Simulation of fluid flow and fluid-structure interactions in microdevices

Submitted by:
Lim Lui Cheng
U036317E

Department of Mechanical Engineering

In partial fulfillment of the Requirements for the Degree of Bachelor of Engineering National University of Singapore

Session 2006/2007
ABSTRACT

The use of microdevices in the life sciences and biotechnology applications has gained popularity over the last few years due to huge improvements in the manufacturing processes needed for such microscale dimensions. The advantages of using microdevices over conventional equipment are plenty. However, as the use of such devices is still in the early stage, there are still a lot of uncertainties over their use.

The project aims to observe deformation shape changes and trajectories of cells moving in a microchannel using a finite element software, Femlab. Finite element analysis hence plays an important role in helping to understand the interactions of the system under conditions that are simulated to replicate nature and provides a tool for visualizing phenomena not possible to be observed using conventional observation equipment. As the field of fluid-structure interaction is one that requires complex computations, the limits of the software and hardware are also stretched in order to obtain the results required. Many problems dealing with computational inaccuracies and choice of solver parameters have had to be dealt with along the way and repeated testing have to be done to ensure the fidelity of the experiments.

The study is able to show the stresses and deformation when fluid load is applied on the cell while it is flowing through a microchannel. The level of stress applied by the fluid load is too low to damage the cells and is about an order lower than that
commonly experienced in cell structure testing but higher stress may possibly be incurred when the cell collides with the microchannel walls.

Trajectories of the cell motion have also been obtained but are susceptible to instabilities of the numerical methods used in the time-stepping algorithm. Improvements have been made by using a more robust algorithm in the programming but more research and improvement in technology is required to further this field of study.

The overall project has covered a large range of materials from multiphysics finite element simulation to numerical methods used in programming and has been a fruitful experience for the researcher.
ACKNOWLEDGEMENTS

I wish to express sincere appreciation for the following people, who in some way or another have provided valuable assistance and contribution to the progress and success of this project.

First and foremost, I would like to thank my supervisor, Assistant Professor Lim Kian Meng for his help and guidance in the weekly meetings. A/P Lim Kian Meng has taught me not only engineering principles but also critical thinking methodology and has also helped me without hesitation in other academic matters.

I would like to thank the examiner, Associate Professor Prakash Thamburaja for his assessment and pointers during the interim examination.

I would also like to express my gratitude to the staff at Dynamics Lab who has helped me along the way. I would like to thank Mr Cheng, Ahmad, Amy and Priscilla for their invaluable help in the laboratory.

I would also like to thank Mr Rémi Magnard and Mr Anders Ekeroth of Comsol Technical Support for their advice in troubleshooting the problems encountered in the process of using Comsol Multiphysics.

Last but not least, I would like to thank my fellow FYP colleagues in the Dynamics Laboratory for their assistance and support in this project.
# TABLE OF CONTENTS

ABSTRACT i

ACKNOWLEDGEMENTS iii

TABLE OF CONTENTS iv

LIST OF FIGURES v

LIST OF TABLES vi

LIST OF SYMBOLS vii

1 INTRODUCTION 1
    1.1 BACKGROUND
    1.2 OBJECTIVE

2 LITERATURE REVIEW 4

3 THEORY 5
    3.1 THE FINITE ELEMENT METHOD
    3.2 ARBITRARY LAGRANGIAN EULERIAN FRAMES
    3.3 EXPLICIT EULER AND RUNGE KUTTA (RK4) NUMERICAL METHODS
    3.4 3D ROTATIONAL BODIES DYNAMICS

4 METHODOLOGIES 12
    4.1 FEMLAB MODELLING
    4.2 COMSOL SCRIPT PROGRAMMING

5 RESULTS AND ANALYSIS 19

6 CONCLUSION 31

7 RECOMMENDATIONS 32

REFERENCES 33

APPENDICES 35
    A1: EXPLICIT EULER METHODOLOGY
    A2: RUNGE KUTTA 4 TH ORDER (RK4) METHODOLOGY
    A3: TABLE OF CONSTANTS USED
    A4: TABLE OF BOUNDARY CONDITIONS USED
    A5: MATLAB FILES
LIST OF FIGURES

1. Representation of Euler Angles
2. Material constants and dimensions of model
3. Important boundary conditions used in model
4. Example of mesh deformation in ALE application mode
5. Flowchart of manual mesh regularization
6. Deformation plot of 2D solid elastic cell
7. Deformation of 2D membrane created using 2D solid elements
8. Time evolution using Femlab’s transient analysis solver
9. Mesh without partitioning
10. Mesh with partitioning
11. Comparison of forces in y-direction with respect to displacement from center
12. Time evolution using Comsol programming script
13. Comparison of trajectories of 2 similar spheres with different time steps using Explicit Euler method
14. Comparison of trajectories of 2 similar spheres with different time steps using RK4 method
15. Stress analysis of RBC models at different orientations
16. Force vs Orientation angle of a RBC model
17. Angular displacement against time of red blood cell
18. Time evolution of a RBC model to determine stable orientation
LIST OF TABLES

1. Table of constants
2. Table of boundary conditions
### LIST OF SYMBOLS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Acceleration, m/s²</td>
</tr>
<tr>
<td>$F_d$</td>
<td>Drag Force, N</td>
</tr>
<tr>
<td>m</td>
<td>Mass, kg</td>
</tr>
<tr>
<td>t</td>
<td>Time, s</td>
</tr>
<tr>
<td>u</td>
<td>Velocity x – component, m/s</td>
</tr>
<tr>
<td>v</td>
<td>Velocity y – component, m/s</td>
</tr>
<tr>
<td>V</td>
<td>Fluid velocity, m/s</td>
</tr>
<tr>
<td>w</td>
<td>Velocity z – component, m/s</td>
</tr>
<tr>
<td>$P$</td>
<td>Position vector used in calculation of rotational matrices, m</td>
</tr>
<tr>
<td>$V$</td>
<td>Velocity vector used in calculation of rotational matrices, m/s</td>
</tr>
<tr>
<td>$R$</td>
<td>Radius of sphere, m</td>
</tr>
<tr>
<td>$U_\infty$</td>
<td>Free stream velocity, m/s</td>
</tr>
<tr>
<td>{A}</td>
<td>Fixed reference plane</td>
</tr>
<tr>
<td>{B}</td>
<td>Rotating reference plane</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>Angle between x – axis of {A} and line of intersection of {A} and {B}, rad</td>
</tr>
<tr>
<td>$\beta$</td>
<td>Angle between z – axis of {A} and z – axis of {B}, rad</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>Angle between x – axis of {B} and line of intersection of {A} and {B}, rad</td>
</tr>
<tr>
<td>$\Omega$</td>
<td>Angular velocity, rad/s</td>
</tr>
<tr>
<td>$\mu$</td>
<td>Dynamics viscosity, kg/m/s</td>
</tr>
<tr>
<td>$\nu$</td>
<td>Kinematic viscosity, m²/s</td>
</tr>
<tr>
<td>$\rho$</td>
<td>Density, kg/m³</td>
</tr>
<tr>
<td>$\pi$</td>
<td>Pi, mathematical number approximately equal to 3.141592654</td>
</tr>
</tbody>
</table>
1 INTRODUCTION

1.1 BACKGROUND

In the recent years, there has been rapid progress in the manufacturing of microfluidic devices used for various aspects of engineering and biomedical purposes. Microfluidics refers to a set of technologies that control the flow of minute amounts of liquids or gases, typically measured in nano- and picoliters, in a miniaturized system and microfluidic devices are characterized by microchannels having dimensions in the micrometer (µm) region. The most mature application of microfluidic technology is in the commonly used inkjet printer which uses orifices less than 100µm in diameter to generate ink droplets. Microfluidic devices have over the years moved to applications in biotechnology such as the development of DNA chips and lab-on-a-chip technology where they are being used to detect bacteria, viruses and cancer cells and having many advantages as compared to conventional analysis. Some of these advantages are lower fluid consumption, better process control, higher analysis speed and a lower fabrication cost; all of which reduce cost and time and are beneficial to present and future patients.

