Article

Code Revival: Fluid Motion in a Curved Pipe

Nils Tångefjord Basse *

Posted Date: 13 February 2025

doi: 10.20944/preprints202411.1211.v4

Keywords: fluid motion in a curved pipe; laminar flow; code revival; FORTRAN 66; modern Fortran

 doi:10.20944/preprints202411.1211.v4

Article

Code Revival: Fluid Motion in a Curved Pipe

Nils Tångefjord Basse

RISE Research Institutes of Sweden, Brinellgatan 4, 504 62 Borås, Sweden; nils.basse@ri.se

Abstract: This paper presents a revival of FORTRAN 66 code which calculates flow through curved pipes. Results from the code were originally presented in [Greenspan, D. Secondary flow in a curved tube. *J. Fluid Mech.* **1973**, *57*, 167-176]. The coupled non-linear system of partial differential equations was solved numerically using a finite difference method. We demonstrate a step-by-step code revival process and compare original (coarse) results to updated (fine) solutions. Both the structure of streamwise (primary) and secondary flows are covered. The purpose of our paper is 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; FORTRAN 66; modern Fortran

1. Introduction

The theoretical treatment of laminar flow in curved pipes was pioneered by Dean in the late 1920s [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:

$$D = 4Re\sqrt{\frac{2a}{L}}, \tag{1}$$

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]. The purpose of our paper is to convert - and thus revive - this code to modern Fortran. The reason for this is twofold: (i) we need the revived code for planned future research and (ii) we want to make this code available to the scientific community for others who may need a fast and efficient code to calculate basic solutions of laminar flow in slightly curved pipes. This flow system can be considered to belong to the class of canonical laminar flows, which remains relevant for both applied and fundamental research. Note that we also cover a later paper, where improved numerical solutions over than same range of D were published [6].

The Dean number of 5000 is from [7], 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 [8] during the period 1997-1998. Thus, the task at hand, to convert FORTRAN 66 code, was feasible.

The paper is structured as follows: In Section 2, we briefly summarise the equations of motion. This is followed by the FORTRAN 66 code revival steps which are detailed in Section 3; results from the revived code are presented in Section 4. A brief discussion is presented in Section 5 and finally we

 doi:10.20944/preprints202411.1211.v4

conclude in Section 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]:

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

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

where:

$$\nabla_1^2 = \frac{\partial^2}{\partial r^2} + \frac{1}{r}\frac{\partial}{\partial r} + \frac{1}{r^2}\frac{\partial^2}{\partial \alpha^2} \tag{4}$$

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".

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

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 in 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.

Our initial attempts at extracting characters from the scanned document were done using Python with EasyOCR [9] and PyTesseract [10]. However, these tools performed poorly on the document.

Our next attempt was with two online converters [11,12]. The results were better than for the Python tools, but still with a correct percentage of around 50%.

The best conversion, with a success rate of roughly 80%, was the Amazon Textract tool from Amazon Web Services (AWS) [13].

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. Conversion

Once the code was available in digital form, the main conversion from FORTRAN 66 to modern Fortran was to replace the Hollerith string and data notation [14]. To achieve this goal, we also used

 doi:10.20944/preprints202411.1211.v4

ChatGPT-4 [15] to convert the Hollerith commands to modern Fortran. An example of the process can be found in [16].

3.3. Debugging and Testing

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

A laptop with Microsoft Windows 11 [17] installed was chosen as the hardware/software platform. Microsoft Visual Studio [18] was installed as the graphical user interface along with the Intel Fortran Compiler (ifx) [19]. In terms of Fortran versions, ifx uses the latest standards from Fortran 2018 and some Fortran 2023 features. In that sense, we consider the updated code to represent modern Fortran.

What remained was to compile (build solution) 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 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.

The two appendices can be compared to learn what has been deleted from the original FORTRAN 66 code in the modern Fortran code.

4. Results

All postprocessing in this section was done using GNU Octave [20].

4.1. Evolution of Spatial Structures

We include two figures with contour plots of ϕ and w, Figures 1 and 2, to facilitate a direct comparison to 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. 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.

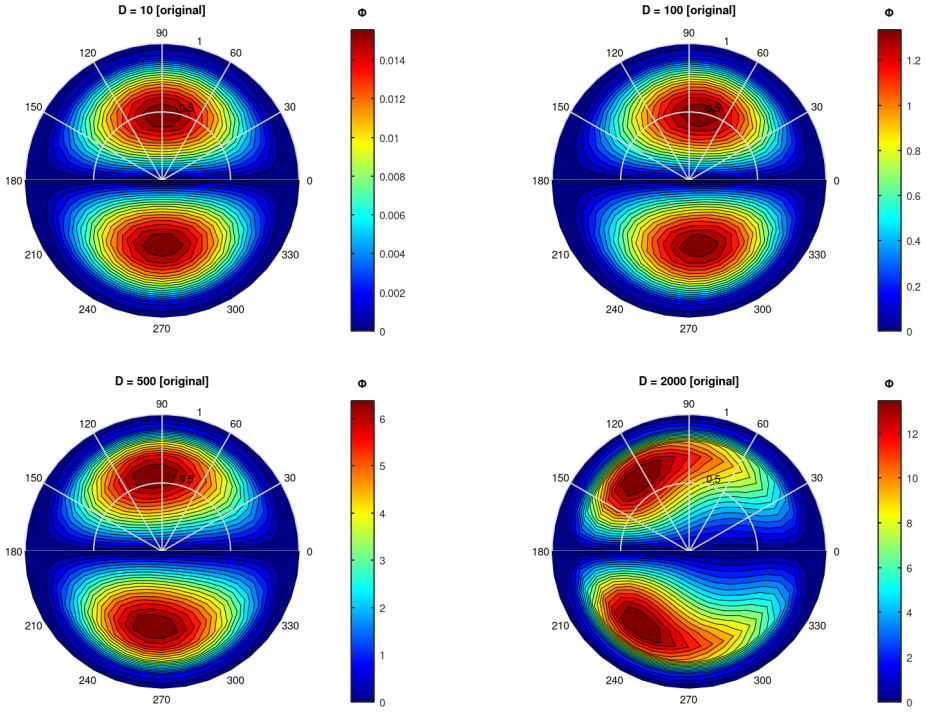

Figure 1. *Cont.*

 doi:10.20944/preprints202411.1211.v4

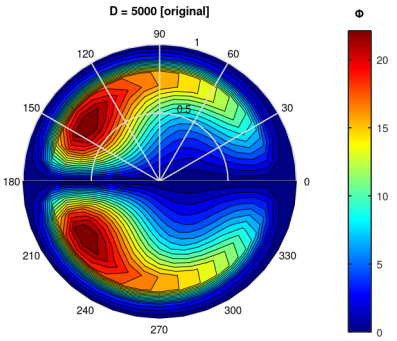

Figure 1. Original ϕ contour plots for $D = 10$, $D = 100$, $D = 500$, $D = 2000$ and $D = 5000$.

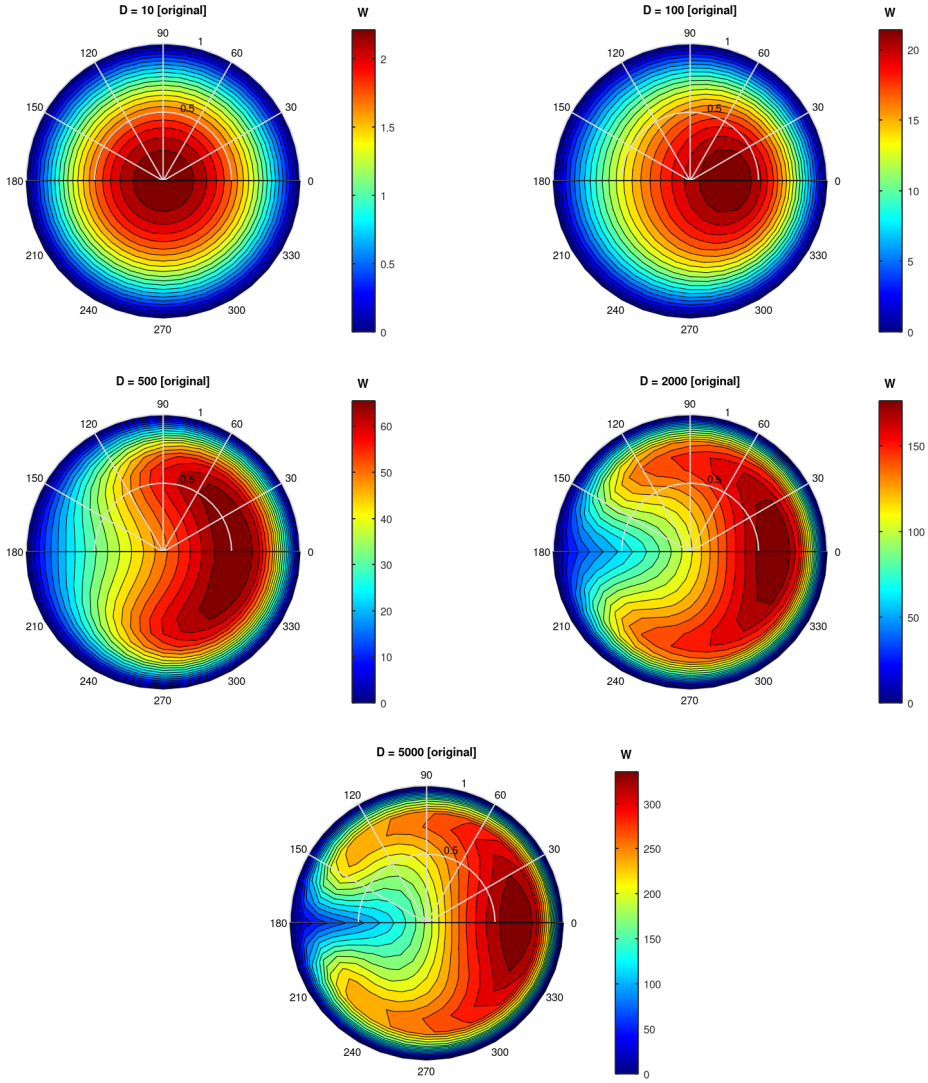

Figure 2. Original w contour plots for $D = 10$, $D = 100$, $D = 500$, $D = 2000$ and $D = 5000$.

4.2. Grid Resolution Improvement

Having a complete 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 in Appendix B, these resolutions are named `NR` and `NA`, respectively. We updated the code to run with double resolution, both for r and ϕ. To distinguish the grid resolutions, we call them "original" and "updated".

 doi:10.20944/preprints202411.1211.v4

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

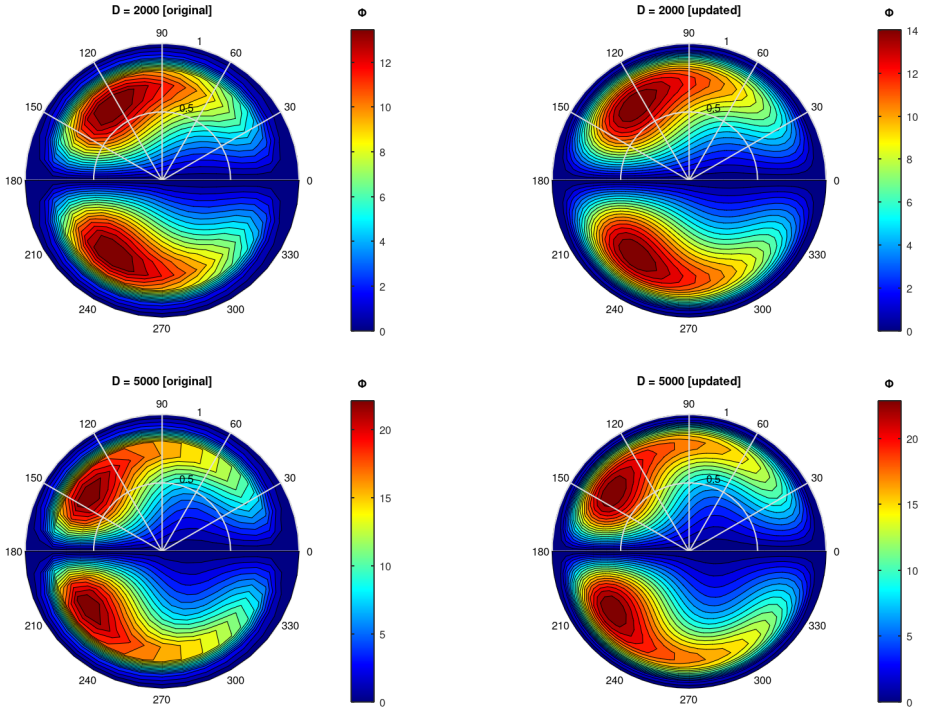

Figure 3. ϕ contour plots for $D = 2000$ and $D = 5000$. Left-hand column: Original (coarse) resolution, right-hand column: Updated (fine) resolution.

