Performance Comparisons of Programs in Different Programming Languages Converted from Legacy Finite Element Codes

Thiruchelvam Arudchelvam¹ and Samuel Ratnajeevan Herbert Hoole²

1. Department of Computing and Information Systems, Faculty of Applied Sciences, Wayamba University of Sri Lanka, Kuliyapitiya 60200, Sri Lanka
2. Department of Electrical and Computer Engineering, Michigan State University, East Lansing, MI 48824-1226, USA

Abstract: The FORTRAN programming language was used in early days of writing finite element field computation to write programs. Much of those codes were developed in an ad hoc way. Today, modern software developers face problems in understanding, modifying and utilizing those codes. As modern software engineers are very concerned with object oriented design, if those codes are converted into an object oriented language, they could be redesigned and deployed in an object oriented system. Those legacy codes often need to be converted not only into an object oriented programming language such as Java but also into functional oriented languages such as C. Conversion of those legacy codes into modern languages gives many advantages. The purpose of this paper is to compare the performances of such converted legacy finite element codes originally written in FORTRAN, the relevant converted C program and Java program. Sample finite element programs written in FORTRAN are converted for purposes of comparison into modern languages such as C and Java. Performances are compared based on the execution time. In addition to that, the memory sizes of the execution file of FORTRAN and C programs are also compared. Java being interpretive there is no execution file to compare.

Key words: Finite element method, legacy software, performance evaluation, re-engineering.

1. Legacy Finite Element Code

The development of code for finite elements-based field computation has been going on at a pace since 1970s. Today, we have legacy codes running into millions of lines, implemented without planning and not using proper state-of-the-art software design tools [1, 2]. In the old days, often a professor was requested to solve a problem. He/she thereupon developed codes with his or her research team. Then, new researchers were also asked to use those codes to continue that work. There were some threats in that approach. If the professor who knows every nook and cranny of the project leaves the team, the whole project should thoroughly be studied by others in order to modify, expand or even utilize those codes for some other purposes. Further, supposing that someone wants to choose better methods, a better programming language or a better user interface, he needs for a particular program. He then needs to understand the available FORTRAN program to do what he has to do [3]. Therefore, there is a need to convert those codes available in the FORTRAN language into some modern programming languages. Then, it would be easier for modern software developers and engineers to solve engineering problems. Refs. [4, 5] describe the use of converting those legacy finite element codes into an object oriented model. Nanjundiah and Sinha [6] had enhanced FORTRAN codes into another later version of FORTRAN codes. Therefore, that work is just an enhancement of existing codes. Rather than enhancing, those codes should have been converted into an object oriented model so that it can ever be used...
in future. Therefore, we suggest applying the concept of reverse engineering in order to produce an object oriented model [1].

2. Re-engineering FORTRAN

One of the most important aspects of reverse engineering would involve means of using legacy code written originally in FORTRAN since much of the legacy code is in FORTRAN.

Now facilities are available to convert FORTRAN to C and Java [7, 8] and even from C to Java. There are some issues in the conversion of FORTRAN codes into Java codes. These are:

(1) Classes cannot directly be created from FORTRAN code using forward engineering. Therefore, to make it possible, FORTRAN code should be converted/translated first into Java code and thereafter those classes can be created, using reverse engineering;

(2) Earlier FORTRAN programs were written based on a functional approach to programming rather than an object oriented approach. Therefore, even if FORTRAN code is converted to Java code, it will not be in object oriented design;

(3) Therefore, if the FORTRAN codes are converted into Java code first, then that converted Java program may be easily modified to make it object oriented. Java codes could also be easily understood by modern software developers. Therefore, software developers can easily understand how the problem was solved in the original FORTRAN coding and how that program can be enhanced using the modern developments available in the new programming concepts, facilities and software. Further, if that FORTRAN program had been written in an ad-hoc manner, it can be generalised and widely used for similar problems. Then finally, the object oriented java program can be developed. Once the object oriented program is available in Java, class diagrams can be created by applying reverse engineering [1, 5, 7]. Then, these class diagrams could ever be used to create an object oriented program in any object oriented programming language.

![Diagram](image1.png)

**Fig. 1** Creation of a class diagram for a legacy FORTRAN program.

![Diagram](image2.png)

**Fig. 2** Conversion of object oriented program from one programming language to another.

The above scenario can be depicted as shown in Fig. 1. Further, reverse engineering and forward engineering can be used to convert an object oriented program from one object oriented programming language to a program in another object oriented programming language (Fig. 2).

