

METHODOLOGY

Open Access



Application of artificial intelligence to revive numerical studies of fluid motion in a curved pipe

Nils Tångefjord Basse^{1*}

*Correspondence:

Nils Tångefjord Basse
nils.basse@ri.se

¹RISE Research Institutes of Sweden,
Brinellgatan 4, 504 62 Borås,
Sweden

Abstract

This paper presents an artificial intelligence (AI) assisted revival of FORTRAN 66 code which calculates laminar flow through curved pipes. Dean numbers D for the entire laminar range are covered: $10 \leq D \leq 5000$. Results from the code were originally presented in Greenspan (Secondary flow in a curved tube. *J Fluid Mech* 57: 167–176, 1973). The coupled non-linear system of partial differential equations was solved numerically using a finite difference method. We demonstrate a step-by-step AI-assisted code revival process and compare original (coarse grid resolution) results to updated (fine grid resolution) solutions. Both the structure of streamwise (primary) and secondary flows are covered. The purpose of our paper is both to demonstrate the revival of legacy scientific code and to make the code available as modern Fortran for the scientific community. The code runs quickly on modern hardware architectures and enables fast understanding of the physical effects included.

Keywords Fluid motion in a curved pipe, Laminar flow, Code revival, Artificial intelligence, FORTRAN 66, Modern Fortran

1 Introduction

The theoretical treatment of laminar flow in curved pipes was pioneered by Dean in the late 1920 s [1, 2]. He showed that in addition to a displacement of streamwise flow, a secondary flow also exists, taking the form of two vortices, the so-called Dean vortices. A constant D , the Dean number, was identified as the key parameter:

$$D = 4Re\sqrt{\frac{2a}{L}}, \quad (1)$$

where Re is the bulk Reynolds number, a is the pipe radius and L is the radius of curvature of the curved pipe. Dean [1] obtained simplified equations of motion for the approximation $a/L \ll 1$, which can be interpreted as $a/L \leq 0.01$ [2].

Dean [1] obtained theoretical solutions to the equations for the range $0 \leq D \leq 96$ and following this, McConalogue and Srivastava expanded the solutions numerically



to $96 \leq D \leq 605.72$ [3]. In this paper we treat a subsequent study by Greenspan and Schubert [4, 5], where numerical solutions were extended to $D \leq 5000$. This extension was done with a FORTRAN 66 code which is available as an Appendix in [4]. Note that we also cover a later paper, where improved numerical solutions over the same range of D were published [6].

The purpose of our paper is to convert - and thus revive - the code presented in [4] to modern Fortran. There is a multitude of reasons for this revival, the most important ones being:

1. To make the artificial intelligence (AI) assisted legacy code revival methodology available to the scientific community for e.g. educational/ pedagogical purposes.
2. To make the revived code itself accessible as the tangible result of this paper.
3. To understand and explain differences between numerical schemes such as those in [4, 5] and [6].
4. To use the code as a first step to recreate the results in [6].
5. As discussed in the Conclusions: (i) To address historical aspects and (ii) to use the code for our planned future research.

This flow system - laminar flow in slightly curved pipes - can be considered as belonging to the class of canonical laminar flows, which remains relevant for both applied and fundamental research. Additional details on this topic until the early 1980 s can be found in a review paper [7].

The Dean number of 5000 is from [8], where experiments determined the critical Re to be 5000 for $L/a = 31.9$.

On a personal note, I have previous experience writing and running FORTRAN 77 code on an IBM mainframe at JET [9] during the period 1997-1998. Thus, the task at hand, to convert FORTRAN 66 code, was feasible.

The paper is structured as follows: In Sect. 2, we briefly summarise the equations of motion. This is followed by the FORTRAN 66 AI-assisted code revival steps which are detailed in Sect. 3; results from the revived code are presented in Sect. 4. A discussion is presented in Sect. 5 and finally we conclude in Sect. 6. The paper thus encompasses the complete path from theory through simulations to results and disseminates the accompanying modernised Fortran code.

2 Equations of motion

The equations of motion, also called the Dean equations, can be written [3-5]:

$$\nabla_1^2 w + D = \frac{1}{r} \left(\frac{\partial \phi}{\partial \alpha} \frac{\partial w}{\partial r} - \frac{\partial \phi}{\partial r} \frac{\partial w}{\partial \alpha} \right) \quad (2)$$

$$-\nabla_1^4 \phi = \frac{1}{r} \left(\frac{\partial \phi}{\partial r} \frac{\partial}{\partial \alpha} - \frac{\partial \phi}{\partial \alpha} \frac{\partial}{\partial r} \right) \nabla_1^2 \phi + w \left(\frac{\partial w}{\partial r} \sin \alpha + \frac{\partial w}{\partial \alpha} \frac{\cos \alpha}{r} \right), \quad (3)$$

where:

$$\nabla_1^2 = \frac{\partial^2}{\partial r^2} + \frac{1}{r} \frac{\partial}{\partial r} + \frac{1}{r^2} \frac{\partial^2}{\partial \alpha^2} \quad (4)$$

Here, ϕ is the non-dimensional stream function for the secondary flow, w is the non-dimensional velocity in the streamwise direction, r is the normalised pipe radius and α is the angle centered on the pipe axis.

To derive the Dean equations, it has been assumed that $a/L \ll 1$ and that the motion is steady (laminar).

In [3], w and ϕ were rewritten as Fourier series and the resulting coupled non-linear equations were solved numerically. In contrast, in [4, 5], the Dean equations were rewritten to a system of second order equations to facilitate solving the equations using finite differences. This finite difference method was improved upon in [6] by the application of an additional step to ensure "that the final solutions are correct to central-difference accuracy". See the discussion in Sect. 5.1 for more details.

