

What Do We Really Know About Fortran?

Austen Rainer ✉ 

Queen's University Belfast, UK

Andrew Brown ✉ 

Queen's University Belfast, UK

Rebecca Taylor ✉ 

University of Southampton, England

Simon Hettrick ✉ 

University of Southampton, England

Abstract

Background. Anecdotal evidence suggests that the Fortran ecosystem – the language, tooling, codebases and broad community – continues to make a significant international contribution to science and engineering. Yet the common perception appears to be that Fortran is old and obsolete and, as a result, one should divest from it. There is no empirical evidence – e.g., no contemporary or historical survey – to corroborate or contradict such perception. A consequence is that this perception may be not only mistaken but also damaging to the Fortran ecosystem and to the global contribution that the ecosystem continues to make to society, the economy and environment.

Aims. To gather empirical evidence on the current state of the global Fortran ecosystem, in order to corroborate or challenge perceptions about the language, tooling, codebases and community.

Method. We conduct an international survey of the Fortran community, collecting information from 150 respondents across 25 countries.

Results. We report results about codebases (e.g. code size varies from 1KLOC to 5MLOC), community demographics (e.g., most respondents are male; most respondents hold doctorates), challenges (e.g., code complexity, human resource) and perceptions of Fortran (only 15% consider their codebases to be legacy codebases).

Conclusions. Our results provide the first descriptive benchmark of the Fortran ecosystem which can act as a resource or reference for future research and practice. Also, our results suggest that Fortran may be better understood as a highly-valuable, niche, contemporary ecosystem, one which continues to contribute to world-leading research in many science and engineering disciplines. As an “edge case”, it is ideally placed to provide unique and insightful contributions to our understanding of (scientific) software engineering, e.g., technical debt, refactoring, code complexity, and the significance of application domain knowledge. Since all codebases, regardless of language, have the potential to become legacy systems in the future, Fortran is actually at the forefront of confronting long-term challenges and consequences of engineering complex (scientific) software systems.

2012 ACM Subject Classification Software and its engineering; Applied computing → Physical sciences and engineering; Social and professional topics

Keywords and phrases Fortran, demographics, software engineering, research software engineer, computational scientist, software sustainability

Digital Object Identifier 10.4230/LIPIcs.ESEM.2026.16

Category Technical Track Paper

Supplementary Material *Dataset (Raw data, classifications, scripts for analysis, copies of figures in the paper):* <https://zenodo.org/records/20185150>

Funding The work of all authors is supported by grant UKRI4343 *Fortran.Future: a Network for Sustainable Fortran-based Science* from the UK's Engineering and Physical Sciences Research Council (EPSRC).

Andrew Brown: EPSRC grant EP/V05208X/1

Simon Hettrick: AHRC grant AH/Z000114/1



© Austen Rainer, Andrew Brown, Rebecca Taylor, and Simon Hettrick;
licensed under Creative Commons License CC-BY 4.0

20th International Symposium on Empirical Software Engineering and Measurement (ESEM 2026).

Editors: Robert Feldt, Maria Paasivaara, Daniel Mendez, Stefan Wagner, and Marvin Muñoz Barón;

Article No. 16; pp. 16:1–16:20



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Acknowledgements We thank those who piloted the survey, and all 150 survey participants, including the following respondents who chose to identify themselves: AdAstraPerAspera, Amir Shahmoradi, Amy Roberts, Anon77, An RSE from a Helmholtz research institute in Germany, ariiJudeIS, BiathlonFan, Boteh, CabraLattice, CampanologicalToxophilite, CampFire, CJM, CLaSSGroupLead, cmaapic, CoolFortran, Dalekopera, dmio7839r, DogCatMouse, DrFortranFan, dwa, ForProfit, FortranIsStillMyFavorite, FortranKPDH, Fotranista123BEva, FragmentedEchidna, Gaits, Gideon, GOTO_EndOfSurvey, GoTo_remove, HerbertPeck, hjf, ibperth, IndAtHomeFort18, InTartifletteWeTrust, jasNW, “Keeping people and society safe”, lsm, Machalot, Martin Diehl, mathemagician, Milan Skocic, Modeller2026, mwatkins, NumberCrunching4QD, OldCrustyAstroGuy, otisthecat, paleoprogrammer, Participant 21, ParticipantJad, PDS, PH, Phoenix, platipodium, Polis, PRIMA, RSEwithAGPL, SelaBuri, Simulacrum, T3, TNT, upg88743, yarchik.

1 Introduction

Anecdotal evidence suggests that the Fortran ecosystem – the language, tooling, codebases and broad community – continues to make a significant international contribution to science and engineering. For example, software written in Fortran accounts for between 65% and 80% (depending on assumptions) of the usage of the UK’s ARCHER2 supercomputer [7] and the language’s international standard continues to evolve [16]. At the same time, common perception appears to be that Fortran is old and obsolete and, as a result, one should divest from it. There is a lack of contemporary, or even historical, surveys or systematic reviews of either the state of the Fortran ecosystem or of the continuing contribution of that ecosystem to science and engineering, and to society, the economy and the environment. One consequence is the lack of rigorous, empirical evidence to corroborate or challenge the anecdotal evidence. A more serious consequence is that the lack of evidence also makes it hard(er) for policy makers, funders and resource-providers (e.g., hardware and compiler vendors) to properly judge the scale and impact of the ecosystem, with the risk that such stakeholders are both less likely to support that ecosystem and more likely to advocate divesting from it. A recent example is the Los Alamos National Laboratory’s evaluation of risks associated with relying on Fortran for mission critical software [30]. A reinforcing cycle arises: the state and contribution of the international Fortran ecosystem continues to not be empirically documented, encouraging the misperception that it is non-existent or unimportant. This, in turn, reduces support for the ecosystem, which makes it harder to collect empirical evidence to reliably assess its state and continuing contribution. It becomes increasingly difficult to advocate for and sustain Fortran-based science and engineering and to sustain the contribution of that science and engineering to society, the economy and the environment.

The **aim** of this paper is to intervene in the reinforcing cycle. The **objectives** of the paper are: 1) to report the first international survey of the current state of the Fortran ecosystem; and 2) to use the findings from the survey to promote further research into the Fortran ecosystem. Aligned with the objectives, the **main contribution** of the paper is the provision of a unique set of insights into an under-studied and under-appreciated, but also a highly valuable and complex, field of (scientific) software engineering. This study is **novel** in that no findings of this kind have been reported in the peer-reviewed literature; it is **significant** because of the continuing dependence of many science and engineering disciplines on Fortran; and it is **timely** because, as the findings will show, there are a set of risks accumulating for Fortran-based science and engineering.

The remainder of the paper is organised as follows: Section 2 presents a brief overview of the Fortran ecosystem, and then reviews prior research, using the overview and the review to motivate the study; Section 3 explains the design of the study, including design choices,

piloting the survey, participant recruitment, and data selection and analysis for the current paper; Section 4 reports descriptive results, using tables and visualisations, and provides a narrative explanation that brings together the results; Section 5 considers implications of the results, the challenges of generalising from the results, other limitations to the study, and some directions for future work. The appendices acknowledge respondents, list the survey questions examined in the paper, and explain the data package for open science.

2 Background and related work

2.1 Brief overview of the Fortran ecosystem

Fortran is the first of the third-generation programming languages, having been created at IBM by John Backus and others over the period 1953–1957. As a *language*, Fortran continues to be developed, with the latest revision to the Fortran standard being Fortran 2023 and a schedule now established for new revisions every five years [18]. The term *Modern Fortran* refers to revisions to the standard since Fortran 2003 inclusive, for which there have been four revisions so far: 2003, 2008, 2018 and 2023. The term *Fortran* either refers generally to the Fortran programming language or specifically to Fortran 90. The term *FORTTRAN* refers to revisions prior to and including FORTRAN 77.

