Numerical solution to the depth-confined 3D lid-driven cavity flow problem in the Brinkman limit

Sungwon La
(Brown University)

Abstract

Thin fluid flows that are highly confined in the depth direction are present in many different academic and industrial applications, such as electronic chips or thin biomedical implants that are very thin in one direction and carry fluid flow. The Navier-Stokes equations governing the flow of fluids can be greatly simplified for these types of thin flows, which are referred to flows in the Brinkman limit. Depth-averaging the flow and transforming into streamfunction-vorticity formulation yields the following two simplified equations:

$$\frac{\partial \bar{\omega}_z}{\partial t} = -\frac{6}{5}\ell_z^2\left(\frac{\partial \psi}{\partial y}\frac{\partial \bar{\omega}_z}{\partial x} - \frac{\partial \psi}{\partial x}\frac{\partial \bar{\omega}_z}{\partial y}\right) + \nu\left(\frac{\partial^2 \bar{\omega}_z}{\partial x^2} + \frac{\partial^2 \bar{\omega}_z}{\partial y^2}\right) - \frac{12\nu}{\ell_z^2}\bar{\omega}_z$$

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = -\frac{\bar{\omega}_z}{\ell_z^2}$$

These two equations can be solved numerically to yield solutions for $\bar{\omega}$ and ψ, from which the depth-averaged velocity profiles of $\bar{u}$ and $\bar{v}$ can be found. This project numerically solved the steady-state lid-driven cavity-flow problem using these equations, in which one side of a very thin box is moving at a constant velocity, creating a circulating flow inside the box. Equation 1 was used to solve for ω using discretizations of the first derivatives using the second-order central difference scheme, while equation 2 was then used to solve for ψ using the Jacobi iteration scheme. The numerical simulations matched the intuitive lid-driven cavity flow results commonly seen in other problems with similar setups, with the exception of regions near the boundaries. The derived equations are thus likely to able to provide accurate depth-averaged velocities for the entire class of flow problems in the Brinkman limit in the non-boundary regions, while a different set of governing equations must be derived to match the physics of the boundary regions.

1 INTRODUCTION

The Navier-Stokes equations, which are the governing equations for fluid flow, are generally very difficult to solve due to the nonlinearity of the equations and the complex geometries to which they are generally applied to [1]. However, in a special class of fluid flows called the Brinkman Limit, the equations can be greatly simplified such that relatively cost-effective numerical methods can be used to solve for the flow profile. The Brinkman limit refers to a case when the flow is highly confined in the depth direction, such that it is quasi-two-dimensional when considering the depth-averaged flows. This is the same as lubrication theory, but the velocities are depth-averaged such that the 3D flow problem is characterized by a simpler 2D problem [2].

These types of thin narrow-channel flows have numerous applications, including microfluidic devices that have very shallow channel depths such as computer chips and biomedical tools. One particular area of importance highlighted by the recent COVID-19 pandemic is rapid PCR powered by microfluidics, which is conducted with a lab-on-a-chip device. Convection and thermal cyclings power the microfluidics inside the device, and the flows are highly confined in the depth direction to the scale of a few micrometers, as they are in the geometry of a chip [3]. This is an example of a flow in the the Brinkman limit, and thus far, only full 3D simulations have been conducted for the analysis of fluid flows inside these devices. Common methods used include 3D coupled Lagrangian modeling, which though it may yield accurate results, can be computationally costly and time-consuming, and may be more difficult for complex geometries [4]. The depth-averaged equations can describe the motion of fluids in these types of microfluidic devices in a more computationally effective manner. These equations are very general in that they are applicable to an entire class of flow problems where the depth scale is thin compared to the other two length

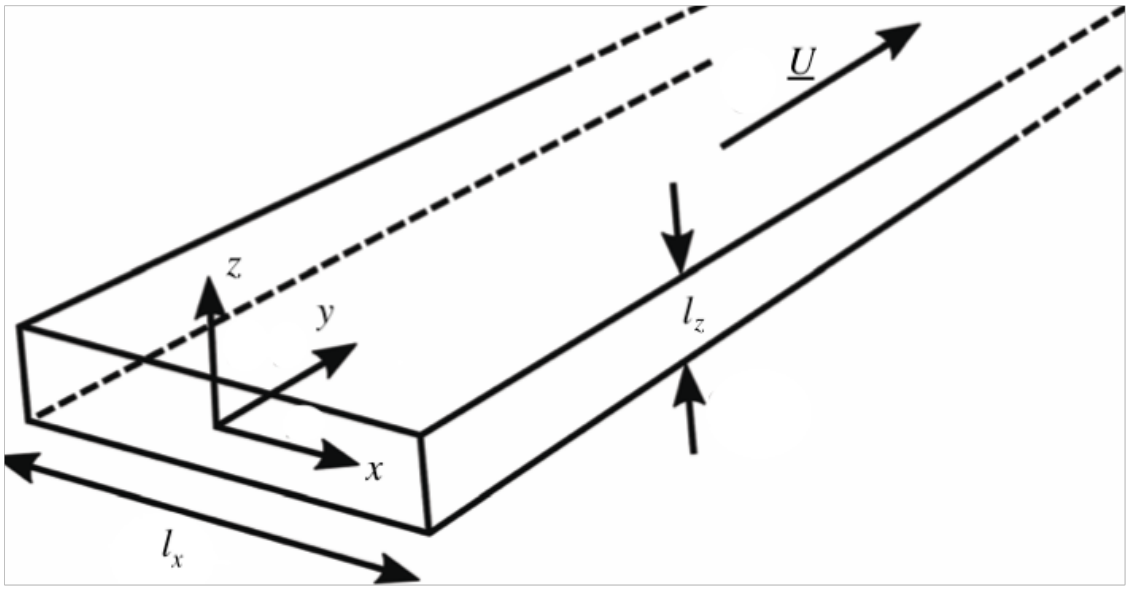

Figure 1: Diagram of an example of a flow in the Brinkman limit, where the depth length scale across the z direction is much smaller than the length scales in the x and y directions

scales, which is virtually present in almost all microfluidic devices, and utilization of these equations can make complicated and expensive 3D flow problems characterized by much simpler and cost-effective 2D depth-averaged problems.

