DISCRETE AND CONTINUOUS MODELS AND APPLIED COMPUTATIONAL SCIENCE

Volume 33 Number 4 (2025)

Founded in 1993

Founder: PEOPLES' FRIENDSHIP UNIVERSITY OF RUSSIA NAMED AFTER PATRICE LUMUMBA

DOI: 10.22363/2658-4670-2025-33-4

Edition registered by the Federal Service for Supervision of Communications, Information Technology and Mass Media
Registration Certificate: ПИ № ФС 77-76317, 19.07.2019

ISSN 2658-7149 (Online); 2658-4670 (Print)
4 issues per year.
Language: English.

Publisher

Peoples' Friendship University of Russia named after Patrice Lumumba (RUDN University).

Indexed by

- Scopus (https://www.scopus.com),
- Ulrich's Periodicals Directory (http://www.ulrichsweb.com),
- Directory of Open Access Journals (DOAJ) (https://doaj.org),
- Russian Index of Science Citation (https://elibrary.ru),
- CyberLeninka (https://cyberleninka.ru).

Aim and Scope

Discrete and Continuous Models and Applied Computational Science arose in 2019 as a continuation of RUDN Journal of Mathematics, Information Sciences and Physics. RUDN Journal of Mathematics, Information Sciences and Physics arose in 2006 as a merger and continuation of the series "Physics", "Mathematics", "Applied Mathematics and Computer Science", "Applied Mathematics and Computer Mathematics".

Discussed issues affecting modern problems of physics, mathematics, queuing theory, the Teletraffic theory, computer science, software and databases development.

It's an international journal regarding both the editorial board and contributing authors as well as research and topics of publications. Its authors are leading researchers possessing PhD and PhDr degrees, and PhD and MA students from Russia and abroad. Articles are indexed in the Russian and foreign databases. Each paper is reviewed by at least two reviewers, the composition of which includes PhDs, are well known in their circles. Author's part of the magazine includes both young scientists, graduate students and talented students, who publish their works, and famous giants of world science.

The Journal is published in accordance with the policies of COPE (Committee on Publication Ethics).

The editors are open to thematic issue initiatives with guest editors. Further information regarding notes for contributors, subscription, and back volumes is available at http://journals.rudn.ru/miph

E-mail: miphj@rudn.ru, dcm@rudn.su

Editorial board

Computer Design: *Anna V. Korolkova*, *Dmitry S. Kulyabov*
English Text Editors: *Nikolay E. Nikolaev*, *Ivan S. Zaryadov*, *Konstantin P. Lovetskiy*
Address of editorial board: 3 Ordzhonikidze St, 115419, Moscow, Russia, +7 (495) 955-07-16, e-mail: publishing@rudn.ru
Editorial office: +7 (495) 952-02-50, e-mail: miphj@rudn.ru, dcm@rudn.su, site: http://journals.rudn.ru/miph

---

Approved for printing: 01.12.2025. Published: 05.12.2025.
Paper size 70×100/16. Offset paper. Offset printing. Typeface “Adobe Source.”
Conventional printed sheets: 9.67. Printing run 500 copies. Open price. The order: 1655.
PEOPLES' FRIENDSHIP UNIVERSITY OF RUSSIA NAMED AFTER PATRICE LUMUMBA
6 Miklukho-Maklaya St, Moscow, 117198, Russian Federation
Printed at RUDN Publishing House:
3 Ordzhonikidze St, Moscow, 115419, Russian Federation,
+7 (495) 955-08-61; e-mail: publishing@rudn.ru

Discrete & Continuous Models & Applied Computational Science
ISSN 2658-7149 (Online), 2658-4670 (Print)

2025, 33 (4)
http://journals.rudn.ru/miph

Contents

Editorial

Computer science

Modeling and Simulation

Physics and Astronomy

Discrete & Continuous Models
& Applied Computational Science
ISSN 2658-7149 (Online), 2658-4670 (Print)

2025, 33 (4) 355–360
http://journals.rudn.ru/miph

Editorial

Editorial

DOI: 10.22363/2658-4670-2025-33-4-355-360 EDN: HMXQHS

Basic elements of the BibTeX file structure

Dmitry S. Kulyabov[1, 2], Anna V. Korolkova[1], Leonid A. Sevastianov[1, 2], Yuri P. Rybakov[1]

[1] RUDN University, 6 Miklukho-Maklaya St, Moscow, 117198, Russian Federation

[2] Joint Institute for Nuclear Research, 6 Joliot-Curie St, Dubna, 141980, Russian Federation

Abstract. BibTeX is used to prepare bibliographic information for our journals. This article describes the basic structure of BibTeX files.

Key words and phrases: bibliography typesetting, LaTeX, BibTeX

For citation: Kulyabov, D. S., Korolkova, A. V., Sevastianov, L. A., Rybakov, Y. P. Basic elements of the BibTeX file structure. *Discrete and Continuous Models and Applied Computational Science* **33** (4), 355–360. doi: 10.22363/2658-4670-2025-33-4-355-360. edn: HMXQHS (2025).

1. Introduction

When preparing manuscripts for the journal, BibTeX is used to format bibliographies [1]. BibTeX is a file format for storing bibliographic data. BibTeX files have the extension `.bib` and contain records, each describing a single source (book, article, report, etc.).

2. Entry

- The record type (e.g., `@article`, `@book`, `@misc`) indicates the type of source.
- The unique key is the record identifier, which is used to cite the source in the document text.
- Each record begins with the source type and a unique key.

A BibTeX record begins with the `@` sign, followed by the record type name. Everything pertaining to the record is enclosed in curly braces:

`@book` `{ ... }`

BibTeX offers 14 record types. Record type names are case-insensitive.
List of BibTeX record types:

- *`article`*: an article published in a periodical, such as a journal article or magazine article;
- *`book`*: a book;
- *`booklet`*: like a book, but without a publisher;
- *`conference`*: conference proceedings;
- *`inbook`*: a section or chapter in a book;
- *`incollection`*: an article in a collection;
- *`inproceedings`*: a conference paper (same as the *`conference`* record type);

- *manual*: technical manual;
- *masterthesis*: master's thesis;
- *misc*: used when nothing else applies;
- *phdthesis*: Ph.D. thesis;
- *proceedings*: conference proceedings (full);
- *techreport*: technical report, government report, or white paper;
- *unpublished*: proceedings that have not been formally published.

3. Fields

- These are record attributes containing specific information about the source (author, title, year of publication, etc.).
- Fields are separated by commas.
- Some fields may be required for certain record types, while others are optional.

3.1. Standard field types

Let's list the standard BibTeX field types:

- *address*: publisher or institution address;
- *annote*: annotation;
- *author*: list of authors of the work;
- *booktitle*: book title;
- *chapter*: chapter number in the book;
- *edition*: book edition number;
- *editor*: list of book editors;
- *howpublished*: publication date for non-standard publications;
- *institution*: name of the institution that published and/or sponsored the report;
- *journal*: name of the journal or magazine in which the article was published;
- *month*: month of publication of the work;
- *note*: notes;
- *number*: report number or issue number of the journal article;
- *organization*: name of the institution that organized or sponsored the conference or published the guideline;
- *pages*: page numbers or page range;
- *publisher*: publisher;
- *school*: name of the university or institution awarding the degree;
- *series*: name of the series or book set;
- *title*: title of the work;
- *type*: type of technical report or dissertation;
- *volume*: volume number;
- *year*: year of publication of the work.

3.2. Custom field types

Not all BibTeX styles support non-standard fields:

- *doi*: DOI number;
- *issn*: ISSN number;

- *isbn*: ISBN number;
- *url*: URL of a web page.

3.3. Structure of the author field

Let's describe the structure of the most frequently used field — *author*.

Authors' names are written in the following format:

- Lastname, Firstname Patronymic;
- Firstname Patronymic Lastname;
- Lastname, Initials;
- Initials Lastname.

Basic rules:

- To separate multiple authors, use the *and* operator.
- If the author's last name is written before the initials, a comma is placed after the last name (,).
- Initials are separated by periods.
- Spaces are placed between initials.

Examples:

- Single author: *author = {Ivanov, I. I.}*
- Multiple authors: *author = {Petrov, P. P. and Ivanov, I. I.}*.

3.3.1. Format: {Firstname Lastname}

First name, then last name. This format is widely used, but may not provide the necessary distinction between first and last names, especially with complex last names.

```
@article{Doe2023,
title  = {Title of the Article},
year   = 2023,
author = {Isaac Newton},
...}
```

3.3.2. Format: {Lastname, Firstname}

List the last name first, then the first name, separated by a comma. This format is recommended as it clearly separates the first and last names.

```
@article{Doe2023,
title  = {Title of the Article},
year   = 2023,
author = {Newton, Isaac},
...}
```

3.3.3. Format: {Lastname, Suffix, Firstname}

This format is useful when a suffix (e.g., Jr., Sr., III, etc.) is required. It is important to strictly follow the order to accurately reflect the author's name.

```
@article{Doe2023,  title  = {Title of the Article},  year   = 2023,  author
 ↪  = {King, Jr, Martin Luther},  ...}
```

3.3.4. Names of multiple authors

If a work has multiple authors, BibTeX allows you to specify them using the *and* separator. This separator is used to clearly list the authors according to the selected format.

```
@article{Doe2023,
title  = {Title of the Article},
year   = 2023,
author = {Fisher, James and Clark, John},
...}
```

4. Values

- This is the data stored in the fields.
- Values are enclosed in curly braces {…} or double quotes "…".
- Using curly braces is preferred as it avoids problems with interpreting special characters.

5. Example entry in a BibTeX file

Example bibtex entry:

```
@book{ivanov2023book,
author = {Ivanov, I. I.},
title = {Book title},
year = {2023},
publisher = {Publisher},
address = {City}
}
```

Here:

- `@book`: record type (book);
- `ivanov2023book`: record unique key;
- `author`, `title`, `year`, `publisher`, `address`: record fields;
- `{Ivanov, I. I.}`, `{Book Title}`, etc.: field values.

6. Recommendations for working with BibTeX files

- Uniqueness of keys: Each entry key must be unique within the file to avoid confusion when citing.
- Using comments: Comments can be added to a BibTeX file, starting them with a percent sign.
- A BibTeX file can contain multiple entries, each describing a separate bibliographic source.

Author Contributions: The contributions of the authors are equal. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: No new data were created or analysed during this study. Data sharing is not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Declaration on Generative AI: The authors have not employed any Generative AI tools.

References

1. Mittelbach, F., Goossens, M., Braams, J., Carlisle, D. & Rowley, C. *The LaTeX Companion* 2nd ed. 1120 pp. (Addison-Wesley Professional, 2004).

Information about the authors

Kulyabov, Dmitry S.—Professor, Doctor of Sciences in Physics and Mathematics, Professor of Department of Probability Theory and Cyber Security of RUDN University; Senior Researcher of Laboratory of Information Technologies, Joint Institute for Nuclear Research (e-mail: kulyabov-ds@rudn.ru, ORCID: 0000-0002-0877-7063, ResearcherID: I-3183-2013, Scopus Author ID: 35194130800)

Korolkova, Anna V.—Docent, Candidate of Sciences in Physics and Mathematics, Associate Professor of Department of Probability Theory and Cyber Security of RUDN University (e-mail: korolkova-av@rudn.ru, ORCID: 0000-0001-7141-7610, ResearcherID: I-3191-2013, Scopus Author ID: 36968057600)

Sevastianov, Leonid A.—Professor, Doctor of Sciences in Physics and Mathematics, Professor of Department of Computational Mathematics and Artificial Intelligence of RUDN University (e-mail: sevastianov-la@rudn.ru, ORCID: 0000-0002-1856-4643, ResearcherID: B-8497-2016, Scopus Author ID: 8783969400)

Rybakov, Yuri P.—Professor, Doctor of Sciences in Physics and Mathematics, Professor of the Institute of Physical Research and Technologies of RUDN University (e-mail: rybakov-yup@rudn.ru, ORCID: 0000-0002-7744-9725, ResearcherID: S-4813-2018, Scopus Author ID: 16454766600)

DOI: 10.22363/2658-4670-2025-33-4-355-360 EDN: HMXQHS

Основные элементы структуры файла BibTeX

Д. С. Кулябов[1,2], А. В. Королькова[1], Л. А. Севастьянов[1,2], Ю. П. Рыбаков[1]

[1] Российский университет дружбы народов, ул. Миклухо-Маклая, д. 6, Москва, 117198, Российская Федерация

[2] Объединённый институт ядерных исследований, ул. Жолио-Кюри, д. 6, Дубна, 141980, Российская Федерация

Аннотация. BibTeX используется для подготовки библиографической информации для журнала. В статье описывается базовая структура файлов BibTeX.

Ключевые слова: набор библиографии, LaTeX, BibTeX

Discrete & Continuous Models
& Applied Computational Science
ISSN 2658-7149 (Online), 2658-4670 (Print)

2025, 33 (4) 361–373
http://journals.rudn.ru/miph

Computer science

Research article

UDC 519.872, 519.217
PACS 07.05.Tp, 02.60.Pn, 02.70.Bf
DOI: 10.22363/2658-4670-2025-33-4-361-373 EDN: HYUCMC

Leaf disease recognition using deep learning methods

Ali Salem Muthana, Elena V. Lyapuntsova

National University of Science and Technology MISIS, 44 building 2 Vavilova St, Moscow, 119049, Russian Federation

(received: August 21, 2025; revised: September 25, 2025; accepted: September 30, 2025)

Abstract. The digitalization of crop production has placed leaf-image-based disease recognition among the top research priorities. This paper presents a compact and reproducible system designed for rapid deployment in cloud environments and subsequent adaptation. The proposed approach combines multitask learning (simultaneous prediction of plant species and disease), physiologically motivated channel processing, and error-tolerant data preparation procedures. Experiments were conducted on the New Plant Diseases Dataset (Augmented). To accelerate training, six of the most represented classes were selected, with up to 120 images per class. Images were resized to 192×192 and augmented with geometric and color transformations as well as soft synthetic lesion patches. The ExG greenness index was embedded into the green channel of the input image. The architecture was based on EfficientNet-B0; the proposed HiP^2-Net model included two classification heads for disease and species. Training was carried out in two short stages, with partial unfreezing of the base network's tail in the second stage. Evaluation employed standard metrics, confusion matrices, test-time augmentation, and integrated gradients maps for explainability. On the constructed subset, the multitask HiP^2-Net consistently outperformed the frozen baseline model in accuracy and aggregate metrics. Synthetic lesions reduced background sensitivity and improved detection of mild infections, while incorporating ExG enhanced leaf tissue separation under variable lighting. Integrated gradient maps highlighted leaf veins and necrotic spots, strengthening trust in predictions and facilitating expert interpretation. The proposed scheme combines the practicality of cloud deployment with simple, physiology-inspired techniques. Adopting the "species + disease" setup together with ExG preprocessing and soft synthetic lesions improves robustness to lighting, background, and geometric variations, and makes it easier to transfer models to new image collections.

Key words and phrases: plant disease recognition, leaf images, deep learning, transfer learning, multi-task classification, explainability, lightweight models

For citation: Muthana, A. S., Lyapuntsova, E. V. Leaf disease recognition using deep learning methods. *Discrete and Continuous Models and Applied Computational Science* **33** (4), 361–373. doi: 10.22363/2658-4670-2025-33-4-361-373. edn: HYUCMC (2025).

1. Introduction

The development of precision agriculture has posed a practical question for agro-information systems: how to detect plant diseases from leaf images quickly, reliably, and without unnecessary costs for labeling and computation. Field conditions rarely tolerate excessive algorithmic complexity: light changes due to cloudiness, the background is cluttered with soil and plant residues, and mobile devices limit the amount of available memory and response time. The growing wave of research on recognizing leaf diseases using deep neural networks shows that even compact architectures can achieve high accuracy on standard datasets while maintaining acceptable operating speed and explainability of decisions [1–4]. Recent reviews note a shift in research attention to lightweight models resistant to the "noise" of field images and to methods capable of transferring knowledge between tasks and datasets [5].

Against the backdrop of impressive accuracy figures, a common limitation emerges: training and evaluation are often performed on controlled datasets with uniform backgrounds and static lighting, which provokes hidden "adaptation" to background features. Publications on channel attention and structural pruning confirm that meaningful redistribution of attention within the model reduces redundancy and restrains overfitting while maintaining a similar level of accuracy [6]. Comparisons of the AlexNet, MobileNet, and EfficientNet families on multi-class disease classification show that compact models perform better where operation on edge devices without a graphics card is required, especially with careful tuning of augmentations and hyperparameters [2, 7]. Multi-scale convolutional blocks selectively extract textural and macroscopic features of spots and coatings, which is useful for differentiating similar symptoms between crops [3]. Review articles also highlight another trend: interest in joint formulations where the network simultaneously learns to recognize plant species and diagnosis, obtaining more universal features and additional regularization [8, 9].

The present work is based on the idea of physiologically motivated input. The Excess Green (ExG) index, calculated as a linear combination of RGB channels, highlights the leaf mesophyll and reduces the influence of unstable lighting and cluttered background, therefore including such an indicator in the preprocessing pipeline is appropriate for field conditions. Instead of explicit segmentation of the leaf blade, a softer approach was chosen: ExG is embedded into the green channel without violating the expected input format of the pretrained EfficientNet-B0 convolutional network. Such a strategy is consistent with the course toward compactness and careful channel handling described in studies on attention and compression [1], and with survey recommendations on enhancing the interpretability of features [10].

Comparison with related studies reveals two lines of similarity and one principal difference. First, transfer learning on a pretrained architecture is used, as in most works of recent years [2, 7, 11]. Second, a multitask formulation "species + disease" is applied, which has already demonstrated the ability to improve generalization due to additional taxonomic signals [8]. The main peculiarity of the present study is connected with engineering simplicity and physiological motivation of features: the ExG index is integrated directly into the data stream, and symptom variability is extended with synthetic "soft" spots, which makes it possible to model contrast and lesion shape without heavy annotation procedures and without the risk of execution errors. This approach relates the work to the line of research on reducing computational load and increasing robustness while maintaining readability of internal representation [8, 12–14].

The problem solved by the article is formulated as the need to obtain a reproducible and easy-to-deploy recognition system capable of working on reduced class subsets and under conditions of variable image quality. Taking into account the sensitivity to domain shift between laboratory and field images noted in the literature, the main research question becomes verification: will the combination of simple physiological indicators and multitask learning lead to more stable predictions

on a test set formed from real images with different lighting and background texture [15]. The main hypothesis states that soft embedding of ExG into the green channel reduces the model's dependence on background patterns and improves recognition of weakly expressed symptoms. An additional hypothesis links multitasking with a regularization effect: parallel prediction of plant species keeps features in a biologically meaningful subspace, reduces overfitting, and ensures gains on previously unseen examples [16]. The technical hypothesis concerns the training mode: a two-stage scheme with partial unfreezing of the "tail" of EfficientNet-B0 and label smoothing increases accuracy compared to a fully frozen baseline at a similar epoch budget, which is consistent with results on structural efficiency and attention to the last convolutional blocks [1, 2].

The goal of the study is to design and experimentally verify a hierarchical, physiologically guided network for leaf disease recognition, suitable for rapid deployment in a cloud environment and subsequent transfer to applied scenarios. The tasks include the development of a robust preprocessing pipeline considering leaf physiology, the design of a multitask head for simultaneous recognition of species and diagnosis, the introduction of synthetic lesions to expand the variability of training examples, as well as analysis of explainability through integrated gradients and evaluation of the contribution of individual design decisions. The chosen theoretical approach – a combination of transfer learning, regularization through a secondary task, and physiologically motivated enhancement of the green channel – directly follows from trends outlined in recent review and applied works, and meets the demand for lightweight, interpretable, and domain-robust models [3, 4, 7, 8].

2. Materials and methods

The empirical base was formed from the open set "New Plant Diseases Dataset (Augmented)" loaded via the Kaggle website, followed by automatic unpacking and robust root directory searching, regardless of nesting depth. The original structure contains separate train / valid folders; however, to ensure proper distribution control, all images were reindexed into a single table, followed by stratified division into training, validation, and test sections, without relying on the original subdirectories. To speed up the experiment and constrain the computational budget, a subset of the six most widely represented disease classes was selected; the number of images in each class was limited to a maximum of 120, corresponding to an upper estimate of approximately 720 images. This truncation, although reducing overall representativeness, allows for comparison of architectural solutions with a fixed training time and for monitoring the impact of augmentations and multitasking. Representativeness within the selected six diagnoses was ensured by stratification by disease label at each sampling step. Random samples were generated at a fixed grain size so that any researcher could reproduce an identical cut of the data. The research design was built around comparing two interpretations of the same pretrained backbone. EfficientNet-B0 with ImageNet-trained weights was used as the base architecture, with the convolutional "stem" and feature blocks kept frozen. The baseline model had a single classification head for "disease" and a dummy output for "species" to unify the training interface. The proposed HiP^2-Net retained the same convolutional backbone but included two heads: "disease" and "species," trained jointly. This multi-output setup is consistent with observations on the regularizing effect of parallel tasks in plant classifiers, where species features stabilize symptom recognition [4, 8]. The choice of a compact backbone and moderately sized heads follows the trend toward lightweight and transferable solutions described in studies on channel attention and compression [1], as well as in comparative works on mobile networks [2, 7]. The image preprocessing procedure was designed with consideration of leaf physiology. Each frame was resized to 192×192 pixels and scaled to the $[0, 1]$ range, followed by normalization according to

the EfficientNet input requirements. To reduce dependence on background and lighting, the ExG greenness index

$$\mathrm{ExG} = 2G - R - B$$

was computed, then normalized per frame and softly blended into the green channel of the original RGB image by linear mixing, where the original G channel retained the dominant share. This insertion, without altering the expected input dimensionality, emphasizes mesophyll tissues and reduces the influence of shadows, which is critical under field conditions and aligns logically with the trend of channel attention noted in modern literature [1, 8]. Geometric and color transformations included random horizontal and vertical flips, rotations by multiples of 90°, as well as soft variations in brightness, contrast, saturation, and hue. To model lesion sites, synthetic "spots" were applied in the form of several semi-transparent radial masks with variable radius and warm yellow–brown tones; the masks were overlaid in an additive-blending manner without altering data type, and generated with a probability that limited the proportion of aggressive transformations. Such interventions reproduce variability in contrast and lesion shape without manual annotation and improve robustness to domain shift, consistent with review findings on the importance of realistic augmentations [8].

Subsampling was performed in two steps, maintaining stratification by disease label: first, 80% of the images were sent to the training part, with the remaining 20% serving as test. Next, 15% of the training part was used to create the validation set. The resulting splits were approximately 68% training, 12% validation, and 20% test. The data pipeline was built on `tf.data`, batching images into 32-image batches and prefetching on the fly. This organization minimizes downtime due to disk operations and allows for consistent performance during additional training.

Hyperparameters were selected with an eye on reproducibility and a limited epoch budget. The AdamW optimizer was used with a starting learning rate of 3×10^{-4} in the first stage and 1×10^{-4} in the second, and a weight decay coefficient decreased from 1×10^{-4} to 5×10^{-5} between stages. Cross-entropy with label smoothing of 0.10 and 0.05 was used for the "disease" and "species" heads, respectively, to reduce overconfidence on frequent patterns. The loss vector was set to 0.8 for diagnosis and 0.2 for species, reflecting the priority of the primary task and simultaneously allowing species to act as a soft source of regularization. Early stopping was focused on the validation accuracy for disease with a margin of three epochs; adaptive training step reduction was tied to the validation loss for diagnosis with a coefficient of 0.5 and a lower threshold of 10^{-6}. In the second stage, several dozen final convolutional layers of the foundation were unfrozen, thereby ensuring fine-tuning of high-level features to the specific texture of leaves and symptoms. The chosen scheme is consistent with the practice of careful retraining of the last blocks and demonstrates gains with similar computational costs, which has been repeatedly noted in comparative works [2, 7].

The evaluation procedures included two levels. At the first level, we calculated the accuracy of the test portion, an extended classification report with precision, recall, and F1-score for each class, and a confusion matrix for analyzing persistent confusions between closely related diagnoses. At the second level, we tested a simple test augmentation during output by averaging the predictions of the original and horizontally mirrored images, which allowed us to assess the model's sensitivity to leaf blade symmetries. To clarify the solution mechanics, contribution maps were constructed using the integrated gradient method; this technique was chosen due to the absence of requirements for internal activations and its ease of application to a "pure" classification model, in contrast to approaches requiring specialized intermediate feature maps. Interpretation relied on checking the consistency of hot spots with veins, spot boundaries, and necrotic foci, as required by the modern explainability agenda in leaf disease diagnostics [4, 8].

Table 1

Training summary (validation and test)

Stage	Epoch	LR	disease_acc	disease_loss	species_acc	species_loss	val_disease_acc	val_disease_loss	val_species_acc	val_species_loss	val_loss
S1	1	3.0e-04	0.2874	1.7461	0.5268	1.3863	0.4713	1.5699	0.4943	1.3863	1.5332
S1	2	3.0e-04	0.4850	1.5361	0.5112	1.3863	0.6667	1.4061	0.4943	1.3863	1.4022
S1	3	3.0e-04	0.6824	1.3500	0.5034	1.3863	0.8046	1.2701	0.4943	1.3863	1.2934
S1	4	3.0e-04	0.7378	1.2108	0.5000	1.3863	0.8391	1.1634	0.4943	1.3863	1.2080
S1	5	3.0e-04	0.8192	1.1320	0.5063	1.3863	0.8506	1.0784	0.4943	1.3863	1.1399
S1	6	3.0e-04	0.8298	1.0454	0.5030	1.3863	0.8506	1.0080	0.4943	1.3863	1.0836
S2	1	3.0e-04	0.2130	1.9955	0.3223	1.3976	0.2759	1.7306	0.5632	1.1527	1.6150
S2	2	3.0e-04	0.2768	1.7029	0.5422	1.1287	0.5057	1.5336	0.5977	1.0131	1.4295
S2	3	3.0e-04	0.4848	1.5186	0.6161	1.0227	0.6782	1.3803	0.7586	0.8975	1.2837
S2	4	3.0e-04	0.6103	1.3706	0.6835	0.9147	0.7471	1.2558	0.8046	0.8037	1.1654
S2	5	3.0e-04	0.7357	1.2532	0.7728	0.7843	0.8391	1.1516	0.8391	0.7225	1.0658
S2	6	3.0e-04	0.7332	1.1734	0.8237	0.7303	0.8851	1.0687	0.8506	0.6610	0.9872
S3	1	1.0e-04	0.6046	1.3831	0.6811	0.9359	0.9080	0.9073	0.8966	0.5812	0.8421
S3	2	1.0e-04	0.8130	0.9839	0.8129	0.7826	0.9195	0.7811	0.9770	0.4969	0.7242
S3	3	1.0e-04	0.9285	0.7868	0.8747	0.6516	0.9310	0.6936	1.0000	0.4248	0.6399
S3	4	1.0e-04	0.9681	0.6543	0.9400	0.5202	0.9425	0.6399	1.0000	0.3679	0.5855

Table 2

Comparison of Baseline vs. HiP^2-Net classification metrics

Class	Baseline P	Baseline R	Baseline F1	Support	HiP^2-Net P	HiP^2-Net R	HiP^2-Net F1	Support
Apple___Apple_scab	0.8947	0.7083	0.7907	24	0.9444	0.7083	0.8095	24
Apple___Black_rot	1.0000	0.9167	0.9565	24	0.9565	0.9167	0.9362	24
Apple___healthy	0.6286	0.9167	0.7458	24	0.7742	1.0000	0.8727	24
Orange___Haunglongbing_(Citrus_greening)	1.0000	0.9583	0.9787	24	1.0000	1.0000	1.0000	24
Pepper, bell___healthy	0.9200	0.9583	0.9388	24	0.9583	0.9583	0.9583	24
Soybean___healthy	0.9500	0.7917	0.8636	24	1.0000	1.0000	1.0000	24

The computing environment prepared video memory for dynamic growth, thereby preventing crashes at session start; randomness was captured at the Python, NumPy, and TensorFlow levels. Training sessions were conducted with explicit graph execution, eliminating random autograph errors during complex user augmentations. This set of engineering measures reduces protocol variability and enables "rapid replication" of the experiment in a cloud environment without manual environment tuning. In the context of the literature, the adopted methodology occupies a "middle ground": instead of profound architectural innovation, the emphasis is on physiologically based input, multi-task formulation, and careful fine-tuning of a compact database — an approach consistent with current practice in developing lightweight and portable recognizers for the agricultural sector [1, 3].

3. Results

The robustness of the pipeline was tested on a reduced subset of six diagnoses. Figures are accompanied by bilingual captions; console summaries have been transferred to tables.
Note: S1/S2/S3 — three consecutive training stages; S3 — additional training with a smaller step; TTA — averaging of predictions.

According to the tables, the multi-task HiP^2-Net, after partially unfreezing the last convolutional blocks, achieved a test accuracy of 0.9306, adding 5.6 percentage points to the baseline EfficientNet-B0 (0.8750). Averaging two predictions with horizontal reflection yielded a slight increase to 0.9375.

Table 3

Summary macrometrics and final accuracy

Model	Accuracy (no TTA)	Accuracy (with TTA*)	Macro Precision	Macro Recall	Macro F1	Samples
Baseline (EfficientNet-B0)	0.8750	0.8750	0.8989	0.8750	0.8790	144
HiP^2-Net (multitask)	0.9306	0.9375	0.9389	0.9306	0.9295	144

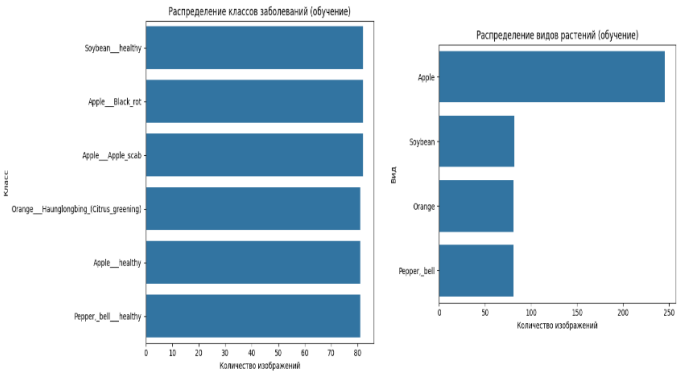


Figure 1. Distribution of disease classes and plant species in the training set; the balance of classes within the selected subset is close to equal

Macro-precision and macro-F1 increased from 0.8989/0.8790 to 0.9389/0.9295, which is consistent with the idea of regularization through parallel prediction of the form. The most noticeable improvement was observed for the `Apple___healthy` class: recall increased from 0.9167 to 1.0000, F1—from 0.7458 to 0.8727. The weak point remained the confusion `Apple___Apple_scab` ↔ `Apple___healthy`, where the recall remained at 0.7083 while the prediction accuracy increased.

4. Discussion

The obtained data fully confirmed the main hypothesis that including the ExG physiological index in the green channel reduces the model's dependence on background patterns and improves recognition stability under varying shooting conditions[16]. The additional hypothesis regarding the benefits of a multi-task approach was also confirmed: jointly predicting plant species and disease improved generalization, as evidenced by the increase in macro-accuracy and F1 metrics. The technical prediction of a gain from the two-stage scheme with partial unfreezing of the final layers of EfficientNet-B0 was also confirmed: validation accuracy for disease increased to almost 0.94, and the final test metric was 0.9306 versus 0.8750 for the baseline model.

Comparison with results from other researchers allows us to place our observations in a broader scientific context. Architectures based on channel-based attention and structural compression show that emphasizing informational channels provides gains while maintaining compactness [1]. Our result with ExG-boosting can be viewed as a simple and physiologically sound variation of channel-based attention, but without increasing the complexity of the model. Previously described compact variants based on MobileNet demonstrated up to 97.3% accuracy on a set of classes[2]; in our case, the gain was achieved on a smaller subset, but the increase compared to the pure baseline confirms the value of this approach.

Studies focusing on multi-task architectures show that parallel species and disease prediction leads to trait stabilization and better transfer across crops [17, 18]. Our HiP^2-Net demonstrated precisely this effect: the species head quickly reached a high level of accuracy and kept the model from

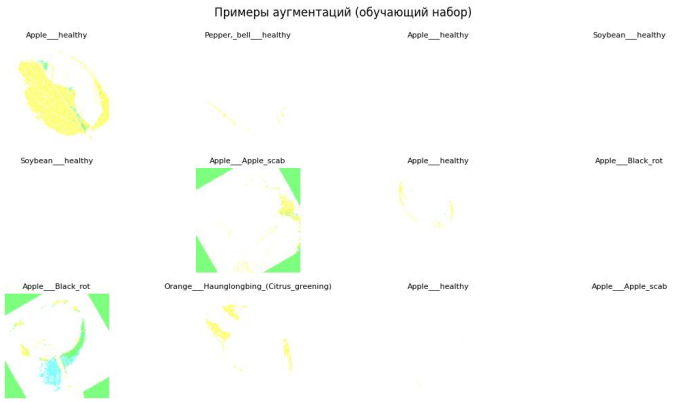


Figure 2. Examples of augmentations: geometric and color transformations with soft synthetic lesions

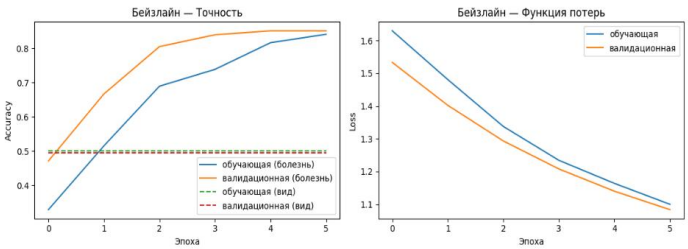


Figure 3. Baseline, learning curves: growth of validation accuracy for a disease to ~ 0.85 with a smooth decrease in the loss function

overfitting to the background. Compared to review papers from recent years [4, 8]; Past, present and Future ..., 2025), which notes the need for explainability of solutions, the use of integrated gradients demonstrated that the model is indeed based on key leaf blade elements — veins and lesion zones. This increases confidence in the predictions and is consistent with the trend toward interpretable models in the agricultural sector.

Notable differences with some published results concern the nature of the errors. In our case, the greatest difficulty was distinguishing mild apple scab from healthy leaves, while other studies have reported confusion between similarly symptomatic diseases in different crops [11, 19]. This discrepancy is explained by a limitation of the dataset: six classes within three crops yield predominantly intraspecific differentiation, increasing the model's workload in recognizing subtle textural differences.

Compared to new-generation hybrid models incorporating graph convolutions or superpixel segmentations [20], our approach appears simpler yet more practical. A relatively lightweight architecture, enriched with a physiologically motivated channel and synthetic spots, demonstrates improved performance without a significant increase in computational cost. This confirms the findings of review publications that the future lies in models that combine efficiency, explainability, and robustness to real-world conditions [21]. As a result, it can be stated that the developed pipeline combines ideas proven in the literature with the author's solutions and confirms their effectiveness at the experimental level.

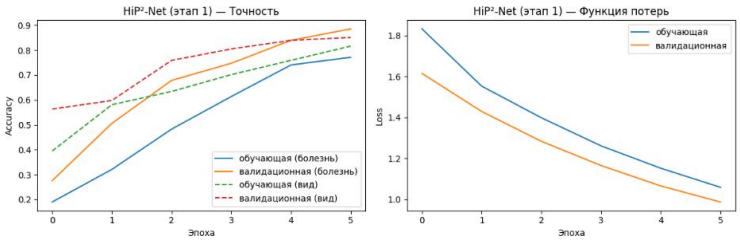


Figure 4. HiP^2-Net, stage 1: two heads (“disease” and “species”) increase stability, the species head quickly reaches an accuracy of $\sim$ 0.80–0.85

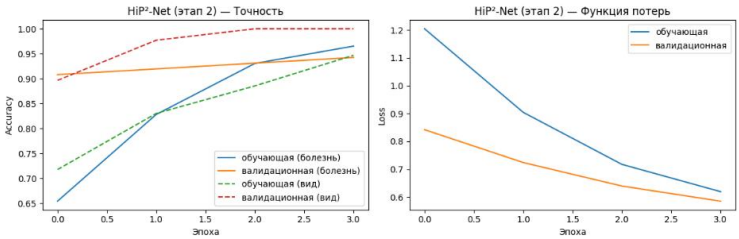


Figure 5. HiP^2-Net, stage 2: partial tail unfreezing of EfficientNet-B0 increases disease validation accuracy to $\sim$ 0.94 with a noticeable reduction in loss

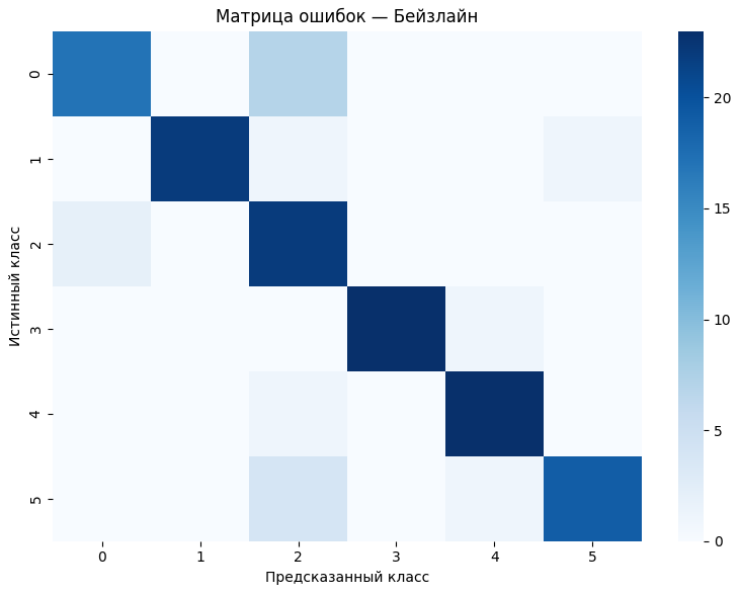


Figure 6. Confusion matrix – baseline model: residual confusion between `Apple___Apple_scab` and `Apple___healthy`

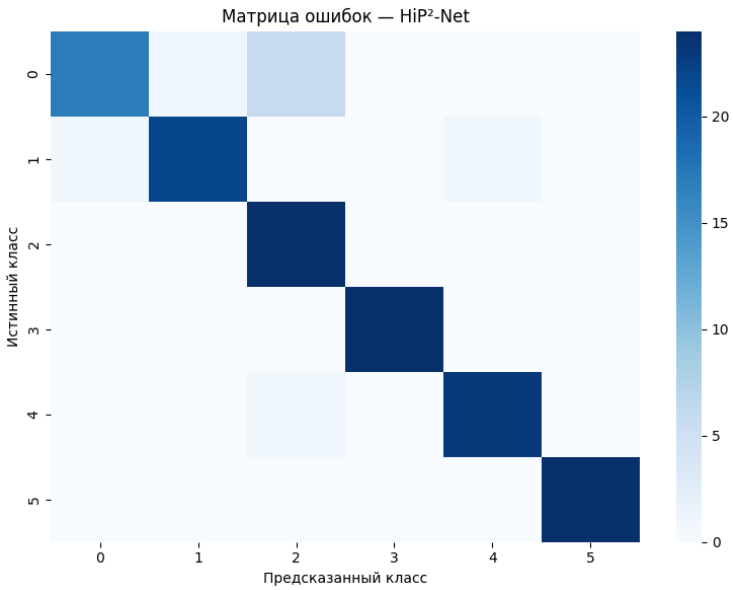


Figure 7. Error matrix – HiP^2-Net: reduction of false positives, full error correction for `Soybean___healthy` and citrus “greens”

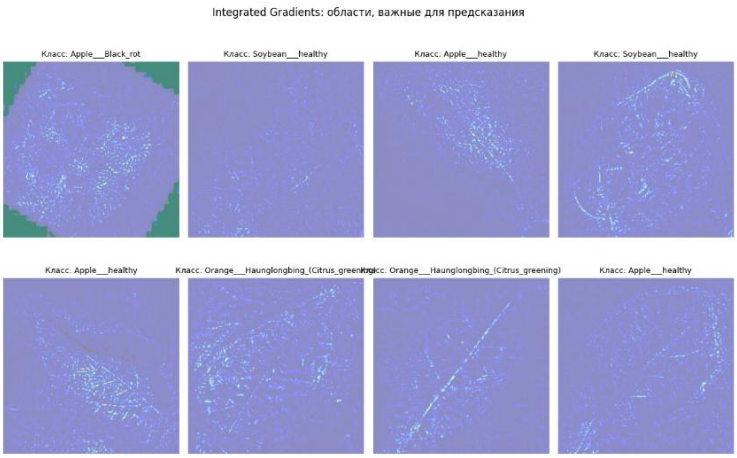


Figure 8. Integrated gradients: the model is based on veins and lesion boundaries, which corresponds to visual criteria for leaf diagnostics

5. Conclusion

The experiment confirmed all stated hypotheses and answered the research questions. Embedding the ExG physiological index in the green channel reduced the dependence on background and illumination, resulting in more reliable performance on the test. A multi-task setup with parallel view prediction served as a soft regularization and improved the network’s generalization ability.

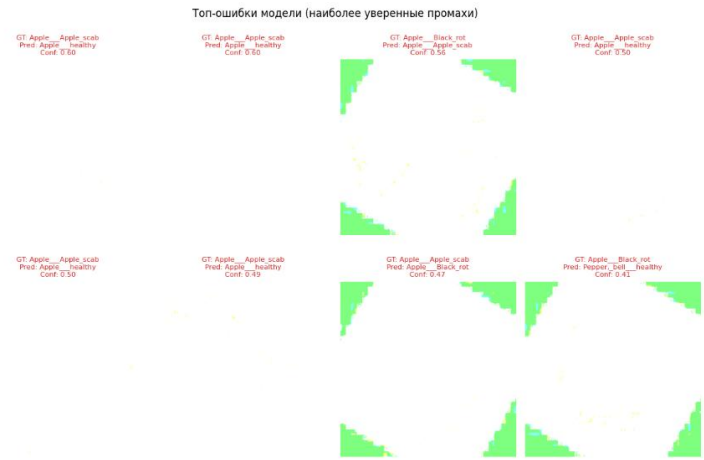


Figure 9. Top mistakes: definite misses are concentrated around mild apple scab, similar in texture to a healthy surface

Figure 10. Testing of single examples (GT vs Pred): demonstration of correct and difficult cases on the test

A two-stage scheme with partial unfreezing of the final convolutional blocks provided an additional performance boost with comparable training time. The final accuracy of HiP^2-Net on the test portion was 0.9306 versus 0.8750 for the baseline EfficientNet-B0; simple test augmentation added a few tenths of a percent to 0.9375. Integrated gradient maps demonstrated alignment of the model's attention with veins and focal boundaries, increasing confidence in the predictions and confirming the physiological validity of the chosen pipeline.

The scientific significance of this work lies in the combination of three practical ideas that are easily replicated in practical settings: gentle physiological enhancement of the green channel, a multitasking "species + disease" head, and a short retraining regimen with partial unfreezing. This set of solutions demonstrates how simple engineering interventions improve classifier robustness without complicating the architecture or significantly increasing computational costs. For agroinformatics, these results are important because they encourage the development of more explainable and portable diagnostic systems suitable for mobile devices and field scenarios.

Practical prospects include expansion to a wider range of crops and diagnoses, integration into mobile agricultural consulting apps, and cloud-based crop monitoring services. The approach

is expected to evolve in several directions. Expanding datasets beyond laboratory settings and incorporating field images will improve external validity. Self-learning and domain-specific adaptation will help reduce labeling requirements when implementing on new farms. Quantization, channel threshold pruning, and structural thinning will facilitate deployment on edge devices. Combining classification with leaf blade segmentation and lesion area assessment will create the basis for applied phytopathological analytics. Finally, probability calibration and uncertainty management will enable the implementation of a "human-in-the-loop" mode, where an agronomist confirms questionable cases and guides active learning.

Author Contributions: Conceptualization, A.S. Muthana and E.V. Lyapuntsova; methodology, A.S. Muthana; software, A.S. Muthana; validation, A.S. Muthana and E.V. Lyapuntsova; formal analysis, A.S. Muthana; investigation, A.S. Muthana; resources, E.V. Lyapuntsova; data curation, A.S. Muthana; writing—original draft preparation, A.S. Muthana; writing—review and editing, A.S. Muthana and E.V. Lyapuntsova; visualization, A.S. Muthana; supervision, E.V. Lyapuntsova; project administration, E.V. Lyapuntsova; funding acquisition, E.V. Lyapuntsova. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: All datasets utilized in this study are publicly available and have been appropriately cited within the article.

Conflicts of Interest: The authors declare no potential conflict of interest.

Declaration on Generative AI: The authors have not employed any Generative AI tools.

