Advances in CML Air Bearing Design and Simulation Tools

Brendan J. Cox and David B. Bogy
Computer Mechanics Laboratory
Department of Mechanical Engineering
University of California
Berkeley, CA 94704

Abstract

Numerical simulation plays a large role in the design of air bearing sliders for hard disk drives (HDD). With the ever increasing drive towards lower magnetic spacing and shorter product cycles, CML's suite of simulation tools has undergone extensive modifications in order to keep pace with the demands of air bearing designers. This paper details the major changes to CML software over the past five years of development. These changes include modifications to the numerical methods used by CML solvers, the addition of new analysis tools and the inclusion of automation capabilities in CMLAir, a graphical user interface for the design of air bearing sliders. A manual describing the use of CMLAir is included in the second half of this report.

1. Introduction

Numerical simulation is an important tool for HDD air bearing designers. In order to effectively develop quality products, engineers need effective solvers as well as powerful tools for creating and running air bearing simulations. Over the past 15 years, many CML students have worked to develop a functional suite of air bearing design and simulation tools. These tools have become quite successful in meeting the needs of air bearing designers and are used throughout the hard disk drive industry. CML's numerical simulators include Quick 4, a static fly height simulator, Dyn 4, a dynamic simulator and CML PIP, a slider Parameter Identification

Program. A graphical user interface, CMLAir has also been designed to allow users to easily create air bearing designs, run numerical simulations and analyze simulation results. While modifications have been made to all of CML's software tools, the primary goal of this paper is to discuss changes made to Quick 4 and CMLAir over the past five years of development.

The ever present push for lower fly heights brings with it a need for smaller tolerances for manufacturing defects. Before the creation of an etching mask is even considered for a candidate slider design, it is necessary to run tens or even hundreds of simulations on that slider design, testing everything from altitude sensitivities to a multitude of manufacturing tolerances. Designing an air bearing slider is largely an iterative process where designs are continually tweaked and optimized. While a full dynamic simulation can provide quite a bit of insight into the performance of a slider design, static simulations are orders of magnitude faster and can be just as elucidating. Thus, Quick 4 is used extensively in the iterative process of slider design.

At the heart of Quick 4 is the solution of the non-dimensional steady state generalized Reynolds equation

$$\frac{\partial}{\partial X}\left(QPH^3 \frac{\partial P}{\partial X} - \Lambda_x PH \right) + \frac{\partial}{\partial Y}\left(QPH^3 \frac{\partial P}{\partial Y} - \Lambda_y PH \right) = 0$$

where $P = p / p_a$, $H = h / h_m$, $X = x / L$, $Y = y / L$ are the non-dimensionalized pressure, bearing clearance, slider length and slider width, respectively; p_a is the ambient pressure; h_m is the reference clearance at the trailing edge center (TEC); L is the length of the slider; $\Lambda_x = \frac{6\mu UL}{p_a h_m^2}$ and $\Lambda_y = \frac{6\mu VL}{p_a h_m^2}$ are the bearing numbers in the x and y directions respectively; U and V are the x and y velocity components of the disk and $Q = f\left(\frac{K_n}{PH}\right)$ is the flow factor, typically based upon a linearized Boltzmann model developed by Fukui and Kaneko [3].

The control volume descretization of Patankar [7] is used to solve the Reynolds equation for slider air bearings with arbitrarily shaped rails. The solution method uses a fixed point scheme to deal with the nonlinear terms and a power law upwind scheme is used to stabilize the solution in the presence of strong convection as discussed in [5]. Finally an alternating line by line solver is coupled with a powerful nonlinear Full Approximation Storage multigrid scheme to improve the convergence of the solution.

2. Meshing and Multigrid Techniques

For the complex geometries found on today's state of the art air bearing sliders, a high quality computational grid is necessary in order to successfully solve the Reynolds equation. The CML static simulator uses a rectangular computational grid that adapts to pressure and geometry gradients. This meshing strategy allows the solver to make better use of available grids and concentrate the grid in areas where the solution or geometry is rapidly changing. To further improve simulation results for complex geometries, grid lines are snapped to nearby slider nodes. This "grid snapping" helps to better capture geometric discontinuities and improves grid convergence, allowing accurate simulations for smaller grid sizes [5].

In addition to a high quality mesh, the solution of the discretized Reynolds equation requires a fast nonlinear solver. When it comes to solving the system of equations resulting from Patankar's dicretization of the Reynolds equation, single grid methods have proven to be inefficient. To speed up the convergence of the solution, Sha Lu [5] implemented a standard Full Approximation Storage multigrid method developed by Brandt [2] using interpolation operators developed by Shyy and Sun [8].

The continued push towards high stiffness sliders has pushed the limits of the multigrid methods used by Lu. For slider designs that appeared as early as 2003, Quick 4 began to exhibit

convergence problems that resulted in stagnating or oscillating residuals. At each iteration, the calculated pressure under the slider was changing but not converging to the correct solution. Through investigations it became clear that the convergence problems were being caused by the coarser computational grids used by the FAS multigrid. With fewer available grid lines, the coarse grids were not able to accurately capture the rails and steep pressure gradients present in complex slider designs. In a sense, the coarser grids were solving a different problem than the finest grid. These invalid coarse-grid corrections would be interpolated back to the finer grid and applied to the solution. Thus, the fine grid solution would be thrown off by the coarse grid correction, resulting in incorrect pressures that would not converge to the true solution.

The easiest way to solve such a problem is for the solver to detect an oscillating or stagnating residual and automatically reduce the number of grid levels used by the multigrid solver. While this method works well to ensure convergence, the solution speed can suffer drastically. In the worst case, the solver will use only a single grid and will loose out on all the speed advantages of the multigrid. With only a single grid, a simulation that should typically take minutes can take hours or even days to complete.

