Advances in CML Air Bearing Design Software

Ling Huang and David B. Bogy

Computer Mechanics Laboratory
Dept. of Mechanical Engineering
University of California
Berkeley, CA 94720

Abstract

The computer simulation of the head-disk interface (HDI) has been an indispensable tool for air bearing design. CML has a long history of working in this area, and several Ph.D. students have made important contributions in this field. Based on this work CML has developed, used, and distributed air bearing design tools, which greatly improve the efficiency of slider design. Recently, several improvements have been achieved in CMLAir32 software, such as:

1) Several new tools have been implemented in the MultiModule function to allow more flexible combinations of input parameters based on predefined distributions.

2) In Quick4 "bugs" in the code for implementing different slip models were removed. All slip models now work correctly in Quick4.

3) Several new slip modes were implemented in both Quick4 and Quick5.

4) In order to achieve accurate and optimal results for ultra-low FH slider simulation, we incorporated intermolecular forces in the CML software for flying attitude simulation.

5) A new function, Gap Extrapolation, was developed for slider flying attitude measurement.

6) Geometric modeling and parametric constraints in slider design, simulation, and optimization were also incorporated in the program

1. Introduction

In modern hard disk drive, data are written and stored in the form of magnetic bits on stacks of disks. The disks are mounted on a rotating spindle, which usually rotates at a speed from a few thousands to tens of thousands revolutions per minute. The accessing of the magnetic bits is achieved through a read/write transducer attached to a small air bearing slider. The slider is mounted on the tip of a suspension arm. A position actuator to locate the magnetic bits on the disk controls the suspension arm. The Assembly of hard disk drive is shown in Fig.1.

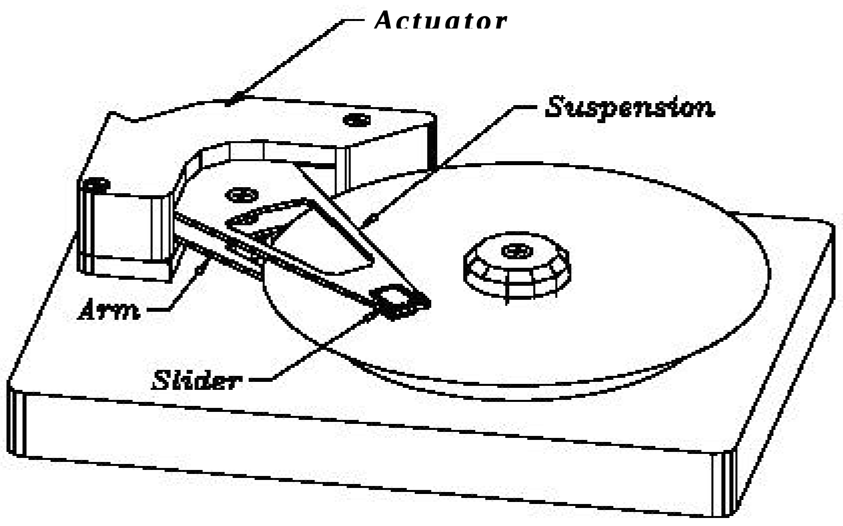


Fig.1 Assembly of hard disk drive

When the disk rotates at the designed speed, the slider flies above the disk because a thin air layer gets squeezed into the narrow space between the slider and the disk surface. The air bearing is created in the head disk interface (HDI) to lift the slider to a certain height to separate it from the disk surface to avoid disk wear. The slider’s flying attitude has a profound influence on the read back signals and consequently on the performance of the hard disk drives. It is determined by the balance of the forces and torques generated by the spring suspension, the air bearing, and other forces. The flying attitude goal is obtained by carefully designing the geometric shape of the slider’s air bearing surface (ABS). Simulation of the air bearing pressure and spacing accurately and efficiently becomes the key issue in the design of sliders. To improve the efficiency and shorten the design cycle, accurate and intelligent computer simulation software for the HDI has been an indispensable tool for air bearing design.

2. CML Air Bearing Software

The objective of the HDI simulation is to predict the steady state and dynamic flying attitudes for a given slider air bearing design and disk surface topology by solving the lubrication equation together with the equations of motion of the slider. It involves two steps to simulate the air-bearing problem accurately. The first step is to model the problem accurately and obtain a physically justified governing equation. For the air bearing problem it is a modified Reynolds lubrication equation. The second one is to develop numerical methods that solve the Reynolds equation accurately and efficiently.

CML has a long history of working in this area, and several Ph.D. students have made important contributions in this field. Based on this work CML has developed, used, and distributed air bearing design tools, which greatly improve the efficiency of slider design.

Ellis T. Cha (1993) and Sha Lu (1997) made prominent contributions to the CML Air Bearing Design Program. Quick300 and Quick4 are two important solvers based on their Ph.D. thesis work. These solvers implemented the multi grid control volume method on rectangle meshes to solve the generalize Reynolds equation. They also implemented an adaptive grid method based on the pressure gradient field, which greatly facilitates grid generation. The implementation of a full approximation storage and full multi grid method dramatically improved the solver efficiency. Parallel with these works, CML also developed graphic user interfaces in both Matlab and Windows to simplify the user's pre- and post- processing. With these tools the user can easily and accurately simulate all of the main air bearing surface features, including multiple recess heights, shaped rails, and arbitrary wall profiles, etc. Quick4, the solver developed for sub-25nm slider seven years ago, still can produce accurate simulation results for sub-10nm sliders in today's hard disk drives. Among such sliders, the NSIC 7nm slider was designed using Quick4.