Microfluidic devices however are not just scaled down versions of conventional testing equipment as the physics changes at the micro-scale level. As the dimensions of a microfluidic device are small, particles suspended in a fluid become comparable in size to the device itself, which dramatically alters system behavior. Although the fluid properties remain the same as that at the micro scale,
Simulation of fluid flow and fluid structure interactions in microdevices

some properties such as surface tension, viscosity, and electrical charges can become dominant forces on a fluid because the surface-to-volume ratio is much greater than for macro-scale systems.

Therefore the stresses and strains which act on biological cells as they flow through a microdevice can be quite different and more complex than through conventional lab equipment. Stresses and strains have been known to produce certain biological and biochemical responses in cells leading to events such as spontaneous cell movement, cell differentiation and even cell death. Hence, the study of forces acting on cells in microfluidic devices through the fluid medium and the resulting deformation is an important first step towards a quantitative study in the change of physical properties of cells or biomolecules which pass through microfluidic devices [1]. Another area of interest is in the trajectories of the cells as they pass through the microchannel as this has implications on cell sorting techniques whereby cells are sorted by size or other parameters through fluorescent excitation.

As the field of microfluidics is relatively new, numerical simulations of microfluidic systems are helpful in providing a research tool. By incorporating the complexities of channel and cell geometry, fluid flow patterns as well as the structural mechanics of cells into a numerical model, the behavior of a system can be accurately predicted when an intuitive prediction may be difficult or impossible to be proven via mathematical methods. Numerical modeling also allows
visualization of complex flow phenomena that will result due to fluid structure interactions that may not be easily obtained experimentally due to the minute dimensional nature of the system.

The weakness of numerical modeling is the fact that it is not guaranteed to exactly replicate events in nature, particularly if there are physical phenomena that are not considered and incorporated into the model. Numerical approximations required by the finite element method also results in slightly inaccurate simulation results. However, careful examination of simulation results and comparison of numerical and experimental or theoretical data can validate the use of the model as a predictive tool. The finite element method (FEM) is implemented through a commercial finite element software, Femlab, and used here together with a programming tool, Comsol Script, to determine the stresses and strains acting on a cell passing through a microfluidic device and its resulting deformation and trajectory.

1.2 OBJECTIVE

The objective of this research project is to investigate the fluid-structure interactions and trajectory of cells in a microchannel using finite element simulation. In particular, a spherical shaped cell and a bi-concaved shaped cell similar to a red blood cell (RBC) are separately analyzed.
2 LITERATURE REVIEW

Much research has been done in both the areas of simulation of fluid-structure interaction and that of experimental deformation of cells. Lim et al has investigated various experimental techniques for the study of biomechanics of single cell and molecule [2] and in particular the deformation of living cells using laser traps [3]. The study also constructed an actual three-dimensional finite element model of a human red blood cell and comprehensive computational studies using ANSYS were performed on the model. The finite element model matches the experimental results under direct stretching of the cell using laser tweezers. Large deformation of red blood cells in shear flow has also been investigated using simulation through the immerse boundary method by Eggleton and Popel [4] while elastic membranes under varying edge constraints and Young’s moduli has been explored using ANSYS by Yang et al [5]. Measurements of red blood cell deformation and velocity in capillaries with a diameter smaller than that of resting red blood cell in vivo have also been investigated by Jeong et al [6] by the use of high speed cameras to capture images of blood flow in capillaries and then estimating the deformation and velocity. An in depth explanation of the theory and application of the use of Arbitrary Lagrangian-Eulerian (ALE) Frames have also been by J. Donea et al [7] which explains the advantages of the ALE method in fluid dynamics and nonlinear solid mechanics. Various models involving fluid flow and fluid structure interactions have also been built and made available in Femlab’s model library with documentation and served as reference for this project.
3 THEORY

3.1 THE FINITE ELEMENT METHOD

The finite element method (FEM) is a numerical solution technique applicable to a broad range of physical problems, the variables of which are related by means of algebraic, differential or integral equations. The technique is now an integral part of the design and analysis process and is used for many industrial and academic applications. With the growth of computing power and commercially available programs, its ease of use, cost effectiveness and reliability is increasing being acknowledged and the technique is now a standard tool for analysis.

The FEM process includes modeling of the geometry of the model, meshing or discretizing the geometry created into elements so as to approximate the solution within an element easily using simple functions, defining material properties which is easy to input but may require extensive testing or review of literature as in the case of this project is to obtain certain hard to determine material properties such as the Young’s modulus of red blood cells. Boundary, initial and loading conditions must also be specified which require experience, knowledge and engineering judgment. Finally, the solution is obtained by solving the discretized system, simultaneous equations for the field variables at the nodes of the mesh. The reader is directed to refer to references [8, 9] for a more detailed explanation of the FEM methodology and its practical uses.
3.2 ARBITRARY LAGRANGIAN EULERIAN FRAMES

Arbitrary Lagrangian Eulerian (ALE) frames is an approach to solving problems in engineering which combines the use of the classical Lagrangian and Eulerian reference frames. It is used largely in the analysis of fluid-structure interaction systems and is very helpful when analyzing structural motions in which the structure is severely deformed, such as an impact problem or the analysis of a very flexible structure.

The Lagrangian Reference Frame is largely used in solid mechanics. It sets up a reference frame by fixing a grid to the material of interest and as the material deforms, the grid deforms with it. In this method, conservation of mass is automatically satisfied because the individual sections of the grid always contain the same amount of mass. For structure motions with large deformation in which the grid becomes excessively distorted, the integration time steps become smaller and smaller because they are based on the size of the smallest section of the grid.

The Eulerian Reference Frame, which is fixed in space, is the typical framework used in the analysis of fluid mechanics problems. In motion predictions solved through the Eulerian approach, the solution is generally measured in the net flow through a certain area and conservation of mass is taken into account explicitly by measuring the flux in and out of each grid section.
The arbitrary Lagrangian-Eulerian (ALE) approach combines the use of the two reference frames. It allows for both a flexible grid and a grid that allows for material to flow through it. In essence, it takes the best part of both reference frames and combines them in to one. This is helpful in problems with large deformations in solid mechanics and in fluid-structure interaction. It allows for the grid to track the material to some extent, but when the grid deforms excessively and distorts the aspect ratio of the grid beyond an acceptable point it adjusts the grid and measures the flux of the material during the adjustment of the grid. The difficulty when using the ALE approach is deciding how much to allow a grid to deform and how much flux to allow. This is usually done by setting a limit on the distortion of a segment of a grid and once it deforms past that limit then that part of the grid is remeshed.

