RESEARCH-ARTICLE

Exploring LLM-Driven Explanations for Quantum Algorithms

GIORDANO D'ALOISIO, University of L'Aquila, L'Aquila, AQ, Italy

SOPHIE FORTZ, King's College London, London, U.K.

CAROL HANNA, University College London, London, U.K.

DANIEL FORTUNATO, University of Porto, Porto, Porto, Portugal

AVNER BENSOUSSAN, King's College London, London, U.K.

EÑAUT MENDILUZE USANDIZAGA, Simula Research Laboratory, Barum, Akershus, Norway

View all

Published: 24 October 2024

ESEM '24: ACM / IEEE International Symposium on Empirical Software Engineering and Measurement
October 24 - 25, 2024
Barcelona, Spain

Conference Sponsors:
SIGSOFT

https://doi.org/10.1145/3674805.3690753
ISBN: 9798400710476

Exploring LLM-Driven Explanations for Quantum Algorithms

Giordano d'Aloisio
University of L'Aquila
L'Aquila, Italy
giordano.daloisio@graduate.univaq.it

Sophie Fortz
King's College London
London, United Kingdom
sophie.fortz@kcl.ac.uk

Carol Hanna
University College London
London, United Kingdom
carol.hanna.21@ucl.ac.uk

Daniel Fortunato
University of Porto
Porto, Portugal
dabf@fe.up.pt

Avner Bensoussan
King's College London
London, United Kingdom
avner.bensoussan@kcl.ac.uk

Eñaut Mendiluze Usandizaga
Simula Research Laboratory
Oslo, Norway
enaut@simula.no

Federica Sarro
University College London
London, United Kingdom
f.sarro@ucl.ac.uk

Abstract

Background: Quantum computing is a rapidly growing new programming paradigm that brings significant changes to the design and implementation of algorithms. Understanding quantum algorithms requires knowledge of physics and mathematics, which can be challenging for software developers. **Aims:** In this work, we provide a first analysis of how LLMs can support developers' understanding of quantum code. **Method:** We empirically analyse and compare the quality of explanations provided by three widely adopted LLMs (Gpt3.5, Llama2, and Tinyllama) using two different human-written prompt styles for seven state-of-the-art quantum algorithms. We also analyse how consistent LLM explanations are over multiple rounds and how LLMs can improve existing descriptions of quantum algorithms. **Results:** Llama2 provides the highest quality explanations from scratch, while Gpt3.5 emerged as the LLM best suited to improve existing explanations. In addition, we show that adding a small amount of context to the prompt significantly improves the quality of explanations. Finally, we observe how explanations are qualitatively and syntactically consistent over multiple rounds. **Conclusions:** This work highlights promising results, and opens challenges for future research in the field of LLMs for quantum code explanation. Future work includes refining the methods through prompt optimisation and parsing of quantum code explanations, as well as carrying out a systematic assessment of the quality of explanations.

CCS Concepts

• **Computer systems organization** → *Quantum computing*; • **Computing methodologies** → *Artificial intelligence.*

ESEM '24, October 24–25, 2024, Barcelona, Spain

ACM ISBN 979-8-4007-1047-6/24/10
https://doi.org/10.1145/3674805.3690753

Keywords

Quantum Computing, Large Language Models, Code Explainability.

ACM Reference Format:
Giordano d'Aloisio, Sophie Fortz, Carol Hanna, Daniel Fortunato, Avner Bensoussan, Eñaut Mendiluze Usandizaga, and Federica Sarro. 2024. Exploring LLM-Driven Explanations for Quantum Algorithms. In *Proceedings of the 18th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '24), October 24–25, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3674805.3690753

1 Introduction

Quantum computing is a multi-disciplinary field comprising computer science, physics, and mathematics, which utilises quantum mechanics to solve specific problems much faster than a classical computer would do [17, 36]. Designing and implementing quantum algorithms fundamentally differs from doing so for classical algorithms. Developing quantum algorithms requires skills in mathematics and physics that software developers lack, which creates an entry barrier for most developers interested in quantum computing.

Lowering this entry barrier should be a priority for the continuously growing quantum software engineering community [46]. We hypothesise that using large language models (LLMs) to generate high-quality explanations of quantum programs can lower the entry barrier for researchers or developers interested in quantum software engineering. Thus, in this work, we investigate LLMs's ability to automatically generate explanations for quantum programs. Specifically, we carry out an empirical study to investigate the ability of three widely adopted LLMs (i.e., Gpt3.5, Llama2, and Tinyllama) to generate explanations for seven state-of-the-art quantum algorithms written in the OpenQSAM programming language [7] when no context is given to the LLMs. In addition, we investigate whether adding a small amount of context to the prompt improves the quality of explanations, and how consistent the generated explanations are over multiple rounds of generation. Finally, we analyse the ability of LLMs to improve existing descriptions of quantum algorithms. All the explanations generated were manually rated by four human-raters with different expertise in software engineering and different degree of familiarity with quantum computing.

According to the human-raters, Llama2 is the LLM providing the best explanations when no context is provided, while Gpt3.5 is better suited to improve already existing explanations. Furthermore, we observe that providing the LLMs with a small amount of context (i.e., just the name of the algorithm implemented and the number of qubits employed) significantly improves the quality of explanations for all the LLMs analysed herein. Finally, we found no difference (both qualitative and syntactical) in explanations generated over multiple rounds, thus proving a good level of stability of the LLMs towards this task.