3 Code revival

There were three aspects to the code revival, and they are treated in the three subsections below:

1. Digitise the scanned document
2. Convert the FORTRAN 66 code to modern Fortran
3. Debugging and testing of the code

The final two steps were both done twice to create two modern Fortran versions, an intermediate minimal modern Fortran version and a complete modern Fortran version.

3.1 Digitisation

We note that the original FORTRAN 66 code is only available in [4] and not available in the official journal paper [5]. As can be seen, the scanned document [4] has a quite poor quality, making it challenging to perform the optical character recognition (OCR) needed. However, with 12 pages (~600 lines) of code, it would be very cumbersome to retype the code entirely by hand.

We did not attempt to improve the quality of the scanned document before conversion. This decision was mainly due to a lack of knowledge of available tools, but a step which should be evaluated to improve our methodology.

Our initial attempts at extracting characters from the scanned document were done using Python with EasyOCR [10] and PyTesseract [11]. However, these tools performed poorly on the document.

Our next attempt was with two online converters [12, 13]. The results were better than for the Python tools, but still with a correct percentage of around 50%.

The best conversion, with an average success rate of roughly 80%, was the Amazon Textract tool from Amazon Web Services (AWS) [14].

In the end, we used the AWS tool to generate the initial digital file and combined it with the output from the two online converters. And finally, a human correction was needed for around 10% of the code.

The original FORTRAN 66 code is provided for reference in Appendix A.

3.2 Minimal modern Fortran

After digitisation, we created an intermediate minimal modern Fortran version with the objective of creating the same output as the original FORTRAN 66 code. The minimal

modern Fortran code is mostly compatible with FORTRAN 77, but some features from Fortran 90 are used, e.g. in the file handling.

3.2.1 Conversion

Once the code was available in digital form, the main conversion from FORTRAN 66 to minimal modern Fortran was to replace the Hollerith string and data notation [15]. As part of achieving this goal, we used ChatGPT-4 [16] to convert the Hollerith commands to modern Fortran. An example of the process can be found in [17].

3.2.2 Debugging and testing

At this stage we had a complete minimal modern Fortran code available, the final step being to debug and test the code.

A laptop with Microsoft Windows 11 [18] installed was chosen as the hardware/software platform. Microsoft Visual Studio [19] was installed as the graphical user interface along with the Intel Fortran Compiler (ifx) [20]. In terms of Fortran versions, ifx uses the latest standards from Fortran 2018 and some Fortran 2023 features.

What remained was to compile (build the executable program) and systematically resolve the errors. Some sections were updated, for example replacing the reading of and writing on punch cards with files. Another major change was to make an IF statement to loop over all D values analysed in [4, 5].

The final result of the steps is the minimal modern Fortran code provided in Appendix B. Changes made to the code in addition to the ones mentioned above have been marked by "NTB 2025" in the code to enable an overview of the changes. These were done in preparation for grid refinement studies and include:

- Increasing MAXSOR and MAXOUT, which control the maximum number of iterations for the successive over-relaxation (SOR) and outer iterations.
- For $D = 250, 2000$ and 5000 : Modifications to the SOR over-relaxation factors compared to the values in Table 1 in [5].
- Calls to the OUTPUT subroutine with NR and NA as inputs (grid resolution, see Sect. 4.2). And a corresponding update to the subroutine to enable the handling of these added inputs.

Appendices A and B can be compared to learn what has been deleted from the original FORTRAN 66 code in the minimal modern Fortran code.

3.3 Complete modern Fortran

Having a working minimal modern Fortran code available, we proceed to create a complete modern Fortran code. This code uses features from "Fortran 90+", which is a combination of Fortran 90 and later standards. An example of a Fortran 2008 feature is NEWUNIT.

Here, we interact with GitHub Copilot (based on ChatGPT-4.1) which is integrated into Microsoft Visual Studio.

3.3.1 Conversion

It was found that the best way to use GitHub Copilot to construct a complete modern Fortran code was to do the changes in smaller steps. A step-by-step overview of the changes is:

1. Conversion from fixed-form to free-form source layout:
 - Change file extension from `.f` to `.f90`
 - New line continuation symbol: `&` instead of `*`
 - New comment symbol: `!` instead of `C`
2. Added the `IMPLICIT NONE` command. A minor error in the code was discovered as a result.
3. Converted `DO` loops to modern Fortran notation.
4. Convert `GO TO` loop (labels 7 to 9) to modern Fortran notation: `DO` and `IF`.
5. Modify `ctr` loop to `DO` and `SELECT CASE` (remove label 5 and `GO TO` 5).
6. Add an `ERROR_MOD` module with subroutines `ERROR_HANDLER` (labels 55-58) and `OUTPUT` (labels 97-99).
7. Change from `ISOR` to `ISOR_PHI`, `ISOR_W` and `ISOR_OMEGA` for improved clarity.
8. Eliminate all `GO TO` usage:
 - Label 10 replaced with `OUTER_ITER DO` loop
 - Labels 14 and 15 replaced with `PHI_SOR DO` loop
 - Labels 26 and 27 replaced with two `DO` loops: `W_RETRY` and `W_SOR`
 - Labels 41 and 42 replaced with two `DO` loops: `OMEGA_RETRY` and `OMEGA_SOR`
9. Split `ERROR_MOD` module into two modules: `ERROR_MOD` and `OUTPUT_MOD`.
10. Create a new `SOR_MOD` module which contains the `SOR_PHI` subroutine.
11. Add the `SOR_W` subroutine in the `SOR_MOD` module.
12. Add the `SOR_OMEGA` subroutine in the `SOR_MOD` module.
13. Replace `AMIN1` with `MIN` and `AMAX1` with `MAX`.
14. Remove `FORMAT` statements and labels which have been replaced by embedded notation.
15. Add new subroutine `SMOOTH` (outer iterate) in module `SOR_MOD`.
16. `DATA` statement removed.
17. `IPCH` (PunCHcard) name changed to `IFIL`.
18. File writing updated:
 - `unit` instead of `hardwired 1`
 - File I/O modernization, dynamic file handling with `NEWUNIT`
19. Where possible: Apply array syntax to initialize arrays to zero (instead of nested loops).
20. Add functionality to calculate the elapsed CPU time using `SYSTEM_CLOCK`.
21. Add new module `KIND_MOD` to define `dp` (kind parameter):
 - Use `_dp` for all (non-integer) numerical constants and `REAL (KIND=dp)` for all `REAL` variables