Lin Wu's (2001) Ph.D. work led to the development of Quick5. This solver uses a triangular mesh for ultra-low flying height slider static simulation. In his thesis work, he also derived two new lubrication equations free of contact pressure singularities. Also he investigated the effect of the intermolecular force, especially the van der Waals force, on the flying attitude of new ultra-low flying sliders, and incorporated the intermolecular force into the balance equation. Based on these developments, he implemented unstructured adaptive triangular mesh generation techniques to handle complex slider air bearing simulation. He employed different refinement and adaptation techniques to generate several levels of good quality mesh over sliders with complex rail shapes. The overall mesh generation procedure offers great flexibility and control over the quality and distribution of the generated mesh.

CMLAir32, currently in version 6, is the CML distribution of the computer-aided air bearing design program for the PC. It consists of a graphic user interface in 32-bit MS Windows and a set of Air Bearing Design Solvers. The current distribution contains three solvers: Quick4, Quick5, and Dyn4. These solvers are actually separate programs, but they can all be used in an integrated environment with the Air Bearing Design Program. Quick4 is the CML Air Bearing Rectangular Grid Solver, which is written in Fortran77 and uses a rectangular mesh to solve the

Reynolds equation. Quick5 is the CML Air Bearing Triangular Grid Solver, which is written in C++ and uses a triangular mesh to solve Reynolds equation. Dyn4 is the CML Air Bearing Dynamic Simulator, which is written in Fortran77 and uses a rectangular mesh for dynamic simulation. A solver that uses the triangular mesh for dynamic simulation is under development. All solvers have been compiled as 32-bit Windows console applications using Digital Visual Fortran and Microsoft Visual C++ 5.0.

Strictly speaking, you do not need the Windows interface in order to use the CML simulation software. All functions of the solvers can be accessed using ASCII input and output files. You will find, however, that the Windows interface vastly simplifies the air bearing design and analysis process. The Windows interface can be run on any Win98/2000/XP PC system. The interface allows for visual pre- and post- processing of air bearing designs. Features include point and click rail design, extensive error checking, visual grid construction, 2-d and 3-d post-processing, local and remote solver capability, VMS and NT remote support, and many others. For detailed description of the program, see the user manual included in the distribution.

Recently, several improvements have been achieved in the CMLAir32 software, as follows:

1) Several new tools were implemented in the MultiModule function to enable more flexible combinations of input parameters based on predefined distributions.

2) In Quick4 “bugs” in the code to implement the different slip models were removed. All slip models now work correctly in Quick4.

3) Several new slip modes were implemented in both Quick4 and Quick5.

4) In order to achieve accurate and optimal results for ultra-low FH slider simulation, we incorporated intermolecular forces in the CML software for flying attitude simulation.

5) A new function, Gap Extrapolation, was developed for slider flying attitude measurement.

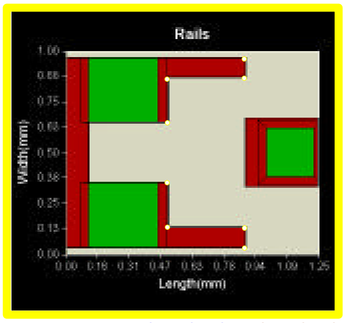
6) Geometric modeling and parametric constraints in slider design, simulation, and optimization were also incorporated in the program.

3. MultiModule

The MultiModule is an extension of the multiple run capabilities to the CML Air Bearing design program. It allows users to generate customized batch run jobs and interpolate results over a user definable search space. The main uses for this capability are optimization and manufacturing tolerance calculations.

For example, after the design of a slider, it needs to be fabricated. In the manufacturing process, errors and deviations from the design dimensions will be introduced into the resulting slider. That is, the slider you get isn't exactly what you designed. There will be some variances between the design parameters and the manufactured parameters. Those variances, even if very small, may have important effects on the slider's flying performance. So we need tools to simulate those variances, and help the user determine the relative robustness of the design with respect to manufacturing. MultiModule is a tool for simulating those variances. In this module you can define tolerances on any slider geometry in terms of deviation ranges for selected parameters, including slider width, length, crown, rail X, rail Y, rail recess, etc. Each parameter type corresponds to a slider geometry defined in the General window and Rail window. The deviation range is the upper and lower search space boundaries for this parameter. The user enters the amount of variation relative to the base, and can select predefined sample (search) type and frequency. MultiModule will automatically and optimally generate a large number of cases for the solver to determine the manufacturing yield and optimization.

An example is shown in Fig. 2, where we use a design similar to the IBM Travelstar slider and define deviation ranges on two rail points. We select a cross sample type and set the sample frequency to be 8, then MultiModule will create 64 cases optimally distributed in the sample space, and automatically run the solvers for calculation.