The promising results obtained in this preliminary analysis encourage further research on the use of LLMs' explanations of quantum code, including exploring prompt optimisation and conducting a larger human-based evaluation.

The contributions of our work are as follows:

- To the best of our knowledge, this is the first work providing insights on LLMs' ability to explain quantum code;
- We detail future research directions in assessing and improving quantum code explanations;
- We provide the full replication package of our work, including the quantum algorithms employed and the scripts to generate explanations [9].

2 Background and Motivation

Quantum programming is an emergent discipline born from recent advancements in quantum mechanics. The era of quantum computing is still in its infancy, marked by the development of the first quantum computers. Despite its early stage, software engineers are increasingly recognising the potential of quantum computing, as evidenced by the surge in quantum-related papers presented at recent software engineering conferences [1, 2, 5, 10, 11, 22, 23, 28, 29]. However, the high entry barrier remains a significant obstacle to popularising the field.

Fundamentally, a quantum program is a circuit where each wire represents a bit or a qubit and employs two types of operators: quantum gates and measurement operators. Quantum gates, usually defined as matrices, perform operations on one or more qubits, altering their states. Measurement operators, on the other hand, extract the result of the quantum circuit and transform it into classical bits that can be interpreted by conventional computers.

Physicists have provided tools to manipulate quantum information, such as languages like OpenQASM. From a physicist's perspective, these tools fulfil the requirement, as software engineers no longer need to manipulate physical qubits directly but can use abstract representations consisting of logical qubits and gates. However, from a software engineer's viewpoint, these languages may seem overly complex, resembling writing code solely with logical circuits.

Despite these advancements, the field of quantum programming still faces significant challenges, particularly in making the technology accessible to a broader range of software engineers. Bridging this gap requires not only the development of more intuitive programming languages and tools but also employing comprehensive resources to lower the entry barrier and foster wider adoption and innovation in quantum computing.

This paper centres on the latter idea: employing resources to diminish the entry barrier and catalyse wider adoption and innovation in quantum computing. Since the release of ChatGPT in November 2022, large language models (LLMs) have been assisting developers and software engineers in several tasks. Among those, code comprehension emerged as one of the tasks in which LLMs have been mostly involved [15]. Hence, we believe that LLMs can be a useful resource to help developers and software engineers understand quantum code. However, as mentioned above, quantum code can not be considered at the same level as traditional code, still being more assembly-level style. Therefore, systematically assessing the ability of LLMs to provide specific explanations for quantum algorithms is still a challenge that has to be addressed. This paper provides a first step in this direction: we engage three LLMs to provide code explanations for seven quantum algorithms using prompts providing different amounts of context and ask human raters to assess each explanation for correctness and potential comprehension enhancement.

3 Related Work

LLMs have been used in previous work to explain different aspects of code. For instance, Sarsa *et al.* [35] employ LLMs to generate code explanations for educational purposes. Also focusing on education, MacNeil *et al.* [26] analyse how LLMs are able to explain numerous aspects of a given code snippet and Sobania *et al.* [37] empirically analyse the ability of LLMs to explain software patches. Balfroid *et al.* [4] use ChatGPT to provide code tours for code onboarding *i.e.,* the process of transitioning new employees into a team's methodology and tools. Leinonen *et al.* [21] use LLMs to improve programming error messages. Concerning the integration between LLMs and quantum code, Easttom [13] analyse the ability of ChatGPT to generate and improve quantum algorithms. Guo *et al.* [18] propose to use ChatGPT for automated quantum program repair, while Ezratty highlights how LLMs can be employed in different aspects of Quantum Computing, such as learning, software development or research [14]. However, to the best of our knowledge, no work has yet attempted to analyse the ability of LLMs to explain quantum code. Our work seeks to fill this gap providing first results and shepherding the way for future research in this field.

4 Methodology

In this section, we describe the methodology followed for our evaluation. We aim to answer the following research questions (**RQs**):
➤ **RQ$_1$:** *Do different LLMs generate quantum algorithm explanation of different quality?* This RQ focuses on assessing the overall quality of the explanations generated by the LLMs and whether there is a model that is more suited for explaining quantum code.
➤ **RQ$_2$:** *To what extend does adding context to the prompt impact the explanation quality?* This RQ investigates whether providing an LLM with a prompt including some context (in our case, only the name of the algorithm and the number of qubits are provided) improves the overall quality of explanations.
➤ **RQ$_3$:** *How consistent are the LLM explanations over different runs?* Given the stochastic nature of LLMs [33], this RQ aims to assess

if running a prompt multiple times leads to both qualitatively and syntactically similar explanations.

➤ **RQ$_4$:** *Do different LLMs exhibit a different ability in improving existing explanations for quantum code?* Instead of explaining a quantum algorithm from scratch, this RQ focuses on assessing how able an LLM is to improve existing descriptions.

In the following, we first present the quantum algorithms employed in our evaluation. Then, we describe the process followed to generate explanations from the LLMs. Finally, we report our the evaluation process we followed to answer each RQ.