- Use explicit kinds for all floating-point variables for improving portability and numerical reliability

22. RETURN statements removed, replaced with EXIT and CYCLE.

23. Add LOGICAL : FAILED to track outer iteration convergence (MAXOUT).

24. Correct initialization errors and add initialization of PHI, W and OMEGA.

Throughout the steps above, the following were implemented:

- Updates to the variable definitions
- Consistent indentation and line spacing for improved readability
- Comments rewritten and added
- Deletion of unused variables

The final result of the steps is the complete modern Fortran code provided in Appendix C. The readability of the code was significantly improved, e.g. by moving sections into modules/subroutines and by removing GO TO loops.

3.3.2 Debugging and testing

Debugging and testing was done after each step of the conversion to make sure the results were the same as for the minimal modern Fortran version of the code.

4 Results

All postprocessing in this section was done using GNU Octave [21].

4.1 Evolution of spatial structures

We include two figures with contour plots of ϕ and w , Figs. 1 and 2, to facilitate a direct comparison to the corresponding Figures 3 and 4 in [5]. Only the upper half of the pipe is calculated by the Fortran code, we mirror this to the lower half for clarity of exposition. Note that ϕ is antisymmetric and w is symmetric for the lower half with respect to the upper half. Angles α of 0 (180) degrees indicate the outboard (inboard) positions, respectively.

The results match the original results very well, both in position and magnitude. The results are identical for both the minimal and complete modern Fortran codes in appendices B and C. We use "modern Fortran" as a common term for these two versions of the code.

4.2 Grid resolution improvement

Having the modern Fortran code available enables improvements to e.g. the computational grid resolution. The original resolution was $\Delta r = 0.1$ and $\Delta\phi = \frac{\pi}{18}$. In the code, these resolutions are named NR and NA, respectively. We updated the codes to run with double resolution, both for r and ϕ . To distinguish the grid resolutions, we call them "original" and "updated". The elapsed CPU time scales roughly linearly with the grid size.

We compare original and updated contour plots for ϕ and w in Figs. 3 and 4. The two largest values of D are compared; qualitatively, the results are very similar; as expected, the contours of the updated resolution are more smooth than those of the original resolution.