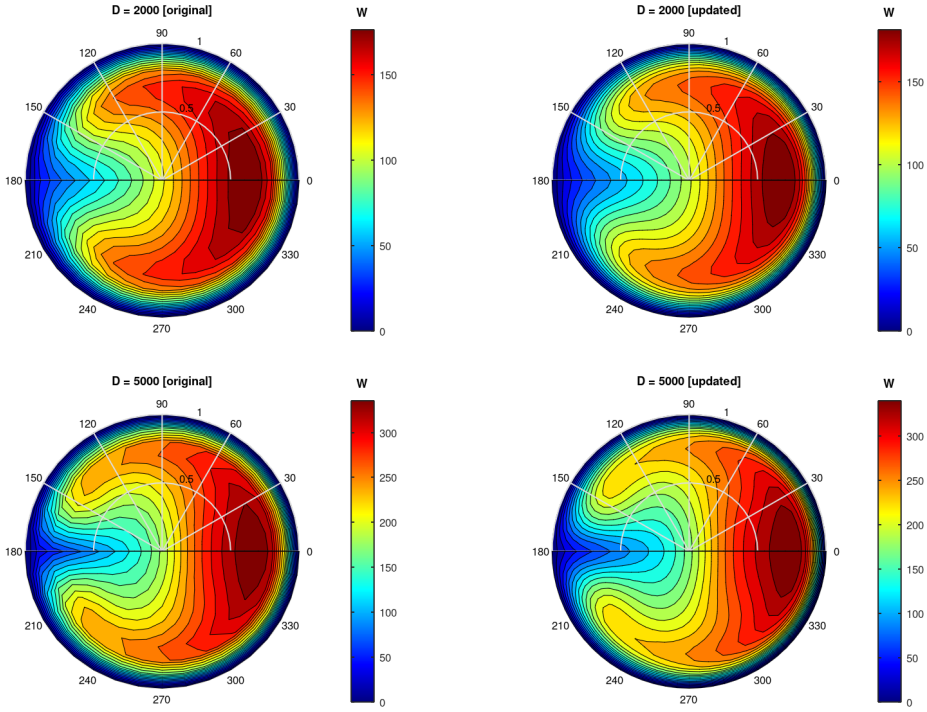

Figure 4. w contour plots for $D = 2000$ and $D = 5000$. Left-hand column: Original (coarse) resolution, right-hand column: Updated (fine) resolution.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 February 2025 doi:10.20944/preprints202411.1211.v4

4.3. A Comparison of Scaling Results

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

In Figure 5, we present the D-scaling of the peak values of ϕ and w, which we name $\phi_{\max}$ 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].

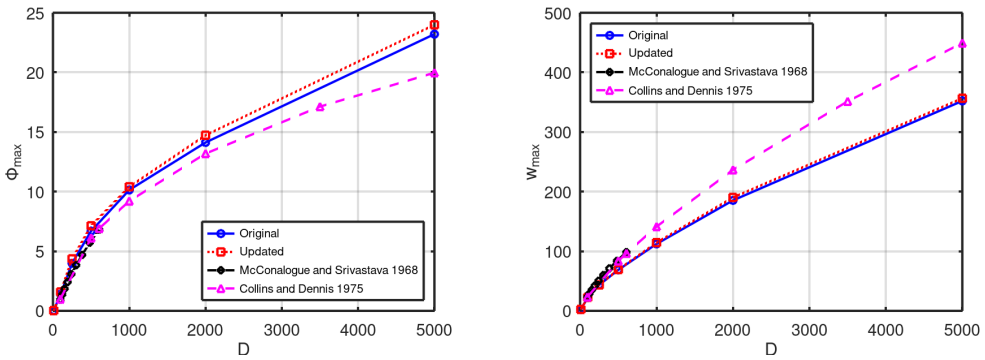

Figure 5. Left-hand plot: Value of $\phi_{\max}$ as a function of D, right-hand plot: Value of $w_{\max}$ as a function of D. Open blue dots are original code results, open red squares are updated code results, filled black dots are from [3] and open magenta triangles are from [6].

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 Fortran code and shown in Figure 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 Figure 5.

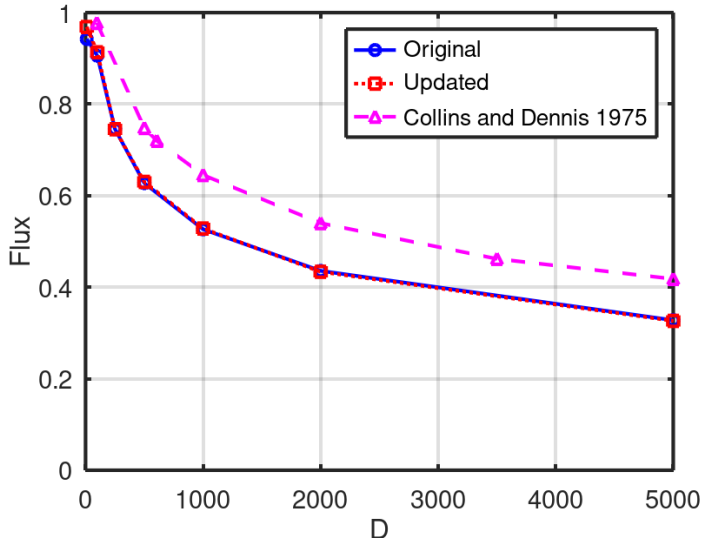

Figure 6. Streamwise flux as a function of D. Open blue dots are original code results, open red squares are updated code results and open magenta triangles are from [6].

A topic of interest for our future research is how $w_{\max}$ moves as a function of D. This is shown in Figure 7 with values estimated from [6]. Here, we find a large difference between the original and updated resolution; the updated values are quite close to the results reported in [6].

 doi:10.20944/preprints202411.1211.v4

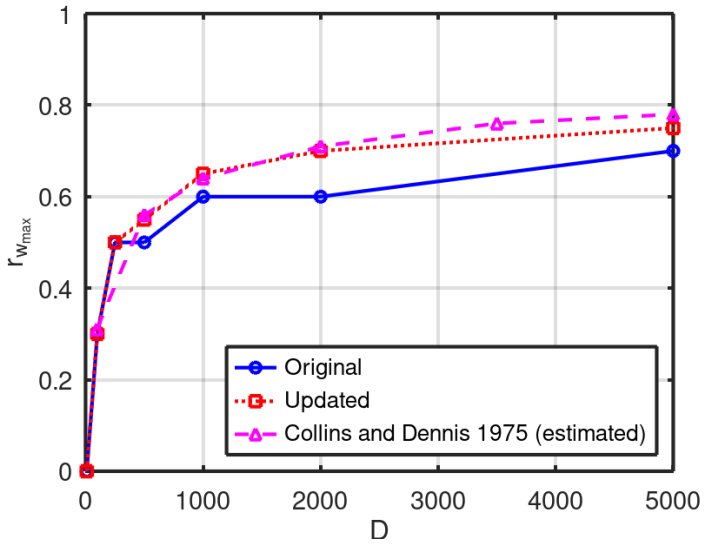

Figure 7. Normalised radius for maximum streamwise velocity $w_{\max}$ as a function of D. Open blue dots are original code results, open red squares are updated code results and open magenta triangles are estimated from Figure 4 in [6].

5. Discussion

For simulations of both laminar [21] and transitional (turbulent) [22] 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 $D < 10$ [21]. However, for measurements, the detectable limit is in the vicinity of $a/L < 0.007$ [23].

For the case we treat with slightly curved pipes, the transition to turbulence is subcritical, which is the same as for straight pipes [22]. From experimental investigations with higher curvature [24], it is interesting to note the persistence of structures in turbulent flow similar to those identified in 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 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, $\phi_{\max}$, $w_{\max}$ and the flux found with the revived code deviate substantially with established results [3,6], even for the increased grid resolution.

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 modern Fortran code in Appendix B 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.

 doi:10.20944/preprints202411.1211.v4

The term computational fluid dynamics (CFD) for numerical flow solutions was introduced in 1972 [25], 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 [26].

We plan to use the code in the future to facilitate interdisciplinary work on fluids and plasmas which was initiated in [27].

Funding: This research received no external funding.

Data Availability Statement: The data presented in this paper can be generated by using the modernised Fortran code in Appendix B.

Acknowledgments: We thank Eurythmics for inspiring the paper title.

Conflicts of Interest: The author declares no conflict of interest.

Appendix A. Original FORTRAN 66 Code

For reference, this appendix includes the original FORTRAN 66 code written by A.B.Schubert in 1972 [4].

This code cannot be compiled and run directly, one reason being that physical punch cards are required.

 doi:10.20944/preprints202411.1211.v4

C:\Fortran codes\Schubert_1972_FORTRAN_66.f 1

```
C     Appendix: FORTRAN Program for Secondary Flow, by A. B. Schubert (1972)
C
C     TUBEFLOW    A PROGRAM WHICH COMPUTES THE SECONDARY
C     (CROSS-SECTIONAL) FLOW OF A FLUID IN A CURVED TUBE AND THE
C     VELOCITY AND FLUX IN THE AXIAL DIRECTION OF THE TUBE BY
C     DISCRETIZATION OF THE CROSS-SECTION AND OF THE GOVERNING
C     DIFFERENTIAL EQUATIONS.

      PARAMETER NR=10,NA=18,NRP1=NR+1,NAP1=NA+1,NRM1=NR-1,NAM1=NA-1,
     * NAH=NA/2+1

C         NR = NUMBER OF GRID SPACES ON (0,1) IN RADIAL (R) DIRECTION.
C         NA = NUMBER OF GRID SPACES ON (0,PI) IN ANGULAR (ALPHA)
C              DIRECTION.

C     PRINCIPAL DISCRETE FUNCTIONS COMPUTED ARE DEFINED AS FOLLOWS.
C       PHI(I,J,K)= NON-DIMENSIONAL STREAM FUNCTION VALUE FOR SECONDARY
C                   FLOW AT POLAR GRID POINT CORRESPONDING TO I-TH
C                   RADIAL VALUE (WHERE I= 1 CORRESPONDS TO R = 0 AND
C                   I = NR+1 CORRESPONDS TO R = 1) AND J-TH ANGULAR
C                   VALUE (WHERE J = 1 CORRESPONDS TO ALPHA = 0 AND
C                   J = NA+1 CORRESPONDS TO ALPHA = PI.
C                   K = 1 IMPLIES A VALUE AT THE PREVIOUS OUTER
C                         ITERATION.
C                   K = 2 IMPLIES A VALUE AT THE PREVIOUS SOR ITERATION
C                   K = 3 IMPLIES A VALUE AT THE CURRENT SOR AND OUTER
C                         ITERATION.
C        W(I,J,K) = NON-DIMENSIONAL VELOCITY IN THE AXIAL DIRECTION,
C                   WITH SUBSCRIPTS DEFINED AS FOR PHI.
C    OMEGA(I,J,K) = INTERMEDIATE VARIABLE INTRODUCED IN ANALYTICAL
C                   DESCRIPTION OF PROBLEM, WITH SUBSCRIPTS DEFINED
C                   AS FOR PHI.

      DIMENSION PHI(NRP1,NAP1,4),W(NRP1,NAP1,4),OMEGA(NRP1,NAP1,4),
     * XI(4),RHO(3),EPS(3),SA(NAP1),COSA(NAP1),RINV(NRP1),RINV2(NRP1),
     * RHOC(3),B(NRP1,5),C(NRP1,NAP1),EPPS(3),XIC(4),E(NRP1,NAP1,6),
     * EE(NRP1),EF(NRP1),CA(NRP1,NAP1),DELA(NR),CO(NR),DOR(5)

      DATA PI/3.14159255/,MAXSOR,MAXOUT/250,60/,NOR/3/,(DOR(I),I=1,3)
     */.2,.2,.2/,ISTART/1/

C     MAXSOR = MAXIMUM NUMBER OF ITERATIONS TO BE ALLOWED FOR COMPUTING
C              SOLUTION TO ANY SYSTEM BY SUCCESSIVE OVER-RELAXATION
C     MAXOUT = MAXIMUM NUMBER OF OUTER ITERATIONS TO BE ALLOWED IN THE
C              COMPUTATION OF PHI, W, AND OMEGA.
C        NOR = NUMBER OF DIFFERENT OVER-RELAXATION FACTORS TO BE TRIED
C              IN ANY DATA CASE.
C     DOR(I) = AMOUNT OF DECREASE IN OVER-RELAXATION FACTOR AT I-TH
C              CHANGE.
C     ISTART = INPUT CONTROL VARIABLE INDICATING WHETHER OR NOT THE
C              FIRST OUTER ITERATES ARE TO BE READ FROM CARDS.
C              ISTART.EQ.O IMPLIES STARTING VALUES WILL BE SET = 0 BY
C                          PROGRAM.
```