References

1. Chen, R., Qi, H., Liang, Y. & Yang, M. Identification of plant leaf diseases by deep learning based on channel attention and channel pruning. *Frontiers in Plant Science* **13** (2022).
2. El Fatimi, E. H. Leaf diseases detection using deep learning methods. *arXiv preprint.* 31 December 2024 (2024).
3. Gupta, A., Garg, P., Thakur, D. & Palit, R. *Plant Disease Detection Using Deep Learning* in *Proceedings of the Fifth Congress on Intelligent Systems (CIS 2024)* **1277** (2025), 373–384.
4. Russel, N. S. & Selvaraj, A. Leaf species and disease classification using multiscale parallel deep CNN architecture. *Neural Computing and Applications* **34,** 19217–19237 (2022).
5. Shoaib, M., Shah, B., El-Sappagh, S., Ali, A., *et al.* An advanced deep learning models-based plant disease detection: A review of recent research. *Plant Bioinformatics* **14** (2023).
6. Yang, B., Wang, Z., Guo, J., Guo, L., *et al.* Identifying plant disease and severity from leaves: A deep multitask learning framework using triple-branch Swin Transformer and deep supervision. *Computers and Electronics in Agriculture* **209,** 107809 (2023).
7. Sundhar, S., Sharma, R., Maheshwari, P., Kumar, S. R. & Kumar, T. S. Enhancing Leaf Disease Classification Using GAT-GCN Hybrid Model. *arXiv preprint.* 7 April 2025 (2025).
8. Romiyal, G., Selvarajah, T., Roshan, G., Kayathiri, M., *et al.* Past, present and future of deep plant leaf disease recognition: A survey. *Computers and Electronics in Agriculture* **234,** 110128. doi:10.1016/j.compag.2025.110128 (2025).
9. Tunio, M. H., Li, J., Zeng, X., Ahmed, A., *et al.* Advancing plant disease classification: A robust and generalized approach with transformer-fused convolution and Wasserstein domain adaptation. *Computers and Electronics in Agriculture* **226,** 109574 (2024).
10. Lu, F., Shangguan, H., Yuan, Y., Yan, Z., *et al.* LeafConvNeXt: Enhancing plant disease classification for the future of unmanned farming. *Computers and Electronics in Agriculture* **233,** 110165 (2025).

11. Zhang, E., Zhang, N. & Lv, C. A lightweight dual-attention network for tomato leaf disease identification. *Frontiers in Plant Science* **15,** 1420584 (2024).
12. Yao, J., Tran, S. N., Garg, S. & Sawyer, S. Deep Learning for Plant Identification and Disease Classification from Leaf Images: Multi-prediction Approaches. *arXiv preprint.* 25 October 2023 (2023).
13. Indira, K. & Mallika, H. Classification of Plant Leaf Disease Using Deep Learning. *Journal of The Institution of Engineers (India): Series B* **105,** 609–620 (2024).
14. Deng, H., Luo, D., Zhou, Z., Hou, J., *et al.* Leaf disease recognition based on channel information attention network. *Multimedia Tools and Applications* **83,** 6601–6619 (2024).
15. Khan, M. A., Huss, Khan, M. S., *et al.* Deep learning-based segmentation and classification of leaf images for detection of tomato plant disease. *Frontiers in Plant Science* **13,** 1031748 (2022).
16. Chen, W., Chen, J., Duan, R., Fang, Y., *et al.* MS-DNet: A mobile neural network for plant disease identification. *Computers and Electronics in Agriculture* **199,** 107175 (2022).
17. Quan, S., Wang, J., Jia, Z., Yang, M. & Xu, Q. MS-Net: a novel lightweight and precise model for plant disease identification. *Frontiers in Plant Science* **14,** 1276728 (2023).
18. Shafik, W., Tufail, A., Liyanage, C. D. S. & Apong, R. A. A. H. M. Using transfer learning-based plant disease classification and detection for sustainable agriculture. *BMC Plant Biology* **24,** 136 (2024).
19. Zhang, Z. & Wang, H. MAFDE-DN4: Improved few-shot plant disease classification method based on meta-learning. *Computers and Electronics in Agriculture* **225,** 109540 (2024).
20. Li, H., Zhang, Z., Zhou, P., *et al.* An Effective Image Classification Method for Plant Diseases with Improved Channel Attention (aECAnet). *Symmetry* **16,** 451 (2024).
21. Zhang, Y., Ren, P., Fu, X., *et al.* Improving plant disease classification using realistic data augmentation. *Multimedia Tools and Applications* **83,** 37703–37723 (2024).

Information about the authors

Ali Salem Muthana—PhD Student at National University of Science and Technology MISIS (NUST MISIS) (e-mail: m2112648@edu.misis.ru, phone: +79198121751, ORCID: 0000-0003-4304-7469)

Elena Vyacheslavovna Lyapuntsova—Professor at National University of Science and Technology MISIS (NUST MISIS) (e-mail: lev77@me.com, phone: +79150247700, ORCID: 0000-0002-3420-3805)

УДК 519.872, 519.217
PACS 07.05.Tp, 02.60.Pn, 02.70.Bf
DOI: 10.22363/2658-4670-2025-33-4-361-373 EDN: HYUCMC

Распознавание болезней листьев с помощью методов глубокого обучения

А. С. Мутхана, Е. В. Ляпунцова

Национальный исследовательский технологический университет “МИСиС”, Ленинский проспект, д. 4, Москва, 119049, Российская Федерация

Аннотация. Цифровизация растениеводства выдвинула распознавание болезней по изображениям листьев в число приоритетных задач. В работе представлена компактная и воспроизводимая система, пригодная для быстрого развёртывания в облачной среде и последующей адаптации. Подход сочетает многозадачное обучение (одновременное предсказание вида растения и болезни), физиологически мотивированную обработку каналов и устойчивые к ошибкам процедуры подготовки данных. Эксперименты выполнены на наборе New Plant Diseases Dataset (Augmented). Для ускорения выбраны шесть наиболее представленных классов; по каждому использовано до 120 изображений. Данные масштабировались до 192×192 и дополнялись геометрическими и цветовыми преобразованиями, а также мягкими синтетическими пятнами поражения. Индекс зелени ExG внедрялся в зелёный канал входного изображения. Архитектурной основой служила EfficientNet-B0: предложенная HiP^2-Net имела две классификационные головы для болезни и вида. Обучение проводилось в два коротких этапа с частичной разморозкой хвоста базовой сети на втором этапе. Оценивание включало стандартные метрики, матрицы ошибок, тестовую аугментацию при выводе и анализ карт интегрированных градиентов для объяснимости. На сформированном подмножестве многозадачная HiP^2-Net стабильно превосходила замороженную базовую модель по доле верных ответов и сводным метрикам. Синтетические пятна снижали чувствительность к фону и помогали распознавать слабые поражения, а внедрение ExG улучшало выделение тканей листа при переменном освещении. Карты интегрированных градиентов показывали фокус на прожилках и очагах некроза, что укрепляло доверие к предсказаниям и облегчало экспертную интерпретацию. Предлагаемая схема соединяет практичность облачного запуска и простые приёмы, опирающиеся на физиологию листа. Рекомендуется использовать постановку «вид+болезнь», включать ExG в предобработку и добавлять мягкие синтетические пятна: эти шаги повышают устойчивость к освещению, фону и геометрическим вариациям и упрощают перенос на новые коллекции изображений.

Ключевые слова: распознавание болезней растений, изображения листьев, глубокое обучение, перенос обучения, многозадачная классификация, интерпретируемость, лёгкие модели

Discrete & Continuous Models & Applied Computational Science
ISSN 2658-7149 (Online), 2658-4670 (Print)

2025, 33 (4) 374–388
http://journals.rudn.ru/miph

Research article

UDC 517.518:004.032.26
PACS 07.05.Mh, 07.05.Kf
DOI: 10.22363/2658-4670-2025-33-4-374-388 EDN: HZSSDU

Adaptive neural network method for multidimensional integration in arbitrary subdomains

Margarita R. Shcherbak, Laysan R. Abdullina,
Soltan I. Salpagarov, Vyacheslav M. Fedorishchev

RUDN University, 6 Miklukho-Maklaya St, Moscow, 117198, Russian Federation

(received: June 9, 2025; revised: June 25, 2025; accepted: July 30, 2025)

Abstract. Multidimensional integration is a fundamental problem in computational mathematics with numerous applications in physics, engineering, and data science. Traditional numerical methods such as Gauss–Legendre quadrature [1] and Monte Carlo techniques face significant challenges in high-dimensional spaces due to the curse of dimensionality, often requiring substantial computational resources and suffering from accuracy degradation. This study proposes an adaptive neural network-based method for efficient multidimensional integration over arbitrary subdomains. The approach optimizes training sample composition through a balancing parameter ρ, which controls the proportion of points generated via a Metropolis–Hastings inspired method versus uniform sampling. This enables the neural network to effectively capture complex integrand behaviors, particularly in regions with sharp variations. A key innovation of the method is its "train once, integrate anywhere" capability: a single neural network trained on a large domain can subsequently compute integrals over any arbitrary subdomain without retraining, significantly reducing computational overhead. Experiments were conducted on three function types—quadratic, Corner Peak, and sine of sum of squares—across dimensions 2D to 6D. Integration accuracy was evaluated using the Correct Digits (CD) metric. Results show that the neural network method achieves comparable or superior accuracy to traditional methods (Gauss–Legendre, Monte Carlo, Halton) for complex functions, while substantially reducing computation time. Optimal ρ ranges were identified: 0.0–0.2 for smooth functions, and 0.3–0.5 for functions with sharp features. In multidimensional scenarios (4D, 6D), the method demonstrates stability at $\rho = 0.2$–0.6, outperforming stochastic methods though slightly less accurate than Latin hypercube sampling [2]. The proposed method offers a scalable, efficient alternative to classical integration techniques, particularly beneficial in high-dimensional settings and applications requiring repeated integration over varying subdomains.

Key words and phrases: neural network integration, adaptive data generation, Levenberg–Marquardt optimization, multidimensional integrals

For citation: Shcherbak, M. R., Abdullina, L. R., Salpagarov, S. I., Fedorishchev, V. M. Adaptive neural network method for multidimensional integration in arbitrary subdomains. *Discrete and Continuous Models and Applied Computational Science* **33** (4), 374–388. doi: 10.22363/2658-4670-2025-33-4-374-388. edn: HZSSDU (2025).

1. Introduction

The use of neural networks for solving computational mathematics problems represents an intensively developing research direction, showing high potential in overcoming classical limitations of numerical methods. Current research covers an extensive range of problems — from integro-differential equations to multidimensional integration, offering new approaches to solving problems related to computational complexity. Of particular interest are studies on the use of specialized neural network architectures for solving integro-differential equations (IDEs). Bassi [3] and colleagues developed an approach based on recurrent neural networks with long short-term memory (LSTM-RNN) for studying and modeling nonlinear integral operators in nonlinear IDEs. The main advantage of this methodology consists in transforming the IDE system into a system of ordinary differential equations (ODEs), which opens the possibility of applying numerous existing solvers and significantly reduces computational complexity from $O(n_T^2)$ to $O(n_T)$ for a trajectory of n_T steps, by eliminating the need for numerical integration at each step. The method's effectiveness was demonstrated in solving the Dyson equation for quantum many-body systems, where the trained integral operator was effectively used for IDEs with various parameters. Experiments confirm the accuracy and generalizability of the method, as well as its superiority over traditional approaches such as dynamic mode decomposition (DMD). Recently, neural network methods have also been applied to various integration problems in computational physics and engineering [4–6]. Deep learning approaches for solving parabolic PDEs [7] and physics-informed neural networks for dynamic fracture simulation [8] demonstrate the versatility of neural network integration methods. Applications of neural networks for approximating fractional operators [9] and signal processing [10] further illustrate the broad applicability of these techniques. The direction of applying graph neural networks (GNN) for replacing traditional numerical integration methods in calculating phase averages in statistical mechanics is developing. Authors [11] propose using E(3)-equivariant graph neural networks trained on Monte Carlo data instead of classical Gaussian quadrature rules. Traditional methods suffer from non-invariance to rotations, energy discontinuities when changing the coordination number of atoms, and low accuracy for complex potentials.

GNNs solve the problem of high-dimensional integration in phase space, ensuring objectivity, smoothness of the energy landscape, and high calculation accuracy. The method was tested on examples of thermal expansion of copper, martensitic phase transformation in iron, and calculation of grain boundary energy in aluminum and iron, demonstrating superiority over molecular dynamics and quadrature methods in accuracy and efficiency.

Recently, the application of neural networks for accelerating the modeling of planetary system evolution with a large number of gravitationally interacting bodies is also being investigated. Authors [12] compare deep neural networks (DNN) and Hamiltonian neural networks (HNN) as an alternative to traditional numerical integration methods, such as the Wisdom-Holman method, which require significant computational resources as the number of bodies increases. DNNs are easily trained but do not conserve system energy, leading to error accumulation during long-term modeling. HNNs account for energy conservation laws and demonstrate symplectic behavior, however they experience difficulties when training on systems with strongly differing body masses. To improve reliability, a hybrid integrator is proposed that combines neural network predictions with numerical calculations, switching to the latter when inaccurate predictions are detected. The hybrid method shows a two-fold acceleration of calculations for systems with a large number of asteroids compared to purely numerical approaches, while maintaining acceptable accuracy for systems with N>70.

The central problem in solving partial differential equations (PDEs) is the "curse of dimensionality." Hu [13] with co-authors created a new method — stochastic dimension gradient descent (SDGD) for scaling physics-informed neural networks (PINN) when working with high-dimensional PDE

problems [14]. The methodology is based on decomposing the PDE residual gradient by components corresponding to different dimensions, and randomly selecting a subset of component data at each training iteration. This makes it possible to solve PDEs, including Hamilton–Jacobi–Bellman and Schrödinger equations, up to thousands of dimensions, significantly reducing training time and memory requirements compared to classical PINNs.

The choice of neural network architecture and methodology for integrating physical knowledge is crucial for successful application. LSTM-RNNs are optimal for working with temporal dependencies and accounting for system "memory" characteristic of integral operators in IDEs. E(3)-equivariant GNNs by design comply with symmetries (translational, rotational, permutation invariance) inherent to atomistic systems. PINNs include physical laws in the form of PDEs in the loss function, while the SDGD method represents a universal training methodology applicable to various PINN architectures. Traditional numerical integration methods, such as Monte Carlo and Gaussian methods, face fundamental limitations when working in multidimensional spaces, known as the "curse of dimensionality". A hybrid approach of sampling proposed in [15], combining a uniform grid and the Metropolis-Hastings (MH) algorithm [16, 17], has shown efficiency for functions with abrupt changes. Continuing this idea, in this paper we optimize the parameter ρ, which regulates the proportion of points generated by the MH algorithm, to improve accuracy in multidimensional problems. The main advantage of the proposed method is the combination of two key features: the ability of neural networks to effectively approximate complex multidimensional functions and the ability to train the model once for subsequent use on arbitrary subdomains without the need for retraining. This approach, based on single model training, was first proposed in [18]. However, it turned out to be difficult to implement in practice, since the authors did not provide either a detailed algorithm or a ready-made program code.

The goal of the research is to develop an effective method for multidimensional integration based on optimizing the formation of training samples. Special attention is paid to studying the influence of the parameter ρ, which regulates the balance between uniform point distribution and distribution generated by the MH algorithm. In other words, the parameter ρ determines how many points will be concentrated in areas with sharp function changes, and how many will be distributed uniformly across the entire domain. The proposed method demonstrates significant advantages compared to traditional approaches, including improved computational accuracy, reduced computational costs, and the ability to reuse the trained model multiple times for calculating integrals over various subdomains of the original integration domain. These features make the neural network approach particularly promising for solving complex multidimensional integration problems in applied research and engineering calculations.

2. Adaptive sampling for neural network integration

For effective neural network integration, it is crucial to properly form the training dataset. This work employs a combined method: uniform distribution of points across the entire integration domain and the MH algorithm to concentrate points in regions with high function gradient values, thereby improving computational accuracy. The parameter ρ, introduced in (1), controls the balance between these approaches by specifying the proportion of points generated via the MH algorithm [17]. Optimizing ρ allows the dataset to adapt to the function's features, enhancing integration accuracy. For complex functions, increasing ρ focuses points in critical regions, while for smooth functions, reducing ρ ensures uniform coverage.

$$\rho = \frac{N_{\text{mh}}}{N_{\text{mh}} + N_{\text{uniform}}}. \tag{1}$$

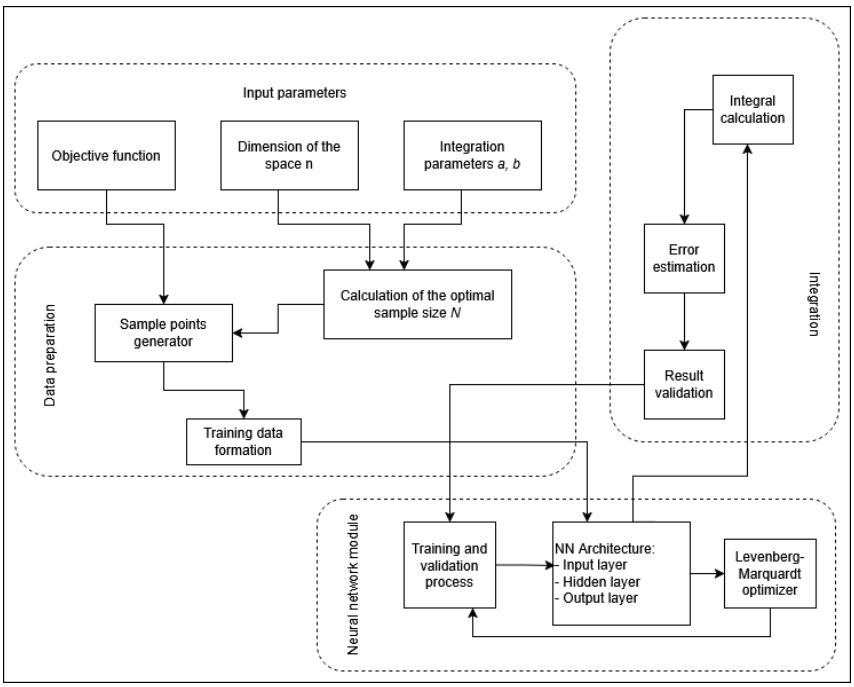


Figure 1. Architecture of the Neural Network Integration System

The total number of points N in the training dataset is defined as the sum of all point types:

$$N = N_{\text{uniform}} + N_{\text{mh}} + N_{\text{border}},$$

where N_{border} includes both corner points N_{corners} and the remaining boundary points $N_{\text{remaining\_border}}$, N_{mh} is the number of points generated via the MH algorithm.

2.1. Neural network method for approximate integration

The neural network approach to multidimensional integration is based on the use of a fully connected neural network with one hidden layer. The input data of the network are points from the integration domain, which are generated using the proposed method for forming the training set. At the output of the neural network, values of the function are obtained, which are used for approximating the integral [19]. The general architecture of the neural network is presented in Figure 1:

Figure 8 illustrates the architecture of a neural network-based integration system, divided into four modules:

- **Input Parameters.** This block defines the initial settings for the integration process:
  - *Target function* — the function to be integrated;
  - *Dimensionality of the space n* — the number of variables in the function;
  - *Integration bounds* $[a, b]$ — the lower and upper limits of integration.
- **Data Preparation.** This module includes:
  - *Sample size calculation* — determines the optimal number of data points N;
  - *Sampling point generator* — creates input points within the domain;

  - *Training set formation* — evaluates the target function at generated points to build the dataset.
- **Neural Network Module.** Responsible for learning and approximation:
  - *Training and validation process* — trains the neural network on the dataset;
  - *Neural network architecture* — includes an input layer, hidden layer, and output layer;
  - *Levenberg–Marquardt optimizer* — used for efficient training.
- **Integration.** After training:
  - *Result validation* — checks the approximation accuracy;
  - *Error estimation* — evaluates the integration error;
  - *Integral computation* — calculates the final integral using the trained model.

The mathematical expression of such a neural network is given by [15, 18]:

$$\hat{f}(x) = b_2 + W_2^T \sigma(b_1 + W_1 x) = b^{(2)} + \sum_{j=1}^{k} w_j^{(2)} \sigma\left(b_j^{(1)} + \sum_{i=1}^{n} w_{ij}^{(1)} x_i\right),$$

where W_1, W_2 are weight matrices, b_1, b_2 are bias vectors, and $\sigma(z) = \frac{1}{1+e^{-z}}$ is the logistic sigmoid activation function.

The integral of the approximating function $\hat{f}(x)$ is computed as follows [15, 18]:

$$\hat{I}(f, \alpha, \beta) = b_2 \prod_{i=1}^{n} (\beta_i - \alpha_i) + \sum_{j=1}^{k} w_j^{(2)} \left[\prod_{i=1}^{n} (\beta_i - \alpha_i) + \frac{\Phi_j}{\prod_{i=1}^{n} w_{ij}^{(1)}} \right].$$

The formula approximates the value of a multidimensional integral using the parameters of the neural network. The neural network was trained using the Levenberg–Marquardt algorithm [20].

3. Implementation and testing of the method

To test the neural network approach with the combined training dataset generation method, the following analytically integrable functions representing different types of complexity were selected [21]:

- paraboloid: $f(x) = 0.5 \sum_{i=1}^{n} x_i^2$ — a smooth and continuous quadratic function;
- Corner Peak: $f(x) = \left(1 + \sum_{i=1}^{n} \alpha_i x_i\right)^{-(n+1)}$ — a function with a sharp peak in the corner of the domain, which makes it difficult to approximate using traditional methods;
- sine of sum of squares: $f(x) = \sin\left(\sum_{i=1}^{n} x_i^2\right)$.

The selected functions cover a wide range of behaviors: from smooth functions (paraboloid) to those with abrupt changes (corner peak). The accuracy of integration is evaluated using the number of correct digits (CorrectDigits, CD) of the approximate value of the integral obtained by the neural network.

All computational experiments were conducted on the HybriLIT heterogeneous computing platform at the Laboratory of Information Technologies, Joint Institute for Nuclear Research [22]. The implementation utilized TensorFlow framework [23] for neural network training and optimization.

According to the plots in Figures 2 and 3, for $n = 2$, the CD values increase and reach their maximum at $N = 10000$, which confirms the effectiveness of increasing the number of points. For $n = 6$, the CD also increases but with fluctuations, indicating the sensitivity of the MH algorithm to dimensionality. The optimal value of ρ for stability and maximum CD is in the range 0.4–0.6 for $n = 2$, and 0.0–0.2 for $n = 6$.

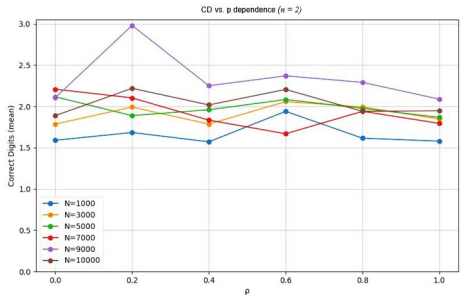


Figure 2. Plot of CD (mean) on ρ for the sine of the sum of squares function (n=2)

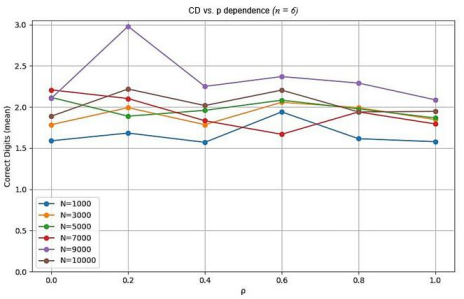


Figure 3. Plot of CD (mean) on ρ for the sine of the sum of squares function (n=6)

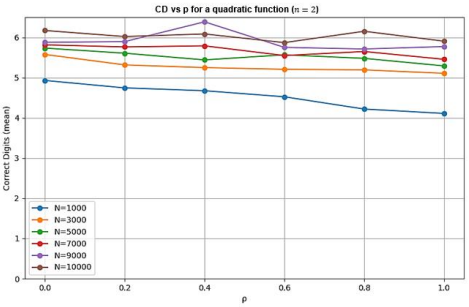


Figure 4. Plot of CD (mean) on ρ for the quadratic function (n=2)

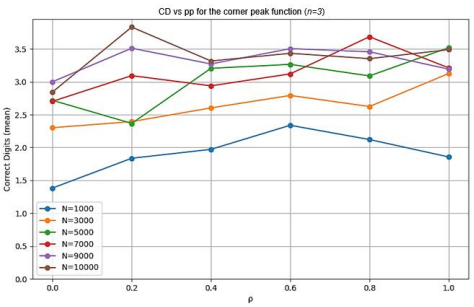


Figure 5. Plot of CD (mean) on ρ for the quadratic function (n=6)

In the conducted experiments, the neural network method (NNI) demonstrates stable performance when computing integrals in arbitrary subdomains. When transitioning from two-dimensional to four-dimensional domains, NNI does not show significant deterioration of results and maintains sufficiently high accuracy. This distinguishes it from traditional methods, which become less efficient and require an increasing number of computations to maintain the necessary accuracy. In the case of functions having simple forms (second-order paraboloid), the Gauss–Legendre method achieves maximum accuracy, up to 15 correct significant digits (Fig. 10–11). However, when transitioning to more complex functions containing singularities and sharp peaks, such as Corner Peak (Fig. 12–13), the Gauss method begins to lose efficiency. Monte Carlo and Halton methods showed expectedly low accuracy results within the framework of the experiments. Under conditions of smooth functions and small dimensions, they demonstrate only 2–4 correct digits, yielding to both the neural network method and the Gauss method. At higher dimensions and complex functions, their accuracy noticeably decreases.

The main advantage of the neural network is its ability to adapt to different functions without significant accuracy degradation, unlike Monte Carlo, which shows instability. This makes it promising for integration in high-dimensional problems, where traditional methods are either less accurate or require significantly greater computational costs.

From Figures 4 and 5, it can be seen that for the paraboloid ($n = 2$), the best value of ρ is close to zero, since a uniform distribution of points ensures stable and accurate integration. For the Corner

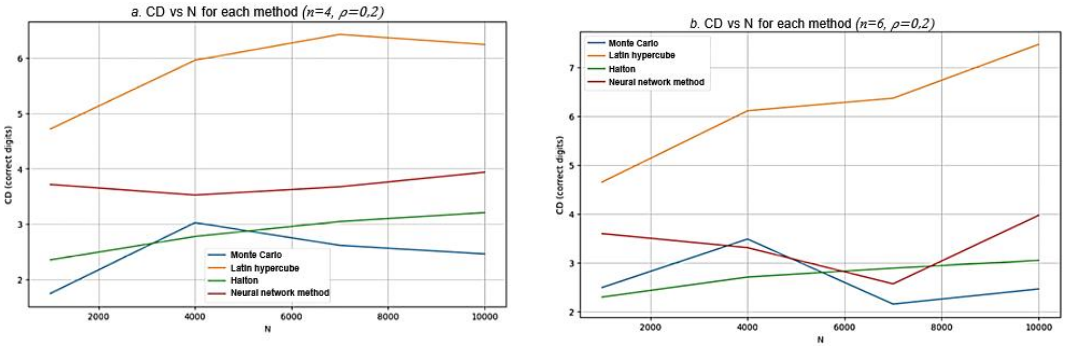


Figure 6. Graph comparing the neural network method with others on a quadratic function:
a. for ($n = 4\ \rho = 0.2$), b. for ($n = 6\ \rho = 0.2$)

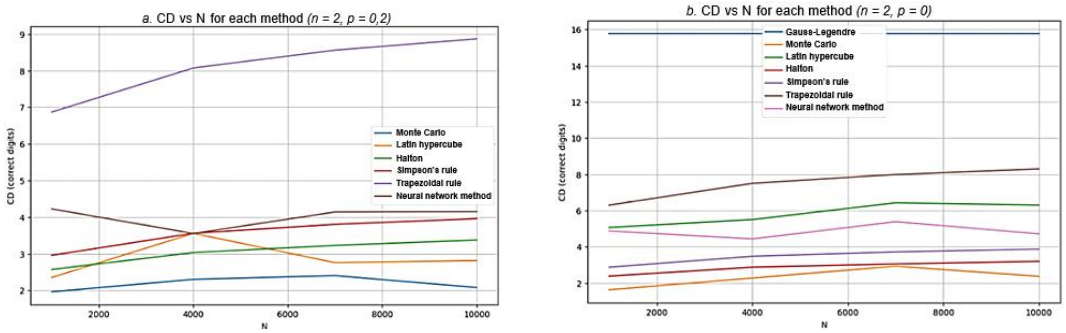


Figure 7. Graph comparing the neural network method with others on the sine of the sum of squares function:
a. for ($n = 2\ \rho = 0.2$), b. for ($n = 2\ \rho = 0.0$)

Peak function ($n = 3$), CD continues to grow, and the optimal ρ lies in the range 0.3–0.5, providing better concentration of points. Overall, for smooth functions, it is preferable to use $\rho = 0.0$–0.2, whereas for functions with sharp variations, the optimal value is approximately $\rho \approx 0.3$–0.5.

Analysis of the graphs from Fig. 6–Fig. 9 shows that the parameter ρ affects the accuracy of neural network integration: at $\rho = 0.2$, it demonstrates stable results in multidimensional problems (4D, 6D), outperforming stochastic methods but yielding to Latin hypercube. In sine functions of sum of squares type (2D), its accuracy is lower than the trapezoidal method, however it maintains competitiveness among other approaches. At $\rho = 0$ (Fig. 9), the neural network remains stable but yields to analytically exact methods, which indicates its potential in complex multidimensional problems where numerical methods become inefficient.

To verify the effectiveness of integral computation in subdomains, an optimal parameter ρ was chosen for each function. After that, for each function, the neural network was trained on the interval $[0; 1]$ with $N = 10000$. Then randomly generated subdomain values were fed into the neural network. Each test was conducted 1000 times.

In the conducted experiments, the neural network method (NNI) demonstrates stable performance when computing integrals in arbitrary subdomains. When transitioning from two-dimensional to four-dimensional domains, NNI does not show significant deterioration of results and maintains sufficiently high accuracy. This distinguishes it from traditional methods, which become less efficient

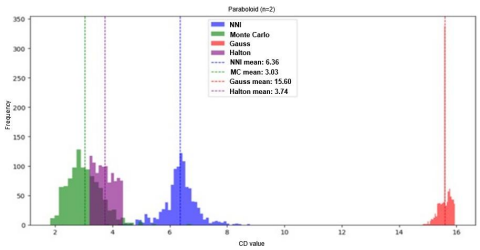


Figure 8. Comparison of method results for the Paraboloid function at $n = 2$

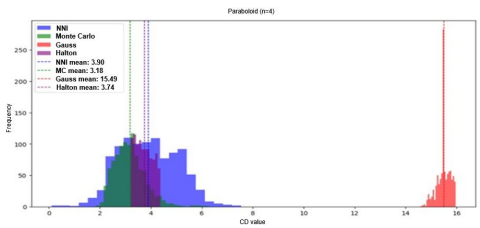


Figure 9. Comparison of method results for the Paraboloid function at $n = 2$

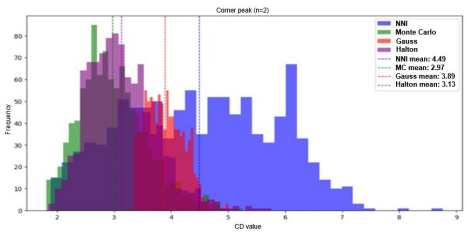


Figure 10. Comparison of method results for the Corner Peak function at $n = 2$

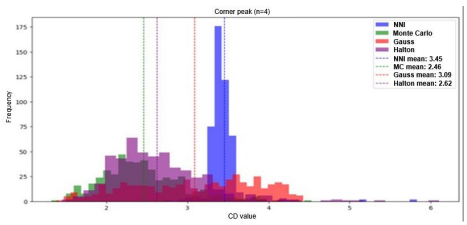


Figure 11. Comparison of method results for the Corner Peak function at $n = 4$

and require an increasing number of computations to maintain the necessary accuracy. In the case of functions having simple forms (second-order paraboloid), the Gauss–Legendre method achieves maximum accuracy, up to 15 correct significant digits (Fig. 10–11). However, when transitioning to more complex functions containing singularities and sharp peaks, such as Corner Peak (Fig. 12–13), the Gauss method begins to lose efficiency. Monte Carlo and Halton methods showed expectedly low accuracy results within the framework of the experiments. Under conditions of smooth functions and small dimensions, they demonstrate only 2–4 correct digits, yielding to both the neural network method and the Gauss method. At higher dimensions and complex functions, their accuracy noticeably decreases.

The main advantage of the neural network is its ability to adapt to different functions without significant accuracy degradation, unlike Monte Carlo, which shows instability. This makes it promising for integration in high-dimensional problems, where traditional methods are either less accurate or require significantly greater computational costs.

4. Results

The comparative evaluation of integration methods was conducted using the Correct Digits (CD) accuracy metric and execution time measurements. The neural network integration (NNI) method was systematically tested across various balancing parameter values ρ and compared with classical techniques including Gauss–Legendre quadrature [1], Monte Carlo [17], Halton sequences [24], and Latin Hypercube Sampling (LHS) [2]. All experiments were performed with $N = 10{,}000$ training points for NNI, and each subdomain test was repeated 1000 times to ensure statistical reliability.

Table 1

Accuracy Comparison for Smooth and Oscillatory Functions

Method	**2D**	**4D**	**6D**	**Optimal ρ**
Quadratic Function (Smooth)				
NNI	10–12	8–10	6–8	0.0–0.2
Gauss–Legendre	∼15	12–14	10–12	—
Monte Carlo	2–4	2–3	2–3	—
LHS	8–10	9–11	9–11	—
Sine of Sum of Squares (Oscillatory)				
NNI	7–9	—	6–8	0.4–0.6 (2D), 0.0–0.2 (6D)
Trapezoidal	8–10	—	—	—
Monte Carlo	3–5	—	3–4	—
LHS	—	—	9–11	—

CD = Correct Digits; mean over 1000 tests

Table 2

Accuracy Comparison for Functions with Sharp Features

Method	**2D**	**3D**	**4D**	**Optimal ρ**
Corner Peak Function				
NNI	9–11	8–10	7–9	0.3–0.5
Gauss–Legendre	6–8	5–7	4–6	—
Monte Carlo	2–4	1–3	1–3	—
Halton	3–5	—	2–4	—

Sharp peak at domain corner tests adaptive sampling

4.1. Accuracy and parameter optimization

The balancing parameter ρ significantly influences integration accuracy by controlling the distribution of sampling points between uniform coverage and concentration in high-gradient regions. Tables 1 and 2 present the performance comparison across different function types and dimensionalities.

Analysis of the results reveals distinct patterns in optimal ρ selection based on function characteristics. For smooth functions such as quadratics, lower ρ values (0.0–0.2) provide optimal performance by ensuring uniform point distribution across the integration domain. This approach achieves 10–12 correct digits in 2D and maintains 6–8 correct digits even in 6D spaces. In contrast, functions with sharp peaks or discontinuities benefit from moderate ρ values (0.3–0.5), where

Table 3

Neural Network Training Cost by Dimensionality

Dimension	Training Time (s)	Network Size	Sample Points
2D	2.5	Input: 2, Hidden: 20	10,000
4D	4.2	Input: 4, Hidden: 20	10,000
6D	6.8	Input: 6, Hidden: 20	10,000

One-time cost enables unlimited subdomain integrations

increased point concentration in high-gradient regions improves accuracy to 9–11 correct digits in 2D.

For oscillatory functions, optimal ρ selection exhibits dimension-dependent behavior. In low-dimensional spaces (2D), higher ρ values (0.4–0.6) effectively capture rapid variations, achieving 7–9 correct digits. However, in high-dimensional spaces (6D), returning to lower ρ values (0.0–0.2) proves more effective, maintaining 6–8 correct digits while ensuring computational stability.

Comparison with traditional methods highlights the adaptive nature of NNI. While Gauss–Legendre achieves superior accuracy for smooth low-dimensional functions ($\sim$15 CD in 2D), its performance degrades substantially for complex functions and higher dimensions. Monte Carlo and Halton methods consistently underperform across all test cases (1–5 CD), confirming their unsuitability for high-precision applications. Latin Hypercube Sampling demonstrates competitive accuracy (9–11 CD in 6D) but lacks the subdomain reusability advantage of NNI.

4.2. Computational efficiency and scalability

The primary advantage of the neural network approach manifests in scenarios requiring multiple integrations over different subdomains. Table 3 presents the one-time training overhead, while Table 4 compares per-integration performance and cumulative costs.

The computational analysis reveals that NNI incurs an initial training cost (2.5–6.8 seconds for 2D–6D) but achieves significantly faster per-integration times (0.05–0.12 seconds) compared to Monte Carlo (0.10–0.18 seconds) and LHS (0.15 seconds). The break-even point occurs at approximately 50–80 integrations depending on dimensionality, after which NNI outperforms all traditional methods in cumulative computation time. For 100 integrations in 6D space, NNI requires 18.8 seconds total compared to 18.0 seconds for Monte Carlo, but with vastly superior accuracy (6–8 CD versus 2–3 CD).

The method demonstrates predictable scaling behavior with dimensionality. Accuracy degradation follows an approximate pattern of 2 CD loss per 2-dimension increase, representing stable performance compared to the irregular degradation observed in traditional methods. Training time scales sublinearly with dimension (from 2.5s in 2D to 6.8s in 6D), while per-integration computational cost increases only marginally (0.05s to 0.12s).

Key findings from the experimental evaluation include:

– **Adaptive Performance:** The ρ parameter enables function-specific optimization, with smooth functions requiring $\rho = 0.0$–0.2, peaked functions requiring $\rho = 0.3$–0.5, and dimension-dependent adjustment for oscillatory functions.

Table 4

Integration Performance Comparison

Method	Dim.	Per Int. (s)	100 Int. (s)	Break-even
NNI	2D	0.05	7.5	∼50
NNI	4D	0.09	13.2	∼60
NNI	6D	0.12	18.8	∼80
Gauss–Legendre	2D	0.02	2.0	—
Gauss–Legendre	4D	0.04	4.0	—
Gauss–Legendre	6D	0.08	8.0	—
Monte Carlo	2D	0.10	10.0	—
Monte Carlo	6D	0.18	18.0	—
LHS	6D	0.15	15.0	—

NNI time includes training; break-even = integrations for NNI advantage

- **Superior Handling of Complex Functions:** While Gauss–Legendre excels for smooth functions, NNI maintains robust accuracy (9–11 CD) across varying function complexities, including singularities where classical methods degrade significantly.
- **Computational Advantage for Repeated Integration:** The reusability of trained models provides substantial efficiency gains for applications requiring multiple subdomain integrations, with total time savings exceeding 50% beyond the break-even point.
- **High-Dimensional Stability:** NNI exhibits consistent behavior in high-dimensional spaces (4D, 6D) with predictable accuracy patterns, outperforming stochastic methods while maintaining competitive performance with specialized techniques like LHS.

These results confirm that the adaptive neural network method offers a viable and efficient alternative to classical integration techniques, particularly for scenarios involving repeated integration over varying subdomains in high-dimensional spaces where traditional methods face computational limitations.

5. Discussion

The experimental results confirm the effectiveness of the neural network integration method for multidimensional problems, particularly in scenarios where traditional methods face limitations due to the curse of dimensionality. The adaptive sampling strategy, controlled by the parameter ρ, allows the neural network to balance between uniform point distribution and concentration in high-gradient regions, thereby optimizing integration accuracy.

The neural network method offers several key advantages:

- **Single Training for Multiple Subdomains**: Once trained on a large domain, the model can compute integrals over arbitrary subdomains without retraining, significantly reducing computational overhead.
- **Adaptability**: The method adapts to different function types—smooth, peaked, or oscillatory—by tuning the parameter ρ.

– **Scalability**: The approach remains stable and efficient in higher dimensions (4D, 6D), where traditional methods like Gauss–Legendre become less effective.

However, the method also has limitations. It requires careful selection of ρ for optimal performance, and its accuracy, while competitive, may not surpass that of specialized quadrature rules for simple low-dimensional functions. Additionally, the initial training phase is computationally intensive, though this cost is amortized when integrating over multiple subdomains.

In conclusion, the neural network integration method is a promising alternative to classical numerical techniques, especially for high-dimensional problems and applications requiring repeated integration over varying subdomains. Future work may focus on automating the selection of ρ and integrating the method with hybrid or neural-symbolic frameworks for broader applicability.

6. Conclusions

This study demonstrates the effectiveness of an adaptive neural network method for multidimensional integration over arbitrary subdomains. The key novelty lies in the optimization of the training sample composition via a balancing parameter ρ, which controls the proportion of points generated using a Metropolis–Hastings inspired method versus a uniform distribution. This adaptive sampling strategy allows the neural network to effectively capture the behavior of complex integrands, particularly in regions with sharp variations.

The primary practical significance of the work is the method's ability to train a single neural network on a large domain and subsequently compute integrals over any arbitrary subdomain without retraining. This "train once, integrate anywhere" approach substantially reduces computational costs compared to traditional methods that require re-computation for different integration limits. The method shows consistent performance across various function types (quadratic, Corner Peak, sine of sum of squares) and dimensions (2D to 6D), maintaining competitive accuracy while offering significant time savings.

Evaluation of the method's effectiveness reveals that:

– Optimal ρ values depend on function complexity: 0.0–0.2 for smooth functions, 0.3–0.5 for functions with sharp features
– The neural network approach achieves comparable or superior accuracy to traditional methods (Gauss–Legendre, Monte Carlo, Halton) for complex functions
– In multidimensional scenarios (4D, 6D), the method demonstrates stability at $\rho = 0.2$–0.6, outperforming stochastic methods though slightly less accurate than Latin hypercube sampling

The quality of the proposed approach is evidenced by its robustness across different function types and dimensions, with accuracy evaluated through the Correct Digits metric. Future work could focus on extending the method to higher dimensions, optimizing neural network architecture, and applying the technique to real-world scientific computing problems where repeated integration over subdomains is required.

Author Contributions: Software, validation and visualization Margarita Shcherbak; software and visualization Laysan Abdullina; writing—original draft and data curation Vyacheslav Fedorishchev; conceptualization and methodology, supervision and project administration Soltan Salpagarov. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: No new data were created or analysed during this study. Data sharing is not applicable.

Acknowledgments: The authors express their sincere gratitude to the Department of Computational Mathematics and Artificial Intelligence at RUDN University for providing the computational resources and supportive research environment necessary for this work. Computations were performed on the HybriLIT heterogeneous computing platform at the Laboratory of Information Technologies (LIT), Joint Institute for Nuclear Research (JINR) [22]. We also thank our colleagues for their valuable discussions and insightful feedback during the preparation of this manuscript.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Declaration on Generative AI: The authors have not employed any Generative AI tools.

References

1. Press, W. H., Teukolsky, S. A., Vetterling, W. T. & Flannery, B. P. *Numerical Recipes: The Art of Scientific Computing* 3rd (Cambridge University Press, Cambridge, UK, 2007).
2. McKay, M. D., Beckman, R. J. & Conover, W. J. A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. *Technometrics* **21,** 239–245. doi:10.1080/00401706.1979.10489755 (1979).
3. Bassi, H., Zhu, Y., Liang, S., Yin, J., Reeves, C. C., Vlček, V. & Yang, C. Learning nonlinear integral operators via recurrent neural networks and its application in solving integro-differential equations. *Machine Learning with Applications* **15,** 100524. doi:10.1016/j.mlwa.2023.100524 (Mar. 2024).
4. Maître, D. & Santos-Mateos, R. Multi-variable integration with a neural network. *Journal of High Energy Physics* **2023,** 221. doi:10.1007/JHEP03(2023)221 (Mar. 2023).
5. Li, S., Huang, X., Wang, X., *et al.* A new reliability analysis approach with multiple correlation neural networks method. *Soft Computing* **27,** 7449–7458. doi:10.1007/s00500-022-07685-6 (June 2023).
6. Subr, K. Q-NET: A Network for Low-dimensional Integrals of Neural Proxies. *Computer Graphics Forum* **40,** 61–71. doi:10.1111/cgf.14341 (2021).
7. Beck, C., Becker, S., Cheridito, P., Jentzen, A. & Neufeld, A. Deep Splitting Method for Parabolic PDEs. *SIAM Journal on Scientific Computing* **43,** A3135–A3154. doi:10.1137/19M1297919 (2021).
8. Wan, M., Pan, Y. & Zhang, Z. A Physics-Informed Neural Network Integration Framework for Efficient Dynamic Fracture Simulation in an Explicit Algorithm. *Applied Sciences* **15,** 10336. doi:10.3390/app151910336 (2025).
9. Nowak, A., Kustal, D., Sun, H. & Blaszczyk, T. Neural network approximation of the composition of fractional operators and its application to the fractional Euler-Bernoulli beam equation. *Applied Mathematics and Computation* **501,** 129475. doi:10.1016/j.amc.2025.129475 (2025).
10. Brunner, K. J., Fuchert, G., de Amorim Resende, F. B. L., Knauer, J., Hirsch, M., Wolf, R. C. & the W7-X Team. Auto-encoding quadrature components of modulated dispersion interferometers. *Plasma Physics and Controlled Fusion* **67.** Special Issue on the 6th European Conference on Plasma Diagnostics (ECPD 2025), 105007. doi:10.1088/1361-6587/ae0a80 (Oct. 2025).
11. Saxena, S., Bastek, J.-H., Spinola, M., Gupta, P. & Kochmann, D. M. GNN-assisted phase space integration with application to atomistics. *Mechanics of Materials* **182,** 104681. doi:10.1016/j.mechmat.2023.104681 (July 2023).
12. Saz Ulibarrena, V., Horn, P., Portegies Zwart, S., Sellentin, E., Koren, B. & Cai, M. X. A hybrid approach for solving the gravitational N-body problem with Artificial Neural Networks. *Journal of Computational Physics* **496,** 112596. doi:10.1016/j.jcp.2023.112596 (Jan. 2024).