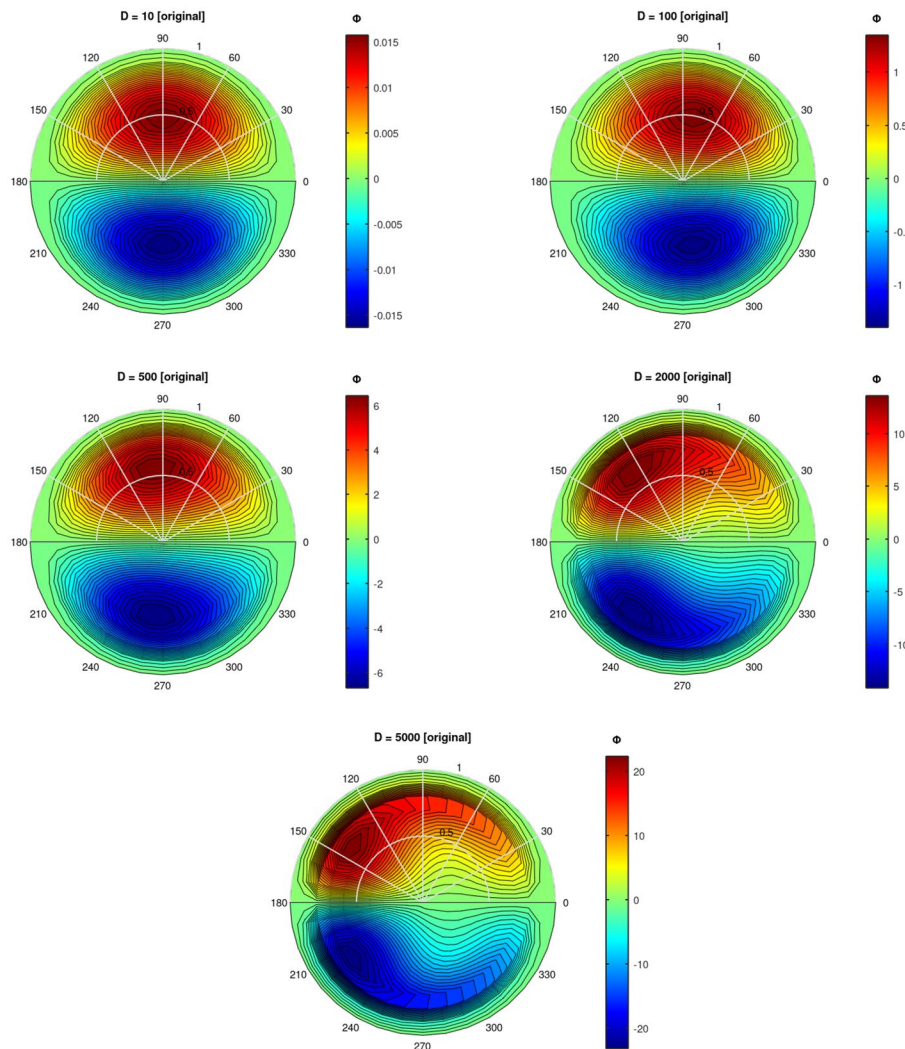


Fig. 1 Original ϕ contour plots for $D = 10, D = 100, D = 500, D = 2000$ and $D = 5000$

4.3 A comparison of scaling results

We compare results from postprocessing of the modern Fortran code with literature findings.

In Fig. 5, we present the D -scaling of the peak values of ϕ and w , which we name ϕ_{\max} and w_{\max} , respectively. Both the original and updated solutions are included, along with results from [3, 6]. The difference between the original and updated solutions is quite small, but there is a substantial difference between these results and those in [3, 6]. Measurements (included in [6]) demonstrate that the values from [3, 6] are more accurate than our results based on [4, 5].

An important effect of the secondary flow is the blockage effect, i.e. that it obstructs the streamwise flow. This is also calculated in the modern Fortran code and shown in Fig. 6 along with values from [6]. Here, the flux (or flow rate) is normalised to that of a straight pipe. The results are consistent with the w_{\max} scaling, i.e. the larger flux from [6] corresponds to a larger w_{\max} , see the right-hand plot in Fig. 5.

A topic of interest for our future research is how w_{\max} moves as a function of D . This is shown in Fig. 7 with values estimated from Figure 4 in [6]. Here, we find a large

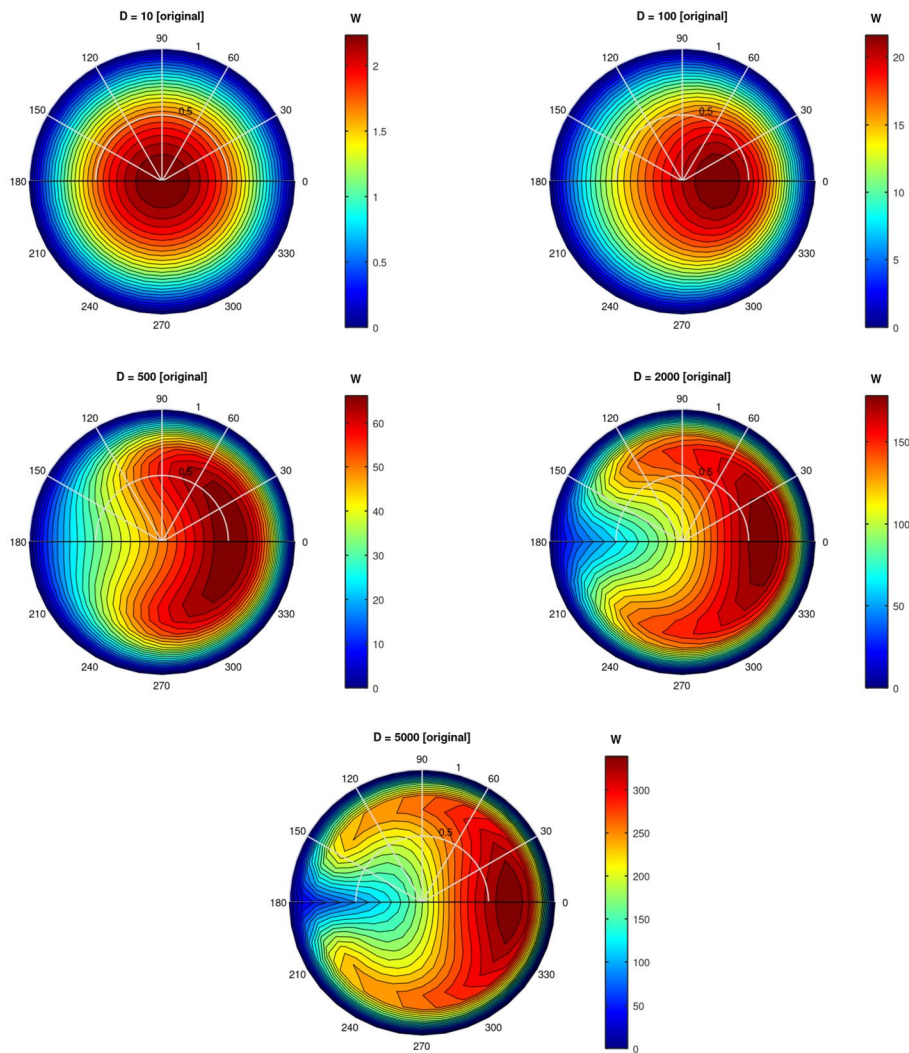


Fig. 2 Original w contour plots for $D = 10$, $D = 100$, $D = 500$, $D = 2000$ and $D = 5000$

difference between the original and updated resolution; the updated values are quite close to the results reported in [6].