The particular problem attempted in this project using these derived depth-averaged equations for flow in the Brinkman limit was the lid-driven cavity-flow problem, in which one side of a very thin box is moving at a constant velocity, creating a circulating flow inside the box. This problem is a classic benchmarking problem in fluid dynamics to validate techniques, methods, and theory [5]. Typically, this problem is solved for the actual velocities at every point, but for the purposes of this project and utilization of the derived governing equations in the Brinkman limit, the depth-averaged velocities will be calculated.

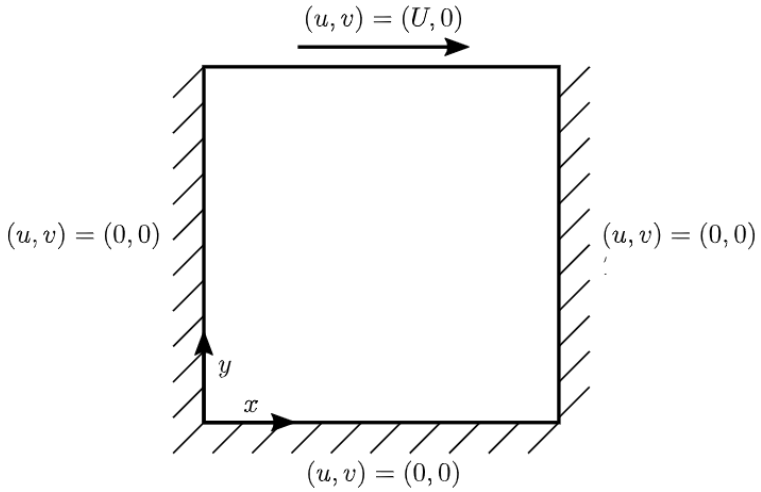

Figure 2: Diagram of the cavity-flow problem setup, with the top wall moving at a velocity U, and the other three walls stationary. The walls of the cavity extend in and out of the page in the z direction, but the depth is very small compared to the lengths of the cavity in the x and y directions.

2 PHYSICAL & NUMERICAL MODELS

The main assumption made when simplifying the Navier-Stokes equations in thin channel flow is $\frac{\ell_z}{\ell_x} << 1$ and $\frac{\ell_z}{\ell_y} << 1$, where z is the thin depth direction and x and y are the other non-thin depth directions, and ℓ_x, ℓ_y, ℓ_z are the length scales of the problem in the x, y, and z directions, respectively. For the cavity-flow problem, ℓ_x and ℓ_y would be the sides of the long square edges, and ℓ_z would be the depth in the z direction. In addition, the flow is assumed to be steady-state, so there is no time-dependence. Using these assumptions to simplify the Navier-Stokes equations, depth-averaging the velocities across the thin channel scale z, and reformulating into streamfunction-vorticity formulation, the following equations with ψ representing streamfunction of the flow and $\bar{\omega}$ representing depth-averaged vorticity of the flow are given:

$$\frac{\partial \bar{\omega}_z}{\partial t} = -\frac{6}{5}\ell_z^2\left(\frac{\partial \psi}{\partial y}\frac{\partial \bar{\omega}_z}{\partial x} - \frac{\partial \psi}{\partial x}\frac{\partial \bar{\omega}_z}{\partial y}\right) + \nu\left(\frac{\partial^2 \bar{\omega}_z}{\partial x^2} + \frac{\partial^2 \bar{\omega}_z}{\partial y^2}\right) - \frac{12\nu}{\ell_z^2}\bar{\omega}_z \tag{1}$$

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = -\frac{\bar{\omega}_z}{\ell_z^2} \tag{2}$$

2.1 NUMERICAL METHODS

The cavity in the problem was set to have side lengths of 1. The grid sizes are uniform such that $\Delta x = \Delta y = h$. The first and second derivatives in equation 1 and equation 2 were discretized using second-order central difference schemes, given by:

$$\frac{\partial P}{\partial x} = \frac{P_{i+1,j} - P_{i-1,j}}{2h}$$

$$\frac{\partial P}{\partial y} = \frac{P_{i,j+1} - P_{i,j-1}}{2h}$$

$$\frac{\partial^2 P}{\partial x^2} = \frac{P_{i+1,j} - 2P_{i,j} + P_{i-1,j}}{h^2}$$

$$\frac{\partial^2 P}{\partial y^2} = \frac{P_{i,j+1} - 2P_{i,j} + P_{i,j-1}}{h^2}$$

Semi-discretization of equation 1 (with ω now representing depth-averaged vorticity $\bar{\omega}_z$ for simplicity) leads to:

$$\frac{\partial \omega}{\partial t} = -\frac{6}{5}\ell_z^2\left(\left(\frac{\psi_{i,j+1} - \psi_{i,j-1}}{2h}\right)\left(\frac{\omega_{i+1,j} - \omega_{i-1,j}}{2h}\right) - \left(\frac{\psi_{i+1,j} - \psi_{i-1,j}}{2h}\right)\left(\frac{\omega_{i,j+1} - \omega_{i,j-1}}{2h}\right)\right)$$

$$+\nu\left(\frac{\omega_{i+1,j} - 2\omega_{i,j} + \omega_{i-1,j}}{h^2} + \frac{\omega_{i,j+1} - 2\omega_{i,j} + \omega_{i,j-1}}{h^2}\right) - \frac{12\nu}{\ell_z^2}\omega_{i,j}$$