	Point 1 (mm)	Point 2 (mm)
Min	0.5	0.825
Max	0.6	0.925
Cases	8	8
Total Cases	64	

Point 1	Point 2	Nominal FH	Pitch	Roll	Min
0.000516667	0.000908333	15.590022	82.26419	-4.8123112	23.7728
0.000533333	0.000908333	15.651913	82.664669	-4.8264496	23.8426
...	...	...	...	...	...
0.000583333	0.000908333	15.89342	83.829382	-4.8980198	24.1035
[0.0006]	[0.000925]	15.130019	[84.654044]	-4.9704237	23.3512
<0.0005>	[0.000925]	<14.692105>	82.339938	[-4.7452252]	<22.8848>
0.000516667	[0.000925]	14.74126	82.726651	-4.7657783	22.9408
0.000533333	[0.000925]	14.791456	83.135458	-4.7637012	23.001
0.00055	[0.000925]	14.851102	83.545786	-4.794952	23.0668
0.000566667	[0.000925]	14.92617	83.944535	-4.8183974	23.1486
0.0005	0.000825	14.692105	79.19793	-5.2688636	22.8848
0.0006	0.000925	21.295749	84.654044	-4.7452252	29.3995
0.000549219	0.000825651	18.31066908	81.65962352	-5.05088178	26.45034426

a) Deviation ranges and sampling frequency   b) Cases generated for solver calculation

Fig. 2 Input and output of MultiModule works

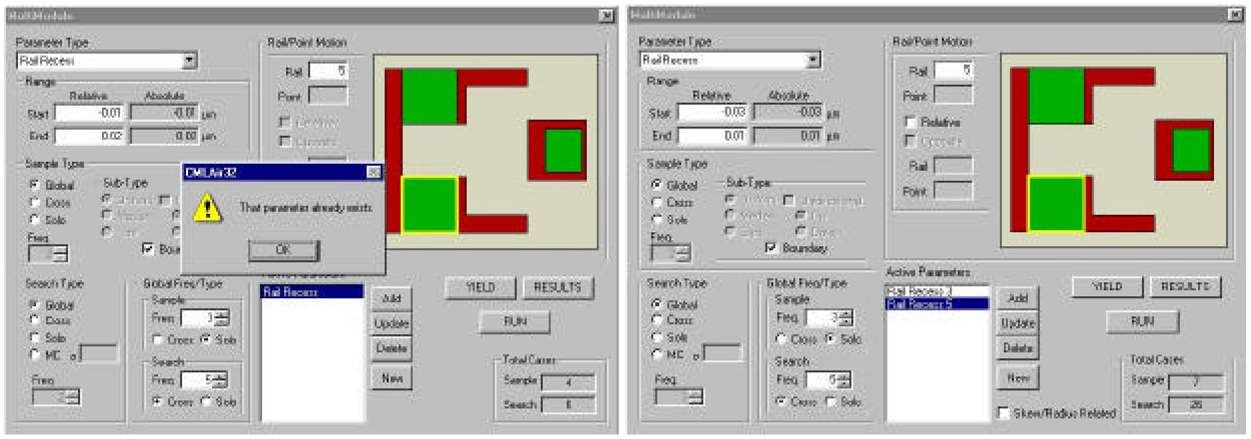
Each MultiModule run is defined and separated into sample space and search space. Sample space defines the accurate calculation runs by the solver. It is the space that is spanned by all cases that are actually sent to the solver and calculated. Search space is the space that is spanned by all cases for which interpolated results are sought. Search space results are generated through linear interpolation of sampled results. It is important to have a strong understanding of the relationship between sample and search space in order to maximize the efficiency of MultiModule. You should first do the calculation on the sample space. It defines the nodes for which results are generated using an air bearing solver (Quick4 or Quick5). Those results will be automatically stored in the program, and you can define and conduct calculation in the search space based on those results. The choice of sample space is of utmost importance in determining the efficiency of a MultiModule run. The time required for a run is approximately directly proportional to the number of nodes in the sample space. Denser sample spaces result in more accurate search spaces, but take a longer time. On the contrary, time and CPU cost of calculation on the search space is low. It only involves linear interpolation, with no additional solver calculation.

Next we achieve several improvements in the MultiModule from its original design and implication, including: 1) Variance definition for Multi Rail Recess. 2) Relative variance definition for Rail. Those new improvements allow all combinations of input parameters based on arbitrary distributions.

3. 1 Variation for Multi-Rail Recess

The slider is manufactured by etching processes, especially the different rails and their recesses. Etching is a chemical process that removes material level by level. In a multi-rail and multi-recess slider, this process involves several steps. Different rail recesses are gradually formed from small to large, step by step. After manufacturing, each rail recess will have its own independent tolerance and deviation.

Corresponding to this reality, we should have variance definitions for different rails to simulate manufacturing tolerance on the different rail recesses, but in the previous version, only one tolerance could be defined for all rail recesses. As show in Fig. 3, one could define the tolerance on the rail recess for rail 3. But rail 5 also can have its own tolerance, which is different from that of rail 3. When defining the deviation for rail 5, the program would pop up an error window, and would not allow you to define it any more. This feature greatly limits its ability to simulate the manufacturing tolerance. In the new version, this limitation is removed, and the user can define specific variance for every rail recess.