5 Discussion

5.1 Finite difference methods

As briefly mention in Sect. 2, finite difference methods are used to solve the equations of motion. In addition to w and ϕ , an equation is also solved for an intermediate variable:

$$\Omega = -\nabla_1^2 \phi, \tag{5}$$

where ∇_1^2 is defined in Equation (4). This Laplacian is calculated using the cross-sectional coordinates (r, α) and the equation is a two dimensional Poisson equation where Ω is the non-dimensional streamwise vorticity. Contour plots of Ω corresponding to Figs. 1 and 2 are presented (using the updated grid resolution) in Fig. 8. As for ϕ , Ω is also antisymmetric with respect to the horizontal plane of the pipe.

Two differencing schemes are used in [4, 5], the central and upwind techniques. Here, we briefly summarise the excellent treatment of these (and other) schemes contained in

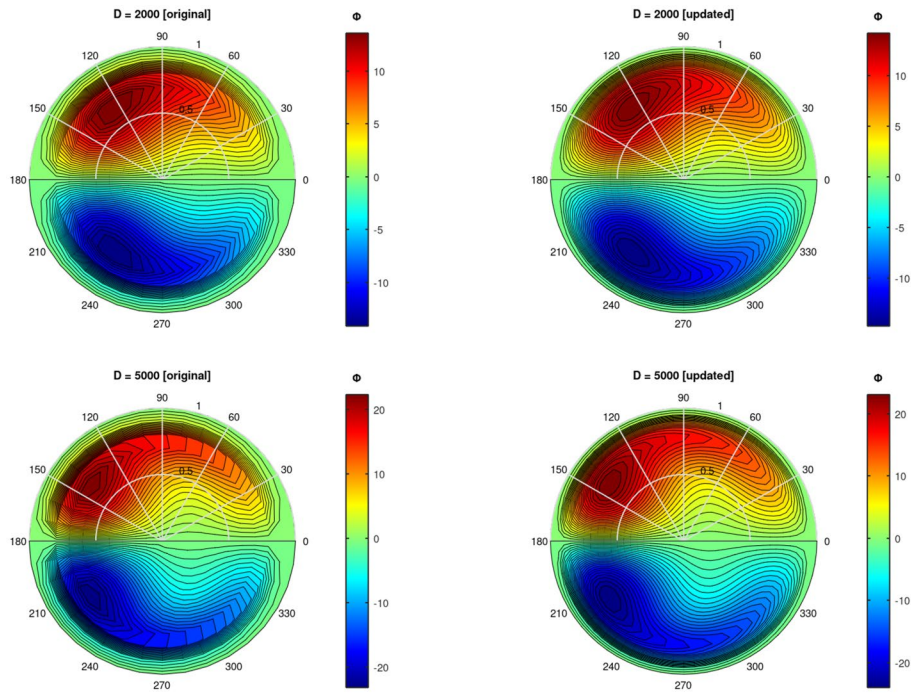


Fig. 3 ϕ contour plots for $D = 2000$ and $D = 5000$. Left-hand column: Original (coarse) resolution, right-hand column: Updated (fine) resolution

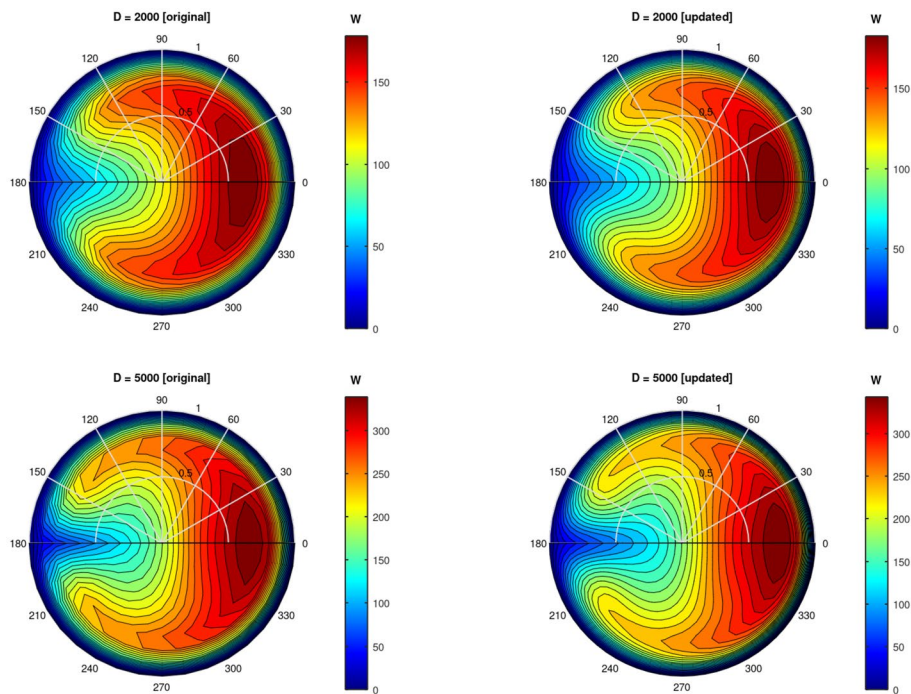


Fig. 4 w contour plots for $D = 2000$ and $D = 5000$. Left-hand column: Original (coarse) resolution, right-hand column: Updated (fine) resolution