First, to solve forward in time, the forward-Euler time-stepping method was used with $\frac{\partial \omega}{\partial t}$ from equation 1 above. Given that the transient solution did not matter and the final steady-state solution was what was really of interest, the initial condition was simply set to $\psi = 0$ everywhere, and $\omega = 0$ everywhere except for at the boundaries (boundary conditions are stated in section 2.2).

$$\omega_{i,j}^{n+1} = \omega_{i,j}^{n} + \Delta t\frac{\partial \omega}{\partial t}$$

Then, equation 2 was used to solve for ψ^{n+1} using the Jacobi iteration method with the new values of ω^{n+1}. Discretization and further rearrangement of equation 2 leads to:

$$\frac{\psi_{i+1,j} - 2\psi_{i,j} + \psi_{i-1,j}}{h^2} + \frac{\psi_{i,j+1} - 2\psi_{i,j} + \psi_{i,j-1}}{h^2} = -\frac{\omega_{i,j}^{n+1}}{\ell_z^2}$$

$$\psi_{i,j} = \frac{1}{4}\left(\psi_{i+1,j} + \psi_{i-1,j} + \psi_{i,j+1} + \psi_{i,j-1} + \omega_{i,j}^{n+1}\frac{h^2}{\ell_z^2}\right)$$

The values for ψ were updated repeatedly using the equation above with ω^{n+1} and the most recently updated values of ψ until the largest difference at any node between the previous iteration for ψ and the most recent value for ψ fell below a certain user-determined tolerance level. This converged value of ψ was taken to be ψ^{n+1}.

These updated values of ψ^{n+1} and ω^{n+1} at time step $n+1$ were used again in equation 1 to solve for new values of ω forward in time, which were then used in equation 2 to solve for new values of ψ again. This process was repeated until the values of ψ and ω stopped changing with iterations in time, signifying that they had converged to a steady-state solution. The final steady-state values of vorticity ω and streamfunction ψ were used to extract the steady-state depth-averaged velocity profiles for $\bar{u}$ and $\bar{v}$, with $\bar{u} = \frac{\partial \psi}{\partial y}$ and $\bar{v} = -\frac{\partial \psi}{\partial x}$. Discretizing the equations for $\bar{u}$ and $\bar{v}$ using the central difference scheme for first derivatives leads to:

$$u = \frac{\psi_{i,j+1} - \psi_{i,j-1}}{2h}$$

$$v = -\frac{\psi_{i+1,j} - \psi_{i-1,j}}{2h}$$

2.2 BOUNDARY CONDITIONS

The outer walls of the cavity can all be treated as a closed streamfunction, which means that all around the edge, there will be a constant value for streamfunction. The actual value of streamfunction that will be set does not matter, since in the equations above, only the derivatives of streamfunction are included. For simplicity, the value of streamfunction all around the edges were set to 0. Thus, the boundary conditions for these equations are that the streamlines $\psi = 0$ all along the walls.

To derive boundary conditions for vorticity ω, Taylor series expansions can be performed for the inner streamfunction node just inside the boundary boundary streamfunction node. Taking the top wall for example, expanding $\psi_{i,N-1}$ about $\psi_{i,N}$,

$$\psi_{i,N-1} = \psi_{i,N} - \frac{\partial \psi}{\partial y}_{i,N} h + \frac{\partial^2 \psi}{\partial y^2}_{i,N} \frac{h^2}{2}$$

Some terms in the equation can be replaced with known values. For example, $\psi_{i,N} = 0$ is known from the streamfunction boundary condition, and $\frac{\partial \psi}{\partial y}_{i,N} = U_{top}$ is known from the definition of $u = \frac{\partial \psi}{\partial y}$. Additionally, from the governing Poisson equation $\frac{\partial^2 \psi}{\partial x^2}_{i,N} + \frac{\partial^2 \psi}{\partial y^2}_{i,N} = \frac{\omega_{i,N}}{\ell_z^2}$, since $\frac{\partial^2 \psi}{\partial x^2}_{i,N} = 0$ due to streamfunction being constant in the x direction along the top wall, it can be found that $\frac{\partial^2 \psi}{\partial y^2}_{i,N} = \frac{\omega_{i,N}}{\ell_z^2}$.

Substituting these known values into the Taylor series expansion, we arrive at the boundary condition for vorticity ω at the top wall:

$$\omega_{i,N} = 2\ell_z^2 \left(\frac{\psi_{i,N-1}}{h^2} + \frac{U_{top}}{h} \right)$$

The boundary conditions for the other walls can be derived in a similar manner, and are stated below:

$$\omega_{i,1} = 2\ell_z^2 \left(\frac{\psi_{i,2}}{h^2} - \frac{U_{bottom}}{h} \right)$$

$$\omega_{N,j} = 2\ell_z^2 \left(\frac{\psi_{N-1,j}}{h^2} + \frac{U_{right}}{h} \right)$$

$$\omega_{1,j} = 2\ell_z^2 \left(\frac{\psi_{2,j}}{h^2} - \frac{U_{left}}{h} \right)$$

The boundary conditions for ψ are constant and do not change with iterations, but the boundary conditions for ω need to be updated with each iteration.

2.3 ACCURACY

The order of accuracy for the numerical method used depends on the finite difference scheme used for space, as well as the time-stepping method used for time. The first and second spatial derivatives present in the equations were all discretized using the first and second derivative central difference schemes, which are both second-order accurate. Thus, the numerical method is second-order accurate in space. For time, the first-order forward-Euler time stepping scheme was used. Thus, the numerical method is first-order accurate in space.