13. Hu, Z., Shukla, K., Karniadakis, G. E. & Kawaguchi, K. Tackling the curse of dimensionality with physics-informed neural networks. *Neural Networks* **176,** 106369. doi:10.1016/j.neunet.2024.106369 (Aug. 2024).
14. Cho, J., Nam, S., Yang, H., Yun, S.-B., Hong, Y. & Park, E. *Separable PINN: Mitigating the Curse of Dimensionality in Physics-Informed Neural Networks* 2023.
15. Ayriyan, A., Grigorian, H. & Papoyan, V. Sampling of Integrand for Integration Using Shallow Neural Network. *Discrete and Continuous Models and Applied Computational Science* **32,** 38–47 (2024).
16. Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. & Teller, E. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics* **21,** 1087–1092. doi:10.1063/1.1699114 (1953).
17. Hastings, W. K. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* **57.** _eprint: https://academic.oup.com/biomet/article-pdf/57/1/97/23940249/57-1-97.pdf, 97–109. doi:10.1093/biomet/57.1.97 (Apr. 1970).
18. Lloyd, S. Using Neural Networks for Fast Numerical Integration and Optimization. *IEEE Access* **8,** 84519–84531. doi:10.1109/ACCESS.2020.2991966 (2020).
19. Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control Signals and Systems* **2,** 303–314. doi:10.1007/BF02551274 (Dec. 1989).
20. Marquardt, D. W. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *Journal of the Society for Industrial and Applied Mathematics* **11.** Publisher: Society for Industrial and Applied Mathematics, 431–441. doi:10.1137/0111030 (June 1963).
21. Genz, A. *A Package for Testing Multiple Integration Subroutines* in *Numerical Integration: Recent Developments, Software and Applications* (eds Keast, P. & Fairweather, G.) 337–340 (Springer, 1987). doi:10.1007/978-94-009-3889-2_33.
22. Anikina, A. *et al. Structure and Features of the Software and Information Environment of the HybriLIT Heterogeneous Platform* in *Distributed Computer and Communication Networks* (eds Vishnevsky, V. M., Samouylov, K. E. & Kozyrev, D. V.) 444–457 (Springer Nature Switzerland, Cham, 2025). doi:10.1007/978-3-031-80853-1_33.
23. Abadi, M. *et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems* Software available from tensorflow.org. 2015.
24. Halton, J. H. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numerische Mathematik* **2,** 84–90. doi:10.1007/BF01386213 (1960).

Information about the authors

Shcherbak, Margarita R.—Student of Department of Computational Mathematics and Artificial Intelligence of RUDN University (e-mail: 1032216537@rudn.ru, ORCID: 0000-0002-9229-2535)

Abdullina, Laysan R.—Student of Department of Computational Mathematics and Artificial Intelligence of RUDN University (e-mail: 1032216538@rudn.ru, ORCID: 0000-0002-3918-3620)

Salpagarov, Soltan I.—PhD of Physical and Mathematical Sciences, Associate Professor of Department of Computational Mathematics and Artificial Intelligence of RUDN University (e-mail: salpagarov-si@rudn.ru, ORCID: 0000-0002-5321-9650, Scopus Author ID: 57201380251)

Fedorishchev, Vyacheslav M.—Student of Department of Computational Mathematics and Artificial Intelligence of RUDN University (e-mail: 1142230295@rudn.ru, ORCID: 0009-0003-5906-9993)

УДК 517.518:004.032.26
PACS 07.05.Mh, 07.05.Kf
DOI: 10.22363/2658-4670-2025-33-4-374-388 EDN: HZSSDU

Адаптивный нейросетевой метод многомерного интегрирования для произвольных подобластей

М. Р. Щербак, Л. Р. Абдуллина, С. И. Салпагаров, В. М. Федорищев

Российский университет дружбы народов, ул. Миклухо-Маклая, д. 6, Москва, 117198, Российская Федерация

Аннотация. Многомерное интегрирование является фундаментальной задачей вычислительной математики, имеющей многочисленные приложения в физике и инженерии. Традиционные численные методы, такие как квадратура Гаусса–Лежандра и методы Монте-Карло, сталкиваются со значительными трудностями в пространствах высокой размерности из-за «проклятия размерности»: они требуют больших вычислительных ресурсов и часто теряют точность. В данной работе предлагается адаптивный метод многомерного интегрирования, основанный на нейронной сети, для эффективного вычисления интегралов по произвольным подобластям. Подход оптимизирует состав обучающей выборки с помощью параметра балансировки ρ, который регулирует долю точек, сгенерированных методом, использующим модификацию алгоритма Метрополиса–Гастингса, по сравнению с равномерным выбором. Это позволяет нейронной сети эффективно определять сложное поведение подынтегральной функции, особенно в областях с резкими изменениями. Ключевым элементом данного метода является принцип «обучи один раз — интегрируй где угодно»: одна нейронная сеть, обученная на большом домене, может впоследствии вычислять интегралы на любых произвольных подобластях без повторного обучения, что значительно снижает вычислительные затраты. Эксперименты проведены на трёх типах функций — квадратичной, Corner Peak и синусе суммы квадратов — в размерностях от 2 до 6. Точность интегрирования оценивалась с помощью метрики Correct Digits (CD). Результаты показывают, что наш метод обеспечивает сравнимую или более высокую точность по сравнению с традиционными методами (Гаусс–Лежандр, Монте-Карло, Халтона) для сложных функций, при этом существенно сокращая время вычислений. Оптимальные диапазоны ρ составляют 0.0–0.2 для гладких функций и 0.3–0.5 для функций с резкими особенностями. В многомерных случаях (4D,6D) метод демонстрирует устойчивость при $\rho = 0.2$–0.6, превосходя стохастические методы, хотя и немного уступая латинскому гиперкубическому выбору. Предложенный метод представляет собой масштабируемую и эффективную альтернативу классическим методам интегрирования, особенно полезную в задачах высокой размерности и в приложениях, требующих многократного вычисления интегралов на различных подобластях.

Ключевые слова: нейросетевое интегрирование, адаптивная генерация данных, оптимизация Левенберга–Марквардта, многомерные интегралы

Discrete & Continuous Models & Applied Computational Science
ISSN 2658-7149 (Online), 2658-4670 (Print)

2025, 33 (4) 389–403
http://journals.rudn.ru/miph

Modeling and Simulation

Research article

UDC 519.872, 519.217
PACS 07.05.Tp, 02.60.Pn, 02.70.Bf
DOI: 10.22363/2658-4670-2025-33-4-389-403 EDN: HZYRKN

Optimal eight-order three-step iterative methods for solving systems of nonlinear equations

Tugal Zhanlav[1,2], Khuder Otgondorj[2]

[1] Institute of Mathematics and Digital Technology, Mongolian Academy of Sciences, Ulaanbator, 13330, Mongolia

[2] Department of Mathematics, Mongolian University of Science and Technology, Ulaanbator, 14191, Mongolia

(received: April 18, 2025; revised: May 20, 2025; accepted: June 10, 2025)

Abstract. In this paper, we for the first time propose the extension of optimal eighth-order methods to multidimensional case. It is shown that these extensions maintained the optimality properties of the original methods. The computational efficiency of the proposed methods is compared with that of known methods. Numerical experiments are included to confirm the theoretical results and to demonstrate the efficiency of the methods.

Key words and phrases: newton-type methods, systems of nonlinear equations, convergence order, optimality and extension of methods, efficiency index

For citation: Zhanlav, T., Otgondorj, K. Optimal eight-order three-step iterative methods for solving systems of nonlinear equations. *Discrete and Continuous Models and Applied Computational Science* **33** (4), 389–403. doi: 10.22363/2658-4670-2025-33-4-389-403. edn: HZYRKN (2025).

1. Introduction

The problem of finding solution of nonlinear system

$$F(x) = 0, \quad x = (x_1, x_2, \cdots, x_n)^T \in R^n, \tag{1}$$

is often appeared in different fields of science and engineering (see [1–13] and references therein). In general, the solutions of systems cannot be obtained exactly; therefore, numerous iterative methods have been developed and are widely used to solve Equation (1). Quite recently, several methods with vector or scalar coefficients have appeared in the literature [4, 5, 7, 11]. They are distinguished by the simplicity of the algorithm compared to other well-known methods with matrix coefficients. Methods with vector coefficients constructed in R^n with point-wise multiplication and division of vectors [5, 12]. Let $x = (x_1, x_2, \dots, x_n)^T \in R^n$, $y = (y_1, y_2, \dots, y_n)^T \in R^n$. Then

$$x \cdot y = (x_1 y_1, x_2 y_2, \dots, x_n y_n)^T \in R^n, \tag{2}$$

$$\frac{x}{y} = \left(\frac{x_1}{y_1}, \frac{x_2}{y_2}, \dots, \frac{x_n}{y_n}\right)^T \in R^n. \tag{3}$$

A direct consequence of (2), (3) is

$$x \cdot y = y \cdot x, \tag{4}$$

$$x^2 = x \cdot x = (x_1^2, x_2^2, \dots, x_n^2)^T, \tag{5}$$

$$\mathbf{1} = (1, 1, \dots, 1)^T. \tag{6}$$

Using (2), (3), (4), (5) and (6) it is easy to show that the following vector expansion holds true

$$\frac{\mathbf{1}}{\mathbf{1} - x} = \mathbf{1} + x + x^2 + x^3 \dots, \quad \text{for } \|x\| = \max_i |x_i| < 1. \tag{7}$$

We consider three-step iterations

$$\begin{aligned} y_k &= x_k - F'(x_k)^{-1}F(x_k), \\ z_k &= \phi_p(x_k, y_k), \qquad \left(z_k = y_k - \bar{\tau}_k F'(x_k)^{-1}F(y_k)\right), \\ x_{k+1} &= z_k - \alpha_k F'(x_k)^{-1}F(z_k). \end{aligned} \tag{8}$$

Here $z_k = \phi_p(x_k, y_k)$ is an iteration of order $p \geqslant 2$ i.e., $F(z_k) = O(F(x_k)^p)$. The following Theorem 1 was proved in [8, 9].

Theorem 1. *Suppose that $F(x)$ is sufficiently differentiable in the open convex domain $D \in R$ containing the simple zero x^* of* (1) *and $F(x)$ is continuous and non-singular at x^*. Let $x_0 \in D$ be an initial approximation sufficiently close to x^*. Then the local order of convergence of iteration* (8) *equal to $p + 4$ if and only if α_k satisfies*

$$\alpha_k = I + 2\eta_k + 6\eta_k^2 + 3d_k + 20\eta_k^3 + 20d_k\eta_k + c_k + O(h^4), \tag{9}$$

where

$$\begin{aligned} \eta_k &= \frac{1}{2}F'(x_k)^{-1}F''(x_k)\xi_k, \quad d_k = -\frac{1}{6}F'(x_k)^{-1}F'''(x_k)\xi_k^2, \\ c_k &= \frac{1}{6}F'(x_k)^{-1}F^{(4)}(x_k)\xi_k^3, \quad \xi_k = F'(x_k)^{-1}F(x_k). \end{aligned}$$

It should be pointed out that in the construction of high-order iterations, in particular, in the proof of the above mentioned theorem, the following Taylor expansion lemma and permutation properties [4] of q-derivative of the vector function $F(x)$ were used.

Lemma 1. *Let $F : D \subseteq R^n \to R^n$ be $p-$times Fréchet differentiable in a open convex set $D \subseteq R^n$, then for any $x, \hat{h} \in D$ the following expression holds:*

$$F(x + \hat{h}) = F(x) + F'(x)\hat{h} + \frac{1}{2!}F''(x)\hat{h}^2 + \frac{1}{3!}F'''(x)\hat{h}^3 + \dots + \frac{1}{p!}F^{(p)}(x)\hat{h}^p + R_p,$$

where

$$\|R_p\| \leq \frac{1}{p!}\sup_{0<t<1}\|F^{(p)}(x + t\hat{h})\|\|\hat{h}\|^p \ \text{ and } \ \hat{h}^p = (\overbrace{\hat{h}, \hat{h}, \cdots, \hat{h}}^{p}),$$

and $\|\cdot\|$ denotes any norm in R^n, or a corresponding operator norm.

The q-th derivative of F at $u \in R^n$, $q \geq 1$, is the q-linear function $F^{(q)}(u) : R^n \times \dots \times R^n \to R^n$ such that $F^{(q)}(u)\left(v_1, \dots, v_q\right) \in R^n$

(i) $F^{(q)}(u)\left(v_1, \dots, v_{q-1}, \cdot\right) \in L\left(R^n\right)$,
(ii) $F^{(q)}(u)\left(v_{\sigma(1)}, \dots, v_{\sigma(q)}\right) = F^{(q)}(u)\left(v_1, \dots, v_q\right)$, for all permutations σ of $\{1, 2, \dots, q\}$.

From the above properties, we can use the following notation:

(a) $F^{(q)}(u)\left(v_1, \dots, v_q\right) = F^{(q)}(u)v_1 \dots v_q$,
(b) $F^{(q)}(u)v^{q-1}F^{(p)}(u)v^p = F^{(q)}(u)F^{(p)}(u)v^{q+p-1}$.

For convenience, we also recall the conjecture given by [4].

Conjecture. The order of convergence of any iterative method without memory for solving nonlinear systems cannot exceed $2^{k_1+k_2-1}$, where k_1 is the number of evaluation of the Jacobian matrix and k_2 is the number of evaluations of the nonlinear function per iteration, and $k_1 \le k_2$. When the scheme reach this upper bound, we say that it is optimal.

In the second step of iteration (8) we use

$$z_k = y_k - \bar{\tau}_k F'(x_k)^{-1} F(y_k),$$

with vector parameter

$$\bar{\tau}_k = \frac{\mathbf{1} + a\Theta_k + b\Theta_k^2}{\mathbf{1} + (a-2)\Theta_k + d\Theta_k^2} = \mathbf{1} + 2\Theta_k + O(F(x_k)^2), \quad a, b, d \in R, \tag{10}$$

where

$$\Theta_k = \frac{F(y_k)}{F(x_k)}. \tag{11}$$

In [12], it was proven that $p = 4$ under choice (10). Using $\Theta_k = O(F(x_k))$ and the expansion (7), we rewrite (10) as

$$\begin{aligned} \bar{\tau}_k = \mathbf{1} + 2\Theta_k + (2(2-a) + b - d)\,\Theta_k^2 + (2(2-a)^2 \\ + (2-a)(b-2d) - ad)\Theta_k^3 + \dots . \end{aligned} \tag{12}$$

The optimal fourth-order two-step methods were first proposed in [4, 12]. The purpose of this paper is to develop families of optimal eighth-order methods based on the optimal fourth-order methods introduced in [12].

2. Extensions of several iterations to multidimensional case

First, we consider three-step iteration

$$\begin{aligned} y_k &= x_k - \frac{f(x_k)}{f'(x_k)}, \\ z_k &= y_k - \bar{\tau}_k \frac{f(y_k)}{f'(x_k)}, \\ x_{k+1} &= z_k - \alpha_k \frac{f(z_k)}{f'(x_k)}, \end{aligned} \tag{13}$$

for solving nonlinear scalar equation $f(x) = 0$. Let the iteration parameter $\bar{\tau}_k$ satisfies

$$\bar{\tau}_k = 1 + 2\theta_k + \beta\theta_k^2 + \gamma\theta_k^3 + O(f_k^4), \quad \theta_k = \frac{f(y_k)}{f'(x_k)}. \tag{14}$$

In [8], it was proven that the local order of the iteration (13) is eight if and only if α_k satisfies the condition

$$\alpha_k = 1 + 2\theta_k + (\beta+1)\theta_k^2 + (2\beta+\gamma-4)\theta_k^3 + (1+4\theta_k)\frac{f(z_k)}{f(y_k)} + O(f_k^4). \tag{15}$$

We now consider a formal extension of the iteration (13) to R^n with point-wise multiplication and division of vectors replacing $f(x)$ by $F(x)$. Then (13) leads to (8). The parameter choices (14) and (15) lead to

$$\bar{\tau}_k = \mathbf{1} + 2\Theta_k + \beta\Theta_k^2 + \gamma\Theta_k^3 + \dots, \tag{16}$$

and

$$\alpha_k = \mathbf{1} + 2\Theta_k + (\beta+1)\Theta_k^2 + (2\beta+\gamma-4)\Theta_k^3 + (\mathbf{1}+4\Theta_k)\frac{F(z_k)}{F(y_k)}, \tag{17}$$

respectively, where Θ_k is defined by formula (11). In this connection, the question arises: how is the local order of iteration (8) preserved or reduced when vector parameters satisfy conditions (16) and (17)? The formal extension with parameters satisfying (16) and (17) seems to be actual extension with good convergence properties. Namely, we can state and prove the following result:

Theorem 2. *Suppose that the assumptions of Theorem 1 are satisfied. Then the iterative method* (8) *has a eight-order of convergence if and only if the parameters $\bar{\tau}_k$ and α_k satisfy* (16) *and* (17)*, respectively.*

Proof. From (10), (12) and (16) it is clear that $p = 4$ maintained under (16). To prove the theorem it suffices to show what the conditions (9) and (17) are equivalent. Using Lemma 1 and permutations properties (i), (ii) one can easily obtain

$$\Theta_k = \frac{F(y_k)}{F(x_k)} = \eta_k + d_k + \frac{c_k}{4} + O\big(F(x_k)^4\big), \tag{18}$$

$$s_k = \frac{F(z_k)}{F(y_k)} = I - \hat{t}_k\bar{\tau}_k + \bar{\tau}_k^2\eta_k(\eta_k + 4d_k) + O\big(F(x_k)^4\big), \tag{19}$$

where

$$\hat{t}_k = F'(x_k)^{-1}F'(y_k) = I - 2\eta_k - 3d_k - c_k + O\big(F(x_k)^4\big). \tag{20}$$

Substituting (16), (20) into (19) we obtain

$$s_k = d_k + (5-\beta)\eta_k^2 + \frac{c_k}{2} + (14-2\beta)\eta_k d_k + (4+2\beta-\gamma)\eta_k^3 + O\big(F(x_k)^4\big). \tag{21}$$

Further using (18), (21) in (17) we obtain (9) i.e., from (17) we get (9). Analogously, if we take into account formulas (18) and (21), then (9) immediately leads to (17). □

Remark 1. If we ignore terms with order $O\big(F(x_k)^3\big)$ in (17) then the order of convergence of iteration (8) decreased by unit i.e., the iteration (8) has $p + 3$ order of convergence if and only if $\bar{\tau}_k$ and α_k satisfy (16) and

$$\alpha_k = \mathbf{1} + 2\Theta_k + (\beta+1)\Theta_k^2 + \frac{F(z_k)}{F(y_k)} + O\big(F(x_k)^3\big),$$

respectively.

Since $p = 4$ under choice (16), the condition (17) guarantees eight-order convergence of iteration (8). This extension (8) with (16), (17) is scheme with vector coefficients and optimal according to the conjecture, because of $k_1 = 1$, $k_2 = 3$. In [2] the authors found that it is better not to use polynomials as weight functions. Following this idea, we use (10) and rewrite formula (17) in rational function form as

$$\alpha_k = \left(\frac{\mathbf{1} - \Theta_k}{\mathbf{1} - 2\Theta_k}\right)^2 H_k - \frac{\gamma_1 \Theta_k^2}{(\mathbf{1} - \gamma_2 \Theta_k)}, \tag{22}$$

where

$$t_k = \frac{F(z_k)}{F(x_k)}, \quad s_k = \frac{F(z_k)}{F(y_k)}, \quad H_k = \frac{\mathbf{1} + \dfrac{t_k}{(\mathbf{1} + \alpha t_k)}}{(\mathbf{1} - t_k)(\mathbf{1} - s_k)},$$
$$\gamma_1 = 2a + d - b, \quad \gamma_2 = \frac{8 + 2\gamma_1 - 2(2-a)^2 - (2-a)(b-2d) + ad}{\gamma_1}.$$

So, we obtain a family of optimal eight-order iterations

$$\begin{aligned} y_k &= x_k - F'(x_k)^{-1} F(x_k), \\ z_k &= y_k - \bar{\tau}_k F'(x_k)^{-1} F(y_k), \\ x_{k+1} &= z_k - \alpha_k F'(x_k)^{-1} F(z_k), \end{aligned} \tag{23}$$

where $\bar{\tau}_k$ and α_k are given by (10) and (22) respectively. We consider a simpler case of (23). Let $b = d = 0$. Then, equation (23) simplifies to:

$$\begin{aligned} y_k &= x_k - F'(x_k)^{-1} F(x_k), \\ z_k &= y_k - \frac{\mathbf{1} + a\Theta_k}{\mathbf{1} + (a-2)\Theta_k} F'(x_k)^{-1} F(y_k), \\ x_{k+1} &= z_k - \left(\left(\frac{\mathbf{1} - \Theta_k}{\mathbf{1} - 2\Theta_k}\right)^2 H_k - \frac{2a\Theta_k^2}{(\mathbf{1} + (a-6)\Theta_k)} \right) F'(x_k)^{-1} F(z_k), \quad a \in R. \end{aligned} \tag{24}$$

In particular, when $a = 0$ the iteration (24) leads to one, which are called the extension of Sharma's method [6]. In [2], Chun and Neta conducted a comparative analysis of well-known eighth-order optimal methods based on the average number of iterations and CPU time. They found that WL [13], Sharma [6] and CL [3] methods are best one. In principle, one can formulate the extension of any eight-order method to a multidimensional case replacing $f(x)$, $f(y)$, $f(z)$, $f[u_k, v_k]$ by $F(x)$, $F(y)$, $F(z)$, $F[u_k, v_k]$ respectively.

In construction of extension often used formulas

$$F[u_k, v_k] = \frac{F(v_k) - F(u_k)}{v_k - u_k}, \tag{25}$$

and

$$F[x_k, y_k] = F'(x_k)(\mathbf{1} - \Theta_k), \tag{26}$$

$$F[y_k, z_k] = F'(x_k)\frac{\mathbf{1} - s_k}{\bar{\tau}_k}, \tag{27}$$

$$F[x_k, z_k] = F'(x_k)\frac{\mathbf{1} - t_k}{\mathbf{1} + \bar{\tau}_k \Theta_k}, \quad t_k = \frac{F(z_k)}{F(x_k)} = \Theta_k s_k, \tag{28}$$

that followed by (8). So, we restrict here only extensions of the WL [13], SS [6], CL [3], BRW [13] methods. First we note that when $a = b = d = 0$ the iteration (23) includes an extension of Sharma et al. [6] to multidimensional case as a particular case.

1. Now we consider the iteration (8) with parameters $\bar{\tau}_k$, α_k given by (10) and

$$\alpha_k = \frac{F'(x_k)}{F[y_k, z_k] + 2\left(F[x_k, z_k] - F[y_k, x_k]\right) + \dfrac{y_k - z_k}{y_k - x_k}\left(F[y_k, x_k] - F'(x_k)\right)}. \tag{29}$$

Using formulas (25), (26), (27) and (28) and expansion (7) one can rewrite (29) as

$$\alpha_k = \bar{\tau}_k\left(\mathbf{1} + \Theta_k^2 + (2\beta - 6)\Theta_k^3 + (\mathbf{1} + 2\Theta_k)s_k\right) + O\big(F(x_k)^4\big). \tag{30}$$

If we take into account (16), then (30) leads to (17). Then by Theorem 2 the iteration (8) with parameter $\bar{\tau}_k$, α_k given by (10) and (29) is optimal eight-order convergence. Thus, we obtain a wide class of optimal eight-order methods.

Note that the formula (29) is the extension formula (3.63) given by [9, 10] to multidimensional case. When $a = b = d = 0$ in (10), $\bar{\tau}_k$ becomes as

$$\bar{\tau}_k = \frac{\mathbf{1}}{\mathbf{1} - 2\Theta_k}. \tag{31}$$

The iteration (8) with $\bar{\tau}_k$ and α_k given by (31) and (29) is extension of WL method [13]. In [10] was given extension of some classes of optimal three-point iterations for solving nonlinear equations, that works well for any choice of parameter $\bar{\tau}_k$ satisfying the condition (14). According to Theorem 2, these extensions suggested in [10] are immediately extended to multidimensional case.

2. For example, we consider the iteration (8) parameters $\bar{\tau}_k$ satisfying (10) and α_k given by

$$\alpha_k = \left(\mathbf{1} + 2t_k - 2(2a+1)\Theta_k^3\right)\frac{F'(x_k)}{F[z_k, y_k] + \dfrac{z_k - y_k}{z_k - x_k}\big(F[z_k, x_k] - F'(x_k)\big)}. \tag{32}$$

As before, it is easy to show that α_k given by (32) satisfies the condition (17). So by Theorem 2 the iteration (8) with parameters given by (10), (32) has a eight-order convergence. Note that a formula (32) is a multidimensional extension of optimal modification (23) in [10]. When $a = -\frac{1}{2}$ the iteration(8) with parameters $\bar{\tau}_k$, α_k given by (10), (32) is considered as multidimensional extension of BRW method [1]. Our finding and generalization show that, in particular, BRW [1] is optimal not only for $a = -\frac{1}{2}$, but also it is optimal for any a. So we call the methods (8) with (10), (32) multidimensional extension of the optimal modification of BRW methods.

3. Now we consider the iteration (8) with parameters $\bar{\tau}_k$ satisfying the condition (16) and α_k given by

$$\alpha_k = \frac{F'(x_k)\left[\mathbf{1} + A\Theta_k + B\Theta_k^2 + C\Theta_k^3 + (\delta + \Delta\Theta_k)s_k\right]}{\omega_1 F[x_k, z_k] + \omega_2 F[z_k, y_k] + \omega_3 F[x_k, y_k]}, \tag{33}$$

where $\omega_1 + \omega_2 + \omega_3 = 1$ and A, B, C, δ, Δ are free parameters to be determined properly. As before, it is easy to show that the iteration (8) with $\bar{\tau}_k$, α_k given by (10) and (33) is optimal eight-order convergence when

$$A = \delta = 1 - \omega_2, \quad B = (\tilde{\beta} - 2)(1 - \omega_2) + 1 - \omega_1, \quad \Delta = 3 - \omega_1 - \omega_2$$
$$C = \tilde{\gamma}(1 - \omega_2) + \tilde{\beta}(1 + \omega_2 - \omega_1) + \omega_1 - \omega_2 - 5.$$

Table 1

Class of optimal eight-order iterations (33)

ω_1	ω_2	ω_3
0	1	0
1	0	0
0	0	1
-1	2	0
1	1	-1
-1	1	1

Thus, we have a wide class of optimal eight-order iterations (8), (10) and (33) containing five free parameters a, b, d and ω_1, ω_2. The formula (33) is an extension of formula (11) in the paper [10] to the multidimensional case. The interesting and easy cases of (33) are in Table 1.

This family of methods (8), (10), (33) contains many multidimensional extensions of well-known eight-order methods for solving nonlinear equations (See [9, 10] and references therein).

4. Now we will derive an extension of CL method [3]. To do this we consider the weight function α_k as

$$\alpha_k = \frac{\mathbf{1}}{\big(\mathbf{1} - H(\Theta_k) - J(t_k) - P(s_k)\big)^2}, \tag{34}$$

where the weight functions should satisfy the following conditions to guarantee eight order:

$$H(0) = 0, \quad H'(0) = 1, \quad H''(0) = \beta - 2, \quad H'''(0) = 3(\gamma - \beta - 2), \tag{35}$$

$$J(0) = 0, \quad J'(0) = 1/2, \quad P(0) = 0, \quad P'(0) = 1/2, \tag{36}$$

where β and γ are constants in (16). These functions satisfying the conditions (35), (36) are given by

$$J(t_k) = \frac{1}{2}\frac{t_k}{\mathbf{1} + \delta_1 t_k}, \quad P(s_k) = \frac{1}{2}\frac{s_k}{\mathbf{1} + \delta_2 s_k}, \quad H(\Theta_k) = \frac{\Theta_k}{2}\frac{2 + (3 - 4\delta_3)\Theta_k}{1 + (1 - 2\delta_3)\Theta_k + \delta_3\Theta_k^2},$$

where δ_1, δ_2, $\delta_3 \in R$.

If we choose $\bar{\tau}_k$ as

$$\bar{\tau}_k = \frac{\mathbf{1}}{(1 - \Theta_k)^2}, \quad (\beta = 3,\ \gamma = 4 \text{ in } (16)), \tag{37}$$

then the iteration (8) with $\bar{\tau}_k$, α_k given by (37) and (34) is extension of CL methods [3].

3. Transition to schemes with scalar coefficients

The eighth-order family of iterations (8) with parameters $\bar{\tau}_k$ and α_k satisfying condition (16) and (17) is scheme with vector coefficients. Now we will show that it is possible transition from scheme with vector coefficients to scheme with scalar coefficients. Indeed, using (16) one can obtain

$$\bar{\tau}_k F(y_k) = (\mathbf{1} + \beta\Theta_k^2)F(y_k) + (2\Theta_k^2 + \gamma\Theta_k^4)F(x_k). \tag{38}$$

Analogously, using (17) and the following

$$t_k = \frac{F(z_k)}{F(x_k)}, \quad s_k = \frac{F(z_k)}{F(y_k)}, \quad s_k\Theta_k = \frac{F(z_k)}{F(x_k)}$$

we get

$$\alpha_k F(z_k) = F(z_k) + (s_k^2 + (\beta+1)\Theta_k t_k)F(y_k) + \big(2\Theta_k^2 t_k + (2\beta+\gamma-4)\Theta_k^2 t_k + 4s_k^2\Theta_k^2\big)F(x_k). \tag{39}$$

Using transition rules [11] (replacing the pointwise multiplication by dotted product)

$$\begin{gathered}\Theta_k t_k = \frac{F(z_k)F(y_k)}{F(x_k)^2} \Longleftrightarrow \alpha_k = \frac{(F(z_k), F(y_k))}{\|F(x_k)\|^2},\\ s_k^2 = \left(\frac{F(z_k)}{F(y_k)}\right)^2 \Longleftrightarrow \beta_k = \frac{\|F(z_k)\|^2}{\|F(y_k)\|^2}, \quad \Theta_k^2 \Longleftrightarrow \upsilon_k = \frac{\|F(y_k)\|^2}{\|F(x_k)\|^2},\\ s_k^2\Theta_k^2 \Longleftrightarrow \delta_k = \upsilon_k\beta_k = \frac{\|F(z_k)\|^2}{\|F(x_k)\|^2}, \quad \mathbf{1} \Longleftrightarrow 1,\end{gathered} \tag{40}$$

in (38) and (39) the iterations (8) can be rewritten as follows:

$$\begin{gathered}y_k = x_k - F'(x_k)^{-1}F(x_k),\\ z_k = y_k - F'(x_k)^{-1}\big((1+\beta\upsilon_k)F(y_k) + (2\upsilon_k + \gamma\upsilon_k^2)F(x_k)\big),\\ x_{k+1} = z_k - F'(x_k)^{-1}\big(F(z_k) + (\beta_k + (\beta+1)\alpha_k)F(y_k) + (2\alpha_k + (2\beta+\gamma-4)\upsilon_k\alpha_k + 4\delta_k)F(x_k)\big).\end{gathered} \tag{41}$$

Thus, we obtain first a family of optimal eighth-order iterations (41) with scalar coefficients. The conversation of iterations (41) to (8) with vector coefficients (38) and (39) is obvious by virtue of rules (40). When $\beta = 1$ and $\gamma = 0$, the iteration (41) reduces to the method obtained by Cordero et al. in [14] as a particular case.

4. Computational efficiency

The computational efficiency index of an iterative method for solving a nonlinear system is defined by $CI = \rho^{\frac{1}{C}}$, where ρ is the order of convergence and C is the computational cost of each method. We will examine the computational efficiency of the proposed methods and compare it with that of the methods introduced in [14–16]. To compute the function F and its derivative F' in any iterative method, we evaluate n and n^2 scalar functions, respectively. In addition, we must account for the number of operations shown in Table 2.

As shown in Fig. 1 and Table 3, the method ESS8 and method (41) demonstrate higher computational efficiency than the other methods considered.

5. Numerical Experiments

To validate the theoretical results concerning the convergence behavior and computational efficiency of the proposed methods, we present several numerical experiments and compare their performance with existing methods of the same order. The experiments were made with an Intel Core processor i5-4590, with a CPU of 3.30 GHz and 4096 MB of RAM memory. All computations are performed in the programming package Mathematica 14 using multiple-precision arithmetics. We have used 1000-digit floating-point arithmetic to minimize round-off errors as much as possible. The iterative process is terminated when the following stopping criterion is satisfied:

$$\|x_{k+1} - x_k\| + \|F(x_{k+1})\| \leq 10^{-30}.$$

Table 2

Computational cost of different operations

	Computational cost
LU decomposition	$\frac{1}{3}(n^3 - n)$
Solution of two triangular systems	n^2
Matrix-matrix multiplication	n^3
Scalar-vector multiplication	n
Component-wise multiplication (division) of vectors	n

Table 3

Comparison of computational efficiency

№	methods	ρ	C_i	CI
1	(8), (10), (22)	8	$C_1 = \frac{1}{3}n^3 + 4n^2 + \frac{50}{3}n$	$8^{1/C_1}$
2	(8), (10), (29)	8	$C_2 = \frac{1}{3}n^3 + 4n^2 + \frac{41}{3}n$	$8^{1/C_2}$
3	(8), (10), (32)	8	$C_3 = \frac{1}{3}n^3 + 4n^2 + \frac{50}{3}n$	$8^{1/C_3}$
4	(8), (10), (33)	8	$C_4 = \frac{1}{3}n^3 + 4n^2 + \frac{44}{3}n$	$8^{1/C_4}$
5	(8), (10), (34)	8	$C_4 = \frac{1}{3}n^3 + 4n^2 + \frac{44}{3}n$	$8^{1/C_4}$
6	(41)	8	$C_6 = \frac{1}{3}n^3 + 4n^2 + \frac{40}{3}n$	$8^{1/C_6}$
7	NLM8	8	$C_7 = \frac{1}{3}n^3 + 13n^2 + \frac{8}{3}n$	$8^{1/C_7}$
8	ZCO8	8	$C_8 = \frac{1}{3}n^3 + 11n^2 + \frac{14}{3}n$	$8^{1/C_8}$

The results of the numerical experiments are presented in Tables 4–7. These tables include the number of iterations k, the elapsed CPU time, the absolute error between consecutive iterates $\|x_{k+1} - x_k\|$, the absolute residual error $\|f(x_{k+1})\|$ of the corresponding function, and the computational order of convergence ρ_{co}. Here, the computational order of convergence ρ_{co} is calculated using the formula

$$\rho_{co} = \frac{\ln(\|F(x_{k+1})\|/\|F(x_k)\|)}{\ln(\|F(x_k)\|/\|F(x_{k-1})\|)}.$$

For the purpose of comparison, our analysis includes the methods proposed in [14–16]. For convenience, we will use the following abbreviations for the methods throughout the remainder of this paper.

- ESS8: Method (8) with (10) and (22), $a = 0$, $b = d = 0$, $\alpha = 0$.
- EWL8: Method (8) with (10) and (29), $a = 0$, $b = d = 0$.
- EBRW8: Method (8) with (10) and (32), $a = 0$, $b = d = 0$.
- EZO8: Method (8) with (10) and (33), $a = 0$, $b = d = 0$, $w_1 = w_3 = 0$, $w_2 = 1$, $\tilde{\beta} = 4$, $\tilde{\gamma} = 3$.
- ECL8: Method (8) with (10) and (34), $a = d = 0$, $d = 1$, $\delta_1 = \delta_3 = 0$, $\delta_2 = 1$.

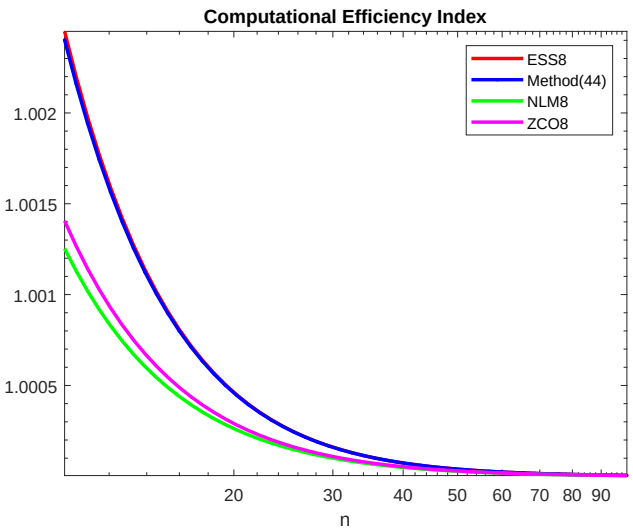


Figure 1. Computational Efficiency Index for $n = 10$ to 100 (logarithmic scale)

- NOM8: Method (41) with $\beta = \gamma = 0$.
- CTT8: Cordero et al. method (2025).
- NLM8: Sharma et al. method (2017).
- ZCO8: Zhanlav et al. method (2020).

Example 1. We begin with the following system of n equations:

$$\sum_{j=1, j\neq i}^{n} x_j - e^{-x_i} = 0, \quad 1 \leq i \leq n,$$

with $n = 50$. The initial approximation is chosen as $x_0 = (0.5, 0.5, \dots, 0.5)^T$.

Example 2. Our second example is given by the following system of nonlinear equations:

$$\begin{cases} x_i x_{i+1} - e^{-x_i} - e^{-x_{i+1}} = 0, & 1 \leq i \leq n-1, \\ x_n x_1 - e^{-x_n} - e^{-x_1} = 0, \end{cases}$$

where $n = 75$. The initial approximation used in all the methods is $x_0 = (1.2, 1.2, 1.2, 1.2, \dots, 1.2, 1.2)^t$.

Example 3. We consider the system of trigonometric equations:

$$x_i - \cos\left(2x_i - \sum_{j=1}^{n} x_j\right) = 0, \quad i = 1, 2, \dots, n,$$

with $n = 100$. The exact solution is $x^* = \{0.2062, 0.2062 \dots, 0.2062\}^T$. To approximate this solution, we choose the initial vector $x_0 = \{0.3, 0.3, \dots, 0.3\}^T$.

Table 4

Results Example 1

Method	CPU Time	Iterations	$\|x_{k+1} - x_k\|$	$\|F(x_{k+1})\|$	ACOC
ESS8	0.312	3	8.3528×10^{-111}	2.8275×10^{-895}	8.00
EWL8	0.344	3	2.0945×10^{-110}	5.0227×10^{-892}	8.00
EBRW8	0.329	3	1.4007×10^{-110}	3.5560×10^{-895}	8.00
EZO8	0.328	3	8.5898×10^{-111}	1.9078×10^{-893}	8.00
ECL8	0.312	3	9.2848×10^{-107}	1.8216×10^{-862}	8.00
NOM8	0.329	3	2.7848×10^{-108}	7.3470×10^{-875}	8.00
CTT8	0.328	3	3.7717×10^{-108}	1.0796×10^{-873}	8.00
NLM8	1.625	3	1.4001×10^{-123}	6.9822×10^{-992}	8.00
ZCO8	1.812	3	4.7542×10^{-125}	7.1425×10^{-999}	8.00

Table 5

Results Example 2

Method	CPU Time	Iterations	$\|x_{k+1} - x_k\|$	$\|F(x_{k+1})\|$	ACOC
ESS8	0.93	3	7.1752×10^{-81}	1.1021×10^{-654}	8.00
EWL8	0.109	3	1.4674×10^{-82}	1.7636×10^{-668}	8.00
EBRW8	0.94	3	4.6937×10^{-73}	1.7645×10^{-591}	8.00
EZO8	0.95	3	1.4790×10^{-78}	3.1931×10^{-636}	8.00
ECL8	0.94	3	9.3866×10^{-69}	6.8655×10^{-555}	8.00
NOM8	0.78	3	7.8886×10^{-57}	3.4590×10^{-458}	8.00
CTT8	0.94	3	7.7321×10^{-59}	1.5257×10^{-474}	8.00
NLM8	1.375	3	1.4355×10^{-94}	9.4181×10^{-540}	8.00
ZCO8	1.102	3	2.312×10^{-91}	1.7645×10^{-538}	8.00

Example 4. Finally, we analyze the performance of the methods on a large-scale nonlinear system:

$$\begin{cases} x_i^2 x_{i+1} - 1 = 0, & 1 \le i \le n-1, \\ x_n^2 x_1 - 1 = 0, \end{cases}$$

with $n = 1000$. The exact solution is $x^* = (1, 1, \dots, 1)^T$ and $x_0 = (1.25, 1.25, \dots, 1.25)^t$ is the initial vector used.

The results obtained from our experiments provide complete support for the convergence theory presented in Sections 2 and 3. Additionally, the methods listed above were compared in terms of CPU

Table 6

Results Example 3

Method	CPU Time	Iterations	$\|x_{k+1} - x_k\|$	$\|F(x_{k+1})\|$	ACOC
ESS8	6.594	3	1.2652×10^{-47}	3.0069×10^{-370}	8.00
EWL8	6.688	3	7.9068×10^{-48}	1.1668×10^{-375}	8.00
EBRW8	6.672	3	1.8246×10^{-45}	5.5836×10^{-354}	8.00
EZO8	6.657	3	2.3788×10^{-48}	5.4431×10^{-376}	8.00
ECL8	6.594	3	6.3970×10^{-46}	5.1656×10^{-368}	8.00
NOM8	6.687	3	1.1417×10^{-38}	1.6423×10^{-273}	8.00
CTT8	6.634	3	3.9494×10^{-39}	2.8353×10^{-277}	8.00
NLM8	21.092	3	1.0486×10^{-42}	2.4170×10^{-330}	8.00
ZCO8	19.304	3	4.5715×10^{-40}	3.1475×10^{-328}	8.00

Table 7

Results Example 4

Method	CPU Time	Iterations	$\|x_{k+1} - x_k\|$	$\|F(x_{k+1})\|$	ACOC
ESS8	27.969	3	1.1391×10^{-41}	2.9879×10^{-338}	8.00
EWL8	28.453	3	7.4365×10^{-45}	3.9437×10^{-364}	8.00
EBRW8	37.765	3	3.4087×10^{-38}	8.8383×10^{-310}	8.00
EZO8	28.375	3	1.1741×10^{-38}	1.2945×10^{-313}	8.00
ECL8	28.625	3	2.1198×10^{-39}	5.0136×10^{-320}	8.00
NOM8	28.766	3	3.6141×10^{-31}	4.9968×10^{-271}	8.00
CTT8	28.984	3	2.8237×10^{-32}	4.2326×10^{-272}	8.00
NLM8	633.995	3	1.1622×10^{-33}	2.3326×10^{-277}	7.99
ZCO8	629.449	3	7.2358×10^{-32}	1.5687×10^{-272}	7.99

time. As shown in Tables 4–7, the proposed method ESS8 demonstrates superior speed compared to the other methods. It is worth noting that iteration (8) with $a = 2$, $b = d = 0$ yields similar results in the same experiments. Finally, based on our experimental results, we conclude that methods with vector and scalar coefficients require less CPU time than other well-known methods with matrix coefficients.

Conclusions

The main contributions of this paper are as follows:

- Investigation in R^n with point-wise multiplication and division not only allows us to derive simple schemes but also to design extensions of many eighth-order iterations to multidimensional case.
- These extensions maintained the optimality properties of the original methods.
- We propose first wide class of optimal eighth-order iterative methods with vector and scalar coefficients for solving systems of nonlinear equations.
- As a whole, the results obtained in this paper, can be considered as new achievement in iteration theory.

