Experimental pipeline definition for surface high-density electromyogram (HDEMG) processing

Matjaž Divjak, Aleš Zamuda

University of Maribor
Faculty of Electrical Engineering and Computer Science, Koroška cesta 46, 2000 Maribor, Slovenija
E-mail: matjaz.divjak@um.si, ales.zamuda@um.si

Definicija eksperimentalnega cevovoda za obdelavo večkanalnih površinskih elektromiogramov (HDEMG)

Večkanalni površinski elektromiogrami (HDEMG) so trenutno eden najbolj uporabljanih postopkov za zajem in analizo EMG signalov v raziskovalne namene. Zaradi velike količine podatkov so dosti bolj zahtevni za obdelavo kot klasični enokanalni EMG signali, zato bi želeli uporabiti čimveč paralelne obdelave podatkov, kot jo omogočajo moderni procesorji, grafične kartice in sistemi HPC. Toda, implementacija takšnih algoritmov je zelo kompleksna in zahteva obilico specializiranega znanja. Zato je zelo zanimiv evropski projekt DAPHNE (Integrated Data Analysis Pipelines for Large-Scale Data Management, HPC, and Machine Learning), v okviru katerega razvijajo rešitve za lažji razvoj visoko zmogljvih algoritmov za obdelavo podatkov, ki učinkovito izkoriščajo moderno strojno opremo, kot so večjedrni procesorji, grafične kartice, namenski FPGA procesorji, sistemi HPC itd.

V tem članku smo preverili trenutno funkcionalnost domensko specifičnega jezika (DSL) DaphneDSL na primeru preprostega cevovoda z matričnimi operacijami, ki so pogoste pri obdelavi HDEMG signalov. Prvi testi kažejo, da je jezik DaphneDSL zelo zanimiva opcija za obdelavo znanstvenih podatkov in da jo lahko učinkovito uporabimo za implementacijo cevovodov za obdelavo HDEMG. Ker je jezik še v razvoju in je vseboval nekaj omejitev in hroščev, ki so bili kasneje sicer hitro popravljeni, takrat še ni bil primeren za obdelavo večjih količin podatkov in za bolj kompleksne cevovode. Pričakujemo, da bo nadaljnji razvoj te omejitve še naprej odpravil, zato bomo nadaljevali s testiranjem in evaluacijo zmogljivosti.

Abstract

High-density surface electromiography (HDEMG) is currently one of the most popular approaches for acquisition and analysis of EMG signals for research purposes. Due to large volume of data such analysis is much more time consuming than classic single-channel EMG and could benefit greatly from parallel processing capabilities of modern computing hardware. Unfortunately, the implementation of such algorithms is fairly complex and requires specialist knowledge. Therefore we are very interested in the Horizon 2020 project DAPHNE (Integrated Data Analysis Pipelines for Large-Scale Data Management, HPC, and Machine Learning), which aims to develop solutions for easier implementation of highly performant data processing algorithms that effectively leverage capabilities of modern hardware, such as multicore CPUs, GPUs, dedicated FPGAs, HPC systems etc.

In this paper we investigate the current functionality of DAPHNE domain specific language (DaphneDSL) on an example of a simple data processing pipeline consisting of matrix operations that are typical for HDEMG processing. Initial results show that DaphneDSL is an interesting option for scientific data processing and that it can be effectively used for implementation of HDEMG processing pipelines. However, since the language is still in development and had some limitations and bugs that were quickly fixed later, it was not yet suitable for large data loads and more complex pipelines. We expect that further development will alleviate those limitations and we will continue with testing and performance evaluation.

1 Introduction

High-density surface electromiography (HDEMG) is currently one of the most popular methods for non-invasive acquisition of muscle activity. Two decades of significant improvements in acquisition systems brought along also the development of decomposition algorithms that decompose the EMG signals into contributions of individual muscle motor units, enabling identification of motor unit firing times [1][2][3][4][5]. The use of this methodology has greatly advanced our understanding of motor control in health [6] and disease [7][8] and enabled numerous new in-vivo neurophysiological studies and insights into the workings of the human motor system [9].

