ISSN:0975-9646

Computer Assisted Solution of Homotopy Continuation Method Algorithm for Darcy-Brinkman Forchheimer Flow through an 3D Animation Visualization

Ashoka S. B. M.C.A., M.Phil., [Ph.D.]a,b.
Research Scholar
*[*a] Department of Computer Science and Applications*
Bangalore University, Bangalore–560 056, INDIA
[b] Assistant Professor, Department of Computer Science
Govt. First Grade College, Jayanagara, Bangalore-70.

Abstract-An algorithm is a systematic method containing a sequence of instructions to solve a computational problem. It takes some inputs, performs a well defined sequence of steps, and produces some output. Once we design an algorithm, we need to know how well it performs on any input. A major criterion for a good algorithm is its efficiency that is, how much time and memory are required to solve a particular problem. Intuitively, time and memory can be measured in real units such as seconds and megabytes. However, these measurements are not subjective for comparisons between algorithms, because they depend on the computing power of the specific machine and on the specific data set. To standardize the measurement of algorithm efficiency. Algorithm can be written in pseudo-code because simplification of the actual implementation is very good way. In this present paper mainly focus on homotopy continuation method algorithm for solving system of non linear algebraic equation, given the generalized uniform approach of three more problems result. We comparison HCM observation results (both time and space complexity) with other different methods algorithms observations. The detailed observations of all the methods of algorithm noted and platen. All algorithm yields unique approach on the results with limiting conditions. But our one of proposed works HCM algorithm given a excellent results, for all set of parameters with high end values. One more highlight of this chapter is obtained the solution also represented through 3D visualization using open source code Mayavi version 0.6.

Key words: Computer Graphics, Darcy, Brikman and Forchheimer number, HCM, 3D, Modeling, texturing, Compositing, Pre-Production, Production and Post Production.

Introduction

Time complexity is a function describing the amount of time an algorithm takes in terms of the amount of input to the algorithm. "Time" can mean the number of memory accesses performed, the number of comparisons between integers, the number of times some inner loop is executed, or some other natural unit related to the amount of real time the algorithm will take. We try to keep this idea of time separate from "clock ticks" time, since many factors unrelated to the algorithm itself can affect the real time (like the language used, type of computing hardware, proficiency of the programmer, code optimization in the compiler, etc.). It turns out that, if we chose the units wisely, all of the other stuff doesn't matter and we can get an independent measure of the efficiency of the algorithm. Time complexity is a measure efficient algorithm plays the major role in determining the running time.

Space complexity is a function describing the amount of memory (space) an algorithm takes in terms of the amount of input to the algorithm. We often speak of "extra" memory needed, not counting the memory needed to store the input itself. Again, we use natural (but fixed-length) units to measure this. We can use bytes, but it's easier to use, say, number of integers used, number of fixed-sized structures, etc. In the end, the function we come up with will be independent of the actual number of bytes needed to represent the unit. Space complexity is sometimes ignored because the space used is minimal and/or obvious, but sometimes it becomes as important an issue as time. Space complexity is a measure of the amount of working storage an algorithm needs. That means how much memory, in the worst case, is needed at any point in the algorithm. As with time complexity, we're mostly concerned with how the space needs grow, in big-Oh terms, as the size N of the input problem grows.
In space complexity, auxiliary space is extra space or temporary space used by the algorithm, which is mostly used in algorithm where we use swapping or temporary variables. The Space complexity means total space taken by the algorithm with respect to input size. Space complexity calculated by both auxiliary space and space used by the input.

The exact speed of an algorithm depends on where the algorithm is run, as well as the exact details of its implementation, computer scientists typically talk about the runtime relative to the size of the input. The time complexity of an algorithm is commonly expressed using big O notation, which excludes coefficients and lower order terms. When expressed this way, the time complexity is said to be described asymptotically, i.e., as the input size goes to infinity. For some optimization problems, we can reach an improved time complexity, but it seems that we have to pay for this with an exponential space complexity. Note that algorithms with exponential space complexities are absolutely useless for real life applications. Theoretical computer science has its uses and applications and can turn out to be quite practical. In this article, targeted at programmers who know their art but who don't have any theoretical computer science background, I will present one of the most pragmatic tools of computer science: Big O notation and algorithm complexity analysis. In this chapter we will try to found the computational complexity of our generalized module of HCM. Computational complexity can be further divided into time complexity and space complexity, which estimate the time and memory requirements of an algorithm, respectively. In general, time complexity is considered much more important than space complexity, in part because the memory requirement of most algorithms is lower than the capacity of current machines. In the rest of the section, all calculations and comparisons of algorithm efficiency refer to time complexity as complexity unless otherwise specified. Also, time complexity and running time can be used interchangeably in most of the cases. The time complexity of an algorithm is calculated on the basis of the number of required elementary computational steps that are interpreted as a function of the input size. Most of the time, because of the presence of conditional constructs (e.g., if-else statements) in an algorithm, the number of necessary steps differs from input to input. Thus, average-case complexity should be a more meaningful characterization of the algorithm. However, its calculations are often difficult and complicated, which necessitates the use of a worst-case complexity metric. An algorithm's worst-case complexity is its complexity with respect to the worst possible inputs, which gives an upper bound on the average-case complexity. As we shall see, the worst-case complexity may sometimes provide a decent approximation of the average-case complexity. The theory of computational complexity was developed Ullman (1984) Papadimitriou (1993, 1998), Wilf (2002). This allows an algorithm's efficiency to be estimated and expressed