Nonetheless, the use of the ALE may lead to failure when the mesh is deformed over certain limits and results in inverted coordinates. Inverted coordinates do not denote the failure of the entire model but implies that that results are these points cannot and will not be used in further iterations. As long as these points are not in the vicinity of the area of interest, the model can be still assumed to be reliable. Nevertheless, if there is a huge number of such inverted coordinates, the accuracy of the solution deteriorates and eventually, the solution will cease to converge.
3.3 EXPLICIT EULER AND RUNGE KUTTA NUMERICAL METHODS

Engineering systems are often subjected to transient excitation and a finite difference method which can be explicit or implicit is basically used to solve the time stepping. Both explicit and implicit methods are approaches for mathematical simulation of physical processes, or in other words, they are numerical methods for solving time-variable ordinary and partial differential equations.

Explicit methods calculate the state of a system at a later time from the state of the system at the current time, while an implicit method finds it by solving an equation involving both the current state of the system and the later one. Certain points to take note are that the time marching in explicit methods is extremely fast and the coding required is also very straightforward. It is particularly suited for simulating highly nonlinear, large deformation, contact and extremely fast events of mechanics.

However, it is conditionally stable which means that if the time step is too large and above some critical limit, then the solution might become unstable. In contrast, implicit methods often require some form of iterative coding which makes it more complicated but it is unconditionally stable and the size of the time step will not affect the stability of the solution. Therefore, implicit algorithms use larger time steps as compared with explicit methods.
For the purpose of this project, either the Explicit Euler or one of the Runge Kutta methods, RK4, numerical algorithm is applied. Both are explicit methods and their basic theories are presented in the Appendix [A1, A2]. The Explicit Euler algorithm was used initially and the RK4 was later adopted as an attempt on the improvement of the numerical solutions due to its advantages.

Although the RK4 requires four sampling routines per time step and a more complex programming routine, the RK4 is more accurate and stable as compared to the Explicit Euler method. This has been proven by mathematician and numerical analyst, Germund Dahlquist, and verified in numerous computational exercises done on various problems [10]. The truncation error using the RK4 is the same order of the time step to the power of four whereas the truncation error using the Explicit Euler is of the same order of the time step. This means that as the time step is reduced, the truncation error associated with RK4 decreases as a much faster pace. The performance of the Runge-Kutta method is therefore vastly superior to that of Euler's method, since the former method is capable of attaining much greater accuracy than the latter using a far smaller number of steps.
3.4 3D ROTATIONAL BODY DYNAMICS

For the trajectory calculations of the cell, a study of rotational 3D body dynamics is needed. As the cell will rotate about its own axis and its moment of inertia will change constantly while moving along the microchannel, there is a need to utilize more than 1 frame of reference. Consider the reference frame to be the Cartesian frame, \{A\}. Another reference frame, \{B\}, is fixed on the center of gravity of the cell. The moment of inertia is calculated on this frame \{B\}, which is rotating with respect to frame \{A\}.

According to Euler’s rotation theorem, any rotation can be described using 3 angles. If the rotations are written in terms of rotation matrices B, C, and D, then a general rotation \( R \) can be written as \( R = B C D \). The three angles giving the three rotation matrices are called Euler angles. There are several conventions for Euler angles, depending on the axes about which the rotations are carried out.

![Figure 1: Representation of Euler Angles [11]](image)
Using the convention represented by the image in the previous page, where the blue lines show the fixed frame \( \{A\} \), the red lines show the rotated frame \( \{B\} \) and the green line, \( N \) shows the intersection of the 2 planes, we obtain the following rotational matrices.

\[
B = \text{rot}(x, \alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix} \quad C = \text{rot}(y, \beta) = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \\
D = \text{rot}(z, \gamma) = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix}
\]

Consider a fixed vector \( \mathbf{BP} \) unchanging with respect to frame \( \{B\} \). Its description in another frame \( \{A\} \) is given as \( ^A\mathbf{P} = ^A\mathbf{R}^B\mathbf{P} \)

If frame \( \{B\} \) is rotating, then \( ^A\mathbf{P} \) will be changing even though \( \mathbf{BP} \) is constant; and

\( ^A\mathbf{P} = ^A\mathbf{R}^B\mathbf{P} \) or \( ^A\mathbf{V}_p = ^A\mathbf{R}^B\mathbf{V}_p^B \) where \( \mathbf{V}_p = \dot{\mathbf{P}} \)

\( \mathbf{BP} \) is then substituted with \( ^A\mathbf{R}^B \mathbf{P} \) to obtain \( ^A\mathbf{V}_p = ^A\mathbf{R}^B\mathbf{R}^{-1}^A\mathbf{V}_p \) which can be simplified as \( ^A\mathbf{V}_p = ^A\mathbf{S}^A\mathbf{V}_p \) giving a relationship between the displacement vector and velocity vector in the reference frame \( \{A\} \).

The \( S \) matrix is known as the angular velocity matrix and is given in 3D by

\[
S = \begin{bmatrix} 0 & -\Omega_z & \Omega_y \\ \Omega_z & 0 & -\Omega_x \\ -\Omega_y & \Omega_x & 0 \end{bmatrix}
\]

This matrix is then used in the calculation of the rotational velocity of vectors in the time evolution analysis.
4 METHODOLOGIES

4.1 FEMLAB MODELLING

Femlab was chosen for this research project due to its ease in use of implementing multiphysics and inbuilt Arbitrary- Lagrangian-Eulerian (ALE) frames. The multiphysics coupling combines fluid flow with structural mechanics by using a Moving Mesh (ALE) application mode to capture the fluid structure interactions. The fluid flow application mode is defined on the ALE frame, whereas the structural mechanics application mode for the solid is defined on the reference frame.

The combinations of application modes used in this project are:

• A structural mechanics application mode using the large-deformation option:
  - Plane Strain and Plane Stress in 2D
  - Axial Symmetry, Stress-Strain in 2D axisymmetry
  - Solid, Stress-Strain in 3D

• Moving Mesh (ALE) using Winslow smoothing

• The Stokes flow application mode with non-ideal weak constraints that provides the fluid loads on the structure.

The fluid structure interactions couplings appear on the boundaries between the fluid and the solid using Lagrange multipliers. The forces acting on the sphere were then obtained by integrating the entire surface and deformation plots were obtained using the post-processing tools.
Simulation of fluid flow and fluid structure interactions in microdevices

The following 2 figures illustrate the important material constants, dimensions and boundary conditions of the model. More detailed information on the actual equations used in the formulation can be referenced in Appendix [A3, A4].