In conclusion, the proposed methods fulfill the fundamental criteria of high-quality algorithms: low computational cost, minimal execution time, and a simple structure.

Author Contributions: The contributions of the authors are equal. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data sharing is not applicable.

Acknowledgments: The authors wish to thank the editor and the anonymous referees for their valuable suggestions and comments on the first version of this paper

Conflicts of Interest: The authors declare no conflict of interest.

Declaration on Generative AI: The authors have not employed any Generative AI tools.

References

1. Bi, W., Ren, H. & Wu, Q. Three-step iterative methods with eighth-order convergence for solving nonlinear equations. *Journal of Computational and Applied Mathematics* **225,** 105–112. doi:10.1016/j.cam.2008.07.004 (2009).
2. Changbum, C. & Neta, B. Comparison of several families of optimal eighth order methods. *Applied Mathematics and Computation* **274,** 762–773. doi:10.1016/j.amc.2015.10.092 (2016).
3. Changbum, C. & Lee, M. Y. A new optimal eighth-order family of iterative methods for the solution of nonlinear equations. *Applied Mathematics and Computation* **223,** 506–519. doi:doi:10.1016/j.amc.2013.08.033 (2013).
4. Cordero, A., Rojas-Hiciano, R. V., Torregrosa, J. R. & Vassileva, M. P. A highly efficient class of optimal fourth-order methods for solving nonlinear systems. *Numerical Algorithms* **95,** 1879–1904. doi:10.1007/s11075-023-01631-9 (2024).
5. Dehghan, M. & Shirilord, A. Three-step iterative methods for numerical solution of systems of nonlinear equations. *Engineering with Computers* **38,** 1015–1028. doi:10.1007/s00366-020-01072-1 (2020).
6. Sharma, J. R. & Sharma, R. A new family of modified Ostrowski's methods with accelerated eighth order convergence. *Numerical Algorithms* **54,** 445–458. doi:10.1007/s11075-009-9345-5 (2010).
7. Singh, H., Sharma, J. R. & Kumar, S. A simple yet efficient two-step fifth-order weighted-Newton method for nonlinear models. *Numerical Algorithms* **93,** 203–225. doi:10.1007/s11075-022-01412-w (2023).
8. Zhanlav, T., Chuluunbaatar, O. & Ulziibayar, V. Necessary and sufficient conditions for two and three-point iterative method of Newton's type iterations. *Computational Mathematics and Mathematical Physics* **57,** 1090–1100. doi:10.1134/S0965542517070120 (2017).
9. Zhanlav, T. & Chuluunbaatar, O. *New development of Newton-type iterations for solving nonlinear problems* 281 pp. doi:10.1007/978-3-031-63361-4 (Switzerland, Springer Nature, 2024).

10. Zhanlav, T. & Otgondorj, K. On the development and extensions of some classes of optimal three-point iterations for solving nonlinear equations. *Journal of Numerical Analysis and Approximation Theory* **50,** 180–193. doi:10.33993/jnaat502-1238180--193 (2021).
11. Zhanlav, T. & Otgondorj, K. High efficient iterative methods with scalar parameter coefficients for systems of nonlinear equations. *Journal of Mathematical Sciences* **279,** 866–875. doi:10.1007/s10958-024-07066-4 (2024).
12. Zhanlav, T. & Otgondorj, K. Development and adaptation of higher-order iterative methods in R^n with specific rules. *Discrete and Continuous Models and Applied Computational Science* **32,** 425–444. doi:10.22363/2658-4670-2024-32-4-425-444 (2024).
13. Wang, X. & Liu, L. Modified Ostrowski's method with eighth-order convergence and high efficiency index. *Applied Mathematics Letters* **23,** 549–554. doi:10.1016/j.aml.2010.01.009 (2010).
14. Cordero, A., Torregrosa, J. R. & Triguero-Navarro, P. First optimal vectorial eighth-order iterative scheme for solving non-linear systems. *Applied Mathematics and Computation* **498,** 129401. doi:10.1016/j.amc.2025.129401 (2025).
15. Sharma, J. & Arora, H. Improved Newton-like methods for solving systems of nonlinear equations. *SeMA Journal* **74,** 147–163. doi:10.1007/s40324-016-0085-x (2017).
16. Zhanlav, T. & Otgondorj, K. Higher order Jarratt-like iterations for solving systems of nonlinear equations. *Applied Mathematics and Computation* **395,** 125849. doi:doi:10.1016/j.amc.2020.125849 (2021).

Information about the authors

Zhanlav, Tugal—Full member of Mongolian Academy of Sciences, professor, doctor of sciences in physics and mathematics, Honorary Doctor of JINR (Dubna) (e-mail: tzhanlav@yahoo.com, phone: +(976)99184824, ORCID: 0000-0003-0743-5587, Scopus Author ID: 24484328800)

Otgondorj, Khuder—Associate Professor of Department of Mathematics at School of Applied Sciences, Mongolian University of Science and Technology (e-mail: otgondorj@gmail.com, phone: +(976)99034852, ORCID: 0000-0003-1635-7971, Scopus Author ID: 57209734799)

УДК 519.872, 519.217
PACS 07.05.Tp, 02.60.Pn, 02.70.Bf
DOI: 10.22363/2658-4670-2025-33-4-389-403 EDN: HZYRKN

Оптимальные трёхшаговые итерационные методы восьмого порядка для решения систем нелинейных уравнений

Т. Жанлав[1, 2], Х. Отгондорж[2]

[1] Институт математики и информационных технологий, Монгольская Академия Наук, Улан-Батор, 13330, Монголия

[2] Факультет Прикладных Наук, Монгольский Государственный Университет Науки и Технологии, Улан-Батор, 14191, Монголия

Аннотация. В данной статье мы впервые предлагаем расширение оптимальных методов восьмого порядка на многомерный случай. Показано, что эти расширения сохранили свойства оптимальности исходных методов. Вычислительная эффективность предлагаемых методов сравнивается с известными методами. Проводится сравнение с другими методами. Для подтверждения теоретических результатов и эффективности методов включены численные эксперименты.

Ключевые слова: методы ньютоновского типа, системы нелинейных уравнений, порядок сходимости, оптимальность и расширение методов, индекс эффективности

Discrete & Continuous Models & Applied Computational Science
ISSN 2658-7149 (Online), 2658-4670 (Print)

2025, 33 (4) 404–410
http://journals.rudn.ru/miph

Research article

UDC 519.872, 519.217
PACS 07.05.Tp, 02.60.Pn, 02.70.Bf
DOI: 10.22363/2658-4670-2025-33-4-404-410 EDN: HSZAJF

On calculating the dimension of invariant sets of dynamic systems

Viktor M. Kadrov[1], Mikhail D. Malykh[1, 2]

[1] RUDN University, 6 Miklukho-Maklaya St, Moscow, 117198, Russian Federation

[2] Joint Institute for Nuclear Research, 6 Joliot-Curie St, Dubna, 141980, Russian Federation

(received: October 12, 2025; revised: October 25, 2025; accepted: October 30, 2025)

Abstract. This work investigates numerical approaches for estimating the dimension of invariant sets onto which the trajectories of dynamic systems "wind", with a focus on fractal and correlation dimensions. While the classical fractal dimension becomes computationally challenging in spaces of dimension greater than two, the correlation dimension offers a more efficient and scalable alternative. We develop and implement a computational method for evaluating the correlation dimension of large discrete point sets generated by numerical integration of differential equations. An analogy is noted between this approach and the Richardson–Kalitkin method for estimating the error of a numerical method. The method is tested on two representative systems: a conservative system whose orbit lies on a two-dimensional torus, and the Lorenz system, a canonical example of a chaotic flow with a non-integer attractor dimension. In both cases, the estimated correlation dimensions agree with theoretical predictions and previously reported results. The developed software provides an effective tool for analysing invariant manifolds of dynamical systems and is suitable for further studies, including those involving reversible difference schemes and high-dimensional systems.

Key words and phrases: correlation dimension, dynamic systems

For citation: Kadrov, V. M., Malykh, M. D. On calculating the dimension of invariant sets of dynamic systems. *Discrete and Continuous Models and Applied Computational Science* **33** (4), 404–410. doi: 10.22363/2658-4670-2025-33-4-404-410. edn: HSZAJF (2025).

1. Introduction

Numerical methods for integrating dynamic systems make it possible to find many thousands and even millions of points along a system's trajectory. However, the invariant sets that arise along the way do not always look like curved lines [1], so the challenge arises of developing software for calculating the dimensions of such invariant "manifolds."

Taking the neighborhood of a point on a smooth curve or surface, we expect to see how the curve turns into a line, and the surface into a plane, as this neighborhood is reduced. However, as early as the 19th century, the first examples of functions whose graphs do not become simpler with increasing scale appeared. Classic examples of such lines are the graph of the Weierstrass function or the trajectory of Brownian motion. A quantitative measure characterizing such an irregular structure was introduced in the 1980s and is called the *fractal dimension* [2].

Although the use of fractal dimension for studying dynamic systems with deterministic chaos [3, 4] is inferior in many ways to the methods of qualitative analysis of such systems [5], the method itself has found wide application in assessing the complexity and predictability of physical signals in natural systems, primarily biological ones [6], as well as in studying the interactions of coupled dynamic systems [7, 8]. We will consider the calculation of dimension as a method that, with relatively low computational costs, can provide a description of the invariant set of a dynamic system.

2. Fractal dimension

The classical approach to determining the fractal dimension of a set (see [2, 9]) is as follows. We describe an n-dimensional grid with a step of r around the set M under study. Let $N(r)$ be the number of cubes with sides r required to cover the set M. Then, as the linear dimensions of the grid decrease, the number of cubes asymptotically behaves as

$$N(r) \sim r^D, \quad (r \to 0),$$

where D is the fractal dimension of the set.

Although this method is quite intuitive, calculations for systems with dimensions greater than two present some difficulties. It should also be noted that it does not take into account the frequency of trajectories hitting a specific cube [10].

3. Correlation dimension

As an alternative to the fractal dimension, similar constructs can be used, among which we have chosen the correlation dimension, originally proposed for time series analysis [9–11].

Let N be a sufficiently large natural number, and $\{\vec{x}_1, \dots, \vec{x}_N\}$ be an arbitrary subset of points in the set M. The correlation sum is defined as

$$C(r) = \frac{1}{N^2} \sum_{i,j} [d(\vec{x}_i, \vec{x}_j) < r],$$

where $d(\vec{x}, \vec{y})$ is the Euclidean distance between points, and $\sum_{i,j}[d(\vec{x}_i, \vec{x}_j) < r]$ is the number of pairs of points satisfying the condition $d(\vec{x}_i, \vec{x}_j) < r$. If, for sufficiently small r, the correlation sum behaves as a power of r, that is,

$$C(r) \simeq r^{D_c}, \tag{1}$$

then its exponent D_c is called the correlation dimension of the set M. In Ref. [10] it is proved that D_c is a lower bound for the fractal dimension.

Defining the dimension of a set as a correlation dimension is computationally more convenient than using a fractal dimension, since the number of operations depends only on the number of points. Not less important is that the correlation dimension can also be applied to manifolds embedded in a high-dimensional space.

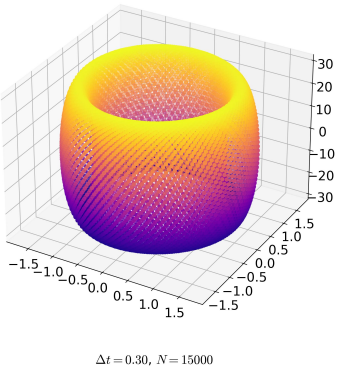


Figure 1. Solution of the system (2) from Example 1

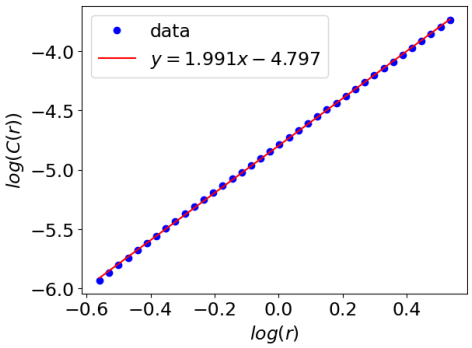


Figure 2. Dependence of the correlation sum on the radius r of the solution of the system (2) from Example 1

4. Calculation of correlation dimension

The calculation of correlation dimension of a given set of points was carried out using a method largely similar to the Richardson–Kalitkin method [12–16]. For various values of parameter r, the values of C were calculated and the dependence of C on r was plotted in logarithmic scale. Then the slope, which is interpreted as the correlation dimension, was determined using the least squares method.

When choosing the range for the parameter r, it should be noted that there are:

1) the maximum value r_{max}, above which all points of the finite sequence of points in the set M will be included in the correlation sum,
2) the minimum value r_{min}, below which the correlation sum is zero.

In practice, the linear segment can be much smaller than $[r_{min}, r_{max}]$, and the estimate can be corrupted by "tails" where the relation (1) is not satisfied. As in Richardson's method, these tails on the left, for small r, are due to roundoff error, while those on the right are due to the role of terms omitted in (1).

If the points $\{\vec{x}_i\}$ fill the classical manifold more or less uniformly, and the number N is sufficiently large, then the correlation dimension coincides with the ordinary dimension of the manifold. We will interpret the presence of a linear section on the logarithmic plot of C versus r as a condition for the applicability of the method.

5. Interpretation of the obtained value

When numerically integrating dynamical systems, we obtain a finite set of orbital points. Using this set of points as the set $\{\vec{x}_i\}$, we can calculate the correlation dimension of the orbit of a continuous dynamical system by making a number of difficult-to-formalize assumptions. The results of such calculations often coincide with theoretical predictions or, at least, are close to those obtained by other methods [11].

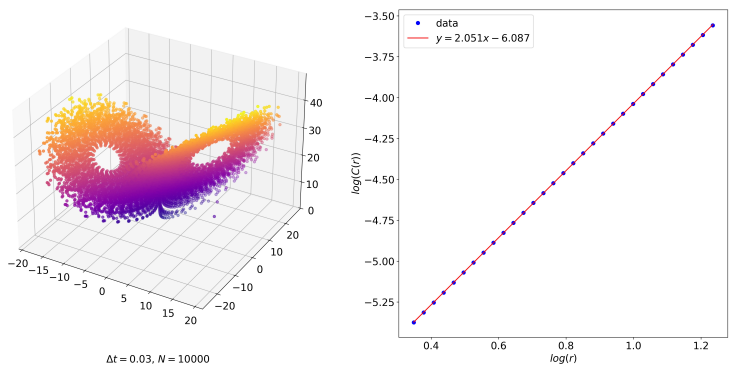


Figure 3. Solution of the Lorenz system (Example 2) and a plot of the correlation sum versus the radius r

Example 1. Consider a conservative system

$$\begin{cases} \dot{x} = y, \\ \dot{y} = -x + 2\epsilon yz, \\ \dot{z} = 1 - y^2 - \epsilon z^2. \end{cases} \tag{2}$$

For a small perturbation $\epsilon = 10^{-4}$, the trajectory passes along a two-dimensional torus [5, p. 3.4.1], which is clearly seen in Fig. 1, obtained by numerically integrating this system at $(x_0, y_0, z_0) = (1.7, 0, 0)$. The correlation dimension D_c of this set will be the slope of the linear section of Fig. 2, constructed on a double logarithmic scale. Therefore, in this way, the dimension is estimated at $D_C \approx 1.99$ against the geometric dimension $D = 2$.

Example 2. A well-known example of a system whose trajectory, found approximately by the Runge–Kutta method, has a non-integer dimension is E. Lorenz's system of convection equations.

$$\begin{cases} \dot{x} = 10(y - x), \\ \dot{y} = x(28 - z) - y, \\ \dot{z} = xy - \dfrac{8}{3}z. \end{cases}$$

Its dimension $D \approx 2.06$ was calculated by various methods [17, 18], and the value of D_c is also close to this number (Fig. 3).

6. Results

The method for computing the correlation dimension enables the efficient estimation of a set of points — such as, in our case, the trajectory trace of a dynamical system. As the examples demonstrate, the method is capable of accurately determining the dimensionality of both regular sets and those commonly classified as fractal. For a conservative system whose trajectory lies on a two-dimensional torus, we obtained a value close to 2; similarly, the estimated dimension of the Lorenz attractor is consistent with values reported in other studies. In contrast to the fractal dimension, the correlation-dimension approach readily allows the computation of this characteristic even for sets embedded in spaces of substantially higher dimension than three.

7. Discussion

In the numerical study of an integral curve, two parameters arise that characterize the numerical method: the time step Δt and the number of points N considered. Their product $T = N\Delta t$ yields the time interval under consideration. In Ref. [11], it was noted that these parameters significantly influence the dimensionality estimate.

If T is small, then what is actually being calculated is the germ of the integral curve, whose points fit perfectly onto the one-dimensional analytic curve by virtue of the local Cauchy theorem. Of course, we do not know the characteristic time T for a dynamical system, during which the system begins to exhibit its "global" properties. If Δt is not small enough, then the numerical solution may differ significantly from the exact solution, and therefore the calculated dimensionality will have no relation to the dimensionality of the desired orbit. Excessive reduction of Δt for a fixed T is very difficult, since it leads to prohibitively complex calculations. Despite the existence of theoretical estimates for the minimum number of points [19, 20], we will adhere to the approach of Ref. [11], which considered four chaotic systems and showed that 6,000-7,000 points were needed to obtain all the information about the trajectory, which is not very many by modern standards.

Some of these issues are resolved by considering the difference scheme as a discrete model describing the same physical phenomenon no worse than a continuous one. With this approach, the question of the dimensionality of the system's orbital closure is separated from the question of how this dimensionality changes depending on the step size Δt. Continuing the present work, we plan to use the software presented here to investigate the dimensionality of invariant sets found using reversible schemes with fixed and even deliberately large time steps [1].

8. Conclusion

The study examined approaches for estimating the dimension of a discrete set, specifically the fractal and correlation dimensions. The latter is more suitable for computational analyses in the task of determining the dimension of an invariant set of dynamical systems; therefore, experiments were conducted using two representative examples: a system whose trajectory lies on a two-dimensional torus, serving as an example of a regular set, and the Lorenz system as an extreme case. In both instances, the method produced satisfactory dimension estimates. The developed program for computing the dimension will facilitate future investigations of invariant manifolds of dynamical systems, particularly in combination with reversible difference schemes.

Author Contributions: Conceptualization, M. Malykh; methodology, M. Malykh and V. Kadrov; software, V. Kadrov; validation, V. Kadrov; writing—original draft preparation, V. Kadrov; writing—review and editing, M. Malykh. All authors have read and agreed to the published version of the manuscript.

Funding: The work was carried out with the financial support of the Russian Science Foundation (project No. 20-11-20257).

Data Availability Statement: Data sharing is not applicable.

Acknowledgments: The authors are grateful to prof. L.A. Sevastianov (RUDN) for attention to their research.

Conflicts of Interest: The authors declare no conflict of interest.

Declaration on Generative AI: The authors have not employed any Generative AI tools.

References

1. Malykh, M., Gambaryan, M., Kroytor, O. & Zorin, A. Finite Difference Models of Dynamical Systems with Quadratic Right-Hand Side. *Mathematics* **12,** 167. doi:10.3390/math12010167 (2024).
2. Mandelbrot, B. B. *The Fractal Geometry of Nature* en (Henry Holt and Company, 1983).

3. Rössler, O. E. Chaos in abstract kinetics: Two prototypes. *Bulletin of Mathematical Biology* **39,** 275–289. doi:10.1016/S0092-8240(77)80015-3 (1977).
4. Thompson, J. M. T. Chaos, fractals and their applications. *International Journal of Bifurcation and Chaos* **26.** Publisher: World Scientific, 1630035 (2016).
5. Магницкий, Н. А. *Теория динамического хаоса* ru (УРСС, 2011).
6. Kannathal, N., Acharya, U. R., Lim, C. & Sadasivan, P. Characterization of EEG—A comparative study. *Computer Methods and Programs in Biomedicine* **80,** 17–23. doi:10.1016/j.cmpb.2005.06.005 (2005).
7. Broock, W. A., Scheinkman, J. A., Dechert, W. D. & LeBaron, B. A test for independence based on the correlation dimension. *Econometric Reviews* **15,** 197–235. doi:10.1080/07474939608800353. eprint: https://doi.org/10.1080/07474939608800353 (1996).
8. Krakovská, A. Correlation Dimension Detects Causal Links in Coupled Dynamical Systems. *Entropy* **21.** doi:10.3390/e21090818 (2019).
9. Liu, Z. Chaotic time series analysis. *Mathematical Problems in Engineering* **2010.** doi:10.1155/2010/720190 (Apr. 2010).
10. Grassberger, P. & Procaccia, I. Measuring the strangeness of strange attractors. *Physica D: Nonlinear Phenomena* **9,** 189–208. doi:10.1016/0167-2789(83)90298-1 (1983).
11. JI, C., ZHU, H. & JIANG, W. Influence of sampling length and sampling interval on calculating the fractal dimension of chaotic attractors. *International Journal of Bifurcation and Chaos* **22,** 1250145. doi:10.1142/S0218127412501453 (2012).
12. Kalitkin, N. N., Al'shin, A. B., Al'shina, E. A. & Rogov, B. V. *Calculations on quasi-uniform grids* In Russian (Fizmatlit, Moscow, 2005).
13. Belov, A. A., Kalitkin, N. N. & Poshivaylo, I. P. Geometrically adaptive grids for stiff Cauchy problems. *Doklady Mathematics* **93,** 112–116. doi:10.1134/S1064562416010129 (2016).
14. Belov, A. A. & Kalitkin, N. N. Nonlinearity Problem in the Numerical Solution of Superstiff Cauchy Problems. *Mathematical Models and Computer Simulations* **8,** 638–650. doi:10.1134/S2070048216060065 (2016).
15. Belov, A. A., Kalitkin, N. N., Bulatov, P. & K., Z. E. Explicit methods for integrating stiff Cauchy problems. *Doklady Mathematics* **99,** 230–234. doi:10.1134/S1064562419020273 (2019).
16. Baddour, A., Gambaryan, M., Gonzalez, L. & Malykh, M. D. On Implementation of Numerical Methods for Solving Ordinary Differential Equations in Computer Algebra Systems. *Program. Comput. Soft.* **49,** 412–422. doi:10.1134/S0361768823020044 (2023).
17. Mori, H. & Fujisaka, H. Statistical Dynamics of Chaotic Flows. *Progress of Theoretical Physics* **63,** 1931–1944. doi:10.1143/PTP.63.1931. eprint: https://academic.oup.com/ptp/article-pdf/63/6/1931/5222197/63-6-1931.pdf (June 1980).
18. Viswanath, D. V. K. The fractal property of the Lorenz attractor. *Physica D: Nonlinear Phenomena* **190,** 115–128 (2004).
19. Smith, L. A. Intrinsic limits on dimension calculations. *Physics Letters A* **133,** 283–288. doi:10.1016/0375-9601(88)90445-8 (1988).
20. Nerenberg, M. A. H. & Essex, C. Correlation dimension and systematic geometric effects. *Physical review. A, Atomic, molecular, and optical physics* **42 12,** 7065–7074 (1990).

Information about the authors

Kadrov, Viktor M.—Student of Department of Computational Mathematics and Artificial Intelligence of RUDN University (e-mail: vmkadrov@yandex.ru, ORCID: 0009-0008-9394-4874)

Malykh, Mikhail D.—DSc., Head of Department of Computational Mathematics and Artificial Intelligence of RUDN University; Senior Researcher of Joint Institute for Nuclear Research (e-mail: malykh-md@rudn.ru, ORCID: 0000-0001-6541-6603, ResearcherID: P-8123-20168, Scopus Author ID: 6602318510)

УДК 519.872, 519.217
PACS 07.05.Tp, 02.60.Pn, 02.70.Bf
DOI: 10.22363/2658-4670-2025-33-4-404-410 EDN: HSZAJF

О вычислении размерности инвариантных множеств динамических систем

В. М. Кадров[1], М. Д. Малых[1, 2]

[1] Российский университет дружбы народов, ул. Миклухо-Маклая, д. 6, Москва, 117198, Российская Федерация

[2] Объединённый институт ядерных исследований, ул. Жолио-Кюри, д. 6, Дубна, 141980, Российская Федерация

Аннотация. В работе рассматриваются численные подходы к оценке размерности инвариантных множеств, на которые «навиваются» траектории динамических систем: методы расчёта фрактальной и корреляционной размерности. Классическая фрактальная размерность становится вычислительно трудоёмкой при работе с пространствами размерности выше двух, тогда как корреляционная размерность представляет собой более эффективную альтернативу. Разработан и реализован вычислительный метод для оценки корреляционной размерности больших дискретных наборов точек, полученных в результате численного интегрирования дифференциальных уравнений. Отмечена аналогия данного подхода с методом Ричардсона–Калиткина для оценки погрешности численного метода. Предложенный метод протестирован на двух характерных примерах: консервативной системе, чья орбита лежит на двумерном торе, и системе Лоренца — классическом примере хаотической система с нецелой размерностью аттрактора. В обоих случаях полученные оценки корреляционной размерности согласуются с теорией и ранее опубликованными результатами. Разработанное программное обеспечение послужит эффективным инструментом для анализа инвариантных многообразий динамических систем и подходит для дальнейших исследований, в особенности для компьютерных экспериментов с использованием обратимых разностных схем, а также для систем высокой размерности.

Ключевые слова: корреляционная размерность, динамические системы

Discrete & Continuous Models
& Applied Computational Science
ISSN 2658-7149 (Online), 2658-4670 (Print)

2025, 33 (4) 411–439
http://journals.rudn.ru/miph

Review

UDC 511.84,512.523.282.2,519.711
DOI: 10.22363/2658-4670-2025-33-4-411-439 EDN: HPZRYA

Dual quaternion representation of points, lines and planes

Migran N. Gevorkyan[1], Nikita A. Vishnevskiy[1], Kirill V. Didus[1],
Anna V. Korolkova[1], Dmitry S. Kulyabov[1,2]

[1] RUDN University, 6 Miklukho-Maklaya St, Moscow, 117198, Russian Federation

[2] Joint Institute for Nuclear Research, 6 Joliot-Curie St, Dubna, 141980, Russian Federation

(received: July 6, 2025; revised: July 25, 2025; accepted: August 10, 2025)

Abstract. *Background* The bulk of the work on dual quaternions is devoted to their application to describe helical motion. Little attention is paid to the representation of points, lines, and planes (primitives) using them. *Purpose* It is necessary to consistently present the dual quaternion theory of the representation of primitives and refine the mathematical formalism. *Method* It uses the algebra of dual numbers, quaternions and dual quaternions, as well as elements of the theory of screws and sliding vectors. *Results* Formulas have been obtained and systematized that use exclusively dual quaternionic operations and notation to solve standard problems of three-dimensional geometry. *Conclusions* Dual quaternions can serve as a full-fledged formalism for the algebraic representation of a three-dimensional projective space.

Key words and phrases: dual numbers, quaternions, dual quaternions, projective space

For citation: Gevorkyan, M. N., Vishnevskiy, N. A., Didus, K. V., Korolkova, A. V., Kulyabov, D. S. Dual quaternion representation of points, lines and planes. *Discrete and Continuous Models and Applied Computational Science* **33** (4), 411–439. doi: 10.22363/2658-4670-2025-33-4-411-439. edn: HPZRYA (2025).

1. Introduction

The parabolic biquaternions (dual quaternions) discussed in this paper were first considered by Clifford. However, they were systematically studied later by E. Studi [1, 2] and A. P. Kotelnikov [3].

The article [4] analyzes a large number of publications and calculates the frequency of mentions of the term dual quaternion. It can be concluded that less than 100 works mentioning dual quaternions were published throughout the 20th century, but already in the early 2000s the number of works increased dramatically. It is worth noting that the authors apparently did not take into account the theory of screws in the calculation, which is related to dual quaternions, but uses a slightly different notation [5–9].

The huge boost in the number of publications can be explained by the development of computer graphics and robotics. It is in these areas that dual quaternions have found their application [10–15], although they were initially developed in works on mechanics.

1.1. Structure of the paper

The paper consists of an introduction, four parts, results and conclusion.

- The first part summarizes the theoretical information about dual numbers.
- The second part contains the necessary theory concerning quaternions. We only avoid the question of using quaternions for rotation in space.
- The third part describes in detail the theory of dual quaternions. The main focus is on notation, so when calculating scalar and helical products, all calculations are described in great detail (it may even seem to some that they are overly detailed).
- In the fourth part, the dual quaternion algebra is used to represent points, lines, and planes in a three-dimensional projective space. Some formulas are derived.
- The main results of the work are summarized in tables, which are given in the section 6.

1.2. Notations and conventions

The following naming conventions are accepted in this article

- Quaternions are indicated by lowercase Latin letters from the end of the alphabet: p, q, r. The components of the quaternions are indicated by the same letters, but with the indexes p_0, p_1, etc.
- Dual quaternions are indicated by uppercase Latin letters from the end of the alphabet: P, Q, R. The components of the quaternions are indicated by the same letters, but with the indexes P_0, P_1, etc.
- Vectors and pure quaternions are indicated by lowercase bold Latin letters: $\mathbf{q}$, $\mathbf{v}$, etc.
- Pure dual quaternions are indicated by uppercase bold Latin letters: $\mathbf{Q}$, $\mathbf{V}$, etc.
- Individual scalars (real numbers) are denoted by the Greek letters α, β, etc.

To avoid ambiguity in the notation system, we do not use multiple quaternions and dual quaternions designated by the same letter, but distinguished by an index. The only exceptions are dual quaternions of points, lines, and planes, the components of which are denoted by letters other than the letters denoting these dual quaternions.

2. Dual numbers

The dual number z is algebraically defined as follows:

$$z = a + \varepsilon b,\ a, b \in \mathbb{R},\ \varepsilon^2 = 0, \varepsilon \neq 0.$$

The number a is called the real or *main* part, and b — is called the imaginary, *dual* or *moment* part. Numbers with zero real part will be called *pure* dual numbers. The special symbol ε is referred to as the dual or parabolic imaginary unit. It is also known as Clifford's complexity symbol [16, p. 43]

2.1. Algebraic form

Using only the definition of the dual unit $\varepsilon^2 = 0$, basic algebraic operations for dual numbers can be introduced $z_1 = a_1 + \varepsilon b_1$ и $z_2 = a_2 + \varepsilon b_2$.

Addition $z_1 + z_2 = (a_1 + a_2) + \varepsilon(b_1 + b_2)$.
Subtraction $z_1 - z_2 = (a_1 - a_2) + \varepsilon(b_1 - b_2)$.
Multiplication $z_1 \cdot z_2 = (a_1 + \varepsilon b_1) \cdot (a_2 + \varepsilon b_2) = a_1 a_2 + \varepsilon a_1 b_2 + \varepsilon a_2 b_1 + \varepsilon^2 b_1 b_2 = a_1 a_2 + \varepsilon(a_1 b_2 + b_1 a_2)$.
Conjugation $\bar{z} = a - \varepsilon b$.

Using the conjugation operation, the squared modulus of a dual number is given by:

$$|z|^2 = z\bar{z} = (a + \varepsilon b) \cdot (a - \varepsilon b) = a^2.$$

Taking the square root of the real number a^2, yields:

Modulus $|z| = |a|$.

The real part of a dual number can be calculated using the formula:

$$\Re z = \frac{1}{2}(z + \bar{z}) = \frac{1}{2}(a + \varepsilon b + a - \varepsilon b) = a.$$

Multiplicative inverse divided by the number z is found by noting that

$$1 = \frac{z\bar{z}}{z\bar{z}} = z\frac{\bar{z}}{z\bar{z}},$$

hence

$$z^{-1} = \frac{\bar{z}}{z\bar{z}} = \frac{a - \varepsilon b}{a^2}.$$

Division of two dual numbers z_1 and z_2 is defined for all z_2 such that $|z_2| \neq 0$:

$$\frac{z_1}{z_2} = \frac{a_1 + \varepsilon b_1}{a_2 + \varepsilon b_2} = \frac{(a_1 + \varepsilon b_1)(a_2 - \varepsilon b_2)}{(a_2 + \varepsilon b_2)(a_2 - \varepsilon b_2)} = \frac{a_1}{a_2} + \varepsilon\frac{b_1 a_2 - b_2 a_1}{a_2^2}.$$

Several important remarks follow from the obtained formulas.

- First, note that the multiplicative inverse is not defined for all nonzero dual numbers z, since the formula for the inverse is valid only when $a \neq 0$.
- It follows from the previous point that the set of dual numbers is not a field, because the requirement of existence of a multiplicative inverse for every nonzero element is not satisfied. Thus, all numbers of the form $z = \varepsilon b$ are nonzero but have no inverse, since $\Re z = 0$.
- Dual numbers of the form $z = b\varepsilon$ are *nontrivial zero divisors*, since for such numbers the equality holds
$$(b_1\varepsilon) \cdot (b_2\varepsilon) = b_1 b_2 \varepsilon^2 = 0,$$
- From the above properties of the algebraic operations, it is clear that in no case do the imaginary parts of numbers contribute to the real part of the result of these operations.

To complement the remark on zero divisors, consider the following definition. An element of a ring $\alpha \in \mathbb{K}$ is called *left zero divisor*, if there exists an element of the ring $\beta \neq 0_{\mathbb{K}}$ such that $\alpha \cdot \beta = 0_{\mathbb{K}}$. Similarly, if $\beta \cdot \alpha = 0_{\mathbb{K}}$, then α is *right zero divisor*. It is obvious that the zero element $0_{\mathbb{K}}$ is both a left and a right zero divisor, since $0_{\mathbb{K}} \cdot \beta = \beta \cdot 0_{\mathbb{K}} = 0_{\mathbb{K}}$. Therefore, $0_{\mathbb{K}}$ is a *trivial* zero divisor, and zero divisors distinct from $0_{\mathbb{K}}$ are called nontrivial or improper.

2.2. "Trigonometric" and "exponential" forms

A dual number z, such that $|z| \neq 0$, can be written in the form:

$$a + \varepsilon b = a\Big(1 + \frac{b}{a}\varepsilon\Big) = a(1 + \varphi\varepsilon),$$

where $\varphi = \operatorname{Arg} z = b/a$ are called the *argument* or *parameter* of the dual number z. This representation is a kind of analogue of both the trigonometric form and the exponential form of an ordinary complex number. Hereinafter, this form will be referred to as the exponential form of a dual number

For the conjugate number $\bar{z}$, the exponential form is:

$$\bar{z} = a(1 - \varphi\varepsilon),$$

modulus $|\bar{z}| = |z|$, and argument $\operatorname{Arg}\bar{z} = -b/a$.

The analogy with the exponential form of a complex number continues for multiplication and division. For the product of two dual numbers z_1 and z_2:

$$z = a_1(1 + \varphi_1\varepsilon) \cdot a_2(1 + \varphi_2\varepsilon) = a_1a_2\big(1 + (\varphi_1 + \varphi_2)\varepsilon\big),$$

that is, when multiplying, the arguments are added and the modules are multiplied. For division

$$\frac{z_1}{z_2} = \frac{a_1(1 + \varphi_1\varepsilon_1)}{a_2(1 + \varphi_2\varepsilon)} = \frac{a_1(1 + \varphi_1\varepsilon)a_2(1 - \varphi_2\varepsilon)}{a_2(1 + \varphi_2\varepsilon)a_2(1 - \varphi_2\varepsilon)} = \frac{a_1\big(1 + (\varphi_1 - \varphi_2)\varepsilon\big)}{a_2\big(1 - \varphi_2\varepsilon + \varphi_2\varepsilon\big)} = \frac{a_1}{a_2}\big(1 + (\varphi_1 - \varphi_2)\varepsilon\big),$$

that is, when dividing, the arguments are subtracted and the modules are divided. In the case of division, the number z_2 must have nonzero modulus $|z_2| \neq 0$.

In exponential form, raising to a natural power n has a particularly simple expression:

$$z^n = \big(a(1 + \varphi\varepsilon)\big)^n = a \cdot a \cdot \ldots \cdot a\big(1 + (\varphi + \varphi + \ldots + \varphi)\big) = a^n(1 + n\varphi\varepsilon) = a^n + \varepsilon n a^{n-1}b.$$

To find $\sqrt[n]{z} = \sqrt[n]{a(1 + \varphi\varepsilon)}$ assume that $\sqrt[n]{z} = z_1$, Then, by definition $z_1^n = z$, hence

$$a_1^n(1 + n\varphi_1\varepsilon) = z = a(1 + \varphi\varepsilon),$$

which yields $a_1 = \sqrt[n]{a}$ and $\varphi_1 = \varphi/n$, and therefore

$$\sqrt[n]{z} = \sqrt[n]{r}\Big(1 + \frac{\varphi}{n}\varepsilon\Big).$$

Note that for odd n the root $\sqrt[n]{z}$ always exists, whereas for even n it exists only for z with nonnegative modulus $|z| = a \neq 0$. In algebraic form, the formula for the root is:

$$\sqrt[n]{a + b\varepsilon} = \sqrt[n]{a} + \frac{ba^{\frac{1-n}{n}}}{n}\varepsilon, \quad \text{в частности} \quad \sqrt{a + b\varepsilon} = \sqrt{a} + \frac{b}{2\sqrt{a}} = \sqrt{a}\Big(1 + \frac{b}{2a}\Big). \tag{1}$$

It is noteworthy that neither raising to a power nor taking a root leads to any contribution of the dual parts to the real parts of the results.

2.3. Matrix form

All algebraic operations on dual numbers can be reduced to matrix operations by setting

$$\varepsilon \leftrightarrow \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \quad z = a + b\varepsilon \leftrightarrow \begin{pmatrix} a & b \\ 0 & a \end{pmatrix}.$$

Then, for example

$$z_1 \cdot z_2 \leftrightarrow \begin{pmatrix} a_1 & b_1 \\ 0 & a_1 \end{pmatrix}\begin{pmatrix} a_2 & b_2 \\ 0 & a_2 \end{pmatrix} = \begin{pmatrix} a_1a_2 & a_1b_2 + a_2b_1 \\ 0 & a_1a_2 \end{pmatrix} \leftrightarrow a_1a_2 + (a_1b_2 + a_2b_1)\varepsilon.$$

Theoretically, from a computational point of view, such a reduction can be justified when the programming language supports vectorized matrix operations. In practice, however, the performance gain is usually not significant.

Table 1
Trigonometric and inverse trigonometric functions of dual numbers expressed through functions of a real variable.

$\sin(a+\varepsilon b) = \sin a + b\varepsilon \cos a$	$\arcsin(a+\varepsilon b) = \arcsin a + \dfrac{b\varepsilon}{\sqrt{1-a^2}}$
$\cos(a+\varepsilon b) = \cos a - b\varepsilon \sin a$	$\arccos(a+\varepsilon b) = \arccos a - \dfrac{b\varepsilon}{\sqrt{1-a^2}}$
$\operatorname{tg}(a+\varepsilon b) = \operatorname{tg} a + \dfrac{b\varepsilon}{\cos^2 a}$	$\operatorname{arctg}(a+\varepsilon b) = \operatorname{arctg} a + \dfrac{b\varepsilon}{1+a^2}$
$\operatorname{ctg}(a+\varepsilon b) = \operatorname{ctg} a - \dfrac{b\varepsilon}{\sin^2 a}$	$\operatorname{arcctg}(a+\varepsilon b) = \operatorname{arctg} a - \dfrac{b\varepsilon}{1+a^2}$

2.4. Taylor expansion

Using the defining property of the dual imaginary unit $\varepsilon^2 = \varepsilon^3 = \ldots = \varepsilon^n = 0$ for any natural power, consider the Maclaurin series for the exponential of a pure dual number:

$$\exp(b\varepsilon) = \sum_{n=0}^{\infty} \frac{(b\varepsilon)^n}{n!} = 1 + b\varepsilon + \frac{b^2\varepsilon^2}{2!} + \ldots = 1 + b\varepsilon,$$
$$\exp(a+b\varepsilon) = e^a e^{b\varepsilon} = e^a(1+b\varepsilon).$$

A more general formula is derived from the formal Taylor series for the function $f(z)$ at the point a:

$$f(a+\varepsilon b) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)(a+\varepsilon b-a)^n}{n!} = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)\varepsilon^n b^n}{n!} =$$
$$= f(a) + f'(a)b\varepsilon + \frac{f''(a)\varepsilon^2 b^2}{2!} + \ldots = f(a) + f'(a)b\varepsilon.$$

This yields an extremely important formula:

$$f(a+\varepsilon b) = f(a) + f'(a)b\varepsilon,$$

which provides a method for calculating the values of functions from a dual number, if the value of the derivative of the real part of the number $f'(a)$ is known. On the other hand, the same formula allows to calculate the value of the derivative for the value of the argument equal to a, which is used in automatic differentiation algorithms.

2.5. Elementary Functions of Dual Numbers

The formula $f(a+\varepsilon b) = f(a) + f'(a)b\varepsilon$ makes it possible to extend elementary functions to the set of dual numbers, since the right-hand side of the formula contains only the values of the function f from the real number a. For illustration, a brief summary of some basic elementary functions is presented in tables 1 and 2.

3. Quaternion algebra

The theory of quaternions is well known. We will highlight the following books [17–19]. Next, we briefly outline the quaternion algebra in order to coordinate the notation and basic concepts with the further presentation of the biquaternion algebra.

Table 2

Power functions, exponent, and logarithm of dual numbers

$(a+\varepsilon b)^n = a^n + na^{n-1}b\varepsilon$	$\exp(a+\varepsilon b) = e^a + \varepsilon b e^a$
$\sqrt[n]{a+b\varepsilon} = \sqrt[n]{a}\left(1+\frac{b\varepsilon}{na}\right)$	$\log_c(a+\varepsilon b) = \log_c a + \frac{b\varepsilon}{a\ln a}$

3.1. Basic concepts of quaternion algebra

3.1.1. Basis elements of a quaternion

Let us denote the neutral element of the quaternion algebra by o (lowercase *O*). Geometrically, it is associated with the origin (the center of the coordinate system) in three-dimensional space $\mathbb{R}^3$. Algebraically, it also serves as the scalar unit [19, 20]. Using this notation, a quaternion is written in the following form:

$$q = q_0\mathrm{o} + q_1\mathrm{i} + q_2\mathrm{j} + q_3\mathrm{k},$$

Where q_0, q_1, q_2, q_3 are some real numbers. The quaternion q can also be associated with a point in projective space, written in homogeneous coordinates $(q_1, q_2, q_3 \mid q_0)$ [20].

In turn, the basis element o is associated with the proper (finite) point of the origin of coordinates, and the basis elements i, j, k with points at infinity:

$$\mathrm{o} \leftrightarrow O = \begin{bmatrix}0\\0\\0\\1\end{bmatrix} \quad \mathrm{i} \leftrightarrow \vec{\mathbf{i}} = \begin{bmatrix}1\\0\\0\\0\end{bmatrix} \quad \mathrm{j} \leftrightarrow \vec{\mathbf{j}} = \begin{bmatrix}0\\1\\0\\0\end{bmatrix} \quad \mathrm{k} \leftrightarrow \vec{\mathbf{k}} = \begin{bmatrix}0\\0\\1\\0\end{bmatrix}.$$

A quaternion of the form $p = \mathrm{o} + x\mathrm{i} + y\mathrm{j} + z\mathrm{k}$ is associated with an affine point having homogeneous coordinates $(x, y, z \mid 1)$ and Cartesian coordinates (x, y, z).

A pure quaternion of the form $v = v_x\mathrm{i} + v_y\mathrm{j} + v_z\mathrm{k}$ is associated with a point at infinity with coordinates $(v_x, v_y, v_z \mid 0)$ or with a free vector $\mathbf{v} = (v_x, v_y, v_z)^T$ in Cartesian space $\mathbb{R}^3$.

