable benchmark [4], [5].

In this paper we present our preliminary benchmark suite that we assembled based on this principle and we demonstrate its usability by measurements that we performed on the algorithms implemented in the Theta [6] model checker.

II. Background

A. Timed automata

1) Basic definitions: Clock variables are a special type of variables, whose value is constantly increasing as the time elapses. The only operation on clock variables is the *reset* operation that sets the value of a clock to a constant. It is an instantaneous operation, after which the value of the clock will continue to increase. The set of clock variables is denoted by $\mathcal{C}$.

A *clock constraint* is a conjunctive formula of *atomic clock constraints*. There are two types of atomic clock constraints:

- the *simple constraint* of the form $x \sim n$ and
- the *diagonal constraint* of the form $x - y \sim n$,

where $x, y \in \mathcal{C}$, $\sim \in \{\leq, <, =, >, \geq\}$ and $n \in \mathbb{N}$. In other words a clock constraint defines upper and lower bounds on the values of clocks and the differences of clocks. The set of clock constraints is denoted by $\mathcal{B}(\mathcal{C})$.

A *timed automaton* extends a finite automaton with clock variables. It is formally defined as a tuple $\langle L, l_0, E, I\rangle$ where

- L is the set of locations (i.e. control states),
- $l_0 \in L$ is the initial location,
- $E \subseteq L \times \mathcal{B}(\mathcal{C}) \times 2^{\mathcal{C}} \times L$ is the set of edges and
- $I : L \to \mathcal{B}(\mathcal{C})$ assigns invariants to locations [7].

The edges in E are defined by the source location, the guard (represented by a clock constraint), the set of clocks to reset, and the target location.

2) Extensions of timed automata: Many extensions have been invented to increase the descriptive and the expressive power of timed automata.

A *network of timed automata* [8] is the parallel composition of a set of timed automata. Communication is possible by shared variables or hand-shake synchronization using *actions* on the edges. Constructing networks of timed automata does not increase the expressive power, as a network of timed automata can be transformed into an equivalent timed automaton, but it does increase the understandability of the model.

A *parametric timed automaton* [9] is an extension of a timed automaton, where instead of constants, unbound parameters are also allowed to appear in clock constraints. Verification of parametric timed automata focuses on finding the parameter bindings that satisfy certain properties.

A timed automaton extended with *data variables* (extended timed automaton [8]) is an extension, where besides clock variables, data variables (discrete variables such as integers, bools, etc.) are also allowed. Data variables can also appear in constraints to enable transitions (*data guards*) and can be modified by transitions (*update*). However, clock variables are not allowed to appear in data guards or updates.

3) Verification: The verification approach depends on the type of property to check. Analysing safety properties is reduced to searching for reachable states in the state space, while checking liveness properties is solved by looking for certain cycles (strongly connected components) in the state space. [10]

B. Reliable benchmarks

Many requirements of reliable benchmarks are described in the literature [4], [5]. Although most methodologies focus on the execution of the benchmarks and not the inputs, some of the requirements do raise expectations for the benchmark suite.

Realistic The inputs should resemble models from industrial case studies.

Simple Just like other aspects of the benchmark, the input models must be understandable.

Scalable Scalable models are necessary in order to support a wide range of approaches.

Portable For portability, the models and the properties should be defined in a widely supported format.

Public A reproducible benchmark requires a public benchmark suite.

Diverse In order to be able to analyze the strengths of a wide range of tools, the models and the problems should be classified and many classes of inputs should be included.

III. Related work

A. Model checking competitions

In case of most common formalisms, generally accepted benchmarks are carried out by model checking competitions, such as the Model Checking Contest[1] for Petri Nets, the SV-COMP[2] on software verification and the Hardware Model Checking Competition[3] for hardware models. These competitions are an effective way of assembling and maintaining realistic benchmark suites and performing reliable benchmarks. However, there is no such competition for timed automata.

B. Benchmarks of timed automata-based models

Since there does not exist a generally accepted benchmark suite, each tool uses its own set of inputs to demonstrate the efficiency of its algorithms. Table I summarizes the characteristics of input sets of the most common tools: name of the tool, input format, total number of models, number of scalable

[1]https://mcc.lip6.fr/

[2]https://sv-comp.sosy-lab.org/2018/

[3]http://fmv.jku.at/hwmcc17/

Tool	input	#models	#scalable	query	ref
UPPAAL	xml, xta	9	3	true	true
Kronos	aut	5	3	true	true
PAT	xml, xta	5	5	false	false
MCTA	xta	5	5	true	true
TChecker	xta	6	5	false	true
REDLIB	d	5	5	true	false
Shrinktech	aut	9	3	false	true
CosyVerif	grml	15	0	false	true
PRISM	pta	7	2	true	false

TABLE I
Available benchmark suites

models, whether they describe a property to check, and whether they provide references to papers where these models are described. While most presented tools operate on networks of extended timed automata, CosyVerif's BenchKit [11] consists of parametric timed automata, and the PRISM benchmark suite contains probabilistic timed automata.

As it can be seen in the table, most model checkers have their own input language. However, the most common input format is `xta` defined by UPPAAL [8]. The presented benchmark suites are small, and share many models – e.g. the scalable models are the same for all benchmark suites, except for MCTA that uses another kind of scalability (see Section IV-D). In many cases the properties to verify are not defined, instead, during benchmarks the complete statespace of the model is explored. In some cases the source and the description of the models are also missing.

In conclusion, the current benchmark suites are small, there are very few scalable models, and portability, diversity and understandability is not always ensured.

IV. Xta Benchmark Suite

In this section, we present the Xta Benchmark Suite that is a collection of inputs we propose for comparing the efficiency of timed automaton verification algorithms. The suite was constructed to meet the requirements enlisted in Section II-B. While Table II describes the contained models, the complete suite, including the models, the queries and the references can be found online[4]. Note, that this is a preliminary suite.

A. Sources

The benchmark suite consists of models from existing benchmarks of UPPAAL, PAT, MCTA and CosyVerif, as well as UPPAAL case studies and other public models. Industrial case studies were included in order to allow realistic benchmarks. References to papers describing the models are provided in order to ease understandability and to assure the benchmark inputs are realistic, public and portable.

B. Format

In order to help portability, the `xta` format was chosen for storing the models, as most timed model checkers are able to parse this format (even if they are not able to transform their own input language to `xta`). Another advantage of this format

[4]https://github.com/farkasrebus/XtaBenchmarkSuite

Name	Description	Source	Type	Scalable
FISCHER	Fischer's mutual exclusion protocol	UPPAAL benchmark	MutEx protocol	true
CSMA	The CSMA/CD protocol	UPPAAL benchmark	CD protocol	true
FDDI	Token Ring/FDDI protocol	UPPAAL benchmark	protocol	true
BANDO	Bang-Olufsen protocol	UPPAAL benchmark	CD protocol	false
BOCDPFIXED	Bang-Olufsen Collision Detection Protocol	UPPAAL benchmark	CD protocol	false
BOCDP	BOCDP - original, faulty version	UPPAAL benchmark	CD protocol	false
CRITICAL	Critical region	PAT benchmark	MutEx protocol	true
LYNCH	Lynch-Shavit protocol	PAT benchmark	MutEx protocol	true
BAWCC	Business Agreement with Coordination Completion protocol	UPPAAL case studies	protocol	false
BAWCCENHANCED	BAWCC - enhanced version	UPPAAL case studies	protocol	false
SCHEDULE	Schedulability Framework model	UPPAAL case studies	algorithm	false
STLS	Single Tracked Line Segment	MCTA benchmark	system	false
MUTEX	Mutual exclusion protocol	MCTA benchmark	MutEx protocol	false
FAS	Fire Alarm System	[12]	system	false
SOLDIERS	The soldiers problem	public model	problem	false
ENGINE	A running engine	public model	system	false
ANDOR	And-Or circuit	CosyVerif BenchKit	circuit	false
BANGOLUFSEN	Bang-Olufsen protocol	CosyVerif BenchKit	protocol	false
EXSITH	Sluice	CosyVerif BenchKit	system	false
FLIPFLOP	Flip-flop circuit	CosyVerif BenchKit	circuit	false
LATCH	Latch circuit	CosyVerif BenchKit	circuit	false
MALER	Maler's Jobshop algorithm	CosyVerif BenchKit	algorithm	false
RCP	Root Connection Protocol	CosyVerif BenchKit	protocol	false
SIMOP	SIMOP Networked Automation System	CosyVerif BenchKit	system	false
SRLATCH	SR-latch circuit	CosyVerif BenchKit	circuit	false
TRAIN	Train gate controller protocol	CosyVerif BenchKit	system	false

TABLE II
XTA BENCHMARK SUITE MODELS

is that it is possible to define scalable models in a way that the size of the model can be modified by setting a single constant.

C. *Transformations*

In many cases the models were transformed. In case of timed automata, UPPAAL was used to transform the `xml`-based models to `xta`. Parametric timed automata are stored in an `xml` based format (`grml`) by CosyVerif, that was transformed to `xta` programmatically. As the `xta` format does not allow parameters, they were parsed as `const int` and manually bound to a value taken from the Shrinktech model of the same system, where it was available. Additionally, one of the models (*TRAIN*) was modified to a generalized version (that allows more trains) in order to increase scalability.

D. *Scalability*

In most existing benchmark suites, scalable models represent communication protocols and scalability is introduced by changing the number of participants – i.e. introducing new timed automata to the network that behave similarly to the original ones. On the other hand, in the MCTA benchmark suite scalability is introduced to the model by increasing constants used in clock constraints.

In the Xta Benchmark Suite only the former type of models are considered scalable, since the most commonly used algorithms are not sensitive to the values of bounds [7].

E. *Queries*

While algorithms based on state space exploration can operate without a property to check, in order for benchmarks to be realistic, the Xta Benchmark Suite also provides queries for most models. This allows to perform measurements on a wider range of algorithms – such as backward exploration, that requires the target state to initiate from, or search algorithms with heuristics that are efficient in finding an execution trace to the target state but inefficient in state space exploration.

F. *Classification*

In order to demonstrate diversity the models were classified according to the type of problem they represent. This also determines the types of properties to be checked. More information on the classification can be found online.

V. MEASUREMENTS

Algorithms implemented in the Theta model checking framework were executed on the benchmark suite. Unlike the usual purpose of benchmarks, these measurements were performed to analyze the benchmark suite and not the algorithms.

A. *Procedure*

The measurements were executed on a virtual 64 bit Windows 7 operating system with a 2 core CPU (2.50 GHz) and 4 GB of memory. Each algorithm was run 5 times on each input and the average of the runtimes was taken. The timeout was 5 minutes (300,000 milliseconds).

While the complete suite consists of 26 models, those without a property to check were excluded as well as the ones containing elements that Theta does not support yet, such as broadcast channels. This reduced the number of inputs to 11.

The efficiency of six algorithms were compared. Algorithm *LU* is presented in [13], algorithm *ACT* is the improvement of the algorithm described in [7] by applying the *activity* abstraction described in [14] using lazy evaluation and algorithms *BINITP, SEQITP, WEAKBINITP,* and *WEAKSEQITP* are variants of the algorithms described in [15].

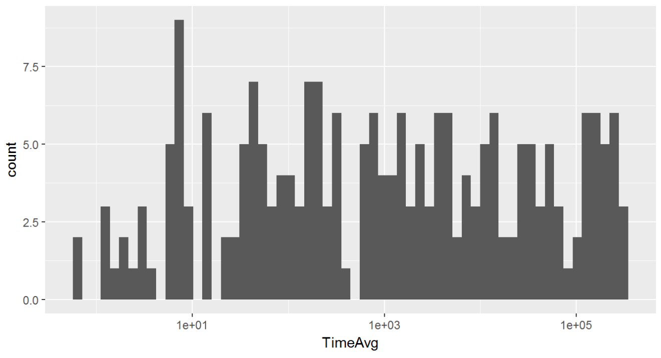
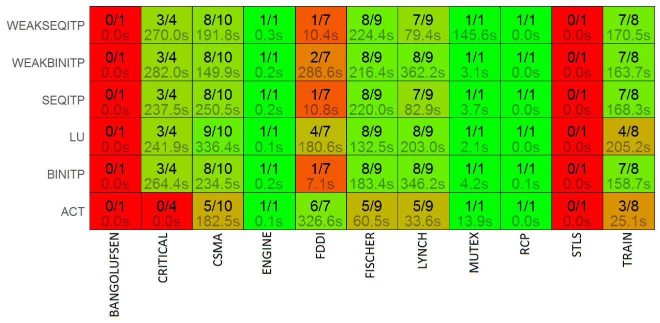


Fig. 1. Distribution of execution times



	BANGOLUFSEN	CRITICAL	CSMA	ENGINE	FDDI	FISCHER	LYNCH	MUTEX	RCP	STLS	TRAIN
WEAKSEQITP	0/1 0.0s	3/4 270.0s	8/10 191.8s	1/1 0.3s	1/7 10.4s	8/9 224.4s	7/9 79.4s	1/1 145.6s	1/1 0.0s	0/1 0.0s	7/8 170.5s
WEAKBINITP	0/1 0.0s	3/4 282.0s	8/10 149.9s	1/1 0.2s	2/7 286.6s	8/9 216.4s	8/9 362.2s	1/1 3.1s	1/1 0.0s	0/1 0.0s	7/8 163.7s
SEQITP	0/1 0.0s	3/4 237.5s	8/10 250.5s	1/1 0.2s	1/7 10.8s	8/9 220.0s	7/9 82.9s	1/1 3.7s	1/1 0.0s	0/1 0.0s	7/8 168.3s
LU	0/1 0.0s	3/4 241.9s	9/10 336.4s	1/1 0.1s	4/7 180.6s	8/9 132.5s	8/9 203.0s	1/1 2.1s	1/1 0.0s	0/1 0.0s	4/8 205.2s
BINITP	0/1 0.0s	3/4 264.4s	8/10 234.5s	1/1 0.2s	1/7 7.1s	8/9 183.4s	8/9 346.2s	1/1 4.2s	1/1 0.1s	0/1 0.0s	7/8 158.7s
ACT	0/1 0.0s	0/4 0.0s	5/10 182.5s	1/1 0.1s	6/7 326.6s	5/9 60.5s	5/9 33.6s	1/1 13.9s	1/1 0.0s	0/1 0.0s	3/8 25.1s

Fig. 2. Summary of results

B. *Results*

The distribution of execution times can be seen in Figure 1. Figure 2 summarizes the success rates and runtimes of the executions. The rows correspond to algorithms and the columns correspond to inputs.

The first row of a cell contains a fraction representing the success rate of the algorithm on the input: the denominator is the total number of instances of the model (in case of non-scalable models it is one) and the nominator is the number of instances successfully verified by the algorithm. The second row presents the runtime on the largest instance that was successfully verified or $0.0s$ if there was none.

Models *BANGOLUFSEN* and *STLS* turned out to be too large to be verified by any of the examined algorithms in the given time. Executing the algorithm *ACT* on input *CRITICAL* resulted in an exception for all instances.

Results show that for each algorithm the benchmark suite contained at least one model where the applicability of the algorithm was demonstrated and that the execution times are well distributed on the logarithmic scale.

VI. Conclusions

This paper proposed a benchmark suite to perform reliable benchmarks of verification algorithms for timed automata. The requirements of such a benchmark suite were identified, then a preliminary collection of inputs were presented. To demonstrate the applicability of the proposed benchmark suite the models were used to compare the algorithms implemented in the Theta model checking framework. The results of the presented benchmark suggest that the benchmark suite meets the described requirements.

Future works include increasing the size of the benchmark suite by importing more models from benchmarks of other tools or even other timed formalisms, focusing on scalable models. We also plan to include more industrial case studies that can be found in the literature.

References

[1] R. Alur and D. L. Dill, "The theory of timed automata," in *Real-Time: Theory in Practice, REX Workshop, Mook, The Netherlands, June 3-7, 1991, Proceedings*, 1991, pp. 45–73.

[2] J. Bengtsson, W. O. D. Griffioen, K. J. Kristoffersen, K. G. Larsen, F. Larsson, P. Pettersson, and W. Yi, "Verification of an audio protocol with bus collision using UPPAAL," in *Computer Aided Verification, 8th International Conference, CAV '96, New Brunswick, NJ, USA, July 31 - August 3, 1996, Proceedings*, 1996, pp. 244–256.

[3] O. Maler and A. Pnueli, "Timing analysis of asynchronous circuits using timed automata," in *Correct Hardware Design and Verification Methods, IFIP WG 10.5 Advanced Research Working Conference, CHARME '95, Frankfurt/Main, Germany, October 2-4, 1995, Proceedings*, 1995, pp. 189–205.

[4] K. Huppler, "The art of building a good benchmark," in *Performance Evaluation and Benchmarking, First TPC Technology Conference, TPCTC 2009, Lyon, France, August 24-28, 2009, Revised Selected Papers*, 2009, pp. 18–30.

[5] D. Beyer, S. Löwe, and P. Wendler, "Reliable benchmarking: requirements and solutions," *International Journal on Software Tools for Technology Transfer*, Nov 2017.

[6] T. Tóth, A. Hajdu, A. Vörös, Z. Micskei, and I. Majzik, "Theta: a framework for abstraction refinement-based model checking," in *Proceedings of the 17th Conference on Formal Methods in Computer-Aided Design*, D. Stewart and G. Weissenbacher, Eds., 2017, pp. 176–179.

[7] J. Bengtsson and W. Yi, "Timed automata: Semantics, algorithms and tools," in *Lectures on Concurrency and Petri Nets*, ser. LNCS. Springer Berlin Heidelberg, 2004, vol. 3098, pp. 87–124.

[8] J. Bengtsson, K. G. Larsen, F. Larsson, P. Pettersson, and W. Yi, "UPPAAL - a tool suite for automatic verification of real-time systems," in *Hybrid Systems III: Verification and Control, Proceedings of the DIMACS/SYCON Workshop on Verification and Control of Hybrid Systems, October 22-25, 1995, Ruttgers University, New Brunswick, NJ, USA*, 1995, pp. 232–243.

[9] R. Alur, T. A. Henzinger, and M. Y. Vardi, "Parametric real-time reasoning," in *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, 1993, pp. 592–601.

[10] C. Baier and J. Katoen, *Principles of model checking*. MIT Press, 2008.

[11] F. Kordon and F. Hulin-Hubard, "Benchkit, a tool for massive concurrent benchmarking," in *14th International Conference on Application of Concurrency to System Design, ACSD 2014, Tunis La Marsa, Tunisia, June 23-27, 2014*, 2014, pp. 159–165.

[12] S. F. Arenis, B. Westphal, D. Dietsch, M. Muñiz, and A. S. Andisha, "The wireless fire alarm system: Ensuring conformance to industrial standards through formal verification," in *FM 2014: Formal Methods - 19th International Symposium, Singapore, May 12-16, 2014. Proceedings*, 2014, pp. 658–672.

[13] F. Herbreteau, B. Srivathsan, and I. Walukiewicz, "Lazy abstractions for timed automata," in *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, 2013, pp. 990–1005.

[14] C. Daws and S. Yovine, "Reducing the number of clock variables of timed automata," in *Proceedings of the 17th IEEE Real-Time Systems Symposium (RSS '96*. Washington - Brussels - Tokyo: IEEE, Dec. 1996, pp. 73–81.

[15] T. Tóth and I. Majzik, "Lazy reachability checking for timed automata using interpolants," in *Formal Modelling and Analysis of Timed Systems*, ser. Lecture Notes in Computer Science, A. Abate and G. Geeraerts, Eds. Springer, 2017, vol. 10419, pp. 264–280.

Mix-and-Match Composition in the Gamma Framework

Bence Graics, Vince Molnár
Budapest University of Technology and Economics,
Department of Measurement and Information Systems
Budapest, Hungary
Email: bence.graics@gmail.com, molnarv@mit.bme.hu

Abstract—**The Gamma Statechart Composition Framework is a modeling tool that supports the hierarchical composition of statechart components with well-defined compositonal semantics, as well as source code generation and formal verification. The purpose of the framework is to provide common ground for modeling and verification tools, as well as to support component-based system design building on existing statechart modeling tools. Currently, the framework has a single composition semantics, which executes the components in a lockstep fashion. This paper presents a new composition language for the Gamma Framework, adding support for two more semantics. Asynchronous-reactive semantics supports the proper abstraction of distributed communication, synchronous-reactive supports the modeling of highly synchronous communication, and cascade composition is a sequential decomposition of a single function.**

I. Introduction

Statecharts [1] are a widely used formalism to describe the behavior of reactive systems, which process stimuli from the environment and react with respect to their internal states. Statecharts introduce complex elements to aid the modeling of such systems, e.g., variables, hierarchical state refinement, history states and complex transitions (e.g., inner transitions).

The requirements such systems have to meet are getting more complex, which can result in very large system models, hindering verifiability, maintenance and extensibility. A well-known solution for managing complexity is decomposition. In case of statecharts, one way of decomposition is to define individual reactive components that, by means of communication, realize a more complex behavior. There are a number of modeling tools that aim to support this practice with various model-driven software development techniques such as code generation and verification.

The Gamma Statechart Composition Framework is one such tool, providing a layer for composing individual statechart components (possibly coming from other tools) while extending the capabilities of automatic code generation and verification and validation (V&V). Functionalities of the Gamma Framework as well as a case study are presented in [2]. In this paper, we propose a new composition language for Gamma that enables the hierarchical mixing of different composition semantics, with a focus on features and modeling aspects.

Asynchronous-reactive: Such models represent a set of components that are executed independently. Asynchronous components communicate by means of messages and message queues. This semantics is convenient when decomposition is both logical and physical, e.g., for distributed controllers.

Synchronous-reactive: A synchronous model represents a coherent unit consisting of strongly coupled but concurrent components which communicate in a synchronous manner using signals. This semantics is suitable for the logical decomposition of synchronous systems or the modeling of hardware-related designs.

Cascade: Cascade models represent a set of filters that are applied sequentially to derive an output from an input. This variant supports the design of adapters, runtime monitors and units with a batch-like execution.

The rest of the paper is structured as follows. Section II presents a short summary of tools that inspired the design of the composition language. The elements of the language itself and an example are introduced in Section III. Finally, Section IV provides concluding remarks and ideas for future work.

II. Related Tools

We present three tools that also support the *mixing of different composition semantics* for component-based systems.

Ptolemy II[1] [3], [4] is an open-source framework aiming to support the modeling of hierarchical composite systems with numerous component variants and communication semantics. Communication semantics are determined by *directors*, which are responsible for defining a *model of computation* on a particular hierarchy level. Various directors (e.g., process network, synchronous data flow or synchronous reactive) can be combined through different hierarchy levels, facilitating the design of complex model behavior. One of the main strengths of Ptolemy II is its simulation capability. However, source code generation and formal verification are not supported.

BIP[2] [5], [6] is a modeling framework that focuses on the formal definition of heterogeneous systems. BIP offers a language to define hierarchical composite models, where the interactions of constituent components are based on *synchronization*. BIP defines a clear operational semantics that describes the behavior for both atomic components (transition

[1]http://ptolemy.eecs.berkeley.edu/

[2]http://www-verimag.imag.fr/Rigorous-Design-of-Component-Based.html?lang=en

system model) and compound components (rigorous rules for component interactions). Moreover, BIP offers a comprehensive tool set, which provides model transformers for third-party models, code generators, and formal verification capabilities.

Stateflow[3] [7] is a commercial framework that supports the modeling of reactive systems. Stateflow supports the design of statecharts and provides various scheduling algorithms. Furthermore, Stateflow supports the simulation, validation and verification of models as well as source code generation. Stateflow is a mature commercial software product with professional support. As such, the possibilities of extending or integrating it with other software is limited. Furthermore, it is very expensive even for research purposes.

Gamma was designed to provide an extensible framework for the design of statechart-based composite systems, inspired by the merits and limitations of the presented tools.

III. The Gamma Composition Language

The *Gamma Composition Language* (GCL) supports the definition of communicating composite models built from individual reactive components. The design of composite systems starts with the definition of interfaces, which define the possible event types that can be transmitted between components. The interfaces can be realized by ports of components, which can be connected with channels, enabling communication. Communicating components can be wrapped by composite models, creating an independent composite reactive unit with rigorously defined interaction patterns.

A. Communication Elements

In the GCL, components communicate through *ports*. Each port defines a point of service through which certain *event notifications* can be sent or received. An event notification (or event for short) is a piece of information passed between components, which can also have *parameters* to forward data. An event is called *message* in case of asynchronous components and *signal* in case of synchronous components. Events are declared on *interfaces*, which may be realized by ports. An event may be declared as *input*, *output* or *in/out*, which means that it can be received, sent or both through the realizing port. The declared direction will be reversed, however, if the port does not *provide*, but *require* the interface, which are the two possible modes in which a port can realize an interface. A *broadcast interface* is a special type of interface on which every event is *output*. This approach is similar to how the Franca Interface Definition Language defines interfaces.[4]

The concept of input and output events with the two modes of interface realization may be unusual at first sight, since ports in UML-like modeling languages support event reception and method declarations only, both of which are services that can be invoked. Our goal with this solution is to investigate the possibilities of precise interface-based communication in the domain of reactive systems. On the other hand, it is possible to use only *out* events on every interface – then provided mode is “output” mode and required mode is “input” mode.

The following snippet defines an interface that has a single input event with an integer parameter as well as an interface with a single output event, that is, a broadcast interface.

```
interface Status {
  in event query
  // Event with integer parameter
  out event status(code : integer)
}
interface PoliceInterrupt { // A broadcast interface
  out event interrupt
}
```

B. Components

Components are the basic building blocks of composite Gamma models. The declaration of a component always defines one or more ports in the header which may be referred to in the definition, as presented in the following snippet.

```
sync Crossroad [
  port police : requires PoliceInterrupt,
  port priorityLight : provides LightCommands,
  port secondaryLight : provides LightCommands
]
```

A component can be either *atomic*, wrapping a single statechart, or *composite*, wrapping one or more component instances. Statecharts can be defined in the *Gamma Statechart Language* (GSL), presented in [2]. Composite components may adopt three types of semantics, but regardless of that, they will comprise of the same basic modeling elements: component instances, port bindings and channels.

a) Component instance: Component declarations can be instantiated in composite component definitions, but they may not contain themselves (not even transitively). Composite components may contain instances of other composite components, yielding a hierarchical composition. Component instances inherit the declared ports through which they can communicate.

```
// Instance of a component type
component crossroadInstance : Crossroad
```

b) Port binding: The port binding element is responsible for mapping the declared ports of a composite component to one of the ports of its constituent components.

```
// Binding component ports to internal ports
bind police -> controller.policeInterrupt
bind priorityLight -> priorityLight.lightCommands
```

In the example above, the *police* port of the composite Crossroad component declaration is mapped to the *policeInterrupt* port of the component instance *controller*. This means that events received through the *police* port will be instantaneously forwarded to the *policeInterrupt* port, and events sent through the *policeInterrupt* port will be sent through the *police* port.

c) Channel: Communication may happen through *channels*. *Simple channels* can connect two ports if they implement the same interface but in different modes, i.e., the signal directions will be exactly the opposite on the two ports. *Broadcast*

[3]https://www.mathworks.com/products/stateflow.html

[4]http://www.eclipse.org/proposals/modeling.franca/

channels are similar, but they allow a single port *providing a broadcast interface* to be connected to multiple ports *requiring the same broadcast interface*. Note that the language will restrict the usage of channels in certain composition semantics, which is discussed in Section III-C.

```
// A simple and a broadcast channel
channel [controller.priotityControl] -o)-
  [priorityLight.control]
channel [controller.policeInterrupt] -o)-
  [priorityLight.police, secondaryLight.police]
```

C. Composite Component Variations

Composite components can be *synchronous* or *asynchronous*, which will determine how they receive events and how they execute their constituent components.

1) Synchronous Components: Synchronous components represent models that communicate in a synchronous manner using *signals*. Constituent components of synchronous composite components must be synchronous themselves and are executed in a lockstep fashion, triggered by the enclosing asynchronous component or an external actor. When executed, synchronous components process incoming signals and produce output signals in accordance with their internal states. Input signals are not queued but sampled: upon execution, the component can access the most recent signal for every event on every port since the last execution (if any). Similarly, output signals are reset in every execution and every output event on every port may get a new signal assigned to it. Synchronous components in Gamma are *synchronous* and *cascade composite components* that can be freely mixed and *statechart definitions* as atomic components.

a) Synchronous composite component: During the execution of a synchronous composite component, every constituent component is executed exactly once. The execution has two phases: *1)* all components sample their inputs, then *2)* outputs and new internal states are computed. This strategy guarantees that an output generated by a component will affect other components only in the next execution – therefore the execution order is insignificant and the composition of deterministic components yields a deterministic composite component. This behavior was one of the most important design goals for synchronous-reactive compositions.

The only case that could violate the deterministic behavior would arise when an input event has more than one sources. In this case, one signal would overwrite the other, and their "order" would be unspecified. To avoid this, the language restricts the way ports can be connected with channels: every port may be the endpoint of exactly one channel *or* be bound to exactly one port of the enclosing component.

b) Cascade composite component: Conceptually, cascade composite components represent a set of "filters" through which inputs are transformed into outputs. Therefore, constituent components will immediately see the output signals of other components in the same composite component. This is achieved by merging the sampling and computation phases and performing them both when executing a component. Deterministic behavior is achieved by executing the components in the *order of their instantiation*.