Figure 2: Material constants and dimensions of model

Figure 3: Important boundary conditions used in model
Simulation of fluid flow and fluid structure interactions in microdevices

The model used here is an elastic cell without a membrane due to limitations by the computer in solving a cell with a membrane structure. Nevertheless, much care has been taken to choose suitable values corresponding to such a model. Some important points to highlight are that the literature values for Young’s modulus values of red blood cells vary and the value $26\pm 7$ kPa obtained by Dulinska et al [12] was chosen for use in this research project as it agrees with other literature and is obtained as an average value using Atomic Force Microscopy thus being more suited for a elastic solid sphere in contrast with membranous sphere. Literature values for Poisson’s ratio are in the range of 0.49 to 0.5 but as the software is unable to handle the incompressible formulation when the value is close to 0.5, a compromised value of 0.4 was used instead.

An iterative solver, generalized minimal residual method (GMRES), was chosen with incomplete LU decomposition as the preconditioner with a restart value of 50. An iterative solver was chosen due to the large memory requirements needed for a 3D model. The GMRES in particular was chosen for its stability in ill-conditioned systems such as that encountered in fluid-structure problems which are unsymmetrical and have zeros on the diagonals. The drawback to the GMRES is that the work and storage requirements increase linearly with the number of iterations and is thus computationally expensive. The solution therefore is to restart the iterative process after some number of iterations, m.
The transient analysis of a cell in the fluid conduit was able to be modeled and solved using Femlab but the movement of the cell was restricted to only a small displacement before the mesh becomes deformed to the extent that many inverted coordinates were present and the mesh quality became too low to ensure accurate computation. This problem is as shown below in the figures using a sample 2D model.

![Figure 4: Example of mesh deformation in ALE application mode](image)

As seen from the figure above, the mesh becomes distorted as time progress and any simulation and values obtained using such a deformed mesh cannot be accepted. There was no built-in check for mesh regularization and hence the use of a “user initiated” mesh regularization technique must be used to ensure that the mesh quality is kept within acceptable limits. This procedure is shown using the flow chart in the following page.
Figure 5: Flowchart of manual mesh regularization technique

This method of mesh regularization technique helps in keeping the computational mesh as regular as possible during the solution process, thereby avoiding excessive distortions and squeezing of the computing zones and preventing mesh inversion. This method also decreases the numerical errors due to mesh distortion.

This process however, was time intensive and only restricted to the finite length of the microchannel. An automatic solution was sought through the use of programming and 3D dynamics.
4.2 COMSOL SCRIPT PROGRAMMING

A program was written using Comsol Script which calls upon Femlab to do the finite element simulation required at each time step. The programming language is similar to Matlab programming language with the exception of certain functions related to the finite element methods which are not available in the Matlab library. Force and moments were obtained using the finite element method whereas mass, volume and moment of inertia for the irregularly shaped RBC were calculated using Solidworks. The required 3D linear and rotational dynamics were then coded and calculated and the model was redrawn at each incremental time step. The rotational dynamics used included Euler angle and angular rotational matrices commonly used in robotics were used to obtain the orientation of the cell as it moved through the microchannel. The time stepping was obtained initially using an explicit Euler numerical approach while the RK4 approach is later applied as an improvement on the numerical stability towards the later stages of the project.

By using the program, one is able to calculate and model the trajectory of the cell in a microchannel. The transverse forces acting on the object are 2 orders smaller compared to the forces acting in the direction of the fluid flow. Due to the limited length of the simulated microchannel, the final position of the cell released from an off centre position was unable to be obtained within the finite enclosure. Ideally, this analysis should be done on a long tube but that would mean a large and expensive computational domain which is unviable.
A solution would be to split the domain to follow the moving particle and assume that the pressure drop across the length of these (shorter) domains to be the same each time. The particle will influence and affect the pressure field but it is assumed here that the particle is small enough such that it will not affect the pressure change. Although this assumption might not be true, it is undertaken to minimize the computational requirements. Hence, the program can be edited such that the cell will not be moved with the flow and will remain at the center of the computational domain while being free to rotate or move in the transverse plane.

The programs used for the analysis of the trajectories of cells in microchannels are appended in appendix [A5].
5 RESULTS AND ANALYSIS

A series of 2D models of a cell in a cylinder were first created to ensure the formulation and the couplings of the multiphysics model are correct. Deformation plots in the 2D regime are as predicted and are shown in the figures below where the oval lines shows the deformation after fluid flows pass the sphere.

Axial symmetric 2D models of a sphere in a circular tube were then simulated to ascertain the correct use of the ALE and compare to theoretical results using Stokes’s formula, \( F_D = 6\pi\mu U_x R \). However, the Stokes’s formula is used for a sphere in free stream and therefore a series of 2D axial symmetric models were created with increasing cylinder diameter. The results were extrapolated to determine if the simulated model will yield a force equivalent to the theoretical answer when the cylinder is infinitely large or when the sphere is in free stream. The results agree with an error of approximately 5% and the error is deemed acceptable by FEM standards.
Using Femlab in the transient analysis mode, we are able to determine the trajectory of the sphere for a small finite displacement as shown in the following figures.

For the stress distribution of the sphere in transient analysis, we are able to see that the stresses acting on the sphere while moving in the fluid is not on the face facing the fluid flow but on the sides parallel to the flow. The stresses are also very small and do not cause much deformation of the sphere. In particular, as the forces in the transverse direction to the flow are small, the displacements in the x-z plane are not visible. A limitation due to memory constraints also resulted in the solver not being
able to solve the bi-concave red blood cell model due to the curved surfaces. Hence, various methods have been sought to reduce the memory usage. This includes trying out various different solvers, optimizing the tradeoff between preconditioner quality and memory usage, and creating a partition of mesh as seen in the figure below.

![Fig 9: Mesh without partitioning](image1.png)  ![Fig 10: Mesh with partitioning](image2.png)

The idea of portioning is commonly used in FEM whereby areas of higher stress concentration are meshed finely to obtain a higher accuracy and areas where the results are of less interest are meshed coarsely to save on memory. In this case, a slight variation is used whereby the volume of fluid near to the sphere is modeled as an ALE mesh whereas the volume away from the sphere can be modeled using an Eulerian frame because the grid has no need to flex. The method was successfully implemented but has only marginal improvement in terms of memory usage and hence an alternative method was sought.
In order to reduce the memory requirements due to the various application modes used, a decision was made to have the ALE application mode was removed and a program to be written for the time stepping algorithm. In this way, the finite element software would only focus on the fluid simulation and would require less memory. However, certain experiments had to be first run to ensure that the forces experience with and without the ALE would not differ by a large extent to cause discrepancy in the motion.

The 3D sphere was then shifted to various positions in the conduit to determine the effect of the forces on the sphere by the fluid flow with and without the ALE and plotted in Figure below.

![Graph of forces vs displacement](image.png)

*Figure 11: Comparison of forces in y-direction wrt displacement from center*
Simulation of fluid flow and fluid structure interactions in microdevices

Solving the model with and without the ALE mode only resulted in a slight difference in the forces acting in the direction parallel to the flow and forms the basis for not using the ALE mode when programming the time stepping function. However, for detailed deformation of the cells, it is still encouraged that the ALE mode be used.

Using the Comsol Script and writing a time stepping program which calls upon Femlab to do the finite element analysis and obtain the forces and moments at each time step, the displacement of the sphere and the red blood cell in a microchannel were obtained. An example of a spherical cell being released from an off centre stationary point is shown in the figures below.