2.4 STABILITY ANALYSIS

To determine the stability of the time-stepping scheme, the modified wavenumber analysis can be performed on equation 1. In order to simplify the analysis, only the term that impacts stability the most will be looked at.

$$First\ Term: -\frac{6}{5}\ell_z^2\left(\left(\frac{\psi_{i,j+1}-\psi_{i,j-1}}{2h}\right)\left(\frac{\omega_{i+1,j}-\omega_{i-1,j}}{2h}\right)-\left(\frac{\psi_{i+1,j}-\psi_{i-1,j}}{2h}\right)\left(\frac{\omega_{i,j+1}-\omega_{i,j-1}}{2h}\right)\right)$$

The first term in equation 1 represents a convection term with the first derivatives of ω. However, there is a coefficient of ℓ_z^2 attached to this term, and with the problem setup, it is known that the length scale in the depth direction, ℓ_z, is very small. The entire first term will thus be very small, so the first term is not expected to be the primary actor for determining stability.

$$Second\ Term: \nu\left(\frac{\omega_{i+1,j}-2\omega_{i,j}+\omega_{i-1,j}}{h^2}+\frac{\omega_{i,j+1}-2\omega_{i,j}+\omega_{i,j-1}}{h^2}\right)$$

$$Third\ Term: -\frac{12\nu}{\ell_z^2}\omega$$

The second term in equation 1 represents a diffusion term, while the third term in equation 1 represents a source term. Diffusion for forward Euler is known to have a more restrictive stability criterion; thus, modified wavenumber analysis will be performed on the following reduced equation to determine stability for the numerical scheme:

$$\frac{\partial\omega}{\partial t}=\nu\left(\frac{\omega_{i+1,j}-2\omega_{i,j}+\omega_{i-1,j}}{h^2}+\frac{\omega_{i,j+1}-2\omega_{i,j}+\omega_{i,j-1}}{h^2}\right)$$

Note that the discretization is in 2D, so the substitution for ω must include both the x and y directions. The following substitution can be made:

$$\omega=\phi(t)e^{ikx_j}e^{iky_j}$$

$$e^{ikx_j}e^{iky_j}\frac{\partial\phi}{\partial t}=\frac{\nu}{h^2}\left(2\phi e^{ikx_j}e^{iky_j}e^{-ikh}-4\phi e^{ikx_j}e^{iky_j}+2\phi e^{ikx_j}e^{iky_j}e^{ikh}\right)$$

Dividing both sides by $e^{ikx_j}e^{iky_j}$,

$$\frac{\partial\phi}{\partial t}=\frac{2\nu}{h^2}\left(e^{-ikh}-2+e^{ikh}\right)\phi$$

The trigonometric substitution $e^{-ikh}+e^{ikh}=2cos(kh)$ can be made.

$$\frac{\partial\phi}{\partial t}=\frac{2\nu}{h^2}\left(2cos(kh)-2\right)\phi$$

Now, this equation is reduced to resemble the model problem for the forward-Euler time stepping scheme, $y'=\lambda y$. In this case, for $\lambda=\frac{2\nu}{h^2}\left(2cos(kh)-2\right)$, the "worst case scenario" that makes λ have the highest magnitude happens when $cos(kh)=-1$. Thus, the "worst case" highest magnitude λ is given by:

$$\lambda=-\frac{8\nu}{h^2}$$

When the forward-Euler time-stepping method is applied to the model problem $y'=\lambda y$, the time step is limited by $\Delta t\leq\frac{2}{|\lambda_{real}|}$. Thus, the stability criterion for equation 1 is given by:

$$\Delta t\leq\frac{h^2}{4\nu}$$

3 RESULTS & DISCUSSION

3.1 PARAMETERS FOR NUMERICAL SIMULATION

The experimental parameters used in the numerical simulation, performed using MatLAB, were:

Wall Length ℓ_x, ℓ_y	$1m$
Depth ℓ_z	$0.01m$
Top Wall Velocity U_{top}	$1m/s$
Kinematic Viscosity ν	$1000m^2/s$

The numerical simulation parameters used were:

Number of Nodes in x and y	200
Grid Spacing h	$0.005m$
Time Spacing Δt	$\frac{h^2}{10\nu} = 2.5 * 10^9$
Tolerance for Jacobi Iteration	10^{-8}
Tolerance for Velocity Change	10^{-20}

Tolerance for Jacobi iteration is the convergence criteria for the maximum difference between previous iteration value of ψ and the most recent value of ψ, and determines when ψ represents the value for ψ^{n+1}. Tolerance for velocity change is the convergence criteria for the maximum difference between value of u or v at previous time step and the most recent value of u or v, and determines when the solution is approximately steady-state.

3.2 VELOCITY PLOTS

The velocity profiles extracted from the numerical simulation are shown in figures 3 and 4. For the cavity-flow problem with the top wall moving to the right, it is expected intuitively that the shearing due to the wall will create a circular flow inside the cavity, with the greatest velocity being near the wall. The results match the intuition, as the x-component of velocity, u, increases in magnitude as y gets closer and closer to y = 1 (the moving wall). In addition, a circular flow would induce a positive y-component of velocity on one side of the vertical walls, and a negative y-component of velocity on the other wall. This is what is seen in the graphs of the y-component of velocity. The values for v are also shown to be increasing in magnitude as y gets closer and closer to y = 1 (the moving wall), which also lines up with the intuitive result. Thus, the governing equations appear to provide a physically rational flow result.

3.3 TIME STEP GREATER THAN STABILITY CRITERION

The stability criterion for this numerical method was found to be:

$$\Delta t \leq \frac{h^2}{4\nu} = 6.25 * 10^9$$

The numerical simulation was carried out with a time step value smaller than the criterion, $\Delta t = \frac{h^2}{10\nu} = 2.5 * 10^9$. To test the stability criterion, the same simulation was carried out with a time step slightly greater than the criterion, of $\Delta t = 6.5 * 10^9$. As expected, the solution diverged and yielded unstable results, as shown in figure 5. Thus, the modified wavenumber stability analysis was correct in its yielding of the maximum possible time step.