4.1 Evaluation Benchmark

For our evaluation, we employ seven quantum algorithms selected from the MqtBench [31] benchmark written in the OpenQASM 3 programming language [8] for the Qiskit quantum compiler. The selected algorithms are the following: *Amplitude Estimation (AE)*: this algorithm aims to find an estimation for the amplitude of a certain quantum state [39]; *Deutsch-Jozsa (DJ)*: this algorithm determines, whether an unknown oracle mapping input values either to 0 or 1 is constant (always output 1 or always 0) or balanced (both outputs are equally likely) [6]; *Grover*: Grover's algorithm finds a certain goal quantum state determined by an oracle [24]; *Quantum Fourier Transform (QFT)*: this algorithm embodies the quantum equivalent of the discrete Fourier transformation [42]; *Quantum Fourier Transform with entanglement (QFT-ent)*: this algorithm is a variation of QFT to entangled qubits; *Quantum Phase Estimation (QPE)*: this algorithm estimates the phase of a quantum operation [12]; *Quantum Walk (QW)*: the quantum equivalent to classical random walks [40]. The rationale for the selection of these algorithms is twofold; they are among the most popular quantum algorithms, and they are used in other empirical studies [16]. Moreover, we adopted the OpenQASM implementation of these algorithms instead of their Python ones to better assess the ability of LLMs to explain algorithms written in less common programming languages.

4.2 Generation of Code Explanations

We analyse explanations generated by using three different LLMs, namely: Gpt3.5 turbo 16k, Llama2, and Tinyllama. We have chosen Gpt3.5 and Llama2 because they are two of the most widely adopted general-purpose LLMs. Moreover, we also analyse explanations generated from Tinyllama to check if a smaller LLM is also able to generate good-quality quantum code explanations. Concerning Llama and Tinyllama, we adopt the implementations provided by the `ollama` ecosystem with their default hyper-parameters,[1] while for Gpt3.5 we employ the model provided by the OpenAI API[2] and, following a previous work about quality assessment of code explanations from Gpt3.5 [37], we set its `temperature` to 0.8.

To generate the code explanations for **RQ$_1$, RQ$_2$, and RQ$_3$**, we feed each LLM with two different prompt styles: the first style, which we call *Non Context-Aware*, does not provide any information about the quantum code we feed as input:

"Can you give a high-level explanation of this code? <algorithm code>"

The second style, which we call *Context-Aware*, includes a few basic information about the quantum code feed as input, i.e., the name of the algorithm implemented by the quantum code and the number of qubits (this information was obtained from the MqtBench benchmark):

"Can you give a high-level explanation of this code? <algorithm code>. *The name of the algorithm is:* <algorithm name>. *The code includes* <number of qubits> *qubits"*

The structure of the above prompts has been extensively discussed and agreed by all the authors of the paper. Note that we ask the LLM to provide *high-level* explanations for both prompts to assess better the LLM's ability to understand an algorithm's behaviour instead of just describing each line of code.

Finally, to address the consistency of the explanations, following the methodology employed in previous work [37], we repeated the generation process three times for each LLM (3), quantum algorithm (7), and prompt style (2) combination, yielding a total of 126 different explanations.

Concerning **RQ$_4$**, we extract from the MqtBench repository a short description for each quantum algorithm and ask the LLMs to improve it using the following prompt:

"Can you improve the following explanation: <code explanation> *for the following quantum algorithm:* <algorithm code> *named*: <algorithm name> *making it more informative but also keeping it simple?"*

Like the previous case, this prompt has been discussed and approved by all the authors. In this context, however, we perform only one round of generation due to limited computational resources and human-effort needed to rate multiple explanations. Future works can investigate the consistency of description improvements over multiple rounds of generation.

4.3 Evaluation Process and Metrics

After collecting the different explanations, four independent respondents evaluate them. These are software engineers (SE) working in quantum computing. We employed SE with knowledge of quantum computing because they can provide a more reliable evaluation of quantum explanations and are more able to detect errors in the explanations compared with people with no expertise on the topic. In particular, two of the evaluators reported 3-4 years of experience in SE, while the other two have 5-10 years. As for the experience in quantum computing, one respondent reported less than a year of experience, two have 1-2 years, and one has 3-4 years, highlighting a heterogeneous group of raters.

We asked them to give a score from 1 to 5 for each explanation for each quantum algorithm, where 1 means an entirely wrong explanation, while 5 indicates a high-quality explanation. To help the evaluators assess the correctness and quality of each explanation, we provided them with the source code of each algorithm and the one-line description of the algorithm, as given by MqtBench [31].

To avoid potential bias during the human evaluation, we anonymise the LLM and prompt style used to generate the explanations for each quantum algorithm. Hence, concerning **RQ$_1$**, **RQ$_2$**, and **RQ$_3$**, for each quantum algorithm, the human-raters had to evaluate 18 different explanations (i.e., explanations from 3 LLMs × 2 prompts styles × 3 runs) without knowing the LLM and prompt style that

[1] https://ollama.com/
[2] https://openai.com/index/openai-api/

generated them. For the $\mathbf{RQ}_4$, the raters had to evaluate three explanations (one for each LLM) for each quantum algorithm.

After collecting the evaluations, we performed the following steps to answer each RQ. First, for each explanation, we compute the mean among the scores given by each evaluator. Next, to answer the $\mathbf{RQ}_1$, we compare the mean and median scores of all explanations generated from each LLM. In addition, we perform the non-parametric *Wilcoxon signed-ranked* test [45] to assess if there is a statistically significant difference among the scores of each LLM. We adopt this test because the experiment follows a *one factor with two treatments paired comparison* design (i.e., all multiple subjects evaluate all different treatments) [44]. Moreover, we perform this test instead of the parametric *Paired t* test [19] because the data does not follow a normal distribution, as confirmed by the *Shapiro-Wilk* test for normality [32]. This same evaluation process has also been performed to evaluate LLM improved explanations to assess the $\mathbf{RQ}_4$.