Uniquely, the Fortran language *standard* continues to be backward- and forward-compatible, such that, for example, code written in FORTRAN 77 can compile and execute with code written in Fortran 2023 (provided the respective compiler supports it). Fortran is therefore the first and the oldest third-generation language, the oldest language with an international standard and with an active international standards working group, and is a language that continues to be actively developed. It is also the longest surviving “continuous language”, avoiding the kind of “breaking changes” that occurred with, e.g., Python 2 and Python 3. These are remarkable features of the Fortran language.

In addition to the *language*, there is now a standard library, a package manager and registry, an interactive compiler, and a language server¹ that integrates with leading IDEs. New technologies are being developed in Fortran, both to improve the Fortran ecosystem and to develop new applications. Rouson’s Fortran Group², at Lawrence Berkeley National Laboratory, is a world-leading example, developing a parallel runtime interface for Fortran [27], together with a parallel runtime library (Caffeine), a deep-learning framework implemented in modern parallel Fortran (Fiats), a correctness checking framework (Julienne), and a library for runtime checking (Assert) [17]. As a technical ecosystem, Fortran is in a stronger position now than at any time in its history [16]. Two online fora for the international Fortran community – `fortran-lang.org` and `fortran-lang.discourse.group` – complement the technical ecosystem.

Fortran continues to be used in a very wide range of science and engineering disciplines. Six of the top ten codebases running on the UK’s ARCHER2 Tier 1 HPC service are written in Fortran, namely VASP (electronic structure and atomistic materials modelling) [36], CP2K (molecular dynamics and condensed-matter simulation) [8], CASTEP (materials science) [5], MetOfficeUM (weather prediction and climate modelling) [32], SENG (combustion and turbulent reacting flows) [29], and NEMO (ocean and sea-ice modelling) [23]. International examples of the continued use of Fortran for large-scale applications include the

¹ <https://fortls.fortran-lang.org/index.html>

² <https://amcr.lbl.gov/departments/computer-science-department/class/fortran-at-lbl/>

Weather Research and Forecasting (WRF) Model [37], the Community Earth System Model (CESM) [22], DFTB+ (quantum mechanical and atomistic simulation) [10], Molpro (ab initio quantum chemistry) [21], and ParaMonte (high-performance Monte Carlo and statistical simulation) [25]. Taken together, these Fortran codebases, and many others, directly power a significant number of highly successful and impactful scientific and engineering endeavours.

The indirect scientific importance of Fortran is arguably even larger, though also much less visible: core numerical operations of Numpy and Scipy (Python), Matlab, Julia and R *all* depend on executables compiled from established Fortran-based libraries, such as LAPACK [33], a library for high-performance linear algebra routines.

For more detailed reviews of the history of Fortran see, for example, Metcalf [19] and Snyder [34].

2.2 Prior research into the Fortran ecosystem

There is a reasonably sizeable body of prior research investigating technical aspects of Fortran codebases. As examples: Vanderbauwhede [35] demonstrates the automatic refactoring of FORTRAN 77 code into type-safe Fortran 90 code; Dhruv and Dubey [11] report the use of large language models to translate Fortran code into C++ code; García-Rodríguez, Añel and Rodeiro-Iglesias report on their development of a Fortran static code analyser [13] that can analyse code written with all revisions of Fortran (and FORTRAN); and Hasselbring, Jung and Schnoor [15] report the combined use of dynamic software profiling with static code analysis to assess two widely used earth system models.

Some surveys include Fortran amongst their responses; however they do not focus specifically on Fortran. For example, Giordani et al. [14] report a replication of Prabhu et al.'s [26] in-depth survey of the practices of computational science at leading research institution in the USA. These surveys focused on computational practices, not on languages, though languages are considered in the responses. Giordani et al. [14] found that the percentage (27%) of language use for Fortran had not changed *at that university* over the decade since the original survey [26].

In terms of *scientific* software, Farhoodi et al. [12] report a systematic mapping study of primary studies investigating the development of scientific software. Farhoodi et al. found, from their sample, that the majority of scientific software systems were written in Fortran (36/130 studies; 28%) followed by C++, C, Python, Java and Matlab.

Whilst a recent article discussed the state of the Fortran *language* [16], and a fairly recent report [31] from the British Computer Society's Fortran SIG presented raw results of a survey of 427 respondents into the benefits of Fortran *standardisation*, we are unaware of any research – recent or in the (distant) past – into the *global* state of the Fortran *ecosystem*.

2.3 Summary

The lack of prior assessments of the state of the international Fortran ecosystem and the evidence of the continued use of Fortran in science and engineering, and “within” other programming languages, motivate the survey we report here, i.e., to gather empirical evidence on the current state of the global Fortran ecosystem, in order to corroborate or challenge perceptions about the language, tooling, codebases and community.

3 Method

3.1 Overview

The **aim** of the survey – to better understand the current state and impact of the global Fortran community – translates into several study **objectives**, i.e., to collect information on:

- Those who work on, or with, Fortran codebases, e.g., research software engineers, computational scientists.
- The environment, e.g., the kinds of institution within which research software engineers and computational scientists work, and within which the Fortran codebases are used.
- The perceptions of Fortran, e.g., whether Fortran codebases are perceived as legacy code.
- The characteristics of the Fortran codebases, e.g., size, dependencies, licensing.
- The challenges confronting those who work on the codebases, as well as suggestions on how those challenges might be addressed.

To increase both the frequency and the quality of response, the survey was designed to collect information on targeted items of relevance, rather than designed to collect a wide range of information that might be relevant. Consequently, and consistent with all empirical studies, we intentionally decided not to collect some information. Three examples illustrate the design choices. First, whilst diversity and inclusion are known issues in software engineering and warrant research attention (e.g., [9]) the survey was designed to collect information only on demographics directly relevant to specific hypotheses we had on gender, age and expertise. Second, participants were asked to provide responses relating to a single codebase of their choice. Whilst this decision reduces the prospective coverage of codebases, it helps reduce responses of the form, “it depends”. It also increases the accuracy and reliability of responses because each response is about a single Fortran codebase. As a complementary question, however, we asked respondents to indicate how many codebases they work on and the basis for choosing the single codebase they did. These complementary questions allow us to assess the impact of our design choice to focus on single codebases. We return to this point in Section 5.2, where we consider the generalisability of the findings. Third, and as discussed in Section 3.2, we conducted a two-stage pilot study to assure the relevance of our design choices.

The final survey comprises 39 questions. A copy of the survey is provided in the accompanying data package (see Appendix B). Publication constraints mean we are unable to report on all the data in this single publication. Instead we select particular questions to report on. In anticipation of future publications and to support cross-referencing across open access datasets, we reference the original survey question numbers when we report our results.

The study, including its protocol, has been approved by the University’s Faculty Research Ethics Committee, with protocol number [redacted for the review to preserve anonymity]. Three of the co-authors have prior experience of national and international surveys of software engineers or similar. Later, in Section 5.3, we use Molléri et al.’s [20] 10-stage checklist for software engineering surveys to assure the quality of the survey.

3.2 Piloting the survey

To complement the review of the draft survey by the four co-authors, we asked several professionals working within, or with, the Fortran global community to pilot the survey. Pilot respondents were asked to not only review the questions but to complete the survey. All those we approached agreed. Pilot respondents are summarised in Table 1.

■ **Table 1** Demographics of respondents to the pilot study.