It is supposed that there is an object oriented program in Java. Then the class diagram of that program can be created by applying reverse engineering. That class diagram can then be used to create an object oriented program in any object oriented program (for example C++) by applying forward engineering.
3. Conversion of Finite Element Code from FORTRAN to C and Java

After a finite element program written in FORTRAN has been converted into the C and Java languages, our algorithm shown in Fig. 3 is used by the authors to compare the performances of those programs in all three programming languages.

The number of mesh points is increased up to 1,000 and more if necessary in the above algorithm. Therefore, the same program is considered like several tasks for the programs by changing the number of mesh points. Then statistical analysis is carried out to compare the performances of those programs.

Further, the same algorithm is applied to three different finite element programs written in FORTRAN and the statistical analysis is carried out for comparing performances of those programs.

4. Statistical Analysis

Graphs of execution times of FORTRAN, C and Java programs versus number of mesh points of programs 1, 2 and 3 are shown in Figs. 4, 5 and 6.

Hypothesis tests to check performances of all three programs in FORTRAN, C and Java are done.

Hypothesis Test for Program 1

Two-Sample T-Test and CI [9-11]: FORTRAN, C

<table>
<thead>
<tr>
<th></th>
<th>N</th>
<th>Mean</th>
<th>StDev</th>
<th>SE mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORTRAN</td>
<td>14</td>
<td>0.0820</td>
<td>0.0834</td>
<td>0.022</td>
</tr>
<tr>
<td>C</td>
<td>14</td>
<td>0.0766</td>
<td>0.0779</td>
<td>0.021</td>
</tr>
</tbody>
</table>

where, \( N \)—number of programs, Mean—mean value, StDev—standard deviation, SE mean—standard error of the mean.

\[
\text{Difference} = \mu (\text{FORTRAN}) - \mu (C)
\]

Estimate for difference: 0.005429
95% CI for difference: (-0.057366, 0.068223)

Hypothesis Test

\( H_0 \)—Execution time of FORTRAN = Execution time of C

\( H_1 \)—Execution time of FORTRAN \(<>\) Execution time of C

T-Test of difference = 0 (vs. not =):

\[
\text{T-Value} = 0.18; \ P-Value = 0.860; \ DF = 25 \quad (1)
\]

In this two-sample T-Test, it was checked whether the performances of FORTRAN and C programs are equal or not. From Eq. (1), since the P-Value is greater than 0.05, it can be concluded that the performances of both FORTRAN and C programs are equal.

Two-Sample T-Test and CI: FORTRAN, Java

<table>
<thead>
<tr>
<th></th>
<th>N</th>
<th>Mean</th>
<th>StDev</th>
<th>SE mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORTRAN</td>
<td>14</td>
<td>0.0820</td>
<td>0.0834</td>
<td>0.022</td>
</tr>
<tr>
<td>Java</td>
<td>14</td>
<td>0.297</td>
<td>0.159</td>
<td>0.043</td>
</tr>
</tbody>
</table>

where, \( N \)—number of programs, Mean—mean value, StDev—standard deviation, SE mean—standard error of the mean.

\[
\text{Difference} = \mu (\text{FORTRAN}) - \mu (\text{Java})
\]

Estimate for difference: -0.214500
95% CI for difference: (-0.315122, -0.113878)

Fig. 3 Algorithm followed to compare execution times of programs in FORTRAN, C and Java.

Fig. 4 Graph of execution times of FORTRAN, C and Java programs vs. number of mesh points of Program 1.
Performance Comparisons of Programs in Different Programming Languages
Converted from Legacy Finite Element Codes

Fig. 5  Graph of execution times of FORTRAN, C and Java programs vs. number of mesh points of Program 2.

Fig. 6  Graph of execution times of FORTRAN, C and Java programs vs. number of mesh points of Program 3.

Hypothesis Test

H₀—Execution time of FORTRAN = Execution time of Java
H₁—Execution time of FORTRAN < Execution time of Java
T-Test of difference = 0 (vs. <):
T-Value = -4.46; P-Value = 0.000; DF = 19   (2)

From Eq. (2), since the P-Value is less than 0.05, it can be claimed that the execution time of FORTRAN is significantly less than the execution time of the converted Java program. Therefore, it can be further concluded that the performance of FORTRAN program is better than that of the converted Java program.