 doi:10.20944/preprints202411.1211.v4

C:\Fortran codes\Schubert_1972_FORTRAN_66.f 2

```
C              ISTART.NE.0 IMPLIES STARTING ITERATES WILL BE READ FROM
C                          CARDS IN THE FORMAT (8E10.5).

C     TEST FOR NECESSITY OF READING STARTING OUTER ITERATES FOR PHI, W,
C     AND OMEGA.
C     DSTART = VALUE OF PARAMETER D FOR WHICH THE STARTING VALUES
C     ARE A SOLUTION.

      IF(ISTART.NE.0) READ(5,88) DSTART,((PHI(I,J,4),J=1,NAP1),I=1,NRP1)
     *,((W(I,J,4),J=1,NAP1),I=1,NRP1),((OMEGA(I,J,4),J=1,NAP1),I=1,NRP1)
      KRP1=NRP1

C     DR = GRID SPACING IN RADIAL DIRECTION.
C     DA = GRID SPACING IN ANGULAR DIRECTION.

      DR=1./NR
      DA=PI/NA
      DAH=.5*DA
      DRH=.5*DR
      DRDAM=DR*DA
      DRDA=DR/DA
      DADR=DA/DR

C     COMPUTE ELEMENTS OF AREA, DELA(I),I=1,...,NR, IN RADIAL DIRECTION
C     FOR USE IN FLUX CALCULATION.

      DO 1 I=1,NR
    1 DELA(I)=(2*I-1)*DRH*DRDAM

C     ON THE GRID DEFINED BY DR AND DA, COMPUTE DISCRETE FUNCTIONS
C     DEPENDENT ONLY ON RADIUS AND ANGLE.

      SA(1)=0.
      DO 2 J=2,NA
      SA(J)=SIN((J-1)*DA)*DAH
    2 COSA(J)=COS((J-1)*DA)
      SA(NAP1)=0.
      COSA(NAP1)=-1.
      RINV(NRP1)=1.
      RINV2(NRP1)=1.
      DO 3 I=2,NR
      RINV(I)=1./((I-1)*DR)
      DRRH=DRH*RINV(I)
      RINV2(I)=RINV(I)**2
      DO 3 J=2,NA
    3 CA(I,J)=DRRH*COSA(J)

C     COMPUTE SOR COEFFICIENTS FOR PHI.

      DO 4 I=2,NR
      BO=2.*(DADR+DRDA*RINV2(I))+DA*RINV(I)
      B(I,1)=(DADR+DA*RINV(I))/BO
      B(I,2)=DRDA*RINV2(I)/BO
```

 doi:10.20944/preprints202411.1211.v4


```
      B(I,3)=DADR/BO
      B(I,4)=B(I,2)
    4 B(I,5)=DRDAM/BO

C     READ OUTER ITERATION SMOOTHING PARAMETERS (XI), OVER-RELAXATION
C     FACTORS (RHO), OUTER ITERATION CONVERGENCE TOLERANCES (EPS),
C     PARAMETER RELATED TO REYNOLDS NO. (D), AND INPUT/OUTPUT CONTROL
C     VARIABLES DEFINED AS FOLLOWS.
C     ISAVE.NE.0 IMPLIES SAVE THE SOLUTION FOR THIS CASE (IF OBTAINED)
C                IN MEMORY FOR USE AS STARTING VALUES FOR A SUCCEEDING
C                CASE WITH A DIFFERENT VALUE OF D.
C     ISAVE.EQ.0 IMPLIES DO NOT DO THE ABOVE.
C      IUSE.NE.0 IMPLIES TAKE STARTING VALUES FOR OUTER ITERATION FROM
C                MEMORY.
C      IUSE.EQ.0 IMPLIES INITIALIZE OUTER ITERATES TO ZERO.
C      IPCH.NE.0 IMPLIES PUNCH SOLUTION FOR THIS CASE (IF OBTAINED)
C                OUT ON CARDS IN THE FORMAT (8E10.5).
C      IPCH.EQ.0 IMPLIES DO NOT PUNCH SOLUTION OBTAINED FOR THIS CASE.

    5 READ(5,99,END=70) XI,RHO,EPS,D,ISAVE,IUSE,IPCH

C     PRINT OUT INPUT PARAMETERS.

      WRITE(6,98) D,RHO,XI,EPS

C     COMPUTE PARAMETERS DEPENDENT ON THE INPUT PARAMETER D, INCLUDING
C     THE COEFFICIENTS, CO(I),I=1,...,NR, IN THE NUMERICAL INTEGRATION
C     FOR THE FLUX.

      DDRDAM=D*DRDAM
      DDR2=D*DR**2
      CON=4./(PI*D)
      CO(1)=16.*DELA(1)/(3.*PI*D)
      DO 105 I=2,NR
  105 CO(I)=CON*DELA(I)

C     COMPUTE COMPLEMENTS (RELATIVE TO 1) OF SMOOTHING PARAMETERS AND
C     OVER-RELAXATION FACTORS.
C     ALSO COMPUTE SOR CONVERGENCE TOLERANCES, EPPS(I),I=1,2,3, AS
C     FUNCTIONS OF OUTER ITERATION TOLERANCES, EPS(I),I=1,2,3.

      DO 6 I=1,3
      XIC(I)=1.-XI(I)
      RHOC(I)=1.-RHO(I)
    6 EPPS(I)=.05*EPS(I)
      XIC(4)=1.-XI(4)

C     TEST WHETHER INITIAL OUTER ITERATES ARE TO BE OBTAINED FROM
C     MEMORY, HAVING BEEN PLACED THERE AS THE SOLUTION FOR A PREVIOUS
C     VALUE OF D EITHER BY CARD INPUT OR BY A PREVIOUS COMFUTATION
C     THIS RUN.

      IF(IUSE.EQ.0) GO TO 7
```

 doi:10.20944/preprints202411.1211.v4


```
      DO 106 I=1,NRP1
      DO 106 J=1,NAP1
      PHI(I,J,3)=PHI(I,J,4)
      W(I,J,3)=W(I,J,4)
  106 OMEGA(I,J,3)=OMEGA(I,J,4)
      WRITE(6,87) DSTART
      GO TO 9

C     INITIALIZE OUTER ITERATES TO ZERO IF NEITHER INPUT NOR COMPUTED
C     PREVIOUSLY THIS RUN.

    7 DO 8 I=1,NRP1
      DO 8 J=1,NAP1
      PHI(I,J,3)=0.
      W(I,J,K)=0.
    8 OMEGA(I,J,3)=0.

C     INITIALIZE OUTER ITERATION COUNTER (IOUT) AND COUNTERS OF NUMBER
C     OF OVER-RELAXATION FACTORS USED FOR W AND OMEGA (IRW,IRO, RESP.)
C     FOP THIS DATA CASE.

    9 IOUT=0
      IRW=0
      IRO=0

C     TEST FOR MAXIMUM NUMBER OF OUTER ITERATIONS.

   10 IF(IOUT.GE.MAXOUT) GO TO 58

C     UPDATE OUTER ITERATION COUNTER, WRITE MESSAGE INDICATING NEW
C     OUTER ITERATION NUMBER, AND SET OUTER ITERATION CONVERGENCE
C     INDICATOR (ICV) TO ZERO. AFTER COMPUTATION OF ALL CURRENT OUTER
C     ITERATES AND COMPARISON WITH PREVIOUS OUTER ITERATES AGAINST THE
C     SPECIFIED TOLERANCES, ICV WILL STILL BE ZERO IF CONVERGENCE HAS
C     BEEN OBTAINED, OTHERWISE ICV WILL BE NON-ZERO.

      IOUT=IOUT+1
      WRITE(6,92) IOUT
      ICV=0

C     UPDATE PREVIOUS OUTER ITERATES.

      DO 12 I=1,NRP1
      DO 12 J=1,NAP1
      PHI(I,J,1)=PHI(I,J,3)
      W(I,J,1)=W(I,J,3)
   12 OMEGA(I,J,1)=OMEGA(I,J,3)

C     COMPUTE CONSTANT SOR COEFFICIENT FOR PHI.

      DO 13 I=2,NR
      DO 13 J=2,NA
   13 C(I,J)=B(I,5)*OMEGA(I,J,1)
```

 doi:10.20944/preprints202411.1211.v4


```
C     INITIALIZE SOR ITERATION COUNTER FOR PHI.

      ISOR=0

C     TEST FOR MAXTMUM NUMBER OF SOR ITERATIONS FOR PHI.

   14 IF(ISOR.GE.MAXSOR) GO TO 55

C     UPDATE SOR ITERATION COUNTER IF MAXIMUM NUMBER HAS NOT BEEN
C     ACHIEVED.

      ISOR=ISOR+1

C     UPDATE PREVIOUS SOR ITERATES FOR PHI.

      DO 16 I=2,NR
      DO 16 J=2,NA
   16 PHI(I,J,2)=PHI(I,J,3)

C     SET SOR ITERATION CONVERGENCE INDICATOR (ICONV) TO ZERO. AFTER
C     COMPUTATION OF CURRENT SOR ITERATE AND COMPARISON WITH PREVIOUS
C     ITERATE AGAINST THE TOLERANCE, ICONV WILL STILL BE ZERO IF
C     CONVERGENCE HAS BEEN OBTAINED. OTHERWISE IT WILL BE NON-ZERO.

      ICONV=0

C     COMPUTE CURRENT SOR ITERATE FOR PHI...
C     ...FIRST ON INTERIOR OF REGION, EXCLUDING 'INNER BOUNDARY'
C        R = 1.-DR.

      DO 18 I=2,NRM1
      DO 18 J=2,NA
      PHI(I,J,2)=RHOC(1)*PHI(I,J,2)+RHO(1)*(B(I,1)*PHI(I+1,J,2)
     * +B(I,2)*PHI(I,J+1,2)+B(I,3)*PHI(I-1,J,3)+B(I,4)*PHI(I,J-1,3)
     * +C(I,J))
      IF(ABS(PHI(I,J,2)-PHI(I,J,3)).GT.EPPS(1)) ICONV=1
   18 CONTINUE

C     ...THEN ON 'INNER BOUNDARY' R = 1.-DR.

      DO 19 J=2,NA
      PHI(NR,J,3)=RHOC(1)*PHI(NR,J,2)+RHO(1)*.25*PHI(NRM1,J,3)
      IF(ABS(PHI(NR,J,2)-PHI(NR,J,3)).GT.EPPS(1)) ICONV=1
   19 CONTINUE

C     TEST FOR CONVERGENCE OF SOR ITERATION FOR PHI.

      IF(ICONV.NE.0) GO TO 14

C     SOR CONVERGENCE ATTAINED FOR PHI. SMOOTH OUTER ITERATE.

      DO 20 I=2,NR
```