However, such data processing pipelines presents a significant computing challenge and require substantial computing power. This makes it a good candidate for high performance parallel data processing on modern CPUs, GPUs, or HPC clusters. Unfortunately, coding such algorithms can be very complex and time consuming and requires specialist knowledge. Most researchers working in the field of EMG signal processing use Matlab of Python and do not have enough knowledge and experience to develop code for high-performance computing.

The DAPHNE project is funded by the European Union's Horizon 2020 research and innovation program under grant agreement number 957407 from 12/2020 through 11/2024.

The Horizon 2020 project DAPHNE (Integrated Data Analysis Pipelines for Large-Scale Data Management, HPC, and Machine Learning) offers an interesting solution for this problem of modern programming with latest mainstream hardware [10]. As vendor companies like Intel [11] are also among project partners and as there is a vibrant community of experts in Computer Architecture involved in the project, one of aims of the DAPHNE system is to ease the development of highly performing algorithms, computed on various deployment hardware, and to also increase the accessibility of the DAPHNE runtime [12] and Machine Learning [13]. Considering that DAPHNE efficiently accesses threading, multiple CPUs, GPU kernels, FPGA devices, and distributed nodes using various modes of communication, the DaphneDSL [14] high-level domain specific language (DSL) is chosen in this paper as a welcomed approach to ease data pipeline programming [15], optimization, and development of data science methods like HDEMG processing. The DaphneDSL syntax [14] is inspired by C/Java-like languages and the DSL is a case-sensitive language inspired by ML systems as well as languages and libraries for numerical computation like Julia, Python NumPy, R and Apache SystemDS DML [16], but also by compiler hints for data/operator placement (i. e. local/distributed, CPU / GPU / FPGA, computational storage, still in experimental state and not guaranteed by the compiler) [14]. The decision to choose a HPC DSL is substantiated further especially after considering that University of Maribor is part of SLING (Slovenian National Supercomputing Network), the coordination body of various HPCs in Slovenia, including the first EuroHPC supercomputer [17], the Vega with 240 A100 GPUs and 122 thousand CPU cores [18]. On Vega, it has already been demonstrated that DAPHNE can run applications with iterated and randomised algorithms for optimisation [15] and already identified earlier as an opportunity for scaling parallel structures [19].

In this paper we present initial investigation and tests of the DAPHNE system for processing HDEMG signals and working with large matrices. We prepared a simple data processing pipeline that demonstrates a few realistic matrix operations that are often used in EMG processing, such as concatenation, extension, and calculation of correlation matrix. We implemented those operation using the DaphneDSL library [14] and checked how the compiler generates the intermediate representation calls, verified the correctness of the results and measured the required run time.

In the next section, the methods are described, then the results are provided in the third section, and the conclusion as the fourth section, followed by references.

2 Methods

High density surface EMG signals are recorded with special flexible electrodes consisting of multiple channels (Figure 1). Typically, 32, 64 or more channels are used. Each channel is stored as one row in a large data matrix, so 64 channels results in 64 rows. The number of columns depends on the recording length and sampling frequency: typically we use 4096 Hz sampling frequency which results in 245,760 samples (columns) per one minute of recording, stored in double format. This results in fairly large matrices, which can easily reach 10 - 30 GB in size.