To answer the $\mathbf{RQ}_2$, we compare the mean and median scores of all explanations generated by each LLM using the *Non Context-aware* and *Context-aware* prompt styles. As done for the $\mathbf{RQ}_1$, we employ the *Wilcoxon signed-ranked* test to assess if, for each LLM, there is a statistically significant difference in the scores of explanations obtained using *Context-aware* and *Non context-aware* styles.

To answer the $\mathbf{RQ}_3$, we evaluate both the qualitative and syntactical similarity of explanations generated by each LLM over multiple rounds. To assess the qualitative similarity, we compare the mean and median scores of explanations obtained by an LLM with a given prompt style over multiple rounds. We also employ the non-parametric *Kruscal-Wallis H* test [27] to assess if there is a statistically significant difference in the scores obtained over multiple rounds. Again, we adopt this test instead of the parametric *ANOVA* [38] since the data does not follow a normal distribution. To assess the syntactical similarity, we compute the *cosine similarity* among the explanations generated for the same algorithm over multiple rounds of the same LLM with a specific prompt style. *Cosine similarity* is a widely adopted metric in the Natural Language Processing and Information Retrieval domains, which measures the similarity of two documents as the cosine of their term-frequency vector representation [34]. This metric ranges from 0 to 1, where 0 means that two documents are entirely different, while 1 means that two documents are syntactically identical. Since we repeated the generation process three times, we report the mean cosine similarity among the three documents.

Following standard practices [3, 44], we consider a statistical test significant if its p-value is < 0.05. However, since we perform multiple hypothesis testing for each RQ (3 tests for $\mathbf{RQ}_1$, $\mathbf{RQ}_2$ and $\mathbf{RQ}_4$ and 6 tests for $\mathbf{RQ}_3$), we apply the *Bonferroni* correction [43] and consider a test significant if its p-value is $< 0.05/3$ for $\mathbf{RQ}_1$, $\mathbf{RQ}_2$ and $\mathbf{RQ}_4$ and $< 0.05/6$ for $\mathbf{RQ}_3$. In addition, following the standard guidelines in [41], we support the obtained p-values for $\mathbf{RQ}_1$, $\mathbf{RQ}_2$ and $\mathbf{RQ}_4$ with the *Cliff's* δ effect size to assess the difference magnitude [25].

To assess the reliability of human evaluation scores, we use the *Krippendorff's* α inter-rater agreement metric, defined as:

$$\alpha = \frac{p_a - p_e}{1 - p_e}$$

Table 1: Mean and median explanation scores for each LLM.

	Gpt3.5	Llama2	Tinyllama
Mean	2.50	**2.78**	1.87
Median	2.50	**2.75**	1.86

Table 2: Statistic and Bonferroni corrected p-value of the Wilcoxon test and Cliff's δ among LLM explanation scores.

	W Stat	p-value	δ
Gpt3.5 - Llama2	141.50	0.02	-0.18
Gpt3.5 - Tinyllama	53.50	$3.07 * 10^{-5}$	0.44
Llama2 - Tinyllama	38.00	$1.01 * 10^{-6}$	0.52

where p_a represents the observed weighted percentage agreement (i.e., how often the reviewers actually agreed) and p_e represents the chance weighted percentage agreement (i.e., the percentage agreement the raters would achieve with random scores) [20]. This metric ranges from -1 to 1, where -1 indicates a systematic disagreement, 0 means random guessing, and 1 means total agreement among the evaluators. We adopted this metric despite other widely adopted inter-rater agreement metrics like *Cohen's* κ because it enables the comparison of more than two raters and handles *interval* ratings (i.e., Likert scale evaluations).

5 Results

In the following, we report the results for our RQs. Concerning the reliability of the human evaluations, the *Krippendorff's* α reported an inter-rater agreement of 0.47 for the evaluation conducted to answer $\mathbf{RQ}_1$, $\mathbf{RQ}_2$, $\mathbf{RQ}_3$, and a result of 0.9 in the evaluation conducted to answer $\mathbf{RQ}_4$, highlighting an overall positive agreement among the respondents.

5.1 $\mathbf{RQ}_1$: LLM-Driven Quantum Explanations

Table 1 reports the mean and median scores of explanations obtained by each LLM, while Table 2 reports the statistics and the Bonferroni corrected p-value of the *Wilcoxon* test and the *Cliff's* δ between each LLM pair. It can be observed from Table 1 how Llama2 is the LLM obtaining the highest evaluation score with a mean of 2.78 and a median of 2.75. Gpt3.5 reports instead an average evaluation score of 2.5, which is close to Llama2, as confirmed by a small effect size but still statistically significant, as shown in Table 2. Concerning Tinyllama, we observe instead a lower mean (1.87) and median (1.86) scores compared with both Gpt3.5 and Llama2. This difference is also confirmed by a lower p-value of the *Wilcoxon test* and a greater effect size.

Figure 1 reports the distribution of explanation scores for each LLM. While we observe a quite similar distribution of scores between Gpt3.5 and Llama2, Tinyllama reports a significantly higher amount of 1.0 scores compared with the other LLMs, meaning that many explanations from Tinyllama were systematically wrong.