A better solution to the problem can be found by taking steps to improve the quality of the mesh for the coarser grids. To do this, each level of mesh is adapted independently. Selective node snapping is applied to each level of the grid in order to better capture the slider geometry. Adapting each level of mesh independently is an ambitious solution because modifying the coarse grid meshes independently of the fine grid mesh requires a move from nested multigrid methods to non-nested methods. Most multigrid techniques have been designed for nested meshes where the coarse grids are created by removing grid lines from the finer meshes. This technique results in simple interpolation operators and simple bookkeeping for

transferring variables between meshes in a multigrid V-cycle. Since our goal was to adapt each mesh independently, it became necessary to investigate multigrid methods for non-nested meshes. One of the first applications of multigrid methods to unstructured meshes was in Mavriplis and Jameson [6] who developed novel restriction and prolongation functions that are suitable for non-nested unstructured triangular meshes.

In multigrid schemes for unstructured meshes, the variables, residuals and corrections are all transferred in different manners. For the Reynolds equation, the transfer of pressure from the fine grid to the coarse grid is taken as a simple interpolation. In figure 1, consider node *N* on the coarser grid, and first find the triangle *abc* on the fine grid that encloses *N*. A simple linear interpolation of the pressure over triangle *abc* is performed to find the pressure at node *N*. Likewise, a linear interpolation is used when prolonging the correction from the coarse grid to the fine grid. To get the correction at fine grid node, *n*, we find triangle *ABC* that encloses *n* and use a linear interpolation of the correction over *ABC*. The transfer of the fine grid residual is performed in a completely different manner. The residual at the fine grid node *r* is linearly distributed to the nodes *ABC* of the enclosing triangle on the coarse grid. The distribution function is such that, if *r* and *A* coincide then the entire residual at *r* is distributed to *A* and none of the residual is distributed to *B* or *C*. If *r* is moved away from *A* and closer to *B* and *C*, more of the residual is distributed to *B* and *C* and less to *A*. From the description, the obvious choice for this distribution function is the barycentric coordinates of *r* in triangle *ABC*.

The above scheme works well for standard multigrid test problems but exhibits slow convergence of the solution when applied to slider designs. This is due mostly to the geometry of modern HDD slider designs where intricate rails and walls cause sharp changes in geometry over the surface of the slider. By distributing the residual from the fine-grid to the coarse-grid in

the manner specified above, the residual interpolation operator effectively smoothes the residual over a large area. For non-smooth underlying geometries this can place large residuals in places where they don't belong. This in turn leads to incorrect coarse grid corrections and very slow convergence of the solution. Instead of smoothing out the residual, it is more appropriate to transfer it using the same interpolation operators used for pressure. This results in vastly improved convergence as the residual is applied directly to problem areas in the solution.

With the above modifications to the mesh and multigrid, Quick 4 can solve most state of the art slider designs in a reasonable amount of time. However, a very small percentage of sliders still have trouble converging and experience oscillating residuals when simulated. This is due to highly complex rail shapes that, no matter what grid adaptation strategy is used, cannot accurately be modeled by coarse rectangular meshes. To ensure convergence for all slider designs, the code automatically detects a stagnating or oscillating normalized residual and decreases the number of multigrid levels used in the computation. While this causes the code to run slower, it prevents time consuming stagnations in the solution and ensures convergence for almost all slider designs.

In addition to the above modifications, there are other parameters in the multigrid scheme that can be tweaked and tuned to speed up the convergence of the solution. One of the most important parameters is the number of iterations of the line-by-line Gauss-Seidel solver used on each level of the grid. Textbook multigrid calls for 1-3 iterations on each level depending on whether the solution is moving down to coarser grids or being corrected up to finer grids. Due to the nonlinear nature of the Reynolds equation and the high condition number of the resulting system matrix, different numbers of iterations are needed on each level. Previous versions of CMLAir simply used 10 iterations of the Gauss-Seidel solver on each level of the mesh. The

number of iterations currently used by the solver was tuned using a variety of slider designs available in CML in 2005 resulting in improved simulation times for the majority of modern slider designs.

3. Rail and Geometry Improvements

The rails window has been updated in several ways to improve the efficiency of air bearing design. Whether a designer is brainstorming to rapidly create a new design or tweaking an existing design, the modifications described below can greatly speed up the design process and help the user accurately model the slider geometry.

3.1. Undo and Redo of rail changes

Often times, errant mouse work can cause a user to move a slider feature in a way that was not intended. Because of this, probably the most helpful recent addition to the interface is the ability to undo errant modifications made to the rails and walls. To do this, CMLAir maintains a list of all slider design modifications made since a case was opened. If the user wants to undo a recent set of changes to the slider, pressing the *Undo* button repeatedly will replace the current slider design with previous states of the slider design. Users can control the number of changes that can be undone by setting the *Undo Depth* preference in the *Application Preferences* dialog.

3.2. Rail Shapes

Two new rail shape options have been added to the interface. The *Circle Tool* allows users to easily add circular shaped rails to the slider. These rails are often used to create landing pads for contact start stop sliders. The *Spline Tool* allows users to add simple spline shaped rails

to a slider or insert spline features into existing rails. Use of the *Spline Tool* is detailed in the manual section of this paper.

3.3. Rail Manipulation

The rail manipulation section of the rails dialog allows the user to easily modify existing rails. The four arrow buttons allow the user to nudge currently selected points up, down, left or right. The distance of the nudge is controlled by the value specified in the S*tep Size* edit box. Users can also expand and shrink the currently selected rail. The change in rail size is also controlled by the S*tep Size* parameter. The *Insert Node* button will insert nodes into the currently selected rail. Nodes will be inserted along the line connecting the last selected node in the rail to next node in the rail. The node will be offset from this line a bit to prevent inserting collinear points into the rail. The *Round Corner* tool allows the user to round sharp corners in the rail. If the user selects a radius of curvature that is too large for the corner, CMLAir will warn the user and abort the operation. The *Snap to Nodes* option will cause drawing or dragging of rail nodes to snap to other nearby rail nodes. Finally, the *Background Grid* option is useful for prototyping and generating a rough outline shape for a new slider design. All drawing and dragging of rail nodes will be snapped to a background grid. The distance between the background grid points is determined by the value in the S*tep Size* edit box.