ALGORITHM FOR HOMOTOPY CONTINUATION METHOD

```
Step1: Start.
Step2: Accept initial parameters values [$x_0$, $x_1$,….xn].
Step3: FOR W = 0 DO 4. // Chosen $u_{j+1}$ closer to $u(t_{j+1})$
Step4: FOR K = 0 DO 4. //Calculate slope at the
       beginning, midpoint and end point.
Step5: Find
       a). SUBROUTINE Normal_constant_matrix().
       b). SUBROUTINE Differential_matrix().
       c). SUBROUTINE Inverse_matrix().
       d). SUBROUTINE Multiplication_matrix().
Step6: Calculate
```

$$k_1 = f(t_n, y_n),$$
$$k_2 = f(t_n + \tfrac{h}{2}, y_n + \tfrac{h}{2}k_1),$$
$$k_3 = f(t_n + \tfrac{h}{2}, y_n + \tfrac{h}{2}k_2),$$
$$k_4 = f(t_n + h, y_n + hk_3).$$

Where n = 0, 1, 2, 3, . . .

k_1 is the increment based on the slope at the beginning of the interval, using $\dot{y}$,

k_2 is the increment based on the slope at the midpoint of the interval, using $\dot{y} + \frac{h}{2}k_1$;

k_3 is again the increment based on the slope at the midpoint, but now using $\dot{y} + \frac{h}{2}k_2$;

k_4 is the increment based on the slope at the end of the interval, using $\dot{y} + hk_3$.

```
Step7: End of " K " loop.
Step8: Calculate
```

$$y_{n+1} = y_n + \tfrac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$
$$t_{n+1} = t_n + h$$

Now pick a step-size $h > 0$

```
Step9: End of " W " loop.
Step10: Print "Roots".
Step11: Stop.
```

SUBROUTINE Normal_constant_matrix(set of parameters received **)**

```
Step1: Start.
Step2: Accept A, B, N, v, δ.
where  A = $\Lambda Da^2$
       B = $C_F \Lambda \operatorname{Re} Da$ .
Step3: For n = 0 DO (n-1).
Repeat the step4 to step5
till n=0 to (n-1).
Step4: Calculate
```

$$y_n(n) = \varepsilon + n\frac{(1-\varepsilon)}{N}$$

$$G(y_n) = e^{\frac{\log(1+v)}{1-\varepsilon}(y_n-\varepsilon)}$$

$$G'(y_n) = \frac{\log(1+v)}{1-\varepsilon}\, e^{\frac{\log(1+v)}{1-\varepsilon}(y_n-\varepsilon)}$$

where $\varepsilon = R_2 R_1^{-1}$ (ratio of inner to outer cylinder radii).