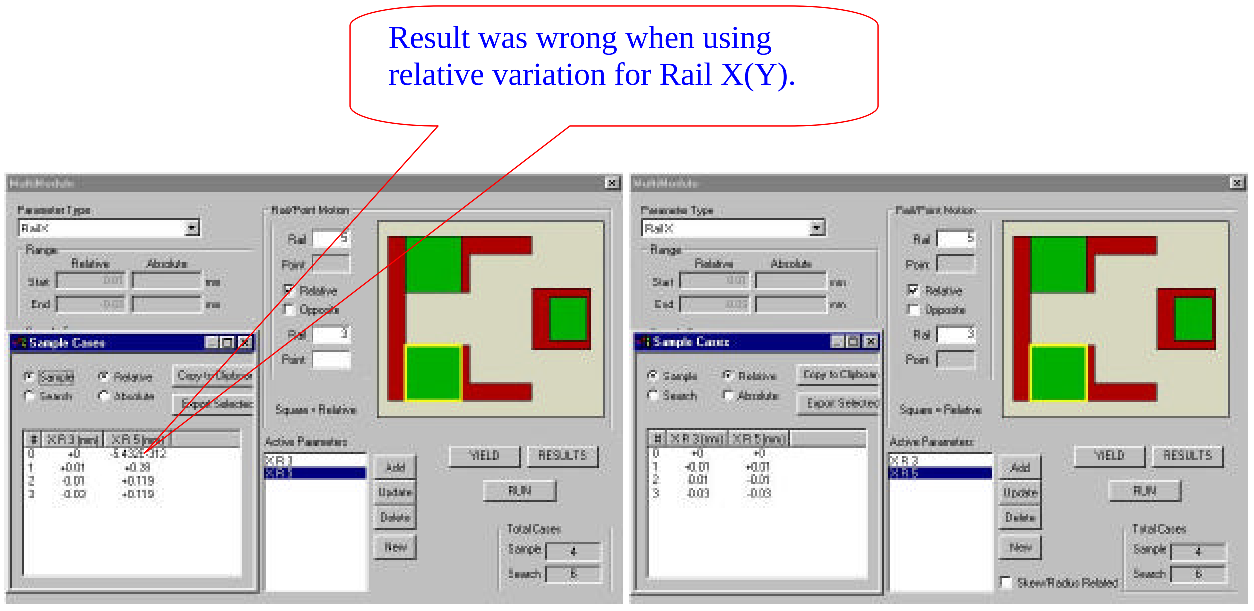
a) Previous version b) Current version

Fig. 3 MultiModule parameter defining: each rail recess can have its own variation.

3.2 Relative variance definition for Rail

Users can define the motion of a Rail/Point. If the parameter is of type Rail or Rail Point, the rail and/or point numbers must be specified. The user can specify them either by entering rail and point index numbers, or by selecting them in the plot area on the right. With a selection in the plot area, the actual rail and point index values for the parameter will automatically change. When working in the plot area, one can select rails and points with the mouse in the same manner as one selects rails and points in the Rails Window. When the parameter is of type Rail, one can select a rail by double clicking the left mouse key on the desired rail. If you use the Rail Point parameter type, you should first select a rail, and then select a point within this rail by right clicking on the point. All the locations are plotted in the chart to the right. The rail to be moved is outlined in yellow. The point to be moved is outlined with a circle.

In the manufacturing procedure, some of the components on the slider are fabricated simultaneously using the same process. Those components should have the same tolerance and deviation. So it is convenient for the user to define deviations relative to each other. This can be done by checking the *Relative* box, and the parameter will follow the motion of the relative point. If the Relative and Opposite boxes are checked, the parameter's motion will be exactly opposite that of the relative point. Note that there can be multiple levels of relativity and motion. For instance, Rail A can be relative to Point B that can be relative to Point C. In the previous version, the result was incorrect when using the relative deviation. The new version removes this bug, and the user can conveniently define the tolerances relative to each other.



Previous version, Sample Cases:

#	X R 3 [mm]	X R 5 [mm]
0	+0	-5.4321-312
1	+0.01	+0.39
2	-0.01	+0.119
3	-0.02	+0.119

Current version, Sample Cases:

#	X R 3 [mm]	X R 5 [mm]
0	+0	+0
1	+0.01	+0.01
2	0.01	-0.01
3	-0.03	-0.03

a) Previous version   b) Current version

Fig. 4 Rail X (Y) can vary relatively to each other

4. Correlation of Radial Position and Skew Angle

Radius position and skew angle change as the slider moves across the disk, for example, when the slider moves in the track seeking motion to locate a track on the disk. One needs to input the values for radius position and skew angle for air bearing simulation. In CMLAir32 they are input as a pair in the run setup interface. In the previous release the user could define Radius/Skew pairs point by point. Run Setup Dialog is the window for setting up the operating conditions for a static simulation, including Radial Pos./Skew, RPMS, Altitude, MFP, etc. For the input of Radial Position and skew angle, the user could use the point-by-point method to define them. For Quick4, if multiple pairs are specified, the simulation will be run at each of these pairs. Quick5 does not support multiple radius/skew pairs.

Actually the radial position and the skew angle have a correlation. Fig.5 a) shows the assembly of the actuator, arm, slider and disk. Radial position is the distance from the geometrical center of the slider to the disk center. Skew angle is the angle between the track tangent line and the longitudinal direction of the slider. Two issues contribute the skew angle of a slider: initial assembly angle and the rotation of the arm. The slider is fixed by a suspension on the arm of the actuator. It may or may not be assembled with an initial skew angle. When the arm rotates, the slider moves toward or away from the center. The radial position is changed with the arm movement, and the slider can move to any track on the disk to read data. The skew angle is also changed, and the air is directed toward different components of the slider. A triangle is formed by the radius, arm length and distance between the pivot of the arm and the center of the disk. Radial position and skew angle are correlated in this triangle. Their relationship can be formulated as follows:

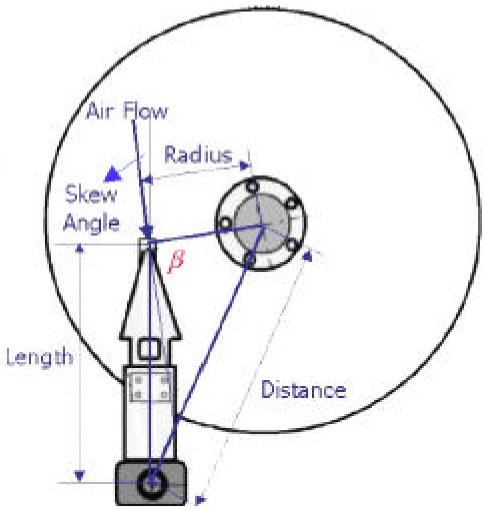
$$Skewangle = b - \frac{p}{2}, \qquad b = \frac{Radius^2 + Length^2 - Distance^2}{2 * Length * Radius} \tag{1}$$

Where b is the angle between the arm line and radial line.