ID	Gender	Age	Country	Role	Access- ibility	Duration ^d	Completed survey
1	Male	35-44	UK	CS		<10min	Yes
2 ^a	Female	35-44	Germany	RSE		NS	Yes
3	Female	45-54	UK	CS		15min	Yes
4 ^a	Male	35-44	USA	RSE		10 - 15min	Yes
5	Female	25-34	USA	CS		18min+7min ^c	Yes
6	Male	NS	UK	RTP	Yes	NS	No
7	Female	NS	UK	RSE ^b		NS	No

NS = Not stated; CS = Computational Scientist; RSE = Research Software Engineer

RTP = Research Technical Professional

^aRespondent re-reviewed a revised version of the survey.

^bRespondent does not work directly on Fortran codebases but provides professional development to RSEs, including Carpentry courses on Fortran.

^cRespondent videoed themselves thinking-out-loud as they completed the survey (18 minutes) followed by post-survey reflections (7 minutes).

^dRespondents were asked to record duration to get a sense of their perception.

The pilot was initially intended to be a single round of review, however the feedback from some respondents identified significant limitations to the survey. This led us to revise the survey and conduct a second round of review, with feedback from some respondents. Respondents who re-reviewed the revised survey are identified in Table 1.

Respondent PR5 chose to video themselves as they completed the survey, thus providing a “think out loud” form of feedback. This feedback was complemented by their videoed self-reflections at the end of completing the survey, together with a set of email comments.

Respondent PR6 was unable to complete the survey for accessibility reasons. This was in itself an important test of the survey. The way we had designed the online survey, given the survey platform we were using (and recommended by the University who approved the study), meant that respondents would need to “page through” each section of the online survey form. This presented difficulties for PR6. One alternative was to design the survey to be presented in its entirety on a single scrollable webpage. For other reasons, such a design can also overwhelm and discourage respondents. As a compromise, we retained the page-by-page design but also produced a PDF version of the entire survey, and made the PDF available with the Participant Information Sheet and a copy of the Consent Form. The PDF would allow all respondents to review the questions prior to attempting the survey. This is an unsatisfactory compromise, but a necessary one, given the constraints.

Finally, due to time constraints, Respondent PR7 was not able to complete the survey, but did feedback on some of the survey questions.

Overall, the pilot study led to the following kinds of changes to the survey:

- Changes to the visual presentation of the survey to make it less distracting, e.g., use a plain background with a larger font size.
- Revised the introduction to clarify the purpose of the survey and to link to additional information, such as the PDF version of the entire survey.
- Added additional options for some multiple-choice questions.
- Changed the overall structure of the survey, i.e., re-ordered the sequence of some questions.
- Changed the phrasing of some questions, e.g., changing “versions” of Fortran to “revisions to the Fortran standard”.

■ **Table 2** Channels for promoting the survey.

Channel	Membership	Scope	Comment
<i>Direct promotion by the co-authors</i>			
fortran-lang discourse	1800	Global	Discourse channel for fortran-lang.org
SocRSE Fortran Slack	200	UK	UK Society of RSEs Fortran slack channel
DE-RSE github site	–	Germany	Society of RSEs in Germany
SSI blog post	–	Global	Software Sustainability Institute’s blog post
Mailing list	75	Global	Mailing list created by us from the annual <i>Back to the Fortran Future</i> workshops held as part of the RSECon in the UK.
CoSeC JISCMAIL	–	UK	Mailing list for the UK’s <i>Computational Science Centre for Research Communities</i>
<i>Indirect promotion by members of the community</i>			
American Physical Society Division of atomic, molecular, and optical physics forum			
American Association of Physics Teachers PICUP (computational physics) Slack channel			
Atomic Molecular and Optical Science gateway Community			
Los Alamos National Lab, USA			
Georgia Tech, USA			
Jülich Research Centre, Germany			

- Introduced new questions, in particular in response to suggestions from Respondent PR2.
- Improved methods of answering questions, e.g., converted the question about country to a drop-down box.

3.3 Recruitment of participants

We directly promoted the survey to prospective respondents through a number of online channels. Members of the global Fortran community also promoted the survey indirectly. Direct and (known) indirect channels of promotion are summarised in Table 2.

As per the requirements of the respective research ethics board, we explicitly invited only healthy adults who worked with Fortran codebases to complete the survey.

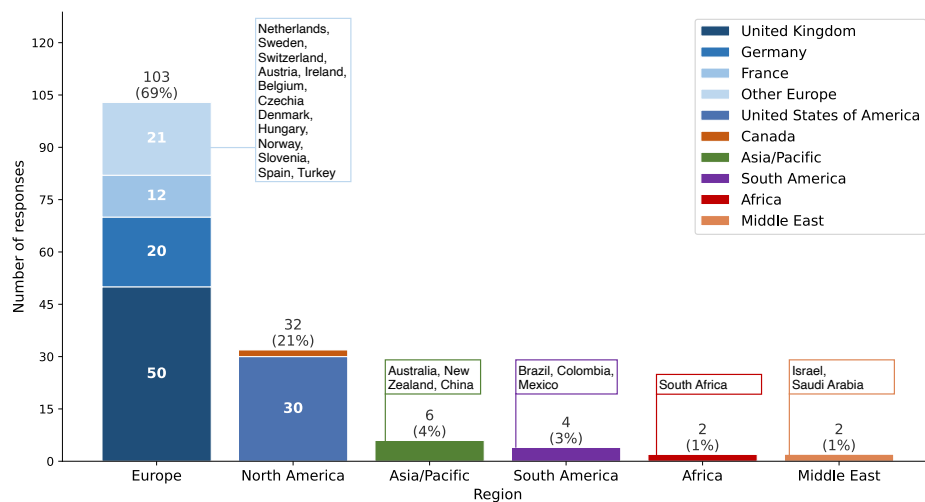
We included in the survey the option for participants to uniquely and anonymously identify themselves so that we can explicitly acknowledge their contributions. Participants who chose to identify themselves in some way are listed in the Acknowledgments section of the paper’s Appendices.

3.4 Analysis of “open” and “closed” questions

For “closed” questions, comprising a series of pre-defined answers, we report frequency counts in our results.

There is one “open” question we report in the current paper, for which respondents were asked to provide qualitative comments. This question asks about the most significant challenge – technical or non-technical – personally confronted with the codebase by the respondent (survey question Q23). We analysed this question using the following procedure:

1. The *second* author independently reviewed all responses to the question, constructed a classification, and then coded responses according to the classification. The second author is a physicist and research software engineer, and has used a Fortran codebase for much of their scientific career.



■ **Figure 1** Geographical location of the organisation that owns the respondent's codebase (Q25).

2. The *first* author then used *only* the classifications to independently code all responses, and did so for each question. The first author is a software engineering researcher. The first author did not see how the second author had classified the responses.
3. After coding all responses independently, the first author then compared the two authors' classifications, and assessed inter-rater reliability, sharing that assessment with the second author.
4. The two authors then discussed differences for the first 50 responses. During their discussion, the definitions for the classifications were clarified, and two categories were merged.
5. The two authors then independently reviewed the classification of the remaining 100 responses.
6. The two authors then met to discuss and resolve remaining differences.

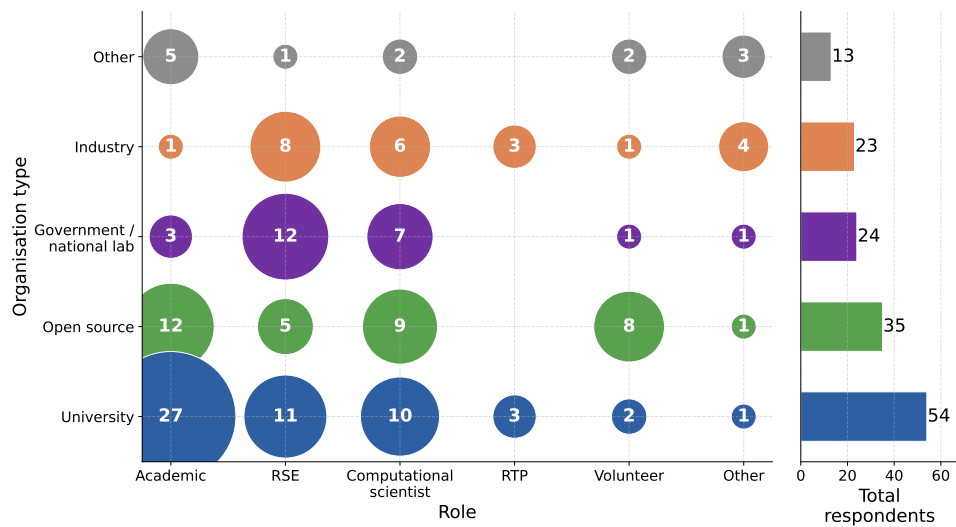
The final classifications are provided in the accompanying data package.