![Figure 12: Time evolution using Comsol programming script](image)

Various similar simulations have been carried out to determine the robustness of the program. One of the parametric studies that was carried out involved the size effect of the spherical cell on the trajectory. By tracking various spherical cells with different diameters over a long distance, an attempt was made to determine their trajectories within the microchannel and to make comparisons. The results will have
possible applications on the process of cell sorting where cells are sorted based on their sizes.

However, the results obtained only gives slight indication that spheres of a smaller diameter moved towards the center and were deemed not to be able to formulate conclusive evidence about the size effects on the stable position of the cell.

In order to verify the accuracy of the data values, convergence tests on the mesh and the time stepping algorithm were carried out. Convergence tests using a finer mesh showed that the results agree but when the time step was reduced, the discrepancy of the trajectory grew with increasing time steps. The displacement data was collated and presented using Matlab graphics. This is shown in the figure in the following page where the boundaries of the graph represent the walls of the microchannel and the lines represent the trajectories of the spheres as they move through the microchannel.
Fig 13: Comparison of trajectories of 2 similar spheres with different time steps using Explicit Euler method

The Explicit Euler was found to be not suitable for the time stepping algorithm and a more robust algorithm, RK4 was used in place as an improvement on the program as although both of them are explicit codes, the RK4 has better convergence properties and have an error with an order of 4 times the size of the time step compared to Explicit Euler which has an error proportional to the size of the time step. This means that when the time step is halved, the error induced in the Explicit Euler algorithm will reduce by half as well but the error caused by the RK4 will reduce by 16 times.
Fig 14: Comparison of trajectories of 2 similar spheres with different time steps using RK4 method

The same experiment was plotted using the new RK4 algorithm and it shows better results. This allows the user to have higher confidence in running simulations to determine the trajectories of a cell released from any location in the microchannel and can be used for further testing in future.

Further analysis was then extended to the bi-concave red blood cell. Different orientations of the red blood cell were investigated as part of a quasi-static analysis and this process was automated using Comsol Script programming. Sample figures of the red blood cell at different orientation in the microchannel are shown in Figure 15 on the next page.
A graph of the total forces in each direction acting on the cell at different orientations were plotted and shown below.

The graph above fits the theoretical prediction of the forces acting on the model at different orientations. The force acting in parallel to the fluid flow (Force Y) will be the highest and the force acting transverse in the top to bottom direction is the lowest when the angle of orientation is at 90 degrees. It is noted that the maximum
force exerted on the cell is $40 \times 10^{-9} \text{N}$ or 40pN which is much lesser than the forces exerted for cell deformation by Lim et al [2] in the range of 70pN ~ 200pN and hence the fluid load does not cause excessive deformation in the red blood cell travelling in a microchannel. This analysis is also the background for the next analysis which is to find the stable orientation of a red blood cell in a microchannel. Using the time evolution algorithm, simulated analyses have shown that the red blood cell will travel along the microchannel in the direction highly dependent on their orientation, behaving much like a sail and hitting the walls of the microchannel and resulting in a crash of the simulation as the boundaries of the cell and microchannel overlap and hence is of little value of research.

However, by studying the case of a red blood cell being held in the center of a microchannel through means of a laser trap, a stable orientation can be determined. The simulation was modified to reflect such a condition and a graph of the magnitude of the angular displacement against time was plotted to visualize the angular motion of the RBC as shown in the following page.
Figure 17: Angular displacement against time of red blood cell

Figure 18: Time evolution of a RBC model to determine stable orientation
The simulations show that the red blood cell structure move away from its original position parallel to the fluid flow, reach an equilibrium orientation almost parallel to the fluid flow and oscillate around this position as the model rotates. The orientation is not expected and is most probably due to minor perturbations in the fluid flow that had caused it to displace from its original position and is not entirely impossible due to the concave nature of the red blood cell that might affect the fluid flow. Schmid-Schönbein and co-workers [13] reported that red blood cells monitored using an inverted interference contrast microscope also exhibit behavior of alignment almost parallel to the fluid flow and rotate in a tank thread-like behavior after reaching this stable orientation. The difference is that for the case in this research, the red blood cells are stiff as we are only analyzing the fluid interactions. The situation shown is therefore also similar to an irregularly shaped aerofoil in steady flow.
6 CONCLUSION

Finite element simulations of a spherical cell and a bi-concave red blood cell have been carried out to study cell deformation and trajectory. Time evolution of a cell traveling in a microchannel was achieved and simulations were studied to determine interesting phenomenon. The forces acting on an elastic sphere at fluid loads commonly used in microdevices are not substantial enough to cause large deformation and hence is implied that they will not result in serious biological responses. However, the model used in the research project is termed as a drop-like model with no membrane and only models a living cell to a certain degree. Hence, caution must be exercised in applying results obtained to that of living cells.

The current research does not yield substantial accurate information on the trajectory of the cell for a microchannel of infinite length but progress is still underway to improve the algorithm used in the calculation of the trajectory. Although some of the results were not able to be adequately obtained, this research project has been beneficial in that many different concepts and theories have been revised or learnt along the way. In order to solve the problems and work around the resource constraints, a thorough understanding of finite element concepts, ALE, 3D rotational dynamics and numerical methods have to be understood. The fluid-structure interactions concepts learnt and applied are not limited to just that of biotechnology but also to applicable to a large range of other problems such as flutter of bridges, sloshing in tanks and instability of aerospace structures.
7 RECOMMENDATIONS

Some recommendations for improvement of the project would be to change the algorithm for the time stepping to one that has better stability compared to the Explicit Euler method. The RK4 method is one such improvement but iterative solvers such as the implicit Newmark’s method should also be considered for use which will result in better stability.

The simulations ran in this research project stops whenever the cell impacts the walls of the microchannel due to boundary respecting errors and internal errors arising from the FEM structure. This is to be expected but a probable way of solving this problem would be to set boundary conditions on the surface of the cell such that the cell will experience a force “from” the wall proportional to its distance from the wall and also its approach velocity when it is very close to the wall. If done in a proper way, one will be able to obtain information of the forces acting on the cell upon impact which will be most probably larger then that imposed by the fluid loads and will have significance on the cell.
REFERENCES


Let the initial value problem be specified as follows:

\[ y'(t) = f(t, y(t)) \]
\[ y(t_0) = y_0 \]

where \( f \) is a function that maps \([t_0, \infty) \times \mathbb{R}^d \) to \( \mathbb{R}^d \), and the initial condition \( y_0 \in \mathbb{R}^d \) is a given vector.