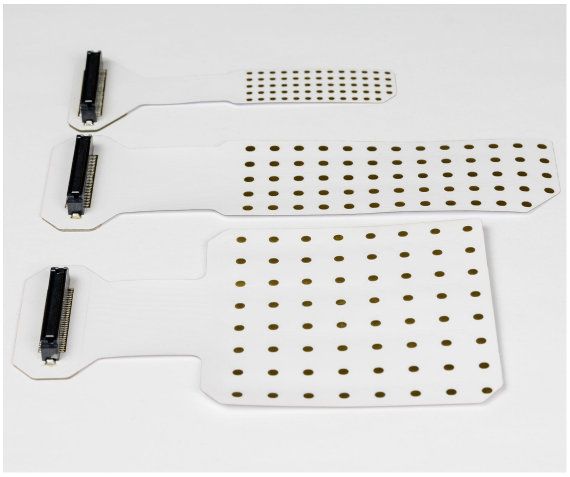
Figure 1: Three examples of 64-channel flexible surface HDEMG electrodes with different shape, size and interelectrode distance.

```
// load input data
Y1 = readMatrix("../data/simulatedEMG_Lib1_F10_10.csv");
Y2 = readMatrix("../data/simulatedEMG_Lib1_F30_10.csv");
Y1 = transpose(Y1);   Y2 = transpose(Y2);

// read and check dimensions of input data
nrowY1 = nrow (Y1);   ncolY1 = ncol (Y1);
nrowY2 = nrow (Y2);   ncolY2 = ncol (Y2);

// concatenate signals
time_before = now();
Y = as.matrix<f64>(fill (0, nrowY1, ncolY1+ncolY2));
Y[0:nrowY1, 0:ncolY1] = Y1[,];
Y[0:nrowY2, ncolY1:ncolY1+ncolY2] = Y2[,];
time_after = now();
print ("Concatenation time: " + as.scalar((time_after
  - time_before) * msec_factor) + " ms");

// extend signals
time_before = now();
eY = extendMatrix (Y, extFactor);
time_after = now();
print ("Extending time: " + as.scalar((time_after
  - time_before) * msec_factor) + " ms");

// calculate correlation matrix
time_before = now();
Ry = eY @ transpose (eY);
time_after = now();
print ("Correlation time: " + as.scalar((time_after
  - time_before) * msec_factor) + " ms");

// calculate pseudoinverse of the correlation matrix
time_before = now();
iRy = pinv2 (Ry);   // user defined function
time_after = now();
print ("Pseudoinverse time: " + as.scalar((time_after
  - time_before) * msec_factor) + " ms");
```

Figure 2: Part of a script that implements the proposed data pipeline in DaphneDSL.

To demonstrate a typical HDEMG processing operation we prepared the following simple pipeline (see Figure 3):

1. Input matrices A and B are loaded from CSV files.

2. Matrices A and B are horizontally concatenated.
3. The resulting matrix is extended by a factor R: R-1 time-delayed (right-shifted) copies of each row are added to the matrix.
4. Calculation of correlation matrix.
5. Calculation of pseudoinverse of the correlation matrix.

This pipeline was implemented as a script in DaphneDSL language (Figure 2) using the latest libraries from `daphne-eu/daphne/main` GitHub repository [20] at the time of initial writing of the paper, commit with hash `4e96943` from 2024-07-18. The DAPHNE binary was executed with additional parameters to print the MLIR-based intermediate representation (DaphneIR):

```
daphne --explain parsing_simplified,propert
y_inference ckc_simple_pipeline2.daph
```

The code includes a main function and two sub-functions: for matrix extension and for matrix pseudoinverse.

For realistic input data we prepared several sets of simulated HDEMG signals using the multilayer cylindrical volume conductor model by Farina et al. [21]. We generated signals for the Biceps Brachii muscle, with the following parameters: up to 500 motor units, electrode grid with 10 rows and 9 columns (90 channels), 5 mm interelectrode distance, constant force of 10, 30, 50 and 70 % of maximum voluntary contraction, 4096 Hz sampling rate, 600 seconds in length, no added noise, monopolar recording mode. Data is stored as a 2D matrix of raw EMG values in CSV format. For the initial tests, presented in this paper, we used only 10 different subsets of the available data, with 90 rows and 10,000, 20,000, ... and up to 100,000 columns.

We prepared a dedicated DAPHNE runtime environment in a virtual machine created with the QEMU/KVM hypervisor software ver. 4.0.0 on a Linux host computer with Kubuntu 22.04.4 LTS operating system, 64 GB RAM, AMD Ryzen Threadripper 1920X 12-core CPU and Samsung 980 PRO 2 TB NVME SSD. The virtual machine has Kubuntu 24.04 operating system installed, 16 GB RAM, 4 virtual CPUs and 100 GB of available disk. We installed all the required libraries and dependencies and built the DaphneDSL executables from the GitHub repository source code [20].