This strategy guarantees that the effects of an input signal will be observable in the immediate output of the composite component (in case of synchronous composite components, the effect may be delayed by several execution cycles). Signals sent through feedback connections (i.e., when a component sends a signal to another that comes earlier in the execution order) will be delayed until the next execution (they may be used for e.g., abort signals from monitor components). Note that the synchronous and cascade composite components are semantically incompatible, i.e., there are modeling designs which can be described in only one of the variants.

The typical arrangement of a synchronous or cascade composite component definition is as follows.

```
[sync|cascade] Crossroad [
  // Port declarations
  port police : requires PoliceInterrupt
  ...
] {
  // Component instances
  component controller : Controller
  ...
  // Binding composite model ports to internal ports
  bind police -> controller.policeInterrupt
  ...
  // Channel definitions connecting internal ports
  channel [controller.priorityControl] -o)-
    [priorityLight.control]
  ...
}
```

2) Asynchronous Components: Asynchronous components represent independently running instances that communicate with *messages*, collected in *message queues*. There is no guarantee on the execution time or the execution frequency of components. There are two types of asynchronous components in Gamma: *asynchronous composite components* and *synchronous component wrappers*.

a) Synchronous component wrapper: A synchronous component wrapper is used to declare an asynchronous component implemented by a single synchronous component, facilitating the hierarchical mixing of the composition variants. In addition to the ports of the wrapped component, synchronous component wrappers may declare additional ports for control messages, as well as zero or more *clocks*, which emit *tick* events at defined timed intervals.

A synchronous component wrapper has one or more *message queues*, which have the following attributes: *capacity* specifies the maximum number of messages that can be stored in the particular queue; *priority* specifies the order in which message queues are processed during the execution of the asynchronous component (the next message is always retrieved from a non-empty queue with the highest priority); *event types* specify the messages that will be put in the particular queue. Messages can be referred to either by specifically naming an event on a port, referring to all events on a port (with the *any* keyword instead of an event), or specifying the name of a clock to refer to its tick event.

During execution, messages are retrieved individually from messages queues. A message is always taken from the highest priority non-empty queue. If the particular message was received on a port that is implicitly derived from the wrapped component, the message is converted to a signal (as synchronous components communicate with signals) and transmitted to the wrapped synchronous component (potentially overwriting previously sent signals). Otherwise, the message is not transmitted.

A synchronous component wrapper also has one or more *control specifications*, which describe when and how to execute the wrapped component. The trigger messages can be specified by referring to events as in case of message queues, while the execution mode can be one of the following: *"run once"* (*run* keyword) to trigger a single execution of the wrapped component; *"run to completion"* (*full step* keyword) to repeatedly execute the wrapped component until no more internal signals are generated (applicable only to composite components); *reset* (*reset* keyword) to reinitialize the wrapped component to its initial state.

The following code snippet presents an example wrapper for the previously defined *CrossroadComponent*, which defined a control port named *execution*. Messages received on this port are stored in the higher priority *executionQueue*, whereas the messages of additional (implicit) ports are stored in the other queue *crossroadsQueue* which has a capacity of 5. According to the control specifications, upon processing an *execute* message, the wrapped component is run to completion, while either a clock signal or an *interrupt* signal of port *police* triggers a single execution.

```
async AsyncCrossroad of CrossroadComponent [
  // Additional control ports
  port execution : provides Executable
] {
  // Clock definitions
  clock clockSignal(rate=100ms)
  // Control specifications
  when execution.execute / full step
  when clockSignal / run
  when police.interrupt / run
  // Message queues
  queue executionQueue(priority=2) {
    execution.execute, clockSignal
  }
  queue crossroadsQueue(priority=1, capacity=5) {
    police.any, priorityLight.any,
    secondaryLight.any
  }
}
```

b) Asynchronous composite component: Asynchronous composite components support the hierarchical definition of asynchronous components, just like synchronous composite components do for the synchronous case. The structure of asynchronous composite components is identical to their synchronous counterparts, but the type of constituent components must be asynchronous, i.e., either synchronous component wrappers of other composite components. The execution semantics of constituent components is totally asynchronous, any component may be executed any time, given that their message queues are not empty. Produced messages are placed into message queues of components on the other end of channels, and there is no restriction on how ports are connected. Note that the hierarchy of asynchronous components may always be flattened without affecting the meaning of the model.

IV. Conclusion and Future Work

In this paper, we have proposed an extension to the composition language of the Gamma Framework to support the mixing of three semantic variants for the composition of reactive components: asynchronous-reactive, synchronous-reactive and cascade semantics. Asynchronous components represent independently running components, which communicate with messages stored in message queues. This semantics is suitable for designing distributed or parallel processes. Synchronous-reactive components are useful for modeling a single executing unit consisting of multiple, functionally decomposed components. This composition mode is for the design of different aspects of a complex, but single-threaded component. Cascade composition is practical for designing units with pipeline-like behavior: the input fed into the model is processed by multiple consecutive filters in a single run.

Subject to future work, we plan to extend the code generation and formal verification services of Gamma to support the proposed language and the semantic variants it introduces. For code generation, we already have code templates for the more significant methods. The formal analysis of asynchronous-reactive models, however, will be worth more research, as the interleaving semantics of asynchronous models poses a serious challenge to model checkers that should be handled in the transformation to the formal input models of these tools.

Acknowledgment

Partially supported by the ÚNKP-17-2-I and ÚNKP-17-3-I New National Excellence Program of the Ministry of Human Capacities.

References

[1] D. Harel, "Statecharts: A visual formalism for complex systems," *Sci. Comput. Program.*, vol. 8, no. 3, pp. 231–274, Jun. 1987.

[2] B. Graics, "Model-Driven Design and Verification of Component-Based Reactive Systems," Bachelor's thesis, Budapest University of Technology and Economics, 2016. [Online]. Available: https://gamma.inf.mit.bme.hu

[3] J. Eker, J. W. Janneck, E. A. Lee, J. Liu, X. Liu, J. Ludvig, S. Sachs, Y. Xiong, and S. Neuendorffer, "Taming heterogeneity - the ptolemy approach," *Proceedings of the IEEE*, vol. 91, no. 1, pp. 127–144, 2003. [Online]. Available: http://chess.eecs.berkeley.edu/pubs/488.html

[4] C. Brooks, "Ptolemy II: An open-source platform for experimenting with actor-oriented design," February 2016, poster presented at the "https://www.terraswarm.org/conferences/16/bears/" 2016 Berkeley EECS Annual Research Symposium. [Online]. Available: http://chess.eecs.berkeley.edu/pubs/1166.html

[5] S. Yovine, "BIP: Language and tools for component-based construction," 2007. [Online]. Available: http://users.cs.york.ac.uk/~burns/papers/bip.pdf

[6] M. D. Bozga, V. Sfyrla, and J. Sifakis, "Modeling synchronous systems in BIP," in *Proceedings of the seventh ACM international conference on Embedded software*. ACM, 2009, pp. 77–86.

[7] S. T. Karris, *Introduction to Stateflow with Applications*. Orchard Publications, 2007.

Adaptive Sequential Laboratory Diagnostic Tests: Joint Bayesian Analysis for Optimality

Zeyneb Guenfoud, Péter Antal
Department of Measurement and Information Systems
Budapest University of Technology and Economics
Budapest, Hungary
Email:{guenfoud, antal}@mit.bme.hu

***Abstract*—Laboratory diagnostic tests provide a fundamental, traditional level in clinical practice and biomedical research. Despite the detailed diagnostic characterization of individual laboratory tests, their overall interdependences has not been investigated. We summarize a probabilistic framework to define optimal measurements, which relies on comprehensive, multivariate probabilistic models of laboratory diagnostic tests formalized as probabilistic graphical models. Within the probabilistic framework we propose sequential inference schemes to improve requested tests. We discuss the challenges and propose scenarios for the integrated application of a decision support system optimizing the selection of laboratory tests with an interplay of clinicians and the laboratory. We also present results of preprocessing a dataset from a central laboratory of a large medical center, which will be the basis of later real-world evaluations.**

***Index Terms*—artificial intelligence, probabilistic graphical models, Bayesian learning, optimal decision, laboratory diagnostic test, cost efficiency**

I. Introduction

The availability of massive, electronic health datasets provides an unprecedented opportunity in early diagnosis and personalized medicine. Modern cornerstones of personal health information are the various molecular biological datasets corresponding to different biological levels and corresponding measurement technologies, such as genetics, transcriptomics or proteomics. However, clinical laboratories are still have a central role in clinical practice: for myriads of clinical requests about diagnostic tests from a wide range, they provide standardized, high-quality information with strict time-constraints. On the one hand, this separated service model focusing on the measurement of the requested tests may increase efficiency at the population level, e.g. optimizing the measurement process of multiple requests according to the laboratory infrastructure and workload. But on the other hand, it can decrease efficiency at the level of patient, e.g. the measurement process in an actual case could be guided by the measured information, optionally involving the clinical expert as well. Specifically, a sequential, adaptive measurement process of laboratory tests, potentially with an interaction between the laboratory and clinical diagnostician could result in the following:

1) *Canceled measurements* The laboratory could inform the clinician that certain requested tests are confidently predictable based on earlier measurements from the patient's history and from current measurements. Depending on the suspected disease and corresponding diagnostic protocol, the clinician could decide that the in silico predictions are sufficient and could cancel the pending requests.
2) *Extended measurements* The laboratory could inform the clinician that the value of certain not requested tests are abnormal with high confidence, predicted based on earlier measurements from the patient's history and from current measurements. Depending on the suspected disease and corresponding diagnostic protocol, the clinician could decide that the measurements could be indeed valuable for those variables and expand the request.

We investigate the following scenario of adaptive, sequential laboratory diagnostic tests, for which the assumptions are motivated by our real-world cooperation with the Central Laboratory of the Semmelweis University:

1) *Separated laboratory information* The laboratory has no access to the patient's medical history and current suspected diseases, but basic demographic information, such as gender and age and earlier laboratory tests for the patients may be available.
2) *Requested tests with urgent/compulsory subsets and suggested ordering* The clinician could ask the measurement of test(s) indicating also that certain measurements are urgent and/or compulsory. Suggested ordering for their sequential measurement can be also indicated.
3) *Predictable tests* The laboratory could inform the clinician that the value of certain tests are confidently predictable based on earlier measurements from the patient's history and from current measurements.

The central assumption of our approach is that laboratory diagnostic tests have a robust probabilistic dependency structure, in fact, the redundancy of the current set of tests is one of the main challenge in laboratory medicine [15]. Because in our scenario information about indications and diseases are not available, we focus on the separated, standalone dependency structure of the tests. Note that this property excludes the usage of information about well-known disease-specific multivariate tests. Furthermore, to simplify our task, as a first approximation, we ignore the temporal aspect of laboratory tests, e.g. we do not perform a time-series analysis and do not model that certain tests are used to monitor the result of a surgery.

Based on these assumptions, the main questions of our work is twofold:

1) *Predictable measurements* We try to estimate the distribution, variance and expected value of the number of correctly predictable measurements.
2) *Unmeasured abnormalities* We try to estimate the distribution, variance and expected value of the number of unmeasured tests with abnormal values.

Two notes are in order. Note that certain measurements are prescribed by medical protocols, e.g. to exclude vital conditions. Thus, the expected value of predictable measurements provides only an upper bound for the potentially avoidable laboratory tests. Analogously, certain measurements are trivially abnormal in certain medical conditions, so they are not requested to measure, consequently the expected value of unmeasured abnormalities is only an upper bound for missed, medically relevant measurements.

In short, our assumptions are twofolds. Firstly, it relies on a non-temporal learning phase, in which laboratory test measurements in a given, recent period are merged into a vectorial description as current state, and earlier measurements are neglected. This first phase results in a *a posteriori* distribution over models. Secondly, we will approximate inference for a given patient using this *a posteriori* distribution over models and perform a sequential inference in a given model using laboratory tests as evidences of the current state of a patient.

II. Earlier works

The broader context of our formalization and approximation for an adaptive, sequential use of laboratory diagnostic tests encompasses the full-fledged Bayesian decision theoretic framework, including actions for the selection of tests for measurements, for the rejection of a measured value and an action for stopping with the measurements and possibly suggesting interventions based on the diagnosis. The optimal selection of actions resulting in interventions and observations, leads to the concept of expected value of an experiment (EVE) [18].

Ignoring the potential interventional consequences, the utilization of the temporal sequence of measurements for a given patient can be seen as a time-series analysis, especially in the prequential and online learning approaches for non-stationer processes. This scenario corresponds to the monitoring of the result of a surgery using a panel of biomarkers for example in oncology.

The real-world constraints on measuring laboratory tests, especially the financial and temporal constraints, can be also investigated in the frameworks of active learning and budgeted learning.

The measurement itself of a laboratory test usually indicates an increased belief for a potential abnormal value, i.e. the informativeness of the mere availability of a measured test. Thus, the usual laboratory test datasets violate the missing-at-random (MAR) assumption and require special approaches [5].

Assuming that the inductive part results in a limited number of models, the adaptive, sequential use of laboratory tests in this phase corresponds various types of inferences in a complex, fixed model to explore the current state of a given patient. If utility functions are available, then value of information calculations could be used to support information gathering [4], [6], [7], [9], [11]. Lacking informative utility functions, general domain specific functions can be constructed and/or sensitivity of inference could be used to support information gathering. Another approach is to use explanation generation methods [3], [10].

The joint analysis of all laboratory tests is a natural extension of the network paradigm from diseases, genes, drugs, phenotypes and symptoms [20]. Indeed, both data mining and deep analysis of electronic health records are among the top priorities for improving health care [2], [8], with special emphasis on using laboratory diagnostic tests [12]–[17], [19].

Unfortunately, these earlier frameworks and methods are not directly applicable for this problem, so there are no available benchmark results.

III. Domain and datasets

The raw dataset contains all results for the measurements of 225 most relevant laboratory blood tests between 2011 and 2015 October at the Central Laboratory of the Semmelweis University. From this 4-year period, the original dataset contains 13,754,888 measurements from 1,376,759 orders for 1,392 laboratory tests and 202,976 persons.

The measurements are grouped by their orders, which usually correspond to a visit to a medical professional and a respective blood sampling. For each order, the urgency of the measurement and the institute of the doctor ordering the tests are indicated, but not used in the current analysis. For each patient, gender and age will be available, but could not yet accessed. The identifiers and the abbreviated names of the laboratory diagnostic tests are the World Health Organization (WHO) codes and the Logical Observation Identifier Names and Codes (LOINC) codes. The reference interval and the measurement unit for each measurement is separately indicated in the database, which allows the following semi-quantitative coding and interpretation:

1) **Non-measured (0)**: not suspicious or relevant, default assumption for the unmeasured value is normal.
2) **Measured-normal (1)**: suspicious and relevant, but measured test value is in the reference range.
3) **Abnormally-low (2)**: the measured test value is below the lower bound of the reference range.
4) **Abnormally-high (3)**: the measured test value is above the upper bound of the reference range.

In the current analysis, for each patient the measurements in a given, most recent period are merged and earlier measurements are neglected. We treat these merged tests vectorial representation as the *current state* of the patient.

The applied combinations for the window size and merge function are as follows. We split the data for 5 sub-data as last month (1m), last 3 months (3m), last half year (6m), last year (1y) and last 2 years (2y). In the continuous version of the merge, we aggregate all the sub-data calculating the maximum (max), minimum (min), average (avg) and median (med) values or keeping only the last measurements (last). In the discretized version of the merge, we first convert each value into a binary normal(0)/abnormal(1) value, then we aggregate all the sub-data calculating their AND (and) and OR (or) combinations or keeping only the last measurements (last, there is no difference in this case between the continuous and discretized version).

For each aggregation, we generated the following three types of matrices with semi-quantitative values, when the rows are the patients and the columns are the laboratory tests.

1) *Measurement indicator* matrix (I_M) has binary values, where 1 means that the laboratory test is performed for the patient in the given period, and 0 if not performed.
2) *Abnormality data* matrix (D) The value *2* means that lab-test is performed for patient and its merged value is *outside* the reference/normal range in the continuous case and 1 in the discretized case. The value *1* means the analogous case and empty means that the lab-test is *not performed* for the patient in the given period.
3) *Ternary data* matrix (T). Technically, it is a completed version of the *Abnormality data* matrix by setting its all empty items to 0. An intuitive interpretation is that a non-measured test indicate an *a priori* normal value, which is in certain cases suggest a smaller risk then a measured, i.e. suspicious, but normal value.

Fig. 1 shows the data preprocessing pipeline.

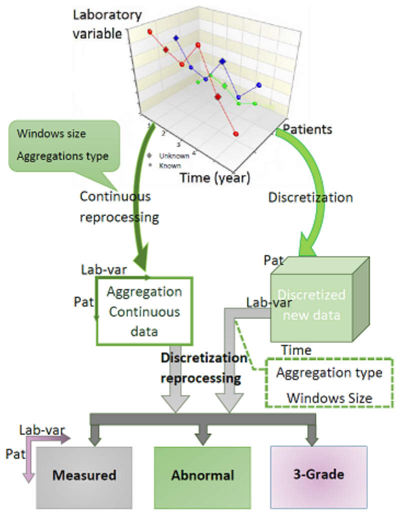


Fig. 1. The preprocessing pipeline resulting discrete data matrices.

We have developed Python scripts for these preprocessing steps and exploratory statistical data analysis.

IV. Methods

The central tasks according to our assumptions are the indication of confidently predictable tests among the requested ones and tests with confidently abnormal values among the unrequested ones. In our non-temporal approach using the described merging and discretization, we conceive these tasks as (1) the indication of confidently predictable tests among the known tests after merging and (2) as the indication of tests with confidently abnormal values among the unknown tests after merging. We introduce concepts for the probabilistic formalization of these questions in case of a new, actual patient with index $N+1$, assuming that the same preprocessing is applied in this case as for the data matrix D_N with N samples. Let K_{N+1} denote the set of the indices of known tests for patient $N+1$: $i \in K$ iff $I_{N+1,i} = 1$. The known set is divided to an evidence set $E \subset K$ and query set $Q = K \setminus E$. Using these index sets, $D_{N+1,K}$ denotes the subvector of known tests. We call the tests in Q as $l-\tau$ predictable iff $|E| = l$ and tests $D_{N+1,Q}$ are predictable with at most τ probability:

$$\forall i \in Q : \tau < \max_{j=1,2} p(D_{N+1,i} = j | D_{N+1,E}, D_N). \quad (1)$$

For a given τ, the minimal value with this property is denoted with $l_{N+1}(\tau)$, i.e. the size of the minimum number of tests from the known set sufficient the predict the rest of the known ones. Note that the set of potentially redundant tests could be defined more precisely using a multivariate approach. Using this univariate formalization, the number of τ-probably redundant tests is defined as

$$r_{N+1}(\tau) = |K_{N+1}| - l_{N+1}(\tau). \quad (2)$$

Analogously, the set $C_{N+1}(\tau)$ of probably abnormal variables with threshold τ is defined as follows

$$i \in C_{N+1} : \tau < p(D_{N+1,i} = \text{"}abnormal\text{"} | D_{N+1,K}, D_N).$$

We apply Bayesian network models M_i in the Bayesian model averaging framework to approximate the predictive distributions, i.e. for target Y

$$p(Y|D_{N+1,K}, D_N) \approx \sum_{M_i} p(Y|M_i, D_{N+1,K}) p(M_i|D_N).$$

For this purpose, currently we are extending and evaluating Markov Chain Monte Carlo (MCMC) methods over Bayesian network structures developed earlier in our group [1]. In the general batch case for M patients $D_{N+1:N+M}$, we treat the cases separately, because of computational limitations.

V. Results

The laboratory test measurements are highly incomplete, as they are specific to diseases and clinical conditions. Using only the last laboratory visit for each patient, Fig. 2 shows the proportion of cases with measurement, valid value and normal value for the laboratory tests (proportion of "Valid" is not shown as nearly all measurements have proper syntax, thus valid). The existence of a measurement means its presence in the dataset, its validity means that it has proper syntactic format and the reference interval (a.k.a. normal region) is available, finally, normality means that the measurement of a test has a valid decoding in the reference region.

In the current approach we merge the laboratory visits using varying window size. Table I represents proportions of

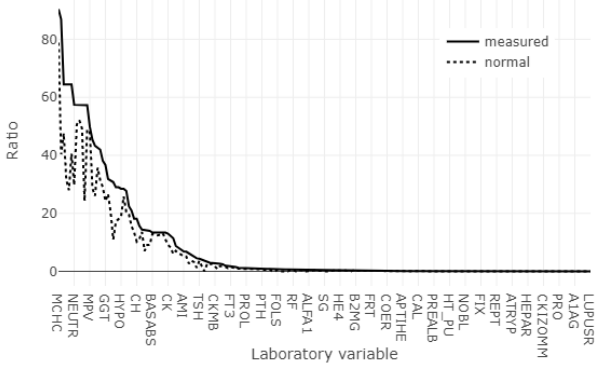


Fig. 2. Proportion of cases that were requested (”Measured”) and their value is in the reference range (”Normal”).

measured tests, valid values and normal values in the dataset using 1 month (1M), 3 month (3M) and 6 month (6M) window sizes for merging the results per patients.

TABLE I
PROPORTIONS OF MEASURED TESTS BY ACCUMULATING RESULTS.

	Measured	***Valid***	***Normal***
1M	10.46%	10.44%	8.03%
3M	10.66%	10.64%	8.39%
6M	21.75%	21.71%	17.13%

As results in Table I show the effect of merging laboratory test results in a 3 month period is negligible, which probably related to clinical protocols limiting the repetition of certain tests. These results indicate that the high level of incompleteness of the laboratory test data remains a major challenge, as the ratio of valid data is still around 20% after merging results in a 6 month period (homogeneity assumptions of the clinical state for longer periods usually cannot be expected). However, incompleteness is informative for laboratory tests, as indicated by the proposed semi-quantitative coding and interpretation, which property suggest the use of respective, complete datasets.

Currently, we are investigating the effect of discretization, approaches to cope with incomplete data and computational schemes to perform Bayesian inference using Monte Carlo methods jointly over the missing part of the dataset and predictive models.

VI. Conclusion and Future Work

The prediction of unknown tests could be used both in actual clinical decision support and in evaluation of health policies. From clinical point of view, the investigated methods aim to support the cost-effective use of laboratory capacities, as the set of the requested tests can be adaptively modified. Additionally, these functionalities can also support quality control implementing professional protocols, but it could also help the design and refinement of diagnostic protocols.

References

[1] Péter Antal, András Millinghoffer, Gábor Hullám, Csaba Szalai, and András Falus. A bayesian view of challenges in feature selection: feature aggregation, multiple targets, redundancy and interaction. In *New Challenges for Feature Selection in Data Mining and Knowledge Discovery*, pages 74–89, 2008.

[2] David Blumenthal and Marilyn Tavenner. The “meaningful use” regulation for electronic health records. *N Engl J Med*, 2010(363):501–504, 2010.

[3] Urszula Chajewska and Joseph Y Halpern. Defining explanation in probabilistic systems. In *Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence*, pages 62–71. Morgan Kaufmann Publishers Inc., 1997.

[4] Clifford Champion and Charles Elkan. Visualizing the consequences of evidence in bayesian networks. *arXiv preprint arXiv:1707.00791*, 2017.

[5] Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*, volume 2. CRC press Boca Raton, FL, 2014.

[6] David Heckerman, Eric Horvitz, and Blackford Middleton. An approximate nonmyopic computation for value of information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(3):292–298, 1993.

[7] Finn V Jensen and Thomas Dyhre Nielsen. Probabilistic decision graphs for optimization under uncertainty. *Annals of Operations Research*, 204(1):223–248, 2013.

[8] Peter B Jensen, Lars J Jensen, and Søren Brunak. Mining electronic health records: towards better research applications and clinical care. *Nature Reviews Genetics*, 13(6):395–405, 2012.

[9] Andreas Krause and Carlos E Guestrin. Near-optimal nonmyopic value of information in graphical models. *arXiv preprint arXiv:1207.1394*, 2012.

[10] Carmen Lacave and Francisco J Díez. A review of explanation methods for bayesian networks. *The Knowledge Engineering Review*, 17(2):107–127, 2002.

[11] Wenhui Liao and Qiang Ji. Efficient non-myopic value-of-information computation for influence diagrams. *International Journal of Approximate Reasoning*, 49(2):436–450, 2008.

[12] Giuseppe Lippi, Antonella Bassi, and Chiara Bovo. The future of laboratory medicine in the era of precision medicine. *Journal of Laboratory and Precision Medicine*, 1(3), 2016.

[13] Giuseppe Lippi, Chiara Bovo, and Marcello Ciaccio. Inappropriateness in laboratory medicine: an elephant in the room? *Annals of translational medicine*, 5(4), 2017.

[14] Giuseppe Lippi and Camilla Mattiuzzi. The biomarker paradigm: between diagnostic efficiency and clinical efficacy. *Pol Arch Med Wewn*, 125(04):282–288, 2015.

[15] Giuseppe Lippi and Mario Plebani. False myths and legends in laboratory diagnostics. *Clinical chemistry and laboratory medicine*, 51(11):2087–2097, 2013.

[16] Giuseppe Lippi and Mario Plebani. Laboratory economics. risk or opportunity? *Clinical Chemistry and Laboratory Medicine (CCLM)*, 54(11):1701–1703, 2016.

[17] Martina Montagnana and Giuseppe Lippi. The risks of defensive (emergency) medicine. the laboratory perspective. *Emergency Care Journal*, 1(1), 2016.

[18] Changwon Yoo and Gregory F Cooper. An evaluation of a system that recommends microarray experiments to perform to discover gene-regulation pathways. *Artificial Intelligence in Medicine*, 31(2):169–182, 2004.

[19] Zhongheng Zhang. The role of big-data in clinical studies in laboratory medicine. *Journal of Laboratory and Precision Medicine*, 2(6), 2017.

[20] XueZhong Zhou, Jörg Menche, Albert-László Barabási, and Amitabh Sharma. Human symptoms–disease network. *Nature communications*, 5:4212, 2014.

A Preliminary Analysis on the Effect of Randomness in a CEGAR Framework

Ákos Hajdu[1,2], Zoltán Micskei[1]
[1]Budapest University of Technology and Economics, Department of Measurement and Information Systems
[2]MTA-BME Lendület Cyber-Physical Systems Research Group
Email: {hajdua, micskeiz}@mit.bme.hu

***Abstract*—Formal verification techniques can check the correctness of systems in a mathematically precise way. Counterexample-Guided Abstraction Refinement (CEGAR) is an automatic algorithm that reduces the complexity of systems by constructing and refining abstractions. CEGAR is a generic approach, having many variants and strategies developed over the years. However, as the variants become more and more advanced, one may not be sure whether the performance of a strategy can be attributed to the strategy itself or to other, unintentional factors. In this paper we perform an experiment by evaluating the performance of different strategies while randomizing certain external factors such as the search strategy and variable naming. We show that randomization introduces a great variation in the output metrics, and that in several cases this might even influence whether the algorithm successfully terminates.**

I. Introduction

Formal verification techniques (such as model checking [1]) can check whether a model (formal representation) of a system meets certain requirements by exhaustively analyzing its possible states and transitions. As our reliance on computer systems grows, the importance of these techniques is also increasing. However, a typical drawback of using formal methods is their high computational complexity. The Counterexample-Guided Abstraction Refinement (CEGAR) approach [2] alleviates this problem by automatically constructing and refining abstractions that over-approximate the behavior of systems. CEGAR starts with a coarse initial abstraction (to minimize complexity) and then applies refinements based on candidate counterexamples until a sufficient precision is reached (that is fine enough for deciding whether the requirement holds).

Theta is a generic framework that includes many configurations (variants) of the CEGAR algorithm in a common environment [3]. The framework relies on first order logic (FOL): the behavior of the models is encoded in graphs annotated with FOL formulas and the algorithms use SAT/SMT solvers [4] as the underlying engine. We already performed experimental evaluations in Theta and in most cases we concluded that the configurations have a diverse performance: different configurations are more suitable for different tasks [5], [6]. However, as the underlying strategies of the configurations are also becoming more advanced, we cannot be certain whether their performance can be attributed to their intentional, algorithmic behavior. Rather, they might be unintentionally influenced by certain external factors in a way that the final outcome is a better performance in certain cases. For example, a different ordering of commutative formulas might unintentionally affect the order in which states are processed.

In this paper we investigate two such factors. A higher level, algorithmic factor is the search strategy in the abstract state space. A configuration may fail to build a suitable abstraction efficiently if a deterministic search strategy guides it in the “wrong” direction. A lower level, external factor is the naming of the variables. Most of the refinement strategies employ an SMT solver for computing over-approximations. We observed that the name of the variables affects their order in certain collections (e.g., sets), which may influence the inner heuristics of the solvers, also affecting the quality of the generated abstractions.

Our experiment shows that randomizing any of the aforementioned factors greatly increases variations in the output metrics (e.g., execution time). Furthermore, randomization often even affects whether the algorithm can successfully terminate within the given time limit. We also examine some cases where a randomized configuration can verify a model for which the deterministic ones fail. Based on this feedback, we can improve the shortcomings of the deterministic configurations and we can also introduce nondeterministic options to the configurations as a viable alternative.