3.4 BOUNDARY REGION

One area where the simulation appeared to be inaccurately representing the physics of the problem was near the moving boundary. There is a very unsmooth transition in velocity in the results, with a very high velocity gradient in the span of just a few grid points. This indicates that though the derived governing equations for the flow can accurately represent inner areas away from the boundary, the physics is different for areas near the boundary, requiring a different set of governing equations. The values for u near the moving wall, for example, are shown in figure 6, and are shown to have an unreasonably large velocity gradient.

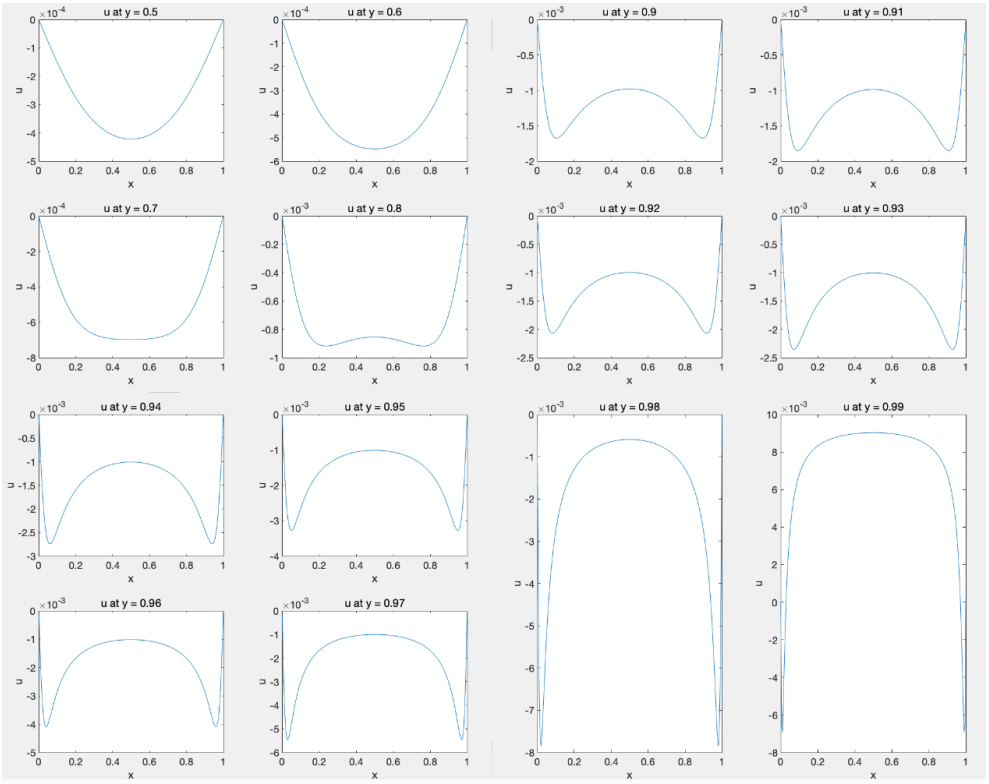

Figure 3: Graph of the x component of velocity (u) vs. x, plotted for different values of y (y = 1 represents the moving top wall)