To test the initial performance of the DAPHNE system we ran the script 5 times for each of the 10 selected matrix sizes ($90 \times 10,000, 90 \times 20,000, 90 \times 30,000, ..., 90 \times 100,000$) and averaged the measured run times. Due to some experimental limitations of the DaphneDSL version at [22], we chose not to test with larger matrix sizes. To verify the correctness of our DaphneDSL implementation we manually compared the numerical results obtained on a small test matrix (10×10 elements) with results of our pre-existing Matlab implementation.

3 Results

Table 1 contains a list of all DAPHNE Intermediate Representation (DaphneIR) calls that are generated at two different stages of the compilation process. The second column shows calls after the parsing pass with some initial optimizations, while the third column shows calls after the property inference pass with additional optimizations. It is evident that the compiler generates correct kernel and built-in function calls, and that the property inference pass significantly reduces the number of generated calls, demonstrating the correct operation of the DaphneDSL compiler.

Table 2 shows the average run times of the main sections of the test script, averaged over 5 consecutive runs. We investigated the performance on 10 different matrix sizes, from $90 \times 10,000, 90 \times 20,000$, ..., up to $90 \times 100,000$. Results show that the run time increases linearly with the size of the input matrices, which is the correct and expected behavior (Figure 4).

Table 1: List of DAPHNE Intermediate Representation (DaphneIR) calls generated after the parsing stage with initial simplifications and after the property inference stage.

Call name	**No. calls after parsing**	**No. calls after inference**
daphne.cast	44	23
daphne.constant	33	42
daphne.concat	24	24
daphne.ewMul	17	8
daphne.print	12	10
daphne.now	10	10
daphne.ewAdd	9	3
daphne.ewSub	8	6
daphne.insertCol	6	6
daphne.insertRow	5	5
daphne.sliceRow	5	5
daphne.fill	4	4
daphne.return	3	3
daphne.generic_call	2	2
daphne.read	2	2
daphne.transpose	2	2
daphne.matMul	1	1
daphne.solve	1	1
daphne.numCols	8	0
daphne.numRows	8	0
daphne.ewGe	2	0
daphne.ewNeq	2	0

4 Conclusion

Preliminary testing of the DaphneDSL language for HDEMG signal processing shows that it is a viable option and can be used for creating data processing pipelines. However, the language is still in development and as such contained some bugs that prevented us to use it for more complex operations and for processing large amounts of data. For example, the number of loop iterations was limited by the available memory stack size, some operations were only available for specific data types, and a sub-

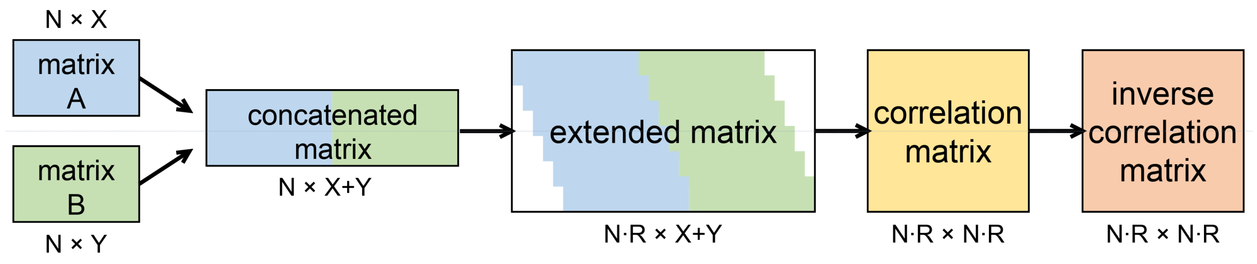

Figure 3: Overview of our test pipeline. Boxes represent matrices. N = number of rows, X,Y = number of columns, R = matrix extension factor.