Two-Sample T-Test and CI: C, Java

N  Mean  StDev  SE mean
C    14  0.0766  0.0779  0.021
Java 14  0.297   0.159  0.043

where, N—number of programs, Mean—mean value, StDev—standard deviation, SE mean—standard error of the mean.

Difference = μ (C) – μ (Java)

Estimate for difference: -0.219929
95% CI for difference: (-0.319527, -0.120330)

Hypothesis Test

H₀—Execution time of C = Execution time of Java
H₁—Execution time of C < Execution time of Java
T-Test of difference = 0 (vs. <):
T-Value = -4.64; P-Value = 0.000; DF = 18   (3)

From Eq. (3), since the P-Value is less than 0.05, it can be claimed that the execution time of the C program is significantly less than that of the converted Java program.

Hypothesis Test for Program 2

Two-Sample T-Test and CI: FORTRAN, C_Prog

N  Mean  StDev  SE mean
FORTRAN 15 0.385 0.372 0.096
C_Prog  15 0.379 0.370 0.096

Hypothesis Test

H₀—Execution time of FORTRAN = Execution time of C
H₁—Execution time of FORTRAN <> Execution time of C

Difference = μ (FORTRAN) – μ (C_Prog)

Estimate for difference: 0.006467
95% CI for difference: (-0.271580, 0.284513)
T-Test of difference = 0 (vs. not =):
T-Value = 0.05; P-Value = 0.962; DF = 27   (4)

In this two-sample T-Test, it was checked whether the performances of FORTRAN and C programs are equal or not. From Eq. (4), since the P-Value is greater than 0.05, it can be concluded that the performances of both programs are statistically equal.

Two-Sample T-Test and CI: FORTRAN, Java

N  Mean  StDev  SE mean
FORTRAN 15 0.385 0.372 0.096
Java  15 0.626 0.461 0.12
Hypothesis Test 1

H₀—Execution time of FORTRAN = Execution time of Java
H₁—Execution time of FORTRAN <> Execution time of Java

Difference = μ(FORTRAN) − μ(Java)
Estimate for difference: -0.240933
95% CI for difference: (-0.555517, 0.073650)
T-Test of difference = 0 (vs. not =):
T-Value = -1.57; P-Value = 0.128; DF = 26
From Eq. (5), since the P-Value is greater than 0.05, it can be concluded that the execution time of FORTRAN is equal to that of the converted Java program.

Two-Sample T-Test and CI: C_Prog, Java

<table>
<thead>
<tr>
<th>N</th>
<th>Mean</th>
<th>StDev</th>
<th>SE mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>C_Prog</td>
<td>15</td>
<td>0.379</td>
<td>0.370</td>
</tr>
<tr>
<td>Java</td>
<td>15</td>
<td>0.626</td>
<td>0.461</td>
</tr>
</tbody>
</table>

Difference = μ(C_Prog) − μ(Java)
Estimate for difference: -0.247400
95% CI for difference: (-0.561160, 0.066360)
T-Test of difference = 0 (vs. not =):
T-Value = -1.62; P-Value = 0.117; DF = 26
From Eq. (6), since P-Value is greater than 0.05, it can be told that the execution time of converted C program is equal to the execution time of converted Java.

Hypothesis Test for Program 3

Two-Sample T-Test and CI: FORTRAN, C

Hypothesis Test

H₀—Execution time of FORTRAN = Execution time of C
H₁—Execution time of FORTRAN <> Execution time of C

Difference = μ(FORTRAN) − μ(C)
Estimate for difference: 0.010571
95% CI for difference: (-0.595806, 0.616949)
T-Test of difference = 0 (vs. not =):
T-Value = 0.04; P-Value = 0.972; DF = 25
In this two-sample T-Test, it was checked whether the performances of FORTRAN and C programs are equal or not. From Eq. (7), since the P-Value is greater than 0.05, it can be concluded that the performances of both programs are equal.