Answer to $\mathbf{RQ}_1$: Llama2 emerged as the LLM that provides better explanations. On the other side, Tinyllama provides many

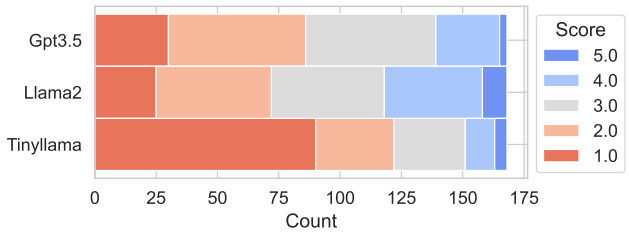

Figure 1: Distribution of explanation scores for each LLM

Table 3: Mean and median values, Wilcoxon test and Cliff's δ between scores of explanations obtained with *Context-aware* and *Non context-aware* prompt styles

	Non context-aware		Context-aware		W Stat	p-value	δ
	Mean	Median	Mean	Median			
Gpt3.5	1.86	1.75	**3.14**	**3.25**	0.00	$2.86 * 10^{-6}$	-0.98
Llama2	2.02	2.00	**3.54**	**3.75**	2.00	$8.58 * 10^{-6}$	-0.89
Tinyllama	1.36	1.25	**2.38**	**2.25**	1.50	$3.21 * 10^{-4}$	-0.77

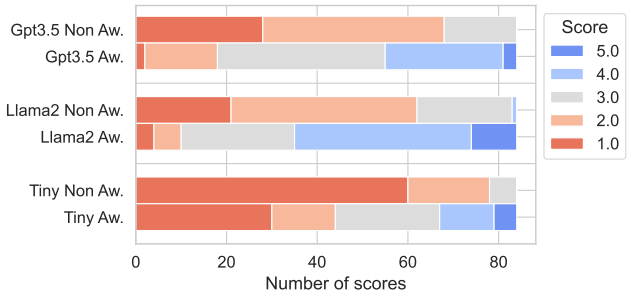

Figure 2: Score distribution for each LLM and prompt style.

explanations that are systematically wrong compared with the other LLMs.

5.2 RQ$_2$: Context-Aware Prompts

Table 3 reports the mean and median values and the results of *Wilcoxon* test and *Cliff's* δ between the scores of explanations obtained with *Non context-aware* and *Context-aware* prompt styles. From the table, we observe how adding basic information to the prompt (i.e., the name of the algorithm implemented and the number of qubits employed) significantly improves the quality of the explanations in all LLMs.

The higher quality of explanations obtained from a *Context-aware* prompt style is also confirmed by the distribution of scores shown in Figure 2. From the figure, we observe how the proportion of explanations with a low score (i.e., 1 or 2) significantly decreases, especially in Gpt3.5 and Llama2.

Answer to RQ$_2$: Adding basic information like the name of the algorithm implemented or the number of qubits employed significantly improves the quality of explanations for all the LLMs employed.

Table 4: Mean and median values and Kruscall-Wallis H test result between scores of explanations obtained over three generation rounds for each LLM.

		Non context-aware				Context-aware			
		Mean	Median	H Stat	p-value	Mean	Median	H Stat	p-value
Gpt3.5	Round 1	2.04	2.00	3.56	1.01	3.32	3.50	2.48	1.74
	Round 2	1.71	1.75			2.93	2.75		
	Round 3	1.82	1.75			3.18	3.25		
Llama2	Round 1	2.07	2.00	1.33	3.08	3.43	3.75	5.05	0.48
	Round 2	2.07	2.00			3.96	4.00		
	Round 3	1.93	2.00			3.21	3.25		
Tinyllama	Round 1	1.43	1.25	1.44	2.91	2.36	2.25	0.28	5.21
	Round 2	1.39	1.25			2.32	2.75		
	Round 3	1.25	1.00			2.46	2.75		

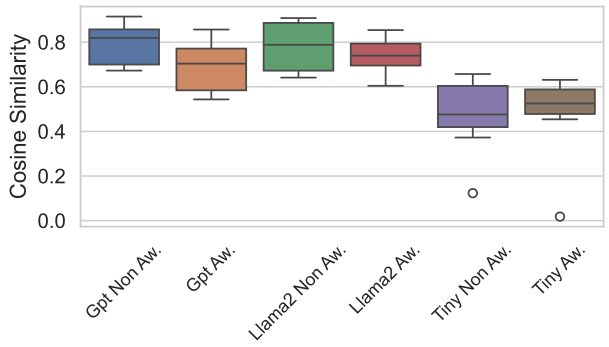

Figure 3: Distribution of cosine similarity of explanations obtained by each LLM with specific prompt style over three generation rounds.

5.3 RQ$_3$: Explanation Consistency

We assess the similarity of explanations from both a qualitative (i.e., if there is a difference in the quality of explanations) and syntactical (i.e., if the explanations are written differently) point of view.

Concerning qualitative similarity, Table 4 reports the mean and median scores and the *Kruskall-Wallis H* test result of explanations obtained from three generation rounds of each LLM with a given prompt style. We observe an overall not significant difference in the scores, especially using a *Non context-aware* prompt style. The difference in scores is also emphasised by an overall high p-value of the *H* test, which does not let us reject the null hypothesis of equal medians among the groups.

Concerning syntactic similarity, Figure 3 reports the distribution of cosine similarity scores among each algorithm's explanation over three generation rounds for each LLM and prompt style pair. We observe how explanations from *Gpt3.5* and *Llama2* have a high similarity score despite the adopted prompt style (with median scores ranging between 0.7 and 0.8). A lower similarity score is instead observed for *Tinyllama*, which can be partially explained by the randomness of some explanations returned by the model.