v variable viscosity.

```
Step5: Calculate
```

$$f_n(\underline{U}) = BY_n G_n^2 U_n^2 -$$
$$\left(\frac{N^2}{(1-\varepsilon)^2}Y_n G_n + N\frac{(\delta G_n - Y_n G_n^1)}{2(1-\varepsilon)}\right)U_{n+1}$$
$$+\left(2\frac{N^2}{(1-\varepsilon)^2} + A\right)Y_n G_n U_n$$
$$-\left(\frac{N^2}{(1-\varepsilon)^2}Y_n G_n - N\frac{(\delta G_n - Y_n G_n^1)}{2(1-\varepsilon)}\right)U_{n-1}$$
$$-\Lambda Y_n G_n^2 .$$

```
Step6: End of n loop
Step7: Return Functional values.
Step8: Stop.
```

```
SUBROUTINE   Differential matrix (set of parameters received)
Step1:   Start.
Step2:   Accept A, B, N, v, δ.
Step3:   For  i =0 DO   N
Step4:   Calculate
$esp1[n] = pow((1-\varepsilon), 2)$
$Y_n = \varepsilon + n\frac{(1-\varepsilon)}{N}$
$G_n = e^{\frac{\log(1+v)}{1-\varepsilon}(y_n-\varepsilon)}$
$G_n^2 = e^{\frac{\log(1+v)}{1-\varepsilon}(y_n-\varepsilon)}$
$G_n^1 = \frac{\log(1+v)}{1-\varepsilon}e^{\frac{\log(1+v)}{1-\varepsilon}(y_n-\varepsilon)}$
         where
$\varepsilon = R_2 R_1^{-1}$ (ratio of inner to outer cylinder radii).
v     variable viscosity.
Step5:   Calculate
$f_n(\underline{U}) = 2*BY_nG_n^2U_n + \left(\frac{N^2}{(1-\varepsilon)^2}Y_nG_n + N\frac{(\delta G_n - Y_nG_n^1)}{2(1-\varepsilon)}\right)U_{n+1}$
$+\left(2\frac{N^2}{(1-\varepsilon)^2}+A\right)Y_nG_nU_n - \left(\frac{N^2}{(1-\varepsilon)^2}Y_nG_n - N\frac{(\delta G_n - Y_nG_n^1)}{2(1-\varepsilon)}\right)U_{n-1} - \Lambda Y_nG_n^2.$
Step6:  End  of  I^th  loop.
Step7:   Return Functional values.
Step8:   Stop.
```

```
SUBROUTINE Inverse_ matrix (set of parameters received).
Step1:   Start.
Step2:   Accept  A, B, Delta, Epsilon,N.
Step3: Find the inverse of Differential Matrix using Matrix determination method applying Gauss Jordan Method
Step4: Read the order of the matrix 'n' and read the coefficients of the linear    equations.
Step5:  For k=1 to n Do
           For  l=k+1 to n+1 Do
           a[k][l] = a[k][l] / a[k][k]
           End for l Loop
           Set a[k][k] = 1
Step6: For i=1 to n Do
                    if (i not equal to k) then,
Step7: For j=k+1 to n+1 Do
                    a[i][j] = a[i][j] – (a[k][j] * a[i][k])
           End for j Loop
           End for i Loop
           End for k Loop
Step8:  For m=1 to n Do
                    x[m] = a[m][n+1]
                    Display x[m]
           End for m Loop.

Step9  : Return Functional values x[m].
Step10: Stop.
```

```
SUBROUTINE Multiplication_matrix(Inverse_Matrix, Constant_Matrix).
Step1:   Start.
Step2:   Accept  N,h.
Step3: Calculate
Step4: For  I = 0  DO   N
Step5: For  J = 0  DO   N
Step6: Multi_Matrix = h*(Inverse_Matrix* Constant_Matrix)
Step7: End  of  J  Loop.
Step8: End  of  I  Loop.
Step9: Return Functional values.
Step10: Stop.
```

3D SIMULATION OF HCM RESULTS

3D simulation software is used to present Darcy-Brikman-Forchheimer flow through annulus. Simulation gives visual sequence of different set of parameter in pours media. 3D animation also called as **Computer Graphics** or **CG**. CG refers to any picture or series of pictures that are generated with the aid of a computer. The process of creating in 3D requires that we need to model or shape objects in a scene, give them color and light, animate them as required, and render them through a virtual camera to make an image.

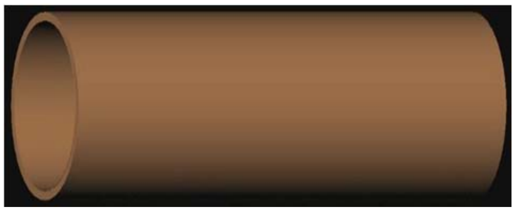
Porous media modelling.

Maya polygon is used to create 3 dimensional porous media.

Step1:

Create cylinder

1. Create a **NURBS cylinder** and name it "*c_i*", for "inferior constraint". Go to the **Channel Box** and set its Y scale to **10**, and its Z scale to **0.4**.
2. Enter "**Insert**" to edit the Pivot Point position, and then go to the **Numerical Input Line**, which is a white space just above the Channel Box. Enter the values **0 -10 0.**



Step 2

Create cylinder with porous media.