3.4. Zoom

If more accuracy is needed when manipulating rail points, the user can use the *Zoom Tool* to zero-in on a particular region of the slider. When the *Zoom Tool* is selected, a single left click on the slider picture will zoom in on that section of the slider while a right click will zoom out. Users can also click and drag the zoom tool to select a rectangular region that will be magnified.

When the mouse button is released, the slider will be zoomed so that the selected rectangular region fills the rails window.

3.5. Wall Update Dialog

It is a common operation for air bearing designers to change rail etch depths in order to change a slider's performance. This provides engineers the ability to tweak a slider's performance after a costly etching mask has been fabricated. When changing the recess of a rail, not only must the recess height of all other rails at that height be changed but the wall profiles must be updated as well. Because of the frequency that this task is performed and labor involved, it is important to automate the changing of slider recess heights. In CMLAir, changing a rail's recess height or the base recess will bring up the U*pdate Wall Preferences* dialog. This dialog allows the user to specify how the walls will be modified to match the change in recess height. The dialog has two sets of options. The first set of options specifies the manner in which the walls will be modified; the second set of options specifies which walls will be modified. All of these options are fully described in the manual section of this paper.

3.6. User Defined Geometry

CMLAir offers air bearing designers the ability to modify the recess height of slider regions without having to add extra rails. This ability can be used to model non-parabolic crown and camber, specify pole trip protrusions or add manufacturing defects to a slider design. The rail recess geometry is modified by specifying an arbitrary sized matrix describing the deformation over a region of the slider at evenly spaced points. Values between specified points are quadratically interpolated to give a smooth surface using a minimum of input points. The input format of the user defined geometry file is specified in manual section of this paper.

3.7. Regional Height Adjustments

In addition to User Defined Geometry files, the user can input multiple *Regional Height Adjustments*. These adjustments allow the user to easily modify the geometry over a region of the slider without having to create an external user defined geometry file. The geometry is modified by specifying a planar deformation defined by slopes in the x and y directions and an offset for the center of the plane. A picture of such a deformation is show in figure 2.

3.8. Geometry Generation

The geometry generation algorithms in previous versions of CMLAir had deficiencies in modeling rail walls that led to large wall protrusions at convex rail corners. An example of this defect can be seen in figure 3. Due to shortcomings in the algorithm, the more acute corners would generate larger spikes in the wall in the corner region. With the continuing increase in complexity of slider designs and advances in slider fabrication abilities, these shortcomings were beginning to hinder the development of accurate slider designs. To eliminate these spikes, a new algorithm was devised to generate rounded convex corners. For these corners, the algorithm determines if the point in question lies in the standard wall region (1) or the corner region (2) as shown in figure 4. Points in the standard wall region are given a recess based on the point's nominal distance *d* from the rail's boundary. This is identical to the method used in previous versions of CMLAir. Points in the corner regions (2) however, are given a recess height based on their distance *d* from the nearest corner of the rail. The computational cost of this fix is negligible while the geometries generated by the resulting algorithm more closely resemble those generated by the slider etching process.

4. New simulations

The main purpose of the CMLAir user interface is to aide the user in designing sliders and running simulations. Most of the previous discussions of additions to the interface have dealt with new additions for designing sliders. The current section discusses new methods for running and automating static and dynamic simulations.

4.1. Shock Simulation

Hard disk drives are being used more and more in mobile applications. In these situations, the robustness of the head disk interface to shock is of great importance. CMLAir can now run simple shock simulations using the dynamic simulator. The shock simulation is implemented by applying a time varying force to the load point of the slider. This is similar to the method used by Zeng [10]. Users have the option of simulating a shock in a simple manner by applying a half sinusoid load to the slider in the direction perpendicular to the surface of the disk or a half sinusoid moment in the pitch or roll directions. If this simple model for shock is unsuitable, it's possible to create a user defined shock file and import it into the interface. The user defined shock file can apply time varying load and moments in the *z*, pitch and roll directions. The format of the user defined shock file is explained in the manual section of this report.

4.2. Static track seeking

The change in flying height during track seeking is a key aspect for air bearing designers. While the dynamic simulator enables the user to simulate the track seeking performance of an ABS design, the resulting simulations are typically very time consuming. A useful alternative is to perform quasi-static approximations of track seeking. Juang and Kubotera [4] have shown that quasi-static approximations can produce accurate track seeking results with much less

computational effort. They showed that the change in effective skew and inertia effects of the seeking motion are the dominant factors in changing the flying height during track seeking. Both of these effects have been taken into account when developing the static track seeking simulation tool for CMLAir. By specifying a limited number of variables, users can model simple track seeking acceleration profiles in the interface and easily run quasi-static track seeking simulations.

4.3. Parameter Identification

CML's Parameter Identification Program is used to obtain modal parameters, modal frequencies, damping ratios and mode shapes of air bearing sliders. In order to extract this information, the Parameter Identification Program requires the user to run four separate simulations. Each simulation requires the user to manually modify input and output files for the dynamic simulator. This process has been automated by CMLAir allowing users to easily set up and run PIP simulations from within the interface. While the actual PIP computations still require the user to export the simulation files from the interface and perform the analysis using Matlab, the process is now much less error prone and cumbersome for the user. The steps required to set up and perform parameter identification are detailed in the manual section of this paper.