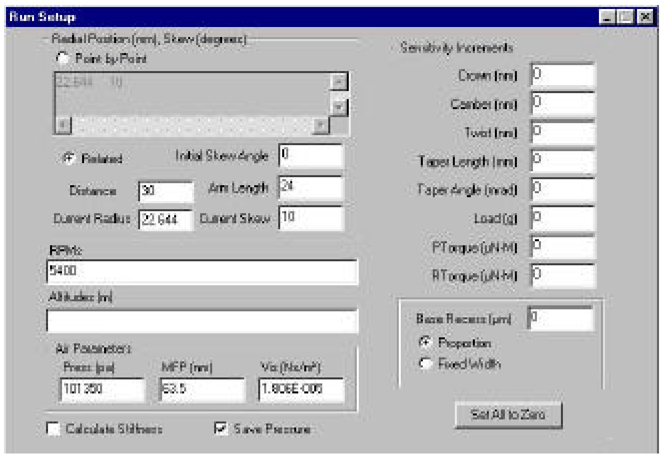
Positive skew indicates that air flows from the outer leading edge to the inner trailing edge. This conforms with the IDEMA standard.

With this formula, the user can define the radial position and skew angle as related. As shown in Fig. 5 b), the interface for Run Setup, a new area, Related, is added under the Point by Point input area. Now one has two choices for defining the radial position and skew angle: Point by

Point or Related. If you use the Point by Point method, you can input the values in the same way as in the previous version, and you must specify at least one Radius/Skew pair. When you use the Related feature, you should specify several parameters, as shown below:



a) Sketch of arm and disk assembly



Radial Position (mm), Skew (degrees)

( ) Point by Point

22.644 10

(●) Related

Initial Skew Angle	0		
Distance	30	Arm Length	24
Current Radius	22.644	Current Skew	10

RPMs: 5400

Altitudes (m):

Air Parameters:

Press (pa)	MFP (nm)	Vis (Ns/m²)
101 350	63.5	1.806E-005

[ ] Calculate Stiffness [✓] Save Pressure

Sensitivity Increments

Crown (nm)	0
Camber (nm)	0
Twist (nm)	0
Taper Length (mm)	0
Taper Angle (mrad)	0
Load (g)	0
PTorque (μN-M)	0
RTorque (μN-M)	0

Base Recess (μm) 0

(●) Proportion
( ) Fixed Width

b) New interface for radial position and skew angle

Fig. 5 Correlation between radial position and skew angle

Initial Skew Angle: is the angle between the centerline of the slider and the centerline of the arm. It is pre-determined at assembly time.

Distance: is the distance (mm) from the pivot of the arm to the center of the disk.

Arm Length: is the length of the arm.

Current Radius: is the current radial position of the slider.

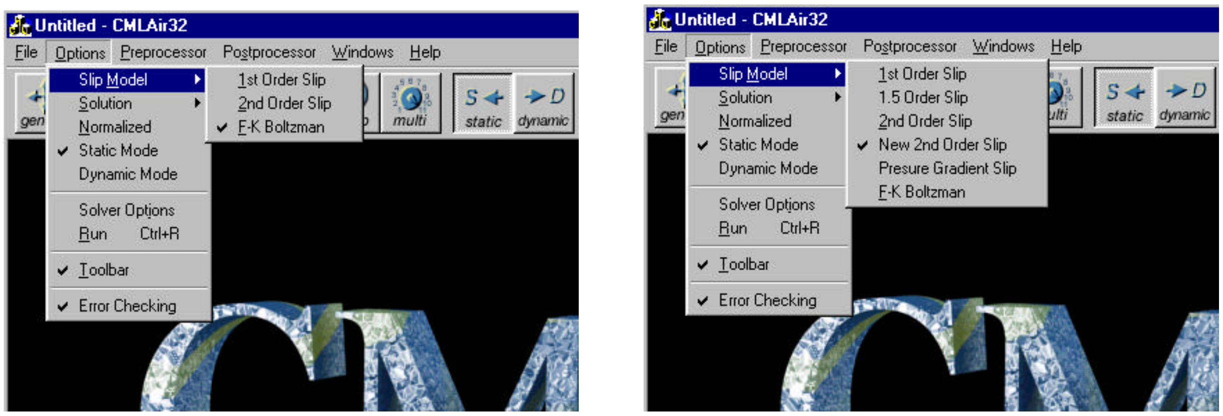
Current Skew: is the current skew angle of the slider.

Current Radius and Current Skew are determined by each other. When you modify either one, the program will automatically calculate the other one using equation (1).