Table 2: Run times of main sections of the test script for 10 different input matrix sizes, in milliseconds. Mean values ± standard deviation were estimated over 5 consecutive runs.

Operation	90 × 10.000	90 × 20.000	90 × 30.000	90 × 40.000	90 × 50.000
loading	271.8 ± 11.0	542.9 ± 18.7	824.9 ± 32.9	1102.2 ± 47.3	1346.2 ± 51.7
concatenation	53.1 ± 4.2	101.9 ± 2.6	179.9 ± 3.2	244.0 ± 5.0	302.0 ± 6.7
extending	1319.7 ± 199.2	2401.2 ± 31.5	4065.9 ± 304.8	5293.8 ± 397.2	6426.0 ± 115.9
correlation	436.2 ± 31.8	779.6 ± 37.3	1149.6 ± 43.1	1511.6 ± 37.2	1825.3 ± 43.8
pseudoinverse	13837.8 ± 179.7	13684.4 ± 262.1	13876.9 ± 402.4	14140.8 ± 163.7	13624.8 ± 116.1
Operation	90 × 60.000	90 × 70.000	90 × 80.000	90 × 90.000	90 × 100.000
loading	1650.4 ± 83.4	1914.3 ± 74.2	2181.4 ± 93.1	2441.2 ± 75.9	2736.5 ± 107.4
concatenation	366.1 ± 11.8	429.8 ± 20.7	482.8 ± 10.6	537.5 ± 8.3	589.4 ± 18.3
extending	7966.2 ± 739.5	8836.9 ± 94.1	10376.1 ± 938.0	11286.0 ± 95.1	12536.5 ± 129.3
correlation	2247.8 ± 182.8	2541.4 ± 37.5	2865.4 ± 24.5	3202.8 ± 48.9	3594.9 ± 48.8
pseudoinverse	13430.0 ± 257.9	13919.4 ± 646.8	13625.4 ± 200.3	13826.5 ± 205.1	13485.8 ± 136.1

set of typical numerical operations on matrices was available. In meantime during writing this paper, however, several bugs were quickly fixed in the language, like the mentioned issue with loops and memory (merge #820), demonstrating quick agility and response of developers in the project. Further development and testing is needed to provide an additionally reliable and convenient solution for parallel and HPC-ready processing of HDEMG data.

References

[1] A. Holobar, D. Zazula, "Multichannel Blind Source Separation Using Convolution Kernel Compensation," *IEEE Trans Signal Process* 55 (9), 2007, 4487–4496.

[2] A. Holobar, D. Farina, M. Gazzoni, R. Merletti, and D. Zazula, "Estimating motor unit discharge patterns from high-density surface electromyogram," *Clin Neurophysiol* 120 (3), 2009, 551–562.

[3] A. Holobar, M. A. Minetto, and D. Farina, "Accurate identification of motor unit discharge patterns from high-density surface EMG and validation with a novel signal-based performance metric," *J Neural Eng* 11 (1), 2014, 016008.

[4] C. J. De Luca, A. Adam, R. Wotiz, L. D. Gilmore, S. H. Nawab, "Decomposition of surface EMG signals," *J Neurophysiol* 96 (3), 2006, pp. 1646-57. doi: 10.1152/jn.00009.2006. PMID: 16899649.

[5] S. H. Nawab, S. S. Chang, C. J. De Luca, "High-yield decomposition of surface EMG signals," *Clin Neurophysiol* 121 (10) , 2010, pp. 1602-15. doi: 10.1016/j.clinph.2009.11.092. PMID: 20430694.

[6] D. Farina, F. Negro, S. Muceli, R. M. Enoka, "Principles of Motor Unit Physiology Evolve With Advances in Technology," *Physiology (Bethesda)* 31, 2016, pp. 83–94.

[7] J. A. Gallego, J. L. Dideriksen, A. Holobar, J. Ibáñez, J. L. Pons, E. D. Louis, E. Rocon, D. Farina, "Influence of common synaptic input to motor neurons on the neural drive to muscle in essential tremor," *J Neurophysiol* 113, 2015, pp. 182–191.