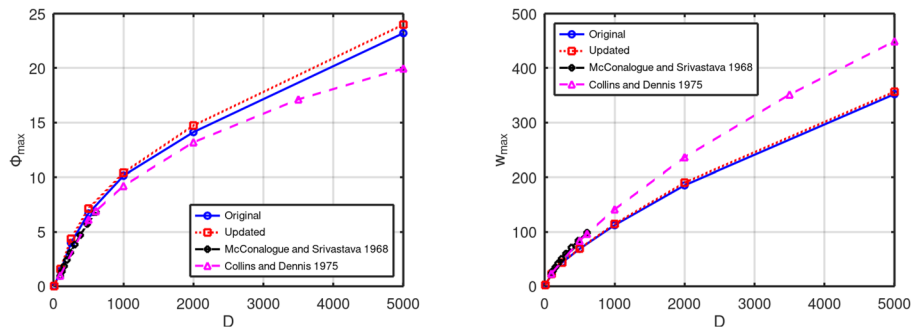


Fig. 5 Left-hand plot: Value of ϕ_{max} as a function of D , right-hand plot: Value of w_{max} as a function of D . Open blue dots are original code results, open red squares are updated code results, filled black dots are from [3] and open magenta triangles are from [6]

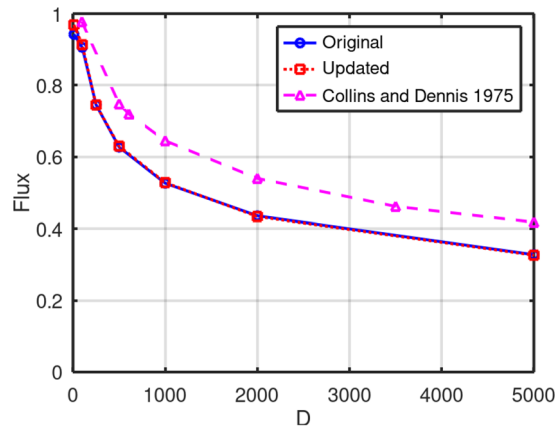


Fig. 6 Streamwise flux as a function of D . Open blue dots are original code results, open red squares are updated code results and open magenta triangles are from [6]

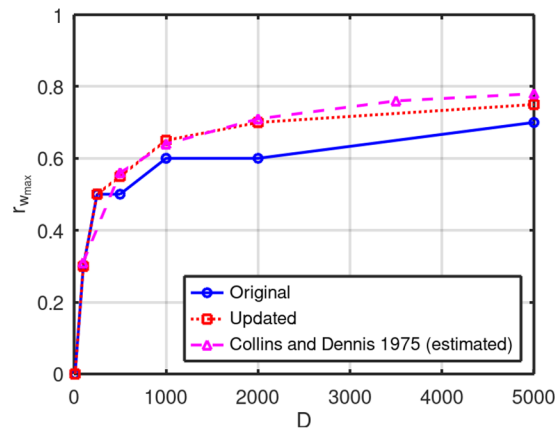


Fig. 7 Normalised radius for maximum streamwise velocity w_{max} as a function of D . Open blue dots are original code results, open red squares are updated code results and open magenta triangles are estimated from Figure 4 in [6]

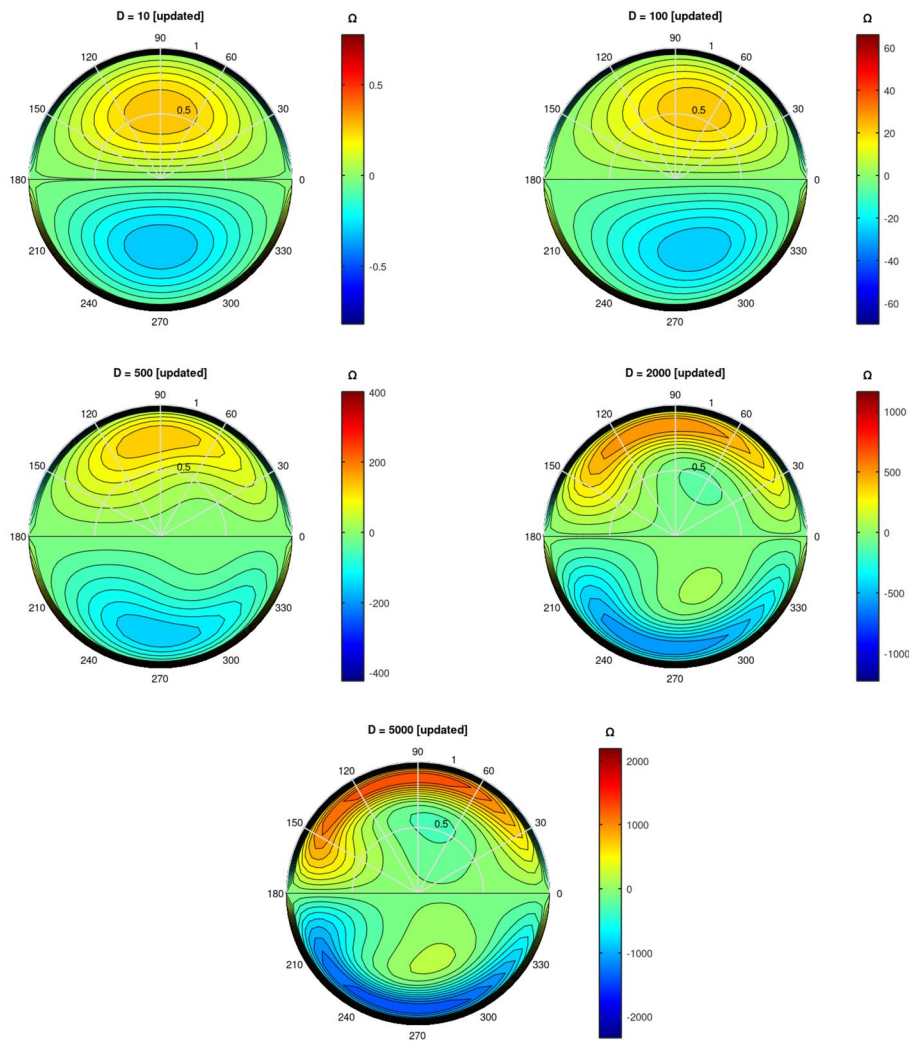


Fig. 8 Updated (fine resolution) Ω contour plots for $D = 10$, $D = 100$, $D = 500$, $D = 2000$ and $D = 5000$