5. Slip models

5.1 Introduction

In the Reynolds equation, in order for the no-slip condition to apply, the film thickness should be large compared to the mean free path of the air, which is around 64 nm in standard conditions. However the slider-disk separation in today's drives is less than 10 nm and is therefore much smaller than the mean free path. So, the rarefaction effect has to be taken into account. Various slip correction models have been introduced to handle the slippage at the boundaries and the rarefaction effect. In the previously released version of the CMLAir32 program, three slip models were available, including the first order slip, second order slip, and F-K model, as shown in Fig. 6 a).



a) Previous version b) Current version

Fig. 6 Interface for slip models

5.2 "Bugs" in Quick4 slip model implementation

In the solver Quick4 there were some bugs in the codes implementing the slip model, so that only the F-k model could work in Quick4. The other two slip models always result in a divergent computation. After analyzing the Fortran77 code of Quick4, we finally found and removed those bugs. Now the three slip models work correctly in Quick4.

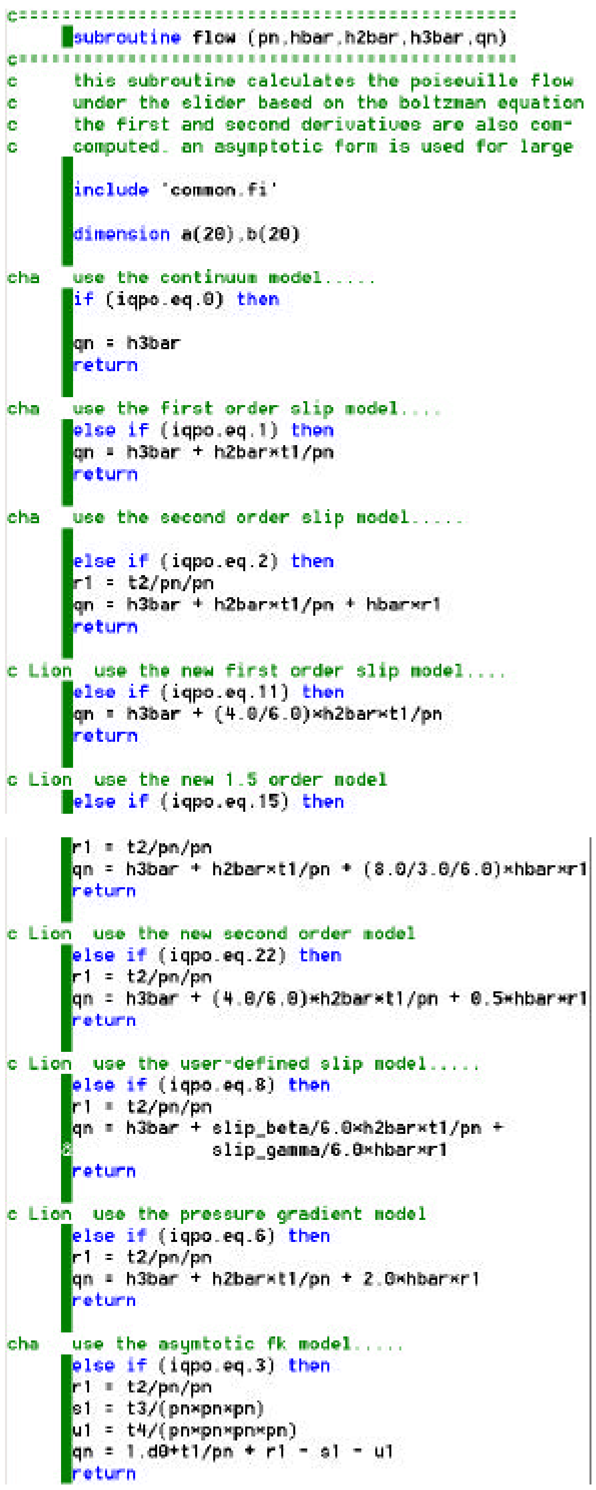
The “bugs” were caused by an improper use of the variable feature in the Fortran programming language, i.e. using the same symbol for a local variable and a global variable.

5.3 New slip model

We implemented several new slip models in the solver. Lin Wu’s Ph.D. thesis pointed out some deficiencies with the current slip models. Both the FK model and the first order slip model predict unbounded pressure fields at a contact point, which means they can not predict the pressure correctly at very large Knudsen numbers. Li Wu (2001) examined the slip models to understand the cause of and to remove those kinds of singularities. He derived several new slip models for isothermal and compressible gas lubrication from a more physical point of view. As shown in Fig. 7, the new models have similar forms as the old counterparts but with different coefficients. The new second order slip model is in better agreement with the FK model than the old second order slip model at smaller inverse Knudsen number, in regimes where the FK model gives reasonable predictions.

Previous slip models	New slip models
First order	1.5 order
$Q = 1 + 6a\dfrac{K_n}{PH}$	$Q = 1 + 6a\dfrac{K_n}{PH} + \dfrac{8}{3}\left(\dfrac{K_n}{PH}\right)^2$
Second order	Second order
$Q = 1 + 6a\dfrac{K_n}{PH} + 6a\left(\dfrac{K_n}{PH}\right)^2$	$Q = 1 + 4a\dfrac{K_n}{PH} + 3\left(\dfrac{K_n}{PH}\right)^2$
F-k model	Pressure gradient model
$Q = 1 + f\,(\dfrac{K_n}{PH})$	$Q = 1 + 6a\dfrac{K_n}{PH} + 12a\left(\dfrac{K_n}{PH}\right)^2$