II. Experiment Planning

In our experiment several *configurations* of the CEGAR algorithm of Theta were executed on various input *models* deterministically and also with randomizing the search strategy or the variable names.

A. Research Questions

The current research questions focus on a preliminary, exploratory analysis of the results.

RQ1 Are there any cases where a randomized configuration could verify a model (at least once) that its deterministic counterpart could not?

RQ2 How does randomization affect the variation of output metrics (e.g., execution time) compared to deterministic configurations? Which yields a greater variation? Randomizing search or variable names?

B. Subjects and Objects

Theta includes many parameters for the CEGAR algorithm. For this experiment we selected the two most prominent,

TABLE I
VARIABLES OF THE EXPERIMENT.

Category	Name	Type	Description
Input (model)	Category	Factor	Category of the model. Possible values: eca, hw, locks, plc, ssh (see Section II-B).
	Model	String	Unique name of the model.
Input (config.)	Domain	Factor	Domain of the abstraction. Possible values: PRED (predicate), EXPL (explicit value).
	Refinement	Factor	Refinement strategy. Possible values: BIN (binary interpolation), SEQ (sequence interpolation).
	Randomized	Factor	Factor that is randomized. Possible values: DET (deterministic, no randomization), SEARCH (random search strategy), VARS (random variable names).
Output (metrics)	Succ	Boolean	Indicates whether the algorithm successfully provided a result within the given time limit.
	TimeMs	Integer	Execution time of the algorithm (in milliseconds).
	Iterations	Integer	Number of refinement iterations until the sufficiently precise abstraction was reached.
	ArgSize	Integer	Number of nodes in the Abstract Reachability Graph (ARG), i.e., the number of explored abstract states.
	ArgDepth	Integer	Depth of the ARG.
	CexLen	Integer	Length of the counterexample, i.e., a path leading to a state of the model that does not meet the requirement.

namely the domain of the abstraction and the refinement strategy. We experimented with predicate [7] and explicit value [8] domains with binary [9] and sequence [10] interpolation-based refinements, as these strategies are also implemented in many other verification tools [11]. The third parameter is the randomized factor, i.e., the search strategy, the variable names, or nothing (deterministic). Therefore, there are a total number of $2 \cdot 2 \cdot 3 = 12$ configurations.

Due to the long execution time of the measurements, we only evaluated the configurations on 30 input models. Nevertheless, we tried to make these models relevant and diverse. Therefore, we picked 10 hardware models (hw) from different categories of the Hardware Model Checking Competition [12], 15 models from 3 categories (eca, locks, ssh) of the Competition on Software Verification [11] and 5 industrial PLC software modules (plc) from CERN [13]. Based on previous measurements, we picked models with different difficulties, including easy (verified by most configurations) and difficult instances (verified by a few or no configurations).

C. Variables

Variables of the experiment are listed in Table I, grouped into three main categories: properties of the model (input), parameters the configuration (input) and metrics of the algorithm execution (output). If the algorithm did not provide a result within the time limit, the variable Succ is false and the other output metrics are empty (NA).

D. Measurement Procedure

Measurements were executed on two 64 bit Windows 7 virtual machines with 2 cores (2.50 GHz), 8 GB RAM and JRE 8 (THETA is implemented in Java). Z3 version 4.5.0 [14] was used as an SMT solver. Each measurement was repeated 30 times with a different random seed. The time limit for each execution was 180 seconds.

E. Analysis Methods

RQ1 can be answered by summarizing heatmaps and filtering the data. Furthermore, it would be interesting to analyze each case separately in more detail using for example the logs produced by THETA. RQ2 can be examined with basic descriptive statistics and summarizing plots (e.g., box plots), yielding a good overview on the variations of the output metrics under different configurations.

F. Threats to Validity

We worked with input models from different sources, including well-known benchmarks sets such as HWMCC [12] and SV-COMP [11]. However, due to time constraints, only 30 models were picked. External validity could be improved by selecting more models both from the same sources and from additional ones. This experiment focused only on THETA, which includes many algorithms known from state-of-the-art tools [11]. However, external validity would benefit from repeating the experiment with different tools. It would also be interesting to experiment with a higher time limit (e.g., SV-COMP uses 900s) and more randomized factors (e.g., reorder commutative formulas in the models). Internal validity is increased by repeating the measurements 30 times on dedicated virtual machines. However, if more resources were available, measurements should be repeated even more times (e.g. 1000 times [15]) on dedicated physical machines.

III. ANALYSIS

This section discusses the analyses and results related to our research questions. The analyses were performed with the R software environment [16]. The raw data, the R script and a detailed report can be found on a supplementary web page.[1]

A. Terminology and Overview

A *run* is a single execution of a *configuration* on a *model*. A run is *successful* if a result (whether the model is correct or not) is provided within the given time limit. A *measurement* is the collection of all repeated runs of the same configuration on the same model. A measurement is *successful* if it includes at least one successful run. In this case we also say that the configuration *verified* the model.

[1]http://dx.doi.org/10.5281/zenodo.1117853

In this experiment 12 configurations were executed on 30 models (from 5 categories), yielding $12 \cdot 30 = 360$ measurements. Each measurement was repeated 30 times, giving $360 \cdot 30 = 10800$ runs. There are $7080/10800$ successful runs (66%) and $261/360$ successful measurements (72%). Fig. 1 summarizes the range and distribution of the output metrics.

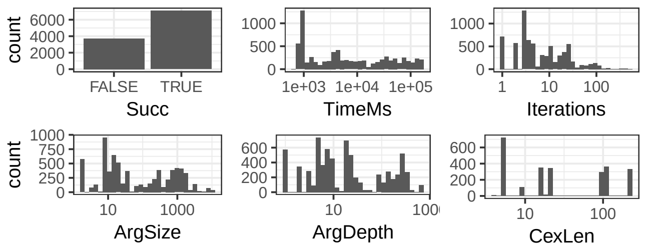


Fig. 1. Distribution of the output metrics.

Fig. 2 shows the number of successful runs for each model. It can be seen that besides the easy models in category locks, their difficulty is gradually increasing, supporting our claim on diversity.

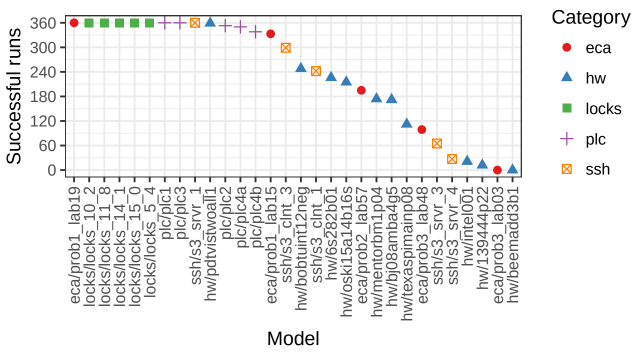


Fig. 2. Number of configurations that verified the models.

Fig. 3 presents the number of verified models and successful runs for each configuration. Configurations are abbreviated with the first letters of their parameters, e.g., PB-V stands for predicate domain, binary interpolation and variable name randomization. The difference between the best and worst configuration is $25 - 19 = 6$ regarding verified models and $685 - 518 = 167$ regarding successful runs.

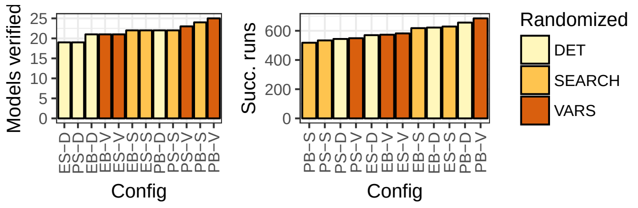


Fig. 3. Number verified models and successful runs for each configuration.

B. RQ1: Successful Verifications

Fig. 4 illustrates the number of successful runs for each measurement. White cells represent no successful runs. It can be seen that in most cases the deterministic configurations have either 0 or 30 successful runs. There are a few exceptions though, where the execution time was close to the time limit.

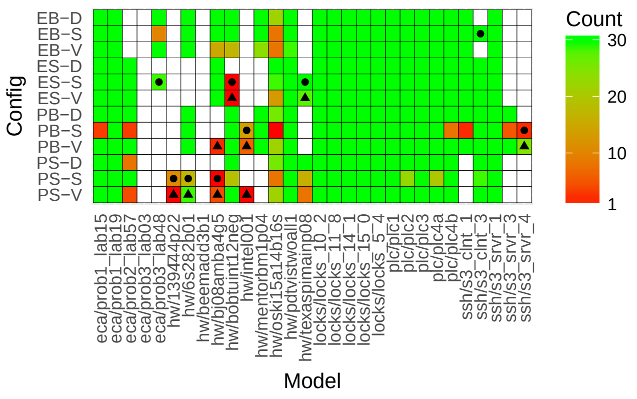


Fig. 4. Number of successful runs for each measurement.

However, there are $9 + 9$ cases where a configuration with randomized search or variable names could verify a model that its deterministic counterpart could not. These cases are marked with dots and triangles respectively in Fig. 4.

Interestingly, there are 3 models (139444p22, intel001, s3_srvr_4) where only randomized configurations were successful. It is a hard task to investigate the behavior of the algorithm in detail for these cases, as these models are very complex (sometimes consisting of thousands of variables and formulas with hundred thousands of terms and operands). Nevertheless, we examined the logs to get some insight on what is happening with and without randomization.

139444p22 is a large model, consisting of 8600 variables and formulas with a total size of 1.6×10^5 (measured in the number of terms and operands). The deterministic configuration runs out of time when checking a candidate counterexample using the SMT solver. The randomized configuration also spends roughly half of its time on checking counterexamples, but in the successful runs, it quickly finds a feasible one, terminating the algorithm. A possible explanation is that the solver can find an easy solution for feasible counterexamples, but fails to prove infeasibility due to the large formulas.

The intel001 model is not large, but the formulas refuting the feasibility of counterexamples can grow unmanageably large. In the successful runs of the randomized configuration, it manages to produce refutation formulas with a maximal size of 4.4×10^4. The deterministic configuration however, generates a refutation formula of size 4×10^6 in the 6th iteration, prohibiting the exploration of the abstract states.

Repeating the measurements for the s3_srvr_4 model (to get logs) revealed that the deterministic configuration can also verify this model, but its execution time is slightly above the limit of 180s. By examining the randomized runs as well, we observed that for this model the success of verification depends on the number of refutation formulas discovered. The deterministic configuration discovers some unnecessary formulas, making the number of abstract states higher. However, the randomized configurations can find a subset of these formulas (in some runs) that is still enough to prove the correctness of the model in less time.

Feedback learned from these cases identified various shortcomings of deterministic configurations and also gave us ideas on how to improve them.

C. RQ2: Variations

For each measurement, the 30 repeated runs form a distribution for each output metric. Variation is usually described by the standard deviation (SD). However, the output metrics have a vastly different range, making SD incomparable between them. Therefore, we calculate the *relative standard deviation* (RSD = SD / mean). Furthermore, for the Boolean variable Succ, we replace false by 0, true by 1 and calculate the SD.

The distribution of the deviations are summarized using box plots in Fig. 5, grouped by the factor that is randomized. There are 5 outlier points between 1.25 and 3.5 that were cropped so that the box plots can be depicted using the same scale.

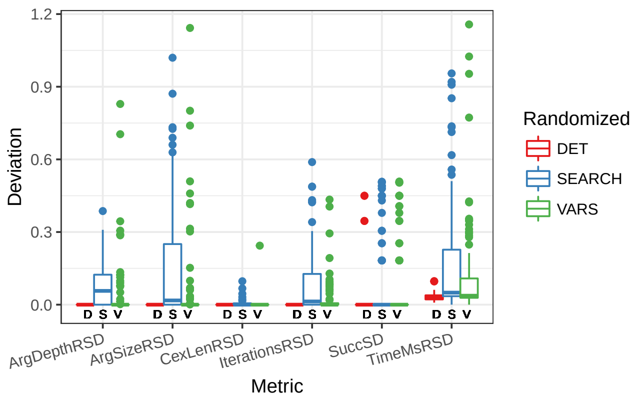


Fig. 5. Distribution of the deviations of the output metrics.

Deterministic configurations also have some small deviations for the execution time and the success indicator. As it was mentioned previously, the latter can be attributed to execution times near the time limit. All other output metrics have 0 deviation, increasing our confidence that these configurations are indeed deterministic.

It can be seen that the randomized configurations have greater deviation for all output metrics. The largest deviations appear for the execution time (TimeMs) and the number of abstract states explored (ArgSize). The length of the counterexample (CexLen) has the lowest deviations. This metric has a smaller sample size, as only 9 out of the 28 verified models were incorrect. Furthermore, counterexamples often correspond to a single concrete execution in the original model, which has a fixed length. It can also be clearly observed that in most cases randomizing the search strategy yields greater deviations than randomizing the variable names.

IV. Conclusions

In our paper we evaluated various configurations of the CEGAR algorithm in the Theta tool under randomized search strategies and variable names. Our experiment highlighted that randomizing these factors introduces a great variation in the output metrics. In several cases this also influences whether a configuration can successfully verify a model. We also examined some cases where a randomized configuration verified a model that none of the deterministic ones could. Feedback from these cases will help us to improve the current shortcomings of the algorithms. Thus, preliminary results are interesting, but to improve their external validity, a more thorough experiment is needed with more models, repetitions and randomized factors as well.

References

[1] E. Clarke, O. Grumberg, and D. Peled, *Model checking*. MIT Press, 1999.

[2] E. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith, "Counterexample-guided abstraction refinement for symbolic model checking," *Journal of the ACM*, vol. 50, no. 5, pp. 752–794, 2003.

[3] T. Tóth, A. Hajdu, A. Vörös, Z. Micskei, and I. Majzik, "Theta: a framework for abstraction refinement-based model checking," in *Proc. 17th Conf. on Formal Methods in Computer-Aided Design*. FMCAD inc., 2017, pp. 176–179.

[4] A. Biere, M. Heule, and H. van Maaren, *Handbook of Satisfiability*. IOS press, 2009.

[5] A. Hajdu and Z. Micskei, "Exploratory analysis of the performance of a configurable CEGAR framework," in *Proc. 24th PhD Mini-Symposium*. BME DMIS, 2017, pp. 34–37.

[6] G. Sallai, A. Hajdu, T. Tóth, and Z. Micskei, "Towards evaluating size reduction techniques for software model checking," in *Proc. 5th Int. Workshop on Verification and Program Transformation*, ser. EPTCS. Open Publishing Association, 2017, vol. 253, pp. 75–91.

[7] S. Graf and H. Saidi, "Construction of abstract state graphs with PVS," in *Computer Aided Verification*, ser. LNCS. Springer, 1997, vol. 1254, pp. 72–83.

[8] D. Beyer and S. Löwe, "Explicit-state software model checking based on CEGAR and interpolation," in *Fundamental Approaches to Software Engineering*, ser. LNCS. Springer, 2013, vol. 7793, pp. 146–162.

[9] K. McMillan, "Applications of Craig interpolants in model checking," in *Tools and Algorithms for the Construction and Analysis of Systems*, ser. LNCS. Springer, 2005, vol. 3440, pp. 1–12.

[10] Y. Vizel and O. Grumberg, "Interpolation-sequence based model checking," in *Formal Methods in Computer-Aided Design*. IEEE, 2009, pp. 1–8.

[11] D. Beyer, "Software verification with validation of results," in *Tools and Algorithms for the Construction and Analysis of Systems*, ser. LNCS. Springer, 2017, vol. 10206, pp. 331–349.

[12] G. Cabodi, C. Loiacono, M. Palena, P. Pasini, D. Patti, S. Quer, D. Vendraminetto, A. Biere, K. Heljanko, and J. Baumgartner, "Hardware model checking competition 2014: An analysis and comparison of solvers and benchmarks," *Journal on Satisfiability, Boolean Modeling and Computation*, vol. 9, pp. 135–172, 2016.

[13] B. Fernández Adiego, D. Darvas, E. Blanco Viñuela, J.-C. Tournier, S. Bliudze, J. O. Blech, and V. M. González Suárez, "Applying model checking to industrial-sized PLC programs," *IEEE Trans. on Industrial Informatics*, vol. 11, no. 6, pp. 1400–1410, 2015.

[14] L. de Moura and N. Bjørner, "Z3: An efficient SMT solver," in *Tools and Algorithms for the Construction and Analysis of Systems*, ser. LNCS. Springer, 2008, vol. 4963, pp. 337–340.

[15] A. Arcuri and L. Briand, "A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering," *Software Testing, Verification and Reliability*, vol. 24, no. 3, pp. 219–250, 2014.

[16] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2017. [Online]. Available: https://www.R-project.org/

Towards Supporting Dynamic Symbolic Execution via Multi-Domain Metrics

Dávid Honfi, Zoltán Micskei
Department of Measurement and Information Systems,
Budapest University of Technology and Economics,
Budapest, Hungary
Email: {honfi,micskeiz}@mit.bme.hu

***Abstract*—A popular code-based test generation technique is dynamic symbolic execution (DSE), which combines concrete executions with symbolic ones. The current maturity of DSE led to industrial usages. However, the complexity of software used in industrial practice poses several challenges for DSE. The issues caused by these are often hard to identify as they are mostly indicated by only the lack of coverage. In this paper, we gather and present metrics that can be used on top of our already elaborated approach that visualizes symbolic executions trees.**

I. Introduction

Today, testing is an inevitable task of software development. Academic research also tackles the topic from various aspects. As test development is a time-consuming task, it is beneficial to introduce automation. Several approaches have been proposed addressing this challenge, including ones automatically generating tests from source code (white-box test generation). One of these techniques is *symbolic execution* (SE).

Symbolic execution is a widely used technique originating from the '80s. It begins the execution on a given entry point in the program. SE replaces the concrete variables with symbolic ones and uses them to form path constraints on each known execution path (path condition). Then, these constraints are transformed to SMT problems, which can be solved using special-purpose tools. The concrete values satisfying the path constraints steer the program execution exactly to the path corresponding to the given constraint.

Dynamic symbolic execution (DSE) enhances the original technique by mixing it with concrete executions. DSE starts a concrete execution from an arbitrary entry point with the simplest concrete input values as possible, while symbolic execution is performed in parallel. During the concrete execution, SE collects the constraints on the given path. When a path in an execution has finished, the DSE transforms (e.g., negates) the collected constraint system and then attempts to obtain a solution by defining it as a Satisfiability Modulo Theories (SMT) problem. If a solution is available, then a new concrete execution is feasible with the new inputs. This process continues until either no more feasible paths are available or an execution boundary (e.g., time limit) is reached.

Both SE and DSE face several challenges in numerous cases. This includes, for instance, the following [1].

- *Constraint solver issues (CSI):* There are typical issues for constraint solvers such as formulas containing floating point arithmetics due to the high precision representation, or large path constraints with diverse types of variables.
- *State space explosion (SSE):* When dynamic symbolic execution is unbounded, the algorithm may explore program states that are out-of-scope or cause fruitless executions (e.g., redundant paths). However, when a boundary is set, it should be defined in a way that it does not hinder exploring important states in the program.
- *Object creation (OC):* A common issue in complex programs is that the objects passed as parameters must be assembled through sequences of method calls or using special factory methods. These are usually hard to guess automatically for a symbolic execution engine, which prevents the program exploration. In these cases, manual intervention or specialized algorithms shall be used.
- *Environment interactions (EI):* Interactions with the environment of the unit under analysis are commonly found in complex software. These interactions are mostly calls to databases, to network or to the file system. Furthermore, there could be invocations to underlying frameworks. Omitting the handling of these calls may lead to undesired behaviors (e.g., writing to file system or database).

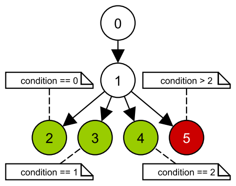
These challenges often cause issues that result in not enough tests being generated. However, identifying and localizing the root causes of the occurring issues may require an excessive amount of effort. The effort spent on the identification may reduce the advantages gained from automation.

To alleviate the identification and localization process, we have already presented an approach that visualizes the symbolic execution [2]. This technique uses an internal representation of the execution called symbolic execution tree and visualizes it with a predefined semantics. The use case of the visualization is twofold: it gives an overall overview of the execution, and gives internal details at each symbolic state (e.g., path condition). In our visualization, we enriched the symbolic execution tree nodes with various metadata: sequence number, location, corresponding runs, path condition, constraint solver calls, generated tests (for leaf nodes). An example code and the corresponding symbolic execution tree is shown in Figure 1.

The attached metadata serves as the basis for identifying

```
public int SwitchBranching(int condition) {
    var divisor = 0;
    switch(condition) {
        case 0: return 0;
        case 1: return −1;
        case 2: return −2;
        default: return (condition / divider);
    };
}
```

(a) The source of SwitchBranching.



(b) The symbolic execution tree.

Fig. 1. A simple method and the corresponding symbolic execution tree extracted from SEViz. The green leaves indicate passing generated tests, while the red leaf shows a test generated for the path, which raises an exception.

the issues of the test generation process. However, in a wider focus area, there is a large number of other metrics that can be attached to a symbolic execution tree. These metrics can be obtained from related domains (e.g., source codes, graphs). With the additional metrics attached, a symbolic execution tree may be more capable of indicating root causes of challenging issues. Moreover, the data obtained for the metrics can be used for recommendations and predictions.

The goal of this paper is to gather from literature, present and describe valuable metrics attached to symbolic execution trees. The metrics we set out to present are obtained from various sources and domains to improve diversity. Furthermore, another goal in the paper is to present the applicability of the metrics on an artificial example.

The rest of the paper is organized as follows. In Section II we present the collected metrics in detail including their original domains. Section III shows examples on how to support test generation and identify issues using the gathered metrics. Section IV discusses the related work, while Section V concludes our contributions.

II. Metric selection

Source. To gain an overview of what metrics can be attached to a symbolic execution tree, we defined four categories from related domains of dynamic symbolic execution-based test generation. For each of the categories, we searched for survey papers that collect the most important and widely-used metrics in their domains. The domains we selected are source code, dynamic symbolic execution, tests, and graphs.

Context. In this paper, we select and detail four most relevant metrics for each category. Also, we provide indications on which metric could be influenced by the issues stated before and vice versa (denoted with the abbreviation of the issue). Albeit the selected metrics could be used as standalone indicators for issue prediction, our paper only focuses on using them for extending symbolic execution trees to perform post-analysis. We defined three *locations*, where a metric can be attached to a symbolic execution tree. We indicate these next to the name of the metric.

- Nodes (N): A node in a symbolic execution tree represents a program state. Each node is mapped to a given location of the program (as a basic block).
- Paths (P): A path is a sequence of nodes in the SE tree that represents a corresponding execution.
- Exploration (E): The exploration refers to the set of all the paths that have been executed. Basically, this is the whole symbolic execution tree.

Selection. We defined the selection criterion of the metrics based on two dimensions: 1) challenges occurring in dynamic symbolic execution and 2) locations where a metric can be placed in a symbolic execution tree. Each of the examined metrics was labeled with values from these dimensions. The defined coverage criterion requires to cover all the combinations of the *locations-challenges* dimensions with at least one metric. The labeling was based on our experiences and intuitions regarding with symbolic execution trees [2].

A. Static code-based metrics (SC)

The domain of source code can be viewed from various aspects, e.g., abstract representations like abstract syntax trees of control-flow graphs. We obtained the source code metrics from two papers. One of them is a study conducted to extract characteristics of 147 open-source Java projects [3], while the other compares programmer opinions to complexity [4].

1) Lines of Code [E][SSE]: The LoC is a textual metric of the source code measuring the number of lines. A very large program with many modules interacting with each other could lead to huge path constraints causing constraint solvers to fail.

2) Cyclomatic complexity [E][CSI, SSE]: CC is a metric that indicates the number of linearly independent paths in the source code. The number of these paths are derived from the control-flow graph (CFG) of the program. Large CC could indicate the presence of loops that would yield large path conditions and search space. Both hinder constraint solvers.

3) Halstead's difficulty [E][CSI, SSE]: The metric is sometimes called the error-proneness, which is regarding with the number of unique operators and operands in the code. Both the operators and operands are usually expanding the search space that the DSE interpreter must explore.

4) Number of method calls [N, P, E][SSE, EI]: This metric is an indicator how many calls are performed to any other methods (e.g., to other classes, libraries, environment). The larger the number of external method calls, the higher the probability of environment accessing issues for DSE. Obviously, if there are other, additional methods to explore, the search space also grows.

B. Dynamic symbolic execution metrics (SE)

In dynamic symbolic execution, one of the key concepts is the path condition that is solved by constraint solvers on each path [3]. Also, an important feature of the explored paths is the variety of instructions found along each path [5].

1) Path condition length [N,P][CSI, OC]: Represents the length of the constraint system collected on an execution path (sum of variables and operators used). Note that this metric does not care about the repeated use of the same variable, constant or operator.

2) Number of variables in path condition [N,P][CSI]: The metric measures the number of distinct variables in the path condition. This represents how dependent is the outcome of the execution path on the symbolic variables.

3) Number of constants in path condition [N,P][CSI, OC]: The metric measures the number of distinct constants in the path condition including all data types that support constants. This denotes how restrictive is the program code in the given path on the variables.

4) Path description vector [P][SSE, EI]: An executed path can be represented as a sequence of interpreted basic blocks. Each basic block contains a given number of low-level instructions, which can be represented using an occurrence vector with a fixed length (based on the number of possible instruction types on the given platform). Then, for each path, a feature matrix can be assembled using the occurrence vectors obtained along the path. This matrix represents the covered program features on the given path [5].

C. Generated test metrics (GT)

At each leaf node of the symbolic execution tree, a test case can be derived that executes the corresponding path, which ends in that node. When a test case is produced, the characteristics of the test (regarding with the inputs, actions and expectations) may give overview of the given path about what is performed along [6], [7].

1) Number of assertions [P][SSE, OC, EI]: This metric tells how many assertions are found in the given generated tests. If there are too many, then the test may be too specific, which could cause false positive outcomes and could lead to Assertion Roulette [8]. On the contrary, if there are few assertions, then it could mean that the test case is too permissive and may omit to check important behaviors. This issue is usually caused by a problem in the DSE engine about what observed behaviors to check.

2) Number of different types of assertions [P][SSE, OC, EI]: It measures the variety of assertions in a generated test. It may indicate that the single test case is checking multiple behaviors. However, a high number for this metric may indicate that the test is too specific. This is usually caused by an issue on observing the behavior of the program.

3) Lines of test code [P][OC]: The length of the test code is usually a good indicator of its complexity. Too long tests may hinder easy understanding, or it could contain unwanted setups or assertions. Long tests may also indicate that the dynamic symbolic execution engine was only able to setup the test environment in an unusual or unnecessary way.

4) Number of constants in test code [P][CSI, SSE]: The metric measures the number of constant values that the dynamic symbolic execution engine was able to generate into the code. If there is a value, it means that the algorithm was able to discover and parse it from a given basic code block. A large number of constants in the code usually yields wrongly handled unbounded loops in the program.

D. Generic graph metrics (GG)

It is typical to apply general metrics for a wide variety of graphs across several domains. This is the case for software engineering as well [9]. The most common abstract representation of a program is its control-flow graph that is by definition contains the possible execution paths between the basic code blocks in the program. However, to our best knowledge, there is no study, which maps these general graph metrics to symbolic execution trees.

1) Average branching factor [E][CSI, SSE, OC]: Let graph G be the symbolic execution tree with a set of vertices $V(G)$. We define the branching factor $d_+(v)$ for each node $v \in V(G)$ as its number of outgoing edges. The overall metric for the whole tree is an average calculated as $\frac{1}{|V(G)|} * \sum_{v \in V(G)} d_+(v)$. If a symbolic execution tree has a low branching factor, it could reveal that the constraint solver faced issues during the solution of complicated path conditions. High branching factor could indicate the presence of unbounded loops.

2) Height of tree [E][SSE]: The height of a tree is given by the length of the longest path selected out of all possible paths from the root to a leaf. A suspiciously deep symbolic execution tree may indicate that the dynamic symbolic execution engine was not able to appropriately handle bounds of execution.

3) Number of leaves [E][SSE]: The number of leaves in a tree provides a way to determine its width. As a leaf node represents the end of an execution path, we use this metric to decide how many tests could have been generated. The metric should be used in strong cooperation with others (e.g., test outcomes, branching factor).

4) Diameter of the tree [E][SSE]: To determine the diameter of a symbolic execution tree, we temporarily remove the directions of edges and calculate the longest path available among all of the node pairs. From the longest paths between all of the node pairs, we select longest one to indicate the diameter of the whole tree. If the diameter of a symbolic execution tree is unusually large, it may yield that search space is too huge for the engine and must be handled properly.

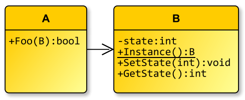
In Table I we summarize the coverage of the metrics on both of the defined dimensions. It can be seen that although the coverage criterion has been fulfilled, yet there are some fields, which have a smaller amount of associated metrics (e.g., object creation issue identification on node level).

TABLE I
SUMMARIZING TABLE OF METRIC COVERAGE OF THE TWO DIMENSIONS DEFINED (LOCATION-CHALLENGE).