 doi:10.20944/preprints202411.1211.v4


```
      DO 20 J=2,NA
      PHI(I,J,3)=XI(1)*PHI(I,J,1)+XIC(1)*PHI(I,J,3)
      IF(ABS(PHI(I,J,1)-PHI(I,J,3)).GT.EPS(1)) ICV=1
   20 CONTINUE

C     PRINT OUT SMOOTHED CURRENT OUTER ITERATE FOR PHI.

      CALL OUTPUT('PHI',ISOR,PHI(1,1,3))

C     COMPUTE COEFFICIENTS FOR SOR ITERATION FOR W...
C     ...FIRST COMPUTE COEFFICIENTS AT THE ORIGIN

      E0=4.+ABS(PHI(2,NAH,3))
      E1=(1.-AMIN1(PHI(2,NAH,3),0.))/E0
      E2=2./E0
      E3=(1.+AMAX1(PHI(2,NAH,3),0.))/E0
      E4=DDR2/E0

C     ...NEXT COMPUTE COEFFICIENTS FOR REMAINDER OF REGION...
C           FIRST FOR RAYS ALONG ALPHA = 0 AND ALPHA = PI, EXCLUDING
C           R = 0 AND R = 1.

      DO 23 I=2,NR
      DELTA1=DA-PHI(I,2,3)
      DELTA2=DA+PHI(I,NA,3)
      EE(I)=2.*(DADR+DRDA*RINV2(I))
      EE1=EE(I)+RINV(I)*ABS(DELTA1)
      EE2=EE(I)+RINV(I)*ABS(DELTA2)
      E(I,1,1)=(DADR+RINV(I)*AMAX1(DELTA1,0.))/EE1
      E(I,NAP1,1)=(DADR+RINV(I)*AMAX1(DELTA2,0.))/EE2
      EF(I)=DRDA*RINV2(I)
      E(I,1,2)=EF(I)/EE1
      E(I,NAP1,2)=EF(I)/EE2
      E(I,1,3)=(DADR-RINV(I)*AMIN1(DELTA1,0.))/EE1
      E(I,NAP1,3)=(DADR-RINV(I)*AMIN1(DELTA2,0.))/EE2
      E(I,1,4)=E(I,1,2)
      E(I,NAP1,4)=E(I,NAP1,2)
      E(I,1,5)=DDRDAM/EE1
   23 E(I,NAP1,5)=DDRDAM/EE2

C           THEN ON INTERIOR OF REGION.

      DO 25 I=2,NR
      DO 25 J=2,NA
      GAMMA=.5*(PHI(I+1,J,3)-PHI(I-1,J,3))
      DELTA=DA-.5*(PHI(I,J+1,3)-PHI(I,J-1,3))
      E(I,J,6)=EE(I)+RINV(I)*(ABS(GAMMA)+ABS(DELTA))
      E(I,J,1)=(DADR+RINV(I)*AMAX1(DELTA,0.))/E(I,J,6)
      E(I,J,2)=(EF(I)+RINV(I)*AMAX1(GAMMA,0.))/E(I,J,6)
      E(I,J,3)=(DADR-RINV(I)*AMIN1(DELTA,0.))/E(I,J,6)
      E(I,J,4)=(EF(I)-RINV(I)*AMIN1(GAMMA,0.))/E(I,J,6)
   25 E(I,J,5)=DDRDAM/E(I,J,6)
```

 doi:10.20944/preprints202411.1211.v4


```
C     INITIALIZE SOR ITERATION COUNTER FOR W.

   26 ISOR=0

C     TEST FOR MAXIMUM NUMBER OF SOR ITERATIONS.

   27 IF(ISOR.GE.MAXSOR) GO TO 56

C     UPDATE SOR ITERATION COUNTER IF MAXIMUM HAS NOT BEEN ACHIEVED.

      ISOR=ISOR+1

C     UPDATE PREVIOUS SOR ITERATES FOR W...
C     ...FIRST AT THE ORIGIN

      W(1,1,2)=W(1,1,3)

C     ...THEN ON REMAINDER OF REGION EXCLUDING R = 1.

      DO 29 I=2,NR
      DO 29 J=1,NAP1
   29 W(I,J,2)=W(I,J,3)

C     SET SOR CONVERGENCE INDICATOR (ICONV) TO ZERO.

      ICONV=0

C     COMPUTE CURRENT SOR ITERATE FOR W...
C     ...FIRST AT THE ORIGIN

      W(1,1,3)=RHOC(2)*W(1,1,2)     +RHO(2)*(E1*W(2,1,2)+E2*W(2,NAH,2)
     * +E3*W(2,NAP1,2)+E4)

C     (SET VALUES OF W FOR R = 0 AND ALL ANGLES ALPHA(I),I=1,...,NA+1,
C     EQUAL TO VALUE OF W AT ORIGIN FOR CONVENIENCE IN SOR COMPUTATION.)

      DO 30 J=2,NAP1
   30 W(1,J,3)=W(1,1,3)
      IF(ABS(W(1,1,2)-W(1,1,3)).GT.EPPS(2)) ICONV=1

C     ...NEXT ALONG RAY ALPHA = 0. EXCLUDING R = 0 AND R = 1.

      DO 32 I=2,NR
      W(I,1,3)=RHOC(2)*W(I,1,2)+RHO(2)*(E(I,1,1)*W(I+1,1,2)+E(I,1,2)*2.*
     * W(I,2,2)+E(I,1,3)*W(I-1,1,3)+E(I,1,5))
      IF(ABS(W(I,1,3)-W(I,1,2)).GT.EPPS(2)) ICONV=1
   32 CONTINUE

C     ...NEXT ON INTERIOR OF REGION

      DO 33 I=2,NR
      DO 33 J=2,NA
      W(I,J,3)=RHOC(2)*W(I,J,2)+RHO(2)*(E(I,J,1)*W(I+1,J,2)+E(I,J,2)*
```

 doi:10.20944/preprints202411.1211.v4


```
     * W(I,J+1,2)+E(I,J,3)*W(I-1,J,3)+E(I,J,4)*W(I,J-1,3)+E(I,J,5))
       IF(ABS(W(I,J,2)-W(I,J,3)).GT.EPPS(2)) ICONV=1
   33 CONTINUE

C      ...FINALLY ALONG RAY ALPHA = PI. EXCLUDING R = 0 AND R = 1.

       DO 34 I=2,NR
       W(I,NAP1,3)=RHOC(2)*W(I,NAP1,2)+RHO(2)*(E(I,NAP1,1)*W(I+1,NAP1,2)+
     * E(I,NAP1,3)*W(I-1,NAP1,3)+2.*E(I,NAP1,4)*W(I,NA,3)+E(I,NAP1,5))
       IF(ABS(W(I,NAP1,2)-W(I,NAP1,3)).GT.EPPS(2)) ICONV=1
   34 CONTINUE

C      TEST FOR CONVERGENCE OF SOR ITERATION FOR W.

       IF(ICONV.NE.0) GO TO 27

C      SOR CONVERGENCE ATTAINED FOR W. SMOOTH OUTER ITERATE...
C      ...FIRST AT THE ORIGIN

       W(1,1,3)=XI(2)*W(1,1,1)+XIC(2)*W(1,1,3)

C      (SET VALUES OF W FOR R = 0 AND ALL DISCRETE ANGLES EQUAL TO
C      SMOOTHED VALUE OF W AT ORIGIN.)

       DO 134 J=2,NAP1
  134  W(1,J,3)=W(1,1,3)
       IF(ABS(W(1,1,1)-W(1,1,3)).GT.EPS(2)) ICV=1

C      ...THEN ON REMAINDER OF REGION, EXCLUDING R = 1.

       DO 35 I=2,NR
       DO 35 J=I,NAP1
       W(I,J,3)=XI(2)*W(I,J,1)+XIC(2)*W(I,J,3)
       IF(ABS(W(I,J,1)-W(I,J,3)).GT.EPS(2)) ICV=1
   35 CONTINUE

C      PRINT OUT SMOOTHED CURRENT OUTER ITERATE FOR W.

       CALL OUTPUT('W',ISOR,W(1,1,3))

C      COMPUTE AND SMOOTH OMEGA ON BOUNDARY R = 1.

       DO 37 J=2,NA
       OMEGA(NRP1,J,3)=XI(3)*OMEGA(NRP1,J,1)-XIC(3)*2.*RINV2(2)*PHI(NR,
     * J,3)
       IF(ABS(OMEGA(NRP1,J,1)-OMEGA(NRP1,J,3)).GT.EPS(3)) ICV=1
   37 CONTINUE

C      COMPUTE CONSTANT COEFFICIENT FOR SOR ITERATION FOR OMEGA.

       DO 40 I=2,NR
       DO 40 J=2,NA
   40  E(I,J,6)=-W(I,J,3)*(SA(J)*(W(I+1,J,3)-W(I-1,J,3))+CA(I,J)*(W(I,
```

 doi:10.20944/preprints202411.1211.v4


```
     * J+1,3)-W(I,J-1,3)))/E(I,J,6)

C     INITIALIZE COUNTER OF SOR ITERATION NUMBER FOR OMEGA.

   41 ISOR=0

C     TEST FOR MAXIMUM NUMBER OF SOR ITERATIONS.

   42 IF(ISOR.GE.MAXSOR) GO TO 57

C     UPDATE SOR ITERATION COUNTER IF MAXIMUM NUMBER HAS NOT BEEN
C     ACHIEVED.

      ISOR=ISOR+1

C     UPDATE PREVIOUS SOR ITERATES FOR OMEGA ON INTERIOR.

      DO 44 I=2,NR
      DO 44 J=2,NA
   44 OMEGA(I,J,2)=OMEGA(I,J,3)

C     SET SOR CONVERGENCE INDICATOR TO ZERO.

      ICONV=0

C     COMPUTE CURRENT SOR ITERATE FOR OMEGA ON INTERIOR.

      DO 46 I=2,NR
      DO 46 J=2,NA
      OMEGA(I,J,3)=RHOC(3)*OMEGA(I,J,2)+RHO(3)*(E(I,J,1)*OMEGA(I+1,J,2)
     * +E(I,J,2)*OMEGA(I,J+1,2)+E(I,J,3)*OMEGA(I-1,J,3)+E(I,J,4)
     * *OMEGA(I,J-1,3)+E(I,J,6))
      IF(ABS(OMEGA(I,J,2)-OMEGA(I,J,3)).GT.EPPS(3)) ICONV=1
   46 CONTINUE

C     TEST FOR CONVERGENCE OF SOR ITERATION FOR OMEGA.

      IF(ICONV.NE.0) GO TO 42

C     SOR CONVERGENCE ATTAINED FOR OMEGA. SMOOTH OUTER ITERATE.

      DO 48 I=2,NR
      DO 48 J=2,NA
      OMEGA(I,J,3)=XI(4)*OMEGA(I,J,1)+XIC(4)*OMEGA(I,J,3)
      IF(ABS(OMEGA(I,J,1)-OMEGA(I,J,3)).GT.EPS(3)) ICV=1
   48 CONTINUE

C     PRINT OUT SMOOTHED CURRENT OUTER ITERATE FOR OMEGA.

      CALL OUTPUT('OMEGA',ISOR,OMEGA(1,1,3))

C     TEST FOR CONVERGENCE OF ALL OUTER ITERATES.
```