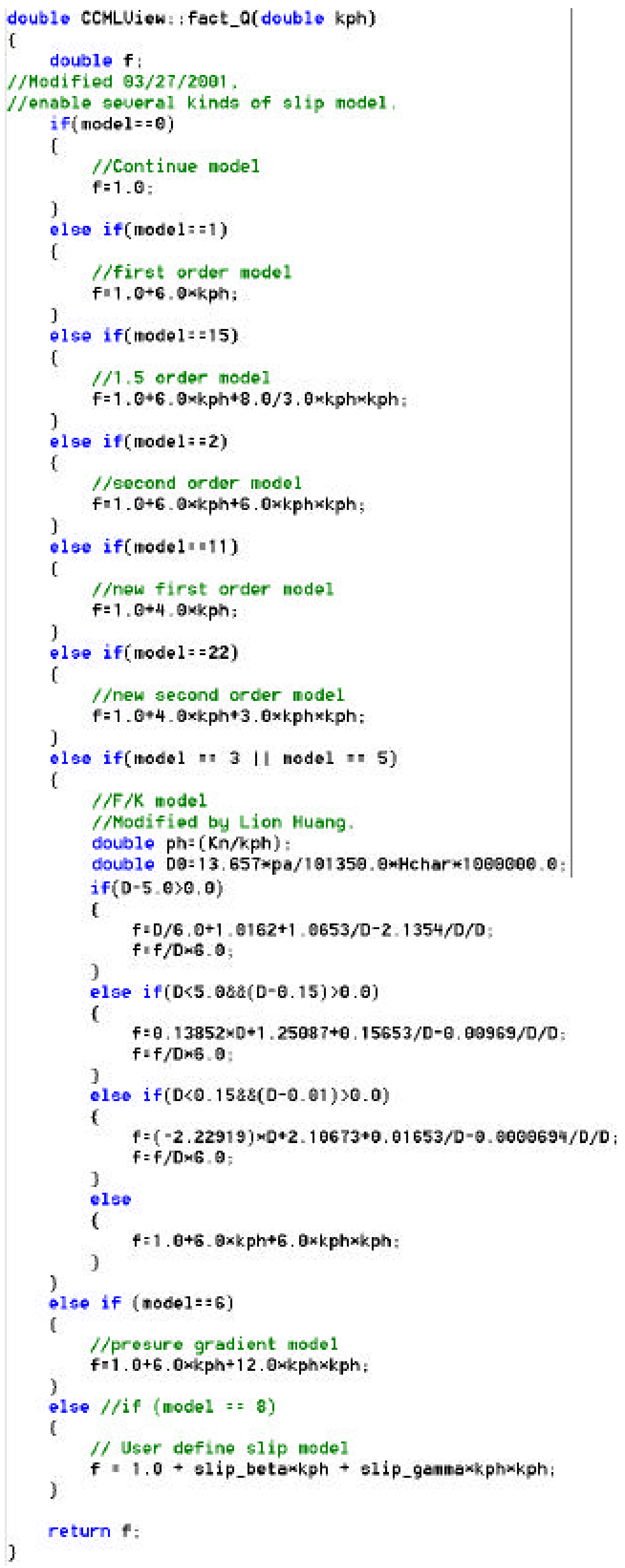
Fig. 7 Formulae for slip models



```
c==============================================
      subroutine flow (pn,hbar,h2bar,h3bar,qn)
c==============================================
c     this subroutine calculates the poiseuille flow
c     under the slider based on the boltzman equation
c     the first and second derivatives are also com-
c     computed. an asymptotic form is used for large

      include 'common.fi'

      dimension a(20),b(20)

cha   use the continuum model.....
      if (iqpo.eq.0) then

      qn = h3bar
      return

cha   use the first order slip model....
      else if (iqpo.eq.1) then
      qn = h3bar + h2bar*t1/pn
      return

cha   use the second order slip model.....

      else if (iqpo.eq.2) then
      r1 = t2/pn/pn
      qn = h3bar + h2bar*t1/pn + hbar*r1
      return

c Lion  use the new first order slip model....
      else if (iqpo.eq.11) then
      qn = h3bar + (4.0/6.0)*h2bar*t1/pn
      return

c Lion  use the new 1.5 order model
      else if (iqpo.eq.15) then

      r1 = t2/pn/pn
      qn = h3bar + h2bar*t1/pn + (8.0/3.0/6.0)*hbar*r1
      return

c Lion  use the new second order model
      else if (iqpo.eq.22) then
      r1 = t2/pn/pn
      qn = h3bar + (4.0/6.0)*h2bar*t1/pn + 0.5*hbar*r1
      return

c Lion  use the user-defined slip model.....
      else if (iqpo.eq.8) then
      r1 = t2/pn/pn
      qn = h3bar + slip_beta/6.0*h2bar*t1/pn +
     &             slip_gamma/6.0*hbar*r1
      return

c Lion  use the pressure gradient model
      else if (iqpo.eq.6) then
      r1 = t2/pn/pn
      qn = h3bar + h2bar*t1/pn + 2.0*hbar*r1
      return

cha   use the asymtotic fk model.....
      else if (iqpo.eq.3) then
      r1 = t2/pn/pn
      s1 = t3/(pn*pn*pn)
      u1 = t4/(pn*pn*pn*pn)
      qn = 1.d0+t1/pn + r1 - s1 - u1
      return
```

…………..