[8] G. Puttaraksa, S. Muceli, J. A. Gallego, A. Holobar, S. K. Charles, J. L. Pons, D. Farina, "Voluntary and tremorogenic inputs to motor neuron pools of agonist/antagonist muscles in essential tremor patients," *J Neurophysiol* 122, 2019, pp. 2043–2053.

[9] S. Avrillon, F. Hug, R. Enoka, A. H. Caillet, D. Farina, "The decoding of extensive samples of motor units in human muscles reveals the rate coding of entire motoneuron pools," *eLife13:* RP97085, 2024. doi: 10.7554/eLife.97085.1

[10] P. Damme, M. Birkenbach, C. Bitsakos, M. Boehm, P. Bonnet, F. Ciorba, M. Dokter, P. Dowgiallo, A. Eleliemy, C. Faerber, G. Goumas, D. Habich, N. Hedam, M. Hofer, W. Huang, K. Innerebner, V. Karakostas, R. Kern, T. Kosar, A. Krause, D. Krems, A. Laber, W. Lehner, E. Mier, M. Paradies, B. Peischl, G. Poerwawinata, S. Psomadakis, T. Rabl, P. Ratuszniak, P. Silva, N. Skuppin, A. Starzacher, B. Steinwender, I. Tolovski, P. Tözün, W. Ulatowski, Y. Wang, I. Wrosz, A. Zamuda, C. Zhang, and X. X. Zhu, "Daphne: An open and extensible system infrastructure for integrated data analysis pipelines," in *12th Conference on Innovative Data Systems Research, CIDR 2022, Chaminade, CA, January 9-12*, 2022.

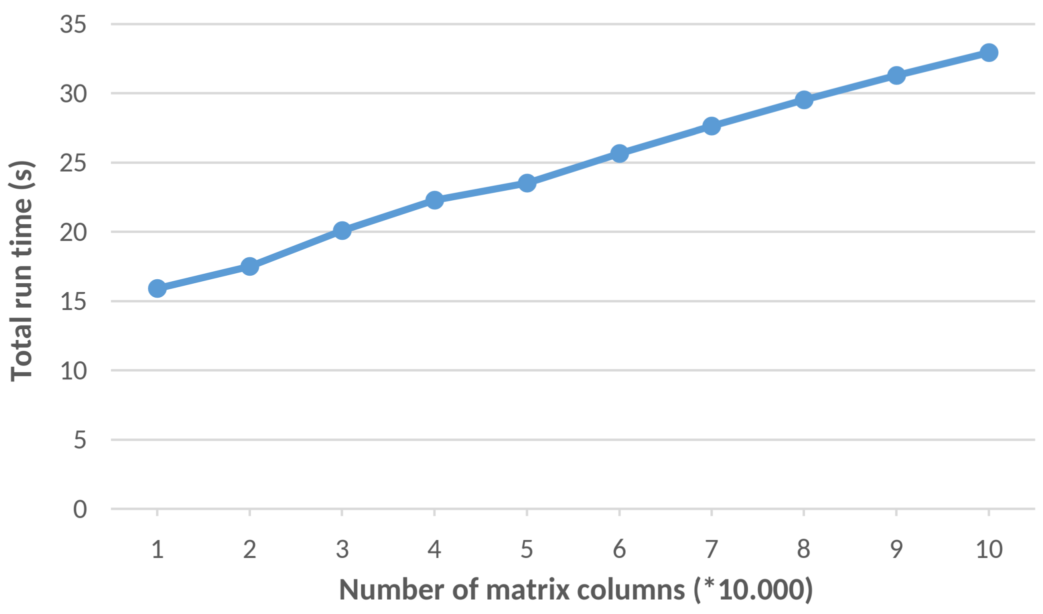

Figure 4: Mean total run time of the script for processing of the example pipeline, estimated over 5 consecutive runs. The input matrices consisted of 90 rows and from 10,000 to 100,000 columns. Time was measured in seconds.