 doi:10.20944/preprints202411.1211.v4


```
      IF(ICV.NE.0) GO TO 10

C     PRINT MESSAGE INDICATING CONVERGENCE OF OUTER ITERATION.

      WRITE(6,97)

C     PUNCH SOLUTION (AND VALUE OF 0) OUT ON CARDS IF INPUT CONTROL
C     PARAMETER IPCH SO DICTATES, AND PRINT MESSAGE INDICATING THAT
C     THIS PUNCHING WAS DONE.

      IF(IPCH.NE.0)PUNCH88,D,((PHI(I,J,3),J=1,NAP1),I=1,NRP1),((W(I,J,3)
     + ,J=1,NAP1),I=1,NRP1),((OMEGA(I,J,3),J=1,NAP1),I=1,NRP1)
      IF(IPCH.NE.0) WRITE(6,86)

C     COMPUTE RATIO OF FLUX IN CURVED TUBE TO THAT IN STRAIGHT TUBE.

      QR=0.
      DO 49 J=1,NA
   49 QR=QR+W(1,J,3)+W(2,J,3)+W(2,J+1,3)
      QR=QR*CO(1)
      DO 249 I=2,NR
      S=0.
      DO 149 J=1,NA
  149 S=S+W(I,J,3)+W(I+1,J,3)+W(I,J+1,3)+W(I+1,J+1,3)
  249 QR=QR+CO(I)*S

C     PRINT OUT VALUE OF FLUX RATIO JUST COMPUTED.

      WRITE(6,90) QR

C     TEST WHETHER OR NOT SOLUTION JUST OBTAINED IS TO BE SAVED IN
C     MEMORY FOR SOME SUCCEEDING CASE WITH A DIFFERENT VALUE OF D.

      IF(ISAVE.EQ.0) GO TO 5

C     SAVE CURRENT SOLUTION IN ANOTHER AREA OF MEMORY.

      DO 50 I=1,NRP1
      DO 50 J=1,NAP1
      PHI(I,J,4)=PHI(I,J,3)
      W(I,J,4)=W(I,J,3)
   50 OMEGA(I,J,4)=OMEGA(I,J,3)

C     SAVE VALUE OF D FOR WHICH SOLUTION WAS JUST OBTAINED IN DSTART.

      DSTART=D
      GO TO 5

C     CONVERGENCE FAILURE MESSAGES

C     PHI ITERATION FAILED. READ NEXT DATA CASE.

   55 WRITE(6,96)
```

 doi:10.20944/preprints202411.1211.v4


```
      GO TO 5

C     W ITERATION FAILED. REDUCE OVER-RELAXATION FACTOR AND TRY AGAIN,
C     UNLESS THIS HAS ALREADY BEEN DONE THE MAXIMUM NUMBER OF TIMES
C     (NOR), IN WHICH CASE READ NEXT DATA CASE.

   56 WRITE(6,95) RH0(2)
      IF(IRW.GE.NOR) GO TO 5
      IRW=IRW+1
      RH0(2)=RH0(2)-DOR(IRW)
      RH0C(2)=1.-RH0(2)
      DO 156 I=1,NRP1
      DO 156 J=1,NAP1
  156 W(I,J,3)=W(I,J,1)
      GO TO 28

C     OMEGA ITERATION FAILED. REDUCE O-R FACTOR AND TRY AGAIN, UNLESS
C     THIS HAS ALREADY BEEN DONE THE MAXIMUM NUMBER OF TIMES (NOR),
C     IN WHICH CASE READ THE NEXT DATA CASE.

   57 WRITE(6,94) RH0(3)
      IF(IRO.GE.NOR) GO TO 5
      IRO=IRO+1
      RHO(3)=RHO(3)-DOR(IRO)
      RHOC(3)=1.-RH0(3)
      DO 157 I=1,NRP1
      DO 157 J=1,NAP1
  157 OMEGA(I,J,3)=OMEGA(I,J,1)
      GO TO 41

C     OUTER ITERATION FAILED. PRINT MESSAGE INDICATING THIS AND READ
C     NEXT DATA CASE.

   58 WRITE(6,93)
      GO TO 5

C     TERMINATION POINT FOR PROGRAM. CONTROL REACHES HERE AFTER ATTEMPT
C     TO READ PAST LAST DATA RECORD.

   70 STOP

   99 FORMAT(11F5.5,3I5)
   98 FORMAT(1H1 6X 'D =' F5.0//13X 'PHI' 7X 'W' 5X 'OMEGA' // 5X
     * 'RHO =' F6.2,2(3X F6.2)/6X 'XI =' F6.4,3(3X F6.4)/5X 'EPS ='
     * F6.4,2(3X F6.4))
   97 FORMAT('OOUTER ITERATION CONVERGED TO GIVEN TOLERANCES.')
   96 FORMAT('OSOR FOR   PHI FAILED.')
   95 FORMAT('OSOR FOR     W FAILED WITH SOR FACTOR =' F6.2)
   94 FORMAT('OSOR FOR OMEGA FAILED WITH SOR FACTOR =' F6.2)
   93 FORMAT('OOUTER ITERATION FAILED TO CONVERGE.')
   92 FORMAT(//'OOUTER ITERATION' I5//)
   91 FORMAT(1X 11E11.5)
   90 FORMAT('OFLUX RATIO =' F10.5)
```

 doi:10.20944/preprints202411.1211.v4


```
   89 FORMAT('OSOR FACTOR FOR W CHANGED TO ' F6.2)
   88 FORMAT(8E10.5)
   87 FORMAT('OINITIAL ITERATE TAKEN FROM SOLUTION FOR D =' F7.0)
   86 FORMAT('OSOLUTION WAS OUTPUT ON PUNCHED CARDS.')

      END

C     THIS ROUTINE PRINTS OUT A DISCRETE FUNCTION IN A RECTANGULAR
C     FORMAT WHICH IS RELATED TN THE FOLLOWING WAY TO THE POLAR GRID
C     IMPOSED ON THE PHYSICAL REGION.
C        THE TOP LINE OF THE PRINT BLOCK CORRESPONDS TO VALUES OF THE
C        FUNCTION ALONG THE ARC R = 1.
C        THE BOTTOM LINE OF THE PRINT BLOCK CORRESPONDS TO THE VALUE
C        OF THE FUNCTION AT THE ORIGIN (R = 0).

      SUBROUTINE OUTPUT(VAR,ISOR,A)
      PARAMETER NR=10,NA=18,NRP1=NR+1,NAP1=NA+1
      DIMENSION A(NRP1,NAP1)
      DATA APHI,AW,AOMEGA/'PHI   ','W     ','OMEGA '/
   99 FORMAT(1H0 A6,5X I5,2X 'S0R ITERATIONS'/)
   98 FORMAT(1X F6.2,17F7.2,F6.2)
   97 FORMAT(1X F6.1,17F7.1,F6.1)
      KRP1=NRP1
      KAP1=NAP1
      WRITE(6,99) VAR,ISOR
      IF(VAR.NE.APHI) GO TO 8
      DO 5 I=1,NRP1
    5 WRITE(6,98) (A(KRP1-I+1,KAP1-J+1),J=1,KAP1)
      RETURN
    8 DO 10 I=1,NRP1
   10 WRITE(6,97) (A(KRP1-I+1,KAP1-J+1),J=1,KAP1)
      RETURN
      END
```

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 February 2025 doi:10.20944/preprints202411.1211.v4

Appendix B. Modernised Fortran Code

This appendix includes the modernised Fortran version of the original FORTRAN 66 code written by A.B.Schubert in 1972 [4].

Compiling and running the solution creates a separate file for each value of D.

```
C:\Fortran codes\Schubert_1972_modern_Fortran.f                          1
C     Appendix: FORTRAN Program for Secondary Flow, by A. B. Schubert (1972)
C
C     Nils T. Basse (2025):
C         Updates to remove obsolescent features (Hollerith strings and
C         data) and to write results to files instead of to punch cards.
C     Other changes marked by "NTB 2025" below.

C     TUBEFLOW    A PROGRAM WHICH COMPUTES THE SECONDARY
C     (CROSS-SECTIONAL) FLOW OF A FLUID IN A CURVED TUBE AND THE
C     VELOCITY AND FLUX IN THE AXIAL DIRECTION OF THE TUBE BY
C     DISCRETIZATION OF THE CROSS-SECTION AND OF THE GOVERNING
C     DIFFERENTIAL EQUATIONS.

C     NTB 2025:
C     Original (coarse) resolution: NR=10,NA=18
C     Updated (fine) resolution:    NR=20,NA=36
      INTEGER, PARAMETER :: NR = 10
      INTEGER, PARAMETER :: NA = 18

      PARAMETER NRP1=NR+1,NAP1=NA+1,NRM1=NR-1,NAM1=NA-1,
     * NAH=NA/2+1

C     NR = NUMBER OF GRID SPACES ON (0,1) IN RADIAL (R) DIRECTION.
C     NA = NUMBER OF GRID SPACES ON (0,PI) IN ANGULAR (ALPHA)
C     DIRECTION.

C     PRINCIPAL DISCRETE FUNCTIONS COMPUTED ARE DEFINED AS FOLLOWS.
C      PHI(I,J,K) = NON-DIMENSIONAL STREAM FUNCTION VALUE FOR SECONDARY
C                   FLOW AT POLAR GRID POINT CORRESPONDING TO I-TH
C                   RADIAL VALUE (WHERE I= 1 CORRESPONDS TO R = 0 AND
C                   I = NR+1 CORRESPONDS TO R = 1) AND J-TH ANGULAR
C                   VALUE (WHERE J = 1 CORRESPONDS TO ALPHA = 0 AND
C                   J = NA+1 CORRESPONDS TO ALPHA = PI.
C                   K = 1 IMPLIES A VALUE AT THE PREVIOUS OUTER
C                   ITERATION.
C                   K = 2 IMPLIES A VALUE AT THE PREVIOUS SOR ITERATION
C                   K = 3 IMPLIES A VALUE AT THE CURRENT SOR AND OUTER
C                   ITERATION.
C        W(I,J,K) = NON-DIMENSIONAL VELOCITY IN THE AXIAL DIRECTION,
C                   WITH SUBSCRIPTS DEFINED AS FOR PHI.
C    OMEGA(I,J,K) = INTERMEDIATE VARIABLE INTRODUCED IN ANALYTICAL
C                   DESCRIPTION OF PROBLEM, WITH SUBSCRIPTS DEFINED
C                   AS FOR PHI.

      DIMENSION PHI(NRP1,NAP1,4),W(NRP1,NAP1,4),OMEGA(NRP1,NAP1,4),
     * XI(4),RHO(3),EPS(3),SA(NAP1),COSA(NAP1),RINV(NRP1),RINV2(NRP1),
     * RHOC(3),B(NRP1,5),C(NRP1,NAP1),EPPS(3),XIC(4),E(NRP1,NAP1,6),
     * EE(NRP1),EF(NRP1),CA(NRP1,NAP1),DELA(NR),CO(NR),DOR(3)

C     NTB 2025:
C     MAXSOR and MAXOUT increased by a factor of ten for fine resolution
      DATA PI/3.14159255/,MAXSOR,MAXOUT/2500,600/,
     * NOR/3/,(DOR(I),I=1,3)
```

 doi:10.20944/preprints202411.1211.v4


```
     */.2,.2,.2/

C      NTB 2025:
C      Added definitions to write files
       CHARACTER (LEN=10) :: file_id
       CHARACTER (LEN=50) :: file_name
       INTEGER :: ctr

C      MAXSOR = MAXIMUM NUMBER OF ITERATIONS TO BE ALLOWED FOR COMPUTING
C               SOLUTION TO ANY SYSTEM BY SUCCESSIVE OVER-RELAXATION
C      MAXOUT = MAXIMUM NUMBER OF OUTER ITERATIONS TO BE ALLOWED IN THE
C               COMPUTATION OF PHI, W, AND OMEGA.
C         NOR = NUMBER OF DIFFERENT OVER-RELAXATION FACTORS TO BE TRIED
C               IN ANY DATA CASE.
C      DOR(I) = AMOUNT OF DECREASE IN OVER-RELAXATION FACTOR AT I-TH
C               CHANGE.

       KRP1=NRP1

C      DR = GRID SPACING IN RADIAL DIRECTION.
C      DA = GRID SPACING IN ANGULAR DIRECTION.

       DR=1./NR
       DA=PI/NA
       DAH=.5*DA
       DRH=.5*DR
       DRDAM=DR*DA
       DRDA=DR/DA
       DADR=DA/DR

C      COMPUTE ELEMENTS OF AREA, DELA(I),I=1,...,NR, IN RADIAL DIRECTION
C      FOR USE IN FLUX CALCULATION.

       DO 1 I=1,NR
     1 DELA(I)=(2*I-1)*DRH*DRDAM

C      ON THE GRID DEFINED BY DR AND DA, COMPUTE DISCRETE FUNCTIONS
C      DEPENDENT ONLY ON RADIUS AND ANGLE.

       SA(1)=0.
       DO 2 J=2,NA
       SA(J)=SIN((J-1)*DA)*DAH
     2 COSA(J)=COS((J-1)*DA)
       SA(NAP1)=0.
       COSA(NAP1)=-1.
       RINV(NRP1)=1.
       RINV2(NRP1)=1.
       DO 3 I=2,NR
       RINV(I)=1./((I-1)*DR)
       DRRH=DRH*RINV(I)
       RINV2(I)=RINV(I)**2
       DO 3 J=2,NA
     3 CA(I,J)=DRRH*COSA(J)
```