Starting with the differential equation (1), we replace the derivative \( y' \) by the finite difference approximation

\[ y'(t) = \frac{y(t + h) - y(t)}{h} \]

which yields the following formula

\[ y(t + h) = y(t) + hf(t, y(t)) \]

This formula is usually applied in the following way. We choose a step size \( h \), and we construct the sequence \( t_0, t_1 = t_0 + h, t_2 = t_0 + 2h, \ldots \) We denote by \( y_n \) a numerical estimate of the exact solution \( y(t_n) \). Motivated by (3), we compute these estimates by the following recursive scheme

\[ y_{n+1} = y_n + hf(t_n, y_n) \]

The restriction for using this formulation is that the timesteps used cannot be too large. Hence, the timesteps are kept small to prevent large variations in the forces and the model would not be able to converge to a stationary point and orientation.
Appendix A2: Runge Kutta 4\textsuperscript{th} Order (RK4) Methodology

Let the initial value problem be specified as follows:

\[ y'(t) = f(t, y(t)) \]
\[ y(t_0) = y_0 \]

Then the RK4 method is given as below

\[ K_1 = f(t_n, y_n) \]
\[ K_2 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2} \times K_1) \]
\[ K_3 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2} \times K_2) \]
\[ K_4 = f(t_n + h, y_n + h \times K_3) \]

Then \[ y_{n+1} = y_n + \frac{h}{6} (K_1 + 2K_2 + 2K_3 + K_4) \]

At each point of the RK4, the value of the next point is obtained by the present point added to the product of the estimated slope of the present point and a time step value. The main advantages of Runge-Kutta methods are that they are easy to implement, are very stable and they can be used even with only 1 initial point as compared to multistep methods. The primary disadvantages of Runge-Kutta methods are that they require significantly more computer time than multi-step methods of comparable accuracy, and they do not easily yield good global estimates of the truncation error.
# Appendix A3: Table of constants used

<table>
<thead>
<tr>
<th>Name and description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Young’s modulus, $E$</td>
<td>26000 Pa</td>
</tr>
<tr>
<td>Viscosity, $\mu$</td>
<td>0.001 Pa.s</td>
</tr>
<tr>
<td>Density of fluid, $\rho_f$</td>
<td>1000 kg/m$^3$</td>
</tr>
<tr>
<td>Density of cell, $\rho_c$</td>
<td>1125 kg/m$^3$</td>
</tr>
<tr>
<td>Poisson’s ratio, $\nu$</td>
<td>0.4</td>
</tr>
<tr>
<td>Maximum velocity at inlet, $u_{\text{max}}$</td>
<td>0.0005 m/s</td>
</tr>
<tr>
<td>Half width of inlet, $h_{\text{width}}$</td>
<td>5e-5 m</td>
</tr>
<tr>
<td>Half height of inlet, $h_{\text{height}}$</td>
<td>5e-5 m</td>
</tr>
<tr>
<td>Length of inlet</td>
<td>3e-4 m</td>
</tr>
</tbody>
</table>
Appendix A4: Table of boundary conditions used

<table>
<thead>
<tr>
<th>Solid Stress State (Smsld) Application Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boundary conditions on surface of cell:</td>
</tr>
<tr>
<td>Face load (force/area) X-dir. (Fx)</td>
</tr>
<tr>
<td>Face load (force/area) Y-dir. (Fy)</td>
</tr>
<tr>
<td>Face load (force/area) Z-dir. (Fz)</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Moving Mesh (ALE) Application Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boundary conditions on surfaces of microchannel:</td>
</tr>
<tr>
<td>Mesh Displacement</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Boundary conditions on surface of cell:</td>
</tr>
<tr>
<td>Mesh Displacement</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stokes Flow (Mmglf) Application Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boundary conditions on cell and side wall of microchannel</td>
</tr>
<tr>
<td>Boundary conditions at inlet</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Boundary conditions at outlet</td>
</tr>
</tbody>
</table>
Appendix A5:

Matlab file 1 - Force and Moment Calculations for different orientations of RBC

flclear fem

% COMSOL version
clear vrsn
vrsn.name = 'COMSOL 3.3';
vrsn.ext = '';
vrsn.major = 0;
vrsn.build = 405;
vrsn.rcs = '$Name: '$;
vrsn.date = '$Date: 2006/08/31 18:03:47 $';
fem.version = vrsn;

% Constants
fem.const = {'rho','1000', ...
'u_max','0.05', ...
'hheight','50e-6', ...
'hwidth','50e-6', ...
'mu','0.001', ...
'x0','0', ...
'y0','150e-6', ...
'z0','0'};

AR = [1,0,0];

% Start of loop
for x = 0:1:90
x
angle = x/180*Pi

% Geometry
g1=block3('100e-6','300e-6','100e-6','base','corner','pos',{'-50e-6','0','-50e-6'},'axis','{0,0,1}','rot','0');

% Using multiple geometries
flclear xfem
% Keep global fields
fields={'functions','xmesh','globalexpr','const','version'};
for ii=1:length(fields)
if isfield(fem,fields{ii})
xfem.(fields{ii})=fem.(fields{ii});
fem=rmfield(fem,fields{ii});
end
end
if isfield(fem,'sol')
    fem=rmfield(fem,'sol');
end
xfem.fem{1}=fem;

% Geometry 2
carr={curve2([0,7.4999999999999999E-6,7.4999999999999999E-6,7.4999999999999999E-6],[1,1,1,1]), ...
curve2([1.2499999999999999E-5,1.9999999999999998E-5,1.9999999999999998E-5],[7.4999999999999999E-6,7.4999999999999999E-6,0],[1,0.7071067811865475,1]), ...
curve2([1.9999999999999998E-5,1.9999999999999998E-5,1.9999999999999998E-5],[0,7.4999999999999999E-6,-7.4999999999999999E-6,0],[1,1,1,1]), ...
curve2([0,0],[-4.0E-6,4.0E-6],[1,1])};
g2=geomcoerce('solid',carr);
g3=revolve(g2,'angles',[0,6.283185307179586],'revaxis',[0 0;0 1],'wrkpln',[-5.0E-5 -5.00000000006776E-5 0.9999;1.5E-4 1.5E-4 1.5E-4;-5.0E-5 0.9999 -4.999999999932247E-5]);

% Geometry 1
g3=move(g3,[50eE6,0,50eE6]);
g3=rotate(g3,angle,AR,[0, 150eE6 ,0]);
fem=x fem.fem{1};

% Analyzed geometry
clear s
s.objs={g1,g3};
s.name={'BLK1','REV1'};
s.tags={'g1','g3'};

fem.draw=struct(s,s);
fem.geom=geomcsg(fem);
g4=geomcomp({g1,g3},'ns',{g1,'g3'},'sf','g1-g3','face','none','edge','all');

% Analyzed geometry
clear s
s.objs={g4};
s.name={'CO1'};
s.tags={'g4'};

fem.draw=struct('s',s);
fem.geom=geomcsg(fem);

% Initialize mesh for geometry 1
fem.mesh=meshinit(fem, ...
    'hauto',5);
xfem.fem{1}=fem;

% (Default values are not included)

fem=xfem.fem{1};