	CSI	**SSE**	**OC**	**EI**
N	SE-1, SE-2, SE-3	SC-4	SE-1, SE-3	SC-4
P	SE-1, SE-2, SE-3, GT-4	SC-4, SE-4, GT-1, GT-2, GT-4	SE-1, SE-3, GT-1, GT-2, GT-3	SC-4, SE-4, GT-1, GT-2
E	SC-2, SC-3, GG-1	SC-1, SC-2, SC-3, SC-4, GG-1, GG-2, GG-3, GG-4	GG-1	SC-4

III. AN EXAMPLE USE OF THE METRICS

We present how the metrics could indicate issues through an example. In this simple program, the identification of an object



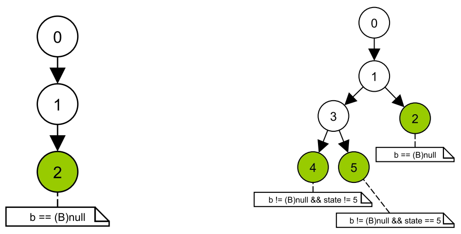
```
public bool Foo(B b) {
    if(b == null) return false;
    if(b.GetState() == 5) return true;
    return false;
}
```

(a) Class diagram of the example. (b) Example method for analysis.

Fig. 2. The example program architecture and code with classes A and B.

creation issue (OC) will be demonstrated via the attached metrics. Consider the example class layout shown in Figure 2a. Class `A` contains the current method under test `Foo` that has an argument of type `B`. Class `B` has a private default constructor and a method (`Instance`) that obtains an instance of this class. We executed dynamic symbolic execution on method `Foo` using Microsoft Pex, a state-of-the-art DSE-based test generator [10]. The outcome visualized in SEViz is shown in Figure 3a. We identified the number of constants in the path condition (II-B3), the lines of test code (II-C3) and the branching factor (II-D1) as a unique indicator set of the OC.

Figure 3b shows the symbolic execution tree after the OC issue has been resolved using a manual factory method. The branching factor of the tree previously was 1, while it increased to 1.66 with using the factory. Furthermore, the generated test code length was 5 LoC in the first case, which increased to 6.33 in the second case. The number of constants in the path condition also confirms the issue: in the first case, there is only one constant used in the only path (`null`), while in the extended case, there are other constants as well (e.g., `state == 5`). Only null constants in the path conditions throughout the tree usually indicate that there is a problem with creating objects for the given variable. In this example, this hypothesis was supported by two facts: the low branching factor of the tree and test cases with short length. The identification of the root cause is fairly simple using the path condition variables, therefore can be automated: one shall examine, which variables were only constrained to null. T



(a) SE tree with OC issue present. (b) SE tree after resolved OC issue.

Fig. 3. SE trees for the example with the OC issue present and resolved.

IV. Related Work

A related technique is Covana [11] that aims to identify OC and EI issues. Covana monitors DSE and collects problem candidates that are examined using data dependence analysis. While this approach is based on runtime monitoring, our approach uses only metrics attached to previously produced symbolic execution trees. SED is a symbolic execution debugger that visualizes symbolic execution trees with metadata [12]. The main difference between their approach is that they use it for debugging, while our approach aims at identifying issues of DSE-based test generation. Baldoni *et al.* survey symbolic execution techniques along with identifying their challenges [1]. They also consider possible solutions to these problems. Our technique may be extended with advising solutions to identified issues. Eler *et al.* analyzed characteristics of Java programs influencing the performance of symbolic execution [3]. We used some of their defined metrics to attach them to the nodes of the generated symbolic execution trees (representing a program state and location).

V. Conclusions and Future Work

In this paper, we gathered and presented metrics from various domains to alleviate the issues of dynamic symbolic execution based on previously extracted symbolic execution trees. We selected 16 metrics from papers of 4 related domains based on a predefined coverage criterion to enhance the problem identification process. The metrics have been presented in detail along with two metadata for each of the metrics indicating that 1) for which issue they can be used and 2) where they can be attached in the symbolic execution tree. We also presented an example issue, where the metrics have perceivable changes caused by the presence of the given issue.

Our future work is twofold: 1) we plan to extend the set of collected metrics in a more systematic way, 2) we elaborate a technique providing automated identifications of DSE problems using the metrics attached to symbolic execution trees.

References

[1] R. Baldoni, E. Coppa, D. C. D'Elia, C. Demetrescu, and I. Finocchi, "A survey of symbolic execution techniques," *CoRR*, vol. abs/1610.00502, 2016. [Online]. Available: http://arxiv.org/abs/1610.00502

[2] D. Honfi, A. Voros, and Z. Micskei, "SEViz: A tool for visualizing symbolic execution," in *ICST*, April 2015, pp. 1–8.

[3] M. M. Eler, A. T. Endo, and V. H. Durelli, "An empirical study to quantify the characteristics of Java programs that may influence symbolic execution from a unit testing perspective," *Journal of Systems and Software*, vol. 121, pp. 281 – 297, 2016.

[4] B. Katzmarski and R. Koschke, "Program complexity metrics and programmer opinions," in *2012 20th IEEE International Conference on Program Comprehension (ICPC)*, 2012, pp. 17–26.

[5] R. P. L. Buse and W. Weimer, "The road not taken: Estimating path execution frequency statically," in *31st IEEE International Conference on Software Engineering*, 2009, pp. 144–154.

[6] V. Garousi and M. Felderer, "Developing, verifying, and maintaining high-quality automated test scripts," *IEEE Software*, vol. 33, no. 3, pp. 68–75, 2016.

[7] D. Bowes, T. Hall, J. Petrić, T. Shippey, and B. Turhan, "How good are my tests?" in *Proceedings of the 8th Workshop on Emerging Trends in Software Metrics*. IEEE Press, 2017, pp. 9–14.

[8] G. Meszaros, *XUnit Test Patterns: Refactoring Test Code*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2006.

[9] P. Bhattacharya, M. Iliofotou, I. Neamtiu, and M. Faloutsos, "Graph-based analysis and prediction for software evolution," in *ICSE*. IEEE, 2012, pp. 419–429.

[10] N. Tillmann and J. de Halleux, "Pex–white box test generation for .net," ser. TAP: Second International Conference, 2008, pp. 134–153.

[11] X. Xiao, T. Xie, N. Tillmann, and J. De Halleux, "Precise identification of problems for structural test generation," in *ICSE*. IEEE, 2011, pp. 611–620.

[12] R. Hähnle, M. Baum, R. Bubel, and M. Rothe, "A visual interactive debugger based on symbolic execution," in *ASE*, 2010, pp. 143–146.

Comparison of Nanopore DNA Sequencing Basecallers on Whole Human Data

Erik Jagyugya, Peter Sarkozy
Department of Measurement and Information Systems
Budapest University of Technology and Economics
Budapest, Hungary
psarkozy@mit.bme.hu

***Abstract*—Since the release of Oxford Nanopore Technologies' MinION single molecule real-time (SMRT) DNA sequencing platform, a multitude of approaches have been evaluated to identify the exact nucleotide sequence passing through the individual pores based on the raw, picoampere level current recorded from the device. Multiple Hidden Markov Model (HMM) and artificial neural network (ANN) based basecalling approaches have been released.**

We examined the most promising academic approaches, and compared them to the reference solution provided by the platform vendor, using the NA12878 whole genome shotgun sequencing dataset. Multiple types of systematic errors offer challenges to each individual solution, thus we propose a framework to unify the strengths of each basecaller, and to aggregate their output in order to increase their accuracy over any single solution.

***Index Terms*—DNA sequencing, Basecalling, Nanopore**

I. INTRODUCTION

The MinION released by Oxford Nanopore Technologies (ONT) is a single-molecule real-time (SMRT) DNA sequencing technology that offers unprecedented read lengths in a novel, compact form factor while greatly simplifying library preparation procedures compared to current next-generation sequencing (NGS) platforms.

The method of identifying the individual nucleotides comprising a DNA sequence utilizes a nano-scale pore embedded in an artificial membrane across a semiconductor surface. The blockage of the picoampere level current passing through the pore while a DNA molecule passes through it is measured and sampled at 4kHz. The traversal of each molecule is slowed down by a ratcheting enzyme, to ensure enough time to sample the current respective of the DNA sequence. The current level is determined by 5-6 consecutive nucleotides (k-mers), and is a characteristic of the types of bases. Despite offering extremely long reads (up to 100kb), the accuracy of the platform is only approximately 90%, compared to the 99-99.9% accuracy of NGS platforms [1].

The original method to transform the current measurement into a sequence of nucleotides (basecalling) splits the current into consecutive segments (events) where the current levels were relatively static. The series of events are then passed into a hidden markov model (HMM) for decoding into a sequence of k-mers [2]. Finally, empirically calibrated quality scores are assigned to each base, and the results stored in the original fast5 HDF [3] container file.

Recently, neural networks models have been developed to perform nanopore DNA basecalling.

In our study we reviewed some of the available basecallers. We investigated characteristics such as read identity, insertion and deletion rate, the size and frequency distribution of insertions and deletions among the software tools and we summarized the overall accuracy. We unified this information and propose consensus calling by pairwise merging.

II. MATERIALS AND METHODS

A. Raw and reference data

We used the publicly available whole genome shotgun sequencing dataset of human sample NA12878 [4]. This is a well characterized genome, and is used as a gold standard in measuring the quality of human DNA sequencing.

For our study, we compared the source of the chromosome 1 data. We chose a subset of chromosome 1 which represent the whole genome well in our case, because the entire dataset was deemed too large. The subset is approximately 300 GB and contains 100000 individual fast5 files with a mean read length of 10kb. Using the rel3 version with R9.4 chemistry. We used the GRCh37 human reference genome from the Reference Genome Consortium [5] instead of NA12878 reference because the deviation between them is two orders of magnitude lower than the expected read error rate.

B. Basecalling DNA sequences

Raw reads were processed by four basecallers, Albacore [11], Chiron [10], Metrichor and Scrappie. The Metrichor data was available directly from the raw fast5 files, as it is a cloud-based basecaller that cannot be run locally.

Albacore, Chiron and Scrappie all utilize GPU support for the majority of their computations, however our version of Chiron had issues with our GPU setup so it was run in CPU only mode. The same neural network architecture is run on the CPU as the GPU, but the computations are orders of magnitude slower. We attempted to include a fifth basecaller, basecRAWler [6], but it had to be removed due errors and poor support for our hardware.

Metrichor

Metrichor is the ONT cloud-based platform basecaller [8], it works by segmenting the raw current signal into events which represent the traversion of a single nucleotide through the pore, and utilizes a HMM on the event sequence to indentify the nucleotide order. Instead of each event representing each the base individually, the current level is influenced by adjacent bases which correlate to the length of the narrowest region of the nanopore. The model supports 5 and 6 adjacent bases in the decoding step [7]. The individual events are decoded into stay and skip probabilities by the HMM, according to their duration, mean current and noise. These state transitions are decoded into the final basecalled sequence.

The HMM model was superseded by a more accurate recurrent neural network (RNN) model in early 2016 [7]. We used the data from the cloud-based version as integrated into the EPI2ME service which relied on HMM. The raw fast5 files already contained the required Metrichor basecall results, and were used subsequently without modification.

Albacore

Albacore is one of the official command-line basecallers of ONT, and is considered the "gold-standard" in accuracy. As the software is unfortunately not open source, the neural network structure used in its model is not public. We used version 2.1.3, the most recent at the time of writing. The neural network does not require segmenting the reads into events, and instead operates on the raw current levels, allowing for great improvements in basecalling accuracy.

Albacore enforces stricter limits on minimum sequence quality, and will refuse to call low quality reads, contributing to it's higher overall performance. In our study, this has no effect as we only used reads called by all 4 basecallers.

Chiron

Chiron is a novel third-party neural network based basecaller [10]. It couples a recurrent neural network (RNN), a convolutional neural net (CNN) and a connectionist temporal classification (CTC) decoder. This structure also enables it to model the raw signal data directly, without use of any segmentation step. It is the first publicly available artificial neural network which can translate raw current signals directly to nucleotide base sequences. The basecaller was trained on only a small subset of data from the lambda phage genome and from the E. coli genome, and yet it has a surprising ability to generalize to larger genomes such plant and mammal genomes. Chiron is as accurate as the ONT designed and trained Albacore in some cases, and outperforms all other existing basecallers on bacterial genomes.

Chiron consists of two sets of layers: a set of convolutional and a set of recurrent ANN see Fig. 1. The convolutional layer defines local patterns from raw signal, whereas the recurrent layer combines these pattern into base probabilities. The CTC decoder makes data segmentation unnecessary through the use of blank labels inserted into the sequences. This allows it to model input and output sequences of varying lengths. It models a two dimensional graph similar to a pairwise alignment matrix, and the neural network predicts the probabilities of the transitions along the alignment matrix.

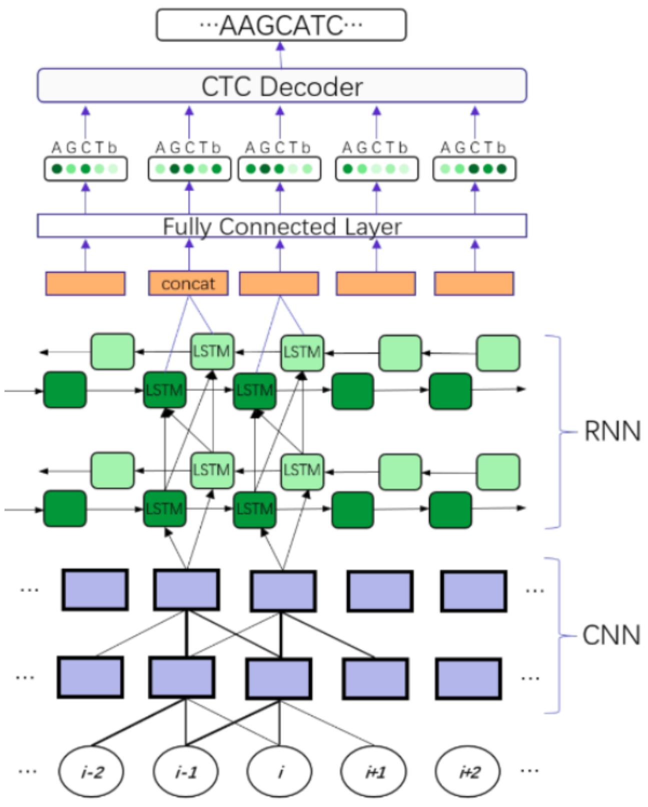


Fig. 1. The neural network architecture used by Chiron [10]

Scrappie

Scrappie is another basecaller developed by ONT, this is the first basecaller that was developed specifically to address homopolymer deletions. One of the major hurdles in SMRT sequencing is accurately determining length of homopolymers [9]. Scrappie was run parallel with Metrichor and NanoNet (a first generation experimental NN basecaller) on human chromosome 20. It was found that Scrappie indeed called homopolymer areas more accurate than the other two basecaller. Homopolymeric stretches of up to 16 bases were called accurately which referred to the transducer-based homopolymer calling[8][12]. We utilized version 1.3.0 with the *raw model* to perform basecalling.

C. Processing DNA sequences

We prepared our dataset by first running all of the basecallers and extracting the relevant data for the entire set. Not all basecallers produce output sequences for every input, so we selected a subset of reads that had calls from each basecaller. All of the software tools use different naming schemes when outputting results, so these had to be unified to contain only the read ID string. The resulting sequences were aligned to the reference genome using the Burrows-Wheeler Aligner which is tolerant with errors given longer query sequences [13][14], and all alignments with mapping qualities below 30 were discarded, because low alignment quality is indicative of the read originating from a different chromosome, especially at the read lengths produced by the ONT platform. In the alignment processing BWA uses soft clipping. Some reads had secondary alignments, but only the highest-scoring alignment was used. We investigated the overall match, mismatch and insertion rates of each dataset.

III. Results

A. Systematic errors

While all tested basecallers perform reasonably well on our dataset, homopolymer stretches (repeating identical nucleotides) still present a challenge. HMM based basecallers struggle with calling homopolymers longer than 5 bases, as the event segmentation process hides the very long times that the current stays static because there are identical nucleotides in the pore, as shown on Fig. 4. Albacore called the overall lowest number of deletions and the highest identity rate too, as NN based tools tend to cope better with homopolymer stretches. Insertions were overall less common, and mostly randomly distributed along the sequence without any sequence-specific bias Fig. 3. A insertions and deletions were twice as common as T indels, with G and C indels being equally common. A/G and C/T nucleotide substitutions are approximately 3x more frequent than other substitutions, as they share similar current signatures due to their similar purine/pyrimidine molecular structures, respectively as shown in Tab. III-A and Tab. III-C.

B. Unified Basecaller

The aggregation of the output of multiple basecallers to obtain a consensus read is nontrivial in the case of ONT reads. While multiple sequence alignment approaches are feasible for individual genes and transcripts, even pairwise alignment quickly becomes computationally expensive with ONT read lengths. We implemented a method to perform the pairwise merging of all basecaller output sequences. We find the optimal pairwise alignments with the Smith-Waterman algorithm, using a match score of 1, a mismatch, gap open and gap extension penalty of -1. Each read is successively merged, taking to account the PHRED-scaled quality scores. On matches, the maximum of the quality is passed on. On mismatches, the higher quality base overrides the lower quality one, and on gaps, a configurable threshold is specified to select the sensitivity in which insertions and deletions are selected. Unfortunately, Scrappie did not provide base quality scores, so we used a flat PHRED score of 10 to approximate it.

C. Read identity

Read identity was derived from total base matches. Albacore and Scrappie have the best performance as shown on Fig. 2. These basecallers developed by ONT. However Metrichor performed the worst which is also an ONT product. As Metrichor relies on a HMM its accuracy is far away from basecallers based on NN. Chiron and Consensus medians are near each other but their distribution is markedly different. Chiron produces more accurate read against Consensus. The overall identity rates are shown on Tab. III-C.

Albacore

Matches	**A**	**C**	**G**	**T**
A	22.98%	0.35%	1.11%	0.34%
C	0.30%	21.16%	2.35%	0.40%
G	0.26%	0.26%	20.91%	0.37%
T	0.34%	0.47%	0.31%	23.95%

Chiron

Matches	**A**	**C**	**G**	**T**
A	20.87%	0.41%	1.34%	0.36%
C	0.33%	20.06%	0.23%	0.59%
G	1.71%	0.35%	19.86%	0.39%
T	0.27%	0.39%	0.25%	22.40%

Metrichor

Matches	**A**	**C**	**G**	**T**
A	20.92%	0.45%	1.44%	0.37%
C	0.27%	19.48%	0.22%	0.46%
G	1.34%	0.34%	19.30%	0.35%
T	0.37%	0.55%	0.32%	22.54%

Scrappie

Matches	**A**	**C**	**G**	**T**
A	22.32%	3.73%	1.32%	0.32%
C	0.31%	20.62%	0.23%	0.37%
G	0.11%	0.27%	20.34%	0.27%
T	0.37%	0.52%	0.33%	23.58%

TABLE I
The match and mismatch rate of all 4 basecallers. Columns specify reference bases, while rows show the read bases.

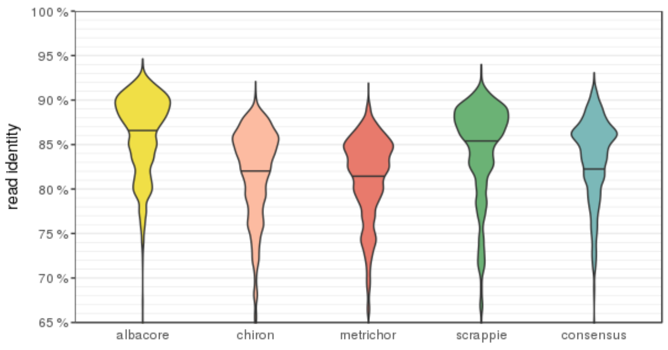


Fig. 2. Read identity distribution and medians which was weighted by read lengths it is marked by black horizontal line.

IV. Conclusion

The characteristics of each basecaller show that currently Albacore performs the best on our dataset. We have confirmed that most of the errors in the sequences crop up at identical positions in the sequences, meaning that they are artifacts of the underlying measurement process, and unifying the output of multiple basecallers can only offer limited improvements in accuracy. The pairwise merging of the output of each basecaller into a consensus sequence benefits from the specification of each basecaller's error characteristics, as they provide

	Consensus			
Matches	**A**	**C**	**G**	**T**
A	22.18%	4.42%	1.02%	0.40%
C	0.46%	20.88%	0.37%	0.46%
G	0.97%	0.36%	20.53%	0.35%
T	0.50%	0.54%	0.47%	23.51%

TABLE II
The match and mismatch rate of consensus sequences. Columns specify reference bases, while rows show the read bases.

Basecaller	**Insertion rate**	**Deletion rate**	**Identity rate**
Albacore	3.57%	5.54%	89.01%
Chiron	1.92%	10.16%	83.2%
Metrichor	1.52%	11.20%	82.25%
Scrappie	2.63%	7.31%	86.86%
Consensus	5.21%	5.33%	88.06%

TABLE III
The overall accuracy of the 4 tested basecallers, and the results of the consensus sequences

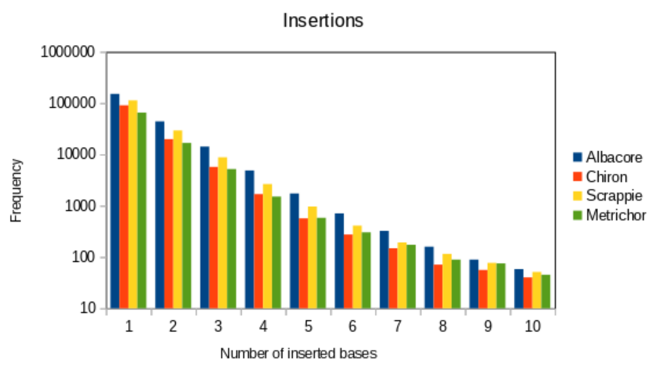


Fig. 3. The size and frequency distribution of insertions among the 4 basecallers

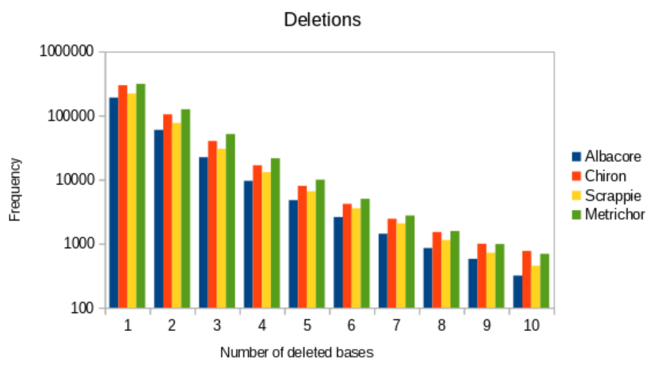


Fig. 4. The size and frequency distribution of deletions among the 4 basecallers

additional information when considering the individual base quality scores. However, the quality scores provided by each basecaller show marked differences, and it could prove useful to perform base quality score recalibration based on empirical results. While our unified basecalls did not show an overall lead above all individual basecallers, it did provide more robust results overall, with a more reliable per-base quality score.

ACKNOWLEDGMENT

The authors acknowledge that they are participants in the Oxford Nanopore Technologies's MinION Access Program. The Titan Xp used for this research was donated by the NVIDIA Corporation. This research was supported by the OTKA-K-112915 Grant, and the Multipurpose Health Monitoring Platform bilateral Croatian-Hungarian grant. The authors state that they have no other conflicts of interest.

REFERENCES

[1] Quail MA, et al, A tale of three next generation sequencing platforms: comparison of Ion Torrent, Pacific Biosciences and Illumina MiSeq sequencers. BMC Genomics 2012; 13:341; DOI:10.1186/1471-2164-13-341
[2] Matei D., et al, Nanocall: an open source basecaller for Oxford Nanopore sequencing data. Bioinformatics 2017 DOIL10.1093/bioinformatics/btw569
[3] The HDF Group. Hierarchical Data Format, version 5, 1997 http://www.hdfgroup.org/HDF5/.
[4] Oxford Nanopore Human Reference Datasets ,https://github.com/nanopore-wgs-consortium/NA12878, 11-15-2017
[5] Human Genome Overview - Genome Reference Consortium, https://www.ncbi.nlm.nih.gov/grc/human, 03-12-2017
[6] Stoiber M., Brown J.: BasecRAWller: Streaming Nanopore Basecalling Directly from Raw Signal BioarXiv 2017 DOI:10.1101/133058
[7] C. V. de Lannoy, D. de Ridder, J. Risse, A Sequencer Coming Of Age: De Novo Genome Assembly Using MinION Reads, https://www.biorxiv.org/content/early/2017/05/26/142711, 2017-12-07
[8] Oxford Nanopore Technologies, https://nanoporetech.com/, 2018
[9] P. Antal P. Sarkozy, Á. Jobbágy. Calling Homopolymer Stretches from Raw Nanopore Reads by Analyzing k-mer Dwell Times. 2016. DOI: 10.1007/978-981-10-5122-7-61.
[10] T. Haotian, C. M. Duc, H. M. B., D. Tania, W. Sheng and C. Lachlan, Chiron: Translating nanopore raw signal directly into nucleotide sequence using deep learning, bioRxiv
[11] Oxford Nanopore Technologies, https://nanoporetech.com/about-us/news/new-basecaller-now-performs-raw-basecalling-improved-sequencing-accuracy, 11-15-2017
[12] S. Wermter, Knowledge Extraction from Transducer Neural Networks, Journal of Applied Intelligence, 12, 27, 44 (2000)
[13] Li H. and Durbin R. (2010) Fast and accurate long-read alignment with Burrows-Wheeler transform. Bioinformatics, 26, 589-595. [PMID: 20080505]
[14] Li H. Burrows-Wheeler transform repository, https://github.com/lh3/bwa, 11-15-2017

Preliminary Performance Assessment of Hyperledger Fabric

Attila Klenik, András Pataricza

Budapest University of Technology and Economics
Department of Measurement and Information Systems
Budapest, Hungary
Email: {klenik, pataric}@mit.bme.hu

***Abstract*—After the success of bitcoin proved the viability of the distributed ledger technology, other frameworks emerged with the goal of providing a general purpose blockchain platform for businesses to execute smart contracts. The private and permissioned platforms are promising replacements for many current systems in several sectors, such as finance, healthcare, and IoT.**

Such timeliness and throughput critical applications require a framework offering predictable temporal characteristics. Benchmarking is a traditional method for measuring (and later predicting) the performance related extra-functional characteristics of a system. However, blockchain frameworks currently lack standard benchmarks due to the novelty of the domain.

This paper present a preliminary performance assessment of a private and permissioned blockchain framework, Hyperledger Fabric. The evaluation is based on a synthetic load and targets the scalability and performance of different functional parts of the framework architecture. Findings are presented for the different phases of the lifecycle of transactions in the framework.

I. Introduction

Motivated by the success of Bitcoin [1], the interest for blockchain technologies grew rapidly in the past years. The irrefutable way of committing and auditing transactions without a third trusted party makes blockchain a promising new technology for numerous sectors, e.g., finance,[1] business workflows [2], healthcare, the government sector,[2] Internet of Things [3], and Cyber-Physical Systems.

The use cases of distributed, "trustless," Bitcoin-inspired peer-to-peer systems are wider than managing cryptocurrencies. This inspiration gave rise to a large number of continuously evolving, experimental systems. These all modify the "Bitcoin architecture" (and protocol) in fundamental ways, but two aspects do not vary: a) there is a blockchain and peers maintain a copy of it and b) peers participate in a distributed consensus protocol to maintain a consistent state of the system.

The complexity of blockchain systems makes it challenging to ensure certain requirements. Such requirements include temporal properties, like throughput and timeliness; dependability properties, like availability; and security aspects, like confidentiality and privacy. Employing model driven development (MDD [4]) allows the application of model-based simulations and (formal) analyses to facilitate ensuring these requirements. A subset of these models captures performance-related behaviours enabling the performance prediction of complex systems [5]. Model-based performance analysis supports the identification of bottlenecks and execution of sensitivity analysis in the early stages of development, both on platform independent models (e.g., functional architecture) and platform dependent models (e.g., the configuration of system components).

This paper presents a preliminary performance assessment of a pilot reference implementation of a general purpose blockchain platform, Hyperledger Fabric,[3] through a systematic methodology.

II. Architecture Elements

In order to discuss the evaluation methodology and the findings, it is important to first introduce the key concepts of Hyperledger Fabric. The blockchain network consists of a set of *nodes*. Each of them maintains its own replica of the same tamper-proof key-value store – called the ledger – through constant communication. Note, that the concept of nodes above is used in the functional sense without constraining the deployment to physical servers. Nodes can take on two main roles in the system:

- *Peer nodes* store and maintain their own local copy of the ledger. If this is their only responsibility, then these nodes are referred to as *committer* peers. However, the users of the network can deploy executable "smart contracts" – chaincodes in Hyperledger Fabric terminology – to some of the peer nodes.
  In this case these nodes also participate in validating the transaction executions, i.e., the chaincode invocation attempts of the user. This process is called endorsement (detailed in Section III), consequently, these nodes are referred to as *endorsing* peers.
- *Orderer nodes* deliver transactions to the committer peers. The ordering service guarantees atomic delivery of messages (also known as total-order broadcast, or consensus in distributed system theory) and some degree of reliability, depending on the actual implementation of

[1]http://www.reuters.com/article/banking-blockchain-bonds-idUSL8N16A30H
[2]https://www.gov.uk/government/publications/distributed-ledger-technology-blackett-review
[3]https://www.hyperledger.org/projects/fabric

the service. Due to efficiency reasons, the service implementations usually broadcast the ordered transactions in batches.