4 Results

This section reports a selection of results from the survey, using tables and visualisations accompanied by narrative explanation. The section first presents and discusses demographic information, then proceeds to information about codebases, then integrates demographic and codebase information, and concludes with the presentation and discussion of the most significant challenges confronted by respondents with their codebases.

Figure 1 shows the geographical location of the organisation that owns the respondent's codebase. The high proportion of respondents from the UK (33%), Europe (69%) and North America (21%) may arise from the historical concentration of scientific and engineering software activity in these regions, or may be due to bias in the data because of the way(s) we promoted the survey, or both. We return to this issue in Section 5.2.

Figure 2 presents information on the organisations that “own” the codebases and the role of the respondents within those organisations. A large number (37%) of responses relate to codebases “owned” by universities, which aligns with the historical use of Fortran in areas of fundamental science and engineering research. However, the industrial/societal relevance



■ **Figure 2** Type of organisation that “owns” the codebase (Q26) vs role of respondent in the organisation (Q27).

of those disciplines is indicated by the number of responses related to codebases owned by industry (15%) and governmental agencies (16%). The roles fulfilled by the respondents again skew heavily towards academic roles: 33% academics, 25% Research Software Engineer (RSE). Interestingly, the majority of RSEs (26/37), and of computational scientists (24/34), responding to the survey are working on codebases that are not owned by universities. This may partly be explained by sources of funding (universities may not have the funding), or by the nature of the work, or by organisational differences; but overall, it suggests that further work is required to understand the kind of work that RSEs and computational scientists – in contrast to academics – are doing in this space.

Figure 3 complements Figure 2, showing the highest academic qualification held by the respondents, and their self-assessment of the relevance of that qualification for their current role. Perhaps unsurprisingly, given the large number of those in academic and computational scientist roles, a large majority (82%) of respondents hold a PhD³. What is more striking is that 88% of those respondents consider their doctorate to be “essential” or “directly relevant” for their current role. This may suggest that the domain-specific knowledge requirements for the Fortran-community are higher than for other fields or applications. This is corroborated by answers to Q23 on challenges (Table 5), where “Domain knowledge” is the eighth most common category of response.

Figure 4 presents age and gender information about the respondents. Gender minorities remain under-represented across software engineering and programming occupations, with recent estimates placing women⁴ at approximately 20–28% of software developers and programmers in the UK and US technology sectors [3, 1]. Research Software Engineering (RSE) communities in the UK exhibit gender disparities similar to those observed in software engineering more broadly. A survey by the Software Sustainability Institute reported that women comprised approximately 14% of the UK RSE community, comparable to estimates for software developers generally, but substantially lower than the proportion of women in

³ Habilitation is an advanced post-doctoral academic qualification in Europe.

⁴ We note that these surveys still adopt binary gender categories.

16:10 What Do We Really Know About Fortran?

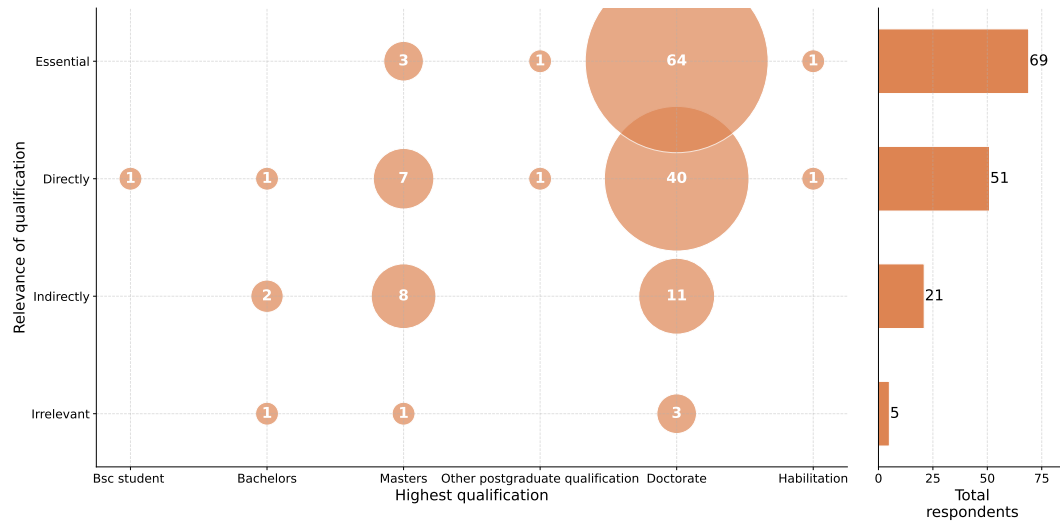


Figure 3 Respondent's highest academic qualification (Q33) vs Relevance of that qualification (Q34).

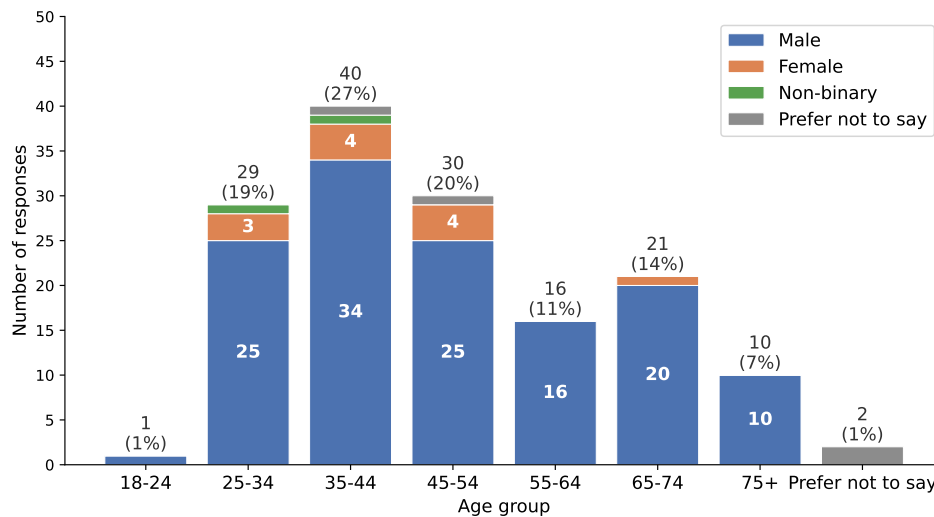


Figure 4 Age (Q31) and gender (Q32) of respondents.

■ **Table 3** Choice of codebase.