1. Create Polygonal sphere and name it "porous Particle"
2. Duplicate porous particles to fill inside the cylinder.

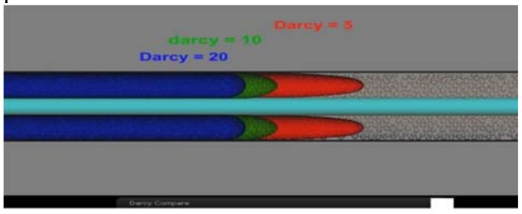
Step 3

Create annulus with porous media

1. Fill the porous particles centered with iron rod inside the cylinder created in step 1

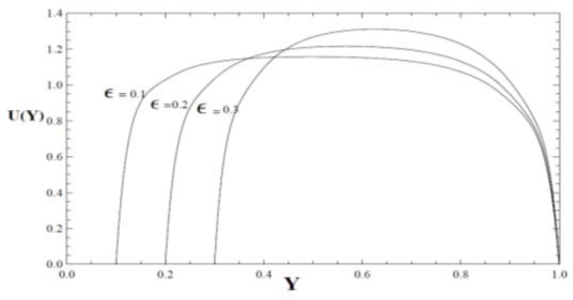
Animation

Create expression to animate velocity profile for different set of parameter.



Parameters	$\Lambda = 1,$ $F = 0,$ $Da = 0$	$\Lambda = 1,$ $F = 0,$ $Da = 0.5$	$\Lambda = 1,$ $F = 2,$ $Da = 0.5$	$\Lambda = 1,$ $F = 0,$ $Da = 100$
Nu present paper	4.33108	4.39733	4.33804	598708
Hooman & Gurgenci[11]	4.36	---	---	---

Table 1: Number of equations required for convergence, for different parameters' combination, for an accuracy of 10^{-4} (applies to both channel and tube flows



Different values of Epsilon with same set of parameters

Parameters	Λ $= F$ $= Da$ $= 1$	$\Lambda = 0.1,$ $F = 1,$ $Da = 1$	$\Lambda = 0.1,$ $F = 1,$ $Da = 31.6622$	$\Lambda =$ $F = 1,$ $Da = 100$
Nu present paper	4.146	4.115	5.12886	5.8479
Nield[9]	4.159	4.122	5.129	—
Hooman[10]	4.181	4.131	5.139	5.8935

Table 2: Rectangular Problem-Comparison between present results on Nusselt number with those of Hooman [10] and Nield[9].

Λ	Re	Da	No.of equations required for convergence
1.2	10	05	20
1.2	10	10	12
1.2	10	20	08
0.9	10	10	13
1.0	10	10	12
1.1	10	10	11
1.2	05	10	12
1.2	10	10	11
1.2	20	10	11

Table 3: Cylindrical Problem-Comparison between present results on Nusselt number with those of Hooman and Gurgenci[16].

RESULS AND CONCLUSIONS

We are designed single module HCM algorithm to solve all four problems, ie, DBF flow through rectangular porous channel problem with constant viscosity DBF flow through a cylindrical porous tube, DBF flow through a cylindrical porous annulus and DBF flow through a rectangular porous channel with variable viscosity, with all different set of parameters in the above there problems.

This algorithm works for all sets of parameters, it succeeds in giving the required solution for large values of Forchheimer number when shooting method fails to do so.

REFERENCES

[1] G. J. Chaitin. On the length of programs for computing finite binary equences.Journal of the ACM, 13(4):547--569, 1966.

[2] P. Gacs. On the symmetry of algorithmic information. Soviet Mathematics Doklady, 15:1477--1480, 1974.

[3] M. Hutter. On Universal Prediction and Bayesian Confirmation. Theoretical Computer Science, 384:1 (2007) 33-48

[4] A. N. Kolmogorov. Three approaches to the quantitative definition of information. Problems of Information and Transmission, 1(1):1--7, 1965.

[5] A. N. Kolmogorov. Combinatorial foundations of information theory and the calculusof probabilities. Russian Mathematical Surveys, 38(4):27--36, 1983.

[6] L. A. Levin. Laws of information conservation (non-growth) and aspects of the foundation of probability theory. Problems of Information Transmission, 10(3):206--210, 1974.

[7] M. Li and P. M. B. Vitanyi. An Introduction to Kolmogorov Complexity and its Applications. Springer, New York, 2nd edition, 1997.

[8] A. K. Zvonkin and L. A. Levin. The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms.Russian Mathematical Surveys, 25(6):83--124, 1970.

[9] D. A. Nield and A. Bejan, Convection in porous media, Springer Verlag, New York, 2006.

[10] K. Hooman, A perturbation solution for forced convection in a porous saturated duct, J.Comput. Appl. Math. 211(1) (2008) 57-66.

[11] K. Hooman and H. Gurgenci, A theoretical analysis of forced convection in a porous saturated circular tube: Brinkman-Forchheimer model, Transport Porous Media, 69(3) (2007) 289-300.