Two-Sample T-Test and CI: FORTRAN, Java

<table>
<thead>
<tr>
<th>N</th>
<th>Mean</th>
<th>StDev</th>
<th>SE mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORTRAN</td>
<td>14</td>
<td>0.872</td>
<td>0.782</td>
</tr>
<tr>
<td>Java</td>
<td>14</td>
<td>1.41</td>
<td>1.08</td>
</tr>
</tbody>
</table>

Hypothesis Test 1

H₀—Execution time of FORTRAN = Execution time of Java
H₁—Execution time of FORTRAN <> Execution time of Java

Difference = μ(FORTRAN) − μ(Java)
Estimate for difference: -0.539929
95% CI for difference: (-1.277229, 0.197372)
T-Test of difference = 0 (vs. not =):
T-Value = -1.51; P-Value = 0.143; DF = 23
As in the previous case, in this two-sample T-Test, it was checked whether the performances of FORTRAN and converted Java programs are equal or not. From Eq. (8), since the P-Value is greater than 0.05, it can be concluded that the performances of both programs are equal.

Two-Sample T-Test and CI: C, Java

<table>
<thead>
<tr>
<th>N</th>
<th>Mean</th>
<th>StDev</th>
<th>SE mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>14</td>
<td>0.862</td>
<td>0.776</td>
</tr>
<tr>
<td>Java</td>
<td>14</td>
<td>1.41</td>
<td>1.08</td>
</tr>
</tbody>
</table>

Hypothesis Test

H₀—Execution time of C = Execution time of Java
H₁—Execution time of C <> Execution time of Java

Difference = μ(C Exec time) − μ(Java Exec time)
Estimate for difference: -0.550500
95% CI for difference: (-1.285894, 0.184894)
T-Test of difference = 0 (vs. not =):
T-Value = -1.55; P-Value = 0.135; DF = 23
From Eq. (10), since the P-Value is greater than 0.05,
it can be concluded that the performances of both programs are equal.

The following results are obtained when the average values of all the above are taken and the same statistical tests are done:

(1) Execution time of FORTRAN program and the execution time of converted C program are equal;

(2) Execution time of FORTRAN program and the execution time of converted Java program are equal;

(3) Execution time of converted C program and the execution time of converted Java program are equal.

In addition to the execution time, memory size is also considered as a factor which determines the performances of programs. As such, the memory sizes of the execution files of FORTRAN program and the relevant converted C program are taken into consideration for this hypothesis test. For this test, the converted Java program could not be used because Java programs have the class file and not the execution files. The memory sizes of the execution files of FORTRAN and C programs are compared in Fig. 7a. There are two dips in the graph. But after that, it increases as expected. Logically speaking, those dips should not have been seen in the graph. The exact reason for this could not be found although we double-checked that they are indeed present in the results.

To check whether the results are dependent on the behaviour and architecture of computers, the same test was carried out on another desktop computer and the resultant graph is shown in Fig. 7b. It should also be noted that one laptop and a desktop have been used and in both computers, Ubuntu has been used as the operating system for this purpose. It is noted that, though the exact values of memory sizes are not the same, the shape is same. There are similarities in shapes and differences in the values. This is also suggested for further analysis.

From the statistical analysis, the results obtained, for Program 1, are that memory size of FORTRAN program is significantly equal to the memory size of the relevant converted C program.

For the Program 2, the same test is also done. There is also a dip when the number of mesh points is 500. The same test is done in another machine and the same shape is obtained as in Program 1. The memory sizes of execution files of FORTRAN and the relevant converted C program of Program 2 is given in Fig. 8.

From the above statistical test, it is found that the memory size of the execution file of FORTRAN program of Program 2 is significantly less than that of the relevant converted C program. Therefore, based on the memory size of execution files, the performance of the FORTRAN program is significantly greater than the performance of the relevant converted C program.

Similarly for Program 3, it is found that based on memory size of execution files, performance of FORTRAN program is significantly greater than that of relevant converted C program and the relevant graph
Performances of FORTRAN and the relevant converted C and Java programs. The memory size of the execution files and the execution times are considered as two factors of the performance indicators. Based on these two factors, the performances of those programs are compared. Accordingly, we tested the hypothesis that the execution times are the same. Our finding is that, on average, the performances of the finite element FORTRAN program, the relevant converted C program and the relevant converted Java program are statistically equal. Further, according to our hypothesis tests, the memory sizes of the execution files of the FORTRAN program are statistically less than that of the relevant C program. The memory test was not relevant to the Java program.

Acknowledgments

Mr. Thiruchelvam Arudchelvam thanks his employer, Wayamba University of Sri Lanka, for the support rendered to carry out this research; University Grants Commission-Sri Lanka for financial assistance to publish this paper; and Ms. W.M.P.M. Wickramasinghe and Ms. P.A.D.A.N. Appuhamy for the support rendered for the statistical analysis of this paper.

References