5. Python Interpreter

While efforts have been made to create special tools that automate common simulation tasks, it is impossible to predict and develop all the tools a user might need for slider development. To overcome this problem, a Python interpreter has been embedded in the CMLAir user interface. This allows the user to write scripts in the Python language that automate the study of air bearing sliders and extend the interface to work in different environments. In the

most basic of situations, a Python script can be used as a macro language to automate tedious or repetitive tasks. In more ambitious situations, the interpreter can be used to submit jobs to remote machines, tie the interface in with other software or create optimization scripts to aide in the design process. In its current incarnation, the user is not required to have Python installed on the host machine, all the major functionality is contained within the interface and a Python library that is installed with CMLAir. However, to get the most out of Python's extensive functionality and libraries, a working Python installation is still encouraged for the user and can be freely downloaded from www.python.org. Only a passing familiarity with the Python language is necessary to effectively use the CMLAir Python API. A basic tutorial on the use of the CML Python API is included in the manual section of this report. Currently the CML Python API allows the user to modify all existing parameters used by the static solver. Users can also save CMLAir cases and run Quick 4 simulations. If there is demand from the industry, the API can easily be extended to work with dynamic input parameters and run dynamic simulations.

6. Postprocessor Improvements

The postprocessor has seen some useful updates and additions over the past few versions of CMLAir. Updates include an improved result table for displaying and plotting simulation results. The 3D rendering behind the plots of pressure, geometry and contact forces has undergone an extensive update to use OpenGL. This not only results in more visually appealing plots but improves rendering times by orders of magnitude which is very helpful for plotting larger data sets. Many new dialogs and analysis tools have been added to the postprocessor section of CMLAir as well. Tools to integrate shear forces and pressure over regions of the slider as well as analyze mass flow and predict lube depletion have been well received by the industry. The more interesting of these changes are detailed below.

6.1. Spacing Analysis Dialog

In early versions of CMLAir, users could instruct the solver to calculate the flying height of four points of interest on the slider. These points would be output to `result.dat` at the end of a simulation. As of CMLAir 7.0 the interface and solvers have been extended to allow the user to specify up to 100 points of interest. Changes to the input files to Quick 4 are detailed in the appendix. The points of interest are useful for cases where the interested location on the slider is known before running a simulation. However the spacing calculator dialog is a useful tool for investigating the fly heights of specific points on the slider after a simulation is run. Users can either click on points of interest or specify them in the edit box on the left hand side of the window.

6.2. Lube Depletion Dialog

The lube depletion dialog allows the user to numerically determine the lube depletion due to air bearing pressure. In order to accurately predict the lube depletion, it is necessary for the user to have experimental data on the surface energy of the lubricant. A full discussion of the model used to predict lubricant depletion can be found in Ambekar and Bogy [1]. Briefly, the lube thickness at a given point satisfies the following equation:

$$a_1 h^{a_2} + b(h) - C = 0$$

where

$$C = a_1 * h_0^{\,a_2} + b(h_0) + p_{\max}$$

In the equations above, a_1 and a_2, come from an exponential curve fit of the disjoining surface energy of the lubricant; h_0 is the thickness of the lubricant and h is the resulting thickness of the lubricant due to the air bearing pressure; $b()$ is a polynomial curve fit of the polar surface energy

of the lubricant. The interface allows for *b()* to be any order of polynomial. To solve the above equation, CMLAir uses a fast nonlinear solver that closely mimics the implementation of Matlab's fzero() function.

6.3. Mass Flow Analysis Dialog

CMLAir features two separate mass flow calculations. The original mass flow calculation, computed by Quick 4, solves the Reynolds mass flow equation:

$$\frac{1}{12\mu}(\rho h \ \frac{\partial p}{\partial x} - 6U\mu\rho h) + \frac{1}{12\mu}(\rho h \ \frac{\partial p}{\partial y} - 6V\mu\rho h) = 0$$

This computation is useful for viewing the average mass flow under the slider. For a more elucidating view of the mass flow patterns at particular spacings under the slider, users should use the mass flow analysis dialog. This dialog allows the user to view the mass flow at a particular spacing under the slider. The spacing at which the streamlines are calculated can be either a percentage of spacing as depicted by the red line in figure 5 or at a particular recess depth relative to the zero recess plane of the slider as depicted by the red line in figure 6.

The mass flow at a particular spacing can be easily calculated. In [9], Wu derived the velocity distribution under the slider using the second order slip model. Briefly, by integrating the Navier-Stokes equation assuming lubrication conditions and applying the following slip velocity boundary conditions

$$u\Big|_{z=0} = U + \frac{2}{3}a\lambda\frac{\partial u}{\partial z}\Big|_{z=0} - \frac{1}{4}\lambda^2\frac{\partial^2 u}{\partial z^2}\Big|_{z=0}$$

$$u\Big|_{z=h} = -\frac{2}{3}a\lambda\frac{\partial u}{\partial z}\Big|_{z=h} - \frac{1}{4}\lambda^2\frac{\partial^2 u}{\partial z^2}\Big|_{z=h}$$

We obtain the velocity *u* of the gas at a particular spacing.

$$u = \frac{1}{2\mu}\frac{\partial p}{\partial x}\left[z^2 - hz - \left(\frac{2}{3}a\lambda\right)h - \frac{\lambda^2}{2}\right] + U\left[1 - \frac{z + \left(\frac{2}{3}a\lambda\right)}{h + 2\left(\frac{2}{3}a\lambda\right)}\right]$$

In the above calculation, u is the velocity of the gas, μ is the viscosity of the gas, p is the pressure, z is the height at which the velocity will be calculated, a is the accommodation coefficient, λ is the mean free path and U is the convection. Using the above equation, it is a simple matter to calculate and display the velocity field at a particular spacing under the slider.

Plotting the mass flow streamlines as a function of the spacing percentage has been shown to be a good indicator of areas on the slider that are susceptible to particle and lube contamination. Comparisons of mass flow streamlines vs. experimental results of lube contamination are shown in figures 7 through 10. From these two cases we can easily see a correlation between mass flow streamlines and lube contamination on the ABS surface.

7. References

[1] Ambekar, R. and Bogy, D., 2006, "Slider-Lubricant Interactions at the Head-Disk Interface," Technical Report No. 06-007, Computer Mechanics Laboratory, Department of Mechanical Engineering, University of California, Berkeley.