Number of codebases respondent works on or with (Q2)	Count	%
One codebase	29	19
More than one codebase	118	79
Other	3	2
Reason(s) for choosing codebase for survey (Q3)	Count	%
codebase I most frequently work on	73	49
most important codebase for me	63	42
most significant codebase to science and engineering, of those I work on	52	35
codebase I contribute to most, relative to all contributors	43	29
most complex codebase (by some definition of complex).	43	29
only codebase I work on	24	16
most impactful codebase to society... of those I work on ^a	18	12
newest codebase	15	10
oldest codebase	12	8
most problematic codebase	11	7
other	2	1
Type of codebase (Q4)	Count	%
Tool, e.g., linter, build tool, package manager	5	3
Library, e.g., stdlib	24	16
Application	113	75
Other	8	5

^amost impactful codebase to society, the economy or the environment, of those I work on

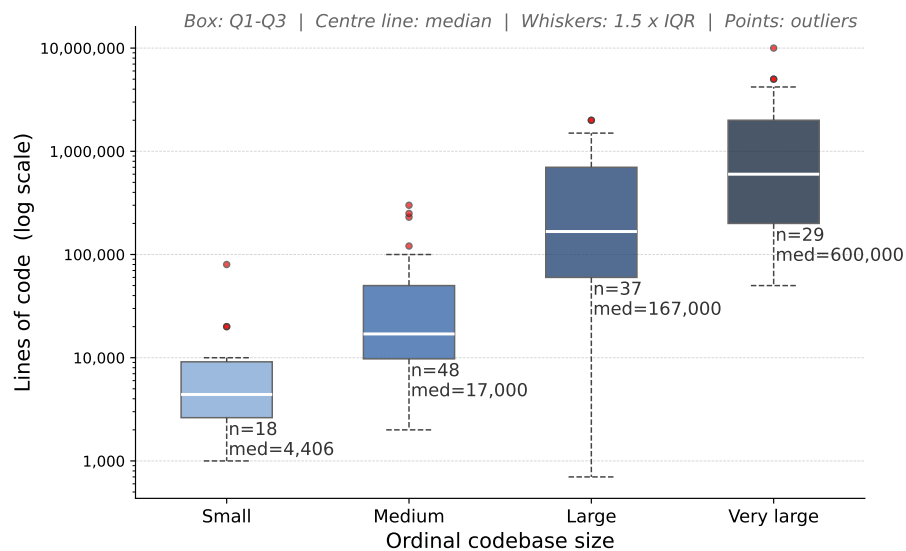
UK academia overall (46%) [6]. Since RSE communities are closely associated with scientific computing, HPC, and computational research infrastructures, these figures may provide a more appropriate comparison baseline for Fortran communities than general technology-sector demographics. Even so, the proportion of respondents identifying as female (8%) or non-binary (<1%) in our survey is markedly lower than even these broader baseline statistics.

More striking perhaps is the age spread of respondents, with 21% over the age of 65, and 32% age 55 or above. This large representation in the older categories is perhaps unsurprising given the age of Fortran itself, but in combination with the other technical challenges facing the community (discussed below), this may represent a (potentially) unique risk for Fortran-based science. We revisit this issue later in this section.

As discussed in Section 3, survey respondents were asked to choose one codebase and provide responses for that codebase only. Table 3 indicates that the majority of respondents (79%) work on more than one Fortran codebase. The table also shows that most respondents are choosing an application (Q4) which is the codebase they most frequently work on, or is most important *for them* (Q3). The (self-)selection of a codebase clearly has an effect on the characteristics of the sample and on inferences that may be made from the sample. We discuss these issues in more detail in Section 5.2, when we consider limitations to generalisability of the findings.

The survey asked respondents to characterise the size of their chosen codebase both numerically and ordinally. The ordinal response is clearly a subjective estimate. The numerical response may also be subjective but is likely to be more objective, given the respondents can (relatively easily) generate a lines-of-code (LOC) count for their own codebase. Responses to these questions are summarised in Figure 5. 50% of the codebases are characterised as

16:12 What Do We Really Know About Fortran?



■ **Figure 5** Ordinal size of the codebase (x-axis; Q8) vs numerical size (y-axis; Q7).

■ **Table 4** Perceptions of legacy.

Personal view (Q20)	Perceptions of others' views (Q21)				Total ^a
	No	No – modernised	Yes	Other	
No	74 (49%)	0 (0%)	4 (3%)	6 (4%)	84 (56%)
No - modernised	21 (14%)	1 (1%)	7 (5%)	1 (1%)	30 (20%)
Yes	2 (1%)	0 (0%)	21 (14%)	0 (0%)	23 (15%)
Other	5 (3%)	0 (0%)	2 (1%)	6 (4%)	13 (9%)
Total ^a	102 (68%)	1 (1%)	34 (23%)	13 (9%)	150

^arounding leads to slight difference with the totals

large or very-large, however Figure 5 shows that “large” spans an enormous range, from 700 to 2M lines of code. This range may be due to errors in the respondents’ answers, e.g., typos, or may reflect a true difference in perception. Nevertheless, there are a large number of substantial codebases, with 21 responses citing at least 1M lines of code.

Survey respondents were also asked whether they considered their chosen codebase to be a legacy codebase (note that in Table 3 only 8% of respondents chose their oldest codebase) and whether they thought other people they worked with thought the codebase was a legacy codebase. Table 4 cross-tabulates the results: the majority of respondents (76%) do not consider their own codebases to be legacy systems. Interestingly, 20% of respondents have modernised a once-legacy codebase.

Figure 6 cross-tabulates the size of the development team with the age of the developers. Figure 7 cross-tabulates size of codebase with the size of the development team. In relation to Figure 6, a staggering 26% of respondents are the sole-maintainers of their codebases, and a further 38% have development teams of between only 2 and 5 people. Figure 7 shows that, in general, the smaller codebases have smaller teams. What these results don’t show, and which we intend to investigate further, is the importance of the respective codebase to science and engineering. Other questions arise too, for example whether sole-maintainers more frequently occur in university or open-source environments.

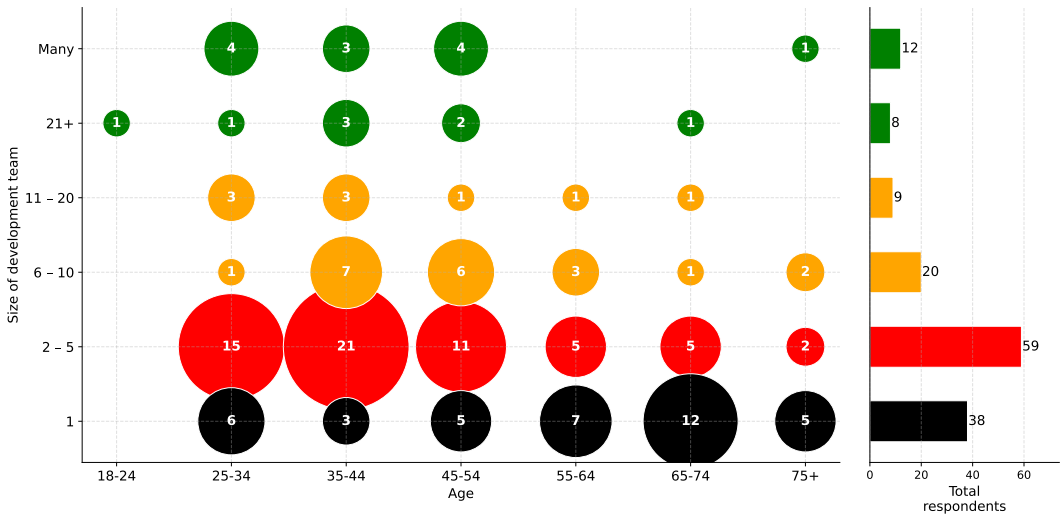


Figure 6 Age (Q31) vs approximate number of people currently working on the codebase (Q6).