The previous versions of Hyperledger Fabric (v0.x) had serious limitations regarding scalability that manifested in a relatively low effective throughput: based on previous measurements [6], around 300 transactions per second. The issue originated from the architectural design that first ordered (in batches) then executed the transactions in a sequential manner. The sequential execution of chaincode invocations proved to be a bottleneck in the system even for simple chaincode implementations.

The new architecture of Hyperledger Fabric v1.0 was subject to a fundamental redesign in order to alleviate this problem. The transaction execution phase now precedes the ordering and is performed in parallel. Moreover there is an additional validation step before committing the ledger modifications of the transactions. The life-cycle of a transaction is detailed in the next section.

III. Transaction Life-cycle

The redesigned architecture resulted in a more complicated transaction life-cycle than in the previous versions. A transaction has to go through numerous steps before its effect is reflected in the ledger. These steps are the cornerstones of the current evaluation, so their detailed understanding is a must. The phases of transaction committing are the following:

1) The client initiates a transaction proposal towards a subset of endorsing peers. This step is usually referred to as sending a transaction for endorsement (detailed in the next step). The transaction is considered endorsed if and only if a "sufficient" number of peers endorse it according to some endorsement policy (e.g., at least 2 out of 3 participants).
2) The endorsing peers (that received a proposal) execute the chaincode with the parameters of the transaction. The chaincode is executed against the current state of the ledger (based on the latest committed transaction block) and every proposed transaction is executed in parallel. This is a significant difference compared to the previous architecture, where the transaction execution was strictly sequential.
   During the execution, the endorsing peer builds a read set and a write set. The read set contains the keys and their versions that were read from the ledger during the execution. The write set contains the keys and their values that were written to the ledger during the execution.
   It is important to note, that the written values are *not* actually committed to the ledger at this point, they are simply gathered by the endorsing peer. Accordingly, this step is also referred to as simulating the transaction.
3) The endorser peers send back a result to the client indicating whether the execution was successful or not, together with the produced read and write set.
4) The client inspects the received results and decides whether to proceed with the next step or not. If the endorsement policy for the chaincode is not satisfied by the received endorsement results, then the transaction will ultimately be rejected as invalid (detailed in step 9), so it is better not to submit it, but try again the proposal. Furthermore, if the produced read sets are different across endorsing peers, then it is recommended to retry the proposal step.
5) The client sends the gathered endorsements along with the read and write sets to the ordering service.
6) The ordering service acknowledges that the transaction was submitted for ordering.
7) The ordering service produces a transaction block from the received transactions, either based on a predefined timeout or on the maximum number of transaction permitted in the block.
8) The ordering service delivers the produced block to the committing peers in the network.
9) The committing peers validate the submitted transaction. At this point the transaction will be definitely included in the blockchain, irrespectively of the validation result. The validation includes checking whether the chaincode policy is satisfied by the submitted endorsements and whether the key versions in the read set match the current state of the ledger. The latter verification is also referred to as multiversion concurrency control (MVCC) in the database domain.
   This means, that the version of every key in the read set must match the version of those keys in the current state of the ledger. If meanwhile an other transaction has already updated a key from the read set, then the transaction will be marked invalid.
   The transaction is considered valid if and only if the endorsement policy is satisfied and the key versions in the read set match the key versions in the current state of the ledger. If this holds, then the write set is committed to the ledger. Note, that at this point, the transaction is not executed again, only its write set is written to the ledger.

The performance of these steps will be the main focus of the evaluation, detailed in Section VI.

IV. Measurement Details

This section presents the goals of the evaluation, the details of the measurement environment (i.e., the used network topology), deployment details and the business logic specific parts of the evaluation, i.e., the chaincode and the client.

A. Evaluation goals

The goal of the current evaluation is not to determine the effective throughput of the platform, but to inspect the scalability and relative performance of the presented transaction life-cycle steps. The main focus is on the chaincode execution times and on the different steps that process the read and write sets produced by the execution.

To this end the evaluation will focus on the response time changes of these steps in case of different ledger update sizes. The sizes of these ledger updates affect the chaincode execution time through the ledger access, and also affect the validation and commitment times of transactions due to sizes of the read and write sets.

B. The environment

The network consists of two participants (referred to as organizations from now on), each maintaining two endorsing peers (for availability reasons).

The ordering service consists of a single orderer node that orders the transactions in a first-come-first-served manner. In production environments it is highly recommended to use a more robust topology for the ordering service (e.g., PBFT [7]).

The entire network is deployed on a single virtual machine using Docker containers. The virtual machine had access to four physical CPU cores (each with 3.1 GHz clock frequency), 8 GB of RAM and was backed by a solid state drive.

C. The chaincode and the client

Due to the lack of standard blockchain platform benchmarks, the deployed chaincode only provides functionality for updating (reading then writing) entries in the ledger. The number and size of the updates are configurable at the time of deployment in order to be capable of approximating the ledger load of real-life chaincodes with different characteristics.

During deployment of the chaincode enough keys were inserted into the ledger at deploy time to provide conflict-free updates for two blocks of transaction, assuming that the block timeout of the ordering service is one second. This means that transactions updating the same entry are approximately two seconds apart, so the updates will be conflict free as long as the committing times of these transactions are under two seconds. As we will see, this is not always the case.

The client that generated the transactions is a general blockchain platform benchmark tool, called Caliper.[4]

V. Evaluation workflow

This section presents the steps of the evaluation and their results. The process followed an iterative workflow (Figure 1, [6]) that systematically guides the evaluation through a sequence of steps with well-defined tasks to perform. This preliminary evaluation only covers the first six steps.

A. Operating envelope

Defining an operating envelope consist of limiting the input space of the system to a subset, that will keep the system within the desired operational conditions throughout the evaluation. Since the goal is to observe the response time changes based on the ledger load, we choose a transaction load rate that minimizes the number of invalid transactions in the system.

Based on some short preliminary test runs, a 100 transactions per second was selected as constant load rate for the rest of the evaluation. Due to the cyclic behaviour of the chaincode, the load duration time was selected to be 2 minutes for each measurement. This means that for each measurement approximately 12000 transactions were submitted and the timing of their steps recorded.

B. Rough functional architecture

The components of the system covered by the measurements correspond to the steps of the transaction life-cycle. Unfortunately, not every step can be measured on its own, at least not without performing additional instrumentation on Hyperledger Fabric.

The platform only provides an event about the commit result of a transaction. This means that the client can only observe the execution times after the endorsement, the orderer acknowledgement, and the commit event. These times will also contain the network latency, but since the network is deployed on a single virtual machine, this time should be negligible. The observed attributes are detailed Section VI

C. Instrumentation and harness

Caliper provides timing data for different steps of the executions that can be processed in an arbitrary manner. The chaincode execution time is logged in the running chaincode and retrieved from the Docker container logs after the measurement is over.

D. Experiment campaigns

The main parameters of the measurements are the number and size of the updates a chaincode invocation will perform. The explored parameter space consists of the combination of the values of the following two variables:

- Number of updates: 1, 2, 4, 8
- Size of updates (in bytes): 8, 32, 128, 512, 2048

These parameters currently have a technical limit, since the underlying remote procedure call library (gRPC[5]) has a maximum message size configured to approximately 4MB. This limits the number and/or size of entries created in the ledger during deploy time, since deployment also produces a read and write set that is sent back to the client.

VI. Exploratory Data Analysis

This section describes the attributes of the collected data and presents preliminary results using exploratory data analysis (EDA). EDA offers a wide variety of tools and methods to intuitively assess larger datasets without detailed knowledge of the measured system. During the measurements the following attributes were collected about each transaction:

- *create time*: The CPU time when the transaction was created, measured with millisecond precision. Unique identifier for transactions.
- *endorse time*: The time it took to send the transaction proposal to the endorsing peers, execute the transaction and receive the endorsements from the peers. Measured with millisecond precision.

[4]https://github.com/Huawei-OSG/caliper

[5]https://grpc.io/

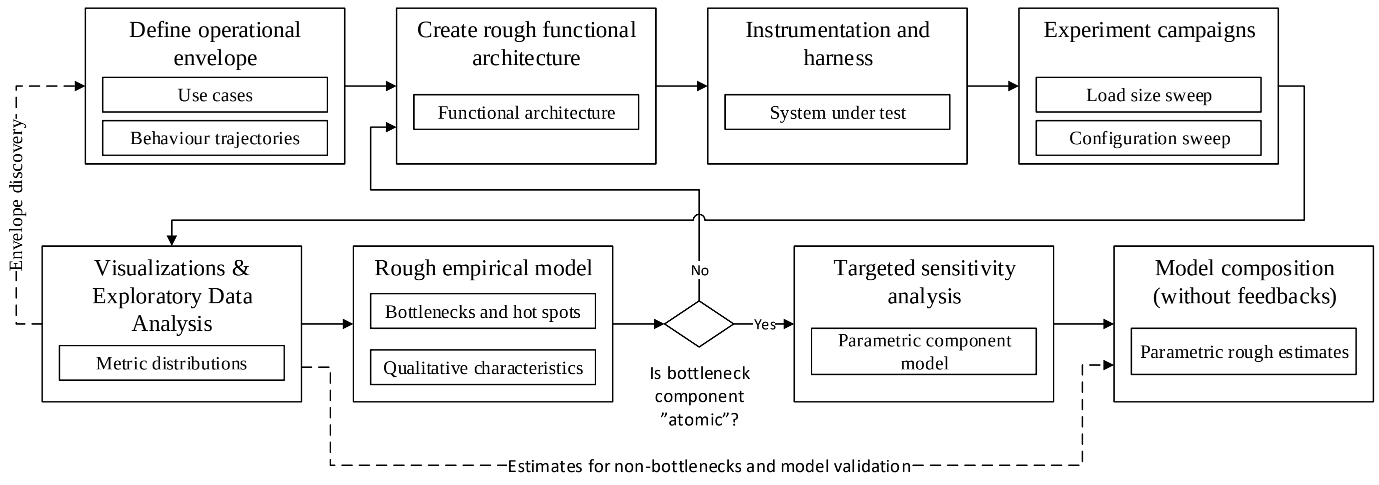

Define operational envelope
Use cases
Behaviour trajectories
Create rough functional architecture
Functional architecture
Instrumentation and harness
System under test
Experiment campaigns
Load size sweep
Configuration sweep
Envelope discovery
Visualizations & Exploratory Data Analysis
Metric distributions
Rough empirical model
Bottlenecks and hot spots
Qualitative characteristics
No
Yes
Is bottleneck component "atomic"?
Targeted sensitivity analysis
Parametric component model
Model composition (without feedbacks)
Parametric rough estimates
Estimates for non-bottlenecks and model validation

Figure 1: Evaluation methodology

- *order time*: The time it took to check the received endorsements, submit them to the ordering service and receive the acknowledgement from the ordering service. Measured with millisecond precision.
- *validation time*: The time it took to order the transaction and validate it at the committing peers and received the event about the result. Measured with millisecond precision.
- *org[1/2] execution time*: The time it took to execution the transaction on the current endorsing peer of the first and second organization. Measured with millisecond precision.

A. Assumptions

Based on a priori knowledge of the platform, the following assumptions were made, which guided the exploratory data analysis:

1) Due to the extra number of entries inserted in the ledger, there should be no conflict between transactions updating the same entries. This means that the transactions should be committed in a two block time range, which is approximately 2 seconds.
2) The execution times of transactions should be symmetric on both endorsing peer, since the same deterministic chaincode is running with the same parameters.
3) The following times should be sensitive for the size of the ledger load: execution time, commit time. Normally, the network communication time would also be in this list, but we omit it now due to the local network communication. Unfortunately the commit time contains the ordering time also, so the pure version validation time cannot be separated in the current evaluation.

B. Assumption checks

During the measurement with the biggest ledger load (8 updates, each 2048 bytes) there were almost 2500 failed transactions. Every other measurement committed every transaction successfully, except for a small transient time period, where the resource utilization of the host machine interfered with the resources of the virtual machine.

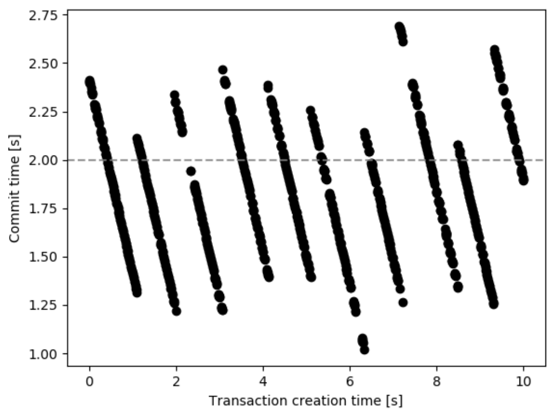


Figure 2: Total execution times of transaction

So the first assumption holds for almost every measurement, except one. Figure 2 shows the total commit time for the transactions initiated in the first 10 seconds of that measurement. The plot shows that numerous transactions had a commit time above 2 seconds. This results in a conflict for the next transaction that tries to update the same value. The second transaction still reads the old value (with the same key version), since the first transaction has not been committed yet. But by the time the second transactions is about to be committed, the first already overwrote that value, resulting in a version conflict.

The second assumption is a reasonable one, since the same code is executed on both endorsing peers in the same way. However, Figure 3 refutes this assumption. If the assumption was true, the observations would lie along the diagonal of the plot. This phenomenon could be explained by the resource constrained environment, but Figure 4 reveals another discrepancy.

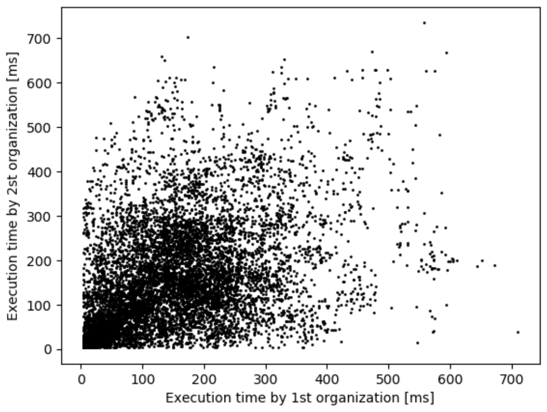


Figure 3: Execution times for the different organizations

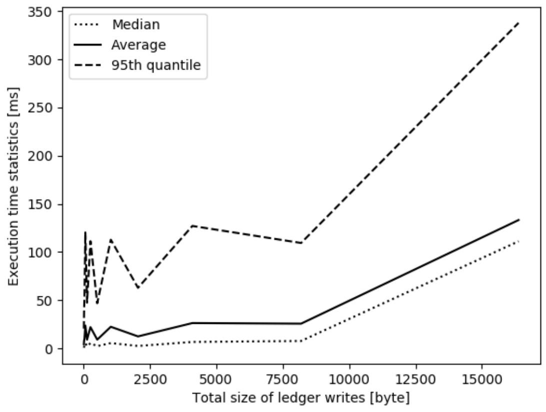


Figure 5: Execution times for the different ledger loads

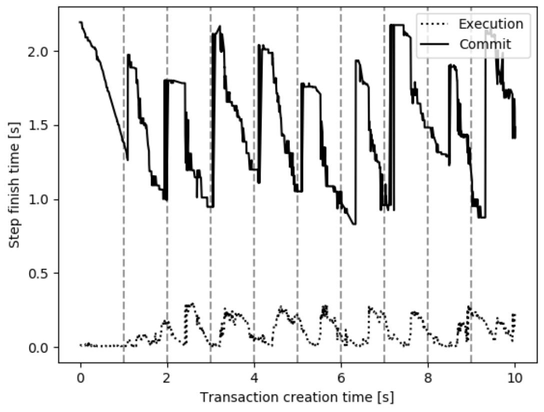


Figure 4: The pattern of execution and validation time of transactions

The vertical dashed lines are denoting the block timeouts, and the sawtooth-like plots approximately fit them. Note, that the transactions have higher execution times when being committed in the middle of a block, except for the first block. This phenomenon requires further, more detailed investigation, possibly through additional instrumentation of Hyperledger Fabric. For this reason, it is outside the scope of this paper.

The third assumption is checked by calculated the mean, median and 95th quantile of the execution times for the different ledger load sizes. This is illustrated in Figure 5. The plot clearly shows the sensitivity of the execution time for the size of ledger writes generated by the chaincode. The sensitivity should be investigated for larger ledger load sizes, but this requires some modification to the chaincode behaviour and Caliper timing data harnessing, consequently it is left as future work.

VII. Conclusion and Future Work

In this paper we presented the detailed working of the new architecture of Hyperledger Fabric, a pilot reference implementation of a general purpose blockchain platform. Then following a systematic methodology, we evaluated the scalability of the platform using different ledger loads. We concluded, that the sensitivity assumption between the ledger load size and the chaincode execution time holds. However, based on current data, it is hard to argue about the timeliness of the platform.

The presented assumption checks showed that the evaluation methodology is capable of delivering important knowledge about the platform. As future work, the evaluation must be repeated in a cloud environment, lifting the current resource constraints, which could be the source of significant noise in the data. Moreover, the ledger load size constraint should be circumvented, and measurements repeated for bigger parameter values. Furthermore, in order to extract network communication delays from the data, a more detailed monitoring or instrumentation is needed, preferably at the transaction lifecycle step level.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[2] I. Weber, X. Xu, R. Riveret, G. Governatori, A. Ponomarev, and J. Mendling, "Untrusted Business Process Monitoring and Execution Using Blockchain," in *Business Process Management: 14th International Conference, BPM 2016, Rio de Janeiro, Brazil, September 18-22, 2016. Proceedings*, Marcello La Rosa, Peter Loos, and Oscar Pastor, Eds. Cham: Springer International Publishing, 2016, pp. 329–347.

[3] K. Christidis and M. Devetsikiotis, "Blockchains and Smart Contracts for the Internet of Things," *IEEE Access*, vol. 4, no. 99, pp. 2292–2303, 2016.

[4] B. Selic, "The pragmatics of model-driven development," vol. 20, no. 5, pp. 19–25.

[5] S. Balsamo, A. D. Marco, P. Inverardi, and M. Simeoni, "Model-based performance prediction in software development: a survey," vol. 30, no. 5, pp. 295–310.

[6] I. Kocsis, A. Klenik, A. Pataricza, M. Telek, F. Deé, and D. Cseh, "Systematic performance evaluation using component-in-the-loop approach," *International Journal of Cloud Computing*, submitted.

[7] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, ser. OSDI '99. Berkeley, CA, USA: USENIX Association, 1999, pp. 173–186.

Oscillometric blood pressure measurement using constant cuff pressure intervals

Péter Nagy, Ákos Jobbágy
Budapest University of Technology and Economics,
Department of Measurement and Information Systems,
Budapest, Hungary
Email: {nagy, jobbagy}@mit.bme.hu

***Abstract*—Oscillometric blood pressure (BP) measurement is a widely used method to assess the state of the cardiovascular system. Fast inflation and slow deflation of the cuff causes a constantly changing excitation for the cardiovascular system. This paper describes an electronic device and a measurement method using a special cuff pressure profile containing constant pressure intervals. Keeping cuff pressure (CP) at constant value can help measure and compensate the reaction of the cardiovascular system to the occlusion. Pulse wave transit time (PWTT), heart rate variability (HRV) as well as the amplitude of the photoplethysmographic (PPG) signal at the fingertip were examined. The device aids home health monitoring, it does not require trained operator.**

***Keywords*—oscillometry, blood pressure measurement; pulse wave transit time; heart rate variability**

I. INTRODUCTION

High BP is an important cardiovascular risk factor, if it is left undetected and uncontrolled, it can lead to vascular and cerebrovascular diseases. Elevated BP is also linked causally to kidney failure and dementia. Accurate BP measurement is the basis of optimal diagnosis and treatment of hypertension [1]. Most indirect BP measurement techniques are based on fast inflation and slow deflation of the cuff, typically placed on the left upper arm [2]. The occlusion of the brachial artery caused by the cuff changes the diameter of the artery as well as the strain in the arterial wall. These effects can be considered an excitation, to which the cardiovascular system reacts [3]. This means that the measurement method itself influences the measured quantity. Keeping CP at constant value can help observe the reaction of the cardiovascular system to a constant excitation. Breathing also influences BP [4]. This influence is superimposed on the changes caused by cuff occlusion and if CP is changing constantly, the effect of breathing is difficult to analyze separately. Applying constant CP can also help overcome this problem. Oscillometric BP measurement is a commonly used method to determine systolic and diastolic BP. The method basically estimates the mean arterial pressure, systolic and diastolic pressure are calculated. Arterial stiffness can have great impact on accuracy of the calculated values, thus reliability of the method is questionable for those with cardiovascular diseases [5].

II. MATERIALS AND METHODS

A. The measurement device

A home health monitoring device (HHMD) was developed at our laboratory more than a decade ago. A novel method for the examination of the state of the cardiovascular system is PWTT, which is the time while the pressure wave generated by the heart propagates from the aortic valve to a peripheral part of the body (typically the fingertip). PWTT can also be used to determine the frequency and the phase of breathing [6]. In order to calculate PWTT, the developed device measures not only CP, but also ECG and PPG signal. In the past 15 years, more than 2000 measurements have been recorded by the device including measurement of patients with cardiovascular disease. Analysis of the measurements has pointed out some weak points of the device and raised the need for the development of a new device. The HHMD used reflection type PPG sensors without fixation of the sensor to the measured finger. As a result of motion artifacts, signal-to-noise ratio of the PPG signal was not sufficient in several measurements, making the calculation of PWTT inaccurate. The new device contains a transmission type PPG sensor that is fixed to the measured finger. The HHMD used ECG electrodes integrated into the housing of the device, the tested person had to place their palms onto the housing. As a result, size of the housing in the longest dimension exceeded 45 cm, making the device hardly portable. The new device uses conventional limb electrodes connected to the device by wires, enabling a housing with portable size (26 cm in the longest dimension). Both HHMD and the new device measure ECG in Einthoven I lead. The sampling frequency is 1 ksample/s for both devices. The HHMD used a single controllable one-way valve for fast deflation of the cuff and a constriction element for slow deflation. The device was not able to keep CP at constant value. In addition to the one-way valve and the constriction element, the new device also contains a two-way valve. The one-way valve and the constriction element are connected in parallel and to them, the two-way valve is serially connected. Thus, when the two-way valve is closed, CP can be kept at constant value. The new device with transmission type PPG sensor, limb electrodes connected by wire, two valves and reduced size is more appropriate for the planned measurements than the original device; it is portable and can be used without special training. Figure 1 shows a picture of the developed device.

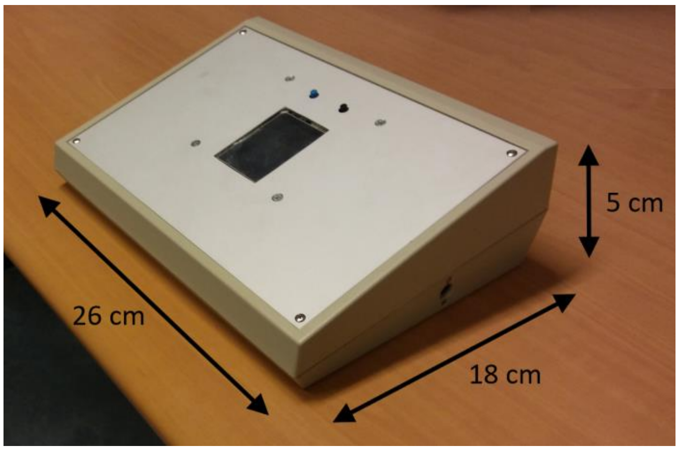


Fig. 1. The developed measurement device and its dimensions.

B. *Tested Persons*

Ten healthy test subjects volunteered for the study, two seniors (age between 66 - 67 years), four middle aged persons (age between 39 - 57 years) and four young adults (age between 22 - 26 years), male:female ratio was 6:4.

C. *Measurement protocol*

During the measurement, tested persons were sitting at rest in a silent room with 25°C temperature. In the first 24 s of the measurement, only ECG and PPG signals were recorded, the cuff was not inflated. After 24 s, inflation started with approximately 6 mmHg/s speed, until 150 mmHg was reached. At 150 mmHg, inflation stopped and deflation started immediately, approximately with the same speed as inflation. Then, deflation stopped at CP = 90 mmHg for 1 minute and at 60 mmHg for 1 minute. When 40 mmHg was reached during deflation, CP changed abruptly to 0 mmHg. After complete deflation, only ECG and PPG signals were recorded for further 24 s, then the measurement ended. Cuff pressure as a function of time during the measurement is shown in Figure 2.

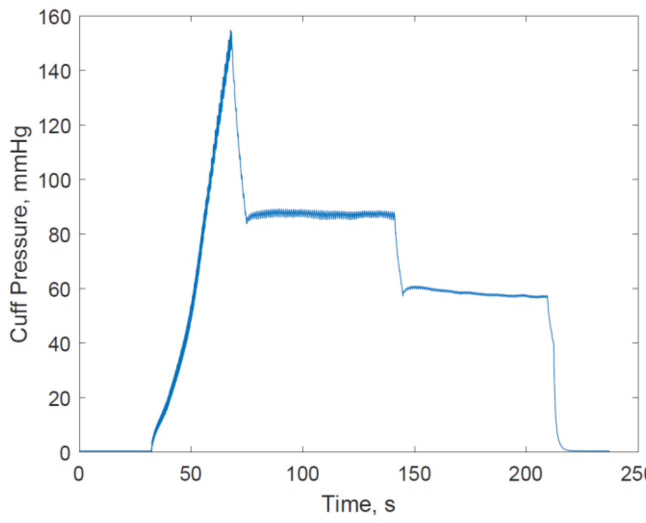


Fig. 2. Cuff pressure as a function of time during the measurement.

III. Data Analysis

A. *Improving signal-to-noise ratio*

ECG, and PPG signals are likely to be distorted by motion artifacts and electrical noise of the measurement device and the measurement environment. Therefore, the signals have to be filtered. However, distortion caused by filtering must be minimized in frequency ranges corresponding to physiological processes of interest. We filtered the ECG and PPG signals using band-pass filters with cutoff frequencies, conventionally used for ECG and PPG filtering. In case of the CP signal, recorded by our device, spectral energy is concentrated below 8 Hz. We used a low-pass filter with cutoff frequency at 8 Hz. Table I shows cutoff frequencies of the applied filters.

Using the CP signal, the time of the propagation of the pulse wave from the heart to the cuff can also be determined. A widely used method for the examination of oscillometric pulses is to use a high-pass filter [7]. For the detection of onset points of signals in general, zero crossing points of the first derivative are commonly used. High-pass filtering however, works as differentiation, therefore we recommend detrending of CP without high-pass filtering, before taking the first derivative. As inflation and deflation speed is not perfectly constant, the method we used performed detrending for each heart cycle separately based on local minima in CP.

TABLE I. Cutoff Frequencies of the Applied Filters

Signal	Lower cutoff frequency (Hz)	Upper cutoff frequency (Hz)
ECG	6	16
PPG	0.5	10
CP	-	8

B. *PWTT calculation*

PWTT was calculated using ECG and PPG signals from the heart to the fingertip, therefore we denote it by PWTTHF. We calculated PWTTHF as the time difference between the R-peak in ECG and the local minimum in the PPG signal, corresponding to the same heart cycle. PWTTHF defined this way is thus the sum of the cardiac pre-ejection period (PEP), that is the period of isovolumetric ventricular contraction, and the vessel transit time [8]. Ahlstrom et al. [8] and Wong et al. [9] pointed out that PEP is an important contributor to the correlation between PWTT and BP, and the inclusion of PEP in PWTT increases the accuracy of BP estimation and respiration monitoring. Detection of the arrival of the pulse wave to the cuff makes it possible to cut PWTTHF into two parts, the propagation time from the heart to the cuff (PWTTHC) and from the cuff to the fingertip (PWTTCF). Mean values of the PWTTHF, PWTTHC and PWTTCF parameters can be calculated for time intervals with constant CP. Denoting the mean PWTTHC value while CP equals 90 mmHg as PWTTHC_90 and the mean PWTTHC value while CP equals 60 mmHg as PWTTHC_60, ratio of these mean values can be defined. PWTTHC_90_60 is defined as in (1).

$$PWTTHC\_90\_60 = PWTTHC\_90 / PWTTHC\_60 \qquad (1)$$

For PWTTHF and PWTTCF, similar ratios can be defined. These ratios can help quantify the change of PWTT as a result of the change in CP.

C. *Assessment of the change in the PPG signal amplitude*

Amplitude of the PPG signal also depends on the blood flow rate into the body part where the PPG sensor is placed. This rate is influenced by cuff occlusion, when the cuff CP exceeds diastolic BP, the artery is closed during a certain part of the cardiac cycle. When the occlusion is eliminated, the blood flow rate in body parts distal to the occluded region increases and exceeds the flow rate that was present before occlusion was applied. This phenomenon is called reactive hyperemia [3]. Increase of blood flow rate during reactive hyperemia is impaired in patients with cardiac risk factors [10]. Denoting the mean PPG amplitude while CP equals 90 mmHg as PPG_90, and mean PPG amplitude while CP equals 60 mmHg as PPG_60, ratio of these mean values can be defined. PPG_90_60 is defined as in (2).

$$PPG\_90\_60 = PPG\_90 / PPG\_60 \tag{2}$$

The defined ratio can help quantify the change in the blood flow rate arriving to the index finger as a result of the occlusion and elimination of the occlusion of the brachial artery.