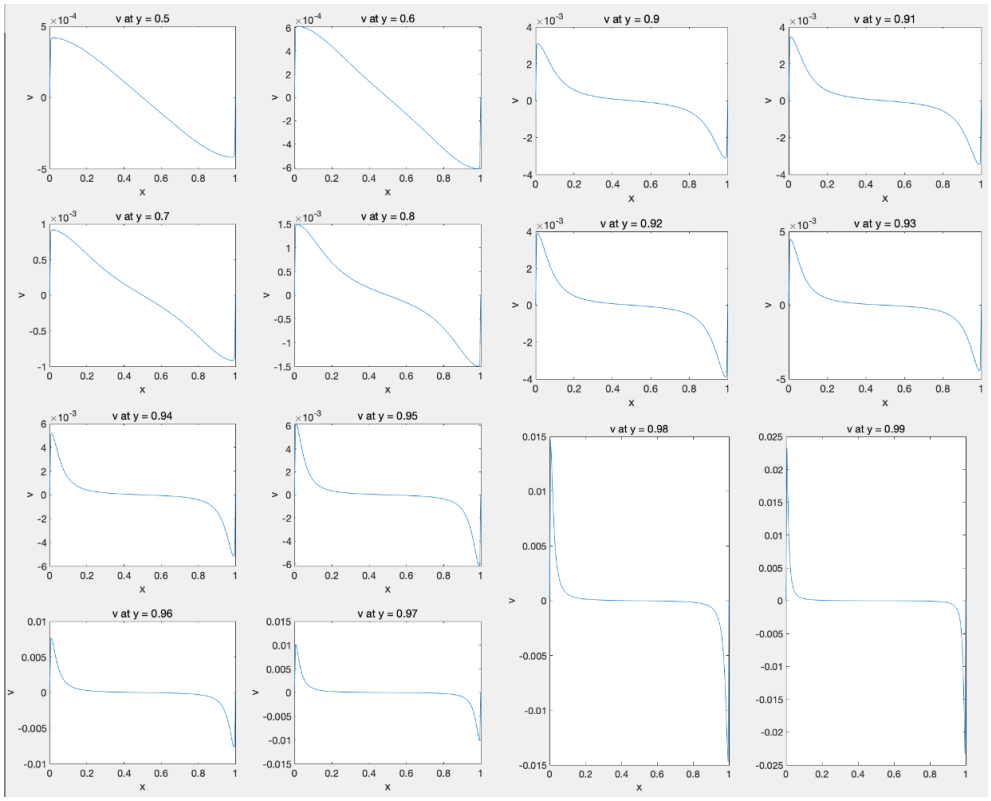

Figure 4: Graph of the y component of velocity (v) vs. x, plotted for different values of y (y = 1 represents the moving top wall)

```
t = 5.252393e-08, num_iters = 4541, error = 9.998236e-09, du_dt = 5.973272e-03, dv_dt = 4.239209e-03
t = 5.908942e-08, num_iters = 6032, error = 9.998991e-09
t = 5.908942e-08, num_iters = 6032, error = 9.998991e-09, du_dt = 8.219595e-03, dv_dt = 7.097606e-03
t = 6.565491e-08, num_iters = 7965, error = 9.999930e-09
t = 6.565491e-08, num_iters = 7965, error = 9.999930e-09, du_dt = 1.167474e-02, dv_dt = 1.155729e-02
t = 7.222040e-08, num_iters = 9413, error = 9.999669e-09
t = 7.222040e-08, num_iters = 9413, error = 9.999669e-09, du_dt = 1.856093e-02, dv_dt = 1.856093e-02
t = 7.878589e-08, num_iters = 11258, error = 9.997926e-09
t = 7.878589e-08, num_iters = 11258, error = 9.997926e-09, du_dt = 2.969271e-02, dv_dt = 2.969271e-02
t = 8.535138e-08, num_iters = 13250, error = 9.998792e-09
t = 8.535138e-08, num_iters = 13250, error = 9.998792e-09, du_dt = 4.754259e-02, dv_dt = 4.754259e-02
t = 9.191687e-08, num_iters = 15162, error = 9.998536e-09
t = 9.191687e-08, num_iters = 15162, error = 9.998536e-09, du_dt = 7.636946e-02, dv_dt = 7.636946e-02
t = 9.848236e-08, num_iters = 17114, error = 9.997660e-09
t = 9.848236e-08, num_iters = 17114, error = 9.997660e-09, du_dt = 1.232107e-01, dv_dt = 1.232107e-01
t = 1.050479e-07, num_iters = 19090, error = 9.996947e-09
t = 1.050479e-07, num_iters = 19090, error = 9.996947e-09, du_dt = 1.997454e-01, dv_dt = 1.997454e-01
t = 1.116133e-07, num_iters = 21079, error = 9.997551e-09
t = 1.116133e-07, num_iters = 21079, error = 9.997551e-09, du_dt = 3.254333e-01, dv_dt = 3.254333e-01
t = 1.181788e-07, num_iters = 23070, error = 9.998704e-09
t = 1.181788e-07, num_iters = 23070, error = 9.998704e-09, du_dt = 5.328200e-01, dv_dt = 5.328200e-01
t = 1.247443e-07, num_iters = 25075, error = 9.997414e-09
t = 1.247443e-07, num_iters = 25075, error = 9.997414e-09, du_dt = 8.765272e-01, dv_dt = 8.765272e-01
```

Figure 5: Number of iterations required for each time step to converge the Jacobi iterations, which is shown to be diverging when time step is set to greater than the stability criterion

0	-0.0018	-0.0033	-0.0044	-0.0051	-0.0054	-0.0054	-0.0054	-0.0052	-0.0050	-0.0048
0	-0.0025	-0.0046	-0.0058	-0.0064	-0.0065	-0.0064	-0.0061	-0.0058	-0.0055	-0.0052
0	-0.0039	-0.0064	-0.0076	-0.0078	-0.0076	-0.0071	-0.0066	-0.0061	-0.0056	-0.0052
0	-0.0059	-0.0086	-0.0090	-0.0084	-0.0076	-0.0067	-0.0058	-0.0051	-0.0045	-0.0039
0	-0.0065	-0.0069	-0.0051	-0.0030	-0.0012	2.7288e-...	0.0014	0.0024	0.0032	0.0038
0	0.0134	0.0232	0.0295	0.0336	0.0363	0.0382	0.0396	0.0407	0.0416	0.0422
1	1	1	1	1	1	1	1	1	1	1

Figure 6: Values of the x-component of velocity (u) near the moving wall at y = 1, shown to have a very unsmooth transition

4 CONCLUSIONS

The theoretically derived depth-averaged governing equations for fluid flows in the Brinkman limit, where the flow is highly confined in the depth direction, was simulated numerically using the forward-Euler time-stepping method and the Jacobi iteration method. The central-difference spatial discretization formulas yielded a stability criterion that when tested, was proven to be accurate. The results yielded velocity profiles that made intuitive sense and were similar to results of other simulated cavity-flow problems - thus, the governing equations are likely to be able to yield accurate results for depth-averaged velocities for thin flows highly confined in the depth direction. This method of finding the depth-averaged flows is much more computationally efficient than performing a full 3D simulation of the flow, and can be utilized to solve for depth-average pressure and drag, helping in designing thin-channel engineering devices such as biomedical devices and computer chips. However, the equations were found to have limitations when describing the flow behavior near the boundary regions. These regions presumably require a different set of governing equations to accurately represent the physics altered due to the presence of the boundaries.

5 BIBLIOGRAPHY

[1] J. Venetis, “On a modified form of navier-stokes equations for three-dimensional flows,” The Scientific World Journal, vol. 2015, pp. 1–9, Mar. 2015. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4396908/B8

[2] B. Tavakol, G. Froehlicher, D. P. Holmes, and H. A. Stone, “Extended lubrication theory: improved estimates of flow in channels with variable geometry,” Proc. R. Soc, vol. 473, no. 2206. Oct. 2017. https://doi.org/10.1098/rspa.2017.0234

[3] Dong, Xiaobin, et al. “Rapid PCR Powered by Microfluidics: A Quick Review under the Background of Covid-19 Pandemic,” TrAC Trends in Analytical Chemistry, vol. 143, June. 2021. https://doi.org/10.1016/j.trac.2021.116377.