[2] Brandt, A., 1977, "Multi-level Adaptive Solutions to Boundary-value Problems," Mathematics of Computations, Vol. 31, pp. 333-390.

[3] Fukui, S., and Kaneko, R., 1990, "A Database for Interpolation of Poiseuille Flow Rates for High Knudsen Number Lubrication Problems," ASME Journal of Tribology, Vol. 112, pp. 78-83.

[4] Juang, J., Kubotera, H. and Bogy, D., 2005, "Effects of Track-Seeking Motion on the Flying Attitudes of Ultralow Flying Sliders", Technical Report No. 05-021, Computer

Mechanics Laboratory, Department of Mechanical Engineering, University of California, Berkeley.

[5] Lu, S., 1997, “Numerical Simulation of Slider Air Bearings,” Doctoral Dissertation, Department of Mechanical Engineering, University of California, Berkeley.

[6] Mavriplis, D. and Jameson, A., 1987, ”Multigrid Solution of the Two-Dimensional Euler Equations on Unstructured Triangular Meshes,” AIAA paper 87-0353.

[7] Patankar, S. V., 1980, *Numerical Heat Transfer and Fluid Flow*, McGraw-Hill, New York.

[8] Shyy, W., and Sun, C. S., 1993, "Development of a Pressure-correction/Staggered-grid Based Multi Grid Solver for Incompressible Recirculating Flows, *Computers and Fluids*, Vol. 22, No. 1, pp. 51-76.

[9] Wu, L., 2002, “Physical Modeling and Numerical Simulation of the Slider Air Bearing Problem of Hard Disk Drives,” Doctoral Dissertation, Department of Mechanical Engineering, University of California, Berkeley.

[10] Zeng, Q. and Bogy, D., 2000**,** “Numerical Simulation of Shock Response of Disk-Suspension-Slider Air Bearing Systems in Hard Disk Drives”, Technical Report No. 00-003, Computer Mechanics Laboratory, Department of Mechanical Engineering, University of California, Berkeley.