D. *Effect of breathing*

Breathing influences BP and PWTT. Systolic pressure decreases during normal inspiration while PWTT increases. The effect of breathing makes the analysis of the dependence of PWTT on CP difficult. Burdened by this distortion, PWTT cannot be interpreted as patient specific diagnostic information, therefore, compensation of this effect is necessary. Respiratory oscillations are present in the ECG and the PPG as amplitude variations as well as frequency variations (Respiratory Sinus Arrhythmia) [6]. Breathing can be estimated based on ECG and PPG [6], [11].

E. *Stress level estimation*

Stress level of the tested person also influences BP. HRV is a recommended measure to determine momentary stress level. HRV characterizes the variation of the beat-to-beat time intervals [12]. In the frequency domain, dominant bands of HRV range from 3.3 mHz to 0.4 Hz. Some sources define a frequency band even below 3.3 mHz. Analysis of the lowest frequency band needs a resolution of at least 2 mHz, requiring a minimum 500-s long recording, during which HRV cannot be considered constant. For short time recordings, time domain analysis is required to characterize HRV. Here, usually the length of heart cycles (NN) is examined. pNN50 is a widely used parameter, it characterizes the differences in successive NN intervals. pNN50 is the ratio of differences exceeding 50 ms to the total number of differences [13]. We also calculated pNN0_20 and pNN20_50, which are the ratios of differences between 0 and 20 ms as well as 20 and 50 ms to the total number of differences, respectively [13]. The CP(t) protocol with 60-s constant pressure values makes a good estimation of the tested person's stress level possible. For the time interval where CP is equal to 90 mmHg the following notations are used: pNN0_20_90mmHg, pNN20_50_90mmHg, pNN50_90mmHg. For the time interval, where CP was equal to 60 mmHg the notation is similar.

IV. RESULTS

Changes of the PWTTHF, PWTTHC and PWTTCF values as a result of cuff occlusion are person specific, but not age-group specific. However, PPG amplitude exhibited a larger change as a result of changes in CP for healthy young persons than for healthy middle aged and senior individuals. Figure 3 shows the PPG signal as a function of time during the measurement for a healthy young adult (upper trace) and a healthy senior (lower trace). Figure 4 shows the values of the PWTTHC_90_60 and PPG_90_60 parameters for each tested person. It can be seen that healthy young adults can be separated from healthy middle aged and senior individuals based on PPG_90_60 values but not on PWTTHC_90_60 values.

For pNN parameters, neither mean values, nor ratio of mean values corresponding to intervals with constant CP are age-group specific. Figure 5 shows the values of the pNN50_60mmHg and pNN50_90mmHg parameters for each tested person. Separation of groups based on these parameters is not possible.

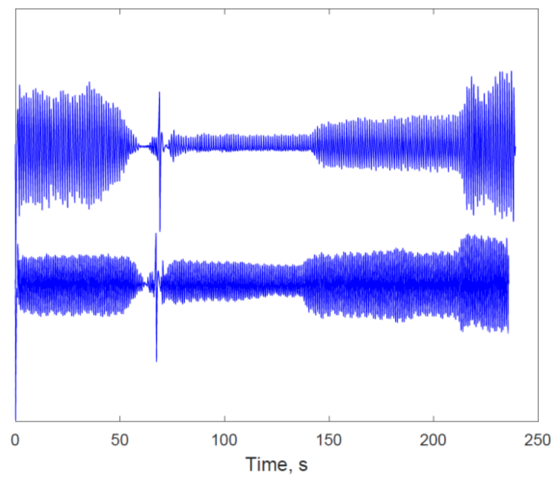


Fig. 3. PPG signal as a function of time during the measurement for a healthy young adult (upper signal) and for a healthy senior (lower signal).

V. DISCUSSION

The extent of the change in PPG amplitudes as a result of decreased CP from 90 mmHg to 60 mmHg was found to be larger for young adults than for middle aged and senior individuals. It can be supposed that this result indicates age related increase in the rigidity of arteries. However, it is difficult to determine whether the used PPG_90_60 parameter characterizes mainly the state of the brachial artery, or the microvascular system in the fingertip. The fact that the change in PWTTHF, PWTTHC and PWTTCF as a result of decreased CP from 90 mmHg to 60 mmHg do not correlate well with the PPG_90_60 parameter suggests that impaired vascular function during reactive hyperemia does not necessarily imply increased PWTT from the heart to the cuff or from the cuff to the fingertip.

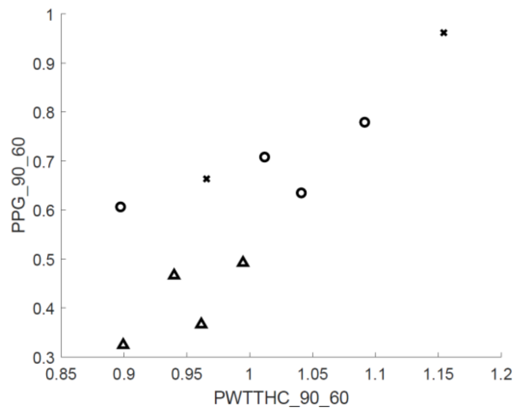


Fig. 4. PWTTHC_90_60 and PPG_90_60 parameters for each tested person. x: healthy seniors, o: healthy middle aged persons, Δ: healthy young adults.

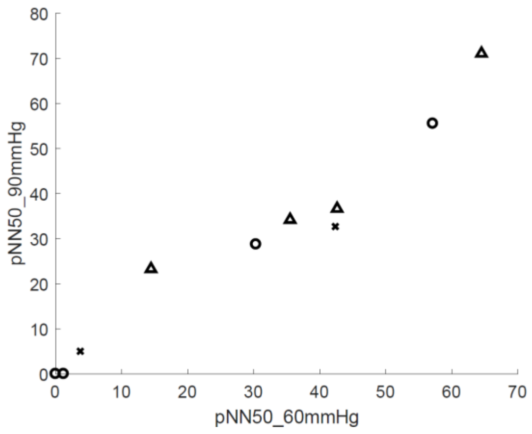


Fig. 5. pNN50_60mmHg and pNN50_90mmHg parameters for each tested person. x: healthy seniors, o: healthy middle aged persons, Δ: healthy young adults.

For the pNN50 parameter, decreased CP from 90 mmHg to 60 mmHg resulted in increased value of the parameter in case of 3 persons, decreased value in case of 6 persons and no change for 1 person. For pNN0_20 and pNN20_50 the direction of change as a result of decreased CP from 90 mmHg to 60 mmHg was also not the same for all persons. This result suggests, that the measured stress level of tested persons changed differently. This is in accordance with our previous experiences: a given stimulus can increase stress level of one individual while the same stimulus decreases stress level of another individual in the same experiment.

VI. Conclusion

Continuous change of CP during oscillometric BP measurement affects accuracy of the assessment of the state of the cardiovascular system in several ways. In this paper we described a BP measurement device that stops cuff deflation at 90 mmHg and 60 mmHg CP and records ECG, PPG and CP signals. We introduced parameters that characterize changes in PWTT, HRV and PPG amplitude values as a result of change in CP. Age-group specific differences were found only for the change in the PPG amplitude. This article presents that using ECG, PPG and CP, doctors can be provided with more information about the patient, than using conventional BP measurement. However, the measurement series, needed to evaluate this extra information has not been carried out yet. For the explanation of the observed phenomena and validation of preliminary results, we aim to organize a new measurement series with more participants in hospitals. BP measurement using the suggested CP(t) protocol not only gives a good estimate of systolic and diastolic pressures but also provides information on the rigidity of arteries. The device we have developed is meant for home health monitoring helping personalized health care.

References

[1] J. Kaczorowski, M. Dawes, M. Gelfer, C.D. Kapse, B.R. Patil, “Measurement of blood pressure: New developments and challenges”, British Columbia Medical Journal, vol. 54, no. 8, 2012, pp. 399-403.

[2] C.D. Kapse, B.R. Patil, “Auscultatory and Oscillometric methods of Blood pressure measurement: a Survey”, International Journal of Engineering Research and Applications, vol. 3, no. 2, 2013, pp. 528-533.

[3] A. Fonyó, Az orvosi élettan tankönyve, 5th ed., Medicina Könyvkiadó Rt., Budapest, 2011, pp. 306.

[4] D.J. Pitson, A. Sandell, R. van den Hout, J.R. Stradling, “Use of pulse transit time as a measure of inspiratory effort in patients with obstructive sleep apnoea”, European Respiratory Journal, vol. 8, no. 10, 1995, pp. 1669–1674.

[5] N.M. van Popele, W.J.W. Bos, N.A.M. de Beer, D.A.M. van der Kuip, A. Hofman, D.E. Grobbee, J.C.M. Witteman, “Arterial Stiffness as Underlying Mechanism of Disagreement Between an Oscillometric Blood Pressure Monitor and a Sphygmomanometer”, Hypertension vol. 36, no. 4, 2000, pp. 484-488.

[6] C. Ahlstrom, A. Johansson, T. Lanne, P. Ask, “A Respiration Monitor Based on Electrocardiographic and Photoplethysmographic Sensor Fusion” Proceedings of the 26th Annual International Conference of the IEEE EMBS, San Francisco, CA, USA, September 1-5, 2004, pp. 2311-2314.

[7] M. Ursino, C. Cristalli, “A mathematical study of some biomechanical factors affecting the oscillometric blood pressure measurement”, IEEE Transactions on Biomedical Engineering, vol. 43, no. 8, 1996, pp. 761-778.

[8] C. Ahlstrom, A. Johansson, T. Lanne, P. Ask, “Pulse wave transit time for monitoring respiration rate”, Medical and Biological Engineering and Computing, vol. 44, no. 6, 2006, pp. 471–478.

[9] M.Y.M. Wong, E. Pickwell-MacPherson, Y.T. Zhang, J.C.Y. Cheng, “The effects of pre-ejection period on post-exercise systolic blood pressure estimation using the pulse arrival time technique” European Journal of Applied Physiology, vol. 111, no. 1, 2011, pp. 135-44.

[10] J.T. Kuvin, A. Mammen, P. Mooney, A.A. Alsheikh-Ali, R.H. Karas, “Assessment of peripheral vascular endothelial function in the ambulatory setting”, Vascular Medicine, vol. 12, no. 1, 2007, pp. 13–16.

[11] A. Johansson, “Neural network for photoplethysmographic respiratory rate monitoring”, Medical and Biological Engineering and Computing, vol. 41, no. 3, 2003, pp. 242–248.

[12] A.J. Camm, et al., "Heart rate variability: standards of measurement, physiological interpretation, and clinical use. Task Force of the European Society of Cardiology and the North American Society of Pacing and Electrophysiology”, Circulation, vol. 93, no. 5, 1996, pp. 1043–1065.

[13] Á. Jobbágy, M. Majnár, L. K. Tóth, P. Nagy “HRV-based stress level assessment using very short recordings”, Periodica Polytechnica EECS, vol. 61, no. 3, 2017, pp. 238-245.

Towards the Verification of Neural Networks for Critical Cyber-Physical Systems

Gábor Rabatin[1], András Vörös[1,2]
[1]Budapest University of Technology and Economics, Department of Measurement and Information Systems, Budapest, Hungary
[2]MTA-BME Lendület Cyber-Physical Systems Research Group, Hungary
mail: rabigabor@gmail.com, vori@mit.bme.hu

***Abstract*—Smart technologies are emerging in the field of Cyber-Physical Systems (CPS) yielding new challenges for system engineers. Rigorous techniques are necessitated to ensure the correct and accident-free behaviour of critical CPS. Neural networks gain increasing popularity in CPS, and even critical functionalities may rely on neural network based solutions. In this paper, we overview the literature and summarize our experiences of a neural network verification algorithm used in an academic case-study.**

***Index Terms*—neural network, verification, CPS**

I. Introduction

Recent advances in the field of artificial intelligence and especially neural networks led to the wide-spread application of smart techniques in Cyber-Physical Systems (CPS). However, a large set of CPSs can be regarded as critical, which necessitates their correct behaviour to be ensured. Neural networks can approximate arbitrary functions by using the so-called teaching process: input-output pairs are provided and the neural network learns an approximation by using various techniques such as back-propagation and optimization techniques to strengthen its generalization ability. Neural networks are gaining increasing importance in critical CPS, such as autonomous vehicles, smart factories and autonomous robots. Ensuring the correct behaviour of critical CPS is essential, where both design-time [1] and also run-time analysis techniques can be used. Typical design time verification techniques are testing and formal verification, both are exploited for neural networks.

A. (Deep) Neural Networks

Over the last decade, Deep Neural Networks (DNNs) achieved an impressive break-through in supervised, unsupervised and also reinforcement learning. DNNs are used in various fields such as image, video and speech recognition and they achieved better results in classification than humans [2]. The progress in the effectiveness of DNNs led to their application in critical systems.

A DNN consists of an input layer, multiple hidden layers and an output layer. Each layer contains multiple neurons as shown on Fig. 1. A *neuron* is a small unit inside a DNN which applies an *activation function* (e.g. sigmoid, hyperbolic tangent and Rectified Linear Unit (ReLU)) on its inputs and it passes the result to other neurons.

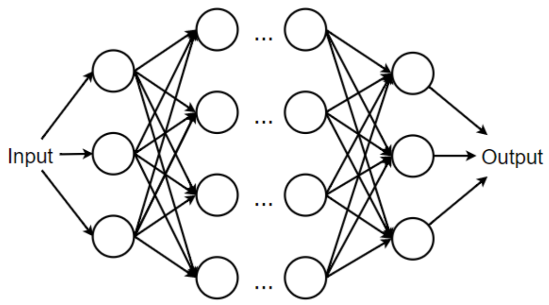


Fig. 1. Structure of a Multi-Layer Perceptron (MLP)

Each neuron has directed connections with neurons in the following layer. These connections have *weight parameters* which represent the strength of each connection. In supervised learning, these weights are learned during the training phase by minimizing a cost function over the training data. The most popular algorithm used for optimization is *gradient descent using backpropagation* [3]. The algorithm computes the gradients iteratively whereby if we changed our weights, then we could minimize the cost function over a given subset of training data.

Besides basic layers consisting only simple neurons, there are several other type of layers which have specific functions, like the convolutional layers which have been very popular recently, because it had reached significant process in image processing tasks.

An important weakness of DNNs is their sensitivity to adversarial examples. An extended definition of adversarial examples is proposed in [4] that captures all the important aspects, building on legal theory and the reasonable person test: a pair of inputs $x; x'$ is an adversarial example for a classifier, if a reasonable person would say they are of the same class but the classifier produces significantly different outputs.

Many techniques were developed to find adversarial examples for/against trained DNNs, where adversarial example synthetization algorithms may use the DNN as a white-box [5] or as a black-box [6].

B. Case-study: MoDeS3

Model-based Demonstrator for Smart and Safe Systems (MoDeS3) is used as a case-study in this paper. MoDeS3 is a demonstration system, where trains travel in a railway system controlled by the users. A camera-based monitoring system is installed to collect visual data and calculate the position information of the trains. If the monitoring system detects a dangerous situation, i.e. when the trains are too close to each other, then the safety subsystem stops the trains. A neural network is trained to detect the trains and estimate their positions. MoDeS3 demonstrates the usage of neural networks in a critical application.

C. Machine Learning Safety

The safety aspects of machine learning approaches are investigated in various papers [7], [8]. The application of AI techniques, and especially DNNs in safety application has to rely on various design time and runtime techniques [9]:

a) Redundancy and Dissimilarity: Redundant architectures are a common solution to improve the dependability of systems. Typically, some computing units (either replicated or dissimilar) calculates local results in parallel, and a voter then compares the different results and decides the final output based on the majority. Similar strategies can successfully be applied in ML.

b) Correct by Construction and Formal Verification: Correct-by-construction approaches heavily rely on formal methods to define and analyze the precise behavior of our systems.

c) Interpretability: It is highly desirable to understand the behavior of AI algorithms, and in our context, the behavior of DNNs both in design time and also at runtime.

II. Verification Techniques for Neural Networks

In the former section, we discussed the safety aspects of using AI technologies. In this section, we will only focus on various verification techniques, namely formal verification, testing and runtime verification.

A. Formal Verification

Formal verification is a technique to find a precise formal proof for the correctness of systems. Formal verification necessitates that both the property and also the system to be described in a formal (mathematically precise) way. Safety verification of DNNs is investigated in [10], where the authors propose a method to compute safe regions in a DNN by using an SMT[1] solver. They succeeded to do exhaustive search and find possible counter-examples or ensure that there cannot be one. Unfortunately, as the neural networks are getting deeper and deeper, the complexity of exhaustive search also increases.

A recent study shows an algorithm based on a modified *SMT solver* - the ReluPlex [11] - which can be used for formal verification on networks that are an order of magnitude larger than the largest networks verified using existing methods. The approach handles DNNs with a specific kind of activation function, called a Rectified Linear Unit (ReLU) and the verification algorithm searches out-of-bound behaviour of DNN-based controllers. The input of ReluPlex is a trained neural network, formally specified statements and a dataset, and it gives as output whether the statements can be satisfied.

[1]Satisfiability modulo theories

B. Testing

On the other hand, in most cases we cannot define a proper specification for the desired task (e.g. recognizing a lamp). Therefore formal verification cannot be used on them. In such cases we can have a separated validation set to determine whether the system generalizes well or overfits [12]. This works well if one can sample the distribution of the problem perfectly. If that does not occur then there is chance to have corner-cases where the model is not tested therefore can behave 'dangerously'. There are many recent studies which aim to develop new methods to ensure systematic testing with generating potential adversarial examples. Such studies are DeepTest [13], - which focuses on automated testing of autonomous cars by generating new inputs with the combination of different domain-specific image transformations - DeepSafe [14] - which tries to determine with data-guided clustering techniques whether *safe regions* within the network are robust against adversarial inputs - and DeepXplore [15] which will be discussed in this paper.

C. Runtime Verification and Monitoring

Runtime verification is a complementary technique when design time verification is not feasible. Monitoring the behaviour of neural networks relies on the traditional approaches used for runtime verification. In the MoDeS3 project, we introduced a monitoring framework to analyze the runtime behavior of the trained DNN responsible for detecting the objects. A Complex Event Processing (CEP) framework was used and a graph language helped us to specify the expected behavior of the system. The specification of the possible situations and also the simplified dynamics of the system was used to detect misclassification and other problems during runtime. The monitoring of DNNs used in real-life has to solve the challenge that typically those DNNs, which are used in CPSs, have to work in an evolving, formerly not fully known environment. This makes it difficult to define the monitored properties during design time.

III. Evaluation of DeepXplore on MoDeS3

Our main goal was to create a reliable system that can predict the positions of the trains in MoDeS3. This rather small demonstrator is used to evaluate and experiment with techniques that are used also by the industry in complex systems. The image recognition and classification task in the demonstrator would not require a complex DNN. However, we wanted to try out large networks to demonstrate the effectiveness of the verification algorithm on an illustrative example where the safety is critical. We used several image processing techniques and neural networks to predict the positions and we used some post-processing mechanisms to filter out the incorrect predictions. In this section, our algorithm and the used techniques are explained and also DeepXplore is introduced to enhance the accuracy of our predictions.

A. *Working of the Algorithm*

We tried all of the state-of-the-art algorithms in object detection with neural networks, e.g. FRCNN [16], YOLO [17], YOLO9000 [18], SSD [19]. We found that we could get higher reliability and accuracy if we used an algorithm that consists of image preprocessing techniques and a neural-network based classifier and not an end-to-end object detector.

We take advantage of that our task is to predict the train positions on a video and there is only a small difference between the frames, therefore we can use background substraction [20] to reduce the complexity of the problem. After that, we use dilation and erosion [21] to get smoother blobs and from these, we filtered out the not proper train candidates by shape and size. The resulting candidates are used as input for the classifier neural network. The output of the network can be one of the train types (*Taurus, SNCF, Red*) or *Other* (e.g. the table, railway line, decorative element or a human's hand). These results are examined whether they could occur and only the possible positions are kept.

B. *Trained Neural Networks*

Different neural network structures are tried for this classification task. We used two pretrained neural networks (InceptionV3 [22], ResNet [23]) via transfer learning [24] and a basic neural network (target network) with convolutional layers in it, similar to LeNet-5 [25]. The test set has been sampled from the original data set, and it has not been used during the training phase at all. The results are shown in Table I.

TABLE I
EVALUATION OF USED MODELS

Model	Accuracy on test set	Training time (sec/epochs)
InceptionV3	99.17%	300
ResNet	99.38%	600
Basic network	99.72%	180

Our experience shows that the training dataset should not contain too many elements, because after about 2000 images per classes there would be many repetitions (due to the limited number of states in the railway systems). To enhance the learning, we used augmentation to artificially create new samples with shearing, rotation and brightening. The results shown above are achieved by early stopping [26].

C. *Details of the Verification*

Our main goal was to develop a trustworthy AI-based demonstrator, so we tried one of most prevalent techniques to verify our system: we used the DeepXplore's approach for testing the recognition capabilities and robustness of our trained DNN. For this purpose, we used three models (showed above) that are different and have been trained on the same train-classifying problem. We used a *verification dataset* consisting of 600 input images different from the training data.

a) Neuron Coverage: The proportion of the neurons that have been activated[2] on a specific input image. The bigger this value is, the bigger part of the neural network is responsible for computing the output. The optimization tries to find an attack with high neuron coverage.

b) Verification: The first step of the verification process is to find misclassification of the input images (described on Fig 2). The different networks are fed by the same input and the algorithm compares their output.

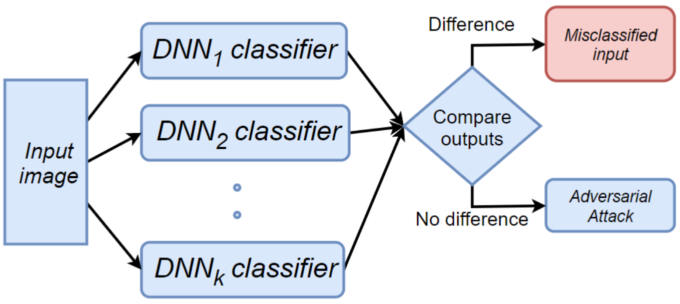


Fig. 2. Searching for misclassified images

If the outputs of the algorithms are the same, then the algorithm tries to modify the image and attack the target network.

c) Optimization Procedure: The joint objective can be described as depicted on Fig 3. the algorithm calculates the gradients of the used models on the images and this gradient value is used to alter the input image as follows. The objective for the modification of the images is to cause misclassification of (only) the target model. In addition, increasing the neuron coverage by the attack is the other optimization factor.

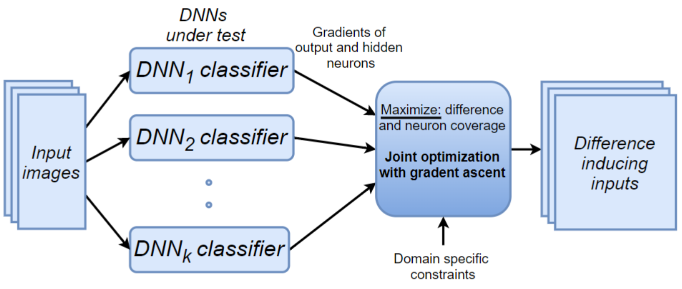

DNNs under test
Input images
DNN$_1$ classifier
DNN$_2$ classifier
DNN$_k$ classifier
Gradients of output and hidden neurons
Maximize: difference and neuron coverage
Joint optimization with gradent ascent
Domain specific constraints
Difference inducing inputs

Fig. 3. Adversarial attack (DeepXplore workflow) [15]

d) Realistic Modifications: It is important to have realistic outputs, which can be ensured by generating inputs that need to satisfy several domain-specific constraints. In this case three different types of constraints are used: (1) *brightening* effects are for simulating different light conditions, (2) *occlusion* effects are for simulating that the attacker potentially is blocking some parts of the camera, and (3) *blackout* effects

[2]having an output greater than a predefined value (zero in this case)

are for simulating dirts on the camera lens. In all the three cases it is ensured that the pixel values are between 0 and 255.

This way, we generated several examples which misled our neural networks, example images are shown on Fig 4.

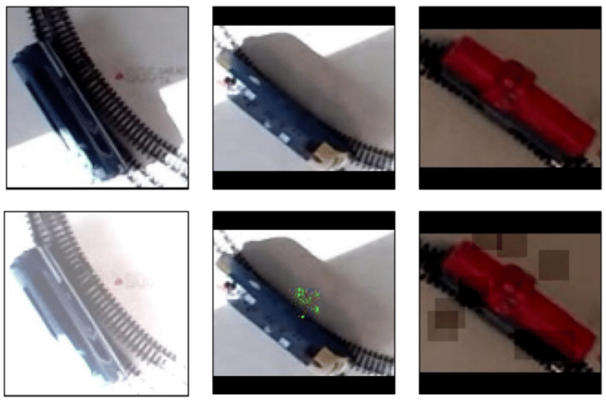


Fig. 4. First row: original images, second row: generated images by DeepXplore algorithm with *brightening, occlusion, blackout* image transformations (respectively)

D. Our Experiences

As we were able to generate a lot of adversarial images, we decided to use this technique as an augmentation technique to generate more training images. After that, we achieved *99.72%* test accuracy with the *Basic network*. We found that it is capable of not just helping to enhance the accuracy but finding the critical corner-cases for our neural network.

IV. Conclusion and Future Work

Our goal with this paper was to overview the current state of techniques for trustworthy DNNs. Various verification techniques exist for DNSs, such as testing, formal verification or monitoring. We chose the DeepXplore testing method and evaluated it on the MoDeS3 case-study.

In the future we plan to integrate the enhanced train-detector model in the MoDeS3 on a Jetson TX2 and experiment with formal verification techniques (e.g. ReluPlex) on a neural network based train controlling system. Besides, we want to develop adversarial transformations which are more similar to 'dangerous' corner-cases we experienced on MoDeS3 during open events.

Acknowledgment

This work was partially supported by the MTA-BME Lendület Cyber-Physical Systems Research Group and the ÚNKP-17-2-I. New National Excellence Program of the Ministry of Human Capacities.

References

[1] Tommaso Dreossi, Shromona Ghosh, Alberto L. Sangiovanni-Vincentelli, and Sanjit A. Seshia. Systematic testing of convolutional neural networks for autonomous driving. *CoRR*, abs/1708.03309, 2017.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015.

[3] Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016.

[4] Andrew Ilyas, Ajil Jalal, Eirini Asteri, Constantinos Daskalakis, and Alexandros G Dimakis. The robust manifold defense: Adversarial training using generative models. *arXiv preprint arXiv:1712.09196*, 2017.

[5] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[6] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against deep learning systems using adversarial examples. *arXiv preprint arXiv:1602.02697*, 2016.

[7] Rick Salay, Rodrigo Queiroz, and Krzysztof Czarnecki. An analysis of iso 26262: Using machine learning safely in automotive software. *arXiv preprint arXiv:1709.02435*, 2017.

[8] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*, 2016.

[9] CW Johnson. The increasing risks of risk assessment: On the rise of artificial intelligence and non-determinism in safety-critical systems. 2018.

[10] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural networks. In Rupak Majumdar and Viktor Kunčak, editors, *Computer Aided Verification*, pages 3–29, Cham, 2017. Springer International Publishing.

[11] Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. *CoRR*, abs/1702.01135, 2017.

[12] Igor V. Tetko, David J. Livingstone, and Alexander I. Luik. Neural network studies. 1. comparison of overfitting and overtraining. *Journal of Chemical Information and Computer Sciences*, 35(5):826–833, 1995.

[13] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. *CoRR*, abs/1708.08559, 2017.

[14] Divya Gopinath, Guy Katz, Corina S. Pasareanu, and Clark Barrett. Deepsafe: A data-driven approach for checking adversarial robustness in neural networks. *CoRR*, abs/1710.00486, 2017.

[15] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. Deepxplore: Automated whitebox testing of deep learning systems. *CoRR*, abs/1705.06640, 2017.

[16] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, June 2017.

[17] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[18] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242, 2016.

[19] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015.

[20] T. Bouwmans, F. El Baf, and B. Vachon. Background modeling using mixture of gaussians for foreground detection a survey. In *Recent Patents on Computer Science*, pages 219–237, 2008.

[21] Jean Serra. *Image Analysis and Mathematical Morphology*. Academic Press, Inc., Orlando, FL, USA, 1983.

[22] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[24] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *CoRR*, abs/1411.1792, 2014.

[25] Yann Lecun, Lon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[26] Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. On early stopping in gradient descent learning. *Constructive Approximation*, 26(2):289–315, Aug 2007.