[22].¹ Overall, the upwind scheme accuracy is first order (lower) and the central scheme accuracy is second order (higher). Both schemes have pros and cons which are discussed in detail in [22] using the concepts of conservativeness (of flux), boundedness (diagonally dominant matrix of coefficients) and transportiveness, i.e. the balance between convection and diffusion as defined by the Péclet number. The central differencing scheme is used for ϕ and the upwind differencing scheme is used for w and Ω , which means that the accuracy is a mixture between first and second order.

The approach in [6] consists of two steps, where step one solves the three equations of motion using the central differencing method. To ensure stability, a smoothing technique was applied to Ω . And to ensure convergence, smaller increments of D were used for $D > 1000$. This first step corresponds to the equations solved in [4, 5], but with higher accuracy. A second step is thereafter done in [6], which is to add correction terms C_0 and E_0 to the equations for w and Ω , respectively.

Returning to the comparison of scaling results in Sect. 4.3, the main reason for the large differences observed can be attributed to the fact that the numerical treatment in

¹ Sometimes the central differencing scheme is also known as the linear interpolation scheme [23].

[4, 5] is less accurate than that in [6]. However, the relatively close match between in the position of the maximum streamwise velocity (Fig. 7) is most likely due to the fact that the stream function is calculated with second order accuracy in both cases. The consequence of this is that the primary and secondary flow structures match quite well using the two approaches.

For the modern Fortran code, we have only doubled the original grid resolution, see Sect. 4.2. To check whether the results would continue to improve, we have done tests with higher grid resolution; as mentioned above, for these cases we also had to use smaller increments of D . However, the conclusion from these even finer grids was that no appreciable change was observed compared to the updated (doubled) grid resolution. So the remaining discrepancies with respect to [6] are caused by differences in the accuracy of the numerical schemes. We note that for the original grid resolution, the modern Fortran code calculates solutions for seven values of D in around five seconds. This can be compared to the statement in [5] that "[...] no case required more than three minutes of computing time."

5.2 Lessons learned

We collect lessons learned and recommendations for the scientific community here.

Digitisation of the scanned document was not 100% successful, but rather in the 80% range for the best tools identified. However, the digitisation could probably be improved by preprocessing the document before digitisation. For future work, we would start by identifying and evaluating preprocessing tools.

Conversion to minimal modern Fortran was done using manual prompting of ChatGPT-4. There was no issues associated with this approach. However, testing and debugging had to be done in an iterative manner with the Fortran compiler.

The next step of converting to complete modern Fortran was easier, since the GitHub Copilot is directly integrated into Visual Studio. This was used for all conversion tasks and to add comments in the code. It was more efficient than the iterative approach for minimal modern Fortran, but we found that small steps should be taken and be continually debugged and tested. One risk is that code is moved into subroutines which should stay in the main program; this happened a few times during the workflow.

A future goal if one were to continue with this code would be to recreate the results from [6]. Ideally, one would provide the papers and our current Fortran code to ChatGPT or a similar AI chatbot and ask it to update the code in one go. However, this does not work; one has to take smaller steps and have sufficient understanding of the steps to evaluate them. A more realistic approach is stepwise, where the main steps would be:

1. Convert the numerical schemes for w and Ω from upwind to central difference
2. Include the correction terms

5.3 On the general validity of a Dean number based analysis

For simulations of both laminar [24] and transitional (turbulent) [25] flow, it has become clear that analysis purely based on the Dean number is not valid in general.

For laminar flow, a Dean number based simulation analysis is valid for pipe curvatures $a/L < 10^{-6}$ and Dean numbers $D < 10$ [24]. However, for measurements, the detectable limit is in the vicinity of $a/L < 0.007$ [26].

For the case we treat with slightly curved pipes, the transition to turbulence is sub-critical, which is the same as for straight pipes [25]. From experimental investigations with higher curvature [27], it is interesting to note the persistence of structures in turbulent flow similar to those identified for laminar flow. Perhaps the laminar flow structures form a skeleton/seed which turbulence builds on?

6 Conclusions

In this paper, we have presented a case study on how to revive older code (FORTRAN 66) to modern Fortran. This AI-assisted code revival has enabled fast calculations of laminar flow through slightly curved pipes. The basis was [5], with the original code being provided in [4], see also Appendix A. The main steps involved were:

1. Digitisation of the original document with OCR tools and human corrections
2. Conversion of the FORTRAN 66 code to modern Fortran
3. Debugging and testing of the modern Fortran code

After the code was operational, we tested increasing the grid resolution and found that the position of the maximum streamwise velocity as a function of D was sensitive to this change. The positions determined using the increased grid resolution are in satisfactory agreement with those found in [6]. However, ϕ_{\max} , w_{\max} and the flux found with the revived code deviate substantially from established results [3, 6], even for the increased grid resolution. This discrepancy is caused by the lower accuracy of the finite difference scheme used in the revived code.

We hope that our work serves as a template and inspiration for other scientists who have a need for bringing older code back to life again. Also, the complete modern Fortran code in Appendix C is now available to the scientific community as a fast method to obtain a physical picture of streamwise and secondary flow in curved pipes. Finally, the results may serve educational purposes as a toy model of a canonical laminar flow.