[11] Publication Office of the European Union, "Fact sheet : Integrated data analysis pipelines for large-scale data management, hpc, and machine learning," in *CORDIS – EU research results*, 2024, p. https://cordis.europa.eu/project/id/957407.

[12] A. Vontzalidis, S. Psomadakis, C. Bitsakos, M. Dokter, K. Innerebner, P. Damme, M. Boehm, F. Ciorba, A. Eleliemy, V. Karakostas, A. Zamuda, and D. Tsoumakos, "DAPHNE Runtime: Harnessing Parallelism for Integrated Data Analysis Pipelines," in *Euro-Par 2023: Parallel Processing Workshops*, ser. Lecture Notes in Computer Science, vol. 14352, D. Zeinalipour, D. B. Heras, G. Pallis, H. Herodotou, D. Trihinas, D. Balouek, P. Diehl, T. Cojean, K. Fürlinger, M. H. Kirkeby, M. Nardellli, and P. D. Sanzo, Eds. Cham: Springer, 2024, pp. 242–246.

[13] A. Zamuda, "Business intelligence through computational intelligence for data science: an overview of the breadth of real numerical optimization solutions – through autonomous submarines, forest modeling, power plant management, space probes, protein modeling, radio signal coding to text summarization," in *lecture presented at SMART FUTURE Data Science and Business Intelligence, UNI.MINDS, 17. november 2020*, 2020.

[14] DAPHNE: An Open and Extensible System Infrastructure for Integrated Data Analysis Pipelines. "DaphneDSL Language Reference," GitHub repository daphne-eu/daphne. https://github.com/daphne-eu/daphne/blob/main/doc/DaphneDSL/LanguageRef.md , visited on 18. 7. 2024.

[15] A. Zamuda, Mark Dokter, "Deploying DAPHNE Computational Intelligence on EuroHPC Vega for Benchmarking Randomised Optimisation Algorithms," in *International Conference on Broadband Communications for Next Generation Networks and Multimedia Applications (CoBCom)*, 2024, https://www.cobcom.tugraz.at/wordpress/proceedings-cobcom-2024/.

[16] Apache Software Foundation (ASF), An open source ML system for the end-to-end data science lifecycle . "DML Language Reference," GitHub repository apache/systemds. https://github.com/apache/systemds/blob/main/docs/site/dml-language-reference.md , visited on 18. 7. 2024.

[17] A. Zamuda, "Monitoring and operational data analytics from a user perspective at first EuroCC HPC Vega supercomputer and nation-wide in Slovenia," in *invited lecture at moda21 : Second International Workshop on Monitoring and Operational Data Analytics, 2 July 2021, Strasbourg (France)*, 2021.

[18] MIZŠ, "Ministrstvo za izobraževanje, znanost in šport (Direktorat za znanost in inovacije): HPC Vega kot temelj za inovacijsko-raziskovalni projekt DAPHNE," in *Portal GOV.SI : spletišče državne uprave s celovitimi informacijami o njenem delovanju in preprostim dostopom do storitev. Ljubljana: Urad Vlade Republike Slovenije za komuniciranje. 5. 1. 2022*, 2022, pp., https://www.gov.si/novice/2022–01–05–hpc–vega–kot–temelj–za–inovacijsko–raziskovalni–projekt–daphne/ , visited on 18. 7. 2024.

[19] ——, "Randomised Optimisation Algorithms in DAPHNE," in *Austrian-Slovenian HPC Meeting 2024 – ASHPC24*, 2024, p. 33.

[20] DAPHNE project GitHub repository, https://github.com/daphne-eu/daphne , visited on 18. 7. 2024.

[21] D. Farina, L. Mesin, S. Martina, and R. Merletti, "A surface EMG generation model with multilayer cylindrical description of the volume conductor," *IEEE Transactions on Biomedical Engineering* 51 (3), 2004, pp. 415–426. doi: 10.1109/TBME.2003.820998.

[22] DAPHNE project GitHub repository - open issues, https://github.com/daphne-eu/daphne/issues , visited on 18. 7. 2024.