Answer to RQ$_3$: Explanations returned by the LLMs over different rounds are overall similar from a qualitative point of view. Concerning the syntactical similarity, *Gpt3.5* and *Llama2* return

Table 5: Mean and median scores of improved explanations.

	Gpt3.5	Llama2	Tinyllama
Mean	**3.54**	2.39	1.89
Median	**4.50**	2.25	1.50

similar explanations, while more variability is observed in the explanations from *Tinyllama*.

5.4 RQ_4: Quantum Explanation Completion

Table 5 reports the mean and median scores of LLM improved explanations. We notice that Gpt3.5 performs better then the others. However, the results of the *Wilcoxon* test do not allow us to reject the null hypothesis that the distribution of scores is the same for all LLMs, providing p-values $> 0.05/3$ for all comparisons (i.e., 0.375 in the comparison between Gpt3.5 and Llama2, 0.297 between Gpt3.5 and Tinyllama, and 0.247 between Llama2 and Tinyllama). This result may be due to the small dimension of samples (i.e., 7 scores for each LLM) which may lead us to the Type II statistical error (i.e., accept the null hypothesis when it should be rejected) [44].

Answer to RQ_4: Gpt3.5 emerges as the LLM performing better in improving existing explanations. However, further research should be conducted to better explore the capabilities of LLMs in improving existing quantum explanations.

6 Threats to Validity

Internal Validity: It is worth noting that our panel of experts possesses some background knowledge in quantum programming, which may appear counter-intuitive to our initial hypothesis of analysing the ability of LLMs to explain quantum code to non-expert software engineers. Nevertheless, we argue that error-free explanations hold greater value. Consequently, in this preliminary evaluation, soliciting evaluations from individuals lacking quantum knowledge might bias results toward superficial aspects, potentially leading to erroneous conclusions. Additionally, all our human-raters boast software engineering backgrounds and have recently transitioned into the realm of quantum computing (see demographic description in Section 4). Thus, they acknowledge their ongoing learning curve in this domain and recognise the utility of such a tool, notwithstanding their ability to discern basic errors. This criterion may be relaxed in the future once our methodology matures, enabling us to pre-select accurate explanations for a comprehensive user study. In addition, the quality of each explanation has been evaluated using a single question system, which may not be best suited given the multi-faced definitions of *quality*. We plan to extend our analysis with a more structured evaluation system in future works.

Construct Validity: The quality assessment of each explanation is based on a subjective evaluation, and it may differ among evaluators. To mitigate this threat, we employed the *Krippendorff's* α inter-rater agreement score, which reported an overall positive agreement among the evaluators.

Conclusion Validity: The results of RQ_4 have a low statistical significance. This may be due to the small data sample employed in this RQ, which may lead to a Type II statistical error (i.e., refusing to reject the null hypothesis when it should be rejected).

External Validity: Our analysis focuses on a limited set of LLMs and quantum algorithms, and the results may not hold for other settings. To mitigate this, we consider widely adopted LLMs and state-of-the-art quantum algorithms. In addition, the results of our analysis are limited to algorithms written in the OpenQASM 3 programming language, while the adoption of higher-level programming languages (like Qiskit, Silq, or Q#) may lead to different results.

7 Conclusion and Future Directions

This evaluation has provided valuable insights into the capability of LLMs to explain quantum code, highlighting that both GPT3.5 and Llama2 are suitable for this task. Our findings show that including a small amount of context in the prompt significantly enhances the quality of explanations. Additionally, the explanations generated by the LLMs remain qualitatively and syntactically consistent across multiple iterations. However, further work is required to analyse and improve LLMs' ability to explain quantum code comprehensively. None of the LLMs achieved an average high score (i.e., 4 or higher) in generating explanations from scratch without prior context provided in the prompt. While evaluators generally agreed positively on the quality of the explanations, the inter-agreement α score of 0.47 underscores the need for a more systematic approach to evaluating quantum explanations (e.g., following the guidelines outlined in [30]). Furthermore, despite existing studies on the quality of LLMs' explanations for various aspects of traditional code (see Section 3), special attention should be devoted to quantum code. The use of logical qubits and quantum gates is not intuitive for software engineers, and writing algorithms using logical circuits can be cumbersome. This makes quantum code significantly different from classical code and necessitates a unique approach to explanation, as discussed in Section 2.

Given these observations, several future research directions emerge. Future work could expand the analysis of GPT3.5 and Llama2 explanations by developing more systematic guidelines for assessing the quality of quantum explanations, involving individuals with varying expertise in both software engineering and quantum computing and employing less-known quantum algorithms. Research should also focus on prompt optimisation for quantum code explanation, determining the minimum amount of context required in the prompt to achieve high-quality explanations. The question of determinism also arises, warranting experiments exploring different temperature settings to observe their impact on the results, especially concerning their consistency over multiple rounds. Additionally, exploring more language models (like GPT4 or Llama3) to investigate their efficiency and efficacy will help identify the best-fitting architecture for this task. Finally, an interesting avenue for further optimisation is response parsing. Implementing a parser to structure the explanations provided by the LLMs can significantly enhance the clarity of these explanations for new users.