Signal Acquisition for Accelerometer-based Fall Detection

Goran Šeketa[1],
[1]*University of Zagreb Faculty of Electrical Engineering & Computing*
Zagreb, Croatia
goran.seketa@fer.hr

Dominik Dzaja[1], Jurica Vugrin[1], Igor Lackovic[1], Ratko Magjarevic[1]
University of Zagreb Faculty of Electrical Engineering & Computing
Zagreb, Croatia

***Abstract*— Rapid growth of wireless technologies enabled the design of a multitude of wearable devices collecting data on individuals' health, behaviour and practices. All these acquisitions may be used for on-line decision making (medical or others) but can also be stored and saved for further use in analysis and research. Scientists welcome that process and request open but secure access to the datasets collected by wearables, however the ownership of the data is not clear in many cases, there is a lack of standardisation in the content and format of the acquired data and practical guidelines for dataset use are often missing. We have been confronted with the challenge of acquiring data for research on optimisation of fall detection algorithms. We present the acquisition procedure and the results of an experimental study with 16 subjects performing activities of daily living and of simulated falls. The subjects were wearing wireless sensor nodes with embedded triaxial accelerometer to track their movements. Different features from the accelerometer data were calculated and used for evaluating accuracy of fall detection.**

Keywords—wearables, wireless device, fall detection, acceleration

I. INTRODUCTION

In the early stage of their development, wearable sensors often were single devices monitoring only one health indicator, physiological parameter, fitness or another measurable parameter. Later such sensor nodes became integrated into closed systems for multiparameter monitoring and/or measurement. Integrated data was processed by a microprocessor built into one of the sensor nodes which had the appropriate embedded sensor. In such an architecture, the data followed a pathway from the equipment holder to a defined user (usually a health professional) through a defined communication channel, and there was an immediate action in case of emergency, or the data was stored and saved for later use. But in some cases, data was discarded after some time or even deleted, especially if it was considered not being medical data but just behavioural. With the global increase of storage capacity, opening of the "clouds", and the rapid development of tools to access and process large amounts of data from multiple sources – big data, the advantages of accessing and analysis options in health care were perceived not only for monitoring and diagnosis, but in prevention and prediction of diseases and/or unwanted events for health or in daily life. In a closed monitoring system, the protocols and the structure of the data are well defined and easy to use for the end user. Such configurations are preferred for clinical settings, e.g. in patients who just left the intensive care unit. Aging population, increase of prevalence and incidence of chronic diseases and the consequential burden to the budget of health care, led to change of health care policy: the patients should be serviced at their home if possible, again by introduction of appropriate technology, accepted by the user. The patient empowerment policy, defined by European Patients Forum as a "process that helps people gain control over their own lives and increases their capacity to act on issues that they themselves define as important" [1], certainly anticipates free choice of technology and monitoring devices ruled for the most part by user acceptance reasoning. The devices which individuals are voluntarily wearing for the purpose of monitoring one or more physiological, wellness, behavioural or other parameters are not any more parts of a closed integrated monitoring system, but bought "off the shelf" and widely equipped with connectivity. These devices may be considered as data sources from the Internet of Things or recently introduced, from the Internet of Medical Things. However, the generated repositories which contain high potential for research, are mainly not a public good but are available to companies mainly in IT sector.

Interest for research and development activities in fall detection have been frequently addressed in the projects financed by the EC, e.g. *Fallwatch, FATE* [3, 4] and they confirm the presence of the technology for fall detection on the market. However, literature reviews confirm that there is little standardised reporting on the sensors used [5]. Also, little of the datasets generated in the research has been made available to the research community [6], and there is a lack of common experimental procedure, guidelines for sensor placement resulting in large dissimilarity in the values and format of the data resulting in no agreement on referent values in activities of daily living (ADL) or for fall detection [7]. Our future work goal is to explore the possible causes of these disparities by analysing the existing publicly available datasets and by creating our own datasets in different conditions (e.g sensor placements, sensor ranges…).

Our recent research activities are associated with monitoring of health and providing preventive programs and IT tools for diabetic and cardiovascular patients as well as the elderly population in general. More details on the monitoring system and platform have been published earlier [2, 3]. Promoting physical activity, both for strengthening endurance and power, while enabling usage of the same monitoring devices for tracking of ADL, brought into consideration inclusion of fall detection as a common problem in the vulnerable elderly population. We have developed a wireless real-time health monitoring and alerting system which minimally interferes with the everyday life of the monitored person. The system enables generation of warnings upon detecting high risk for the health of the monitored person. The aim of the system is monitoring of persons at risk both at home and outside, in case they leave the home.

For the purpose of fall detection, motion sensors detect a fast change in position which may indicate a fall of the monitored person and the real challenge is to generate minimum number of false alarms while detecting all falls, i.e. to have both high sensitivity and high specificity in the fall detection. In threshold-based detection algorithms for fall detection, the choice of appropriate threshold values is crucial for the achievement of acceptable algorithm accuracy. Algorithm threshold values can thereby be set by analysing signals recorded from a test group of subjects performing ADL and simulated falls as we described in our previous work [9]. Procedures we used to acquire the signals for threshold selection and the results are further analysed in this paper.

II. Materials and Methods

A. Subjects and equipment

For the purpose of this study, 16 healthy volunteers were recruited, 11 males and 5 females. They performed ADL and simulated falls in three different fashions. They were informed on the procedures in the study and agreed with informed consent. The age of the subjects ranged from 15 to 44 (mean 23.1, std 6.7), height from 159 cm to 192 cm (mean 176.4, std 8.9) and body mass 55 kg to 93 kg (mean 75.2, std 12.8).

The subjects were carrying a Shimmer3 sensor node (ShimmerSensing, Dublin, Ireland) inertial measurement unit (IMU) attached to the right hip with a Velcro belt. The waist has been referred to be the optimal position for accelerometer based fall detection [8]. The IMU was fitted to the body with the axis oriented as shown in Figure 1.

Shimmer3 contains a triaxial accelerometer, triaxial gyroscope, triaxial magnetometer and an altimeter [10]. The sensor node has two types of accelerometers embedded: a wide range accelerometer with selectable measurement range from ±2g to ±16g and a low noise accelerometer with sensitivity ±2g and less noise). Wide range accelerometer was used for the measurements at the selected range of ±8g. Data from sensors were sampled with a rate of 204.8 Hz and streamed via Bluetooth to a PC.

B. Experiment protocol

Subjects were asked to perform 15 tasks. Twelve activities were considered: walking, fast walking, running, fast running, jumping, jumping over obstacles, sitting down, standing up from sitting position, lying down, getting up

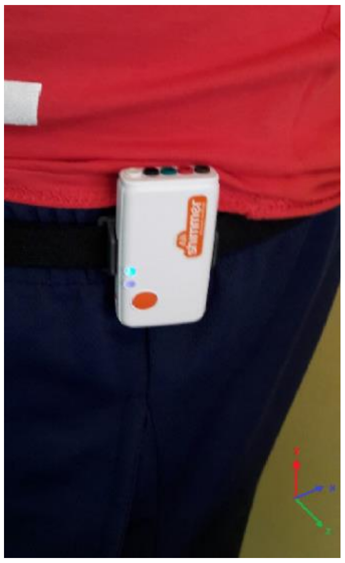


Figure 1. Sensor placement - Y axis was pointing up (vertical), X axis was pointing backward (antero-posterior plane) and Z axis was pointing away from the body (medio-lateral plane)

from lying position, walking down the stairs and walking up the stairs and they were followed by three simulated falls on a trainer. Three types of falls were simulated by falling on a 2 cm thick tatami mat: forward fall, sideways fall and backward fall. All activities were conducted in a safe laboratory environment as shown in Figure 2. Data from each subject was recorded during the sessions. Data was gathered and processed with a GUI implemented in Matlab.

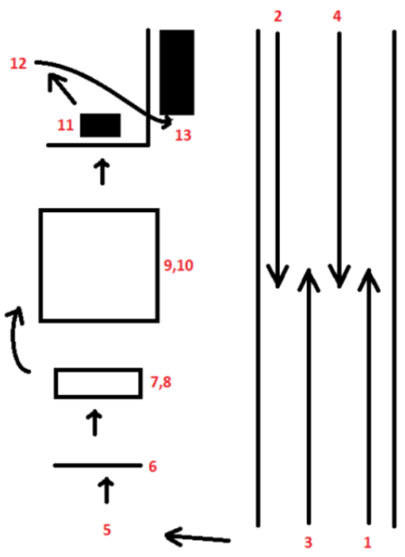


Figure 2. Plan of the laboratory setting where the ADL and simulated falls were performed in a row: 1) walking, 2) fast walking, 3) running, 4) fast running, 5) jumping, 6) jumping high, 7) sitting down, 8) standing up from sitting possition, 9) lying down, 10) getting up from lying position, 11) falling forward 12) falling sideways 13) falling back (walking up and down the stairs has been performed separately outside of the laboratory)

C. *Signal Delimitation*

All the signals were acquired by performing the actions of ADL and simulated falls in a row in order to speed up the measurement process. For the signals to be used for fall detection analysis, the signals had to be delimited in separate files containing only one activity or simulated fall. Matlab was used to manually delimit the signals by the researchers that conducted the measurements with subjects.

D. *Features*

From the acquired raw acceleration signals, we calculated the parameter *Acceleration Gravity Sum Vector Magnitude (AGSVM)* using the following equation according to [11]:

$$AGSVM(n) = \frac{\theta(n)}{90} \cdot ASVM(n)$$

where $Acceleration\ Sum\ Vector\ Magnitude\ (ASVM)$ is calculated by:

$$ASVM(n) = \sqrt{a_x^2(n) + a_y^2(n) + a_z^2(n)},$$

and the *Euler angle* (θ) between the vertical device axis and the direction of gravitational field is obtained from:

$$\theta(n) = \tan^{-1}\left(\frac{\sqrt{a_y^2(n) + a_z^2(n)}}{a_x^2}\right) \cdot \frac{180}{\pi}$$

where $a_x(n)$, $a_y(n)$, $a_z(n)$ represent *x*, *y* and *z* acceleration components of the *n*-th sample.

Each feature was calculated using accelerometer measurements from all subjects and experimental sessions. For the calculation of theta, a four-quadrant inverse tangent function was used that returns values in the closed interval [-π,π].

III. Results and Discussion

From all subjects, *AGSVM* values were calculated for every activity and simulated fall. From each signal recording that contained only one activity/fall, the maximal values of *AGSVM* were extracted and were used to plot the boxplot of the acquired dataset as shown in Figure 3.

The boxplot shows high differences in the distribution and large values of outliers. Though some median values from ADLs involving lower effort from the performers (e.g. 2, 5-8 and 10-12 in Fig.3) might be used for successful threshold based differentiation of those activities from falls (13 – 15 in Fig. 3), there is still overlapping of overlay values, e.g. 7 in Fig. 3. In ADLs requiring more effort (1, 3-4, 9), median values are close to those of the simulated falls. Large differences within the recorded group indicate the need for personalized determination of features to be used in fall detection and to use more complex threshold and machine learning algorithms.

The authors are aware of the limits of the experiments performed for this study. In order to learn more on the repeatability of recorded data for these activities within the group and deviations in data from a single subject, additional recording of ADLs and simulated falls are planned in the continuation of the study. In such a way we plan to obtain large datasets from younger population without any known dysfunctions and to build a well described database. In addition, we plan to organize structured acquisition of ADL and fall data from elderly population in order to obtain more realistic datasets.

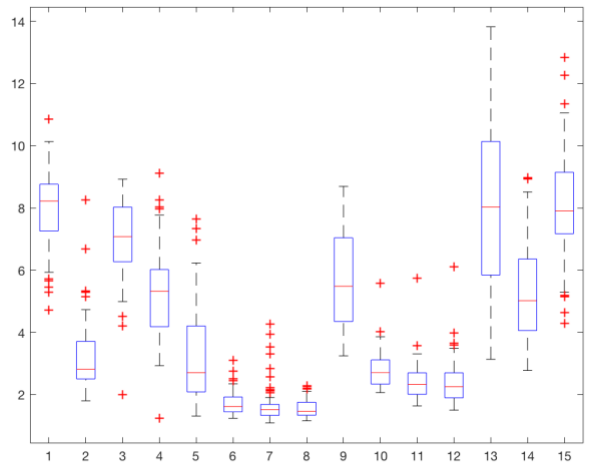


Figure 3. Boxplot of AGSVM values for falls and activities of daily living. Maximum whisker length is specified as 1 times the interquartile range. The vertical axis are values of the AGSVM parameter are in units of gravitational constant (g). Horizontal axis represents following activities/falls: 1-fast run, 2-fast walk, 3-high jump, 4-jump, 5-lie down, 6-stand up from lying, 7-sitting down, 8-standing up from sitting, 9-slow run, 10-walking up the stairs, 11-walking down the stairs, 12-slow walk, 13-fall on the back, 14-fall forward, 15-fall sideways

Acknowledgment

We thank all the subjects for their participation in the signal acquisition process and the Taekwondo club Cigra that provided the space and equipment necessary for the measurement process.

References

[1] http://www.eu-patient.eu/whatwedo/Policy/patient-empowerment/, accessed 22nd December 2017

[2] Šeketa G., Džaja D., Žulj S., Celić L., Lacković I., Magjarević R. (2015) Real-Time Evaluation of Repetitive Physical Exercise Using Orientation Estimation from Inertial and Magnetic Sensors. In: Jobbágy Á. (eds) First European Biomedical Engineering Conference for Young Investigators. IFMBE Proceedings, vol 50. Springer, Singapore

[3] Zulj S. et al. (2017) Supporting Diabetic Patients with a Remote Patient Monitoring Systems. In: Torres I., Bustamante J., Sierra D. (eds) VII Latin American Congress on Biomedical Engineering CLAIB 2016, Bucaramanga, Santander, Colombia, October 26th -28th, 2016. IFMBE Proceedings, vol 60. Springer, Singapore

[4] http://cordis.europa.eu/result/rcn/58267_en.html, accessed 22nd December 2017

[5] https://ec.europa.eu/digital-single-market/content/fate-monitoring-devices-helping-fall-detection, accessed 22nd December 2017

[6] Patel S, Park H, Bonato P et al (2012) A review of wearable sensors and systems with application in rehabilitation. J. Neuroeng. Rehabil, vol. 9, no. 1, pp. 21-38

[7] Casilari E, Santoyo-Ramón JA, Cano-García JM. Analysis of Public Datasets for Wearable Fall Detection Systems.Sensors (Basel). 2017 Jun 27;17(7). pii: E1513. doi: 10.3390/s17071513.

[8] Pannurat N, Thiemjarus S, Nantajeewarawat E (2014) Automatic Fall Monitoring: A Review. Sensors (Basel) vol. 14, issue 7, pp. 12900-12936

[9] Seketa G, Vugrin J, Lackovic I. (2017) Optimal Threshold Selection for Acceleration-based Fall Detection. 3rd International Conference on Biomedical and Health Informatics, ICBHI 2017; Thessaloniki; Greece; 18 November 2017 through 21 November 2017. IFMBE Proceedings Volume 66, 2018, Pages 151-155

[10] Shimmer Sensing Webpage, URL: www.shimmersensing.com, accessed: 15.9.2017

[11] N. H. Kim and Y. S. Yu, "Fall recognition algorithm using gravity-weighted 3-axis accelerometer data," Journal of theInstitute of Electronics and Information Engineers, vol. 50, no. 6, pp. 254–259, 2013.

High Frequency Active Distortion Cancellation

Balázs Varga, György Orosz
Budapest University of Technology and Economics
Department of Measurement and Information Systems
Budapest, Hungary
Email: bvarga92@gmail.com, orosz@mit.bme.hu

***Abstract*—This paper presents a signal processing method for cancelling spurious distortion products in high frequency bandpass signals. The proposed method uses a direct-conversion demodulator to transpose the signal to baseband, and a resonator-based observer to determine the harmonic content of the baseband signal. Based on the observed parameters, the algorithm generates an appropriate signal that is converted back to the carrier frequency and subtracted from the original signal, in order to suppress the distortion products. The method is illustrated by simulations and an experimental implementation.**

***Index Terms*—active distortion cancellation, IQ modulation, digital signal processing**

I. Introduction

Power amplifiers and other active electrical devices often exhibit nonlinearity, which causes spectral components that are not present in the input signal to appear in the output signal. These unwanted distortion products may adversely affect the operation in several applications where high spectral purity is required, such as wireless communications and high-precision instrumentation. In these cases, the effects of nonlinearities must be minimized so that the output signal can be as free of spurious spectral contents as possible.

Methods for compensating nonlinearities can be divided into two major categories: passive and active. *Passive* approaches utilize no on-line adaptation, the distortion compensation mechanism is designed along with the entire system, and is not changed during operation. The simplest example is filtering the signal with an analog or digital bandpass filter tuned to fixed frequencies. A more advanced approach involves the identification of the transfer function and the pre-distortion of the input signal according to the inverse of the identified function [1]. The obvious disadvantage of these passive methods is the necessary presupposition of long-term stability – if the transfer function changes during operation, the compensation will no longer work properly.

In *active* distortion cancellation, nonlinearities are compensated by a digital control loop which adds an appropriate signal to the input or the output of the system, suppressing unwanted frequency components, and making the resulting response appear linear [2]. With active distortion cancellation, it is not necessary to make assumptions about the nature of the nonlinearity at design time; the transfer function is adaptively identified and compensated during operation.

However, active methods are not without drawbacks either. Fig. 1 shows two possible arrangements of an active distortion cancelling system. The addition of the compensating signal

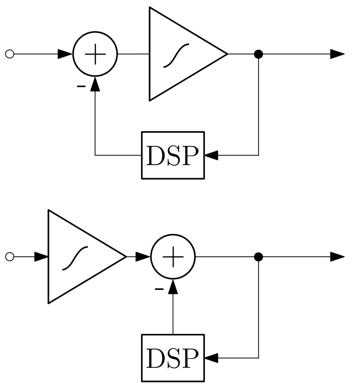


Fig. 1. Summation at input vs. output.

can be performed at the input or the output of the nonlinear system. In the former case, the nonlinear system is part of the control loop, which may have detrimental effects on the stability and dynamical properties. On the other hand, if the summation takes place at the output of a power amplifier, the large signal amplitudes may present a practical challenge [2].

This paper is structured as follows. Section II describes an algorithm for determining the amplitude and phase of the distortion products. In Section III, a digital controller is introduced that can be used in active distortion cancelling systems, and a method is proposed by which this controller can be extended to work on high frequency signals. The described methods are illustrated with numerical simulations in Section IV. Section V highlights some potential difficulties that may rise in practical implementations. In Section VI, an experimental implementation of high frequency active distortion cancellation is presented, along with measurement results. Finally, Section VII concludes the paper.

II. Detection of Distortion Products

Let us consider the nth sample of the signal y as a scalar product of a basis vector $\mathbf{c}_n^T$ and a state vector $\mathbf{x}_n$:

$$y_n = \mathbf{c}_n^T \mathbf{x}_n \tag{1}$$

In the case of the Discrete Fourier Transform (DFT), let $\mathbf{c}_n$ be the nth sample of the set of DFT basis functions:

$$\mathbf{c}_n = [c_{k,n}] = e^{j\omega_k n}, k = -L \dots L \tag{2}$$

$$\omega_{-k} = -\omega_k \tag{3}$$

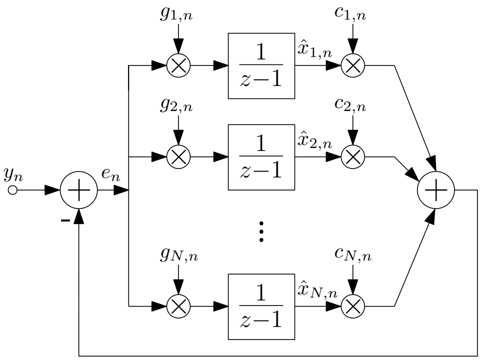


Fig. 2. The resonator-based observer.

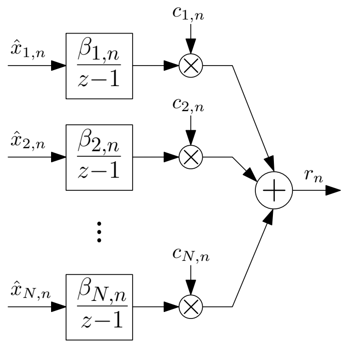


Fig. 3. Traditional controller.

The resonator-based observer (RBO) – introduced and thoroughly analysed in [3] and shown in Fig. 2 – is described by the following equations:

$$\hat{\mathbf{x}}_{n+1} = \hat{\mathbf{x}}_n + \mathbf{g}_n e_n \tag{4}$$

$$e_n = y_n - \mathbf{c}_n^T \hat{\mathbf{x}}_n \tag{5}$$

$$\mathbf{g}_n = [g_{k,n}] = \frac{1}{N}\overline{c}_{k,n} \tag{6}$$

where $\hat{\mathbf{x}}_n$ is the estimated state vector and $N = 2L + 1$. If all equations (1)-(6) are satisfied, the resonator poles are arranged uniformly on the unit circle, and the observer performs a Recursive Discrete Fourier Transform (RDFT) – the estimated state vector $\mathbf{x}_n$ contains the DFT coefficients of y, at frequencies corresponding to the resonator channels [2].

In practice however, usually not all resonator channels need to be realised. In this case, (6) becomes:

$$\mathbf{g}_n = [g_{k,n}] = \alpha\overline{c}_{k,n} \tag{7}$$

where $0 < \alpha < \frac{1}{N}$ is a common convergence parameter, chosen to be small enough to ensure stability. Furthermore, the fundamental frequency f_1 (ω_1) might change over time, or it might be unknown. In such cases, the resonator frequencies can be adaptively tuned to coincide with the harmonics of the input signal [4].

In an active distortion cancelling application, the resonator-based observer can be used to determine the amplitude and phase of the distortion products. The RBO has the following advantages over traditional DFT implementations:

- A current estimate of $\mathbf{x}$ is always available – as opposed to a blockwise DFT implementation.
- Only the necessary resonator channels need to be realised,
- which also allows faster computation.

III. Control Loop

A. Traditional control loop

When the RBO has reached steady state, the error signal is zero and the output follows the input. This feature can be utilized to construct a resonator-based controller, shown in Fig. 3 [2]. The controller contains resonator channels at frequencies that are to be cancelled. The inputs are the corresponding coefficients observed by the RBO. Therefore, if the controller output r_n is subtracted from the input signal y_n, the resulting signal will no longer contain components at the frequencies of the controller channels – assuming the algorithm converges.

In the case when there are only linear systems in the control loop, convergence can be ensured by the appropriate choice of parameters

$$\beta_{k,n} = \frac{\beta}{H(z_k)} \tag{8}$$

where β is a convergence parameter and $H(z_k)$ is the transfer function from the output of the controller to the input of the system, evaluated at the kth frequency [5]. $H(z_k)$ generally cannot be calculated, therefore it has to be measured prior to the beginning of operation. This can be done automatically: the controller generates a sinusoidal signal of the desired frequency, and the RBO observes the corresponding coefficient at the input [2].

When nonlinearities are present in the control loop, stability analysis becomes complicated. However, practical experiments show that the same linear approach works in the majority of these cases as well [2].

B. High frequency control loop

According to the sampling theorem, if a signal contains no frequencies higher than f_B, and it is sampled at a rate of at least $2f_B$, then it can be perfectly reconstructed from its samples. Unfortunately, for high frequency signals – such as those used in wireless communications – blindly applying this rule would necessitate very high sampling rates and computational requirements.

However, different approaches exist for signals with narrow bandwidth that are converted up to a high carrier frequency (also called bandpass, passband, non-baseband or narrowband signals) [6], [7]. In this paper, the method known as IQ modulation is described.

Let us consider the high frequency sinusoidal signal

$$y(t) = A\cos(\omega_c t + \varphi) = \frac{A}{2}\left[e^{j(\omega_c t+\varphi)} + e^{-j(\omega_c t+\varphi)}\right] \tag{9}$$

The spectrum of a real-valued signal always satisfies the Hermitian property – i.e. $Y(-\omega) = \overline{Y(\omega)}$. Multiplication by

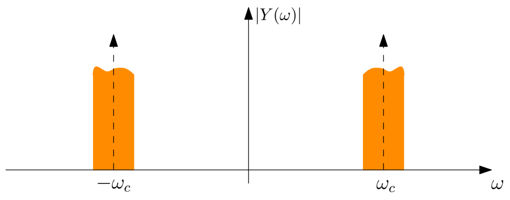


Fig. 4. Bandpass signal.

a complex exponential corresponds to the linear translation of the spectrum and yields a complex-valued signal:

$$\begin{aligned} v(t) &= y(t)e^{-j\omega_0 t} = \\ &= y(t)\cos(\omega_0 t) - jy(t)\sin(\omega_0 t) = \\ &= \frac{A}{2}e^{j((\omega_c-\omega_0)t+\varphi)} + \frac{A}{2}e^{-j((\omega_c+\omega_0)t+\varphi)} \end{aligned} \tag{10}$$

Naturally, a complex-valued signal cannot be realised, however, its real and imaginary part can individually exist as physical signals. If they are digitised separately, then the complex signal can be computationally constructed. Taking into account the anti-aliasing lowpass filters of the analog-to-digital converters (ADCs), and assuming that ω_0 is chosen to be sufficiently close to ω_c, the complex signal becomes

$$v_f(t) = \frac{A}{2}e^{j((\omega_c-\omega_0)t+\varphi)} \tag{11}$$

The filtered complex signal $v_f(t)$ still carries all amplitude and phase information of $y(t)$, but its spectrum has been shifted to the left by ω_0 and the resulting high-frequency component has been omitted. An RBO with a single resonator channel at $\omega_c - \omega_0$ can be used to determine the amplitude and phase of $v_f(t)$, and therefore also of $y(t)$.

The same principle can be used to generate high frequency signals. Let us consider the low frequency complex signal

$$w(t) = Ae^{j(\omega t+\varphi)} \tag{12}$$

The real and imaginary part of $w(t)$ can be individually generated and multiplied by other sinusoidal signals:

$$\begin{aligned} r(t) &= \mathrm{Re}\{w(t)\}\cos(\omega_0 t) - \mathrm{Im}\{w(t)\}\sin(\omega_0 t) = \\ &= A\cos(\omega t+\varphi)\cos(\omega_0 t) - A\sin(\omega t+\varphi)\sin(\omega_0 t) = \\ &= A\cos((\omega+\omega_0)t+\varphi) \end{aligned} \tag{13}$$

Similarly as before, the high frequency real signal $r(t)$ retained the amplitude and phase of its low frequency complex representation $w(t)$.

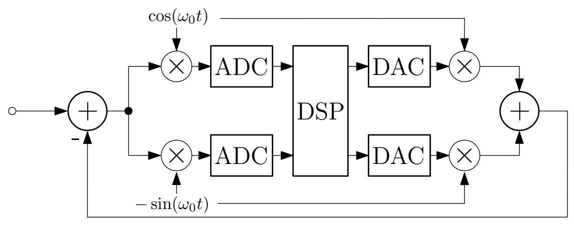


Fig. 5. High frequency controller.

Fig. 5 shows how the method of IQ modulation[1] can be used to extend the traditional distortion cancelling system to work on high frequency signals.

IV. Simulation Results

The high frequency distortion cancelling arrangement shown in Fig. 5 was simulated in MATLAB. The aim was to suppress a single sinusoidal signal with a frequency of $f_c = 110\,\mathrm{kHz}$. The local IQ signals were generated with $f_0 = 100\,\mathrm{kHz}$, resulting in $10\,\mathrm{kHz}$ baseband signals. The complex signal introduced in (11) was assembled and an RBO was applied to it. Then, the control loop was created according to Fig. 5.

The simulation results are shown in figures 6 and 7. The algorithm clearly converges and successfully generates an output signal ($\mathrm{RF_{out}}$) that cancels out the input signal ($\mathrm{RF_{in}}$).

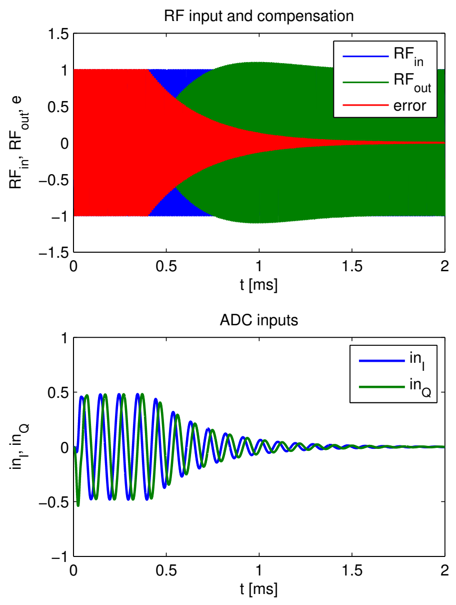


Fig. 6. High frequency (top) and downconverted signals (bottom) at the beginning of the cancellation. The controller is turned on at 0.4 ms.

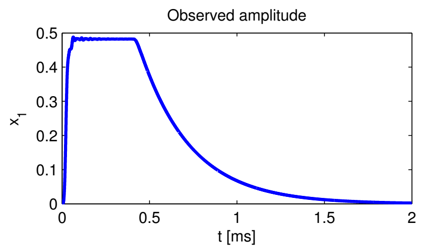


Fig. 7. Magnitude of the estimated state variable $\hat{x}_1$.

[1]The name IQ modulation originates from the fact that the local oscillator produces two signals with a 90 degree phase shift between them: one is often referred to as the in-phase (I) signal, and the other as the quadrature (Q) signal.