8. Figures

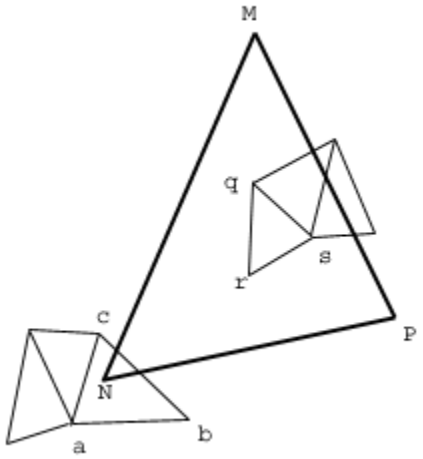

Fig. 1. Transfer of variables in a multigrid V-Cycle

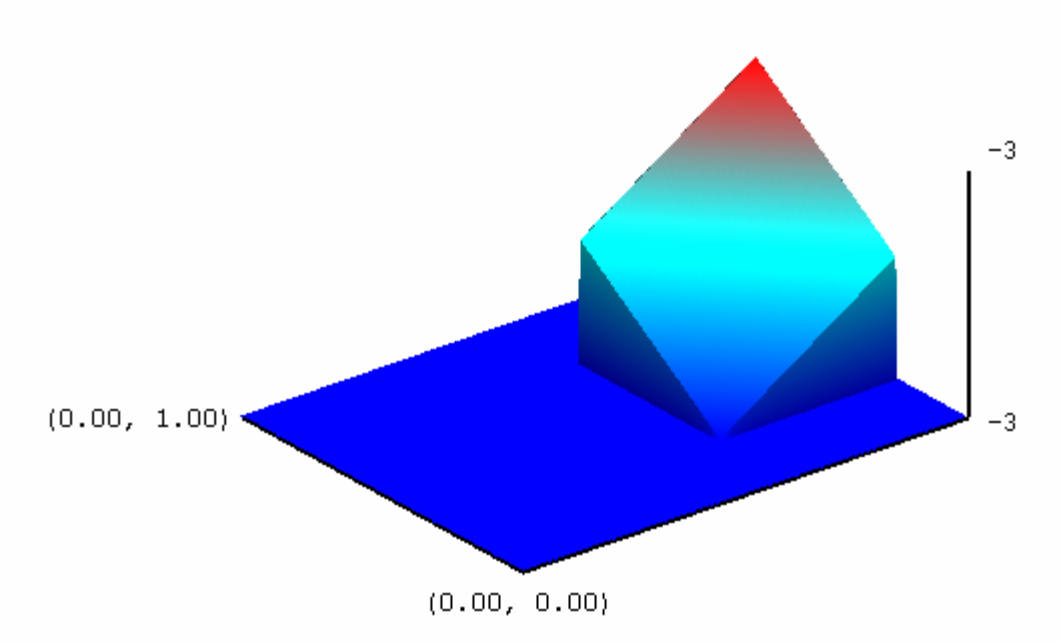

Fig. 2.Regional Height Adjustment applied to the trailing edge of a flat slider

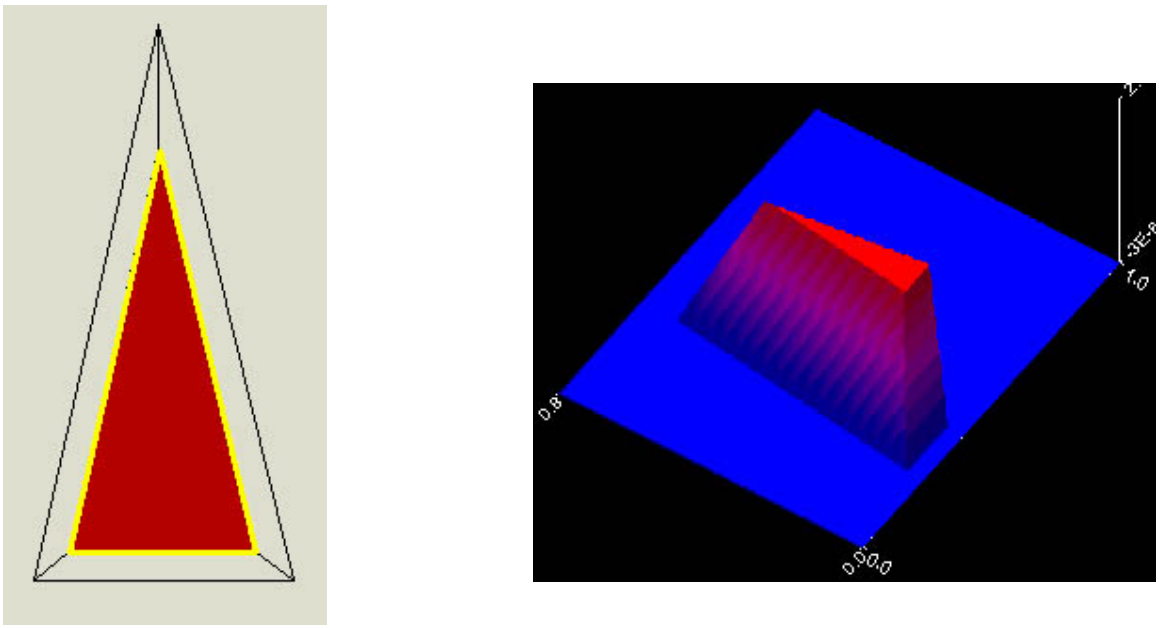

Fig. 3. Two and three dimensional depiction of excess wall protrusions at acute rail corners

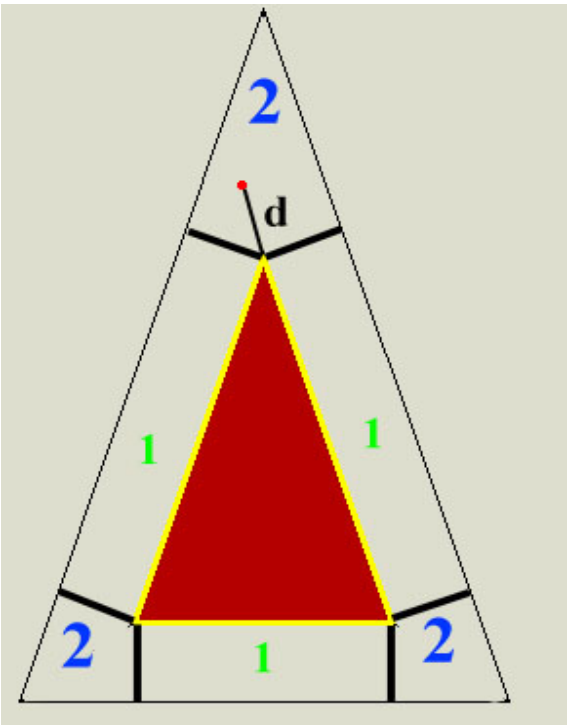

Fig. 4. Illustration of new algorithm to round walls at convex corners

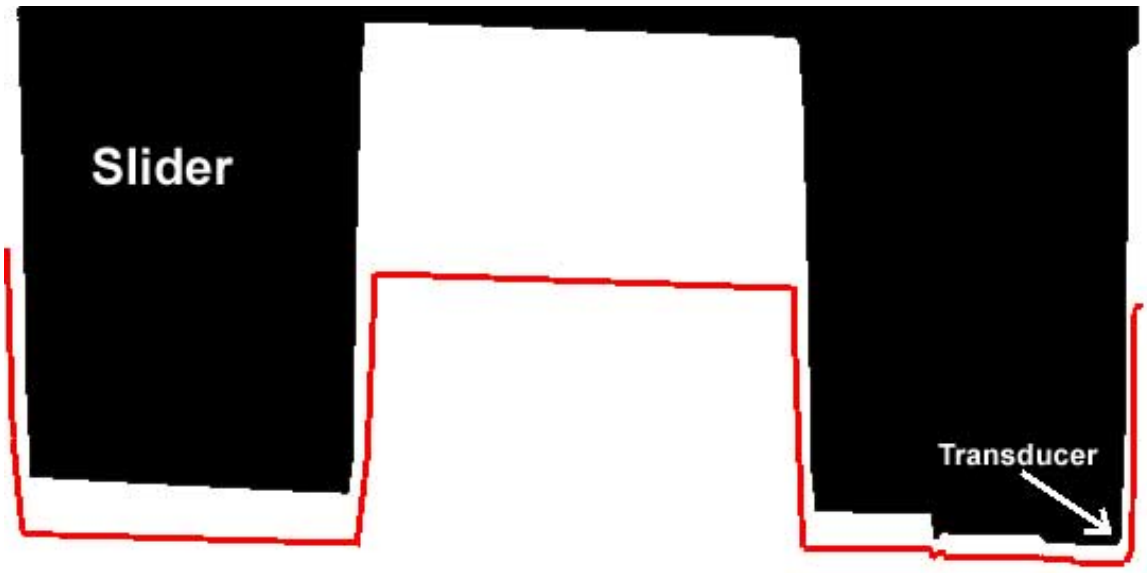

Fig. 5. Cross section of slider showing streamline height based on percentage of spacing