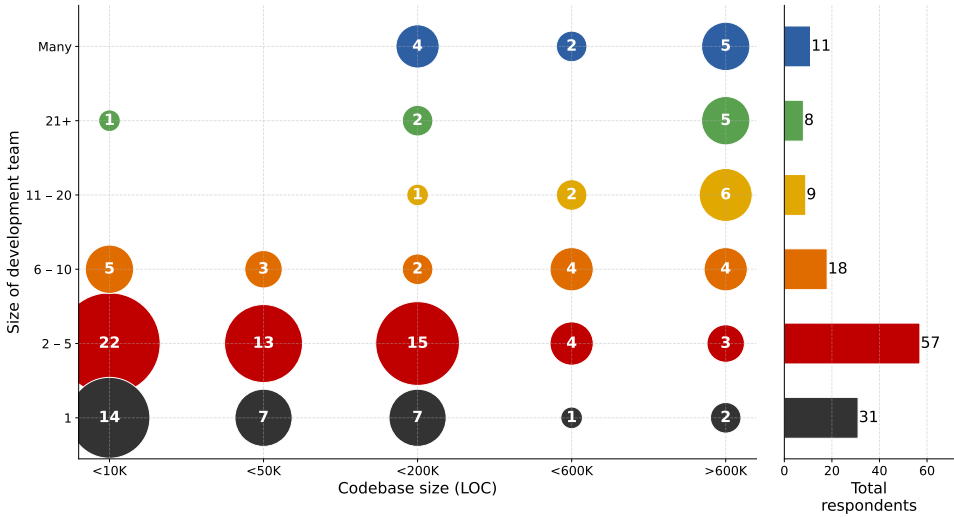


Figure 7 Codebase size vs team size.

■ **Table 5** Most significant challenge personally confronted (Q23).

Category and description	Count
Code complexity: poor structure, size, modularity, brittle design, comprehending/navigating a large codebase	29
Human resource: shortage of developers, recruitment, knowledge concentrated in too few people, difficulty training successors, departure of original authors, personal skills	22
Legacy code & technical debt: inherited old code: Fortran 77 idioms, GOTO statements, COMMON blocks, implicit typing, 6-character names, decades of un-refactored code	18
Compiler & language support: Compiler bugs, slow/inconsistent adoption of standards across compilers, missing language features (generics, polymorphism, functional constructs)	18
Documentation: including where science/design rationale is assumed	17
Testing & debugging: no/inadequate tests, difficulty debugging (including numerical errors and non-obvious wrong results), reproducibility problems, insufficient debugging tooling	16
GPU & HPC: multiple GPU vendors/standards, mixed parallelism (MPI, OpenMP, co-arrays, OpenACC), reliance on HPC infrastructure	14
Domain knowledge: difficulty of the underlying science or mathematics	13
Portability: supporting multiple operating systems/hardware/compilers	11
Time & funding: Insufficient time for maintenance, modernisation, or documentation work; funding constraints	9
Fortran perception: perception that Fortran is “ancient”/legacy: reluctance from students, colleagues, or management to engage with Fortran	9
Dependency management & build systems: external library dependencies, build system friction (CMake, FPM), package management	7
Organisational factors: project politics, organisational decisions (deprecating Fortran), conservative gatekeeping by long-standing contributors, poor adoption by institutions, misaligned career incentives for software maintenance work	6
Software engineering practices & tooling: (lack of) version control, build systems, software design principles	6
Interoperability with other codebases & languages: e.g., Python, C/C++	4

Finally, and briefly, the survey also asked respondents for the most significant challenge, technical or non-technical, they personally confront with the codebase. Table 5 reports categorised results, which have been analysed according to the procedure outlined in Section 3.4. The results show a combination of technical and non-technical challenges. Note that in Table 3 only 7% of respondents chose their most problematic codebase, suggesting a difference between “problematic” and “significant challenge”.

The response categories are a mix of technical (e.g. Code complexity, testing) and “social” (e.g. Human resource, perception of Fortran). This suggests the rich potential for future socio-technical research into the Fortran ecosystem which we discuss in Section 5.

5 Discussion

5.1 Implications of the results

There are several striking features in the survey which we believe mark the Fortran ecosystem as worthy of continued study. Some of the features are *technical*, such as compiler support, testing or tooling. But it is the intersection of these technical features with the *demographics*

and *culture* of the Fortran community, as well as the necessary *domain knowledge*, which – we argue – makes the Fortran ecosystem a unique and, as-yet, largely unexplored area, and one which therefore presents rich opportunities for further research. Consider the following, as a broad summary of the “shape” of the challenge: 1) a third of respondents are over the age of 55; 2) four-fifths of respondents hold a PhD or higher qualification; 3) more than half of those holding a PhD or higher qualification consider their qualification “essential” for the work they do; 4) it is only fairly recently that the professional role of the Research Software Engineer has emerged to support the computational scientist or academic; 5) the best practices of software engineering found in other application domain areas (e.g., web applications, financial systems) are often absent from ongoing Fortran projects [4]; 6) the longevity of the Fortran language, as well as the language standard’s commitment to backward compatibility, means there are “layers” within codebases that are written to different language standards, with different coding conventions, and with undocumented design and coding rationales; 7) many scientific and engineering disciplines, and *other programming languages*, remain dependent on Fortran; 8) codebases need modernising; and 9) many of the codebases are maintained by very small teams.

One respondent crystallised this complex situation: *“A better understanding by researchers on the importance of version controlling, e.g. git. At the moment, the code suite and/or any changes are passed around in a tarball via email, making it very hard to track what changes have been made and when, and also determine if any bugs have been introduced. The other academics are not software-focussed, and have the attitude that this is the way it has always been done, and it works, so we won’t fix it, but it is not sustainable, and on-boarding is nearly impossible.”* Other respondents spoke of challenges relating to:

- **Education:** *“A return to teaching Fortran to undergraduates and postgraduates.”*
- **Perception:** *“An irrational fear of Fortran by potential contributors or users”*
- **Incentives:** *“Finding people with appropriate expertise willing to work for the pittance that UK academia deems it appropriate to pay RSEs.”*
- **Succession:** *“I will retire in say 7 years and I yet have no obvious colleague who can manage to take over the responsibility.”*

One response to the broad summary we provide, and to the example quotes, is that they provide reasons for research to *ignore* the Fortran ecosystem and for science and engineering to *divest* from Fortran. As with all codebases and all programming languages, divesting from that language (e.g., porting the codebase) is often an enormously time-consuming, effortful and expensive undertaking, and one which comes with much risk. Furthermore, such divestment doesn’t prevent the risk that similar challenges will recur in the future.

An alternative response, and one which we advocate, is to understand the current Fortran ecosystem as presenting a vision of the future of other, currently “more modern”, (scientific software) ecosystems. In that sense, Fortran is at the forefront of dealing with multi-decade challenges, such as the challenges of the declining availability of expertise, the accumulation of “layers” of diversely-written code, the mismatch over time in software engineering practices and tools, and the problems of dependencies and interoperability across codebases, languages, language features and compilers. Thus, the current Fortran ecosystem presents unique opportunities to contribute timely and significant knowledge in areas such as requirements, domain expertise, software design (e.g., [28]), technical debt (e.g., [24]), refactoring (e.g., [35]), program comprehension, code analysis, automated code translation, software testing and software assurance, secure software, and continuous integration and deployment.

The Fortran ecosystem also acts as an exemplar to distinguish the nature and challenges of scientific software from those of non-scientific software.

5.2 Limitations in sampling and generalisation

Sampling and generalising to populations is a well-recognised and recurring problem for software engineering surveys [2]. It is also an issue explicitly identified in Stage 3 *Identify Population* and Stage 4 *Sampling Plan* of Molléri et al.'s [20] checklist for survey research in software engineering. For these reasons, we explicitly consider the limitations of sampling and generalisation. Discussing sampling also has the benefit of helping future research, e.g. when researchers are deciding on the selection of indicative Fortran projects or edge-case projects for case study research.