Acknowledgments

G. d'Aloisio is partially funded by the European Union - NextGenerationEU - National Recovery and Resilience Plan (PNRR) - Project: "SoBigData.it - Strengthening the Italian RI for Social Mining and Big Data Analytics" - Prot. IR0000013 - Adv. n. 3264 of 28/12/2021. S. Fortz is partially supported by the EPSRC project on Verified Simulation for Large Quantum Systems (VSL-Q), grant reference EP/Y005244/1, the EPSRC project on Robust and Reliable Quantum Computing (RoaRQ), Investigation 009 Model-based monitoring and calibration of quantum computations (ModeMCQ), grant reference EP/W032635/1 and by the QAssure project from Innovate UK. E. Mendiluze Usandizaga is supported by Simula's internal strategic project on quantum software engineering. D. Fortunato is suported by FCT through grant ref. 2023.02439.BD. F. Sarro is supported by the ERC Grant no. 741278 and the UCL-CS Strategic Research Fund.

References

[1] Mradul Agrawal, Aviral Jain, Rudraksh Thorat, and Shivam Sharma. 2023. Quantum Computing: A Software Engineering Approach. *Quantum Computing in Cybersecurity* (2023), 233–248.

[2] Shaukat Ali, Tao Yue, and Rui Abreu. 2022. When software engineering meets quantum computing. *Commun. ACM* 65, 4 (2022), 84–88.

[3] Andrea Arcuri and Lionel Briand. 2011. A practical guide for using statistical tests to assess randomized algorithms in software engineering. In *Proceedings of the 33rd International Conference on Software Engineering* (Waikiki, Honolulu, HI, USA) *(ICSE '11)*. Association for Computing Machinery, New York, NY, USA, 1–10. https://doi.org/10.1145/1985793.1985795

[4] Martin Balfroid, Benoît Vanderose, and Xavier Devroey. 2024. Towards LLM-Generated Code Tours for Onboarding. In *Workshop on NL-based Software Engineering (NLBSE'24)*.

[5] Luis S Barbosa. 2020. Software engineering for'quantum advantage'. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*. 427–429.

[6] David Collins, KW Kim, and WC Holton. 1998. Deutsch-Jozsa algorithm as a test of quantum computation. *Physical Review A* 58, 3 (1998), R1633.

[7] Andrew Cross. 2018. The IBM Q experience and QISKit open-source quantum computing software. In *APS March meeting abstracts*, Vol. 2018.

[8] Andrew Cross, Ali Javadi-Abhari, Thomas Alexander, Niel De Beaudrap, Lev S. Bishop, Steven Heidel, Colm A. Ryan, Prasahnt Sivarajah, John Smolin, Jay M. Gambetta, and Blake R. Johnson. [n. d.]. OpenQASM 3: A Broader and Deeper Quantum Assembly Language. *ACM Transactions on Quantum Computing* 3, 3 ([n. d.]). https://doi.org/10.1145/3505636

[9] Giordano d'Aloisio, Sophie Fortz, Carol Hanna, Daniel Fortunato, Avner Bensoussan, Eñaut Mendiluze Usandizaga, and Federica Sarro. 2024. *Exploring LLM-Driven Explanations for Quantum Algorithms - Replication Package*. https://doi.org/10.5281/zenodo.12805237

[10] Manuel De Stefano, Fabiano Pecorelli, Dario Di Nucci, Fabio Palomba, and Andrea De Lucia. 2022. Software engineering for quantum programming: How far are we? *Journal of Systems and Software* 190 (2022), 111326.

[11] Olivia Di Matteo. 2024. On the need for effective tools for debugging quantum programs. *arXiv preprint arXiv:2402.09547* (2024).

[12] Uwe Dorner, Rafal Demkowicz-Dobrzanski, Brian J Smith, Jeff S Lundeen, Wojciech Wasilewski, Konrad Banaszek, and Ian A Walmsley. 2009. Optimal quantum phase estimation. *Physical review letters* 102, 4 (2009), 040403.

[13] Chuck Easttom. 2024. Utilizing ChatGPT to Improve Quantum Algorithms. In *IEEE Annual Computing and Communication Workshop and Conference*. IEEE.

[14] Olivier Ezratty. 2023. How AI, LLMs and quantum science can empower each other? https://www.oezratty.net/Files/Publications/How%20LLMs%20and%20quantum%20science%20can%20empower%20each%20other.pdf

[15] Angela Fan, Beliz Gokkaya, Mark Harman, Mitya Lyubarskiy, Shubho Sengupta, Shin Yoo, and Jie M Zhang. 2023. Large language models for software engineering: Survey and open problems. *arXiv preprint arXiv:2310.03533* (2023).

[16] Daniel Fortunato, José Campos, and Rui Abreu. 2022. Mutation Testing of Quantum Programs: A Case Study With Qiskit. *IEEE Transactions on Quantum Engineering* 3 (2022), 1–17. https://doi.org/10.1109/TQE.2022.3195061

[17] Lov K. Grover. 1996. A Fast Quantum Mechanical Algorithm for Database Search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing* (Philadelphia, Pennsylvania, USA) *(STOC '96)*. Association for Computing Machinery, New York, NY, USA, 212–219. https://doi.org/10.1145/237814.237866

[18] Xiaoyu Guo, Jianjun Zhao, and Pengzhan Zhao. 2024. On Repairing Quantum Programs Using ChatGPT. In *2024 IEEE/ACM 5th International Workshop on Quantum Software Engineering (Q-SE)*. IEEE.

[19] Henry Hsu and Peter A Lachenbruch. 2014. Paired t test. *Wiley StatsRef: statistics reference online* (2014).

[20] Klaus Krippendorff. 2011. Computing Krippendorff's alpha-reliability.