 doi:10.20944/preprints202411.1211.v4


```
C     COMPUTE SOR COEFFICIENTS FOR PHI.

      DO 4 I=2,NR
      BO=2.*(DADR+DRDA*RINV2(I))+DA*RINV(I)
      B(I,1)=(DADR+DA*RINV(I))/BO
      B(I,2)=DRDA*RINV2(I)/BO
      B(I,3)=DADR/BO
      B(I,4)=B(I,2)
    4 B(I,5)=DRDAM/BO

C     READ OUTER ITERATION SMOOTHING PARAMETERS (XI), OVER-RELAXATION
C     FACTORS (RHO), OUTER ITERATION CONVERGENCE TOLERANCES (EPS),
C     PARAMETER RELATED TO REYNOLDS NO. (D), AND INPUT/OUTPUT CONTROL
C     VARIABLES DEFINED AS FOLLOWS.
C     ISAVE.NE.0 IMPLIES SAVE THE SOLUTION FOR THIS CASE (IF OBTAINED)
C                IN MEMORY FOR USE AS STARTING VALUES FOR A SUCCEEDING
C                CASE WITH A DIFFERENT VALUE OF D.
C     ISAVE.EQ.0 IMPLIES DO NOT DO THE ABOVE.
C      IUSE.NE.0 IMPLIES TAKE STARTING VALUES FOR OUTER ITERATION FROM
C                MEMORY.
C      IUSE.EQ.0 IMPLIES INITIALIZE OUTER ITERATES TO ZERO.
C      IPCH.NE.0 IMPLIES WRITE SOLUTION FOR THIS CASE (IF OBTAINED)
C                TO FILE
C      IPCH.EQ.0 IMPLIES DO NOT WRITE SOLUTION OBTAINED FOR THIS CASE
C                TO FILE.

C     NTB 2025:
C     Save solution to memory
C     Take starting values from memory
C     Write solution to file
      ISAVE=1
      IUSE=1
      IPCH=1

C     NTB 2025:
C     Set initial conditions for all cases treated
      ctr=0
    5 ctr=ctr+1
      IF (ctr==1) THEN
        D=10
        EPS(1)=10.**(-5.)
        EPS(2)=10.**(-3.)
        EPS(3)=10.**(-4.)
        RHO(1)=1.5
        RHO(2)=1.8
        RHO(3)=1.5
        XI(1)=0.1
        XI(2)=0.1
        XI(3)=0.1
        XI(4)=0.1
      ELSE IF (ctr==2) THEN
        D=100
```

 doi:10.20944/preprints202411.1211.v4


```
        EPS(1)=2.*10.**(-4.)
        EPS(2)=5.*10.**(-3.)
        EPS(3)=5.*10.**(-3.)
        RHO(1)=1.5
        RHO(2)=1.7
        RHO(3)=1.5
        XI(1)=0.1
        XI(2)=0.1
        XI(3)=0.1
        XI(4)=0.1
      ELSE IF (ctr==3) THEN
        D=250
        EPS(1)=2.*10.**(-3.)
        EPS(2)=2.*10.**(-2.)
        EPS(3)=4.*10.**(-2.)
        RHO(1)=1.5
C       NTB 2025:
C       RHO(2) changed from 1.8 (iteration 1), then 1.5
C       to 1.5 for all iterations
        RHO(2)=1.5
        RHO(3)=1.5
        XI(1)=0.1
        XI(2)=0.1
        XI(3)=0.1
        XI(4)=0.1
      ELSE IF (ctr==4) THEN
        D=500
        EPS(1)=4.*10.**(-3.)
        EPS(2)=4.*10.**(-2.)
        EPS(3)=8.*10.**(-2.)
        RHO(1)=1.5
        RHO(2)=1.5
        RHO(3)=1.5
        XI(1)=0.1
        XI(2)=0.1
        XI(3)=0.1
        XI(4)=0.1
      ELSE IF (ctr==5) THEN
        D=1000
        EPS(1)=5.*10.**(-3.)
        EPS(2)=5.*10.**(-2.)
        EPS(3)=17.*10.**(-2.)
        RHO(1)=1.5
        RHO(2)=1.5
        RHO(3)=1.5
        XI(1)=0.5
        XI(2)=0.1
        XI(3)=0.1
        XI(4)=0.5
      ELSE IF (ctr==6) THEN
        D=2000
        EPS(1)=7.*10.**(-3.)
        EPS(2)=8.*10.**(-2.)
```

 doi:10.20944/preprints202411.1211.v4


```
        EPS(3)=3.*10.**(-1.)
C       NTB 2025:
C       RHO(1), RHO(2) and RHO(3) multiplied by a factor 0.8
        RHO(1)=0.8*1.5
        RHO(2)=0.8*1.5
        RHO(3)=0.8*1.3
        XI(1)=0.5
        XI(2)=0.1
        XI(3)=0.1
        XI(4)=0.5
      ELSE IF (ctr==7) THEN
        D=5000
        EPS(1)=10.**(-2.)
        EPS(2)=15.*10.**(-2.)
        EPS(3)=6.*10.**(-1.)
C       NTB 2025:
C       RHO(1), RHO(2) and RHO(3) multiplied by a factor 0.8
        RHO(1)=0.8*1.5
        RHO(2)=0.8*1.5
        RHO(3)=0.8*1.3
        XI(1)=0.3
        XI(2)=0.1
        XI(3)=0.1
        XI(4)=0.7
      ELSE
        PRINT *, 'D-loop complete: To exit, press <ENTER>'
        READ(*,*)
        STOP
      END IF

C     PRINT OUT INPUT PARAMETERS.

      WRITE(6,98) D,RHO,XI,EPS

C     COMPUTE PARAMETERS DEPENDENT ON THE INPUT PARAMETER D, INCLUDING
C     THE COEFFICIENTS, CO(I),I=1,...,NR, IN THE NUMERICAL INTEGRATION
C     FOR THE FLUX.

      DDRDAM=D*DRDAM
      DDR2=D*DR**2
      CON=4./(PI*D)
      CO(1)=16.*DELA(1)/(3.*PI*D)
      DO 105 I=2,NR
  105 CO(I)=CON*DELA(I)

C     COMPUTE COMPLEMENTS (RELATIVE TO 1) OF SMOOTHING PARAMETERS AND
C     OVER-RELAXATION FACTORS.
C     ALSO COMPUTE SOR CONVERGENCE TOLERANCES, EPPS(I),I=1,2,3, AS
C     FUNCTIONS OF OUTER ITERATION TOLERANCES, EPS(I),I=1,2,3.

      DO 6 I=1,3
      XIC(I)=1.-XI(I)
      RHOC(I)=1.-RHO(I)
```

 doi:10.20944/preprints202411.1211.v4


```
     6 EPPS(I)=.05*EPS(I)
       XIC(4)=1.-XI(4)

C      TEST WHETHER INITIAL OUTER ITERATES ARE TO BE OBTAINED FROM
C      MEMORY, HAVING BEEN PLACED THERE AS THE SOLUTION FOR A PREVIOUS
C      VALUE OF D BY A PREVIOUS COMPUTATION THIS RUN.

       IF(IUSE.EQ.0) GO TO 7
       DO 106 I=1,NRP1
       DO 106 J=1,NAP1
       PHI(I,J,3)=PHI(I,J,4)
       W(I,J,3)=W(I,J,4)
   106 OMEGA(I,J,3)=OMEGA(I,J,4)
       WRITE(6,87) DSTART
       GO TO 9

C      INITIALIZE OUTER ITERATES TO ZERO IF NEITHER INPUT NOR COMPUTED
C      PREVIOUSLY THIS RUN.

     7 DO 8 I=1,NRP1
       DO 8 J=1,NAP1
       PHI(I,J,3)=0.
       W(I,J,3)=0.
     8 OMEGA(I,J,3)=0.

C      INITIALIZE OUTER ITERATION COUNTER (IOUT) AND COUNTERS OF NUMBER
C      OF OVER-RELAXATION FACTORS USED FOR W AND OMEGA (IRW,IRO, RESP.)
C      FOP THIS DATA CASE.

     9 IOUT=0
       IRW=0
       IRO=0

C      TEST FOR MAXIMUM NUMBER OF OUTER ITERATIONS.

    10 IF(IOUT.GE.MAXOUT) GO TO 58

C      UPDATE OUTER ITERATION COUNTER, WRITE MESSAGE INDICATING NEW
C      OUTER ITERATION NUMBER, AND SET OUTER ITERATION CONVERGENCE
C      INDICATOR (ICV) TO ZERO. AFTER COMPUTATION OF ALL CURRENT OUTER
C      ITERATES AND COMPARISON WITH PREVIOUS OUTER ITERATES AGAINST THE
C      SPECIFIED TOLERANCES, ICV WILL STILL BE ZERO IF CONVERGENCE HAS
C      BEEN OBTAINED, OTHERWISE ICV WILL BE NON-ZERO.

       IOUT=IOUT+1
       WRITE(6,92) IOUT
       ICV=0

C      UPDATE PREVIOUS OUTER ITERATES.
       DO 12 I=1,NRP1
       DO 12 J=1,NAP1
       PHI(I,J,1)=PHI(I,J,3)
       W(I,J,1)=W(I,J,3)
```

 doi:10.20944/preprints202411.1211.v4


```
   12 OMEGA(I,J,1)=OMEGA(I,J,3)

C     COMPUTE CONSTANT SOR COEFFICIENT FOR PHI.

      DO 13 I=2,NR
      DO 13 J=2,NA
   13 C(I,J)=B(I,5)*OMEGA(I,J,1)

C     INITIALIZE SOR ITERATION COUNTER FOR PHI.

      ISOR=0

C     TEST FOR MAXTMUM NUMBER OF SOR ITERATIONS FOR PHI.

   14 IF(ISOR.GE.MAXSOR) GO TO 55

C     UPDATE SOR ITERATION COUNTER IF MAXIMUM NUMBER HAS NOT BEEN
C     ACHIEVED.

      ISOR=ISOR+1

C     UPDATE PREVIOUS SOR ITERATES FOR PHI.

      DO 16 I=2,NR
      DO 16 J=2,NA
   16 PHI(I,J,2)=PHI(I,J,3)

C     SET SOR ITERATION CONVERGENCE INDICATOR (ICONV) TO ZERO. AFTER
C     COMPUTATION OF CURRENT SOR ITERATE AND COMPARISON WITH PREVIOUS
C     ITERATE AGAINST THE TOLERANCE, ICONV WILL STILL BE ZERO IF
C     CONVERGENCE HAS BEEN OBTAINED. OTHERWISE IT WILL BE NON-ZERO.

      ICONV=0

C     COMPUTE CURRENT SOR ITERATE FOR PHI...
C     ...FIRST ON INTERIOR OF REGION, EXCLUDING 'INNER BOUNDARY'
C        R = 1.-DR.

      DO 18 I=2,NRM1
      DO 18 J=2,NA
        PHI(I,J,3)=RHOC(1)*PHI(I,J,2)+RHO(1)*(B(I,1)*PHI(I+1,J,2)
     * +B(I,2)*PHI(I,J+1,2)+B(I,3)*PHI(I-1,J,3)+B(I,4)*PHI(I,J-1,3)
     * +C(I,J))
      IF(ABS(PHI(I,J,2)-PHI(I,J,3)).GT.EPPS(1)) ICONV=1
   18 CONTINUE

C     ...THEN ON 'INNER BOUNDARY' R = 1.-DR.

      DO 19 J=2,NA
        PHI(NR,J,3)=RHOC(1)*PHI(NR,J,2)+RHO(1)*.25*PHI(NRM1,J,3)
        IF(ABS(PHI(NR,J,2)-PHI(NR,J,3)).GT.EPPS(1)) ICONV=1
   19 CONTINUE
```