3.1.2. Quaternion multiplication

To derive the formula for quaternion multiplication, it is sufficient to derive the multiplication table for the basis elements $\langle \mathrm{o}, \mathrm{i}, \mathrm{j}, \mathrm{k}\rangle$. In turn, for this it is sufficient to the axiomatic relation introduced by Hamilton:

$$\mathrm{i}^2 = \mathrm{j}^2 = \mathrm{k}^2 = \mathrm{ijk} = -1\mathrm{o} = -\mathrm{o}.$$

Let us add the equality $\mathrm{o}^2 = \mathrm{o}$ to it, since algebraically o is a scalar unit. Then the multiplication table of the quaternion basis elements takes the form (2)

	o	i	j	k
o	o	i	j	k
i	i	−o	k	−j
j	j	−k	−o	i
k	k	j	−i	−o

(2)

Now the multiplication of quaternions can be performed using table 2 by expanding the brackets for the replacement of the products of the basis elements:

$$\begin{aligned} pq = (p_0\mathrm{o} + p_1\mathrm{i} + p_2\mathrm{j} + p_3\mathrm{k})(q_0\mathrm{o} + q_1\mathrm{i} + q_2\mathrm{j} + q_3\mathrm{k}) = \\ + p_0q_0\mathrm{o} + p_0q_1\mathrm{i} + p_0q_2\mathrm{j} + p_0q_3\mathrm{k} + p_1q_0\mathrm{io} + p_1q_1\mathrm{ii} + p_1q_2\mathrm{ij} + \\ + p_1q_3\mathrm{ik} + p_2q_0\mathrm{jo} + p_2q_1\mathrm{ji} + p_2q_2\mathrm{jj} + p_2q_3\mathrm{jk} + p_3q_0\mathrm{ko} + p_3q_1\mathrm{ki} + p_3q_2\mathrm{kj} + p_3q_3\mathrm{kk} = \\ = p_0q_0\mathrm{o} + p_0q_1\mathrm{i} + p_0q_2\mathrm{j} + p_0q_3\mathrm{k} + p_1q_0\mathrm{i} - p_1q_1\mathrm{o} + p_1q_2\mathrm{k} - p_1q_3\mathrm{j} + p_2q_0\mathrm{j} - p_2q_1\mathrm{k} - p_2q_2\mathrm{o} + p_2q_3\mathrm{i} + \\ + p_3q_0\mathrm{k} + p_3q_1\mathrm{j} - p_3q_2\mathrm{i} - p_3q_3\mathrm{o}. \end{aligned}$$

Next, we will bring similar and group the terms around the basis elements:

$$\begin{aligned} pq = (p_0q_0 - p_1q_1 - p_2q_2 - p_3q_3)\mathrm{o} + (p_0q_1 + p_1q_0 + p_2q_3 - p_3q_2)\mathrm{i} + \\ + (p_0q_2 - p_1q_3 + p_2q_0 + p_3q_1)\mathrm{j} + (p_0q_3 + p_1q_2 - p_2q_1 + p_3q_0)\mathrm{k} = \\ = \big(p_0q_0 - (p_1q_1 + p_2q_2 + p_3q_3)\big)\mathrm{o} + p_0(q_1\mathrm{i} + q_2\mathrm{j} + q_3\mathrm{k}) + \\ + q_0(p_1\mathrm{i} + p_2\mathrm{j} + p_3\mathrm{k}) + (p_2q_3 - p_3q_2)\mathrm{i} + (p_3q_1 - p_1q_3)\mathrm{j} + (p_1q_2 - p_2q_1)\mathrm{k}. \end{aligned}$$

A more concise form can be written using the scalar and vector products:

$$pq = (p_0q_0 - (\mathbf{p}, \mathbf{q}))\mathrm{o} + p_0\mathbf{q} + q_0\mathbf{p} + \mathbf{p} \times \mathbf{q}.$$

Particularly, for pure quaternions $p = 0\mathrm{o} + \mathbf{p}$ and $q = 0\mathrm{o} + \mathbf{q}$, the formula simplifies to:

$$\mathbf{pq} = -(\mathbf{p}, \mathbf{q})\mathrm{o} + \mathbf{p} \times \mathbf{q}.$$

Using quaternion multiplication, let us compute the square of a quaternion:

$$p^2 = pp = (p_0^2 - \|\mathbf{p}\|^2) + p_0\mathbf{p} + p_0\mathbf{p} + \mathbf{p} \times \mathbf{p} = (p_0^2 - \|\mathbf{p}\|^2)\mathrm{o} + 2p_0\mathbf{p}. \tag{3}$$

The square of a pure quaternion coincides with the square of its norm taken with a minus sign and is a scalar number:

$$\mathbf{p}^2 = \mathbf{pp} = -\|\mathbf{p}\|^2\mathrm{o}$$

For pure quaternions, the scalar and vector products can be expressed through quaternion multiplication. Consider two pure quaternions $\mathbf{p}$ and $\mathbf{q}$ and write

$$\begin{aligned} \mathbf{pq} &= -(\mathbf{p}, \mathbf{q})\mathrm{o} + \mathbf{p} \times \mathbf{q}, \\ \mathbf{qp} &= -(\mathbf{q}, \mathbf{p})\mathrm{o} + \mathbf{q} \times \mathbf{p} = -(\mathbf{q}, \mathbf{p})\mathrm{o} - \mathbf{q} \times \mathbf{p}. \end{aligned}$$

From which stems:

$$\mathbf{pq} + \mathbf{qp} = -2(\mathbf{p}, \mathbf{q})\mathrm{o} \quad \text{and} \quad \mathbf{pq} - \mathbf{qp} = \mathbf{p} \times \mathbf{q} + \mathbf{p} \times \mathbf{q},$$

$$-(\mathbf{p}, \mathbf{q})\mathrm{o} = \frac{1}{2}(\mathbf{pq} + \mathbf{qp}) \quad \text{and} \quad \mathbf{p} \times \mathbf{q} = \frac{1}{2}(\mathbf{pq} - \mathbf{qp}).$$

The expression for the vector product remains valid for quaternions of general form:

$$\left.\begin{aligned} pq &= (p_0q_0 - (\mathbf{p}, \mathbf{q}))\mathrm{o} + p_0\mathbf{q} + q_0\mathbf{p} + \mathbf{p} \times \mathbf{q} \\ qp &= (p_0q_0 - (\mathbf{p}, \mathbf{q}))\mathrm{o} + q_0\mathbf{p} + p_0\mathbf{q} + \mathbf{q} \times \mathbf{p} \end{aligned}\right\} \Rightarrow \mathbf{p} \times \mathbf{q} - \mathbf{q} \times \mathbf{p} = 2\mathbf{p} \times \mathbf{q} = pq - qp,$$

$$\mathbf{p} \times \mathbf{q} = \frac{1}{2}(pq - qp).$$

Let us compute the triple quaternion product of pure quaternions **qpq**, which we will also call the sandwich formula without conjugation.

$$\begin{aligned}\mathbf{qpq} &= \mathbf{q}\big(-(\mathbf{p},\mathbf{q})\mathrm{o} + \mathbf{p}\times\mathbf{q}\big) = \\ &= -(\mathbf{p},\mathbf{q})\mathbf{q} + \mathbf{q}(\mathbf{p}\times\mathbf{q}) = -(\mathbf{p},\mathbf{q})\mathbf{q} - (\mathbf{q},\mathbf{p}\times\mathbf{q})\mathrm{o} + \mathbf{q}\times(\mathbf{p}\times\mathbf{q}) = \\ &= -(\mathbf{p},\mathbf{q})\mathbf{q} + (\mathbf{q},\mathbf{q})\mathbf{p} - (\mathbf{p},\mathbf{q})\mathbf{q} = -2(\mathbf{p},\mathbf{q})\mathbf{q} + (\mathbf{q},\mathbf{q})\mathbf{p} = \|\mathbf{q}\|^2\mathbf{p} - 2(\mathbf{p},\mathbf{q})\mathbf{q}.\end{aligned}$$

$$\mathbf{qpq} = \|\mathbf{q}\|^2\mathbf{p} - 2(\mathbf{p},\mathbf{q})\mathbf{q}. \tag{4}$$

Formula (4) will allow simplifying calculations when computing the reflection of a point relative to a plane[1][2].

3.1.3. Quaternion conjugation

Let us introduce the operation of quaternion *conjugation*. If a quaternion $p = p_0\mathrm{o} + \mathbf{p}$ is given, then its conjugate is defined by the following formula:

$$p^* = p_0\mathrm{o} - \mathbf{p} = p_0\mathrm{o} - p_1\mathrm{i} - p_2\mathrm{j} - p_3\mathrm{k}.$$

To follow up, we compute:

$$pp^* = (p_0\mathrm{o} + \mathbf{p})(p_0\mathrm{o} - \mathbf{p}) = p_0^2\mathrm{o} - p_0\mathbf{p} + p_0\mathbf{p} - \mathbf{pp} = p_0^2\mathrm{o} - \mathbf{pp},$$

Where $\mathbf{pp} = -(\mathbf{p},\mathbf{p})\mathrm{o} + \mathbf{p}\times\mathbf{p} = -(\mathbf{p},\mathbf{p})\mathrm{o}$, which allows us to write:

$$pp^* = p_0^2\mathrm{o} + (\mathbf{p},\mathbf{p})\mathrm{o} = \big(p_0^2 + (\mathbf{p},\mathbf{p})\big)\mathrm{o} = (p_0^2 + p_1^2 + p_2^2 + p_3^2)\mathrm{o}.$$

The *module* of a quaternion is the expression

$$|p| = \sqrt{pp^*} = \sqrt{p_0^2 + p_1^2 + p_2^2 + p_3^2},$$

and the *norm* of a pure quaternion is the expression

$$\|\mathbf{p}\| = \sqrt{(\mathbf{p},\mathbf{p})} = \sqrt{p_1^2 + p_2^2 + p_3^2}.$$

Let us show that

$$(pq)^* = q^*p^*,$$

using the formula of quaternion multiplication:

$$\begin{aligned}(pq)^* &= \big[\big(p_0q_0 - (\mathbf{p},\mathbf{q})\big)\mathrm{o} + p_0\mathbf{q} + q_0\mathbf{p} + \mathbf{p}\times\mathbf{q}\big]^* = \\ &= \big(p_0q_0 - (\mathbf{p},\mathbf{q})\big)\mathrm{o} - p_0\mathbf{q} - q_0\mathbf{p} - \mathbf{p}\times\mathbf{q} = \\ &= \big(p_0q_0 - (-\mathbf{p},-\mathbf{q})\big)\mathrm{o} + p_0(-\mathbf{q}) + q_0(-\mathbf{p}) - (-\mathbf{p})\times(-\mathbf{q}) = \\ &= \big(p_0q_0 - (-\mathbf{q},-\mathbf{p})\big)\mathrm{o} + p_0(-\mathbf{q}) + q_0(-\mathbf{p}) + (-\mathbf{q})\times(-\mathbf{p}) = q^*p^*.\end{aligned}$$

Using the expression $(pq)^* = q^*p^*$ we prove the property of the module of the product of quaternions:

$$|pq| = \sqrt{(pq)(pq)^*} = \sqrt{pqq^*p^*} = \sqrt{p|q|^2p^*} = \sqrt{|p|^2|q|^2} = |p||q| \Rightarrow |pq| = |p||q|.$$

3.1.4. Scalar product of quaternions

Following [20] we introduce the operation of scalar product of two quaternions of general form. Consider two quaternions $p = p_0\mathrm{o} + p_1\mathrm{i} + p_2\mathrm{j} + p_3\mathrm{k}$ and $q = q_0\mathrm{o} + q_1\mathrm{i} + q_2\mathrm{j} + q_3\mathrm{k}$. We compute:

$$pq^* = \big(p_0q_0 + (\mathbf{p},\mathbf{q})\big)\mathrm{o} - p_0\mathbf{q} + q_0\mathbf{p} - \mathbf{p}\times\mathbf{q},$$

$$qp^* = \big(p_0q_0 + (\mathbf{p},\mathbf{q})\big)\mathrm{o} + p_0\mathbf{q} - q_0\mathbf{p} - \mathbf{q}\times\mathbf{p} = \big(p_0q_0 + (\mathbf{p},\mathbf{q})\big)\mathrm{o} + p_0\mathbf{q} - q_0\mathbf{p} + \mathbf{p}\times\mathbf{q}.$$

We calculate the sum of the products

$$pq^* + qp^* = 2\big(p_0q_0 + (\mathbf{p},\mathbf{q})\big)\mathrm{o} \Rightarrow \big(p_0q_0 + (\mathbf{p},\mathbf{q})\big)\mathrm{o} = \frac{1}{2}\big(pq^* + qp^*\big).$$

Let us define the *scalar product of quaternions* by the following formula

$$(p,q) = \frac{1}{2}(pq^* + qp^*) = \big(p_0q_0 + (\mathbf{p},\mathbf{q})\big)\mathrm{o} = p_0q_0 + p_1q_1 + p_2q_2 + p_3q_3.$$

It is easy to see that the definition of the scalar product of quaternions is consistent with the definition of the module of a quaternion:

$$|p|^2 = (p,p) = \big(p_0^2 + \|\mathbf{p}\|^2\big).$$

3.1.5. Unit quaternion

Consider a quaternion $q = q_0\mathrm{o} + q_1\mathrm{i} + q_2\mathrm{j} + q_3\mathrm{k}$ with a unit module $|q|^2 = q_0^2 + \|q\|^2 = 1$. Always there is such a value θ that $q_0^2 = \cos^2\theta$ and $\|\mathbf{q}\|^2 = \sin^2\theta$ due to the basic trigonometric identity

$$|q|^2 = q_0^2 + |\mathbf{q}|^2 = \cos^2\theta + \sin^2\theta = 1.$$

Let us note that the vector part $\mathbf{q}$ may not be a unit vector, but it can always be expressed through a unit vector $\mathbf{u}$ as follows:

$$\frac{\mathbf{q}}{\|\mathbf{q}\|} = \mathbf{u} \Rightarrow \mathbf{q} = \mathbf{u}\|\mathbf{q}\| = \sin\theta\mathbf{u},$$

Where $\mathbf{u} = u_1\mathrm{i} + u_2\mathrm{j} + u_3\mathrm{k}$ and $\|\mathbf{u}\| = 1$. The unit quaternion can then be written in the trigonometric form: $q = \cos\theta + \sin\theta\mathbf{u}$.

The following terminology is sometimes used. The module of an arbitrary quaternion $|q|$ is called the *tensor* of the quaternion, and the normalized quaternion

$$\frac{q}{|q|} = \frac{q_0}{|q|} + \frac{\mathbf{q}}{|q|},$$

is called the *versor* of the quaternion q. Also, if the quaternion is a unit quaternion to begin with, it can be simply called a *versor*.

Let us denote the unit quaternion as u and write it in the following form:

$$u = \cos\theta + \sin\theta\mathbf{u} = \cos\theta + u_1\sin\theta\mathrm{i} + u_2\sin\theta\mathrm{j} + u_3\sin\theta\mathrm{k}.$$

Any non-unit quaternion can be expressed through its module and versor by dividing both sides of the equation $|q|^2 = q_0^2 + \|q\|^2 = 1$ by $|q|^2$, we get:

$$1 = \frac{|q|^2}{|q|^2} = \frac{q_0^2}{|q|^2} + \frac{\|\mathbf{q}\|^2}{|q|^2} \Rightarrow q_0^2 = |q|^2\cos^2\theta,\ \ \|\mathbf{q}\|^2 = |q|^2\sin^2\theta.$$

Now we normalize the vector part of the quaternion $\mathbf{q}$ and write:

$$\frac{\mathbf{q}}{\|\mathbf{q}\|} = \mathbf{u} \quad \Rightarrow \mathbf{q} = \|\mathbf{q}\|\mathbf{u} = |q| \sin\theta\mathbf{u}.$$

Therefore, any quaternion can be expressed through its module $|q|$ and versor $\mathbf{u}$ as follows:

$$q = q_0 + \mathbf{q} = |q| \cos\theta + |q| \cos\theta\mathbf{u} = |q|(\cos\theta + \sin\theta\mathbf{u}) = |q|(\cos\theta + u_1 \sin\theta\mathrm{i} + u_2 \sin\theta\mathrm{j} + u_3 \sin\theta\mathrm{k}).$$

Using the formula (3) we compute the square of a unit quaternion:

$$u^2 = uu = (u_0^2 - \sin^2\theta\|\mathbf{u}\|) + 2u_0 \sin\theta\mathbf{u} = (\cos^2\theta - \sin^2\theta) + 2\cos\theta \sin\theta\mathbf{u} = \cos 2\theta + \sin 2\theta\mathbf{u}.$$

It turns out that to square a unit quaternion it is enough to double the parameter θ.

Let us compute the product of two unit quaternions $u_1 = \cos\theta_1 + \sin\theta_1\mathbf{u}_1$ and $u_2 = \cos\theta_2 + \sin\theta_2\mathbf{u}_2$

$$u_1u_2 = \big(\cos\theta_1 \cos\theta_2 - \sin\theta_1 \sin\theta_2(\mathbf{u}_1, \mathbf{u}_2)\big) + \cos\theta_1 \sin\theta_2\mathbf{u}_2 + \cos\theta_2 \sin\theta_1\mathbf{u}_1 + \mathbf{u}_1 \times \mathbf{u}_2 \sin\theta_1 \sin\theta_2.$$

If the unit quaternions differ only by the parameter θ and have the same unit vector part, that is $\mathbf{u}_1 = \mathbf{u}_2$, then the multiplication formula simplifies significantly:

$$u_1u_2 = (\cos\theta_1 \cos\theta_2 - \sin\theta_1 \sin\theta_2) + (\cos\theta_1 \sin\theta_2 + \cos\theta_2 \sin\theta_1)\mathbf{u} = \cos(\theta_1 + \theta_2) + \sin(\theta_1 + \theta_2)\mathbf{u}.$$

From this formula it follows that the unit quaternions $u_1 = \cos\theta_1 + \sin\theta_1\mathbf{u}$ and $u_2 = \cos\theta_2 + \sin\theta_2\mathbf{u}$ commute when multiplied. It also allows calculating an arbitrary power of a unit quaternion:

$$u^n = \cos n\theta + \sin n\theta\mathbf{u}.$$

And write an analogue of the formula of Moivre for an arbitrary quaternion with a versor u:

$$q^n = |q|^n(\cos n\theta + \sin n\theta\mathbf{u}).$$

Let us consider a unit pure quaternion $\mathbf{u} = u_1\mathrm{i} + u_2\mathrm{j} + u_3\mathrm{k}$ and compute its square:

$$\mathbf{u}^2 = \mathbf{uu} = -(\mathbf{u}, \mathbf{u}) + \mathbf{u} \times \mathbf{u} = -\|\mathbf{u}\|^2 = -1.$$

We obtained that a unit pure quaternion has a property that defines the elliptic imaginary unit.

Let us prove the following formulas for an arbitrary quaternion q, pure quaternions $\mathbf{u}$, $\mathbf{v}$ and a unit pure quaternion $\mathbf{n}$.

The formula $q(\mathbf{u} \times \mathbf{v})q^{-1} = (q\mathbf{u}q^{-1}) \times (q\mathbf{v}q^{-1})$, is valid due to the following chain of equalities:

- $q(\mathbf{u} \times \mathbf{v})q^{-1} = (q\mathbf{u}q^{-1}) \times (q\mathbf{v}q^{-1})$,
- $q(\mathbf{u}, \mathbf{v})q^{-1} = (q\mathbf{u}q^{-1}, q\mathbf{v}q^{-1})$,
- $\mathbf{n}(\mathbf{u} \times \mathbf{v})\mathbf{n} = -(\mathbf{nun}) \times (\mathbf{nvn})$,
- $(\mathbf{u} \times \mathbf{v})^* = -\mathbf{u} \times \mathbf{v}$.

The formula $q(\mathbf{u}, \mathbf{v})q^{-1} = (q\mathbf{u}q^{-1}, q\mathbf{v}q^{-1})$ is valid, because:

$$q(\mathbf{u} \times \mathbf{v})q^{-1} = \frac{1}{2}q(\mathbf{uv} - \mathbf{vu})q^{-1} = \frac{1}{2}(q\mathbf{uv}q^{-1} - q\mathbf{vu}q^{-1}) =$$
$$= \frac{1}{2}\left(q\mathbf{u}q^{-1}q\mathbf{v}q^{-1} - q\mathbf{v}q^{-1}q\mathbf{u}q^{-1}\right) = (q\mathbf{u}q^{-1}) \times (q\mathbf{v}q^{-1}).$$

Formula $q(\mathbf{u}, \mathbf{v})q^{-1} = (q\mathbf{u}q^{-1}, q\mathbf{v}q^{-1})$ is valid because:

$$q(\mathbf{u},\mathbf{v})q^{-1} = -\frac{1}{2}q(\mathbf{uv}+\mathbf{vu})q^{-1} = -\frac{1}{2}(q\mathbf{uv}q^{-1} + q\mathbf{vu}q^{-1}) =$$
$$= -\frac{1}{2}(q\mathbf{u}q^{-1}q\mathbf{v}q^{-1} + q\mathbf{v}q^{-1}q\mathbf{u}q^{-1}) = (q\mathbf{u}q^{-1}, q\mathbf{v}q^{-1}).$$

For a unit pure quaternion $\mathbf{n}$ the equality holds:

$$\mathbf{n}(\mathbf{u}\times\mathbf{v})\mathbf{n} = -(\mathbf{nun})\times(\mathbf{nvn}),$$

To prove this, we will use the fact that $\mathbf{nn} = -1$ and perform a series of transformations:

$$\mathbf{n}(\mathbf{u}\times\mathbf{v})\mathbf{n} = \mathbf{n}\frac{1}{2}(\mathbf{uv}-\mathbf{vu})\mathbf{n} = \frac{1}{2}(\mathbf{nuvn}-\mathbf{nvun}) = \frac{1}{2}(-\mathbf{nunnvn}+\mathbf{nvnnun}) =$$
$$= \frac{1}{2}(\mathbf{nvnnun}-\mathbf{nunnvn}) = (\mathbf{nvn})\times(\mathbf{nun}) = -(\mathbf{nun})\times(\mathbf{nvn}).$$

The formula $(\mathbf{u}\times\mathbf{v})^* = -\mathbf{u}\times\mathbf{v}$ can be derived using the fact that $\mathbf{u}^* = -\mathbf{u}$:

$$(\mathbf{u}\times\mathbf{v})^* = \frac{1}{2}(\mathbf{uv}-\mathbf{vu})^* = \frac{1}{2}\Big((\mathbf{uv})^* - (\mathbf{vu})^*\Big) = \frac{1}{2}(\mathbf{v}^*\mathbf{u}^* - \mathbf{u}^*\mathbf{v}^*) = -\frac{1}{2}(\mathbf{u}^*\mathbf{v}^* - \mathbf{v}^*\mathbf{u}^*) = -\mathbf{u}\times\mathbf{v}.$$

4. Dual quaternions

4.1. Definition of dual dual quaternions

Consider a dual number of the form:

$$Q = q + q^o\varepsilon,$$

where the coefficients q and q^o are quaternions. The quaternion $q = q_0 + \mathbf{q} = q_0 + q_1\mathrm{i} + q_2\mathrm{j} + q_3\mathrm{k}$ is called the *main part*, and the quaternion $q^o = q_0^o + \mathbf{q}^o = q_0^o + q_1^o\mathrm{i} + q_2^o\mathrm{j} + q_3^o\mathrm{k}$ is *moment part*. A hypercomplex number Q constructed in this way is called a *parabolic* or *dual* dual quaternion [21, p. 38, 22, p. 124, 16, p. 66, 20].

If both quaternions q and q^o are pure, that is, have zero scalar parts $q_0 = p_0^o = 0$, then the dual quaternion $\mathbf{Q} = \mathbf{q} + \mathbf{q}^o\varepsilon$ is also called a *pure dual quaternion* and coincides with the notion of a *motor* in screw theory [22, p. 84, 16, p. 169]. The terms *dyad*, bivector [16], dual vector, and line vector [6, p. 5] are also used. The term "bivector" is currently used in the literature to denote an object of a different type.

A dual quaternion Q can be written as a hypercomplex number with eight components:

$$Q = q_0 + q_1\mathrm{i} + q_2\mathrm{j} + q_3\mathrm{k} + q_0^o\varepsilon + q_1^o\mathrm{i}\varepsilon + q_2^o\mathrm{j}\varepsilon + q_3^o\mathrm{k}\varepsilon,$$

with eight basis elements $\langle 1, \mathrm{i}, \mathrm{j}, \mathrm{k}, \varepsilon, \mathrm{i}\varepsilon, \mathrm{j}\varepsilon, \mathrm{k}\varepsilon\rangle$.

To define the dual quaternion product, a multiplication table for the basis elements is required. To do this, in addition to the associativity axiom and the axiomatic relations $\mathrm{i}^2 = \mathrm{j}^2 = \mathrm{k}^2 = \mathrm{ijk} = -1$ and $\varepsilon^2 = 0$, it is necessary to define the commutativity of quaternion imaginary units with the dual unit ε, that is

$$\mathrm{i}\varepsilon = \varepsilon\mathrm{i},\ \mathrm{j}\varepsilon = \varepsilon\mathrm{j},\ \mathrm{k}\varepsilon = \varepsilon\mathrm{k}.$$

The full 8×8 multiplication table for the basis elements of a dual quaternion then has the form shown in table (5).

	1	i	j	k	ε	iε	jε	kε
1	1	i	j	k	ε	iε	jε	kε
i	i	-1	k	$-$j	iε	$-\varepsilon$	kε	$-$jε
j	j	$-$k	-1	i	jε	$-$kε	$-\varepsilon$	iε
k	k	j	$-$i	-1	kε	jε	$-$iε	$-\varepsilon$
ε	ε	iε	jε	kε	0	0	0	0
iε	iε	$-\varepsilon$	kε	$-$jε	0	0	0	0
jε	jε	$-$kε	$-\varepsilon$	iε	0	0	0	0
kε	kε	jε	$-$iε	$-\varepsilon$	0	0	0	0

(5)

Two additional remarks are in order. Firstly, assuming anticommutativity of imaginary units instead of commutativity leads to a contradiction. For example, if ε is presumed to anticommute with i and j, this entails commutativity between ε and k, since

$$\mathrm{i}\varepsilon = -\varepsilon\mathrm{i}, \mathrm{j}\varepsilon = -\varepsilon\mathrm{j} \Rightarrow \varepsilon\mathrm{k} = \varepsilon\mathrm{ij} = -\mathrm{i}\varepsilon\mathrm{j} = \mathrm{ij}\varepsilon = \mathrm{k}\varepsilon.$$

Secondly, it is possible not to introduce the requirement for commutativity between ε and i, j, k, However, in that case, it will be necessary to write the dual unit on the left in all the relations. For example, the dual quaternion Q will have to be defined exclusively as $q + \varepsilon q^o$ and permutations leading to expressions of the type $q^o\varepsilon$ must not be allowed anywhere. This rule is followed, for example, in [16], but it complicates the calculations and does not provide any particular advantage.

Another way to define dual quaternions involves applying the Cayley–Dickson doubling procedure [23–25], generalized to an arbitrary type of complex numbers.

The quaternion $q = q_0 + q_1\mathrm{i} + q_2\mathrm{j} + q_3\mathrm{k}$ undergoes transformation wherein the real coefficients q_0, q_1, q_2, q_3 are substituted with dual numbers Q_0, Q_1, Q_2, Q_3 through the application of the doubling procedure.

$$Q = Q_0 + Q_1\mathrm{i} + Q_2\mathrm{j} + Q_3 k = Q_0 + \mathbf{Q}, \quad Q_i = q_i + q_i^o\varepsilon, q_i, q_i^o \in \mathbb{R},\ i = 0, 1, 2, 3,$$

where Q_0 — *scalar* part (dual number), and $\mathbf{Q}$ — *screw* part. The screw part is also called a dual vector, a *screw* or *pure dual quaternion*. This representation will be called the *dual representation*. In dual representation, all dual quaternion formulas coincide in form with the corresponding quaternion relations, with real numbers replaced by dual numbers.

Terminological remarks. The term biquaternion covers three subtypes [20]:

- if the coefficients of the quaternion are elliptic complex numbers with imaginary unit $\mathrm{i}^2 = -1$, the biquaternion is called an *elliptic biquaternion*;
- In the case of parabolic coefficients (dual numbers with imaginary unit $\varepsilon^2 = 0,\ \varepsilon \neq 0$) the biquaternion is called a parabolic or *dual biquaternion*;
- For hyperbolic complex numbers with imaginary unit $h^2 = 1, h \neq 0$, the biquaternion is called a *hyperbolic biquaternion*.

A dual biquaternion is sometimes called a Clifford biquaternion, and a hyperbolic biquaternion is sometimes referred to as a Hamilton biquaternion [11]. In english-language sources, a dual biquaternion is more commonly called a *dual quaternion* [10, 20]. In english version of our paper we use the term *dual quaternion* and in russian version we use biquaternion as short name for parabolic biquaternion.

The term motor is a syllabic abbreviation formed from the words **mo**ment and vec*tor*.

4.2. Operations on dual quaternions

4.2.1. Addition

Introduce two dual quaternions P and Q and write them in quaternion and dual form

$$P = p + p^o\varepsilon = P_0 + P_1\mathrm{i} + P_2\mathrm{j} + P_3k = P_0 + \mathbf{P}, Q = q + q^o\varepsilon = Q_0 + Q_1\mathrm{i} + Q_2\mathrm{j} + Q_3k = Q_0 + \mathbf{Q}.$$

Addition (and subtraction) is defined by formula:

$$P \pm Q = p \pm q + (p^o \pm q^o)\varepsilon = P_0 \pm Q_0 + (P_1 \pm Q_1)\mathrm{i} + (P_2 \pm Q_2)\mathrm{j} + (P_3 \pm Q_3)\mathrm{k} = P_0 \pm Q_0 + \mathbf{P} \pm \mathbf{Q}.$$

The consequence of this formula is the associativity and commutativity of addition.:

$$P + Q = Q + P \text{ and } P + (Q + R) = (P + Q) + R,$$

where R is some third dual quaternion.

4.2.2. Multiplication by a number

The rule of multiplying a dual quaternion by a real number α is trivial and is determined by the ratio:

$$\alpha Q = \alpha q + \alpha q^o\varepsilon = \alpha Q_0 + \alpha\mathbf{Q}.$$

Multiplication by a dual number $A = \alpha + \alpha^o\varepsilon$ is somewhat more complicated:

$$(\alpha + \alpha^o\varepsilon)(q + q^o\varepsilon) = \alpha q + (\alpha q^o + \alpha^o q)\varepsilon.$$

Similarly, in the dual representation $Q = Q_0 + Q_1\mathrm{i} + Q_2\mathrm{j} + Q_3\mathrm{k}$, $Q_i = q_i + q_i^o\varepsilon$, $i = 0, 1, 2, 3$ yields:

$$AQ_i = (\alpha + \alpha^o\varepsilon)(q_i + q_i^o\varepsilon) = \alpha q_i + (\alpha q_i^o + \alpha^o q_i)\varepsilon.$$

4.2.3. Dual quaternion product

The *dual quaternion product* of P and Q is defined by

$$PQ = (p + p^o\varepsilon)(q + q^o\varepsilon) = pq + (pq^o + p^oq)\varepsilon,$$

where pq, pq^o, and p^oq are quaternion products. In dual representation the formula reproduces the quaternion product, in which all real numbers are replaced by dual components of dual quaternions:

$$PQ = \big(P_0Q_0 - (P_1Q_1 + P_2Q_2 + P_3Q_3)\big) + P_0(Q_1\mathrm{i} + Q_2\mathrm{j} + Q_3\mathrm{k}) + Q_0(P_1\mathrm{i} + P_2\mathrm{j} + P_3\mathrm{k}) + \\ + (P_2Q_3 - P_3Q_2)\mathrm{i} + (P_3Q_1 - P_1Q_3)\mathrm{j} + (P_1Q_2 - P_2Q_1)\mathrm{k}.$$

This expression is conveniently written in the more compact form:

$$PQ = P_0Q_0 - (\mathbf{P}, \mathbf{Q}) + P_0\mathbf{Q} + Q_0\mathbf{P} + \mathbf{P} \times \mathbf{Q},$$

where $(\mathbf{P}, \mathbf{Q}) = P_1Q_1 + P_2Q_2 + P_3Q_3$ is the scalar product, and

$$\mathbf{P} \times \mathbf{Q} = (P_2Q_3 - P_3Q_2)\mathrm{i} + (P_3Q_1 - P_1Q_3)\mathrm{j} + (P_1Q_2 - P_2Q_1)\mathrm{k}$$

is the screw product of pure dual quaternions $\mathbf{P}$ and $\mathbf{Q}$.

For pure dual quaternions:

$$\mathbf{PQ} = -(\mathbf{P}, \mathbf{Q}) + \mathbf{P} \times \mathbf{Q}, \mathbf{QP} = -(\mathbf{P}, \mathbf{Q}) - \mathbf{P} \times \mathbf{Q},$$

$$(\mathbf{P}, \mathbf{Q}) = -\frac{1}{2}\Big(\mathbf{PQ} + \mathbf{QP}\Big), \mathbf{P} \times \mathbf{Q} = \frac{1}{2}\Big(\mathbf{PQ} - \mathbf{QP}\Big).$$

4.2.4. Conjugation operations

Since both ordinary (elliptic) and dual complex numbers are present in the definition of a dual quaternion, three conjugation operations are introduced.

- $Q^* = (q + q^o\varepsilon)^* = q^* + q^{o*}\varepsilon$ is quaternion conjugation, which can also be called complex.
- $\overline{Q} = \overline{q + q^o\varepsilon} = q - q^o\varepsilon$ is dual conjugation.
- $Q^\dagger = (\overline{q + q^o\varepsilon})^* = q^* - q^{o*}\varepsilon$ is biquaternion conjugation.

A dual quaternion conjugation is a combination of two other conjugations. The following properties are valid for the introduced conjugation operations:

$$(PQ)^* = Q^*P^*, \overline{QP} = \overline{P}\overline{Q}, (PQ)^\dagger = Q^\dagger P^\dagger.$$

4.2.5. Scalar product

Scalar product of two arbitrary dual quaternions $P = p + p^o\varepsilon$ and $Q = q + q^o\varepsilon$ is determined by the following formula [20, p. 15]:

$$(P, Q) = \frac{1}{2}(PQ^* + QP^*),$$

where $*$ is quaternion conjugation

$$PQ^* = (p + p^o\varepsilon)(q^* + q^{o*}\varepsilon) = pq^* + [pq^{o*} + p^oq^*]\varepsilon,$$
$$QP^* = (q + q^o\varepsilon)(p^* + p^{o*}\varepsilon) = qp^* + [qp^{o*} + q^op^*]\varepsilon$$

hence:

$$(P, Q) = \frac{1}{2}(pq^* + qp^*) + \frac{1}{2}(pq^{o*} + q^op^*)\varepsilon + \frac{1}{2}(p^oq^* + qp^{o*})\varepsilon = (p, q) + [(p, q^o) + (p^o, q)]\varepsilon.$$

where $(p, q) = p_0q_0 + p_1q_1 + p_2q_2 + p_3q_3$, $(p^o, q) = p_0^oq_0 + p_1^oq_1 + p_2^oq_2 + p_3^oq_3$ and $(p, q^o) = p_0q_0^o + p_1q_1^o + p_2q_2^o + p_3q_3^o$ are quaternion scalar products [20, p. 15], which can be written as $(p, q) = p_0q_0 + (\mathbf{p}, \mathbf{q})$.

If dual quaternions are written in dual representation as $P = P_0 + \mathbf{P}$ and $Q = Q + \mathbf{Q}$, the formula can be written as follows:

$$PQ^* = (P_0 + \mathbf{P})(Q_0 - \mathbf{Q}) = P_0Q_0 - P_0\mathbf{Q} + Q_0\mathbf{P} + (\mathbf{P}, \mathbf{Q}) - \mathbf{P} \times \mathbf{Q},$$
$$QP^* = (Q_0 + \mathbf{Q})(P_0 - \mathbf{P}) = P_0Q_0 - Q_0\mathbf{P} + P_0\mathbf{Q} + (\mathbf{Q}, \mathbf{P}) - \mathbf{Q} \times \mathbf{Q},$$

$$(P,Q)=\frac{1}{2}(2P_0Q_0-P_0\mathbf{Q}+Q_0\mathbf{P}-Q_0\mathbf{P}+P_0\mathbf{Q}+2(\mathbf{P},\mathbf{Q})-\mathbf{P}\times\mathbf{Q}+\mathbf{P}\times\mathbf{Q})=P_0Q_0+(\mathbf{P},\mathbf{Q}),$$

which results in a general formula for the dual quaternion scalar product:

$$(P,Q)=(p,q)+[(p,q^o)+(p^o,q)]\varepsilon=P_0Q_0+(\mathbf{P},\mathbf{Q})=\sum_{i=0}^{3}P_iQ_i.$$

Some simple but important consequences can be immediately derived from the definition of the scalar product.

- The scalar product is a dual number.
- The scalar product of dual quaternions is *symmetric* $(P,Q)=(Q,P)$, since the scalar products of quaternions are symmetric and $(p,q)+[(p,q^o)+(p^o,q)]\varepsilon=(q,p)+[(q,p^o)+(q^o,p)]\varepsilon$.
- The scalar product is *bilinear*. So, for dual quaternions P, Q, R and dual numbers α, β the equalities are fulfilled:

$$(P,\alpha Q+\beta R)=\alpha(P,Q)+\beta(P,R),(\alpha P+\beta Q,R)=\alpha(P,R)+\beta(Q,R).$$

  Indeed:

$$(P,\alpha Q+\beta R)=\frac{1}{2}\big(P(\alpha Q+\beta R)^*+(\alpha Q+\beta R)P^*\big)=\frac{1}{2}(\alpha PQ^*+\beta PR^*+\alpha QP^*+\beta RP^*)=$$
$$=\frac{1}{2}\alpha(PQ^*+QP^*)+\frac{1}{2}\beta(PR^*+RP^*)=\alpha(P,Q)+\beta(P,R),$$

  and the second identity follows from symmetry.
- Written in quaternion form, the dual quaternion scalar product reproduces the dual number multiplication rule; written in dual form, it reproduces the quaternion scalar product formula.
- If $P=0+\mathbf{P}$ and $Q=0+\mathbf{Q}$ are pure dual quaternions, then:

$$(\mathbf{P},\mathbf{Q})=(\mathbf{p},\mathbf{q})+[(\mathbf{p},\mathbf{q}^o)+(\mathbf{p}^o,\mathbf{q})]\varepsilon,$$

  where $(\mathbf{p},\mathbf{q}^o)+(\mathbf{p}^o,\mathbf{q})=\mathrm{mom}(\mathbf{P},\mathbf{Q})$ is the mutual moment of $\mathbf{P}$ and $\mathbf{Q}$.
- If $P=p+0\varepsilon$, $Q=q+0\varepsilon$, then the scalar product of dual quaternions reduces to the scalar product of quaternions $(P,Q)=(p,q)$.
- If $P=p^o\varepsilon$ and $Q=q^o\varepsilon$, despite the fact that both dual quaternions are nonzero, their scalar product turns to zero: $(P,Q)=(0+\mathbf{0},0+\mathbf{0})+[(0+\mathbf{0},p^o)+(0+\mathbf{0},q^o)]\varepsilon=0$.

For calculations of the scalar product of dual quaternions written in quaternion form, calculations by hand are often easier "on paper" if the terms are slightly rearranged.

$$(P,Q)=(p_0+\mathbf{p}+p_0^o\varepsilon+\mathbf{p}^o\varepsilon,q_0+\mathbf{q}+q_0^o\varepsilon+\mathbf{q}^o\varepsilon)=p_0q_0+(\mathbf{p},\mathbf{q})+[p_0q_0^o+(\mathbf{p},\mathbf{q}^o)+p_0^oq_0+(\mathbf{p}^o,\mathbf{q})]\varepsilon,$$

which leads to:

$$(P,Q)=p_0q_0+[p_0q_0^o+p_0^oq_0]\varepsilon+(\mathbf{p},\mathbf{q})+[(\mathbf{p},\mathbf{q}^o)+(\mathbf{p}^o,\mathbf{q})]\varepsilon. \tag{6}$$

The concept of *orthogonality* of two dual quaternions can be introduced as follows: two dual quaternions $P=p+\varepsilon p^o$ and $Q=q+\varepsilon q^o$ are called orthogonal if $(p,q)=0$, which is equivalent to the condition of orthogonality of their main parts.

Consider a special case of basis quaternions, for example, $P=\mathrm{i}$ and $Q=\mathrm{j}$, then, according to formula (6), hence

$$(P,Q)=0\cdot0+[0\cdot0+0\cdot0]\varepsilon+(\mathrm{i},\mathrm{j})+[(\mathrm{i},\mathbf{0})+(\mathbf{0},\mathrm{j})]\varepsilon=(\mathrm{i},\mathrm{j})=0.$$

All members of the formula are specifically listed here to demonstrate how the symbols work. The result is the orthogonality of the basis elements i and j.

Now let $P = \mathrm{i}\varepsilon$ and $Q = \mathrm{j}$, then

$$(P, Q) = 0 \cdot 0 + [0 \cdot 0 + 0 \cdot 0]\varepsilon + (\mathbf{0}, \mathrm{j}) + [(\mathbf{0}, \mathbf{0}) + (\mathrm{i}\varepsilon, \mathrm{j})]\varepsilon = (\mathrm{i}, \mathrm{j})\varepsilon^2 = 0.$$

In the same way, it is possible to prove the mutual orthogonality of the basis elements $\langle 1, \mathrm{i}, \mathrm{j}, \mathrm{k}, \varepsilon, \mathrm{i}\varepsilon, \mathrm{j}\varepsilon, \mathrm{k}\varepsilon\rangle$.

4.2.6. Absolute value of a dual quaternion and angle between dual quaternions

The squared absolute value of a dual quaternion is defined by:

$$|Q|^2 = (Q, Q) = QQ^*,$$

hence the following

$$|Q|^2 = QQ^* = (q + q^o\varepsilon)(q^* + q^{o*}\varepsilon) = qq^* + (qq^{o*} + q^oq^*)\varepsilon = |q|^2 + 2(q, q^o)\varepsilon,$$

where (q, q^o) is the quaternion scalar product of q and q^o. It can be seen from the resulting expression that the square of the absolute value $|Q|^2$ is a dual number.

The *absolute value of a dual quaternion* Q is obtained by taking the square root of the dual number $|Q|^2$:

$$|Q| = \sqrt{QQ^*} = \sqrt{|q|^2 + 2(q, q^o)\varepsilon} = |q| + \frac{(q, q^o)}{|q|}\varepsilon = |q|\left(1 + \frac{(q, q^o)}{|q|^2}\varepsilon\right),$$

in accordance with formula (1) for the square root of a dual number.

The following consequences can be proved.

- $|PQ| = |P||Q|$ так как $(PQ, PQ) = (PQ)(PQ)^* = PQQ^*P^* = P|Q|^2P^* = |Q|^2PP^* = |Q|^2|P|^2$ hence, the desired equality is obtained.
- For any three dual quaternions P, Q, R the following equality holds

  $$(PQ, PR) = |P|^2(P, R),$$

  which follows from

$$(PQ, PR) = \frac{1}{2}\big(PQ(PR)^* + PR(PQ)^*\big) = \frac{1}{2}(PQR^*P^* + PRQ^*P^*) =$$
$$= P\frac{1}{2}(QR^* + RQ^*)P^* = P(Q, R)P^* = |P|^2(Q, R).$$

The scalar product of the main and moment parts (q, q^o) is called the *invariant of the dual quaternion* [16, p. 71], and the real number $k = \frac{(q,q^o)}{|q|^2}$ is called the *parameter of the dual quaternion*. The absolute value can then be written as $|Q| = |q|(1 + k\varepsilon)$.

For a pure dual quaternion (motor) $\mathbf{Q} = \mathbf{q} + \mathbf{q}^o\varepsilon$ the parameter $k = \frac{(\mathbf{q},\mathbf{q}^o)}{\|\mathbf{q}\|^2}$ is determined by the scalar product of the vector $\mathbf{q}$ with its moment $\mathbf{q}^o$. If $(\mathbf{q}, \mathbf{q}^o) = 0$, the motor is called a *screw*. More details of this type of dual quaternions will be discussed below.

Note that the expression for the dual quaternion modulus loses its meaning if the absolute value of the main part is equal to zero $|q| = 0$, since in this case the denominator of the fraction turns to zero. The existence of a singularity at a nonzero value of the square of the absolute value distinguishes a dual quaternion from an ordinary quaternion. The square of the module retains its meaning.