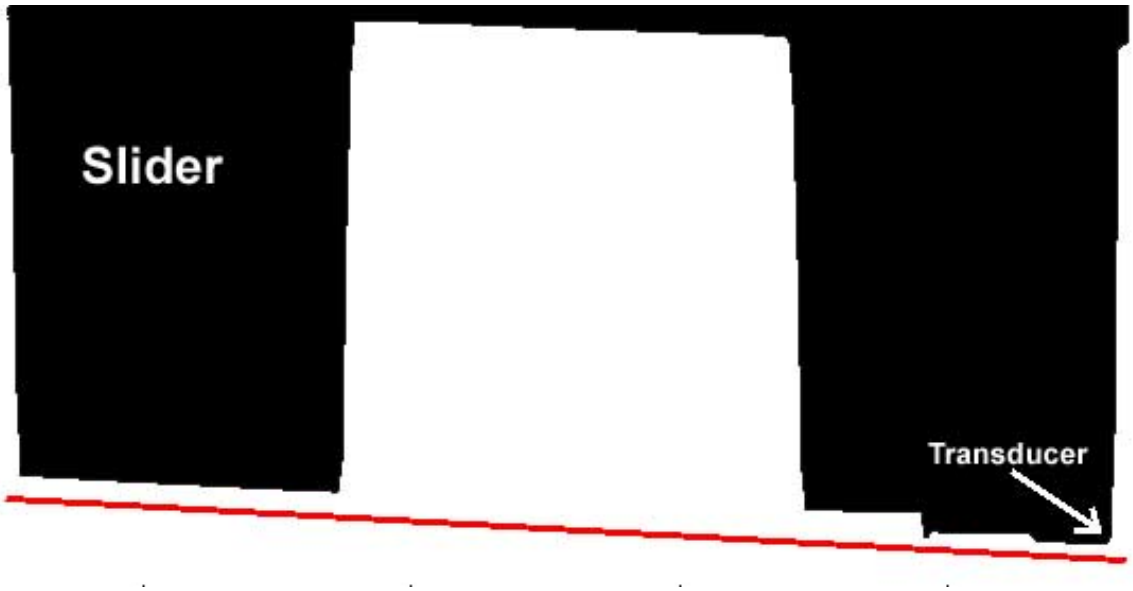

Fig. 6. Cross section of slider showing streamline height based on recess depth

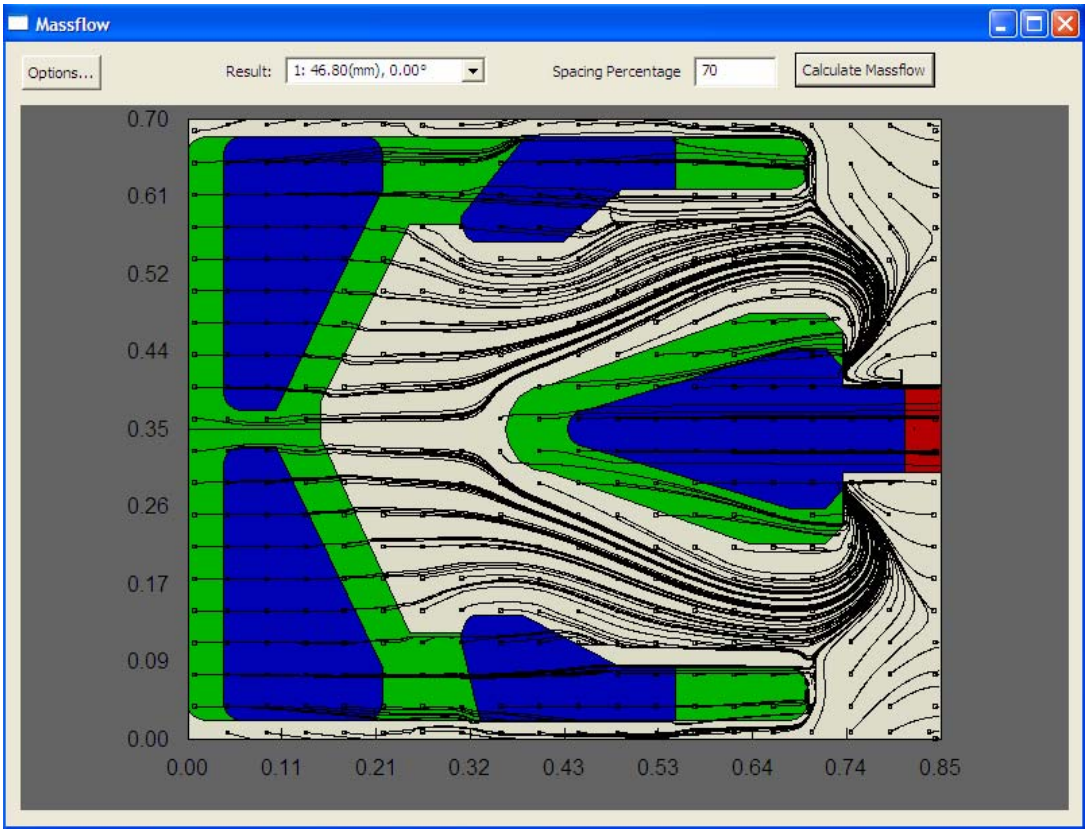

Fig. 7. Mass flow streamlines calculated in CMLAir for CML/INSIC femto slider

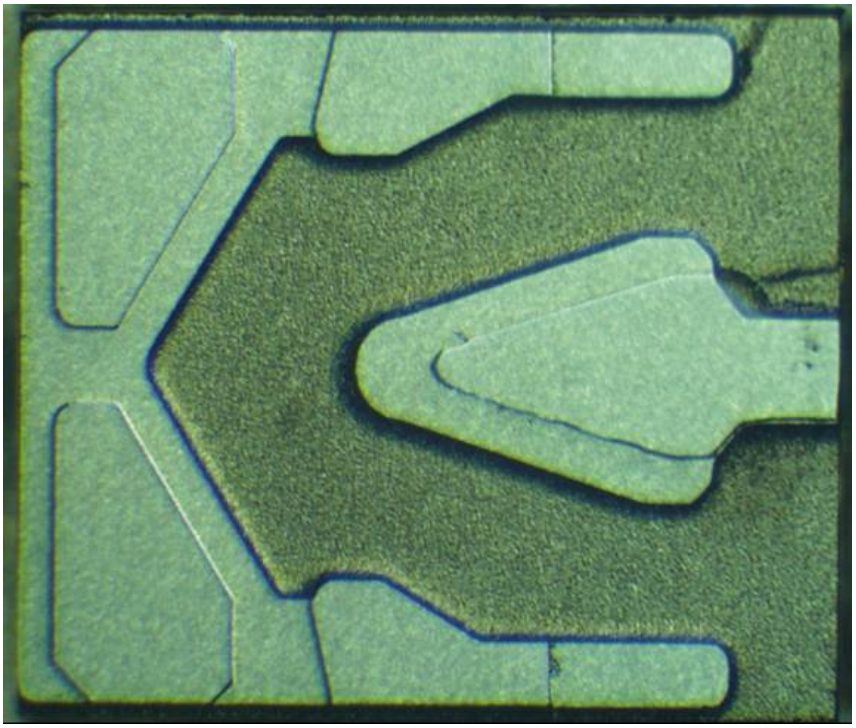
Fig. 8. Experimental lube contamination for CML/INSIC femto slider