[4] Min-Cheol Kim, Zhanhui Wang, Raymond H. W. Lam, and Todd Thorsen , ”Building a better cell trap: Applying Lagrangian modeling to the design of microfluidic devices for cell biology,” Journal of Applied Physics, vol. 103, Issue 4, 044701. Feb. 2008. https://doi.org/10.1063/1.2840059

[5] Tamer A. AbdelMigid, Khalid M. Saqr, Mohamed A. Kotb, Ahmed A. Aboelfarag, ”Revisiting the lid-driven cavity flow problem: Review and new steady state benchmarking results using GPU accelerated code,”Alexandria Engineering Journal, vol.56, Issue 1, pp. 123-135, Mar. 2017. https://doi.org/10.1016/j.aej.2016.09.013.

```
tic;
clc;
clear;
close all;

%In order to change into the original cavity-flow problem, set
%wallLengthZ = 2.45 and change the formula for the non-boundary omega points

%experimental parameters
wallLengthX = 1;
wallLengthY = 1;
wallLengthZ = 0.01; %set to 2.45 in order to make equal to original cavity-
flow problem
%wall movement
velocityTopWall = 1;
velocityBottomWall = 0;
velocityLeftWall = 0;
velocityRightWall = 0;
nu = 10e3; %change to alter value of Re

%numerical approximation parameters
nodesX = 200; %number of nodes in x direction
nodesY = 200; %number of nodes in y direction
x = linspace(0,wallLengthX,nodesX); %x axis
y = linspace(0,wallLengthY,nodesY); %y axis
dx = x(2) - x(1); %distance step along x
dy = y(2) - y(1); %distance step along y
dt = 0.1*min(dx,dy)^2/nu; %time step
t_final = 1e6; %simulation runtime
tolerance = 1e-8; %for Jacobi iteration
vel_tol = 1e-10; %for determining when steady-state is achieved

%initial conditions
omega = zeros(nodesY,nodesX);
domegadt = zeros(nodesY,nodesX);
psi = zeros(nodesY,nodesX); %automatically sets boundary conditions for psi
 values to be 0
u = zeros(nodesY,nodesX);
v = zeros(nodesY,nodesX);

skip = 1000;
outer_counter = 1;
%updating mesh points
for t = 0:dt:t_final
    u_old = u;
    v_old = v;

    %non-boundary points for omega, time step
    domegadt(2:nodesY-1,2:nodesX-1) =
 ((-6*(wallLengthZ^2)/5)*((((psi(3:nodesY,2:nodesX-1)-...
        psi(1:nodesY-2,2:nodesX-1))/(2*dy)).*((omega(2:nodesY-1,3:nodesX)-...
        omega(2:nodesY-1,1:nodesX-2))/(2*dx)))-(((psi(2:nodesY-1,3:nodesX)-...
```

```
        psi(2:nodesY-1,1:nodesX-2))/(2*dx)).*((omega(3:nodesY,2:nodesX-1)-...
        omega(1:nodesY-2,2:nodesX-1))/(2*dy))))) +
 (nu*((((omega(2:nodesY-1,3:nodesX)-...
        (2*omega(2:nodesY-1,2:nodesX-1))+omega(2:nodesY-1,1:nodesX-2)))/
(dx^2))+...
        (((omega(3:nodesY,2:nodesX-1)-(2*omega(2:nodesY-1,2:nodesX-1))+...
        omega(1:nodesY-2,2:nodesX-1)))/(dy^2))))-
(12*nu*omega(2:nodesY-1,2:nodesX-1)/(wallLengthZ^2));
    omega = omega + domegadt*dt;

    %boundary points for omega
    for j = 1:nodesY
        omega(j,nodesX) = ((wallLengthZ^2)*((2*psi(j,nodesX-1)/
(dx^2))+(2*velocityRightWall/dx))); %right wall
        omega(j,1) = ((wallLengthZ^2)*((2*psi(j,2)/(dx^2))-
(2*velocityLeftWall/dx))); %left wall
    end
    for i = 1:nodesX
        omega(nodesY,i) = ((wallLengthZ^2)*((2*psi(nodesY-1,i)/
(dy^2))+(2*velocityTopWall/dy))); %top wall
        omega(1,i) = ((wallLengthZ^2))*((2*psi(2,i)/(dy^2))-
(2*velocityBottomWall/dy)); %bottom wall
    end

    convergence = 1;
    counter = 1;
    while abs(convergence) > tolerance
        oldPsi = psi;

        %non-boundary points for psi
        psi(2:nodesY-1,2:nodesX-1) = (((dx^2*dy^2)/
(2*(dx^2+dy^2)))*(omega(2:nodesY-...
            1,2:nodesX-1)/(wallLengthZ^2))) + ((dy^2/(2*(dx^2+dy^2)))*...
            (psi(2:nodesY-1,3:nodesX)+psi(2:nodesY-1,1:nodesX-2))) + ...
            ((dx^2/
(2*(dx^2+dy^2)))*(psi(3:nodesY,2:nodesX-1)+psi(1:nodesY-2,2:nodesX-1)));

        convergence = norm(psi-oldPsi,inf); %greatest change in streamfunction
 value
        counter = counter+1;
    end
    %fprintf('t = %e, num_iters = %i, error = %e\n', t, counter, convergence)


    %post-process for u and v