V. Implementation issues

Thus far, the in-phase and quadrature signal paths have been assumed perfectly identical. In reality however, imbalances such as offset and gain errors are present. Taking these into account on the downconversion (input) side, (10) becomes:

$$\begin{aligned}\tilde{v}_1(t) &= (y(t)+C)\cos(\omega_0 t) - j\,(y(t)+D)\sin(\omega_0 t) = \\ &= v(t) + C\cos(\omega_0 t) - jD\sin(\omega_0 t) \qquad (14)\\ \tilde{v}_2(t) &= y(t)\,(\cos(\omega_0 t)+C) - jy(t)\,(\sin(\omega_0 t)+D) = \\ &= v(t) + Cy(t) - jDy(t) \qquad (15)\\ \tilde{v}_3(t) &= (1+\epsilon)\,y(t)\cos(\omega_0 t) - jy(t)\sin(\omega_0 t) = \\ &= v(t) + \epsilon y(t)\cos(\omega_0 t) \qquad (16)\end{aligned}$$

As can be seen from (14) and (15), offset errors cause high frequency components to appear, which are of no particular concern, since these are eliminated by the lowpass filters of the ADCs. On the other hand, the gain error results in baseband components, which are sampled by the ADCs.

Similarly, on the upconversion (output) side, (13) becomes:

$$\begin{aligned}\tilde{r}_1(t) &= (\mathrm{Re}\{w(t)\}+C)\cos(\omega_0 t) - \\ &- (\mathrm{Im}\{w(t)\}+D)\sin(\omega_0 t) = \\ &= r(t) + C\cos(\omega_0 t) - D\sin(\omega_0 t) \qquad (17)\\ \tilde{r}_2(t) &= \mathrm{Re}\{w(t)\}(\cos(\omega_0 t)+C) - \\ &- \mathrm{Im}\{w(t)\}(\sin(\omega_0 t)+D) = \\ &= r(t) + AC\cos(\omega t+\varphi) - AD\sin(\omega t+\varphi) \qquad (18)\\ \tilde{r}_3(t) &= (1+\epsilon)\,\mathrm{Re}\{w(t)\}\cos(\omega_0 t) - \\ &- \mathrm{Im}\{w(t)\}\sin(\omega_0 t) = \\ &= \left(1+\frac{\epsilon}{2}\right) r(t) + \frac{A\epsilon}{2}\cos((\omega_0-\omega)t-\varphi) \qquad (19)\end{aligned}$$

In this case, all errors have significant effect on the output signal. A simple way to eliminate these effects is to numerically compensate the errors by the multiplication and addition of appropriate constants – which can be determined via measurement or heuristic (e.g. tune the parameters until the output spectrum is sufficiently pure). This approach proved to be adequate in the experiment presented in Section VI.

VI. Experimental Results

To experimentally illustrate the viability of the described concepts, a high frequency distortion cancelling system prototype was constructed that closely matches the diagram shown in Fig. 5. In our setup, all computation was done by an Analog Devices ADSP-21364 digital signal processor. For the two ADC and two DAC channels, an AD73322L audio codec was used. The IQ signal pair was generated by an AD9854 direct digital synthesizer (DDS) chip designed specifically for this application. The signal multiplications were carried out by four AD835 analog multipliers.

Similarly as in Section IV, the aim was to suppress a high frequency sinusoidal signal. In the measurements presented here, the frequency of the input signal was 101 kHz, and the local oscillator was set to 100 kHz, resulting in 1 kHz baseband signals, which were then sampled by the codec at 64 kHz.

Fig. 8 shows that over 50 dB suppression was achieved. In Fig. 9, the convergence of the algorithm can be observed.

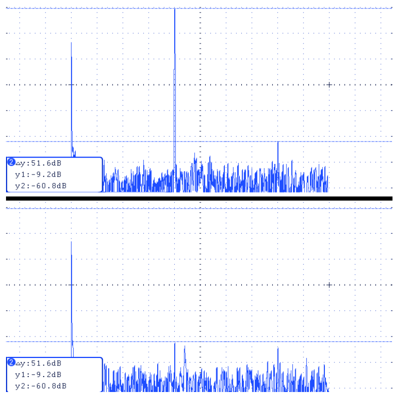


Fig. 8. Output spectrum without (top) and with (bottom) the controller in operation. Horizontal scale 25 kHz/div.

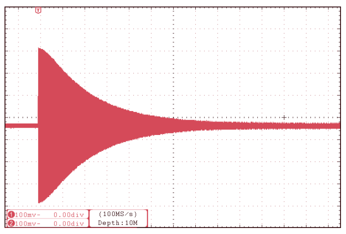


Fig. 9. Settling of the output after applying a 101 kHz signal to the input. Horizontal scale 5 ms/div. Settling time approximately 50 milliseconds.

VII. Conclusion

In this paper, the concept of a high frequency active distortion cancelling system was introduced. The mathematical background was explored and illustrated with simulations, as well as a working prototype. Some practical difficulties were examined and a possible solution was proposed. However, further research should focus on overcoming these issues.

References

[1] M. K. Nezami, "Fundamentals of power amplifier linearization using digital pre-distortion," High Frequency Electronics, pp. 54–59 , September 2004.

[2] L. Sujbert, B. Vargha, "Active distortion cancelation of sinusoidal sources," Proceedings of the IEEE Instrumentation and Measurement Technology Conference, May 2004, Como, Italy, pp. 322–326.

[3] G. Péceli, "A common structure for recursive discrete transforms," IEEE Transactions on Circuits and Systems, vol. CAS-33, pp. 1035–36, October 1986.

[4] F. Nagy, "Measurement of signal parameters using nonlinear observers," IEEE Transactions on Instrumentation and Measurement, vol. IM-41 no. 1, pp. 152–155, February 1992.

[5] L. Sujbert, G. Péceli, "Periodic noise cancelation using resonator based controller," 1997 Int. Symp. on Active Control of Sound and Vibration, ACTIVE 97, pp. 905–916, Budapest, Hungary, August 1997.

[6] R. G. Vaughan, N. L. Scott, D. R. White, "The theory of bandpass sampling," IEEE Transactions on Signal Processing, vol. 39 no. 9, pp. 1973–1984, September 1991.

[7] H. Zhou, Y. Zheng, "An efficient quadrature demodulator for medical ultrasound imaging,", Frontiers of Information Technology & Electronic Engineering, vol. 16 no. 4, pp. 301–310, April 2015.

Improved frequency response measurements using local parametric models

Dieter Verbeke
Department ELEC
Vrije Universiteit Brussel
Brussel, B-1050
Email: dieter.verbeke@vub.be

Johan Schoukens
Department ELEC
Vrije Universiteit Brussel
Brussel, B-1050
Email: johan.schoukens@vub.be

***Abstract*—The concept of local parametric modelling has sparked renewed attention in frequency response function (FRF) measurements. Essentially, these approaches assume a particular parametric structure and approximate the FRF and the leakage errors in a small window around the frequency of interest. Following the successful application of the idea in the local polynomial method (LPM), the local rational method (LRM) was developed. The smoothness of the transfer function in lowly damped systems can be insufficient for polynomial approximation to be effective, provoking us to consider rational functions instead. The power of the LRM has previously been demonstrated in both simulations and experiments. At the cost of increased computation, the LRM reduces the leakage errors with several orders of magnitude w.r.t. its alternatives while the sensitivity to disturbing noise remains comparable to that of classical methods.**

Keywords*—Frequency response function, local parametric modelling, polynomial functions, rational functions, approximation

I. Introduction

Measuring the FRF to characterize the dynamic behaviour of a linear time invariant (LTI) system is a well-studied problem. Spectral analysis methods based on correlation techniques have long been used as standard tools in engineering practice, see e.g. [1], [2]. Any non-parametric method suffers from leakage errors and noise errors. Leakage errors put a major restriction on the classical methods, even in the absence of measurement or process noise. For that reason the problem again attracted considerable attention in more recent years. With the so-called local polynomial method (LPM), the concept of local parametric modelling was introduced in [3], [4], [5]. Basically, LPM locally uses polynomial approximations of the transfer function and the transient behaviour of the system caused by finite data length and initial conditions effects. The idea was subsequently extended [6] replacing polynomial by rational approximants, resulting in what is referred to as the local rational method (LRM). The potential advantage of rational approximations is that they may behave better in the vicinity of poles, and, in particular, may be used to extrapolate or interpolate a function beyond poles that would block the convergence of a polynomial, see e.g. [7], [8]. In this paper we investigate rational approximations of a transfer function in an effort establish an error bound that relates tuning parameters to system properties. In this paper we focus on the LRM. We regard it as a particular instance of a more general local parametric approach. Inspired by the promising results in simulations [6] and in experiments [9] we discuss and compare the different local parametric methods.

II. Local modelling

The local parametric methods focus directly on the relation between input and output DFT spectra. Their essential advantage lies in their ability to dramatically reduce leakage errors. The central idea is to estimate the frequency response by considering a narrow interval around each frequency. Inside that interval the system and transient model can be represented by low order approximations. As such a reduced system identification problem is solved at each frequency of interest.

A. Problem formulation

The data consists of the N-point discrete Fourier transform (DFT) [10] spectra of input and output signals, possibly disturbed by additive noise

$$U(k) = U_0(k) \qquad (1)$$
$$Y(k) = Y_0(k) + N_Y(k). \qquad (2)$$

Inside the window $B = [k-n, k+n]$ the spectra are related according to

$$\begin{aligned} Y(k) &= G(k)U(k) + T_G(k) + \\ &\quad H(k)E(k) + T_H(k), \end{aligned} \qquad (3)$$

with

$$\begin{aligned} G(k) &= B_G(k)/A_G(k), \\ T(k) &= T_G(k) + T_H(k) = I(k)/A_G(k). \end{aligned} \qquad (4)$$

$T_G(\Omega_k)$ and $T_H(\Omega_k)$ are, respectively, the system and noise transient terms. These are rational functions of the generalized frequency variable, Ω_k, that share the same set of poles with respectively $G(\Omega_k)$ and $H(\Omega_k)$. Note that k and Ω_k as the arguments to a function $f(\cdot)$ are used interchangeably throughout the text. The transient terms cause leakage errors in the frequency response function measurements [4].
The unobserved (band-limited) white noise source is assumed to be independent and identically distributed (iid) at the sampling points, resulting in a DFT spectrum $E(k)$ with favourable properties (see [4]). The noise variance is given by

$$\sigma_V^2(k) = \text{Var}(V(k)) \text{ with } V(k) = H(\Omega)E(k) \qquad (5)$$

The goal is now to obtain a non-parametric estimate of the frequency response matrix $G(k)$ and the noise variance $\sigma_V^2(k)$ from the measured input and output spectra in the frequency band of interest. The major difficulty in obtaining accurate estimates is the suppression of the leakage terms.

Both transfer function and transient term in model (3-4) are estimated by minimizing the residuals $\mathcal{E}(\ell)$ inside the interval B for all $\ell = k + r$ with $r = -n, \ldots, 0, \ldots, n$

$$\mathcal{E}(\ell) \quad = \quad Y(l) - G(\ell, \theta_G)U(\ell) - T_G(\ell, \theta_{T_G}), \qquad (6)$$

using a weighted least-squares cost function:

$$V_c(k) = \sum_B W(\ell)|\mathcal{E}(\ell)|^2, \qquad (7)$$

that is minimized with respect to θ_G and θ_{T_G}, the model parameters. Under mild conditions $\sigma_Y^2(k+r) = \sigma_Y^2(k) + \mathcal{O}(r/N)$ ([4], Appendix B), so that we may reasonably assume that the output noise variance is constant in the interval B. In the following we revise several options for the weighting function $W(\ell)$ and justify our choice in the final algorithm.

B. Weighting the least-squares cost

a) Local polynomial method (LPM): $W(\ell) = 1$, $\hat{G} = B_{\hat{G}}$, $\hat{T} = I$. The nonlinear problem is transformed into a linear least-squares problem by setting the denominator equal to one $A_{\hat{G}}(\ell) = 1$. The minimizer of the simplified cost function

$$V_{LPM}(k) = \sum_B |Y(\ell) - B_{\hat{G}}(\ell)U(\ell) - I(\ell)|^2. \qquad (8)$$

is found by solving an overdetermined set of linear equations.

b) Iterative local rational method (ILRM): $W(\ell) = 1$, $\hat{G} = B_{\hat{G}}/A_{\hat{G}}$, $\hat{T} = I/A_{\hat{G}}$. Due to the presence of the denominator $A_{\hat{G}}(\ell)$, the objective function remains truly nonlinear:

$$V_{ILRM}(k) = \sum_B |Y(\ell) - \frac{B_{\hat{G}}(\ell)}{A_{\hat{G}}(\ell)}U(\ell) - \frac{I(\ell)}{A_{\hat{G}}(\ell)}|^2. \qquad (9)$$

The optimization problem must now be solved iteratively.

c) Local (linearized) rational method (LRM): $W(\ell) = |A_{\hat{G}}(\ell)|^2$, $\hat{G} = B_{\hat{G}}/A_{\hat{G}}$, $\hat{T} = I/A_{\hat{G}}$. The objective function is linearized by this particular choice of the weighting function, resulting in another linear least-squares problem, [6],[11]:

$$V_{LRM}(k) = \sum_B |A_{\hat{G}}(\ell)Y(\ell) - B_{\hat{G}}(\ell)U(\ell) - I(\ell)|^2. \qquad (10)$$

C. Comparison of properties

With a short discussion of the properties of the three outlined methods, we highlight the pertinence of the LRM.

a) Iterative local rational method: The ILRM appears to be the most straightforward, and the increased computational cost does not entail an unreasonable burden. A more critical issue is its high noise sensitivity. Excess poles and zeros can create large spikes in case of closely spaced poles and zeros. These artifacts can only be avoided by dedicated model tuning (at each single frequency). Obviously, this slashes the robustness of the overall process. The effect of the cost function on pole-zero cancellation is the subject of Section V.

b) Local polynomial method: The simplified approach using polynomial approximation proves to be very attractive. The identification problem is linear in the parameters, and it turns out that model selection is not critical. Clearly, a polynomial can only approximate a rational form adequately in a finite frequency window B. It was shown in [12] that in case of lowly damped systems the approximation error E_{LPM} is bounded by $(B_{LPM}/B_{3\mathrm{dB}})^{R+2}$. In this expression B_{LPM} is the local bandwidth of the LPM, R the degree of the local polynomial (assumed to be even), and $B_{3\mathrm{dB}}$ the 3-dB bandwidth around a resonance, a system property. Compared to the spectral windowing methods, a huge gain is made in the reduction of leakage errors. The length of the local window is subject to a trade-off [13]. On the one hand the interval should be as small as possible to reduce model errors. On the other hand it must contain enough frequencies to estimate all complex coefficients in the polynomial approximants. In addition, increasing the length of B reduces the sensitivity of the estimate to noise.

c) Local rational method: For systems with low damping, as often occurs in advanced mechanical applications (and other applications dealing with high resonances), the constraint $B \leq B_{3dB}$ may not be satisfied. The potential advantage of rational approximations is that they may behave better in the vicinity of poles, and, in particular, may be used to extrapolate or interpolate a function beyond poles that would block the convergence of a polynomial, see e.g. [7], [8].

III. A CLOSER VIEW ON THE LOCAL RATIONAL METHOD

The basic idea of the LRM is very simple. The frequency response G and the transient term T are smooth functions of the frequency. Furthermore, for a linear time invariant system, they have a rational form. As a result, they can be approximated in a narrow band around a particular frequency k by low-order complex rational functions. The complex parameters are estimated in a local band and the function value at the center frequency k is retrieved as the FRM at that point.

A. Basic equations

The crucial property used in local parametric approaches is that $G(\Omega_k)$, $H(\Omega_k)$, $T(\Omega_k)$ are smooth functions of the frequency. Returning to the local frequency variable $r = -n, \ldots, 0, \ldots, n$ the output spectrum at line $k + r$ can be written as:

$$Y(k+r) = G(k+r)U(k+r) + T(k+r) + V(k+r), \quad (11)$$

where $T(k) = T_G(k) + T_H(k)$ since there is no means of discriminating between system and noise transients at that level. $G(k)$ and $T(k)$ are assumed to be rational functions, consequently having continuous derivatives up to any order. Then $G(k+r)$ and $T(k+r)$ can be expanded at $k+r$ as:

$$G(k+r) \quad = \quad G(k) + \sum_{s=1}^{R} g_s(k)r^s + O((r/N)^{(R+1)}), \quad (12a)$$

$$T(k+r) \quad = \quad \sum_{s=1}^{R} t_s(k)r^s + N^{-\frac{1}{2}}O((r/N)^{(R+1)}). \quad (12b)$$

This expression originates from applying Taylor's formula with remainder to (11) and is the basis for the LPM. The remainders are, respectively, the system interpolation error and the sum of the residual system and noise leakage errors.

In the LRM the above polynomial approximation is substituted for a (linearized) rational approximation $\hat{G}(k) = B(k)/A(k)$. Note that the subscript $\hat{G}$ is dropped to unburden the notation. Neglecting the remainders results in

$$\begin{aligned} Y(k+r) &= \frac{B(k+r)}{A(k+r)}U(k+r) + \\ & \quad \frac{I(k+r)}{A(k+r)} + V(k+r) \end{aligned} \tag{13}$$

where $\mathcal{X}(k+r) = \mathcal{X}(k) + \sum_{s=1}^{R} x_s(k) r^s$ with $\mathcal{X} = I, B, A$ and $x = i, b, a$ denote the polynomials corresponding to the transient term, and the transfer function's denominator and numerator. Note that for the expansion of the system transfer function numerator and denominator, as well as the leakage term, different orders are allowed. However, it turns out that if an R-th order approximation is suitable for $\hat{G}$ (effectively A), it is also appropriate for $\hat{T}$. The reason is that G and T have the same poles, and that the system leakage term T_G is dominant with respect to the noise leakage term T_H.

B. Linearized least squares

Rearranging (13) (without the noise contribution) leads to

$$A(k+r)Y(k+r) = B(k+r)U(k+r) + I(k+r). \tag{14}$$

In order to avoid the trivial solution $\Theta = 0$, $A(k)$ is set equal to one. With this additional assumption the equation becomes:

$$\begin{aligned} Y(k+r) &= B(k+r)U(k+r) + I(k+r) \\ & \quad -\tilde{A}Y(k+r) + \tilde{V}(k+r) \\ &= \Theta K(k+r) + \tilde{V}(k+r), \end{aligned} \tag{15}$$

where we have introduced $\tilde{A}(k+r) = A(k+r) - 1$, and $\tilde{V} = A(k+r)V(k+r)$. Θ is the $1 \times (3R+2)$ vector of the unknown complex parameters

$$\begin{aligned} \Theta = \; & [b_0(k)\, b_1(k)\, b_2(k) \, \ldots \, n_R(k) \ldots \\ & i_0(k)\, i_1(k)\, i_2(k) \, \ldots \, i_R(k) \ldots \\ & a_1(k)\, a_2(k) \, \ldots \, a_R(k) \,] \end{aligned} \tag{16}$$

$K(k+r)$ is a vector of size $(3R+2) \times 1$:

$$K(k+r) = \begin{bmatrix} \kappa(r,\beta)U(k+r) \\ \kappa(r,\iota) \\ -\kappa(r,\alpha)Y(k+r) \end{bmatrix} \tag{17}$$

where $\beta, \iota = [0\,1 \ldots R]^T$ and $\alpha = [1\,2 \ldots R]^T$. $\kappa(r,\chi)$ is defined as

$$\kappa(r,\chi) = r^{\chi} \in \mathbf{Z}^{n_\chi \times (2n+1)}, \tag{18}$$

with n_X the number of elements of $\underline{X}$. Collecting (15) for $r = -n, -n+1, \ldots, 0, \ldots, n-1, n$ gives

$$Y_n = \Theta K_n + \tilde{V}_n \tag{19}$$

where Y_n, K_n and V_n are, respectively, $1 \times (2n+1)$, $(3R + 2 \times (2n+1)$ and $1 \times (2n+1)$ matrices

$$X_n = (X(k-n)X(k-n+1) \ldots X(k) \ldots X(k+n)) \tag{20}$$

with $X = Y,\, K,\, \tilde{V}$.
When $2n+1 \geq 2(R+1)+R$. Eq. (19) is an overdetermined set of equations that can be solved in the least squares sense as

$$\hat{\Theta} = Y_n K_n^H (K_n K_n^H)^{-1}. \tag{21}$$

x^H denotes the Hermitian transpose of x.

The local rational estimate of the frequency response function and the transient term at frequency k is then obtained via

$$\hat{G}(k) = \hat{\Theta}\,(1\,\mathbf{0})^H = \hat{\Theta}_{[1]} \tag{22}$$

$$\hat{I}(k) = \hat{\Theta}\,(\mathbf{0}\,1\,\mathbf{0})^H = \hat{\Theta}_{[(R+1)+1]} \tag{23}$$

where $X_{[1,j]}$ selects the j-th element of X. Because the DFT lines $k+r$, for $r = -n, n-n+1, \ldots, n$ are used for estimating the FRF at k, $\hat{G}(k)$ is correlated with $\hat{G}(k+r)$ for $r = -n, -n+1, \ldots 0, \ldots, n$.

IV. Approximation error

When the system under test exhibits strongly resonant behaviour, the LRM performs remarkably better than the classical spectral analysis techniques, and even, the LPM. Figure 1 supports that claim. Given that for a certain class of systems the interpolation errors dominate stochastic errors, it is of prime importance to understand the source of approximation errors induced by rational approximation.

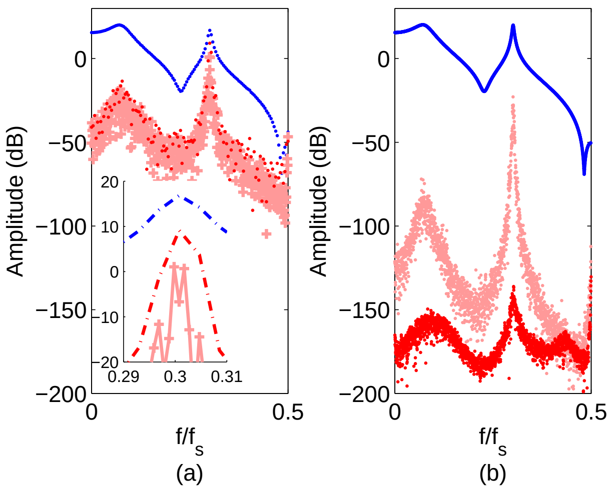


Fig. 1. A comparison of errors in FRF measurements for a simple numerical example using undisturbed data. A system composed of a highly and a lowly damped resonance is excited with filtered white noise (bandwidth of $0.4 f_s$) The full record contains 4096 samples. Figure (a) shows the results for a spectral analysis method with Hann window applied on subrecords of length $N = 256$ (red) and $N = 1024$ (pink) samples, both with an overlap of $R = 2/3 \times N$. In the inset of (a) a zoom around the second resonance is given. The LPM (pink) and LRM (red) in figure (b) is applied to the full length record. Observe that the errors of the LPM/LRM are an order of magnitude smaller than those of the Hanning method. The LRM outperforms the LPM.

V. Pole-zero cancellation

In Section II-C we mentioned that closely spaced zeros and poles can create large spikes in the ILRM. We used this argument to reject the ILRM in favour of the LRM. Here we support that observation with a simplified analysis of their respective cost functions. We consider the simplest transfer function one can imagine: a SISO transfer function with one pole-zero pair that cancels itself. Then we add a disturbance on the zero and study the effect on the respective cost functions.

A. *Simplified analysis*

We consider the simplest transfer function one can imagine: a SISO transfer function with one pole-zero pair that cancels itself. Then we add a disturbance on the zero and study the effect on the respective cost functions.

$$\frac{B}{A} = \frac{1+bs}{1+as} \quad \text{with} \quad \begin{cases} b & = (d+\epsilon)j \\ a & = (d+0)j \end{cases} \tag{24}$$

In case of the ILRM we work with the cost function

$$E_{ILRM} = Y - \frac{B}{A}U, \tag{25}$$

and find that the perturbed transfer function is equal to

$$\frac{1+djs+\epsilon js}{1+djs} = 1 + \frac{\epsilon js}{1+djs}. \tag{26}$$

Evaluating the above in $s = j(1/d + \Delta)$ gives an error expression

$$E_{ILRM} = \epsilon \frac{1/d+\Delta}{\Delta} U \tag{27}$$

In case of the LRM the error function has the form

$$E_{LRM} = AY - BU. \tag{28}$$

Evaluating again at $s = j(1/d+\Delta)$, this leads to an error term

$$E_{LRM} = \epsilon(1/d+\Delta)U \tag{29}$$

Summing over all neighbouring frequencies $r = -n, \ldots, 0, \ldots, n$ we get

$$\sum_r E^2_{ILRM,r} = \sum_r \epsilon^2 \frac{|1/d+\Delta_r|^2}{\Delta_r^2}|U|^2 \tag{30}$$

$$\sum_r E^2_{LRM,r} = \sum_r \epsilon^2 (1/d+\Delta_r)^2 |U|^2 \tag{31}$$

Figure 2 compares the error terms (30) and (31). From this graphical display it becomes clear that the ILRM is much more sensitivity to residual effects due to closely spaced poles and zeros.

VI. Conclusion

In this contribution we introduce the local parametric modelling approaches to FRF estimation from the point of view of the cost function in the subordinate identification problems. We investigate the errors due to rational approximation and consider the effect of pole-zero cancellation. We find that rational functions can suppress approximation errors, consequently lowering the requirements on frequency resolution, shortening measurement time, while simultaneously being robust to perturbations.

Acknowledgment

This work was supported in part by the Fund for Scientific Research (FWO-Vlaanderen), and by the ERC advanced grant SNLSID, under contract 320378. Dieter Verbeke is supported by an SB-grant of the FWO.

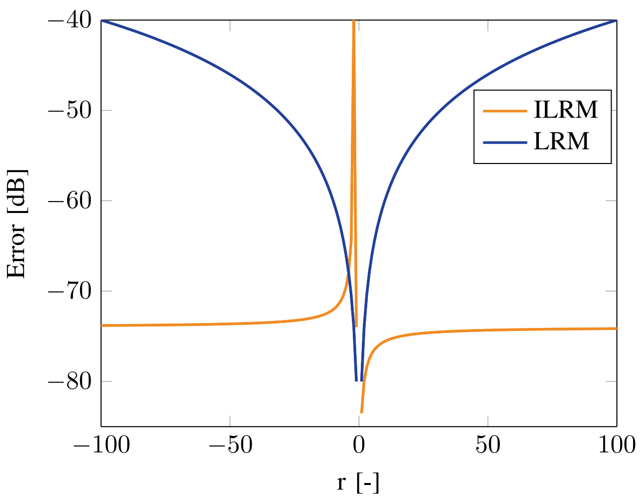


Fig. 2. Pole-zero cancellation and the effect of small perturbations on the error function. In this numerical example $c = 0.01$, $\epsilon = 0.0001$ and $n = 100$.

References

[1] J. Bendat and A. Piersol, *Engineering applications of correlation and spectral analysis*. Wiley, 1980.

[2] K. Godfrey, "Correlation methods," *Automatica*, vol. 16, no. 5, pp. 527 – 534, 1980.

[3] J. Schoukens, G. Vandersteen, K. Barbé, and R. Pintelon, "Nonparametric preprocessing in system identification: A powerful tool," *European Journal of Control*, vol. 15, no. 3-4, pp. 260–274, 2009.

[4] R. Pintelon, J. Schoukens, G. Vandersteen, and K. Barbé, "Estimation of nonparametric noise and FRF models for multivariable systems—Part I: Theory," *Mechanical Systems and Signal Processing*, vol. 24, no. 3, pp. 573 – 595, 2010.

[5] ——, "Estimation of nonparametric noise and FRF models for multivariable systems—Part II: Extensions, applications," *Mechanical Systems and Signal Processing*, vol. 24, no. 3, pp. 596 – 616, 2010.

[6] T. McKelvey and G. Guérin, "Non-parametric frequency response estimation using a local rational model1," *IFAC Proceedings Volumes*, vol. 45, no. 16, pp. 49 – 54, 2012.

[7] P. Gonnet, "Robust rational interpolation and least-squares," *Electronic Transactions on Numerical Analysis*, vol. 38, pp. 146–167, 2011.

[8] W. Van Assche, "Padé and Hermite-Padé approximation and orthogonality," *Surveys in Approximation Theory*, vol. 2, pp. 61–91, 2006.

[9] D. Verbeke and J. Schoukens, "Improved nonparametric identification of lightly-damped mechanical multiple-input multiple-output systems," *Proceedings of the International Conference on Noise and Vibration (ISMA 2016)*, pp. 2983 – 2993, 2016.

[10] R. Pintelon and J. Schoukens, *System identification: a frequency domain approach*. John Wiley & Sons, 2012.

[11] E. Levy, "Complex-curve fitting," *IRE transactions on automatic control*, no. 1, pp. 37–43, 1959.

[12] J. Schoukens, G. Vandersteen, R. Pintelon, Z. Emedi, and Y. Rolain, "Bounding the polynomial approximation errors of frequency response functions," *IEEE Transactions on Instrumentation and Measurement*, vol. 62, no. 5, pp. 1346–1353, 2013.

[13] P. Thummala and J. Schoukens, "Estimation of the frf through the improved local bandwidth selection in the local polynomial method," *IEEE Transactions on Instrumentation and Measurement*, vol. 61, no. 10, pp. 2833–2843, Oct 2012.