[21] Juho Leinonen, Arto Hellas, Sami Sarsa, Brent Reeves, Paul Denny, James Prather, and Brett A Becker. 2023. Using large language models to enhance programming error messages. In *Procs. of the ACM Technical Symposium on Computer Science Education V.1*.

[22] Neilson Carlos Leite Ramalho, Higor Amario de Souza, and Marcos Lordello Chaim. 2024. Testing and Debugging Quantum Programs: The Road to 2030. *arXiv e-prints* (2024), arXiv–2405.

[23] Heng Li, Foutse Khomh, Moses Openja, et al. 2021. Understanding quantum software engineering challenges an empirical study on stack exchange forums and github issues. In *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 343–354.

[24] Gui-Lu Long. 2001. Grover algorithm with zero theoretical failure rate. *Physical Review A* 64, 2 (2001), 022307.

[25] Guillermo Macbeth, Eugenia Razumiejczyk, and Rubén Daniel Ledesma. 2011. Cliff's Delta Calculator: A non-parametric effect size program for two groups of observations. *Universitas Psychologica* 10, 2 (2011), 545–555.

[26] Stephen MacNeil, Andrew Tran, Dan Mogil, Seth Bernstein, Erin Ross, and Ziheng Huang. 2022. Generating diverse code explanations using the gpt-3 large language model. In *Procs. of the ACM Conference on Int. Computing Education Research-Volume 2*.

[27] Patrick E McKight and Julius Najab. 2010. Kruskal-wallis test. *The corsini encyclopedia of psychology* (2010), 1–1.

[28] Andriy Miranskyy, Lei Zhang, and Javad Doliskani. 2021. On testing and debugging quantum software. *arXiv preprint arXiv:2103.09172* (2021).

[29] Juan M Murillo, Jose Garcia-Alonso, Enrique Moguel, Johanna Barzen, Frank Leymann, Shaukat Ali, Tao Yue, Paolo Arcaini, Ricardo Pérez, Ignacio García Rodríguez de Guzmán, et al. 2024. Challenges of Quantum Software Engineering for the Next Decade: The Road Ahead. *arXiv preprint arXiv:2404.06825* (2024).

[30] Meike Nauta, Jan Trienes, Shreyasi Pathak, Elisa Nguyen, Michelle Peters, Yasmin Schmitt, Jörg Schlötterer, Maurice van Keulen, and Christin Seifert. 2023. From Anecdotal Evidence to Quantitative Evaluation Methods: A Systematic Review on Evaluating Explainable AI. *Comput. Surveys* 55, 13s (July 2023), 295:1–295:42. https://doi.org/10.1145/3583558

[31] Nils Quetschlich, Lukas Burgholzer, and Robert Wille. 2023. MQT Bench: Benchmarking Software and Design Automation Tools for Quantum Computing. *Quantum* (2023). MQT Bench is available at https://www.cda.cit.tum.de/mqtbench/.

[32] Nornadiah Mohd Razali, Yap Bee Wah, et al. 2011. Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. *Journal of statistical modeling and analytics* 2, 1 (2011), 21–33.

[33] Walid S Saba. 2023. Stochastic llms do not understand language: Towards symbolic, explainable and ontologically based llms. In *International Conference on Conceptual Modeling*. Springer, 3–19.

[34] Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management* 24, 5 (1988), 513–523.

[35] Sami Sarsa, Paul Denny, Arto Hellas, and Juho Leinonen. 2022. Automatic generation of programming exercises and code explanations using large language models. In *Procs. of the ACM Conference on Int. Computing Education Research-Volume 1*.

[36] Peter W. Shor. 1999. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Rev.* (1999).

[37] Dominik Sobania, Alina Geiger, James Callan, Alexander Brownlee, Carol Hanna, Rebecca Moussa, Mar Zamorano López, Justyna Petke, and Federica Sarro. 2023. Evaluating Explanations for Software Patches Generated by Large Language Models. In *Procs. of Int. Symposium on Search Based Software Engineering*. Springer.

[38] Lars St, Svante Wold, et al. 1989. Analysis of variance (ANOVA). *Chemometrics and intelligent laboratory systems* 6, 4 (1989), 259–272.

[39] Yohichi Suzuki, Shumpei Uno, Rudy Raymond, Tomoki Tanaka, Tamiya Onodera, and Naoki Yamamoto. 2020. Amplitude estimation without phase estimation. *Quantum Information Processing* 19 (2020), 1–17.

[40] Salvador Elías Venegas-Andraca. 2012. Quantum walks: a comprehensive review. *Quantum Information Processing* 11, 5 (2012), 1015–1106.

[41] Ronald L Wasserstein and Nicole A Lazar. 2016. The ASA statement on p-values: context, process, and purpose. , 129–133 pages.

[42] Yaakov S Weinstein, MA Pravia, EM Fortunato, Seth Lloyd, and David G Cory. 2001. Implementation of the quantum Fourier transform. *Physical review letters* 86, 9 (2001), 1889.

[43] Eric W Weisstein. 2004. Bonferroni correction. *https://mathworld. wolfram. com/* (2004).

[44] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in Software Engineering*. Springer Berlin Heidelberg, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-29044-2

[45] Robert F Woolson. 2005. Wilcoxon signed-rank test. *Encyclopedia of Biostatistics* 8 (2005).

[46] Jianjun Zhao. 2021. Quantum Software Engineering: Landscapes and Horizons. arXiv:2007.07047 (Dec. 2021). http://arxiv.org/abs/2007.07047 arXiv:2007.07047 [quant-ph].