% Application mode 1
clear appl
appl.mode.class = 'GeneralLaminarFlow';
appl.module = 'MEMS';
appl.gporder = {4,2};
appl.cporder = {2,1};
appl.assignsuffix = '_mmglf';
clear prop
prop.analysis='static';
prop.nisot='Off';
prop.inerterm='Off';
appl.prop = prop;
clear bnd
bnd.type = {'noslip','uv','strout'};
bnd.v0 = {0,'u_max*(x+hwidth)*(hwidthEx)*(z+hheight)*(hheightE ...}
    0};
bnd.ind = [1,2,1,1,3,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1];
appl.bnd = bnd;
clear equ
equ.rho = 'rho';
equ.eta = 'mu';
equ.cporder = {{1;1;1;2}};
equ.gporder = {{1;1;1;2}};
equ.ind = [1];
appl.equ = equ;
fem.appl{1} = appl;
fem.frame = {'ref'};
fem.border = 1;
clear units;
units.basesystem = 'SI';
fem.units = units;
xfem.fem{1} = fem;

flclear fem
fem.sdim = {'x','y'};
fem.border = 1;
clear units;
units.basesystem = 'SI';
fem.units = units;
xfem.fem{2} = fem;

% Multiphysics
xfem=multiphysics(xfem);

% Extend mesh
xfem.xmesh=meshextend(xfem,'geoms',1);

% Solve problem
xfem.sol=femstatic(xfem, ...
    'solcomp', {'w','u','p','v'}, ...
    'outcomp', {'w','u','p','v'}, ...
    'ntol', 1.0E-3, ...
    'linsolver', 'gmres');

% Save current fem structure for restart purposes
fem0=xfem;

% Plot solution
postplot(xfem, ...
    'flowdata', {'u','v','w'}, ...
    'flowcolor', [1.0,0.0,0.0,0], ...
    'flowlines', 15, ...
    'flowtype', 'line', ...
    'title', 'Streamline: Velocity field [m/s]', ...
    'grid', 'on', ...
    'campos', [0,1.500000071246177E-4,0.001658312452005475], ...
    'camtarget', [0,1.500000071246177E-4,0], ...
    'camup', [0,1,0], ...
    'camva', 11.620844691944823);
view('xy')
saveimage(['Pic' num2str(x)])

% Geometry 2
fem=xfem.fem{2};

% Geometry objects
clear s
s.objs={g2};
s.name={'CO1'};
s.tags={'g2'};

fem.draw=struct('s',s);
xfem.fem{2}=fem;

% Integrate for force
Fx=postint(xfem,'T_x_mmglf', ...  
   'unit','N', ...  
   'dl',[6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21], ...  
   'edim',2);

% Integrate for force
Fy=postint(xfem,'T_y_mmglf', ...  
   'unit','N', ...  
   'dl',[6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21], ...  
   'edim',2);

% Integrate for force
Fz=postint(xfem,'T_z_mmglf', ...  
   'unit','N', ...  
   'dl',[6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21], ...  
   'edim',2);

% Integrate for moment
Mx=postint(xfem,'(yEy0)*T_z_mmglf-(zEz0)*T_y_mmglf', ...  
   'unit','N', ...  
   'dl',[6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21], ...  
   'edim',2);

% Integrate for moment
My=postint(xfem,'-(xEx0)*T_z_mmglf-(zEz0)*T_x_mmglf)', ...  
   'unit','N', ...  
   'dl',[6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21], ...  
   'edim',2);
% Integrate for moment
Mz=postint(xfem,'(x-x0)*T_y_mmglf-(y-y0)*T_x_mmglf', ...
  'unit','N', ...
  'dl',[6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21], ...
  'edim',2);

Force(x:1) = Fx;
Force(x:2) = Fy;
Force(x:3) = Fz
Moment(x:1) = Mx;
Moment(x:2) = My;
Moment(x:3) = Mz

end
Matlab file 2 – Obtain transformation matrix from reference to rotating frame

```
close all
clear all
format long

%initialize original position of reference vectors and normal vector
r1 = [2; 0; 0]
r2 = [0; 0; 2]

r3 = -cross(r1,r2)
len1 = (r3(1)^2+r3(2)^2+r3(3)^2)^0.5

VA = r3./len1        %Rotation frame vector
VB = [0;0;len1]      %Cartesian frame vector of equal length to VA
AY = subspace(VA,VB) %Angle to rotate about yEaxis

len2 = len1.*cos(AY)

VC = [0;0;len2]     %Cartesian frame vector
VD = VA - VC

len3 = (VD(1)^2+VD(2)^2+VD(3)^2)^0.5

VE = [len3;0;0]

AX = 0
AZ = subspace(VD,VE) %Angle to rotate about z-axis

% Pitch
B = [1 0 0;
     0 cos(AX) -sin(AX);
     0 sin(AX) cos(AX)]

%Yaw
C = [cos(AY) 0 sin(AY);
     0 1 0;
     -sin(AY) 0 cos(AY)]

%Roll
D = [cos(AZ) -sin(AZ) 0;
     sin(AZ) cos(AZ) 0;
     0 0 1]

%Euler rotation matrix
T = B*C*D
```
Matlab file 3 – Matlab file for trajectory of sphere

close all
clear all
fclear fem
format long
format compact

% COMSOL version
clear vrsn
vrsn.name = 'COMSOL 3.3';
vrsn.ext = '';
vrsn.major = 0;
vrsn.build = 405;
vrsn.rcs = '$Name:  $';
vrsn.date = '$Date: 2006/08/31 18:03:47 $';
fem.version = vrsn;

% Constants
fem.const = {'rho','1000', ...
  'u_max','0.05', ...
  'hheight','50e-6', ...
  'hwidth','50e-6', ...
  'mu','0.001',...
  'x0','0',...
  'y0','150e-6',...
  'z0','0'};

% Externally defined constants
radius = 20e-6

dist_x = 0;
dist_y = 0;
dist_z = 0;
tot_dist_x = 0;
tot_dist_y = 0;
tot_dist_z = 0;
vel_init_x = 0;
vel_init_y = 0;
vel_init_z = 0;
vel_curr_x = 0;
vel_curr_y = 0;
vel_curr_z = 0;
\[
\begin{align*}
w_0 &= [0;0;0]; \\
\text{angle} &= 0; \\
\text{AR} &= [1,0,0]; \\
\text{density} &= 1125; \\
\text{volume} &= 4/3\pi(20eE6)^3; \\
\text{mass} &= \text{density} \times \text{volume}; \\
I_{xx} &= 2/5\text{mass}(20eE6)^2; \\
r_1 &= [20eE6; 0; 0]; \\
r_2 &= [0; 0; 20eE6]; \\
\text{len1} &= 20eE6 \\
\text{timestep} &= 0.00001
\end{align*}
\]

% Geometry
\[
\begin{align*}
g1 &= \text{block3('100eE6','300eE6','100eE6','base','corner','pos', \{'-hwidth','0','-hwidth\'}, \{'0','0','1\'}, \{'rot','0','const',fem.const\}); \\
g2 &= \text{sphere3('20eE6','pos', \{'15eE6','150eE6','20eE6\'}, \{'0','0','1\'}, \{'rot','0\});}
\end{align*}
\]

% Start of loop
\[
\begin{align*}
\text{for } x = 1:1:1000 \\
\text{tic} \\
x
\end{align*}
\]

\[
\begin{align*}
g3 &= \text{geomcomp}\{g1,g2\}, \{'ns', \{'g1','g2'\}, \{'sf','g1-g2','face','none','edge','all'\};}
\end{align*}
\]