 doi:10.20944/preprints202411.1211.v4


```
C     TEST FOR CONVERGENCE OF SOR ITERATION FOR PHI.

      IF(ICONV.NE.0) GO TO 14

C     SOR CONVERGENCE ATTAINED FOR PHI. SMOOTH OUTER ITERATE.
      DO 20 I=2,NR
      DO 20 J=2,NA
        PHI(I,J,3)=XI(1)*PHI(I,J,1)+XIC(1)*PHI(I,J,3)
        IF(ABS(PHI(I,J,1)-PHI(I,J,3)).GT.EPS(1)) ICV=1
   20 CONTINUE

C     PRINT OUT SMOOTHED CURRENT OUTER ITERATE FOR PHI.

C     NTB 2025:
C     Added NR and NA as inputs
      CALL OUTPUT('PHI  ',ISOR,PHI(1,1,3),NR,NA)

C     COMPUTE COEFFICIENTS FOR SOR ITERATION FOR W...
C     ...FIRST COMPUTE COEFFICIENTS AT THE ORIGIN

      E0=4.+ABS(PHI(2,NAH,3))
      E1=(1.-AMIN1(PHI(2,NAH,3),0.))/E0
      E2=2./E0
      E3=(1.+AMAX1(PHI(2,NAH,3),0.))/E0
      E4=DDR2/E0

C     ...NEXT COMPUTE COEFFICIENTS FOR REMAINDER OF REGION...
C           FIRST FOR RAYS ALONG ALPHA = 0 AND ALPHA = PI, EXCLUDING
C           R = 0 AND R = 1.

      DO 23 I=2,NR
      DELTA1=DA-PHI(I,2,3)
      DELTA2=DA+PHI(I,NA,3)
      EE(I)=2.*(DADR+DRDA*RINV2(I))
      EE1=EE(I)+RINV(I)*ABS(DELTA1)
      EE2=EE(I)+RINV(I)*ABS(DELTA2)
      E(I,1,1)=(DADR+RINV(I)*AMAX1(DELTA1,0.))/EE1
      E(I,NAP1,1)=(DADR+RINV(I)*AMAX1(DELTA2,0.))/EE2
      EF(I)=DRDA*RINV2(I)
      E(I,1,2)=EF(I)/EE1
      E(I,NAP1,2)=EF(I)/EE2
      E(I,1,3)=(DADR-RINV(I)*AMIN1(DELTA1,0.))/EE1
      E(I,NAP1,3)=(DADR-RINV(I)*AMIN1(DELTA2,0.))/EE2
      E(I,1,4)=E(I,1,2)
      E(I,NAP1,4)=E(I,NAP1,2)
      E(I,1,5)=DDRDAM/EE1
   23 E(I,NAP1,5)=DDRDAM/EE2

C     THEN ON INTERIOR OF REGION.

      DO 25 I=2,NR
      DO 25 J=2,NA
      GAMMA=.5*(PHI(I+1,J,3)-PHI(I-1,J,3))
```

 doi:10.20944/preprints202411.1211.v4


```
      DELTA=DA-.5*(PHI(I,J+1,3)-PHI(I,J-1,3))
      E(I,J,6)=EE(I)+RINV(I)*(ABS(GAMMA)+ABS(DELTA))
      E(I,J,1)=(DADR+RINV(I)*AMAX1(DELTA,0.))/E(I,J,6)
      E(I,J,2)=(EF(I)+RINV(I)*AMAX1(GAMMA,0.))/E(I,J,6)
      E(I,J,3)=(DADR-RINV(I)*AMIN1(DELTA,0.))/E(I,J,6)
      E(I,J,4)=(EF(I)-RINV(I)*AMIN1(GAMMA,0.))/E(I,J,6)
   25 E(I,J,5)=DDRDAM/E(I,J,6)

C     INITIALIZE SOR ITERATION COUNTER FOR W.

   26 ISOR=0

C     TEST FOR MAXIMUM NUMBER OF SOR ITERATIONS.

   27 IF(ISOR.GE.MAXSOR) GO TO 56

C     UPDATE SOR ITERATION COUNTER IF MAXIMUM HAS NOT BEEN ACHIEVED.

      ISOR=ISOR+1

C     UPDATE PREVIOUS SOR ITERATES FOR W...
C     ...FIRST AT THE ORIGIN

      W(1,1,2)=W(1,1,3)

C     ...THEN ON REMAINDER OF REGION EXCLUDING R = 1.

      DO 29 I=2,NR
      DO 29 J=1,NAP1
   29 W(I,J,2)=W(I,J,3)

C     SET SOR CONVERGENCE INDICATOR (ICONV) TO ZERO.

      ICONV=0

C     COMPUTE CURRENT SOR ITERATE FOR W...
C     ...FIRST AT THE ORIGIN

      W(1,1,3)=RHOC(2)*W(1,1,2)+RHO(2)*(E1*W(2,1,2)+E2*W(2,NAH,2)
     * +E3*W(2,NAP1,2)+E4)

C     (SET VALUES OF W FOR R = 0 AND ALL ANGLES ALPHA(I),I=1,...,NA+1,
C     EQUAL TO VALUE OF W AT ORIGIN FOR CONVENIENCE IN SOR COMPUTATION.)

      DO 30 J=2,NAP1
   30 W(1,J,3)=W(1,1,3)
      IF(ABS(W(1,1,2)-W(1,1,3)).GT.EPPS(2)) ICONV=1

C     ...NEXT ALONG RAY ALPHA = 0. EXCLUDING R = 0 AND R = 1.

      DO 32 I=2,NR
      W(I,1,3)=RHOC(2)*W(I,1,2)+RHO(2)*(E(I,1,1)*W(I+1,1,2)+E(I,1,2)*2.*
     * W(I,2,2)+E(I,1,3)*W(I-1,1,3)+E(I,1,5))
```

 doi:10.20944/preprints202411.1211.v4


```
      IF(ABS(W(I,1,3)-W(I,1,2)).GT.EPPS(2)) ICONV=1
   32 CONTINUE

C     ...NEXT ON INTERIOR OF REGION

      DO 33 I=2,NR
      DO 33 J=2,NA
      W(I,J,3)=RHOC(2)*W(I,J,2)+RHO(2)*(E(I,J,1)*W(I+1,J,2)+E(I,J,2)*
     * W(I,J+1,2)+E(I,J,3)*W(I-1,J,3)+E(I,J,4)*W(I,J-1,3)+E(I,J,5))
      IF(ABS(W(I,J,2)-W(I,J,3)).GT.EPPS(2)) ICONV=1
   33 CONTINUE

C     ...FINALLY ALONG RAY ALPHA = PI. EXCLUDING R = 0 AND R = 1.

      DO 34 I=2,NR
      W(I,NAP1,3)=RHOC(2)*W(I,NAP1,2)+RHO(2)*(E(I,NAP1,1)*W(I+1,NAP1,2)+
     * E(I,NAP1,3)*W(I-1,NAP1,3)+2.*E(I,NAP1,4)*W(I,NA,3)+E(I,NAP1,5))
      IF(ABS(W(I,NAP1,2)-W(I,NAP1,3)).GT.EPPS(2)) ICONV=1
   34 CONTINUE

C     TEST FOR CONVERGENCE OF SOR ITERATION FOR W.

      IF(ICONV.NE.0) GO TO 27

C     SOR CONVERGENCE ATTAINED FOR W. SMOOTH OUTER ITERATE...
C     ...FIRST AT THE ORIGIN

      W(1,1,3)=XI(2)*W(1,1,1)+XIC(2)*W(1,1,3)

C     (SET VALUES OF W FOR R = 0 AND ALL DISCRETE ANGLES EQUAL TO
C     SMOOTHED VALUE OF W AT ORIGIN.)

      DO 134 J=2,NAP1
  134 W(1,J,3)=W(1,1,3)
      IF(ABS(W(1,1,1)-W(1,1,3)).GT.EPS(2)) ICV=1

C     ...THEN ON REMAINDER OF REGION, EXCLUDING R = 1.

      DO 35 I=2,NR
      DO 35 J=I,NAP1
      W(I,J,3)=XI(2)*W(I,J,1)+XIC(2)*W(I,J,3)
      IF(ABS(W(I,J,1)-W(I,J,3)).GT.EPS(2)) ICV=1
   35 CONTINUE

C     PRINT OUT SMOOTHED CURRENT OUTER ITERATE FOR W.

C     NTB 2025:
C     Added NR and NA as inputs
      CALL OUTPUT('W    ',ISOR,W(1,1,3),NR,NA)

C     COMPUTE AND SMOOTH OMEGA ON BOUNDARY R = 1.

      DO 37 J=2,NA
```

 doi:10.20944/preprints202411.1211.v4


```
      OMEGA(NRP1,J,3)=XI(3)*OMEGA(NRP1,J,1)-XIC(3)*2.*RINV2(2)*PHI(NR,
     * J,3)
        IF(ABS(OMEGA(NRP1,J,1)-OMEGA(NRP1,J,3)).GT.EPS(3)) ICV=1
   37 CONTINUE

C     COMPUTE CONSTANT COEFFICIENT FOR SOR ITERATION FOR OMEGA.

      DO 40 I=2,NR
      DO 40 J=2,NA
   40 E(I,J,6)=-W(I,J,3)*(SA(J)*(W(I+1,J,3)-W(I-1,J,3))+CA(I,J)*(W(I,
     * J+1,3)-W(I,J-1,3)))/E(I,J,6)

C     INITIALIZE COUNTER OF SOR ITERATION NUMBER FOR OMEGA.

   41 ISOR=0

C     TEST FOR MAXIMUM NUMBER OF SOR ITERATIONS.

   42 IF(ISOR.GE.MAXSOR) GO TO 57

C     UPDATE SOR ITERATION COUNTER IF MAXIMUM NUMBER HAS NOT BEEN
C     ACHIEVED.

      ISOR=ISOR+1

C     UPDATE PREVIOUS SOR ITERATES FOR OMEGA ON INTERIOR.

      DO 44 I=2,NR
      DO 44 J=2,NA
   44 OMEGA(I,J,2)=OMEGA(I,J,3)

C     SET SOR CONVERGENCE INDICATOR TO ZERO.

      ICONV=0

C     COMPUTE CURRENT SOR ITERATE FOR OMEGA ON INTERIOR.

      DO 46 I=2,NR
      DO 46 J=2,NA
      OMEGA(I,J,3)=RHOC(3)*OMEGA(I,J,2)+RHO(3)*(E(I,J,1)*OMEGA(I+1,J,2)
     * +E(I,J,2)*OMEGA(I,J+1,2)+E(I,J,3)*OMEGA(I-1,J,3)+E(I,J,4)
     * *OMEGA(I,J-1,3)+E(I,J,6))
      IF(ABS(OMEGA(I,J,2)-OMEGA(I,J,3)).GT.EPPS(3)) ICONV=1
   46 CONTINUE

C     TEST FOR CONVERGENCE OF SOR ITERATION FOR OMEGA.

      IF(ICONV.NE.0) GO TO 42

C     SOR CONVERGENCE ATTAINED FOR OMEGA. SMOOTH OUTER ITERATE.

      DO 48 I=2,NR
      DO 48 J=2,NA
```