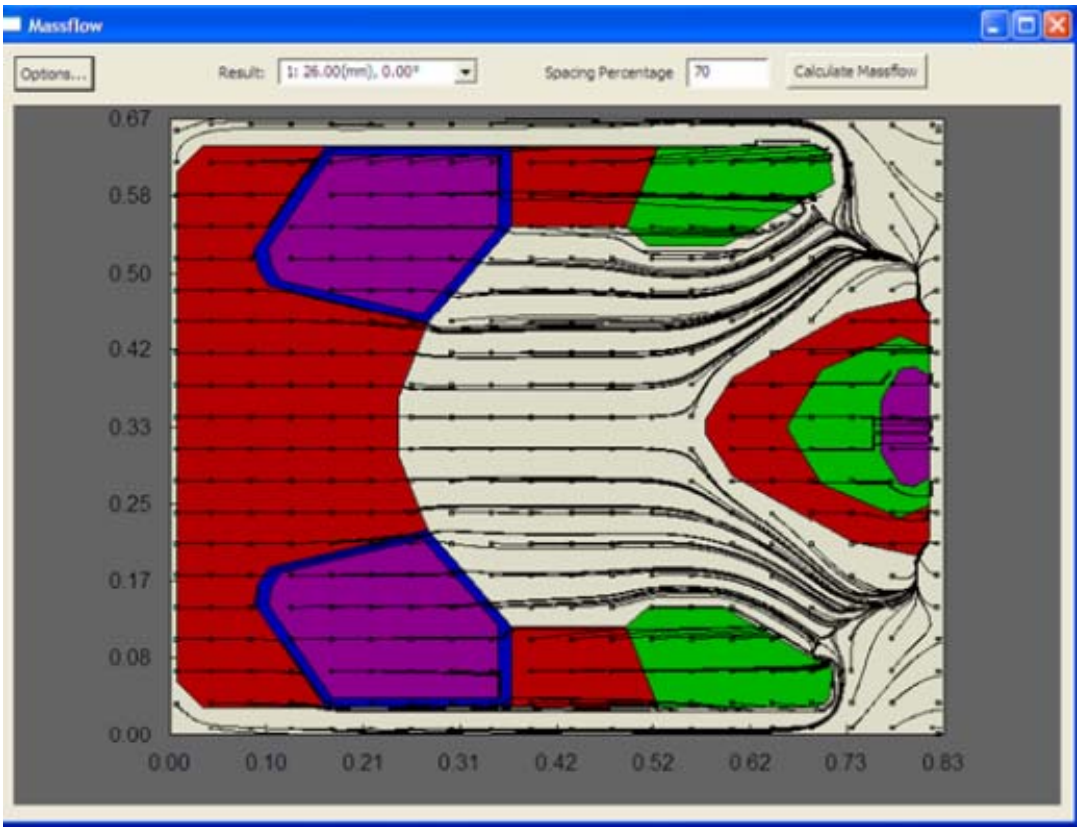

Fig. 9. Mass flow streamlines for DSI/INSIC femto slider

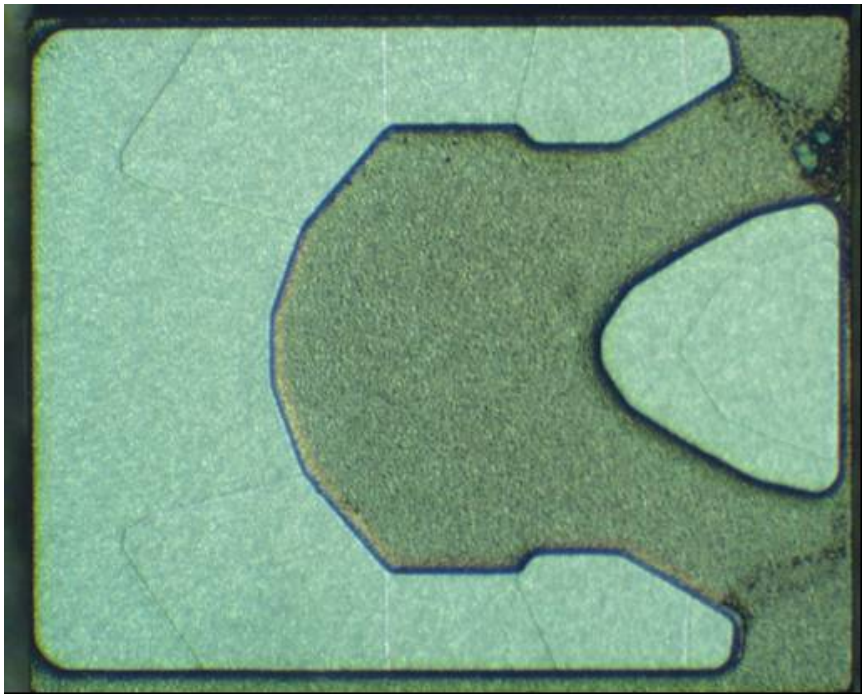
Fig. 10. Experimental lube contamination for DSI/INSIC femto slider