Using the scalar product, the cosine of the *dual angle* between two dual quaternions is defined by:

$$\cos \measuredangle(P, Q) = \cos \Theta = \frac{(P, Q)}{|P||Q|}.$$

Here $\cos \Theta$ is a dual number, and, hence, Θ is also a dual number. The geometric meaning of this angle is discussed later, and note that for pure dual quaternions, the formula takes the form:

$$\cos \Theta = \frac{(\mathbf{P}, \mathbf{Q})}{|\mathbf{P}||\mathbf{Q}|} = \frac{(\mathbf{p}, \mathbf{q}) + \mathrm{mom}(\mathbf{P}, \mathbf{Q})}{\mathbf{PQ}}.$$

Unlike QQ^*, the products $Q\overline{Q}$ and $QQ^\dagger$ are not dual numbers. Consider $QQ^\dagger$

$$QQ^\dagger = (q + q^o\varepsilon)(q^* + q^{o*}\varepsilon) = qq^* + (q^oq^* - qq^{o*})\varepsilon = |q|^2 + (q^oq^* - qq^{o*})\varepsilon,$$

$$q^oq^* - qq^{o*} = q_0^oq_0 - q_0^o\mathbf{q} + q_0\mathbf{q}^o - \mathbf{q}^o\mathbf{q} - q_0q_0^o + q^o\mathbf{q}^o - \mathbf{q}q_0^o + \mathbf{q}\mathbf{q}^o = 2(q_0\mathbf{q}^o - q_0^o\mathbf{q}) + 2\mathbf{q} \times \mathbf{q}^o,$$

$$QQ^\dagger = |q|^2 + [2(q_0\mathbf{q}^o - q_0^o\mathbf{q}) + 2\mathbf{q} \times \mathbf{q}^o]\varepsilon.$$

For $Q\overline{Q}$:

$$Q\overline{Q} = (q + q^o\varepsilon)(q - q^o\varepsilon) = q^2 + (q^oq - qq^o)\varepsilon = q^2 + 2\mathbf{q}^o \times \mathbf{q}\varepsilon = q_0^2 - (\mathbf{q}, \mathbf{q}) + 2q_0\mathbf{q} + 2\mathbf{q}^o \times \mathbf{q}\varepsilon.$$

4.2.7. Screw product

The *screw* product [22, p. 102] is defined only for pure dual quaternions and coincides with the vector product of vectors with dual components. For $\mathbf{P} = \mathbf{p} + \mathbf{p}^o\varepsilon$ and $\mathbf{Q} = \mathbf{q} + \mathbf{q}^o\varepsilon$ are two pure dual quaternions, then

$$\mathbf{P} \times \mathbf{Q} = \mathbf{p} \times \mathbf{q} + (\mathbf{p} \times \mathbf{q}^o + \mathbf{p}^o \times \mathbf{q})\varepsilon.$$

For dual representation, the screw product formula has already been written above, but it can be rewritten in a more compact form using a formal determinant:

$$\mathbf{P} \times \mathbf{Q} = \begin{vmatrix} \mathrm{i} & \mathrm{j} & \mathrm{k} \\ P_1 & P_2 & P_3 \\ Q_1 & Q_2 & Q_3 \end{vmatrix} = (P_2Q_3 - P_3Q_2)\mathrm{i} + (P_3Q_1 - P_1Q_3)\mathrm{j} + (P_1Q_2 - P_2Q_1)\mathrm{k}.$$

The operation $\times$ can be extended by setting $\alpha \times \mathbf{P} = \alpha\mathbf{P}$, for $\alpha \in \mathbb{R}$. This extension allows for more compact notation in some formulas.

4.2.8. Unit dual quaternion and trigonometric representation

A *Unit dual quaternion*, is a dual quaternion $U = u + u^o\varepsilon$ satisfying $(U, U) = 1$ or $|U| = 1$, which is equivalent to $|u|^2 + 2(u, u^o)\varepsilon = 1$, so $|u| = 1$ and $(u, u^o) = 0$. Therefore, a unit dual quaternion has zero parameter and zero invariant.

From $(PQ, PR) = |P|^2(P, R)$, multiplication of two dual quaternions by a unit dual quaternion preserves their scalar product: $(UQ, UP) = |U|^2(P, Q)$. Consequently, the dual angle Θ between dual quaternions is preserved.

Consider a unit dual quaternion U in dual representation:

$$U = U_0 + U_1\mathrm{i} + U_2\mathrm{j} + U_3\mathrm{k}, \quad U_i = u_i + u_i^o\varepsilon, \quad i = 0, 1, 2, 3, \quad |U|^2 = U_0^2 + \|\mathbf{U}\|^2 = 1.$$

By virtue of the fundamental trigonometric identity $\sin^2\Theta + \cos^2\Theta = 1$, which is also true for dual numbers, there is always a dual angle Θ such that

$$U_0^2 = \cos^2\Theta, \|\mathbf{U}\|^2 = \sin^2\Theta \Rightarrow U_0^2 + \|\mathbf{U}\|^2 = 1.$$

The screw part $\mathbf{U}$ of the dual quaternion U need not be a unit pure dual quaternion, but it can be normalized:

$$\mathbf{U} = \|\mathbf{U}\|\Big(\frac{U_1}{\|\mathbf{U}\|}\mathrm{i} + \frac{U_2}{\|\mathbf{U}\|}\mathrm{j} + \frac{U_3}{\|\mathbf{U}\|}\mathrm{k}\Big) = \|\mathbf{U}\|\mathbf{E} = \sin\Theta\mathbf{E},$$

где $\mathbf{E}$ is a unit pure dual quaternion (a unit screw).

Thus, any unit dual quaternion $U = U_0 + U_1\mathrm{i} + U_2\mathrm{j} + U_3\mathrm{k}$ can be written in *trigonometric form*:

$$U = \cos\Theta + \sin\Theta\mathbf{E}, \quad \mathbf{E} = E_1\mathrm{i} + E_2\mathrm{j} + E_3\mathrm{k}, \quad \|\mathbf{E}\| = 1,$$

where the dual angle Θ is also called the *dual argument* of the dual quaternion.

An arbitrary dual quaternion P can always be decomposed into the product of a modulus and a unit dual quaternion as follows:

$$P = P_0 + \mathbf{P} = |\mathrm{P}|\Big(\frac{P_0}{|\mathrm{P}|} + \frac{\mathbf{P}}{|\mathrm{P}|}\Big) = |\mathrm{P}|\Big(\frac{P_0}{|\mathrm{P}|} + \frac{\|\mathbf{P}\|}{|\mathrm{P}|}\mathbf{E}\Big) = |\mathrm{P}|(\cos\Theta + \sin\Theta\mathbf{E}),$$

where $\cos^2\Theta = (P_0/|\mathrm{P}|)^2$ and $\sin^2\Theta = (\|\mathbf{P}\|/|\mathrm{P}|)^2$, and the unit dual quaternion $\cos\Theta + \sin\Theta\mathbf{E}$ is sometimes called the *versor* of the dual quaternion P, and the modulus $|\mathrm{P}|$ is called the tensor (this term will not be used).

4.2.9. Dualization Operation

Introduce an algebraic operation closely related to the principle of duality in projective geometry. This connection will become apparent later; here only algebraic properties are considered.

Define the *dualization operation* $+$ acting on the basis elements by

$$\begin{aligned} 1^+ &= \varepsilon, \varepsilon^+ = 1, \\ \mathrm{i}^+ &= \mathrm{i}\varepsilon, (\mathrm{i}\varepsilon)^+ = \mathrm{i}, \\ \mathrm{j}^+ &= \mathrm{j}\varepsilon, (\mathrm{j}\varepsilon)^+ = \mathrm{j}, \\ \mathrm{k}^+ &= \mathrm{k}\varepsilon, (\mathrm{k}\varepsilon)^+ = \mathrm{k}. \end{aligned}$$

Additionally, linearity is required: $(P + Q)^+ = P^+ + Q^+$. Then:

$$P^+ = (p + p^o\varepsilon)^+ = p^o + p\varepsilon.$$

The scalar product of a dualized dual quaternion with itself is given by:

$$(P^+, P^+) = |p^o|^2 + 2(p, p^o)\varepsilon.$$

5. Dual quaternion representation of points, lines and planes

We consider the three-dimensional projective space $\mathrm{P}\mathbb{R}^3$ modeled by the four-dimensional Cartesian space $\mathbb{R}^4$ (see article [26]). In this space, each point can be defined by homogeneous coordinates $(x, y, z \mid w)$, also written as $x : y : z : w$. The fourth coordinate w is called the weight coordinate or *weight*. All points are divided into two classes:

proper points are finite points for which $w \neq 0$, corresponding to affine points P with coordinates $(x/w, y/w, z/w)$;

improper points are points at infinity for which $w = 0$, corresponding to free vectors specifying direction $\mathbf{v} = (v_x, v_y, v_z)$.

The German mathematician A. Möbius (Möbius, August Ferdinand, 1790–1868) introduced a mechanical interpretation of homogeneous coordinates by associating the fourth coordinate not with spatial dimension but with the mass of a point, i.e., an additional attribute inherent to each point in space. Such a point is called a *mass point* [27]. Further, for brevity, we will refer to a point with non-unit w coordinate as a *mass point*, and a point with unit w coordinate as an *affine point*.

5.1. Quaternion representation of points

To each mass point P_w with coordinates $(x, y, z \mid w)$ we assign a dual quaternion of the following form:

$$P = p + p^o\varepsilon = w + \mathbf{p}^o\varepsilon, \quad p = w + \mathbf{0}, \quad p^o = \mathbf{p}^o = x\mathrm{i} + y\mathrm{j} + z\mathrm{k}.$$

The scalar part of the quaternion p consists only of the scalar unit, and the vector part p^o is pure and corresponds to the radius vector of the point $\mathbf{p}^o = (x, y, z)^T$. Here we write the radius vector with the symbol o to emphasize the dual quaternion representation. This symbol in this context does not carry any additional geometric meaning.

To each mass point P_w corresponds an affine point with coordinates $P = (x/w, y/w, z/w)$ or in dual quaternion representation:

$$P = 1 + \frac{1}{w}\mathbf{p}^o\varepsilon = 1 + \frac{x}{w}\mathrm{i}\varepsilon + \frac{y}{w}\mathrm{j}\varepsilon + \frac{z}{w}\mathrm{k}\varepsilon.$$

If $w = 1$, then we get $P = 1 + \mathbf{p}^o\varepsilon$.

Free vectors or improper points, which specify direction in space, can also be represented in dual quaternion form as

$$\mathbf{V} = \mathbf{v}\varepsilon = v_x\mathrm{i}\varepsilon + v_y\mathrm{j}\varepsilon + v_z\mathrm{k}\varepsilon.$$

This is a specific pure dual quaternion $\mathbf{V}$ with zero vector part but non-zero moment part.

If given two points P and Q, then the free vector $\mathbf{PQ}$, which specifies the direction from point P to point Q, can be calculated using dual quaternions as follows:

$$\mathbf{PQ} = Q - P = 1 + \mathbf{q}^o\varepsilon - 1 - \mathbf{p}^o\varepsilon = (\mathbf{q}^o - \mathbf{p}^o)\varepsilon.$$

The notations of quaternions well agree with the notations of points and vectors, so from now on "point P" should be interpreted as "dual quaternion P associated with an affine point".

Note that the dual quaternion $O = w + \mathbf{0} + (0 + \mathbf{0})\varepsilon = w$ is associated with the point of the origin $(0, 0, 0 \mid 1)$, so for any point P we can write $P - O = 1 + \mathbf{p}^o\varepsilon - 1 = \mathbf{p}^o\varepsilon$ and geometrically interpret the result as a free vector $\mathbf{p}^o$. This way of denoting free vectors is accepted in the book [20], however, in our opinion this notation is too cumbersome and we will introduce a more compact notation. We will say that to each point $P = 1 + \mathbf{p}^o\varepsilon$ corresponds a pure dual quaternion $\mathbf{P} = \mathbf{p}^o\varepsilon$ which specifies a free vector, which, when plotted from the origin, will set the point P. The last sentence algebraically expressed in the notation $P = O + \mathbf{P} = 1 + \mathbf{P}$.

If given a mass point $P_w = w + \mathbf{p}^o\varepsilon$, then it will also correspond to a free vector $\mathbf{P} = \frac{1}{w}\mathbf{p}^o\varepsilon$, which, when added to the dual quaternion $O = 1$, will again give an affine point $P = 1 + \frac{1}{w}\mathbf{p}^o\varepsilon$, which can then be converted into a mass point by multiplying by w:

$$wP = P_w = w + \frac{w}{w}\mathbf{p}^o\varepsilon = w + \mathbf{p}^o\varepsilon.$$

Any dual quaternion P_w, representing a mass point, when multiplied by a real number m will represent the same affine point as before multiplication, because

$$mP_w = mw + m\mathbf{p}^o\varepsilon \leftrightarrow P = wm/wm + m\mathbf{p}^o\varepsilon/mw = 1 + \mathbf{p}^o\varepsilon/w.$$

From this it follows that the dual quaternion representation of an affine point is homogeneous, which is consistent with homogeneous coordinates $x : y : z : w$.

5.2. Operations over points

By associating an algebraic object with a point, we will explain the geometric meaning of some dual quaternion operations. In most cases we will give the geometric interpretation to an affine point, even if the calculations were performed over a mass point.

If given an affine point $P = 1 + \mathbf{p}^o\varepsilon$, then the conjugation operations have the following meaning:

- $P^* = 1 - \mathbf{p}^o\varepsilon$ is central symmetry relative to the origin;
- $\bar{P} = P^* = 1 - \mathbf{p}^o\varepsilon$ is also central symmetry;
- $P^\dagger = P$ is identity transformation.

The dual quaternions $P_w = w_1 + \mathbf{p}^o\varepsilon$ and $Q_w = w_2 + \mathbf{p}^o\varepsilon$ commute under dual quaternion multiplication:

$$P_wQ_w = (w_1 + \mathbf{p}^o\varepsilon)(w_2 + \mathbf{p}^o\varepsilon) = w_1w_2 + (w_1\mathbf{q}^o + w_2\mathbf{p}^o)\varepsilon,$$
$$Q_wP_w = (w_2 + \mathbf{p}^o\varepsilon)(w_1 + \mathbf{p}^o\varepsilon) = w_2w_1 + (w_2\mathbf{p}^o + w_1\mathbf{q}^o)\varepsilon.$$

Therefore $P_wQ_w = Q_wP_w = w_1w_2 + (w_2\mathbf{p}^o + w_1\mathbf{q}^o)\varepsilon$ and $PQ = 1 + (\mathbf{p}^o/w_1 + \mathbf{q}^o/w_2)\varepsilon$. The result of multiplication gave an affine point in Cartesian space, the radius vector of which is the sum of the radius vectors of the two multiplied points.

If we perform the addition of dual quaternions $P_w + Q_w = w_1 + w_2 + (\mathbf{p}^o + \mathbf{q}^o)\varepsilon$, then the geometric meaning is only for the sum of affine points $(P + Q)_w = 2 + (\mathbf{p}^o + \mathbf{q}^o)\varepsilon$, therefore $P + Q = 1 + \frac{(\mathbf{p}^o+\mathbf{q}^o)}{2}\varepsilon$ is the midpoint of the segment PQ.

Let's find the scalar product of two mass points P_w and Q_w

$$(P_w, Q_w) = (w_1+\mathbf{0}+(0+\mathbf{p}^o)\varepsilon, w_2+\mathbf{0}+(0+\mathbf{q}^o)\varepsilon) = w_1w_2+[0\cdot0+0\cdot0]\varepsilon+(\mathbf{0},\mathbf{0})+[(\mathbf{0},\mathbf{q}^o)+(\mathbf{0},\mathbf{p}^o)]\varepsilon = w_1w_2,$$

From this it follows that the square of the module of a mass point and an affine point:

$$|P_w|^2 = w^2, \quad |P|^2 = 1.$$

Note that the module of a dual quaternion gives the value of the weight coordinate, not the length of the radius vector. However, using the dualization operation, we can obtain the radius vector of a point:

$$|P_w|^2 = |\mathbf{p}^o|^2 + 2(1 + \mathbf{0}, 0 + \mathbf{p}^o)\varepsilon = |\mathbf{p}^o|^2 = \|\mathbf{p}^o\|^2,$$

And also the norm of a free vector $\mathbf{V} = \mathbf{v}\varepsilon$ as $|\mathbf{V}^+| = \|\mathbf{v}\|$.

5.3. Dual quaternion representation of planes

To uniquely define a plane as in three-dimensional Cartesian space and in three-dimensional projective space, it is sufficient to define a linear equation of the following form:

$$n_xx + n_yy + n_zz + d = 0 \Leftrightarrow (\mathbf{n}, \mathbf{p}) + d = 0,$$

Where $\mathbf{p} = (x, y, z)^T$ is the radius vector of an arbitrary point of the plane, $\mathbf{n} = (n_x, n_y, n_z)$ is the direction vector of the normal to the plane, not necessarily unit, and d is a parameter whose geometric meaning is revealed by the relation $d = \delta\|\mathbf{n}\|$, where δ is the directed distance from the plane to the origin.

The unit normal vector will be denoted as $\hat{\mathbf{n}}$. With its help, the equation of the plane can be written in the simplest form: $(\hat{\mathbf{n}}, \mathbf{p}) + \delta = 0$. However, we also note that any vector $\mathbf{n} = k\hat{\mathbf{n}}$ and parameter $d = \delta\|\mathbf{n}\| = k\delta$ will define the same plane as the pair $[\hat{\mathbf{n}} \mid \delta]$.

The plane given by the pair $[\mathbf{n} \mid d]$ has the following dual quaternion representation:

$$\Pi = n + n^o\varepsilon = \mathbf{n} + d\varepsilon, \text{ that is } n = 0 + \mathbf{n}, \quad n^o = d + \mathbf{0},$$

In this case, in accordance with the reasoning of the previous paragraph, the dual quaternion $k\Pi$, where $k \in \mathbb{R}$ corresponds to the same plane as the dual quaternion Π. In other words, the dual quaternion representation is homogeneous and for any dual quaternion we can perform normalization:

$$\widehat{\Pi} = \frac{1}{\|\mathbf{n}\|}\Pi = \frac{\mathbf{n}}{\|\mathbf{n}\|} + \frac{d}{\|\mathbf{n}\|}\varepsilon = \hat{\mathbf{n}} + \delta\varepsilon.$$

Let's calculate the module for the plane dual quaternion Π:

$$|\Pi|^2 = \Pi\Pi^* = (\mathbf{n} + d\varepsilon)(-\mathbf{n} + d\varepsilon) = -\mathbf{n}\mathbf{n} + d\mathbf{n}\varepsilon - d\mathbf{n}\varepsilon = d^2\varepsilon^2 = -(-(\mathbf{n}, \mathbf{n}) + \mathbf{n} \times \mathbf{n}) = \|\mathbf{n}\|^2 \Rightarrow |\Pi| = \|\mathbf{n}\|.$$

The module $|\Pi|$ allows us to obtain the length of the normal vector of the plane, so the normalization process can be written as:

$$\widehat{\Pi} = \frac{\Pi}{|\Pi|} = \hat{\mathbf{n}} + \delta\varepsilon.$$

From this it follows that the dual quaternion Π is a unit dual quaternion.

Note that unlike the module of a point, the module of a plane dual quaternion equals the norm of a vector, not a scalar parameter. As in the case of a point, applying the dualization operation and calculating:

$$|\Pi^+|^2 = |d + \mathbf{n}\varepsilon|^2 = d^2 \Rightarrow |\Pi^+| = d.$$

Now the module of the dual quaternion equals the scalar parameter. This is easily explained by the fact that when applying the dualization operation Π^+ we get the dual quaternion $d + \mathbf{n}\varepsilon$, which exactly corresponds to the dual quaternion representation of a point with a weight coordinate d and radius vector $\mathbf{n}$. This is an algebraic expression of the fundamental principle of duality of projective geometry.

Let's prove a simple but important statement. The point $P = 1 + \mathbf{p}^o\varepsilon$ belongs to the plane $\Pi = \mathbf{n} + d\varepsilon$ if and only if the equality:

$$(P, \Pi) = 0. \tag{7}$$

To prove this, we use the formula (6) for calculating the scalar product:

$$(P, \Pi) = 1 \cdot 0 + [1 \cdot d + 0 \cdot 0]\varepsilon + (\mathbf{0}, \mathbf{n}) + [(\mathbf{0}, \mathbf{0}) + (\mathbf{p}^o, \mathbf{n})]\varepsilon = (d + (\mathbf{p}^o, \mathbf{n}))\varepsilon,$$

From this it follows that the condition $(P, \Pi) = 0$ is equivalent to the condition $d + (\mathbf{p}^o, \mathbf{n}) = 0$, i.e., the point P satisfies the equation of the plane and therefore belongs to this plane.

Let's also prove that if a point $P = 1 + \mathbf{p}^o\varepsilon$ and a vector $\mathbf{n}$ are given, then the plane with the normal vector $\mathbf{n}$, passing through the point P, is given by the following equality:

$$\Pi = \mathbf{n} - (P, \mathbf{n}).$$

The scalar product of two dual quaternions is understood as follows:

$$(P,\mathbf{n}) = \big(1+\mathbf{0}+(0+\mathbf{p}^o)\varepsilon, 0+\mathbf{n}+(0+\mathbf{0})\varepsilon\big) = 1\cdot 0 + \big[(\mathbf{p}^o,\mathbf{n}) + (1+\mathbf{0},0+\mathbf{0})\big]\varepsilon = (\mathbf{p}^o,\mathbf{n})\varepsilon.$$

Let's also calculate the scalar product $(P,(P,\mathbf{n}))$:

$$\big(P,(P,\mathbf{n})\big) = \big(P,(\mathbf{p}^o,\mathbf{n})\varepsilon\big) = (\mathbf{p}^o,\mathbf{n})(P,\varepsilon) = (\mathbf{p}^o,\mathbf{n})\big((1+\mathbf{0},0+\mathbf{0}) + [(\mathbf{p}^o,\mathbf{0}) + (1+\mathbf{0},1+\mathbf{0})]\big)\varepsilon = (\mathbf{p}^o,\mathbf{n})\varepsilon.$$

Now we subtract $(P,\mathbf{n}) - \big(P,(P,\mathbf{n})\big) = \big(P,\mathbf{n}-(P,\mathbf{n})\big) = (\mathbf{p}^o,\mathbf{n})\varepsilon - (\mathbf{p}^o,\mathbf{n})\varepsilon = 0$ from where, according to condition (7), it follows that the expression $\mathbf{n}-(P,\mathbf{n})$ must be a dual quaternion representing a plane, because the point P belongs to the plane by the condition of the problem.

The condition (7) also allows us to prove that if two points P and Q belong to a plane, then the free vector $\mathbf{PQ}$ also belongs to it. To prove this, we need to calculate

$$(\mathbf{PQ},\Pi) = (Q-P,\Pi) = (Q,\Pi) - (P,\Pi) = 0.$$

We also show that the projection of the origin O onto the plane Π is given by the point $O' = 1+\delta\hat{\mathbf{n}}\varepsilon$. This is obvious from the above reasoning, but we will prove it using condition (7). We write:

$$(O',\Pi) = \big(1+\mathbf{0}+(0+\delta\hat{\mathbf{n}})\varepsilon, 0+\mathbf{0}+(d+\mathbf{0})\varepsilon\big) = \\ (1+\mathbf{0},0+\mathbf{n}) + \big[(1+\mathbf{0},d+\mathbf{0}) + (0+\delta\hat{\mathbf{n}},0+\mathbf{n})\big]\varepsilon = 0 + [d+\delta(\mathbf{n},\hat{\mathbf{n}})]\varepsilon.$$

Now the condition $(O',\Pi)=0$ is equivalent to $d+\delta(\mathbf{n},\hat{\mathbf{n}})=0$ or $d=\delta\|\mathbf{n}\|(\hat{\mathbf{n}},\hat{\mathbf{n}}) = \delta\|\mathbf{n}\|$, but this is the geometric meaning of the parameter d, so the point O' indeed belongs to the plane.

Let's prove another formula. There are 8 possible mutual arrangements of three planes:

$$\Pi_i = \mathbf{n}_i - (\mathbf{p}_i^o,\mathbf{n})\varepsilon, \qquad i=1,2,3.$$

One of these arrangements is the intersection of the planes in a single point. The dual quaternion corresponding to this point can be calculated by the formula [20, p. 23, 28, p. 56]:

$$P = 1 + \frac{(\mathbf{n}_1,P_1)\mathbf{n}_2\times\mathbf{n}_3 + (\mathbf{n}_2,P_2)\mathbf{n}_3\times\mathbf{n}_1 + (\mathbf{n}_3,P_3)\mathbf{n}_1\times\mathbf{n}_2}{(\mathbf{n}_1,\mathbf{n}_2,\mathbf{n}_3)}.$$

To prove this, we need to use the same condition (7) of belonging of a point to a plane by sequentially multiplying P by Π_1, Π_2, Π_3 and verifying the condition.

A line, or more precisely an axis (a line with a specified direction on it) is represented by a pure dual quaternion of the following form:

The coordinates of the Plücker line, which are the six components $\{\mathbf{v}\mid\mathbf{m}\} = \{v_x,v_y,v_z\mid m_x,m_y,m_z\}$, were introduced by the German physicist and mathematician Julius Plücker in a way that is completely unrelated to dual quaternions. The condition $(\mathbf{v},\mathbf{m})=0$ was also obtained by Plücker and is known as the *Plücker condition*.

A straight line, or rather an axis (a straight line with the direction indicated on it) is given by a pure dual quaternion of the following form:

$$\mathbf{L} = \mathbf{v} + \mathbf{m}\varepsilon,$$

with the mandatory condition $(\mathbf{v},\mathbf{m})=0$. The vector $\mathbf{v} = v_x\mathrm{i} + v_y\mathrm{j} + v_z\mathrm{k}$ is the guiding vector of the straight line, the vector $\mathbf{m} = m_x\mathrm{i} + m_y\mathrm{j} + m_z\mathrm{k}$ is called the *moment* of the straight line.

Using the dualization operation allows us to rewrite many formulas in dual quaternion notation. For example, let's say we have a dual quaternion point $P = 1+\mathbf{p}\varepsilon = 1+\mathbf{P}$ and a pure dual quaternion representing a free vector $\mathbf{V} = \mathbf{v}\varepsilon$. We define the moment of the line passing through the point P in the direction $\mathbf{V}$ by the following formula:

This formula completely coincides with the vector definition of the moment. This notation allows us to define the dual quaternion form of a line as the sum of

$$\mathbf{M} = \mathbf{P} \times \mathbf{V}^+ = (0 + \mathbf{0}) \times (0 + \mathbf{v}) + [(0 + \mathbf{0}) \times (0 + \mathbf{0}) + (0 + \mathbf{p}) \times (0 + \mathbf{v})]\varepsilon = \mathbf{p} \times \mathbf{v}\varepsilon = \mathbf{m}\varepsilon.$$

The last form of writing will be valid if we additionally put $\alpha \times \mathbf{V}^+ = \alpha\mathbf{V}^+$, where $\alpha \in \mathbb{R}$

$$\mathbf{L} = \mathbf{V}^+ + \mathbf{M} = \mathbf{V}^+ + \mathbf{P} \times \mathbf{V}^+ = (1 + \mathbf{P}) \times \mathbf{V}^+ = P \times \mathbf{V}.$$

The points of a straight line can be calculated using a parametric representation of a straight line in dual quaternion form:

$$P(t) = P_0 + \mathbf{v}t\varepsilon, \quad P_0 = 1 + \frac{\mathbf{v} \times \mathbf{m}}{\|\mathbf{v}\|^2}\varepsilon.$$

The nearest point to the origin on the line is calculated by the formula:

$$\mathbf{p}_0 = \frac{\mathbf{v} \times \mathbf{m}}{\|\mathbf{v}\|^2}.$$

The dual object to a line is another line: $\mathbf{L}^+ = (\mathbf{v} + \mathbf{m}\varepsilon)^+ = \mathbf{m} + \mathbf{v}\varepsilon = \mathbf{M}^+ + \mathbf{V}$, which is consistent with the principle of duality of projective geometry.

Since a line is represented by a pure dual quaternion, we will say not about the module, but about the norm of a pure dual quaternion. We calculate the norm of the line $\mathbf{L} = \mathbf{v} + \mathbf{m}\varepsilon$ having in mind the Plücker condition $(\mathbf{v}, \mathbf{m}) = 0$.

$$\mathbf{L}\mathbf{L}^* = -(\mathbf{v}+\mathbf{m}\varepsilon)(\mathbf{v}+\mathbf{m}\varepsilon) = -(\mathbf{v}\mathbf{v}+\mathbf{v}\mathbf{m}\varepsilon+\mathbf{m}\mathbf{v}\varepsilon) = -\Big(-(\mathbf{v},\mathbf{v})+\mathbf{v}\times\mathbf{v}-\big((\mathbf{v},\mathbf{m})+\mathbf{v}\times\mathbf{m}-(\mathbf{m},\mathbf{v})+\mathbf{m}\times\mathbf{v}\big)\varepsilon\Big) =$$
$$= -(-\|\mathbf{v}\|^2 - 2(\mathbf{v},\mathbf{m})\varepsilon) = \|\mathbf{v}\|^2 \quad \Rightarrow \quad \mathbf{L}\mathbf{L}^* = \|\mathbf{L}\|^2 = \|\mathbf{v}\|^2.$$

The normalization process for a line reduces to dividing the vectors $\mathbf{v}$ and $\mathbf{m}$ by the norm $\|\mathbf{v}\|$.

5.4. Calculation of distances

For dual quaternions representing points $P_w = w + \mathbf{p}\varepsilon$, lines $\mathbf{L} = \mathbf{v} + \varepsilon\mathbf{m}$ and planes $\Pi = \mathbf{n} + d\varepsilon$ we have found formulas for their scalar products:

$$(P_w, P_w) = w^2, \quad (L, L) = \|\mathbf{v}\|^2, \quad (\Pi, \Pi) = \|\mathbf{n}\|^2.$$

Now let's find the scalar products of the dual objects

$$(P_w^+, P_w^+) = \|\mathbf{p}\|^2, \quad (L^+, L^+) = \|\mathbf{m}\|^2, \quad (\Pi^+, \Pi^+) = d^2.$$

It turns out that the formulas for calculating the distance δ from the origin O to the point P_w, the line $\mathbf{L}$ and the plane Π will have the same form for each of these objects:

$$\delta(O, P_w) = \frac{|P_w^+|}{|P_w|}, \quad \delta(O, \mathbf{L}) = \frac{\|\mathbf{L}^+\|}{\|\mathbf{L}\|}, \quad \delta(O, \Pi) = \frac{|\Pi^+|}{|\Pi|}.$$

6. Results

The main results of the work are summarized in several tables.

– In the table 3, the notation of points, vectors, lines, and planes from different geometries is compared: Euclidean, projective, and dual quaternion representations of projective geometry.
– Table 4 is the main result of our work. The table from the book [29] was taken as a basis (a shorter version of it is also in the book [30]), which has been completely rewritten in the dual quaternion formalism.
– The table 5 contains formulas for normalization of dual quaternions representing points, vectors, lines and planes.
– The table 6 contains a summary of terms that actually refer to the same geometric entity.

As far as the authors can tell, most of the formulas from the table 4 are original results and are not found in publications, although their vector and geometric algebra variants are known.

6.1. Tables

Table 3

Comparison of algebraic representations of points, lines and planes

Geometric object	**Biquaternion representation**	**Homogeneous coordinates**	**Three-dimensional Cartesian space**
Affine point	$P = 1 + \mathbf{p}\varepsilon$,		
$\mathbf{p} = x\mathrm{i} + y\mathrm{j} + z\mathrm{k}$	$\vec{\mathbf{p}} = (\mathbf{p} \mid 1) = (x, y, z \mid 1)$	$\mathbf{p} = (x, y, z)^T$	
Point mass	$P = w + \mathbf{p}\varepsilon$	$\vec{\mathbf{p}} = (\mathbf{p} \mid w) = (x, y, z \mid w)$	$\mathbf{p} = (x/w, y/w, z/w)$
Vector	$\mathbf{V} = \mathbf{v}\varepsilon$,		
$\mathbf{v} = v_x\mathrm{i} + v_y\mathrm{j} + v_z\mathrm{k}$	$\vec{\mathbf{v}} = (\mathbf{v} \mid 0) = (v_x, v_y, v_z \mid 0)$	$\mathbf{v} = (v_x, v_y, v_z)^T$	
Line	$\mathbf{L} = \mathbf{v} + \mathbf{m}\varepsilon$	$\vec{\mathbf{L}} = \{\mathbf{v} \mid \mathbf{m}\}$ $\vec{\mathbf{p}} = (\mathbf{v} \times \mathbf{m} \mid \|\mathbf{v}\|^2)$	$\mathbf{p}(t) = \mathbf{p}_0 + \mathbf{v}t$
Plane	$\Pi = \mathbf{n} + d\varepsilon$, $\mathbf{n} = n_x\mathrm{i} + n_y\mathrm{j} + n_z\mathrm{k}$	$\vec{} = [\mathbf{n} \mid d]$	$ax + by + cz + d = 0$

Table 4

Summary of biquaternion formulas for points, lines, and planes

	Formula	**Description**
A	$(Q_w - P_w)^+ + \mathbf{P} \times \mathbf{Q}^+$	Line through two point masses P_w and Q_w
B	$(Q - P)^+ + \mathbf{P} \times \mathbf{Q}^+$	Line through two affine points P and Q
C	$\mathbf{V}^+ + \mathbf{P} \times \mathbf{V}^+$	Line defined by free vector $\mathbf{V}$ and affine point P
D	$(P - O)^+ + \mathbf{0}\varepsilon$	Line through the origin $O = 1$ and point P

Continued on next page

Table 4

Summary of biquaternion formulas for points, lines, and planes (Continued)

	Formula	Description
E	$(\mathbf{V}^+ \times \mathbf{P} - w\mathbf{M})^+ - (\mathbf{L}^+, P_w)$	Plane containing line $\mathbf{L}$ and point mass P_w
F	$(\mathbf{V}^+ \times \mathbf{P})^+ - (\mathbf{L}^+, P)$	Plane containing line $\mathbf{L}$ and affine point P
G	$(\mathbf{V}^+ \times \mathbf{U})^+ - (\mathbf{L}^+, \mathbf{U})$	Plane containing line $\mathbf{L}$ and free vector $\mathbf{U}$
H	M^+	Plane containing a line and the origin
I	$\Pi_1 \times \Pi_2$	Line of intersection of two planes
J	$\mathbf{L}^+ \times \Pi^+ - d\mathbf{M}^+ - (\Pi^+, \mathbf{L})$	Point mass of the intersection of line and plane
K	$P \times \Pi - (P, \Pi^+)^+$	Line through the point mass P_w perpendicular to line L
L	$(\mathbf{V} \times \Pi)^+ - (\Pi^+, \mathbf{L}^+)$	Plane containing line L, and perpendicular to plane Π
M	$(\mathbf{V} \times \mathbf{P})^+ - (\mathbf{L}, P)$	Plane containing point mass P_w, and perpendicular to line L
N	$(\mathbf{L}, \mathbf{L}) + (\mathbf{L} \times \mathbf{L}^+)^+$	Point mass on line L, the closest to the origin
O	$(\Pi, \Pi^+) - \Pi \times \Pi^+$	Point mass on plane Π, the closest to the origin
P	$(\mathbf{L}^+, \mathbf{L}^+)^+ + \mathbf{L}^+ \times \mathbf{L}$	The farthest plane from the origin containing line $\mathbf{L}$
Q	$(P, P^+) - P \times P^+$	The farthest plane from the origin containing point P
R	$\frac{\lvert P_w \rvert Q_w - \lvert Q_w \rvert P_w}{(P_w, Q_w)}$	Distance between point masses
S	$\frac{\lvert(\mathbf{L}_1, \mathbf{L}_2)^+\rvert}{\lVert \mathbf{L}_1 \times \mathbf{L}_2 \rVert}$	Distance between lines
T	$\frac{\lVert (P_w \times \mathbf{L})^+ \rVert}{\lvert P_w \rvert \lVert \mathbf{L} \rVert}$	Distance from line to point mass P_w
U	$\frac{\lVert \mathbf{L}^+ \rVert}{\lVert \mathbf{L} \rVert}$	Distance from line to the origin
V	$\frac{\lvert(\Pi, P_w)^+\rvert}{\lvert P_w \rvert \lvert \Pi \rvert}$	Distance from plane to point mass P_w
W	$\frac{\lvert \Pi^+ \rvert}{\lvert \Pi \rvert}$	Distance from plane to the origin
X	$\frac{\lvert P_w^+ \rvert}{\lvert P_w \rvert}$	Distance from point mass to the origin

Remark: Point mass $P_w = w + \mathbf{p}\varepsilon$, affine point $P = 1 + \mathbf{p}\varepsilon$, free vector $\mathbf{V} = \mathbf{v}\varepsilon$, line $\mathbf{L} = \mathbf{v} + \mathbf{m}\varepsilon = \mathbf{V}^+ + \mathbf{M}$, plane $\Pi = \mathbf{n} + d\varepsilon$.

Table 5

Normalization of points, lines, and planes

	General form	Normalized form
Point mass	$P_w = w + \mathbf{p}\varepsilon$	$\hat{P} = \frac{P_w}{\lvert P_w \rvert} = 1 + \frac{1}{w}\mathbf{p}\varepsilon$
Line	$\mathbf{L} = \mathbf{v} + \mathbf{m}\varepsilon$	$\hat{\mathbf{L}} = \frac{\mathbf{L}}{\lVert \mathbf{L} \rVert} = \frac{\mathbf{v}}{\lVert \mathbf{v} \rVert} + \frac{\mathbf{m}}{\lVert \mathbf{v} \rVert}\varepsilon$
Plane	$\Pi = \mathbf{n} + d\varepsilon$	$\hat{\Pi} = \frac{\Pi}{\lvert \Pi \rvert} = \frac{\mathbf{n}}{\lVert \mathbf{n} \rVert} + \frac{d}{\lVert \mathbf{n} \rVert}\varepsilon$

Table 6

Correlation of terms of different geometries of space

Affine point	Vector	Point mass	Pure Biquaternion
- finite point - proper point - position vector - bound vector - vector point - quaternion with $q_0 = 1$	- point at infinity - improper point - free vector - direction vector - pure quaternion	- finite point with $w \neq 1$ - quaternion with $q_0 \neq 1$	- screw with zero parameter - dual vector - dyad - null system

7. Conclusion

We have consistently outlined the basics of the algebra of dual numbers, quaternions, and dual quaternions. We have considered the dual quaternion representation of points, lines and planes. The issues of using dual quaternions to describe helical motion were beyond the scope of consideration. We plan to consider these issues in the future. The topic of using dual quaternions to describe helical motion is voluminous, since almost all publications on the topic of dual quaternions are devoted to it. This is because this area is the main, if not the only, application area for dual quaternions.

Author Contributions: Methodology, writing—original draft preparation, Migran N. Gevorkyan, Nikita A. Vishnevskiy, Kirill V. Didus; writing—review and editing, Anna V. Korolkova; conceptualization, Dmitry S. Kulyabov.All authors have read and agreed to the published version of the manuscript.

Funding: This paper has been financially supported within the framework of the RUDN University Development Program "Strategy-2030."

Data Availability Statement: Data sharing is not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Declaration on Generative AI: The authors have not employed any Generative AI tools.

References

1. Blaschke, W. J. E. *Anwendung dualer Quaternionen auf Kinematik* German (Suomalainen tiedeakatemia, Helsinki, 1958).

2. Blaschke, W. J. E. *Kinematics and Quaternions* trans. from the German by Delphenich, D. H. Berlin, 1960. doi:10.1002/zamm.19620420724.
3. Kotelnikov, A. P. *The Screw Calculus and Some of Its Applications to Geometry and Mechanics* 222 pp. (Annals of the Imperial University of Kazan, Kazan, 1895).
4. Farias, J. G., De Pieri, E. & Martins, D. A Review on the Applications of Dual Quaternions. *Machines* **12.** doi:10.3390/machines12060402 (2024).
5. Dimentberg, F. M. *The Screw Calculus and Its Applications in Mechanics* 162 pp. (Foreign Technology Division, Springfield, 1969).
6. Fischer, I. *Dual-Number Methods in Kinematics, Statics and Dynamics* 240 pp. (CRC Press, 1998).
7. Huang, Z., Li, Q. & Ding, H. *Basics of Screw Theory* in *Theory of Parallel Mechanisms* (Springer Netherlands, Dordrecht, 2013). doi:10.1007/978-94-007-4201-7_1.
8. Featherstone, R. A Beginner's Guide to 6-D Vectors (Part 1). *IEEE Robotics and Automation Magazine* **17,** 83–94. doi:10.1109/MRA.2010.937853 (2010).
9. Featherstone, R. A Beginner's Guide to 6-D Vectors (Part 2) [Tutorial]. *IEEE Robotics and Automation Magazine* **17,** 88–99. doi:10.1109/MRA.2010.939560 (2010).
10. Kenwright, B. *A Survey on Dual-Quaternions* 2023. arXiv: 2303.14765 [math.OC].
11. Thomas, F. Approaching Dual Quaternions From Matrix Algebra. *IEEE Transactions on Robotics* **30,** 1–12. doi:10.1109/TRO.2014.2341312 (Aug. 2014).
12. Bruno, V. A. *Robot Kinematic Modeling and Control Based on Dual Quaternion Algebra. Part I: Fundamentals* working paper or preprint. https://hal.science/hal-01478225.
13. Wang, X., Han, D., Yu, C. & Zheng, Z. The geometric structure of unit dual quaternion with application in kinematic control. *Journal of Mathematical Analysis and Applications* **389,** 1352–1364. doi:10.1016/j.jmaa.2012.01.016 (2012).
14. Bekar, M. & Yayli, Y. Kinematics of Dual Quaternion Involution Matrices. *SDU Journal of Science* **11,** 121–132 (2016).
15. Dantam, N. T. *Practical Exponential Coordinates using Implicit Dual Quaternions* (Workshop on the Algorithmic Foundations of Robotics, 2018).
16. Chelnokov, Y. N. *Quaternionic and biquaternionic models and methods of solid mechanics and their applications. Geometry and kinematics of motion* 512 pp. (FIZMATLIT, Moscow, 2006).
17. Dunn, F. & Parberry, I. *3D Math Primer for Graphics and Game Development* 2nd ed. 846 pp. (CRC Press, 2011).
18. Kuipers, J. B. *Quaternions and rotation sequences* a primer with applications to orbits, aerospace and virtual reality. 371 pp. (Princeton University Press, 41 William Street, Princeton, New Jersey 08540, 1999).
19. Goldman, R. *Rethinking Quaternions. Theory and Computation* doi:10 . 2200 / S00292ED1V01Y201008CGR013 (Morgan & Claypool, 2010).
20. Goldman, R. *Dual Quaternions and Their Associated Clifford Algebras* 279 pp. (CRC Press Taylor & Francis Group, Boca Raton, London, New York, 2024).
21. Kotelnikov, A. P. *The Screw Calculus and Some of Its Applications to Geometry and Mechanics* 222 pp. (Annals of the Imperial University of Kazan, Kazan, 1895).
22. Dimentberg, F. M. *The Screw Calculus and Its Applications in Mechanics* 162 pp. (Foreign Technology Division, Springfield, 1969).
23. Kantor, I. L. & Solodownikow, A. S. *Hyperkomplexe Zahlen* German. 144 pp. (Teubner Verlagsgesellschaft, Leipzig, 1978).
24. Dickson, L. E. On Quaternions and Their Generalization and the History of the Eight Square Theorem. *Annals of Mathematics* **20,** 155–171 (1919).
25. Dickson, L. E. Algebras and their arithmetics. *Bulletin of the American Mathematical Society* **30,** 247–257 (1924).

26. Gevorkyan, M. N., Korolkova, A. V., Kulyabov, D. S. & Sevastyanov, L. A. Analytic projective geometry for computer graphics. *Discrete and Continuous Models and Applied Computational Science* **33,** 74–102 (2025).
27. Goldman, R. *An integrated introduction to computer graphics and geometric modeling* 592 pp. (CRC Press Taylor & Francis Group, Boca Raton, London, New York, 2009).
28. Faux, I. D. & Pratt, M. J. *Computational Geometry for Design and Manufacture* 304 pp. (New York, 1979).
29. Lengyel, E. *Projective Geometric Algebra Illuminated* 294 pp. (Terathon Software LLC, 2024).
30. Lengyel, E. *Foundations of Game Engine Development.* 1: *Mathematics* 4 vols. 195 pp. (Terathon Software LLC, Lincoln, California, 2016).