The term computational fluid dynamics (CFD) for numerical flow solutions was introduced in 1972 [28], and the research covered in this paper can indeed be termed CFD, although this nomenclature was not used. An advantage of these early CFD simulations is that they are numerically efficient and thus run extremely fast on modern hardware/software architectures. Therefore, they remain useful as classical solutions which form a foundation for future research. In that spirit, we consider our activities as "numerical archaeology", although this phrase is already in use - albeit with another connotation [29].

We plan to use the code in the future to facilitate interdisciplinary work on fluids and plasmas which was initiated in [30].

Supplementary Information

The online version contains supplementary material available at <https://doi.org/10.1007/s44245-026-00188-w>.

Supplementary file 1.

Acknowledgements

We thank Eurythmics for inspiring the paper title with their song "Revival".

Author contributions

The author confirms that they were solely responsible for the conceptualization, methodology, data analysis, manuscript writing, and final approval of the submitted work.

Funding

Open access funding provided by RISE Research Institutes of Sweden. This research received no external funding.

Data availability

The data presented in this paper can be generated by using the modernised Fortran code in appendices B and C. The original and updated data sets are available at <https://doi.org/10.13140/RG.2.2.35103.01445> and <https://doi.org/10.13140/RG.2.2.26714.40642>, respectively. The GNU Octave programs are available at <https://doi.org/10.13140/RG.2.2.30069.84961>.

Declarations**Ethics approval and consent to participate**

Not applicable.

Consent to publications

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Received: 5 September 2025 / Accepted: 14 January 2026

Published online: 17 February 2026

References

1. Dean WR. Note on the motion of fluid in a curved pipe. *Phil Mag.* 1927;4:208–23.
2. Dean WR. The stream-line motion of fluid in a curved pipe. *Phil Mag.* 1928;5:673–95.
3. McConlogue DJ, Srivastava RS. Motion of fluid in a curved tube. *Proc Roy Soc A.* 1968;307:37–53.
4. Greenspan D, Schubert AB. Secondary flow in a curved tube. University of Wisconsin, Computer Sciences Department, Technical Report. 1972;155. <https://minds.wisconsin.edu/handle/1793/57756>
5. Greenspan D. Secondary flow in a curved tube. *J Fluid Mech.* 1973;57:167–76.
6. Collins WM, Dennis SCR. The steady motion of a viscous fluid in a curved tube. *Quart J Mech Appl Math.* 1975;28:133–56.
7. Berger SA, Talbot L, Yao L-S. Flow in curved pipes. *Ann Rev Fluid Mech.* 1983;15:461–512.
8. Taylor GI. The criterion for turbulence in curved pipes. *Proc Roy Soc A.* 1929;124:243–9.
9. Krom JG. The evolution of control and data acquisition at JET. *Fus Eng Des.* 1999;43:265–73.
10. EasyOCR. <https://pypi.org/project/easyocr/>
11. PyTesseract. <https://pypi.org/project/pytesseract/>
12. PDF.ai. <https://pdf.ai/tools/ocr-pdf>
13. MaxAI.me. <https://www.maxai.me/pdf-tools/pdf-ocr/>
14. Amazon Web Services. <https://aws.amazon.com/>
15. Metcalf M, Reid J. *Fortran 90/95 Explained.* Oxford: Oxford University Press; 1996.
16. OpenAI.com. <https://chatgpt.com/>
17. ChatGPT-4 prompt example. <https://chatgpt.com/share/671ccb3e-7ce0-8006-93ae-3c08828444b4>
18. Microsoft Windows 11 Enterprise. <https://www.microsoft.com/en-us/microsoft-365/windows/windows-11-enterprise>
19. Microsoft Visual Studio 2022. <https://visualstudio.microsoft.com/downloads/>
20. Intel oneAPI HPC Toolkit 2024.2. <https://www.intel.com/content/www/us/en/developer/tools/oneapi/hpc-toolkit.html#gs.i3q2al>
21. GNU Octave, version 9.2.0. <https://octave.org/>
22. Versteeg HK, Malalasekera W. *An introduction to computational fluid dynamics: the finite volume method.* 2nd ed. Harlow: Pearson; 2007.
23. Greenshields CJ, Weller HG. *Notes on computational fluid dynamics: general principles.* UK: CFD Direct Limited: Reading; 2022.
24. Canton J, Örlü R, Schlatter P. Characterisation of the steady, laminar incompressible flow in toroidal pipes covering the entire curvature range. *Int J Heat Fluid Flow.* 2017;66:95–107.
25. Canton J, Rinaldi E, Örlü R, Schlatter P. Critical point for bifurcation cascades and featureless turbulence. *Phys Rev Lett.* 2020;124:014501.
26. Snyder B, Hammersley JR, Olson DE. The axial skew of flow in curved pipes. *J Fluid Mech.* 1985;161:281–94.
27. Kühnen J, Holzner M, Hof B, Kuhlmann HC. Experimental investigation of transitional flow in a toroidal pipe. *J Fluid Mech.* 2014;738:463–91.
28. Roache PJ. *Computational fluid dynamics.* New Mexico, USA: Hermosa Publications: Albuquerque; 1972.
29. Jones AR. Numerical archaeology: Gleanings from the 1253 building accounts of Westminster Abbey revisited. In: Lunnon H, Rodwell W, Tatton-Brown T, editors. *Westminster Part I: the art, architecture and archaeology of the royal abbey.* Abingdon: Routledge; 2017.
30. Basse NT. The chimera revisited: wall- and magnetically-bounded turbulent flows. *Fluids.* 2024;9:34.