    u(:,nodesX) = 0; %right wall
    v(:,nodesX) = velocityRightWall; %right wall
    u(:,1) = 0; %left wall
    v(:,1) = velocityLeftWall; %left wall
    u(nodesY,:) = velocityTopWall; %top wall
    v(nodesY,:) = 0; %top wall
    u(1,:) = velocityBottomWall; %bottom wall
    v(1,:) = 0; %bottom wall
```

```
    u(2:nodesY-1,2:nodesX-1) = (psi(3:nodesY,2:nodesX-1)-
psi(1:nodesY-2,2:nodesX-1))/(2*dy);
    v(2:nodesY-1,2:nodesX-1) = -(psi(2:nodesY-1,3:nodesX)-
psi(2:nodesY-1,1:nodesX-2))/(2*dx);

    %visualization
    %quiver(u,v);
%     if mod(outer_counter, skip) == 0
%         subplot(2,1,1)
%         surf(x,y,u,'linestyle','none');
%         subplot(2,1,2)
%         surf(x,y,v,'linestyle','none');
%         pause(0.000001);
%     end
    outer_counter = outer_counter + 1;

    %end simulation when approximately steady-state
    du_dt = max(max(abs(u-u_old)));
    dv_dt = max(max(abs(v-v_old)));
   %fprintf('t = %e, num_iters = %i, error = %e, du_dt = %e, dv_dt = %e\n', t,
 counter, ...
       % convergence, du_dt, dv_dt)
    if du_dt < vel_tol && dv_dt < vel_tol
        break
    end
end

% subplot(2,1,1)
% surf(x,y,u,'linestyle','none');
% subplot(2,1,2)
% surf(x,y,v,'linestyle','none');
% toc;

figure(1);
subplot(2,2,1);
plot(x,u(0.9*nodesY,:));
xlabel('x');
ylabel('u');
title('u at y = 0.9');
subplot(2,2,2);
plot(x,u(0.91*nodesY,:));
xlabel('x');
ylabel('u');
title('u at y = 0.91');
subplot(2,2,3);
plot(x,u(0.92*nodesY,:));
xlabel('x');
ylabel('u');
title('u at y = 0.92');
subplot(2,2,4);
plot(x,u(0.93*nodesY,:));
xlabel('x');
ylabel('u');
title('u at y = 0.93');
```

```
figure(2);
subplot(2,2,1);
plot(x,u(0.94*nodesY,:));
xlabel('x');
ylabel('u');
title('u at y = 0.94');
subplot(2,2,2);
plot(x,u(0.95*nodesY,:));
xlabel('x');
ylabel('u');
title('u at y = 0.95');
subplot(2,2,3);
plot(x,u(0.96*nodesY,:));
xlabel('x');
ylabel('u');
title('u at y = 0.96');
subplot(2,2,4);
plot(x,u(0.97*nodesY,:));
xlabel('x');
ylabel('u');
title('u at y = 0.97');

figure(3);
subplot(1,2,1);
plot(x,u(0.98*nodesY,:));
xlabel('x');
ylabel('u');
title('u at y = 0.98');
subplot(1,2,2);
plot(x,u(0.99*nodesY,:));
xlabel('x');
ylabel('u');
title('u at y = 0.99');


figure(4);
subplot(2,2,1);
plot(x,v(0.9*nodesY,:));
xlabel('x');
ylabel('v');
title('v at y = 0.9');
subplot(2,2,2);
plot(x,v(0.91*nodesY,:));
xlabel('x');
ylabel('v');
title('v at y = 0.91');
subplot(2,2,3);
plot(x,v(0.92*nodesY,:));
xlabel('x');
ylabel('v');
title('v at y = 0.92');
subplot(2,2,4);
plot(x,v(0.93*nodesY,:));
```

```
xlabel('x');
ylabel('v');
title('v at y = 0.93');

figure(5);
subplot(2,2,1);
plot(x,v(0.94*nodesY,:));
xlabel('x');
ylabel('v');
title('v at y = 0.94');
subplot(2,2,2);
plot(x,v(0.95*nodesY,:));
xlabel('x');
ylabel('v');
title('v at y = 0.95');
subplot(2,2,3);
plot(x,v(0.96*nodesY,:));
xlabel('x');
ylabel('v');
title('v at y = 0.96');
subplot(2,2,4);
plot(x,v(0.97*nodesY,:));
xlabel('x');
ylabel('v');
title('v at y = 0.97');

figure(6);
subplot(1,2,1);
plot(x,v(0.98*nodesY,:));
xlabel('x');
ylabel('v');
title('v at y = 0.98');
subplot(1,2,2);
xlabel('x');
ylabel('v');
plot(x,v(0.99*nodesY,:));
title('v at y = 0.99');


figure(7);
subplot(2,2,1);
plot(x,u(0.5*nodesY,:));
xlabel('x');
ylabel('u');
title('u at y = 0.5');
subplot(2,2,2);
plot(x,u(0.6*nodesY,:));
xlabel('x');
ylabel('u');
title('u at y = 0.6');
subplot(2,2,3);
plot(x,u(0.7*nodesY,:));
xlabel('x');
ylabel('u');
```

```
title('u at y = 0.7');
subplot(2,2,4);
plot(x,u(0.8*nodesY,:));
xlabel('x');
ylabel('u');
title('u at y = 0.8');

figure(8);
subplot(2,2,1);
plot(x,v(0.5*nodesY,:));
xlabel('x');
ylabel('v');
title('v at y = 0.5');
subplot(2,2,2);
plot(x,v(0.6*nodesY,:));
xlabel('x');
ylabel('v');
title('v at y = 0.6');
subplot(2,2,3);
plot(x,v(0.7*nodesY,:));
xlabel('x');
ylabel('v');
title('v at y = 0.7');
subplot(2,2,4);
plot(x,v(0.8*nodesY,:));
xlabel('x');
ylabel('v');
title('v at y = 0.8');
```