Information about the authors

Gevorkyan, Migran N.—Docent, Ph.D. in Physics and Mathematics, Associate Professor of Department of Probability Theory and Cyber Security of RUDN University (e-mail: gevorkyan-mn@rudn.ru, ORCID: 0000-0002-4834-4895, ResearcherID: E-9214-2016, Scopus Author ID: 57190004380)

Vishnevskiy, Nikita A.—PhD student of Department of Probability Theory and Cyber Security of RUDN University (e-mail: 1142240277@rudn.ru, ORCID: 0009-0004-4410-8635)

Didus, Kirill V.—PhD student of Department of Probability Theory and Cyber Security of RUDN University (e-mail: 1142240434@rudn.ru, ORCID: 0000-0002-5622-8480)

Korolkova, Anna V.—Docent, Ph.D. in Physics and Mathematics, Associate Professor of Department of Probability Theory and Cyber Security of RUDN University (e-mail: korolkova-av@rudn.ru, ORCID: 0000-0001-7141-7610, ResearcherID: I-3191-2013, Scopus Author ID: 36968057600)

Kulyabov, Dmitry S.—Professor, Doctor of Sciences in Physics and Mathematics, Professor. of the Department of Probability Theory and Cyber Security of RUDN University; Senior Researcher of Laboratory of Information Technologies, Joint Institute for Nuclear Research (e-mail: kulyabov-ds@rudn.ru, ORCID: 0000-0002-0877-7063, ResearcherID: I-3183-2013, Scopus Author ID: 35194130800)

УДК 511.84,512.523.282.2,519.711
DOI: 10.22363/2658-4670-2025-33-4-411-439 EDN: HPZRYA

Бикватернионное представление точек, прямых и плоскостей

М. Н. Геворкян[1], Н. А. Вишневский[1], К. В. Дидусь[1], А. В. Королькова[1], Д. С. Кулябов[1, 2]

[1] Российский университет дружбы народов им. Патриса Лумумбы, ул. Миклухо-Маклая, д. 6, Москва, 117198, Российская Федерация

[2] Объединённый институт ядерных исследований, ул. Жолио-Кюри, д. 6, Дубна, 141980, Российская Федерация

Аннотация. *Предпосылки* Основная масса работ по бикватернионам, посвящена их применению для описания винтового движения. Представлению с их помощью точек, прямых и плоскостей (примитивов) уделяется мало внимания. *Цель* Необходимо последовательно изложить бикватернионную теорию представления примитивов и доработать математический формализм. *Методы* Используется алгебра дуальных чисел, кватернионов и бикватернионов, а также элементы теории винтов и скользящих векторов. *Результаты* Получены и систематизированы формулы которые использую исключительно бикватернионные операции и обозначения для решения стандартных задач трёхмерной геометрии. *Выводы* Бикватернионы могут служить полноценным формализмом алгебраического представления трёхмерного проективного пространства.

Ключевые слова: дуальные числа, кватернионы, дуальные кватернионы, проективное пространство

Discrete & Continuous Models & Applied Computational Science
ISSN 2658-7149 (Online), 2658-4670 (Print)

2025, 33 (4) 440–460
http://journals.rudn.ru/miph

Physics and Astronomy

Research article

UDC 519.6, 530.145.67
PACS 03.67.Ac, 03.67.Lx,
DOI: 10.22363/2658-4670-2025-33-4-440-460 EDN: HYWEXV

Simulating QAOA operation using Cirq and qsim quantum frameworks

Yuri G. Palii, Alla A. Bogolubskaya, Denis A. Yanovich

Joint Institute for Nuclear Research, 6 Joliot-Curie St, Dubna, 141980, Russian Federation

(received: August 12, 2025; revised: September 2, 2025; accepted: September 10, 2025)

Abstract. The problem of finding the lowest-energy state in the Ising model with a longitudinal magnetic field is studied for two- and three-dimensional lattices of various sizes using the Quantum Approximate Optimization Algorithm (QAOA). The basis states of the quantum computer register correspond to spin configurations on a spatial lattice, and the Hamiltonian of the model is implemented using a sequence of quantum gates. The average energy value is efficiently measured using the Hadamard test. We simulate the QAOA operation on increasingly complex lattice configurations using the software libraries `Cirq` and `qsim`. The results of optimization, obtained using gradient-based and gradient-free methods, demonstrate the superiority of the latter in both modeling performance and quantum computer usage. Key arguments in favor of the advantages of quantum computation for this problem are presented.

Key words and phrases: quantum computing, QAOA, Ising model, quantum simulation, optimization, Cirq, qsim, cuStateVec

For citation: Palii, Y. G., Bogolubskaya, A. A., Yanovich, D. A. Simulating QAOA operation using Cirq and qsim quantum frameworks. *Discrete and Continuous Models and Applied Computational Science* **33** (4), 440–460. doi: 10.22363/2658-4670-2025-33-4-440-460. edn: HYWEXV (2025).

1. Introduction

The phenomenon of quantum entanglement (that is, the existence of special correlations between the states of particles in the microcosm that are not described by classical physics) makes the simulation of a system of hundreds of such particles intractable for supercomputers without significant simplifications. At the same time, for a quantum computer that uses quantum bits (qubits), entanglement serves as a resource for algorithms that are much more efficient than classical ones. An important example of this is quantum field theory on a space-time lattice, where classical computations struggle due to issues such as the Monte Carlo sign problem [1]. There are already quantum algorithms offering polynomial or even exponential speedups compared to their classical counterparts. It is widely believed that quantum computers will lead to breakthroughs in high-energy physics [2], quantum chemistry, and other fields facing computationally challenging problems [3, 4].

Due to the imperfections of quantum computers, the focus today is on hybrid quantum-classical algorithms that implement variational procedures. A quantum computer is used to construct a wave function (ansatz) depending on variational parameters and to measure observables for the modeled system. A classical computer then finds a new set of parameters for quantum gates by exploiting some optimization procedure in order to achieve the desired value of the cost function, which is one of the observables [5]. The theoretical foundation of such algorithms is the quantum mechanical Rayleigh–Ritz variational principle, which states that, for any trial wave function $|\psi(\boldsymbol{\alpha})\rangle$ with parameters $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$, the average value of the Hamiltonian is not less than the ground-state energy: $\langle\psi(\boldsymbol{\alpha})|\,\mathcal{H}\,|\psi(\boldsymbol{\alpha})\rangle \geq E_0$.

Among hybrid algorithms, one of the most promising is considered to be the Quantum Approximate Optimization Algorithm (QAOA) [6], which is the focus in this paper. It has the property of universality due to its adaptability to a specific task, since its Hamiltonian is employed to prepare the variational wave function ansatz. Moreover, since QAOA is a problem-specific ansatz, it avoids the *barren plateau problem* (a common challenge in parameter optimization) [4]. There are arguments in favor of the algorithm's ability to demonstrate the quantum advantage.

The problem of finding the ground state energy and the corresponding state vector is crucial for the theory of phase transitions and critical phenomena in quantum systems. The Ising model, in its various formulations, along with other spin systems models, serves as a starting point for numerous applications in statistical physics, lattice field theory, as well as quantum technologies (e.g., material science and device engineering). Notably, finding the ground state for the Ising model constitutes an NP-hard computational problem. The QAOA algorithm has been widely used to study ground states in models of magnetic materials [7], with analytical approaches developed for improved parameter optimization [8].

This paper provides a concise introduction to the application of the QAOA for the aforementioned problem. Drawing upon resources from Google–Quantum AI's documentation of the `Cirq` quantum computing library [9], we present numerical simulations of spin lattice models in 2D and 3D, with the latter being non-integrable. Our results demonstrate increasing accuracy of the energy estimation when we enlarge the number of the ansatz layers. Furthermore, we implement the Hadamard test to enhance the efficiency of the average energy measurements.

2. QAOA Ansatz for the Ising Model

2.1. Ising Model on a Quantum Computer

Each lattice site corresponds to a qubit in the quantum computer's register. An arbitrary distribution of spin values across the lattice sites can be represented as a binary string $z = z_1 z_2 \ldots z_n$, where each variable z_i specifies the spin orientation at the i-th site and takes values $z_i = \pm 1$. The value of z_i corresponds to the measurement outcome of the Pauli operator $Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$ acting on the i-th qubit in the computational basis:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \leftrightarrow z_i = +1, \qquad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \leftrightarrow z_i = -1.$$

In this way the 2^n basis states of a quantum register are in one-to-one correspondence with the 2^n possible spin configurations on the lattice:

$$\text{bit string } z = z_1 z_2 \ldots z_n \;\longleftrightarrow\; \text{quantum register } |z\rangle = |z_1 z_2 \ldots z_n\rangle.$$

An arbitrary state $|\psi\rangle$ of the quantum register represents a superposition of all possible spin configurations on the lattice with different amplitudes. The Ising model Hamiltonian is constructed from multi-qubit operators $Z^{(i)}$,

$$Z^{(i)} = \mathbb{I} \otimes \cdots \otimes \underset{\substack{\uparrow \\ i\text{-th position}}}{Z} \otimes \cdots \otimes \mathbb{I},$$

where the Pauli-Z operator acts only on the i-th qubit. For nearest-neighbor spin-spin interactions with the coupling constant J and interaction with an external magnetic field h, the Hamiltonian takes the form:

$$\mathcal{H}(Z) = -J \sum_{\langle i,j \rangle} Z^{(i)} Z^{(j)} - h \sum_i Z^{(i)}, \tag{1}$$

where $\langle i, j \rangle$ denotes the set of nearest-neighbor spin pairs, and the second term sums over all lattice sites. The expectation value of the Hamiltonian serves as the cost function in the variational minimization procedure.

2.2. Driver and Mixer Operators in the Ansatz

The variational ansatz $|\psi(\boldsymbol{\alpha})\rangle$ for the QAOA implementation ([5], Chapter 9.3) consists of multiple identical layers of quantum operations. Each layer contains two operators: a *driver* and a *mixer*, which act sequentially on the quantum register.

First, a uniform superposition state is prepared from the state $|0\rangle^{\otimes n}$ by applying Hadamard gates H to each of the n qubits in the register:

$$\left|\psi_0^{(n)}\right\rangle = \mathsf{H}^{\otimes n} |0\rangle^{\otimes n} = \frac{1}{2^{n/2}} \sum_{z \in \{0,1\}^n} |z\rangle, \quad \mathsf{H} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{2}$$

As a result, all possible spin configurations have equal probability amplitudes.

• The driver operator represents the evolution operator corresponding to the Hamiltonian $\mathcal{H}(Z)$ (with $J = 1$):

$$U(\gamma, \mathcal{H}) = e^{i\pi\gamma\mathcal{H}/2} = \prod_{\langle j,k \rangle} e^{-i\pi\gamma Z^{(j)} Z^{(k)}/2} \prod_l e^{-i\pi\gamma h Z^{(l)}/2}, \quad i = \sqrt{-1}, \tag{3}$$

where the variational parameter γ acts as an effective evolution time.

• The mixer operator is constructed from Pauli-X operators:

$$U(\beta, B) = e^{i\pi\beta B/2} = \prod_{j=1}^{n} e^{i\pi\beta X^{(j)}/2}, \quad \text{where } B = \sum_{j=1}^{n} X^{(j)}. \tag{4}$$

Here β serves as the second variational parameter in the ansatz layer. Thus, the p-layer QAOA ansatz looks as follows:

$$|\psi_p(\boldsymbol{\gamma}, \boldsymbol{\beta})\rangle = \underbrace{U(\beta_p, B) U(\gamma_p, \mathcal{H})}_{p} \cdots \underbrace{U(\beta_1, B) U(\gamma_1, \mathcal{H})}_{1} |\psi_0^{(n)}\rangle. \tag{5}$$

From the theorem proved in [6], as the number of layers p increases, the minimal expectation value $E_p(\boldsymbol{\gamma}, \boldsymbol{\beta})$ of the Hamiltonian $\mathcal{H}$ found by the QAOA converges to the minimal value $\min_z \mathcal{H}(z)$ among all possible bit strings z:

$$\lim_{p \to \infty} \min_{\boldsymbol{\gamma}, \boldsymbol{\beta}} E_p(\boldsymbol{\gamma}, \boldsymbol{\beta}) = \min_z \mathcal{H}(z), \quad E_p(\boldsymbol{\gamma}, \boldsymbol{\beta}) \equiv \left\langle \psi_p(\boldsymbol{\gamma}, \boldsymbol{\beta}) \right| \mathcal{H} \left| \psi_p(\boldsymbol{\gamma}, \boldsymbol{\beta}) \right\rangle. \tag{6}$$

In practice, the number of layers p is finite, making the algorithm approximate.

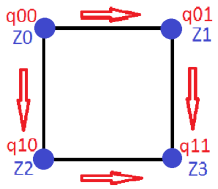


Figure 1. 2D lattice 2 × 2

3. Energy Calculation Methods

3.1. Direct State Vector Computation

This section presents the simplest approach for simulating QAOA on a quantum computer simulator. Using the quantum register's state vector $|\psi_p(\gamma,\beta)\rangle$ provided by the simulator and the diagonal elements of the the Hamiltonian (1), the per-site average energy (6) reduces to a sum:

$$E_p(\gamma,\beta) = \frac{1}{n}\sum_{i=0}^{2^n-1} \overline{\psi_p(\gamma,\beta)}_i \psi_p(\gamma,\beta)_i \mathcal{H}_{ii}, \tag{7}$$

where n is the number of lattice sites and the bar denotes complex conjugation.

Consider a 2×2 spin lattice (Fig. 1). Red arrows indicate pairwise interactions in the Hamiltonian (1). All possible spin configurations are encoded by a 4-qubit register ($n = 4$). An arbitrary normalized final state vector of a register in the computational basis is given by the amplitudes $\alpha_i(\gamma,\beta)$, $i = 0, \dots, 15$:

$$|\psi_1(\gamma,\beta)\rangle = \alpha_0(\gamma,\beta)|0000\rangle + \alpha_1(\gamma,\beta)|0001\rangle + \cdots + \alpha_{15}(\gamma,\beta)|1111\rangle, \tag{8}$$

where $\sum_i \bar{\alpha}_i(\gamma,\beta)\alpha_i(\gamma,\beta) = 1$. For $J = 1$ and $h = 1/2$ in (1), the diagonal elements of the Hamiltonian are:

$$[-1.5, -0.25, -0.25, 0, -0.25, 0, 1, 0.25, -0.25, 1, 0, 0.25, 0, 0.25, 0.25, -0.5].$$

The ground state $|0000\rangle$ (all spins aligned with the field) has energy -1.5. This means that during parameter variation, we aim to find the following values $\gamma_{\min}, \beta_{\min}$ that satisfy the following conditions:

$$\bar{\alpha}_0(\gamma_{\min},\beta_{\min})\alpha_0(\gamma_{\min},\beta_{\min}) = 1, \qquad \alpha_i(\gamma_{\min},\beta_{\min}) = 0,\ i \neq 0.$$

The quantum circuit implementing the single-layer QAOA ansatz (5) (Fig. 2) uses four qubits:

$$q_{(0,0)},\ q_{(0,1)},\ q_{(1,0)},\ q_{(1,1)}$$

which represent spins at lattice sites and form the quantum register.

The horizontal lines represent qubits, and the symbols on them show quantum gates acting sequentially from left to right. In the first (leftmost) column, Hadamard gates H prepare an equal superposition state (2). Subsequent columns contain gates that implement the driver operator $U(\gamma,\mathcal{H})$ and the mixer operator $U(\beta,B)$. The spin-spin interaction part of the driver operator $U(\gamma,\mathcal{H})$ is implemented by ZZ gates (vertical connections in the circuit diagram, labeled ZZ at the top and ZZ^() at the bottom).

For two qubits, the ZZ gate (Fig. 3) corresponds to a rotation operator with variational parameter γ (rotation angle)[1]:

[1]In Cirq, this gate's matrix differs by a factor of $i = \sqrt{-1}$.

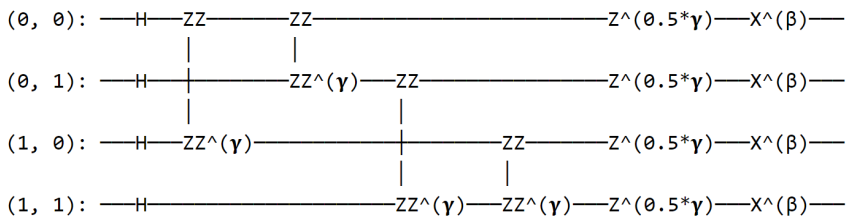


Figure 2. Circuit diagram of the $p = 1$ QAOA ansatz (5) for a 2×2 Ising lattice ($J = 1$, $h = 0.5$), constructed using Cirq [9]

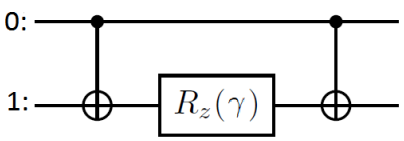


Figure 3. Quantum circuit for the ZZ gate (9)

$$R_{ZZ}(\pi\gamma) \equiv \exp\left(-i\pi\gamma\, Z \otimes Z/2\right) = \begin{bmatrix} e^{-i\pi\gamma/2} & 0 & 0 & 0 \\ 0 & e^{i\pi\gamma/2} & 0 & 0 \\ 0 & 0 & e^{i\pi\gamma/2} & 0 \\ 0 & 0 & 0 & e^{-i\pi\gamma/2} \end{bmatrix}, \tag{9}$$

which can be constructed using CNOT and $R_Z(\gamma)$ gates:

$$\text{CNOT} \cdot (\mathbb{I} \otimes R_Z(\gamma)) \cdot \text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} e^{-i\pi\gamma/2} & 0 & 0 & 0 \\ 0 & e^{i\pi\gamma/2} & 0 & 0 \\ 0 & 0 & e^{-i\pi\gamma/2} & 0 \\ 0 & 0 & 0 & e^{i\pi\gamma/2} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

An important property of the ZZ gate is its ability to create entangled qubit states [10], which gives quantum computations a significant advantage over classical approaches for studying quantum systems. The wave function of an entangled system cannot be factorized into a product of subsystem wave functions and represents a coherent superposition of combined subsystem states.

In the case of bipartite quantum systems — specifically two-qubit systems — the degree of entanglement can be quantified by the *concurrence* measure. As an illustration, consider the entangled state $|\psi_{ZZ}(\gamma)\rangle$ prepared by applying the ZZ interaction to the equal superposition state (Eq. 2) for $n = 2$ qubits:

$$|\psi_{ZZ}(\gamma)\rangle = R_{ZZ}(\pi\gamma)\, |\psi_0^{(2)}\rangle = \frac{1}{2}\Big(e^{-i\pi\gamma/2}|00\rangle + e^{i\pi\gamma/2}|01\rangle + e^{i\pi\gamma/2}|10\rangle + e^{-i\pi\gamma/2}|11\rangle\Big).$$

The density matrix of the pure state $|\psi_{ZZ}(\gamma)\rangle$ is the following one

$$\rho(\gamma) = |\psi_{ZZ}(\gamma)\rangle\langle\psi_{ZZ}(\gamma)| = \frac{1}{4}\begin{pmatrix} e^{-i\pi\gamma} & 1 & 1 & e^{-i\pi\gamma} \\ 1 & e^{i\pi\gamma} & e^{i\pi\gamma} & 1 \\ 1 & e^{i\pi\gamma} & e^{i\pi\gamma} & 1 \\ e^{-i\pi\gamma} & 1 & 1 & e^{-i\pi\gamma} \end{pmatrix},$$

By taking the partial trace over the second qubit (subsystem B of the two-qubit system), we obtain the reduced density matrix for the first qubit (subsystem A), which depends on the parameter γ:

$$\rho_A(\gamma) = \mathrm{Tr}_B\left(\rho(\gamma)\right) = \frac{1}{2}\begin{pmatrix} \cos\pi\gamma & 1 \\ 1 & \cos\pi\gamma \end{pmatrix}.$$

The concurrence [10] is then calculated as:

$$\mathcal{C}(|\psi_{ZZ}(\gamma)\rangle) = \sqrt{2\left(1 - \mathrm{Tr}\left(\rho_A^2(\gamma)\right)\right)} = \sin(\pi\gamma).$$

As evident, the *concurrence* $\mathcal{C}(|\psi_{ZZ}(\gamma)\rangle)$ is non-zero, indicating that the ZZ gate entangles qubit states for $\gamma \notin \mathbb{Z}$. Maximal entanglement occurs at $\gamma = 1/2$, where the operator assumes the form:

$$R_{ZZ}(1/2) = \frac{1}{\sqrt{2}}\left(\mathbb{I}_4 - iZ \otimes Z\right),$$

and the two-qubit state vector $|\psi_{ZZ}(\gamma = 1/2)\rangle$ decomposes into the Bell states:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \quad \text{and} \quad |\Psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle),$$

$$|\psi_{ZZ}(1/2)\rangle = \frac{1}{2}\left[(1-i)|\Phi^+\rangle + (1+i)|\Psi^+\rangle\right].$$

The ZZ gate implementing the interaction between spins at $(0,0)$ and $(1,0)$ in Fig. 1 (qubits $q_{(0,0)}$ and $q_{(1,0)}$) acts via the Pauli operators $Z^{(0)}$ and $Z^{(2)}$:

$$\exp\left(-i\pi\gamma Z^{(0)}Z^{(2)}/2\right), \qquad \begin{cases} Z^{(0)} = Z \otimes \mathbb{I}^{\otimes 3}, \\ Z^{(2)} = \mathbb{I}^{\otimes 2} \otimes Z \otimes \mathbb{I}, \end{cases}$$

and is applied right after the initial Hadamard gates. The spin-spin interaction part of the driver operator $U(\gamma, \mathcal{H})$ consists of four analogous factors,

$$U_{ZZ}(\gamma, \mathcal{H}) = e^{-i\pi\gamma Z^{(2)}\cdot Z^{(3)}/2} e^{-i\pi\gamma Z^{(1)}\cdot Z^{(3)}/2} e^{-i\pi\gamma Z^{(0)}\cdot Z^{(1)}/2} e^{-i\pi\gamma Z^{(0)}\cdot Z^{(2)}/2},$$

and is the diagonal 16×16 matrix with elements:

$$\mathrm{diag}\left[e^{-2\pi i\gamma}, 1, 1, 1, 1, 1, e^{2\pi i\gamma}, 1, 1, e^{2\pi i\gamma}, 1, 1, 1, 1, 1, e^{-2\pi i\gamma}\right].$$

The magnetic field interaction in the driver operator (3) is implemented via Z-axis rotations acting on each qubit:

$$R_z(\pi\gamma h) = e^{-i\pi\gamma h Z/2} = \cos\left(\frac{\pi\gamma h}{2}\right)\mathbb{I} - i\sin\left(\frac{\pi\gamma h}{2}\right)Z = \begin{bmatrix} e^{-i\pi\gamma h/2} & 0 \\ 0 & e^{i\pi\gamma h/2} \end{bmatrix},$$

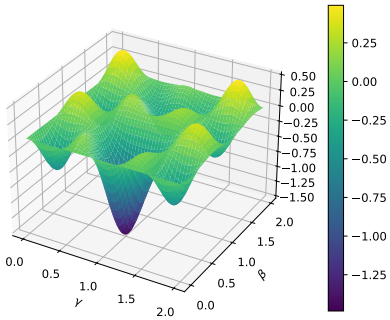


Figure 4. Energy landscape: $E_1(\gamma, \beta)$ computed via the circuit in Fig. 2 with a grid step of 1/100. Lattice of size 2×2. One layer ansatz

denoted as Z^(0.5 ∗) in the circuit diagram. The complete magnetic part of the driver operator is a product of exponentials of Pauli Z-strings performing single-qubit Z rotations and can be represented by a diagonal matrix

$$U_{hZ}(\gamma, \mathcal{H}) = \prod_{j=0}^{3} \exp\left(-i\pi h\gamma Z^{(j)}/2\right),$$

where $Z^{(j)}$ rotates the jth qubit, with diagonal entries:

$$\mathrm{diag}[e^{-2\pi i h\gamma}, e^{-\pi i h\gamma}, e^{-\pi i h\gamma}, 1, e^{-\pi i h\gamma}, 1, 1, e^{\pi i h\gamma}, e^{-\pi i h\gamma}, 1, 1, e^{\pi i h\gamma}, 1, e^{\pi i h\gamma}, e^{\pi i h\gamma}, e^{2\pi i h\gamma}].$$

The mixing operator $U(\beta, B)$ (4) consists of single-qubit X-rotations:

$$R_x(\pi\beta) = e^{-i\pi\beta X/2} = \cos\left(\frac{\pi\beta}{2}\right)\mathbb{I} - i\sin\left(\frac{\pi\beta}{2}\right)X = \begin{bmatrix} \cos\frac{\pi\beta}{2} & -i\sin\frac{\pi\beta}{2} \\ -i\sin\frac{\pi\beta}{2} & \cos\frac{\pi\beta}{2} \end{bmatrix},$$

where X is the Pauli-X gate. In the quantum circuit (Fig. 2), they are denoted as X^(). Thus, the mixing operator is represented by the matrix:

$$U_{Mix}(\beta, B) = e^{-i\pi\beta X^{(3)}/2} e^{-i\pi\beta X^{(2)}/2} e^{-i\pi\beta X^{(1)}/2} e^{-i\pi\beta X^{(0)}/2},$$

which is a product of exponentials of Pauli-X strings $X^{(i)}$ defined as

$$X^{(i)} = \mathbb{I}^{\otimes n} \otimes \underset{\substack{i\text{-th} \\ \text{position}}}{X} \otimes \mathbb{I}^{\otimes(n-i-1)},$$

where the Pauli-X operator acts on the i-th qubit.

Figure 4 shows the energy landscape $E_1(\gamma, \beta)$ (7) for $p = 1$, computed over a two-dimensional grid with a step of 1/100, where the parameters γ and β range from 0 to 2. The characteristic "egg tray" surface exhibits a global minimum (dark blue region) at:

$$\gamma_{\min} = 1.0, \quad \beta_{\min} = 0.5. \tag{10}$$

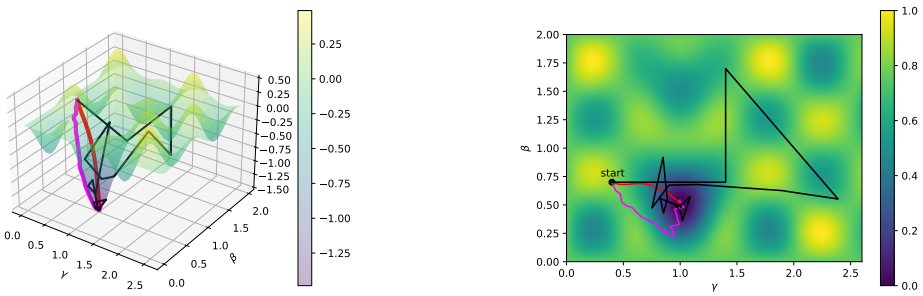


Figure 5. Optimization trajectories

For these parameters, the ground state amplitude α_0 in (8) satisfies $|\alpha_0|^2 \approx 1$ while the amplitudes of all other states vanish within numerical precision.

The energy functional $E_1(\gamma, \beta)$, obtained via the QAOA ansatz, can be minimized using optimization methods implemented on the classical part of the algorithm. Figure 5 shows the optimization trajectories — paths starting from $(\gamma_0, \beta_0) = (0.4, 0.7)$ near a local maximum — on the energy landscape and its 2D projection (contour plot) for three methods[2]:

- Gradient descent method — red trajectory,
- Nelder–Mead method (1965) — magenta trajectory,
- COBYLA method (Powell 1994) — black/white trajectory.

All methods converge to the global minimum (10) within 100 iterations, achieving an error below 1%. Notably, gradient-free methods reach the target more efficiently due to fewer quantum calls — i.e., quantum circuit executions — for the same accuracy. Thus, the COBYLA method achieves an energy minimization precision of 10^{-8} within 40 quantum calls, Nelder–Mead within 90, and gradient descent within 1800.

3.2. Sampling: Estimating Spin Configuration Probabilities

In Sec. 3.1, the energy $E_p(\gamma, \beta)$ was calculated using the state vector with amplitudes α_i (8) prepared by quantum circuits in Fig. 2. However, since the amplitudes α_i cannot be measured directly, the register state is determined through repeated measurements (shots) of the qubits after applying the p-layer QAOA ansatz with fixed values of all $2p$ parameters $\vec{\gamma}, \vec{\beta}$. Then the energy can be approximated by the sum:

$$E_p(\gamma, \beta) \approx \sum_{i=0}^{2^n-1} \mathcal{P}_{i,p}(\gamma, \beta)\mathcal{H}_{ii}, \tag{11}$$

where $\mathcal{P}_{i,p}$ is the empirical probability of finding the register in the i-th basis state, which occurs $n_{ith\ state}$ times in a series of N_{meas} measurements (trials, shots),

$$\mathcal{P}_{i,p}(\gamma, \beta) = n_i / N_{\text{meas}}, \quad \sum_{i=0}^{2^n-1} \mathcal{P}_{i,p} = 1.$$

Convergence to the true energy (7) is achieved as $N_{\text{meas}} \to \infty$.

[2]For references and extended comparison of methods, see section 4.

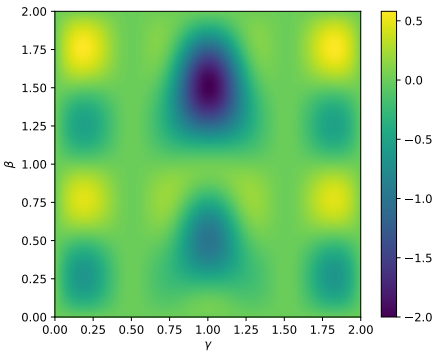


Figure 6. Energy landscape: $E_1(\gamma, \beta)$ computed with a grid step of 1/100. Lattice of size $2 \times 2 \times 2$. One layer ansatz

Table 1

Histogram data of measurement outcomes for the register state $|\psi_p(0.8, 1.2)\rangle$, $N_{\text{meas}} = 10000$

Histogram bin numbers	0	1	2	...	248
Basis state corresponding to the bin	$\|00000000\rangle$	$\|00001000\rangle$	$\|10000000\rangle$	...	$\|10010011\rangle$
Frequency of the basis state	447	301	269	...	1
Energy of the basis state	−2.0	−1.125	−1.125	...	0.5

In this section, we analyze the sampling method for the single-layer QAOA applied to a $2\times2\times2$ Ising model. Figure 6 shows the landscape of the energy $E_1(\gamma, \beta)$ (7), computed via the state vector method from Sec. 3.1 with $J = 1$, $h = 1/2$ in (1). The global minimum $E_1 = -2.0$ occurs at $(\gamma, \beta) = (1.0, 1.5)$, where the amplitude of the state $|00000000\rangle$ equals 1, and the amplitudes of all other states vanish (within numerical precision).

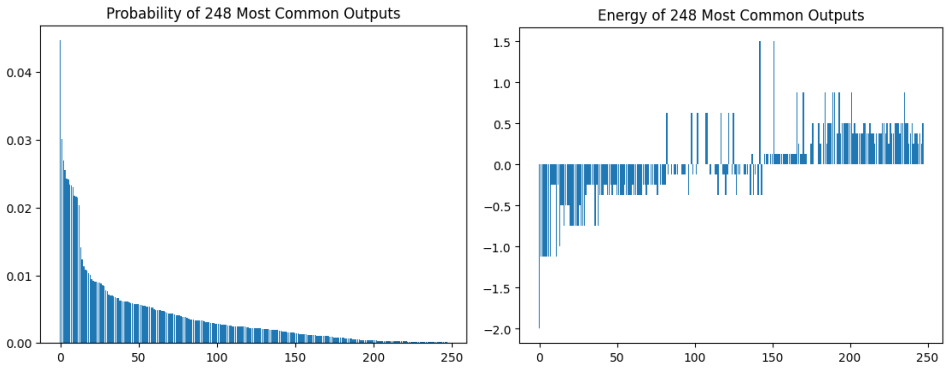
To demonstrate the sampling method, we perform a series of measurements with fixed parameters $(\gamma, \beta) = (0.8, 1.2)$. The computational basis states are enumerated from 0 to 255:

$$|0\rangle \equiv |00000000\rangle,\ |1\rangle \equiv |00000001\rangle,\ \ldots,\ |255\rangle \equiv |11111111\rangle.$$

Table 1 shows the histogram data obtained from $N_{\text{meas}} = 10000$ measurements (shots) of the register state $|\psi_p(0.8, 1.2)\rangle$. The state probabilities and corresponding energies are presented by histograms on Fig. 7. The sampled energy E_1 computed via Eq. (11) agrees with the state vector result from Eq. (7) at $(\gamma, \beta) = (0.8, 1.2)$ with high precision:

$$-0.5187 \text{ (Sampling)} \quad \text{vs.} \quad -0.5184 \text{ (State vector)},$$

with a relative difference of only 3×10^{-4}, which validates the sampling approach.



(a) State Probabilities (b) State Energies

Figure 7. Sampling Histograms

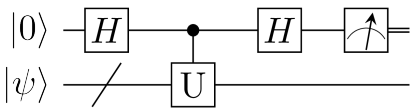


Figure 8. Hadamard test circuit (source: Wikipedia)

3.3. The Hadamard Test: Evaluation of Quantum Averages of Operators

In the examples presented above, the role of the quantum computer was limited to preparing the ansatz state $|\psi_p(\gamma,\beta)\rangle$ (5), while classical computation handled two key tasks. Firstly, it computes energies using either the state vector method (7) or the measurement sampling approach (11). Secondly, it optimizes the parameters (γ,β) based on these energy evaluations.

A more efficient approach involves direct measuring the energy $E_p(\gamma,\beta)$ on the quantum computer exploiting the Hadamard test (Fig. 8) which requires estimating only the final state of a single ancillary qubit. In addition to the QAOA scheme preparing the state $|\psi\rangle \equiv |\psi_p(\gamma,\beta)\rangle$ the following transformations are fulfilled:

$$|0\rangle\otimes|\psi\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(|0\rangle\otimes|\psi\rangle + |1\rangle\otimes|\psi\rangle) \xrightarrow{contr-U} \frac{1}{\sqrt{2}}(|0\rangle\otimes|\psi\rangle + |1\rangle\otimes U|\psi\rangle) \xrightarrow{H}$$
$$\xrightarrow{H} \frac{1}{2}\Big((|0\rangle+|1\rangle)\otimes|\psi\rangle + (|0\rangle-|1\rangle)\otimes U|\psi\rangle\Big) == \frac{1}{2}\Big((|0\rangle\otimes(\mathbb{I}+U)|\psi\rangle + |1\rangle\otimes(\mathbb{I}-U)\psi\rangle\Big).$$

The measurement probabilities of the ancilla qubit's output states $|0\rangle$ and $|1\rangle$ depend on the real part of the operator U:

$$P(0) = \frac{1}{2}\langle\psi|(\mathbb{I}+\mathrm{Re}\,U)|\psi\rangle, \quad P(1) = \frac{1}{2}\langle\psi|(\mathbb{I}-\mathrm{Re}\,U)|\psi\rangle.$$

Given the state normalization $\langle\psi|\psi\rangle = 1$, the real part of the expectation value of the operator equals:

$$\mathrm{Re}\langle\psi|U|\psi\rangle = P(0) - P(1).$$

Using the identity $P(0) + P(1) = 1$, the expectation value can also be expressed as:

$$\mathrm{Re}\langle\psi|U|\psi\rangle = 1 - 2P(1) \quad \text{or} \quad \mathrm{Re}\langle\psi|U|\psi\rangle = 2P(0) - 1.$$

Thus the energy $E(\gamma, \beta)$ (Eq. (6)) can be measured as a sum of expectation values of all Hamiltonian terms:

$$E(\gamma,\beta) = -J \sum_{\langle i,j\rangle} \langle\psi(\gamma,\beta)|Z^{(i)}Z^{(j)}|\psi(\gamma,\beta)\rangle - h\sum_{i}\langle\psi(\gamma,\beta)|Z^{(i)}|\psi(\gamma,\beta)\rangle. \tag{12}$$

4. Optimization

4.1. Computational Setup: Software and Hardware

Initially, the problem was solved interactively on a quantum testbed [11, 12] using `Cirq` [9], a Python library for quantum circuit simulation (supporting up to 30 qubits). However, as the lattice dimension increases, the problem complexity grows exponentially (see Appendix), requiring substantial RAM and high-performance CPU/GPU resources. Even for the $3 \times 3 \times 3$ lattice case, we had to employ the `qsim` simulator, a C++ backend integrated with `Cirq`. It uses SIMD (Single Instruction, Multiple Data) for vectorization and `OpenMP` for CPU multithreading, and supports GPU acceleration via NVIDIA's `cuStateVec` [13], a high-performance, GPU-accelerated library for quantum state vector operations. This hybrid configuration enables simulation of large-scale quantum circuits (up to approximately 40 qubits) while providing significant computational acceleration.

The results presented below were obtained employing the `Cirq+qsim` framework, with computations performed on both a central processing unit (CPU) and a graphics processing unit (GPU) in two configurations:

1. a basic CUDA implementation;
2. the `cuStateVec` library from the NVIDIA `cuQuantum` SDK [14], which delivered superior performance for quantum circuit simulations, especially for sampling-based methods.

We performed simulations on the workstation equipped with CPU 8-core AMD Ryzen 7 3700X, DDR4 RAM 128 GB, and the GPU NVIDIA GeForce RTX 4070 Ti SUPER.

4.2. Computational Resource Requirements for Simulation

Energy minimization was done for six spin lattice configurations, as shown in Table 2. The table contains key parameters that determine computational complexity: the number of qubits in the quantum circuit, the number of terms in the Hamiltonian, and the dimension of the state vector of the register (i.e., the number of spin configurations). For each lattice configuration, we employed a three-layer QAOA ansatz with six variational parameters: $(\gamma_1, \beta_1, \gamma_2, \beta_2, \gamma_3, \beta_3)$.

The energy was computed using four distinct approaches to compare them in terms of numerical accuracy, runtime performance, and memory requirements:

1. QAOA–State vector method: uses Equation (7),
2. QAOA–Sampling method: uses Equation (11),
3. Hadamard–State vector method: uses the Hadamard test circuit (Figure 8) with Equation (7),
4. Hadamard–Sampling method: uses the Hadamard test circuit with ancilla qubit measurement and Equation (12).

Table 2

Spin Lattices. Calculations with the `cuStateVec` library

Lattice Data		Energy Operator			Memory required for state vector [3]	Runtime ratio (Hadamard/QAOA)	
Lattice	qubits, n	ZZ terms	μZ terms	Total		State vector	Sampling 1000 shots
3×3	9	12	9	21	512 B	22.0	20.2
$3 \times 2 \times 2$	12	20	12	32	4 KiB	35.1	29.4
4×4	16	24	16	40	64 KiB	76.0	37.9
$3 \times 3 \times 2$	18	33	18	51	256 KiB	204.0	48.0
5×5	25	40	25	65	32 MiB	566.3	120.3
$3 \times 3 \times 3$	27	54	27	81	128 MiB	997.3	160.9

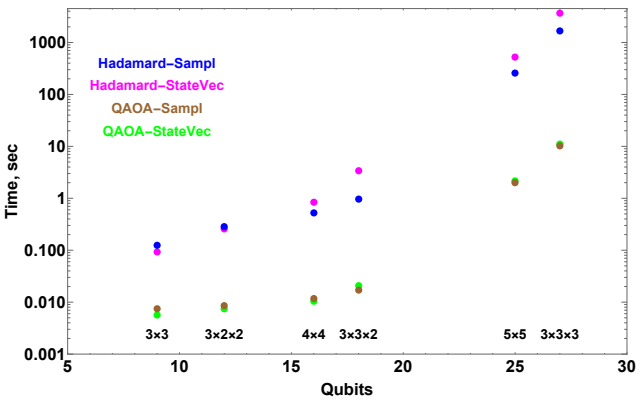


Figure 9. CPU runtime versus number of qubits for energy computations in different quantum simulation approaches

A single energy evaluation reveals significant performance differences among the methods. Figure 9 shows the CPU execution time as a function of the number of qubits. Clearly, the Hadamard-based methods exhibit significantly longer runtimes than the QAOA-based approaches. The time ratio Hadamard/QAOA approaches using the state-vector and sampling methods are shown in the last two columns of Table 2. Timing and memory analysis of all methods — illustrated for the 5×5 2D lattice in the Appendix — demonstrates that the `cuStateVec` library is preferable for the energy minimization procedure.

4.3. Used Local and Global Optimization Methods

For local optimization, we compare a simple gradient-based method with the following gradient-free methods [4]:

1. COBYLA (Constrained Optimization BY Linear Approximation) (Powell, 1994 [15]): a simplex method utilizing trust regions and linear approximation of the objective function.
2. Modified Powell Method (Powell, 1964 [16]): a conjugate direction algorithm.
3. Nelder–Mead Simplex (Nelder and Mead, 1965 [17]): the classical simplex direct search method, known for its simplicity and robustness.

All methods used the same initial point:

$$(\gamma_1, \beta_1, \gamma_2, \beta_2, \gamma_3, \beta_3) = (0.1, 0.1, 0.5, 1.0, 1.5, 0.5).$$

The components of the energy gradient vector were computed numerically using central finite differences. During minimization, the parameters at each iteration were updated in proportion to the gradient components, and the algorithm terminated after 100 iterations.

We tested and compared all energy computation methods described in section 3. Gradient-based optimization succeeded for both QAOA and Hadamard circuits on lattices up to size 5×5 (25 qubits), but only when using state vector simulation. With the sampling approach, gradient-based minimization failed to produce reliable values of the gradients even for small lattices. This suggests that either more advanced algorithms are required or the number of measurements must be significantly increased (well beyond 1000 shots).

Only gradient-free methods were used in conjunction with sampling (1000 shots per parameter set), for three reasons:

1. gradient-based minimization consistently failed under sampling procedure,
2. it is the sampling that is relevant approach for real quantum computers,
3. sampling is significantly faster and less memory intensive for large lattices (5×5, $3 \times 3 \times 3$), as demonstrated in the Appendix.

The derivative-free methods successfully optimized both QAOA and Hadamard circuits for all lattice sizes. Since the energy is evaluated only once per minimization step, gradient-free methods are less sensitive to energy estimation errors than gradient-based methods, especially in high-dimensional parameter spaces.

To compare the gradient-free methods, we present the results for the 5×5 lattice test (Table 3). The number of energy evaluations (NEEs) was independent of lattice size across all methods and simulation schemes. The following conclusions can be drawn from this table:

- COBYLA outperformed other methods in terms of NEEs. The Nelder–Mead method showed the worst performance, requiring the highest number of NEEs, resulting in longer runtime and less accurate energy estimates. In general, QAOA scheme required 5–10 percent fewer NEEs than the Hadamard scheme.
- COBYLA's time per energy evaluation is approximately twice that of a single evaluation in either the QAOA or Hadamard scheme (see Appendix). In contrast, Powell's method requires only a few percent more time than a single evaluation. The Nelder–Mead algorithm occupies an intermediate position.

For the global search of the energy minimum, we employed two methods from the `scipy.optimize` module of the SciPy library:

[4] Gradient-free methods correspond to arguments of the `scipy.optimize.minimize` function in the SciPy library.

Table 3
Performance of local gradient-free optimization methods for QAOA and Hadamard circuits on a 5×5 lattice. The true ground-state energy is -2.1. Simulations performed using sampling with 1000 shots per evaluation and the `cuStateVec` backend

Method	**COBYLA**		**Powell**		**Nelder–Mead**	
Schemes	QAOA	Hadamard	QAOA	Hadamard	QAOA	Hadamard
Energy	-2.0990	-2.0999	-2.0996	-2.0999	-1.9339	-1.9621
Relative error (%)	0.0476	0.0048	0.0190	0.0048	7.9095	6.5667
Found parameters γ_1	0.0027	0.0007	0.0056	1.9979	0.0984	0.1006
Found parameters β_1	0.1159	0.0968	0.4721	1.0655	0.1085	0.1031
Found parameters γ_2	0.4982	0.5010	0.5054	0.5026	0.4343	0.4215
Found parameters β_2	1.0002	1.0000	1.0011	0.9996	0.9976	0.9995
Found parameters γ_3	1.5006	1.4992	1.5018	1.4992	1.4996	1.4930
Found parameters β_3	0.5105	0.5026	0.5079	1.5013	0.5263	0.5325
Total NEEs	78	100	95	105	246	251
%%time Wall time	19.9s	51m 23s	12.4 s	27m 3s	44.2s	1h 29m

- Dual Annealing (DA) [18]: a stochastic global optimization algorithm based on a modified form of simulated annealing;
- Simplicial Homology Global Optimization (SHGO) [19] with Sobol sampling: a global optimization method based on simplicial integral homology and combinatorial topology.

For local optimization both methods used the COBYLA and Powell algorithms, combined with a sampling approach employing 1000 shots per parameter set. The performance of the global optimization methods is summarized in Table 4. Comparing the algorithms, the following observations can be made:

- Both global algorithms achieved solutions within a few percent of the true ground-state energy, independent of lattice size.
- When using COBYLA, fewer energy evaluations were required compared to Powell (consistent with standalone local method comparisons).
- When paired with COBYLA, both global methods required fewer energy evaluations than when using Powell, consistent with standalone comparisons of the local optimizers.
- The combination of SHGO and Powell resulted in the highest computational cost in terms of total energy evaluations.
- The DA algorithm with COBYLA required 3–4 times fewer energy evaluations than SHGO with Powell.
- The number of iterations and energy evaluations was largely independent of lattice size across all algorithm combinations.