 doi:10.20944/preprints202411.1211.v4


```
      OMEGA(I,J,3)=XI(4)*OMEGA(I,J,1)+XIC(4)*OMEGA(I,J,3)
      IF(ABS(OMEGA(I,J,1)-OMEGA(I,J,3)).GT.EPS(3)) ICV=1
   48 CONTINUE

C     PRINT OUT SMOOTHED CURRENT OUTER ITERATE FOR OMEGA.

C     NTB 2025:
C     Added NR and NA as inputs
      CALL OUTPUT('OMEGA',ISOR,OMEGA(1,1,3),NR,NA)

C     TEST FOR CONVERGENCE OF ALL OUTER ITERATES.

      IF(ICV.NE.0) GO TO 10

C     PRINT MESSAGE INDICATING CONVERGENCE OF OUTER ITERATION.

      WRITE(6,97)

C     COMPUTE RATIO OF FLUX IN CURVED TUBE TO THAT IN STRAIGHT TUBE.

      QR=0.
      DO 49 J=1,NA
   49 QR=QR+W(1,J,3)+W(2,J,3)+W(2,J+1,3)
      QR=QR*CO(1)
      DO 249 I=2,NR
      S=0.
      DO 149 J=1,NA
  149 S=S+W(I,J,3)+W(I+1,J,3)+W(I,J+1,3)+W(I+1,J+1,3)
  249 QR=QR+CO(I)*S
C     PRINT OUT VALUE OF FLUX RATIO JUST COMPUTED.

      WRITE(6,90) QR

C     WRITE SOLUTION TO FILE IF INPUT CONTROL
C     PARAMETER IPCH SO DICTATES, AND PRINT MESSAGE INDICATING THAT
C     THIS FILE WRITING WAS DONE.

C     NTB 2025:
C     Write solution to file
      IF (IPCH.NE.0) THEN
          WRITE(file_id, '(i0)') INT(D)
          file_name = 'file_D' // TRIM(ADJUSTL(file_id)) // '.dat'
          PRINT*,"file name is ",file_name
          OPEN(1,FILE = file_name,STATUS = 'replace',RECL = 600000)
          WRITE(1,*) NRP1
          WRITE(1,*) NAP1
          WRITE(1,*) XI
          WRITE(1,*) RHO
          WRITE(1,*) EPS
          WRITE(1,*) D
          WRITE(1,*) QR
          DO 350 I=1,NRP1
  350     WRITE(1,*) PHI(I,:,3)
```

 doi:10.20944/preprints202411.1211.v4


```
          DO 360 I=1,NRP1
  360     WRITE(1,*) W(I,:,3)
          DO 370 I=1,NRP1
  370     WRITE(1,*) OMEGA(I,:,3)
          CLOSE(1)
      END IF

      IF(IPCH.NE.0) WRITE(6,86)

C     TEST WHETHER OR NOT SOLUTION JUST OBTAINED IS TO BE SAVED IN
C     MEMORY FOR SOME SUCCEEDING CASE WITH A DIFFERENT VALUE OF D.

      IF(ISAVE.EQ.0) GO TO 5

C     SAVE CURRENT SOLUTION IN ANOTHER AREA OF MEMORY.

      DO 50 I=1,NRP1
      DO 50 J=1,NAP1
      PHI(I,J,4)=PHI(I,J,3)
      W(I,J,4)=W(I,J,3)
   50 OMEGA(I,J,4)=OMEGA(I,J,3)

C     SAVE VALUE OF D FOR WHICH SOLUTION WAS JUST OBTAINED IN DSTART.

      DSTART=D
      GO TO 5

C     CONVERGENCE FAILURE MESSAGES

C     PHI ITERATION FAILED. READ NEXT DATA CASE.

   55 WRITE(6,96)
      GO TO 5

C     W ITERATION FAILED. REDUCE OVER-RELAXATION FACTOR AND TRY AGAIN,
C     UNLESS THIS HAS ALREADY BEEN DONE THE MAXIMUM NUMBER OF TIMES
C     (NOR), IN WHICH CASE READ NEXT DATA CASE.

   56 WRITE(6,95) RHO(2)
      IF(IRW.GE.NOR) GO TO 5
      IRW=IRW+1
      RHO(2)=RHO(2)-DOR(IRW)
      RHOC(2)=1.-RHO(2)
      DO 156 I=1,NRP1
      DO 156 J=1,NAP1
  156 W(I,J,3)=W(I,J,1)
      GO TO 26

C     OMEGA ITERATION FAILED. REDUCE O-R FACTOR AND TRY AGAIN, UNLESS
C     THIS HAS ALREADY BEEN DONE THE MAXIMUM NUMBER OF TIMES (NOR),
C     IN WHICH CASE READ THE NEXT DATA CASE.

   57 WRITE(6,94) RHO(3)
```

 doi:10.20944/preprints202411.1211.v4


```
      IF(IRO.GE.NOR) GO TO 5
      IRO=IRO+1
      RHO(3)=RHO(3)-DOR(IRO)
      RHOC(3)=1.-RHO(3)
      DO 157 I=1,NRP1
      DO 157 J=1,NAP1
  157 OMEGA(I,J,3)=OMEGA(I,J,1)
      GO TO 41

C     OUTER ITERATION FAILED. PRINT MESSAGE INDICATING THIS AND READ
C     NEXT DATA CASE.

   58 WRITE(6,93)
      GO TO 5

C     TERMINATION POINT FOR PROGRAM. CONTROL REACHES HERE AFTER ATTEMPT
C     TO READ PAST LAST DATA RECORD.

   70 STOP

   98 FORMAT(6X 'D =' F5.0//13X 'PHI' 7X 'W' 5X 'OMEGA' // 5X
     * 'RHO =' F6.2,2(3X F6.2)/6X 'XI =' F6.4,3(3X F6.4)/5X 'EPS ='
     * F6.4,2(3X F6.4))
   97 FORMAT('OUTER ITERATION CONVERGED TO GIVEN TOLERANCES.')
   96 FORMAT('SOR FOR   PHI FAILED.')
   95 FORMAT('SOR FOR     W FAILED WITH SOR FACTOR =' F6.2)
   94 FORMAT('SOR FOR OMEGA FAILED WITH SOR FACTOR =' F6.2)
   93 FORMAT('OUTER ITERATION FAILED TO CONVERGE.')
   92 FORMAT(//'OUTER ITERATION' I5//)
   90 FORMAT('FLUX RATIO =' F10.5)
   87 FORMAT('INITIAL ITERATE TAKEN FROM SOLUTION FOR D =' F7.0)
   86 FORMAT('SOLUTION WAS OUTPUT TO FILE.')

      END

C     THIS ROUTINE PRINTS OUT A DISCRETE FUNCTION IN A RECTANGULAR
C     FORMAT WHICH IS RELATED TN THE FOLLOWING WAY TO THE POLAR GRID
C     IMPOSED ON THE PHYSICAL REGION.
C     THE TOP LINE OF THE PRINT BLOCK CORRESPONDS TO VALUES OF THE
C     FUNCTION ALONG THE ARC R = 1.
C     THE BOTTOM LINE OF THE PRINT BLOCK CORRESPONDS TO THE VALUE
C     OF THE FUNCTION AT THE ORIGIN (R = 0).

C     NTB 2025:
C     Added x (NR) and y (NA) as inputs
      SUBROUTINE OUTPUT(VAR,ISOR,A,x,y)

C     NTB 2025:
C     Added definitions
      CHARACTER*5 APHI, AW, AOMEGA, VAR
      INTEGER :: x,y
      DIMENSION A(x+1,y+1)
      NRP1=x+1
```

 doi:10.20944/preprints202411.1211.v4


```
C:\Fortran codes\Schubert_1972_modern_Fortran.f                    15
      NAP1=y+1

      DATA APHI /'PHI'/, AW /'W     '/, AOMEGA /'OMEGA'/
   99 FORMAT(A6,5X I5,2X 'SOR ITERATIONS'/)
   98 FORMAT(1X F6.2,17F7.2,F6.2)
   97 FORMAT(1X F6.1,17F7.1,F6.1)
      KRP1=NRP1
      KAP1=NAP1
      WRITE(6,99) VAR,ISOR
      IF(VAR.NE.APHI) GO TO 8
      DO 5 I=1,NRP1
    5 WRITE(6,98) (A(KRP1-I+1,KAP1-J+1),J=1,KAP1)
      RETURN
    8 DO 10 I=1,NRP1
   10 WRITE(6,97) (A(KRP1-I+1,KAP1-J+1),J=1,KAP1)
      RETURN
      END
```

References

1. Dean, W.R. Note on the motion of fluid in a curved pipe. *Phil. Mag.* **1927**, *4*, 208-223.
2. Dean, W.R. The stream-line motion of fluid in a curved pipe. *Phil. Mag.* **1928**, *5*, 673-695.
3. McConalogue, D.J.; Srivastava, R.S. Motion of fluid in a curved tube. *Proc. Roy. Soc. A.* **1968**, *307*, 37-53.
4. Greenspan, D; Schubert, A.B. Secondary flow in a curved tube. *University of Wisconsin, Computer Sciences Department, Technical Report #155* **1972**. Available online: https://minds.wisconsin.edu/handle/1793/57756
5. Greenspan, D. Secondary flow in a curved tube. *J. Fluid Mech.* **1973**, *57*, 167-176.
6. Collins, W.M.; Dennis, S.C.R. The steady motion of a viscous fluid in a curved tube. *Quart. J. Mech. Appl. Math.* **1975**, *28*, 133-156.
7. Taylor, G.I. The criterion for turbulence in curved pipes. *Proc. Roy. Soc. A.* **1929**, *124*, 243-249.
8. Krom, J.G. The evolution of control and data acquisition at JET. *Fus. Eng. Des.* **1999**, *43*, 265–273.
9. EasyOCR. Available online: https://pypi.org/project/easyocr/
10. PyTesseract. Available online: https://pypi.org/project/pytesseract/
11. PDF.ai. Available online: https://pdf.ai/tools/ocr-pdf
12. MaxAI.me. Available online: https://www.maxai.me/pdf-tools/pdf-ocr/
13. Amazon Web Services. Available online:; https://aws.amazon.com/
14. Metcalf, M.; Reid, J. *Fortran 90/95 Explained*; Oxford University Press: Oxford, UK, 1996.
15. OpenAI.com. Available online: https://chatgpt.com/
16. ChatGPT-4 prompt example. Available online: https://chatgpt.com/share/671ccb3e-7ce0-8006-93ae-3c08828444b4
17. Microsoft Windows 11 Enterprise. Available online: https://www.microsoft.com/en-us/microsoft-365/windows/windows-11-enterprise
18. Microsft Visual Studio 2022. Available online: https://visualstudio.microsoft.com/downloads/
19. Intel oneAPI HPC Toolkit 2024.2. Available online: https://www.intel.com/content/www/us/en/developer/tools/oneapi/hpc-toolkit.html#gs.i3q2al
20. GNU Octave, version 9.2.0. Available online: https://octave.org/
21. Canton, J.; Örlü, R.; Schlatter, P. Characterisation of the steady, laminar incompressible flow in toroidal pipes covering the entire curvature range. *International Journal of Heat and Fluid Flow* **2017**, *66*, 95-107.
22. Canton, J.; Rinaldi, E.; Örlü, R.; Schlatter, P. Critical point for bifurcation cascades and featureless turbulence. *Phys. Rev. Lett.* **2020**, *124*, 014501.
23. Snyder, B.; Hammersley, J.R.; Olson, D.E. The axial skew of flow in curved pipes. *J. Fluid Mech.* **1985**, *161*, 281-294.
24. Kühnen, J.; Holzner, M.; Hof, B.; Kuhlmann, H.C. Experimental investigation of transitional flow in a toroidal pipe. *J. Fluid Mech.* **2014**, *738*, 463-491.
25. Roache, P.J. *Computational Fluid Dynamics*; Hermosa Publications: Albuquerque, New Mexico, USA, 1972.

 doi:10.20944/preprints202411.1211.v4

26. Jones, A.R. Numerical archaeology: Gleanings from the 1253 building accounts of Westminster Abbey revisited. In: Lunnon, H.; Rodwell, W.; Tatton-Brown, T. (Eds.) *Westminster Part I: The Art, Architecture and Archaeology of the Royal Abbey*; Routledge: Abingdon, UK, 2017.
27. Basse, N.T. The chimera revisited: Wall- and magnetically-bounded turbulent flows. *Fluids* **2024**, *9*, 34.