The survey adopted a convenience sampling approach [2], though this was indirectly complemented by referral-chain sampling [2] as some survey respondents, and some pilot-study respondents, promoted the survey to their peers. The most similar survey, for comparing sampling and samples, is the survey about Fortran standardisation, conducted in 2018 by the British Computer Society's Fortran SIG [31], and briefly discussed in Section 2. That survey reported results from 427 respondents, probably collected through convenience sampling. The BCS ran their survey for approximately six months (30 June 2018 to 3 January 2019) whilst the survey reported here was open for approximately 6 weeks (January 2026 - February 2026). This may partly explain the difference in number of respondents (427 vs. 150). As a further comparison, the `fortran-lang.org` website reports approximately 1800 people are registered on the site.

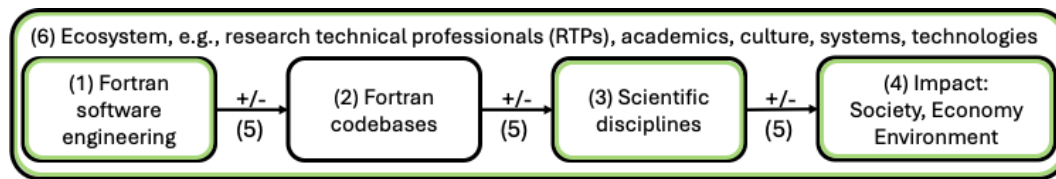
The BCS survey's demographic data suggests their respondents were located across 41 territories, e.g., country or some distinct region identified by the respondent. The highest number of responses to the BCS survey came from the USA (120/427; 31%) and the United Kingdom (117; 30%) and then from Europe (the figures reported are imprecise but approx. 71; 16%). There is, unfortunately, no information about the sampling frame or sampling approach taken by the BCS. Also, at the time that the BCS conducted their survey, the `fortran-lang.org` website – the largest source of prospective respondents for the current survey (see Table 2) – did not exist. The similar proportion of respondents for the USA and the UK – across the two surveys with different time-frames, different numbers of responses, a different spread of responses across countries, and different sources of prospective respondents – suggests that the majority of the Fortran ecosystem activity is indeed located in the USA and the UK, with a sizeable cohort present in Europe too.

The survey reported here comprises two units of analysis: the respondent and the codebase. This is highlighted in Figure 7. To better control for the codebase, and the relationship between respondent and codebase, the survey asked respondents to focus on a single codebase. For each unit of analysis there are likely a small number of instances of double-counting, e.g., two respondents describing the same codebase, or two respondents providing information about the same organisation. Being anonymous responses, these instances are hard to detect. Given the range in the number of people working on codebases, double-counting is more likely to occur for larger codebases; again, see Figure 7.

The number of Fortran codebases explicitly identified, through the respondent providing a URL or similar, is 69. Assuming, *very conservatively*, that all respondents (79%) who work on “more than one” codebase are working on only two codebases, an extremely conservative lower bound for the total number of Fortran codebases in the international Fortran ecosystem is:

$$(0.19 \times 150 \times 1 \text{ codebase}) + (0.79 \times 150 \times 2 \text{ codebases}) = 265 \text{ codebases.}$$

As two examples to illustrate how this estimate is extremely conservative, one respondent indicated they worked on 6 codebases and another respondent indicated they worked on 30 codebases. These responses lead us to interpret 265 as a very conservative lower bound. But importantly, we estimate a lower bound that can contribute to future research.



■ **Figure 8** A framework for studying “routes to impact” of Fortran.

5.3 Other limitations

We used Molléri et al.’s [20] 10-stage checklist for software engineering surveys as a guide to self-evaluate the quality of the survey. The major limitations identified have already been discussed in Section 5.2. In terms of survey design and validation (cf. Molléri et al.’s Stage 5 *Instrument Design* and Stage 6 *Instrument Validation*), the first two authors initially designed the survey, which was then reviewed by the other authors, and then validated through a two-stage pilot study in which the pilot respondents were asked not just to review the survey but complete it too. The survey was also reviewed, as part of the ethics approval process. Participant recruitment (Stage 7) is discussed in Section 3. For (non-)response management (Stage 8) we sent occasional reminders about the survey to the `fortran-lang.org` site, including keeping the survey open for about two weeks longer than originally planned, and also responded to all queries from prospective respondents, encouraging them to complete the survey. Appropriate data has been validated (Stage 9) as discussed in Section 3 and is being made available as an open source data package.

5.4 Further research

Our longer-term aim is to study the “chain of impact” of Fortran, as briefly visualised in Figure 8. The survey provides a starting point to pursue this aim as well as several specific directions for further research: 1) there are more data in the survey to analyse and report; 2) there are sub-samples of the data to investigate, e.g., how responses vary with age, role, territory; and 3), there is the need to clarify and elaborate on information collected from respondents, e.g., the number of codebases respondents are working with, and the challenges that have been identified in Table 5. There are also directions for new research, the most obvious being to investigate the 69 codebases identified by the respondents. We also strongly encourage other researchers to consider how the Fortran ecosystem – the language, standards, tooling, codebases and community – can provide a unique source for novel and significant contributions to their research into (scientific) software engineering.

References

- 1 AIPRM. Women in tech statistics. <https://www.aiprm.com/en-gb/women-in-tech-statistics/>, 2024. Accessed: 2026-05-13.
- 2 Sebastian Baltes and Paul Ralph. Sampling in software engineering research: A critical review and guidelines. *Empirical Software Engineering*, 27(4):94, 2022. doi:10.1007/S10664-021-10072-8.
- 3 BCS, The Chartered Institute for IT. Women in technology: Key findings. <https://www.bcs.org/policy-and-influence/equity-diversity-and-inclusion/bcs-diversity-report-2024-addressing-the-under-representation-of-women-in-technology/women-in-technology-key-findings/>, 2024. Accessed: 2026-05-13.

16:18 What Do We Really Know About Fortran?