% Analyzed geometry
\[
\begin{align*}
\text{clear } s \\
s.\text{objs} &= \{g3\}; \\
s.\text{name} &= \{'CO1'\}; \\
s.\text{tags} &= \{'g3'\};
\end{align*}
\]

\[
\begin{align*}
\text{fem.draw} &= \text{struct('s',s);} \\
\text{fem.geom} &= \text{geomcsg(fem);} \\
\end{align*}
\]

% Initialize mesh
\[
\begin{align*}
\text{fem.mesh} &= \text{meshinit(fem, ...} \\
\text{'hauto',6);} \\
\end{align*}
\]

% (Default values are not included)

% Application mode 1
\[
\begin{align*}
\text{clear appl} \\
\text{appl.mode.class} &= 'GeneralLaminarFlow'; \\
\text{appl.module} &= 'MEMS';
\end{align*}
\]
appl.gporder = {4,2};
appl.cporder = {2,1};
appl.assignsuffix = '_mmglf';
clear prop
prop.analysis='static';
prop.nisot='Off';
prop.inerterm='Off';
appl.prop = prop;
clear bnd
bnd.type = {'noslip','uv','out'};
bnd.v0 = {0,0,'u_max*(x+hwidth)*(hwidth-x)*(z+hheight)*(hheight-z)/((hwidth)^2*(hheight)^2)',...}
0};
bnd.ind = [1,2,1,1,3,1,1,1,1,1,1,1,1,1];
appl.bnd = bnd;
clear equ
equ.rho = 'rho';
equ.eta = 'mu';
equ.cporder = {{1;1;1;2}};
equ.gporder = {{1;1;1;2}};
equ.ind = [1];
appl.equ = equ;
fem.appl{1} = appl;
fem.frame = {'ref'};
fem.border = 1;
clear units;
units.basesystem = 'SI';
fem.units = units;

% Multiphysics
fem=multiphysics(fem);

% Extend mesh
fem.xmesh=meshextend(fem);

% Solve problem
fem.sol=femstatic(fem, ...
  'solcomp',{'w','u','p','v'}, ...  
  'outcomp',{'w','u','p','v'}, ...  
  'linsolver','gmres');

% Save current fem structure for restart purposes
fem0=fem;
% Plot solution
postplot(fem, ...
    'flowdata',{u',v',w'}, ...'
    'flowcolor',[1.0,0.0,0.0,0.0], ...
    'flowlines',15, ...
    'flowtype','line', ...
    'title','Streamline: Velocity field [m/s]', ...
    'grid','on', ...
    'campos',[0,1.500000071246177E4,0.001658312452005475], ...
    'camtarget',[0,1.500000071246177E-4,0], ...
    'camup',[0,1,0], ...
    'camva',11.620844691944823);

view('xy')
saveimage(['Pic_xy' num2str(x) '.bmp'])
view('zx')
saveimage(['Pic_zx' num2str(x) '.bmp'])

% Integrate for force
Fx=postint(fem,'T_x_mmglf', ...
    'unit','N', ...
    'dl',[6,7,8,9,10,11,12,13], ...
    'edim',2);

% Integrate for force
Fy=postint(fem,'T_y_mmglf', ...
    'unit','N', ...
    'dl',[6,7,8,9,10,11,12,13], ...
    'edim',2);

% Integrate for force
Fz=postint(fem,'T_z_mmglf', ...
    'unit','N', ...
    'dl',[6,7,8,9,10,11,12,13], ...
    'edim',2);

% Integrate for moment
Mx=postint(fem,('y-y0)*T_z_mmglf-(z-z0)*T_y_mmglf', ...
    'unit','N', ...
    'dl',[6,7,8,9,10,11,12,13], ...
    'edim',2);

% Integrate for moment
My=postint(fem,(-(x-x0)*T_z_mmglf-(z-z0)*T_x_mmglf)', ...
% Integrate for moment
Mz=postint(fem,'(x-x0)*T_y_mmglf-(y-y0)*T_x_mmglf', ...
    'unit','N', ...
    'dl',[6,7,8,9,10,11,12,13], ...
    'edim',2);

% To find displacement
accer_x = -Fx/mass;
accer_y = -Fy/mass;
accer_z = -Fz/mass;
vel_curr_x = vel_init_x + accer_x*timestep;
vel_curr_y = vel_init_y + accer_y*timestep;
vel_curr_z = vel_init_z + accer_z*timestep;
dist_x = vel_curr_x*timestep;
dist_y = vel_curr_y*timestep;
dist_z = vel_curr_z*timestep;

determine new position of CG
% tot_dist_x = tot_dist_x + dist_x
% tot_dist_y = tot_dist_y + dist_y
% tot_dist_z = tot_dist_z + dist_z

% store change in position
Dist(x,1) = dist_x;
Dist(x,2) = dist_y;
Dist(x,3) = dist_z;

% store Force vectors
ACC(x,1) = accer_x;
ACC(x,2) = accer_y;
ACC(x,3) = accer_z;

% store Force vectors
Force(x,1) = Fx;
Force(x,2) = Fy;
Force(x,3) = Fz;

% store Moment vectors
Moment(x,1) = Mx;
Moment(x,2) = My;
Moment(x,3) = Mz;

% update bulk velocity
vel_init_x = vel_curr_x;
vel_init_y = vel_curr_y;
vel_init_z = vel_curr_z;

% To find angular displacement
M = [Mx;My;Mz];
l = blkdiag(Ixx,Ixx,Ixx);
alpha = l\M;

w1 = alpha.*timestep + w0;
teta = w1.*timestep;

% initialize angular velocity and angular velocity matrix
wx = w1(1);
wy = w1(2);
wz = w1(3);

S = [ 0, -wy, wz;...
wz, 0, -wx;...
-wy, wx, 0];

% Obtain velocity component at each point
V1 = S*r1;
V2 = S*r2;

% obtain new position vectors
r3 = cross(r1,r2);
r4 = r1 + V1.*timestep;
r5 = r2 + V2.*timestep;
r6 = cross(r4,r5);

% determine axis for rotation
rr = r6-r3;
rr = rr./len.*len1;
ar = cross(r3,rr);
AR = ar./len;

% determine angle of rotation
angle = subspace(r3,r6);
AGL(x,1) = AR(1);
AGL(x,2) = AR(2);
AGL(x,3) = AR(3);
AGL(x,4) = angle;

AGV(x,1) = wx;
AGV(x,2) = wy;
AGV(x,3) = wz;

% update data
w0 = w1;
r1 = r4;
r2 = r5;
r3 = r6;

xcen = 15e-6 + tot_dist_x;
ycen = 150e-6;
zcen = 20e-6 + tot_dist_z;

dist_x
dist_y
dist_z
g2=move(g2,[dist_x, 0, dist_z]); %dist_y can be set to zero in this case to prevent
    sphere exiting control volume
 g2=rotate(g2,angle,AR,[xcen, ycen, zcen]); %note change in x and z tot_dist

flclear fem

% Constants
fem.const = {'rho','1000', ...
'u_max','0.05', ...
'hheight','50e-6', ...
'hwidth','50e-6', ...
'mu','0.001',...
'x0',xcen, ...
'y0',ycen,...
'z0',zcen};
toc
end