Table 4

Performance of global optimization methods for QAOA and Hadamard circuits on a 5 × 5 lattice. The true ground-state energy is −2.1. Simulations use sampling with 1000 shots per evaluation and the `cuStateVec` backend. (Here n_{fev}: total number of energy evaluations; n_{lfev}: number of evaluations during local optimization)

Method	**Dual Annealing (DA)**				**SHGO**			
Local optimizer	COBYLA		Powell		COBYLA		Powell	
Scheme	QAOA	Hadamard	QAOA	Hadamard	QAOA	Hadamard	QAOA	Hadamard
Energy	-2.0002	-2.0960	-2.1000	-2.0962	-2.09928	-2.09936	-2.1	-2.1
Relative error (%)	4.7524	0.1905	0.0000	0.1810	0.0333	0.0286	0.0000	0.0000
Optimized parameters γ_1	1.5554	0.5091	0.6325	1.4458	1.8313	0.5015	1.3915	1.8454
β_1	1.0032	1.0019	0.0023	0.9998	1.0000	1.0005	0.9985	0.9988
γ_2	1.3628	0.3831	0.8668	0.5488	0.5059	0.3342	0.4918	0.4991
β_2	1.9072	1.9823	0.9987	0.4257	0.9959	1.9950	0.9932	1.0007
γ_3	1.0862	1.1088	0.4979	0.0746	1.6644	1.1650	0.1174	1.6564
β_3	0.5046	0.5053	1.4992	1.0929	1.5030	0.5024	0.5051	1.4990
Evals of energy: n_{fev} (DA) $n_{\mathrm{fev}}/n_{\mathrm{lfev}}$(SHGO)	1342	1387	2439	3072	1105/487	1146/528	4991/4372	3962/3343
%%time Wall time	2 min 53 s	5 h 55 min	5 min 14 s	13 h 6 min	3 min 47 s	7 h 9 min	11 min 11 s	17 h 2 min
Memory	256 KiB	25.5 MiB	256 KiB	56.0 MiB	213.7 MiB	240.4 MiB	28.5 MiB	99.8 MiB

Regarding timing characteristics:

- For the DA algorithm, the average energy evaluation time closely matched that of standalone simulations for both the QAOA and Hadamard measurement schemes.
- SHGO with COBYLA exhibited 1.5–2× longer average evaluation time compared to standalone runs, while with Powell, the overhead was approximately 10%.

In terms of memory usage, SHGO required several hundred megabytes of additional memory, while DA used only several hundred kilobytes — independent of lattice size. Notably, SHGO provides a set of near-optimal parameter configurations, which represents a key advantage over DA, especially for identifying multiple low-energy states.

5. Summary

We investigate the performance of the Quantum Approximate Optimization Algorithm (QAOA) [6] through simulations of the Ising model in a longitudinal magnetic field. Using the `Cirq+qsim` framework, we construct a QAOA-generated variational wave function ansatz and employ optimization methods from `SciPy`'s library to estimate the ground-state energy.

The algorithm's advantage over other variational methods stems from its task-adaptive structure: the driver operator, defined as the time-evolution operator under the system Hamiltonian, enables efficient optimization while mitigating the barren plateau problem [4]. We employ the Ising model due to its importance in the study of spin systems and quantum field theories on space-time lattices.

We demonstrate a mapping between spin configurations on a spatial lattice and quantum register states, with each spin at a lattice site corresponding to a single qubit.

While the quantum register size scales linearly with the number of spins n, classical simulations require exponential resources (2^n complex amplitudes) to represent the full state vector and Hamiltonian matrix. To maintain high-fidelity ground-state approximations, the QAOA ansatz requires deeper circuits — i.e., an increasing number of layers — as n increases.

We analyze QAOA performance on both 2D and 3D lattice configurations and compare two measurement schemes: the standard QAOA scheme (see Figure 2) and the Hadamard test (see Figure 8), with energy calculations performed using either full state-vector simulations or sampling. The Hadamard test, introduced in Section 3.3, fully leverages the capabilities and demonstrates the advantage of a quantum computer by estimating the average energy through measurement of a single ancilla qubit. Modeling of state-vector simulations requires more memory but demonstrates faster execution than the Hadamard test (Sec. 3). Direct state-vector calculations for energy estimation in both schemes consume significantly more memory than sampling-based approaches. The sampling method exhibits nearly flat resource scaling: increasing the number of shots by an order of magnitude results in only a few percent increase in memory and computation time.

For parameter optimization, we compare gradient-based and gradient-free approaches. Gradient-based methods offer higher precision but are prone to convergence instability. In contrast, gradient-free techniques — such as COBYLA, Powell, and Nelder–Mead — exhibit greater robustness and lower computational overhead. The optimization landscape exhibits multiple local minima, which significantly hinders optimization — a well-known challenge in variational quantum algorithms. This obstacle provides another argument in favor of gradient-free methods, which can avoid local minima through adequate construction of the simplex in the parameter space.

The choice of measurement scheme and optimization algorithm for QAOA simulations ultimately depends on the research objectives — such as ansatz verification versus modeling real quantum hardware — as well as available computational resources and time constraints.

Table 5

Memory requirements for state vectors and unitary operators

Lattice Size	Qubits (n)	$\dim(\|\psi\rangle)$	State Vector Memory	Unitary Operator Memory
3×3	9	512	4 KiB	4 MiB
4×4	16	65,536	512 KiB	64 GiB
5×5	25	33,554,432	256 MiB	16 PiB

The developed computational framework demonstrates straightforward scalability to arbitrary lattice dimensions. However, one should keep in mind the inevitable exponential growth in required computational resources with increasing system size. Our calculations were partially performed on JINR's HybriLIT quantum testbed [12]. Leveraging the cuStateVec library on NVIDIA A100 GPUs dramatically accelerated computations, reducing execution time (compared to an optimized PC used in this work) from three days to 14 minutes for energy surface plot (Fig. 4). Complete testbed implementation details are given in [20].

1. Appendix: Time and Memory Requirements for Energy Computation Methods

A state vector $|\psi\rangle$ and unitary matrix U for an n-qubit register have the following dimensions: $\dim(|\psi\rangle) = 2^n$, $\dim(U) = 2^n \times 2^n = 2^{2n}$, where the state vector uses `complex64` and unitary operators `complex128` by default in the `Cirq` library. The required memory for storing $|\psi\rangle$ and U can be estimated using:

$$\text{mem}(|\psi\rangle) = \frac{2^n \times 64}{8} \quad \text{bytes}, \qquad \text{mem}(U) = \frac{2^{2n} \times 128}{8} \quad \text{bytes}.$$

As concrete examples, consider 2D lattices of different sizes and their corresponding qubit counts:

While the memory requirements for state vectors remain manageable even for larger systems, the storage needed for unitary operators grows exponentially. Despite these computational challenges, the `Cirq+qsim` framework enables efficient simulation of quantum circuits on personal computers through optimized sparse matrix representations, advanced state vector compression techniques, and parallel computation strategies.

Table 6 presents a comparative analysis of computational resources required for energy calculation by all the methods considered above on a 5×5 lattice using the same set of variational parameters (γ, β). We measured time and memory consumption using both the *psutil* utility and IPython's magic commands (%memit, %timeit).

The energy value is determined more accurately using the state-vector method. However, as shown in Table 7, comparable accuracy can be achieved by increasing the number of samples, with only a slight increase in time and memory requirements. As noted in Section 3, all methods employing the Hadamard test (see Table 7) require more time than those using just QAOA scheme (rows labeled "QAOA"). However, the advantage of the Hadamard-based methods lies in their superior memory efficiency, especially when combined with sampling. Based on the data in table 6, we can briefly characterize the studied methods as follows:

1. QAOA–State Vector: exact, low runtime, high memory usage;

Table 6
Performance metrics for 5×5 lattice simulations: wall time and memory usage. Energy values are averaged over 1000 shots for sampling methods (values near zero indicate negligible memory overhead)

Scheme	Backend	Method	Energy	Wall time	Memory increment (%memit, MiB)	RSS (psutil, MiB)
State Vector	CPU	QAOA	-1.6556	2.16 s	512.25	2125.49
		Hadamard	-1.6556	8 min 46 s	1536.41	331.5
	GPU (CUDA)	QAOA	-1.6556	550.2 ms	502.00	2221.46
		Hadamard	-1.6556	5 min 9 s	1536.03	426.46
	GPU (cuStateVec)	QAOA	-1.6556	558.2 ms	484.05	2231.09
		Hadamard	-1.6556	5 min 16 s	1535.87	437.59
Sampling (1000 shots)	CPU	QAOA	-1.6372	1.98 s	255.88	2125.72
		Hadamard	-1.6592	4 min 20 s	512.34	331.55
	GPU (CUDA)	QAOA	-1.6461	4.17 s	0.00	2222.59
		Hadamard	-1.6402	8 min 57 s	0.00	426.48
	GPU (cuStateVec)	QAOA	-1.6526	127.3 ms	0.00	2234.71
		Hadamard	-1.6545	15.23 s	0.12	440.70

Table 7
Time and memory consumption as a function of the number of measurement shots. Simulations for a 5×5 2D lattice using the `cuStateVec` backend.

Metric	QAOA			Hadamard		
Number of shots	1000	5000	10000	1000	5000	10000
Energy value	-1.6526	-1.6518	-1.6508	-1.6545	-1.6511	-1.6553
Wall time (%timeit)	127.3 ms	149.2 ms	175.6 ms	15.23 s	16.23 s	17.46 s
RSS (psutil, MiB)	2234.71	2235.84	2237.59	440.70	441.57	443.20

2. QAOA–Sampling: lower accuracy, lowest runtime, moderate memory;
3. Hadamard–State Vector: exact, highest runtime, low memory;
4. Hadamard–Sampling: lower accuracy, high runtime, lowest memory.

Comparing computational performance between CPU and GPU implementations (using CUDA and `cuStateVec`), the `cuStateVec` library demonstrates superior efficiency across all metrics. For energy calculations, `cuStateVec` outperforms both CPU and native CUDA implementations in terms of computational speed (wall time), memory efficiency (peak memory usage), and numerical accuracy (energy expectation values). The native CUDA implementation shows comparable performance to `cuStateVec` for state-vector simulations but suffers significant performance degradation with sampling-based methods (1000 shots), where it becomes less efficient than both the CPU baseline and the `cuStateVec` implementation.

Author Contributions: Conceptualization, Yuri G. Palii; Methodology, Yuri G. Palii and Alla A. Bogolubskaya; Software, Yuri G. Palii, Alla A. Bogolubskaya and Denis A. Yanovich; Validation, Yuri G. Palii and Alla A. Bogolubskaya; Formal analysis, Yuri G. Palii; Investigation, Yuri G. Palii, Alla A. Bogolubskaya, Denis A. Yanovich; Writing — original draft preparation, Yuri G. Palii; Writing — review and editing, Yuri G. Palii and Alla A. Bogolubskaya; Visualization, Alla A. Bogolubskaya and Denis A. Yanovich; Supervision, Yuri G. Palii; Project administration, Yuri G. Palii and Alla A. Bogolubskaya. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Codes are available by demand from authors.

Acknowledgments: The authors thank Dmitry V. Belyakov, lead programmer at MLIT JINR, and Maxim I. Zuev, research scientist at MLIT JINR, for configuring the `Cirq` library software environment and providing computational technical support.

Conflicts of Interest: The authors declare no conflict of interest.

Declaration on Generative AI: The authors have not employed any Generative AI tools.

References

1. Pedersen, J. W., Lamm, H., Lawrence, S. & Yeter-Aydeniz, K. Quantum Simulation of Finite Temperature Schwinger Model via Quantum Imaginary Time Evolution. *Phys. Rev. D* **108,** 114506. doi:10.1103/PhysRevD.108.114506. arXiv: 2304.01144 [hep-lat] (2023).
2. Jordan, S. P., Lee, K. S. M. & Preskill, J. Quantum Algorithms for Quantum Field Theories. *Science* **336,** 1130–1133. doi:10.1126/science.1217069. arXiv: 1111.3633 [hep-th] (2012).
3. Abhijith, J. *et al.* Quantum Algorithm Implementations for Beginners. *Quantum* **6,** 881. doi:10.22331/q-2022-12-22-881. arXiv: 1804.03719 [cs.ET] (2022).
4. Bharti, K. *et al.* Noisy intermediate-scale quantum (NISQ) algorithms. *Rev. Mod. Phys.* **94,** 015004. doi:10.1103/RevModPhys.94.015004. arXiv: 2101.08448 [quant-ph] (2022).
5. Hidary, J. D. *Quantum Computing: An Applied Approach* 2nd. doi:10.1007/978-3-030-23927-6 (Springer Nature Switzerland AG, Cham, Switzerland, 2021).
6. Farhi, E., Goldstone, J. & Gutmann, S. *A Quantum Approximate Optimization Algorithm* 2014. arXiv: 1411.4028 [quant-ph].
7. Lotshaw, P. C. *et al.* Simulations of Frustrated Ising Hamiltonians using Quantum Approximate Optimization. *Phil. Trans. R. Soc. Lond. A* **381,** 20210414. doi:10.1098/rsta.2021.0414. arXiv: 2206.05343 [quant-ph] (2022).
8. Ozaeta, A., van Dam, W. & McMahon, P. L. Expectation Values from the Single-Layer Quantum Approximate Optimization Algorithm on Ising Problems. *Quantum Sci. Technol.* **7,** 045036. doi:10.1088/2058-9565/ac88d4. arXiv: 2012.0342 [quant-ph] (2022).

9. AI, G. Q. *Cirq: Quantum approximate optimization algorithm for the Ising model* https://quantumai.google/cirq/experiments/qaoa/qaoa_ising. Accessed: 2024-04-01. 2023.
10. Bengtsson, I. & Życzkowski, K. *Geometry of Quantum States. Introduction to Quantum Entanglement* Second. doi:10.1017/9781139030710 (Cambridge University Press, Cambridge, UK, 2017).
11. Palii, Y., Bogolubskaya, A. & Yanovich, D. Quantum Approximation Optimization Algorithm for the Ising Model in an External Magnetic Field. *Phys. Part. Nucl.* **55,** 600–602. doi:10.1134/S1063779624040185 (2024).
12. Belyakov, D. V., Bogolyubskaya, A. A., Zuev, M. I., Palii, Y. G., Podgajny, D. V., Streltsova, O. I. & Yanovich, D. A. *Polygon for quantum computing on the heterogeneous HybriLIT platform* Russian. in *Proc. of the International Conference "Information Technologies and Mathematical Methods" (ITTMM-2024)* (In Russian) (2024), 303.
13. Corporation, N. *NVIDIA cuStateVec: A High-Performance Library for Quantum Circuit Simulation* https://docs.nvidia.com/cuda/cuquantum/custatevec/index.html. Accessed: 2024-04-01. 2023.
14. Bayraktar, H. *et al. NVIDIA cuQuantum SDK, Accelerate quantum computing research* 2023. arXiv: 2308.01999 [quant-ph].
15. Powell, M. J. D. A direct search optimization method that models the objective and constraint functions by linear interpolation. *Advances in Optimization and Numerical Analysis* (eds Gomez, S. & Hennart, J.-P.) 51–67. doi:10.1007/978-94-015-8330-5_4 (1994).
16. Powell, M. J. D. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *Comput. J.* **7,** 155–162. doi:10.1093/comjnl/7.2.155 (1964).
17. Nelder, J. A. & Mead, R. A simplex method for function minimization. *Comput. J.* **7,** 308–313. doi:10.1093/comjnl/7.4.308 (1965).
18. Xiang, Y., Sun, D. Y., Fan, W. & Gong, X. G. Generalized simulated annealing algorithm and its application to the Thomson model. *Phys. Lett. A* **233,** 216–220. doi:10.1016/S0375-9601(97)00484-1 (1997).
19. Endres, S. C., Sandrock, C. & Focke, W. W. A simplicial homology algorithm for Lipschitz optimization. *J. Glob. Optim.* **72,** 181–217. doi:10.1007/s10898-018-0645-y (2018).
20. Palii, Y., Belyakov, D., Bogolubskaya, A., Zuev, M. & Yanovich, D. *Simulation of the QAOA algorithm at the JINR quantum testbed* Russian. in *Proc. of the International Conference "Mathematical Problems in Quantum Information Technologies" (MPQIT 2024)* In press (Dubna, JINR, 2024).

Information about the authors

Palii, Yuri G.—Candidate of Physical and Mathematical Sciences, Senior Researcher of Laboratory of Information Technologies, Joint Institute for Nuclear Research (e-mail: palii@jinr.ru, phone: +7(496)2162235, ORCID: 0000-0001-9000-9794)

Bogolubskaya, Alla A.—Candidate of Physical and Mathematical Sciences, Senior Researcher of Laboratory of Information Technologies, Joint Institute for Nuclear Research (e-mail: abogol@jinr.ru, phone: +7(496)2164015, ORCID: 0000-0003-4356-8336, Scopus Author ID: 6508333497)

Yanovich, Denis A.—Senior Researcher of Laboratory of Information Technologies, Joint Institute for Nuclear Research (e-mail: yan@jinr.ru, phone: +7(496)2162552)

УДК 519.6, 530.145.67
PACS 03.67.Ac, 03.67.Lx,
DOI: 10.22363/2658-4670-2025-33-4-440-460 EDN: HYWEXV

Моделирование работы QAOA с использованием квантовых фреймворков Cirq и qsim

Ю. Г. Палий, А. А. Боголюбская, Д. А. Янович

Объединённый институт ядерных исследований, ул. Жолио-Кюри, д. 6, Дубна, 141980, Российская Федерация

Аннотация. В работе рассмотрено решение задачи поиска состояния с минимальной энергией в модели Изинга с продольным магнитным полем для двух- и трёхмерных решёток различных размеров на квантовом компьютере с использованием квантового приближённого алгоритма оптимизации (QAOA). Базисные состояния квантового регистра соответствуют конфигурациям спинов на пространственной решётке, а гамильтониан модели реализуется с помощью последовательности квантовых вентилей. Среднее значение энергии эффективно измерено с помощью теста Адамара. Работа алгоритма QAOA моделируется для последовательно усложняющихся решёточных конфигураций с применением библиотек `Cirq` и `qsim`. Результаты оптимизации, проведённой градиентным и безградиентными методами, свидетельствуют о предпочтительности последних как с точки зрения моделирования работы, так и с точки зрения использования квантового компьютера. Приведены ключевые аргументы в пользу преимуществ квантовых вычислений для решения данной задачи.

Ключевые слова: квантовые вычисления, квантовый приближённый алгоритм оптимизации (QAOA), модель Изинга, симуляция квантовых вычислений, оптимизация, Cirq, qsim, cuStateVec

Discrete & Continuous Models & Applied Computational Science
ISSN 2658-7149 (Online), 2658-4670 (Print)

2025, 33 (4) 461–470
http://journals.rudn.ru/miph

Research article

UDC 533.924
PACS 52.77.-j, 52.80.Pi,
DOI: 10.22363/2658-4670-2025-33-4-461-470 EDN: HMKEYN

Ar-O_2 plasma of resonant UHF discharge for chitosan's films processed

Andrey V. Artemyev, Andrey S. Kritchenkov

RUDN University, 6 Miklukho-Maklaya St, Moscow, 117198, Russian Federation

(received: August 16, 2025; revised: September 5, 2025; accepted: September 10, 2025)

Abstract. This article explores the modification of chitosan films by treating Ar-O_2 with microwave plasma. The main idea is to use resonant plasma generation methods to treat chitosan films. Spectral and energy characteristics of microwave plasma are obtained. The mechanical properties, swelling, and solubility of chitosan films exposed to microwave plasma are studied. The dependence of film properties on the duration of treatment with resonant microwave plasma is demonstrated.

Key words and phrases: plasma, resonance, UHF discharge, microwave plasma, chitosan, film modification, swelling, solubility, experimental studies

For citation: Artemyev, A. V., Kritchenkov, A. S. Ar-O_2 plasma of resonant UHF discharge for chitosan's films processed. *Discrete and Continuous Models and Applied Computational Science* **33** (4), 461–470. doi: 10.22363/2658-4670-2025-33-4-461-470. edn: HMKEYN (2025).

1. Introduction

Nowadays, promising applications have emerged in the field of plasma modification of surfaces, such as cleaning, sterilization, modification and coatings for biomedical applications. For these purposes, the functions of biomaterial surfaces are frequently controlled by their contact to biological materials. These functionalities are often used to improve the polymer biocompatibility to obtain a selective control of the biological response. Thus, the development of techniques for rapid and convenient directed modification of physicochemical properties of chitosan film materials is an important task for the field of polymer science focusing on direct interaction with regenerative medicine. It is known that low temperature plasma exposure of film materials currently represents one of the most effective and most environmentally friendly approaches aimed at modifying the surface properties of polymeric materials [1–3]. It has been repeatedly demonstrated in several works that high-energy low-temperature plasma exposure holds great promise in molding various materials from chitosan and/or its derivatives and modulating their properties for subsequent potential biomedical applications [4, 5]. In the presented study, we evaluated the effect of exposure to microwave plasma discharges of an argon-oxygen mixture on the mechanical properties, swelling, and solubility of chitosan films. The main difference between this study and early-published works is that the research installation with a resonant microwave discharge as a source of plasma flow provides a wide variation of the generated plasma parameters.

2. Set Up and Diagnostics

The linear magnetized plasma multistage device RAPIRA (Resonant Accelerated Plasma Installation Research & Application) was used as a test bed to address anticipated research and development issues associated with RF(Helicon) and UHF (ECR)-discharges and establish the physics basis for investigation of Plasma Material Interactions (PMI). The ECR and Helicon setups is parts of the RAPIRA device, which is a cylindrical, linearly magnetized plasma experiment. Fig. 1 shows a schematic drawing of RAPIRA indicating the position of the Helicon and the ECR parts. Installation is a 4 m long hybrid stainless steel - quartz vacuum volume. The diameter of the quartz tube is 65 mm, while stainless-steel receiving chamber is of the 0.82 m inner diameter, 2.0 m long.

Vacuum volume including stainless-steel receiving chamber 7, quartz tube 2 and system of gas feeder 1 (Fig. 1) was evacuated down to a base pressure of 10^{-6} Torr by tandem of turbomolecular pump and a rotary pump connected to receiving chamber. Plasma-forming gases (Ar; H_2; N_2; He; Kr; O_2) and their mixtures were fed into the chamber using a mass-flow controller, typical values of pressure during operation lay in the 0.5 mTorr range. Pressure was monitored using Pirani and ionization gauges. This paper presents the results of processing chitosan films with a mixture Argon (80%) and O_2 (20%) plasma flow generated only by a UHF source. ECR plasma sources are known to provide very high ionization efficiencies due to the resonant transfer of microwave energy to the electrons [6, 7]. The experiments with UHF-discharge have been carried out on the part of RAPIRA, which is sketched in Fig. 1. The UHF-cavity 4 excited by a microwave source in pulsed or continuous operation mode immersed in static magnetic field produced by coils 5 which allows to keep induction within cavity's volume up to 1500 Gs and variable longitudinal gradients of magnetic field no less 25 Gs/cm. A 2.45 GHz magnetron (Muegge MX3000D-160KL) delivers up to 3 kW of microwave power, which is coupled via a rectangular-to-coaxial waveguide transition into the cavity. The coaxial adapter, waveguide and the cavity were matched. When the cavity is in resonance UHF-power reflection is very small. But if the cavity is loaded by plasma, power reflected from the cavity comes back to the circulator where it can be measured correctly. The use of a circulator with a cooled load virtually removes restrictions on the level of reflected power. The pulse durations of the microwave source can vary widely from 1 s to continuous operation. The cavity is excited by the TE_{111} oscillation mode, which is optimal for generating electron cyclotron resonance plasma under typical conditions.

The main parameters of the installation are presented in Table 1. The use of these parameters allows to obtain plasma formations with temperature in the range from 5 to 50 eV, concentrations of charged particles in the range from 10^9–10^{11} cm^{-3} and plasma flow, with ionic current density on sample processing up to 200 $\mu A/cm^2$.

For spectrometric studies, we used an MS3504i monochromator-spectrograph with astigmatism compensation. The spectrum was recorded in the monochromator mode. The spectrometer was

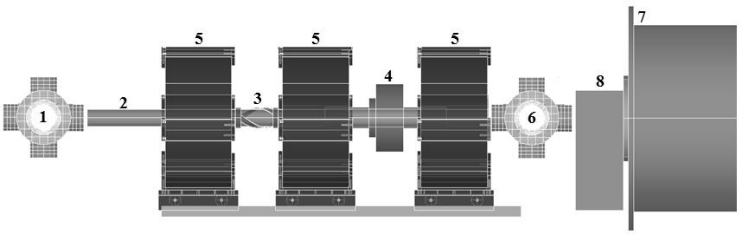


Figure 1. Multistage device RAPIRA

Table 1

Installation parameters

Working gas	Working pressure, Torr	RF-discharge		UHF-discharge		Magnetic system		Film mounting system for processing
		Frequency, MHz	Power, W	Frequency, GHz	Power, W	Field amplitude, G	Field gradient, G/cm	
H_2, N_2, O_2, Ar, gas mixtures	10^{-4} through 10^{-2}	13.56	to 750	2.45	to 3000	to 2800	to 100	From 1 to 12 films are simultaneously placed in the vacuum volume

calibrated over the wavelength by using a DRS 50-1 mercury lamp with allowance for the parameters of the recording system and over the absolute intensity by using an SIRSh 6-40 lamp (certified at the All-Russia Research Institute of Optical and Physical Measurements). The plasma flow generated was diagnosed using multigrid energy analyzer inserted receiving chamber (7, Fig. 1).

For plasma flow treatment, polymer biocompatible biodegradable films based on natural (chitosan, pullulan, gelatin) and synthetic (polylactide) polysaccharides are produced with the possibility of adding inorganic fillers (0.1–5% of layered double hydroxide Mg2+/Fe3+ or commercially available montmorillonite) [8, 9]. Polymer and composite films are produced by pouring molding solutions into Petri dishes, followed by drying at room temperature at either 60°C, or 90°C in a dry oven. Purified water (GF XV) is used as a solvent, and if chitosan is present in the molding solution, a 1% aqueous solution of acetic acid is used.

To accumulate statistical data on the results of plasma treatment, a lot of six film samples was simultaneously loaded into the receiving chamber. The samples were mounted on a mechanical carousel system that ensured precise positioning of the sample under the plasma flow. The remaining films are not exposed and are processed in experiments with changed parameters.

The RAPIRA-Experiments is equipped with a large diagnostic suite. Table 2 gives an overview of the diagnostics used.

The process of film modification needs to be monitored. Mechanical properties of the obtained films were studied in uniaxial tensile mode on a tensile machine (REM, Russia, the size of the samples was 6×2 cm, the speed of uniaxial tension was 50 mm/min). The solubility and swelling of the obtained films in purified water were evaluated gravimetrically [10, 11]. Infrared spectroscopy allows detection of the mentioned processes but has not yet become widespread as a monitoring of surface chemical modification of films [12]. In this work, the aim was to record the presence or absence of changes of plasma-treated chitosan films under UHF-discharge.

Table 2

Summary of RAPIRA-Experiments Plasma and Physicochemical Diagnostics

Plasma diagnostics	Physicochemical diagnostics
Falling and reflected power measurements UHF, RF	Swelling (%)
Distributer pressure measurements	Solubility (%)
Langmuir probes (T_e, n_e)	Tensile strength (MPA)
Retarding field analyzer probe (ion, electron EDF)	Elongation at break (%)
Optical emission spectroscopy (T_e, n_e)	IR-spectroscopy (cm^{-1})

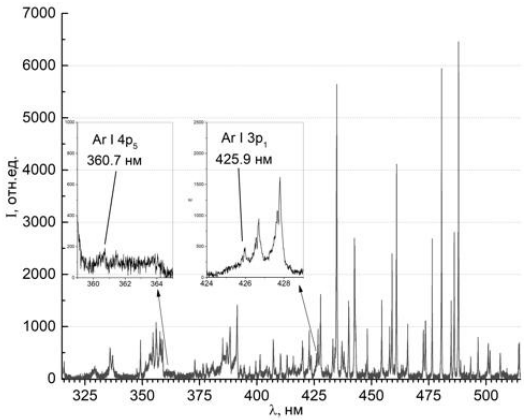


Figure 2. Typical EOS spectrum with marked lines for density analysis

3. Results and discussion

In the main operating modes of the installation, measurements of the plasma flow parameters were carried out in the area of its transportation and in the area of landing on the substrate. Typical results are shown in Fig. 2. The optical emission spectrometry (OES) gives an estimate (Fig. 2) of the electron concentration, relative to the line intensities of 360.7nm and 425.9nm [13–15]. The average concentration of plasma at 1200W of incident microwave power is $\approx 5 \cdot 10^{10} cm^{-3}$.

Based on the results of measurements with a retarding field analyzer probe (Fig. 3).

An estimate of the ion concentration density in the flow in the area of the sample location was obtained $\approx 2 \cdot 10^{9} cm^{-3}$ [16–19].

The treatment was carried out in one of the discharge modes (working gas mixture Ar–O_2, gas pressure $\approx 5 \cdot 10^{-4}$ Torr, input microwave power 1200 W, electron temperature 5 eV, electron concentration $\approx 5 \cdot 10^{10}$ cm^{-3}, current density of the ionic component of the plasma flux in the irradiation region of chitosan samples ≈ 70 $\mu A/cm^2$). Constancy of the treatment mode is maintained by registration of the discharge emission spectrum and stability of operating parameters. Film 6 acted as a control sample, was placed in a vacuum chamber but not treated with plasma, films 1,

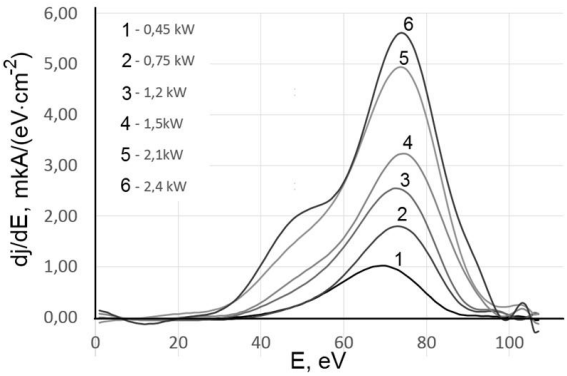


Figure 3. Energy distribution of ion current density in the sample location region with variations in microwave power

Table 3

Qualitative functional analysis of IR spectra of the films

Functional group	$\omega_{C\text{-}H}$	cycl. C-O-C cycl. -OH	$\nu_{C\text{-}O}$	ω_{CH2}	$\delta_{O\text{-}H}$	$\delta_{C\text{-}H}$	$\delta_{N\text{-}H}$	$\nu_{C=O}$	ν_{CH}	ν_{OH} and ν_{NH}
Characteristic frequency, cm^{-1}	893	985, 1024, 1059	1150	1320	1375	1419, 1454	1542, 1589	1649	2868, 2914	3291, 3352

2, 3, 4 and 5 were placed in a vacuum chamber and treated with plasma for 10, 35, 70, 105, 140 s, respectively.

The IR spectra of films treated with plasma for different periods of time (1–5) and the film that was in the RAPIRA apparatus but not exposed to plasma (control film 6) are shown in Fig. 4. Table 3 demonstrates a detailed qualitative functional analysis of the observed signals in the spectra, i. e., the correspondence of characteristic frequencies to certain functional groups.

The IR spectra of the obtained films (Fig. 4) are almost identical, indicating the presence of identical functional groups on the surface of the prepared chitosan-based films. However, the IR spectra of the films show some differences in the wavenumber range of 1300–625 cm^{-1}, that is, in the so-called fingerprint region, which is a strictly individual characteristic of the compound.

Thus, the differences in the fingerprint area do not indicate that all the films are slightly different in their surface characteristics [20, 21]. This is not surprising, as the films were exposed to plasma for different durations. This superficial modification is certainly minor, but nevertheless such changes can result in a pronounced change in the physicochemical characteristics of the film surface, hydrophilic-hydrophobic properties, swelling, adhesion capacity and, consequently, biological properties.

The most important mechanical properties of films — strength and ductility — are quantitatively described by tensile strength and elongation at break, respectively. In this study, we evaluated the

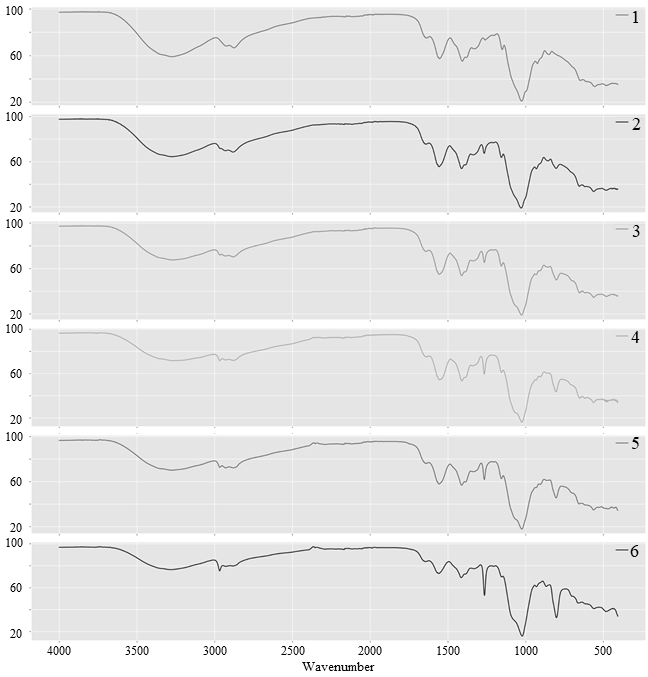


Figure 4. IR spectra of the prepared chitosan-based films

Table 4

Effect of argon-oxygen plasma treatment on the mechanical characteristics of chitosan films

Film	Tensile strength, N/mm^2	Elongation at break, %
1	24.3	36.0
2	26.1	37.4
3	27.2	37.3
4	26.3	38.5
5	24.0	37.1
6	19.9	33.1

effect of argon-oxygen plasma treatment on the mechanical characteristics of chitosan films (Table 4). Plasma treatment affects film strength. All treated films (1–5) were slightly stronger than the film that did not receive direct plasma exposure (6). The maximum increase in strength, 35%, was observed for film 3. Plasma treatment also increased film ductility, but this effect was significantly less pronounced. The maximum increase in ductility was characteristic of film 4 and amounted to only 15%.

The results of the solubility study of the processed films are presented in Fig. 5. All films are characterized by the tendency to dissolve partially in distilled water within 60 min. The solubility

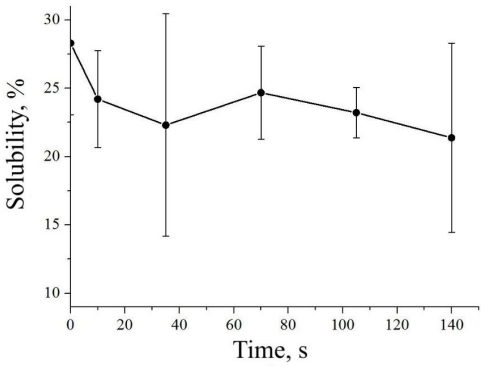


Figure 5. Solubility of films in distilled water at 25°C

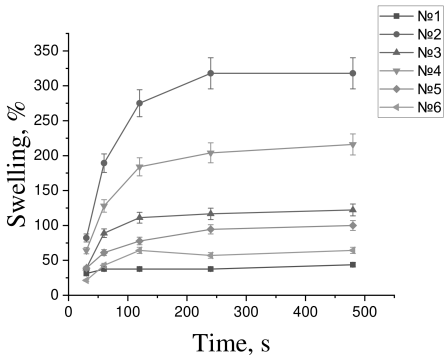


Figure 6. Swelling of the films

of all films 1–5 for the specified time interval practically does not differ from the solubility of the control film 6 and it is in the range of 23–31%.

Fig. 6 exhibits the results of swelling studies of the prepared films. All films exhibit limited swelling. Film 2 swells the most (more than 3 times), while films 1 and 6 swell the least (approximately 50%). The presented film swelling results are consistent with those obtained in their solubility studies (partial solubility is typical of films with limited swelling).

It should be noted that this behavior of films treated with a microwave discharge of the specified mixture differs from their properties when processed in pure argon, where there is an increase in swelling in the specified range of treatment exposures up to 2500%.

This fact indicates a limitation of the hydrophilicity of the resulting film, on the one hand, and the ability to retain large amounts of water, on the other.

4. Conclusion

The influence of Ar/O_2-plasma of resonant UHF discharge parameters on the change of physicochemical properties of thin chitosan films is determined. Plasma treatment leads to some chemical changes on the surface of the films, which is expressed in differences in the IR spectra in the

"fingerprint region". Initial experiments have shown the presence of deviations in the IR properties of the model films. Moreover, plasma discharges of an argon-oxygen mixture processing significantly change the hydrophilicity of chitosan films; improve their solubility and mechanical properties. Further spectroscopic and structural-chemical studies are needed to identify these changes. Further research is aimed at finding plasma generation modes under both microwave and HF discharge conditions and creating a plasma flow with optimal parameters that ensures a given variability in the physico-chemical properties of films.

Author Contributions: The contributions of the authors are equal. All authors have read and agreed to the published version of the manuscript.

Funding: The research was carried out within the State Assignment of Ministry of Science and Higher Education of the Russian Federation (theme No. FSSF-2025-0001).

Data Availability Statement: Data sharing is not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Declaration on Generative AI: The authors have not employed any Generative AI tools.

References

1. Inagaki, N., Narushim, K., Tuchida, N. & K., M. Surface characterization of plasma-modified poly(ethylene terephthalate) film surfaces. *Journal of Polymer Science Part B: Polymer Physics* **42,** 3724–3740. doi:10.1002/polb.20234 (20 2004).
2. Vesel, A. Deposition of Chitosan on Plasma-Treated Polymers-A Review. *Polymers* **15,** 1109. doi:10.3390/polym15051109 (5 2023).
3. Silva, S. S., Luna, S. M., Gomes, M. E., Benesch, J., Pashkuleva, I., Mano, J. F. & Reis, R. L. Plasma surface modification of chitosan membranes: characterization and preliminary cell response studies. *Macromol Biosci* **8,** 568–576. doi:10.1002/mabi.200700264 (6 June 2008).
4. Elashrya, S., ELsaeed, H. & El-Siragy, N. M. Microwave plasma discharge-assisted surface modification of PVA films: coatings and food packaging. *European Physical Journal Plus* **137,** 1–10. doi:10.1140/epjp/s13360-022-03443-7. eprint: https://doi.org/10.1063/1.4986009 (Nov. 2022).
5. Ogino, A., Kral, M., Yamashita, M. & Nagatsu, M. Effects of low-temperature surface-wave plasma treatment with various gases on surface modification of chitosan. *Applied Surface Science* **255,** 2347–2352. doi:10.1016/j.apsusc.2008.07.119 (5 Dec. 2008).
6. Pochelon, A. Electron Cyclotron Resonance Ion Sources and ECR Plasmas. *Plasma Physics and Controlled Fusion - PLASMA PHYS CONTROL FUSION* **39,** 008. doi:10.1088/0741-3335/39/6/008 (June 1997).
7. Gammino, S., Celona, L., Ciavola, G., Maimone, F. & Mascali, D. Review on high current 2.45 GHz electron cyclotron resonance sources (invited). *The Review of scientific instruments* **81,** 02B313. doi:10.1063/1.3266145 (Feb. 2010).
8. Berezin, A., Ishmetova, R., Rusinov, G. & Skorik, Y. Tetrazole derivatives of chitosan: Synthetic approaches and evaluation of toxicity. *Russian Chemical Bulletin* **63,** 1624–1632. doi:10.1007/s11172-014-0645-0 (July 2015).
9. Kritchenkov, A., Egorov, A., Krytchankou, I., Dubashynskaya, N., Volkova, O., Shakola, T., Kurliuk, A. & Skorik, Y. Synthesis of novel 1H-tetrazole derivatives of chitosan via metal-catalyzed 1,3-dipolar cycloaddition. Catalytic and antibacterial properties of [3-(1H-tetrazole-5-yl)ethyl]chitosan and its nanoparticles. *International Journal of Biological Macromolecules* **132,** 340–350. doi:10.1016/j.ijbiomac.2019.03.153 (Mar. 2019).

10. Boudouaia, N., Amine Ahmed, B., Mahmoudi, H., Saffaj, T. & Bengharez, Z. Swelling and adsorption properties of cross-linked chitosan-based films: kinetic, thermodynamic and optimization studies. *DESALINATION AND WATER TREATMENT* **255,** 56–67. doi:10.5004/dwt.2022.28321 (Apr. 2022).
11. Lewandowska, K. & Szulc, M. Rheological and Film-Forming Properties of Chitosan Composites. *International Journal of Molecular Sciences* **23,** 8763. doi:10.3390/ijms23158763 (Aug. 2022).
12. Baker, M. *et al.* Using Fourier transform IR spectroscopy to analyze biological materials. *Nature protocols* **9,** 1771–1791. doi:10.1038/nprot.2014.110 (Aug. 2014).
13. Zhu, X.-M. & Pu, Y.-K. A simple collisional–radiative model for low-pressure argon discharges. *Journal of Physics D: Applied Physics* **40,** 2533. doi:10.1088/0022-3727/40/8/018 (Apr. 2007).
14. Iordanova, S. & Koleva, I. Optical emission spectroscopy diagnostics of inductively-driven plasmas in argon gas at low pressures. *Spectrochimica Acta Part B: Atomic Spectroscopy* **62,** 344–356. doi:10.1016/j.sab.2007.03.026 (Apr. 2007).
15. Wang, H., Zhang, Z., Yang, Y., Tan, C., Cui, R. & Ouyang, J. Axial profiles of argon helicon plasma by optical emission spectroscope and Langmuir probe. *Plasma Science and Technology* **21,** 009. doi:10.1088/2058-6272/ab175b (Apr. 2019).
16. Baloniak, T., Reuter, R., Floetgen, C. & Keudell, A. Calibration of a miniaturized retarding field analyzer for low-temperature plasmas: Geometrical transparency and collisional effects. *Journal of Physics D-applied Physics - J PHYS-D-APPL PHYS* **43,** 203. doi:10.1088/0022-3727/43/5/055203 (Feb. 2010).
17. Pushkarev, A. I. & Isakova, Y. *Diagnostics of high-power ion beams: — A monograph* in (Novosibirsk: Publishing house of ANS SibAK, 2016).
18. Afanasyev, V. P. & Yavor, S. *Electrostatic energy analyzers for charged particle beams* in (M.: Nauka Publ., 1978).
19. Farnell, C., Farnell, S. & Williams, J. Recommended Practice for Use of Electrostatic Analyzers in Electric Propulsion Testing. *Journal of Propulsion and Power* **33,** 638–658. doi:10.2514/1.B35413 (May 2017).
20. Dimzon, I. K. & Knepper, T. Degree of Deacetylation of Chitosan by Infrared Spectroscopy and Partial Least Squares. *International journal of biological macromolecules* **72,** 939–935. doi:10.1016/j.ijbiomac.2014.09.050 (Oct. 2014).
21. Boukhlifi, F. *Quantitative Analysis by IR: Determination of Chitin/Chitosan DD* in (Mar. 2020). doi:10.5772/intechopen.89708.

Information about the authors

Artemyev, Andrey V.—Postgraduate Student at the Institute of Physical Research and Technology, Junior Researcher at the Institute of Physical Research and Technology of RUDN University (e-mail: artemyev_anvl@pfur.ru, ORCID: 0009-0000-1925-4711)

Kritchenkov, Andrey S.—Doctor of Chemical Sciences, Associate Professor, Department of Human Ecology and Bioelementology of RUDN University (e-mail: kritchenkov_as@pfur.ru, ORCID: 0000-0002-6411-5988)

УДК 533.924
PACS 52.77.-j, 52.80.Pi,
DOI: 10.22363/2658-4670-2025-33-4-461-470 EDN: HMKEYN

Ar-O_2 плазма резонансного СВЧ-разряда для обработки хитозановых плёнок

А. В. Артемьев, А. С. Критченков

Российский университет дружбы народов, ул. Миклухо-Маклая, д. 6, Москва, 117198, Российская Федерация

Аннотация. Статья посвящена исследованию модификации хитозановых пленок при обработке Ar-O_2 плазмой СВЧ-разряда. Основная идея заключается в использовании резонансных методов создания плазмы для обработки хитозановых плёнок. Получены спектральные и энергетические характеристики плазмы СВЧ-разряда. Исследованы механические свойства, набухание и растворимость хитозановых плёнок под воздействием плазмы СВЧ-разряда. Показана зависимость свойств плёнок от времени обработки плазмой резонансного СВЧ-разряда.

Ключевые слова: плазма, резонанс, СВЧ-разряд, микроволновая плазма, хитозан, модификация плёнок, растворимость, набухание, экспериментальные исследования