- 4 Andrew Brown. What's next for atomic and molecular physics software? *Nature Reviews Physics*, 2(2):68–69, February 2020. doi:10.1038/s42254-019-0139-y.
- 5 CASTEP Developers. Castep. <https://www.castep.org/>, 2026. Accessed: 2026-05-15.
- 6 Neil P. Chue Hong, Jeremy Cohen, and Caroline Jay. Understanding equity, diversity and inclusivity challenges within the research software community. In Maciej Paszynski, Dieter Kranzlmüller, Valeria V. Krzhizhanovskaya, Jack J. Dongarra, and Peter M. A. Sloot, editors, *Computational Science – ICCS 2021*, volume 12747 of *Lecture Notes in Computer Science*, pages 390–403. Springer, Cham, 2021. doi:10.1007/978-3-030-77980-1_30.
- 7 Jim Cownie. Is fortran a dead language. Accessed 1/4/26. URL: <https://cpufun.substack.com/p/is-fortran-a-dead-language>.
- 8 CP2K Developers Group. Cp2k open source molecular dynamics. <https://www.cp2k.org/>, 2026. Accessed: 2026-05-15.
- 9 Daniela Damian, Kelly Blincoe, Denae Ford, Alexander Serebrenik, and Zainab Masood. *Equity, Diversity, and Inclusion in Software Engineering: Best Practices and Insights*. Springer, 2024.
- 10 DFTB+ Developers Group. Dftb+: Software package for efficient atomistic simulations. <https://www.dftbplus.org/>, 2026. Accessed: 2026-05-15.
- 11 Akash Dhruv and Anshu Dubey. Leveraging large language models for code translation and software development in scientific computing. In *Proceedings of the Platform for Advanced Scientific Computing Conference*, pages 1–9, 2025. doi:10.1145/3732775.3733572.
- 12 Roshanak Farhoodi, Vahid Garousi, Dietmar Pfahl, and Jonathan Sillito. Development of scientific software: A systematic mapping, a bibliometrics study, and a paper repository. *International Journal of Software Engineering and Knowledge Engineering*, 23(04):463–506, 2013. doi:10.1142/S0218194013500137.
- 13 Michael García-Rodríguez, Juan A Añel, and Javier Rodeiro-Iglesias. Assessing and improving the quality of fortran code in scientific software: Fortrananalyser. *Software Impacts*, 21:100692, 2024. doi:10.1016/J.SIMPA.2024.100692.
- 14 Jeremiah Giordani, Ziyang Xu, Ella Colby, August Ning, Bhargav Reddy Godala, Ishita Chaturvedi, Shaowei Zhu, Yebin Chon, Greg Chan, Zujun Tan, et al. Revisiting computation for research: Practices and trends. In *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–14. IEEE, 2024.
- 15 Wilhelm Hasselbring, Reiner Jung, and Henning Schnoor. Software architecture evaluation of earth system models. *Journal of Software Engineering and Applications*, 18(3):113–138, 2025.
- 16 Laurence J Kedward, Balint Aradi, Ondřej Čertík, Milan Curcic, Sebastian Ehlert, Philipp Engel, Rohit Goswami, Michael Hirsch, Asdrubal Lozada-Blanco, Vincent Magnin, et al. The state of fortran. *Computing in Science & Engineering*, 24(2):63–72, 2022. doi:10.1109/MCSE.2022.3159862.
- 17 Lawrence Berkeley National Laboratory. Berkeley lab github repositories. <https://github.com/berkeleylab/>, 2026. Includes Caffeine, Fiats, Julienne, and Assert repositories. Accessed: 2026-05-15.
- 18 Steve Lionel. Revised strategic plans for wg5. <https://wg5-fortran.org/N2201-N2250/N2220.txt>, 2023. ISO/IEC JTC1/SC22/WG5 document N2220, accessed 2026-05-15.
- 19 Michael Metcalf. The seven ages of fortran. *Journal of Computer Science and Technology*, 11(01):1–8, 2011.
- 20 Jefferson Seide Molléri, Kai Petersen, and Emilia Mendes. An empirically evaluated checklist for surveys in software engineering. *Information and Software Technology*, 119:106240, 2020. doi:10.1016/J.INFSOF.2019.106240.
- 21 Molpro Development Team. Molpro quantum chemistry package. <https://www.molpro.net/>, 2026. Accessed: 2026-05-15.
- 22 National Center for Atmospheric Research. Community earth system model (cesm). <https://www.cesm.ucar.edu/>, 2026. Accessed: 2026-05-15.
- 23 NEMO System Team. Nemo ocean engine. <https://github.com/NEMO-ocean>, 2026. Accessed: 2026-05-15.

- 24 Nikolaos Nikolaidis, Dimitrios Zisis, Apostolos Ampatzoglou, Alexander Chatzigeorgiou, and Dimitrios Soudris. Experience with managing technical debt in scientific software development using the exa2pro framework. *IEEE Access*, 9:72524–72534, 2021. doi:10.1109/ACCESS.2021.3079271.
- 25 ParaMonte Library Developers. Paramonte: Parallel monte carlo library. <https://github.com/cdslaborg/paramonte>, 2026. Accessed: 2026-05-15.
- 26 Prakash Prabhu, Thomas B. Jablin, Arun Raman, Yun Zhang, Jialu Huang, Hanjun Kim, Nick P. Johnson, Feng Liu, Soumyadeep Ghosh, Stephen Beard, Taewook Oh, Matthew Zoufaly, David Walker, and David I. August. A survey of the practice of computational science. In *State of the Practice Reports*, SC '11, New York, NY, USA, 2011. Association for Computing Machinery. doi:10.1145/2063348.2063374.
- 27 Damian Rouson. Task parallelism in fortran. *UC Berkeley Electronic Scholarship Repository*, 2021. URL: <https://escholarship.org/uc/item/3cc8373m>.
- 28 Damian WI Rouson, Helgi Adalsteinsson, and Jim Xia. Design patterns for multiphysics modeling in fortran 2003 and c++. *ACM Transactions on Mathematical Software (TOMS)*, 37(1):1–30, 2010. doi:10.1145/1644001.1644004.
- 29 SENG Development Team. Senga2. <https://www.ukctrf.com/index.php/senga/>, 2026. Accessed: 2026-05-15.
- 30 Galen M Shipman and Timothy C Randles. An evaluation of risks associated with relying on fortran for mission critical codes for the next 15 years. Technical report, Los Alamos National Laboratory (LANL), Los Alamos, NM (United States), 2023.
- 31 Anton Shterenlikht. Benefits of continuing fortran standardisation survey: final report, January 2019. Accessed 12/5/2026. URL: https://fortran.bcs.org/2019/FortranBenefitsSurvey_final.pdf.
- 32 UK Met Office. Unified model. <https://www.metoffice.gov.uk/research/approach/modeling-systems/unified-model>, 2026. Accessed: 2026-05-15.
- 33 University of Tennessee and University of California Berkeley and University of Colorado Denver. Lapack – linear algebra package. <https://www.netlib.org/lapack/>, 2026. Accessed: 2026-05-15.
- 34 W Van Snyder. Scientific programming in fortran. *Scientific Programming*, 15(1):3–8, 2007. doi:10.1155/2007/930816.
- 35 Wim Vanderbauwhede. Making legacy fortran code type safe through automated program transformation. *The Journal of Supercomputing*, 78(2):2988–3028, 2022. doi:10.1007/S11227-021-03839-9.
- 36 VASP Development Team. Vienna ab initio simulation package (vasp). <https://www.vasp.at/>, 2026. Accessed: 2026-05-15.
- 37 WRF Development Team. Weather research and forecasting (wrf) model. <https://www2.mmm.ucar.edu/wrf/users/>, 2026. Accessed: 2026-05-15.

A List of survey questions reported in this paper

Table 6 concisely lists the survey questions reported in the main paper, to support cross-referencing with the questions in the full survey, provided in the Zenodo repository.

B Data Availability Statement

The survey data and associated materials used in this study are publicly available via Zenodo at <https://doi.org/10.5281/zenodo.20185149>. The repository provides persistent archival access to the dataset and associated metadata in accordance with the open science and reproducibility expectations of ESEM/ESEIW submissions. The repository includes the raw survey data for the questions explored in this paper, including categorised/cleaned data,

16:20 What Do We Really Know About Fortran?

■ **Table 6** Summary of the survey questions.

Q#	Question
Q2	How many Fortran codebases do you work on, or with, e.g., you change or test the code?
Q3	Tell us your reasons for deciding on that codebase
Q4	What is the type of codebase?
Q6	Approximately, how many people currently work on the codebase, e.g., change or test the code, in contrast to people who use the codebase?
Q7	What is the approximate size of the current codebase?
Q8	In subjective terms, how would you characterise the size of the codebase?
Q20	Do you personally consider the codebase to be a legacy codebase?
Q21	Do others you work with consider the codebase to be a legacy codebase?
Q23	What is the most significant challenge - technical or non-technical - you personally confront with the codebase?
Q25	In what country is the organisation you work for located?
Q26	What type of organisation “owns” the codebase?
Q27	What is your main role in the organisation that “owns” the codebase?
Q31	What is your age?
Q32	What is your gender?
Q33	What is your highest academic qualification?
Q34	How relevant is your highest qualification to the work you do on, or with, this codebase?

as well as PDFs of the survey questions, consent forms, instructions and advertisement. Data analysis scripts which can be used to produce the figures in this paper are also provided, as well as hi-res copies of all figures.

The survey was publicly promoted at: <https://fortran-lang.discourse.group/t/please-participate-global-survey-on-the-state-and-impact-of-fortran/10620>