a) Code for Quick4



```
double CCMLView::fact_Q(double kph)
{
    double f;
//Modified 03/27/2001.
//enable several kinds of slip model.
    if(model==0)
    {
        //Continue model
        f=1.0;
    }
    else if(model==1)
    {
        //first order model
        f=1.0+6.0*kph;
    }
    else if(model==15)
    {
        //1.5 order model
        f=1.0+6.0*kph+8.0/3.0*kph*kph;
    }
    else if(model==2)
    {
        //second order model
        f=1.0+6.0*kph+6.0*kph*kph;
    }
    else if(model==11)
    {
        //new first order model
        f=1.0+4.0*kph;
    }
    else if(model==22)
    {
        //new second order model
        f=1.0+4.0*kph+3.0*kph*kph;
    }
    else if(model == 3 || model == 5)
    {
        //F/K model
        //Modified by Lion Huang.
        double ph=(Kn/kph);
        double D0=13.657*pa/101350.0*Hchar*1000000.0;
        if(D-5.0>0.0)
        {
            f=D/6.0+1.0162+1.0653/D-2.1354/D/D;
            f=f/D*6.0;
        }
        else if(D<5.0&&(D-0.15)>0.0)
        {
            f=0.13852*D+1.25087+0.15653/D-0.00969/D/D;
            f=f/D*6.0;
        }
        else if(D<0.15&&(D-0.01)>0.0)
        {
            f=(-2.22919)*D+2.10673+0.01653/D-0.0000694/D/D;
            f=f/D*6.0;
        }
        else
        {
            f=1.0+6.0*kph+6.0*kph*kph;
        }
    }
    else if (model==6)
    {
        //presure gradient model
        f=1.0+6.0*kph+12.0*kph*kph;
    }
    else //if (model == 8)
    {
        // User define slip model
        f = 1.0 + slip_beta*kph + slip_gamma*kph*kph;
    }

    return f;
}
```

b) Code for Quick5

Fig. 8 Code implementation for slip models

Based on those formulas, we implemented and incorporated these new slip models in both Quick4 and Quick5. So in the new version we have six slip models for the users' choice, as shown in Fig. 6 b). The main code implementation is shown in Fig. 8. One neat thing about the slip models is that the different slip models have differences only in two coefficients in their formulas. We can use a common equation to represent all the slip models, as shown in follows:

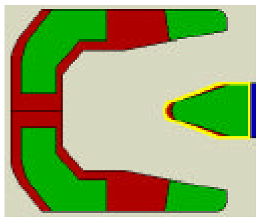
$$Q = 1 + b\,\frac{K_n}{PH} + g\left(\frac{K_n}{PH}\right)^2 \qquad (2)$$

where b and g are slip coefficients derived from physical phenomena of slippage and rarefaction effects at the boundaries.

This gives a uniform way to implement the different slip models. Only one subroutine is required for the calculation of all slip models.

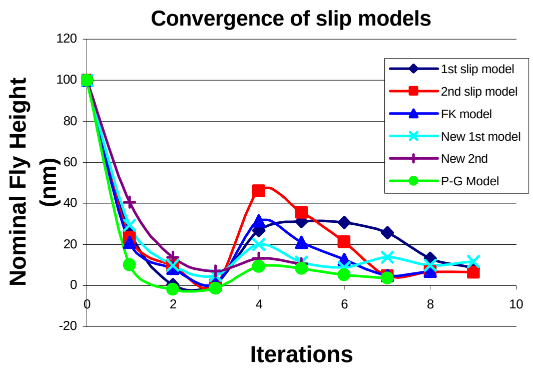
5.4 Comparison of different slip models

We performed some simulations to compare the results from the different slip models, and they are shown in Fig. 9 and Fig. 10. An NSIC 7nm slider design was used to demonstrate the convergence properties and the final results of these slip models. The slider flies around 7nm with a 1.5g force and zero moment pre-load from the suspension. The disk rotation speed is 7200 rpm. Fig. 9 a) shows the rail sketch of the slider and its initial conditions. We set its initial flying height at TEC to be 100nm, which is far from its final result flying height. Figs. 9 b) and c) show how the flying height and load error vary with the number of iterations. From these, it is seen that all slip modes have good convergence. Starting from an initial flying height of 100nm, the calculation of every slip model quickly reaches a converged value after 8 – 10 iterations. We also see that the load error quickly reduces to zero after one or two oscillation.

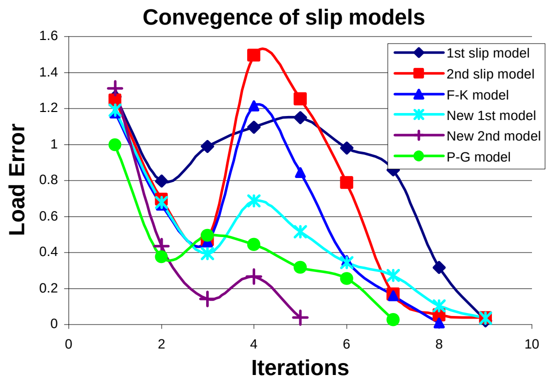


Initial conditions:
TEC: 100nm
Pitch: 150 *mrad*
Roll: 5 *mrad*

a) Slider and initial conditions



b) Flying height



c) Load Error

Fig. 9 Convergence property of different slip models

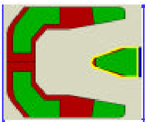
	1st order	New 1st order	1.5 order	2nd order	New 2nd order	Pressure Gradient	F-K
Fly Height	8.35	11.76	8.37	6.53	10.99	3.94	7.21
Pitch	168.5	187.8	177.6	174.5	194.3	168.5	168.0

Fig. 10 Final results of slip models

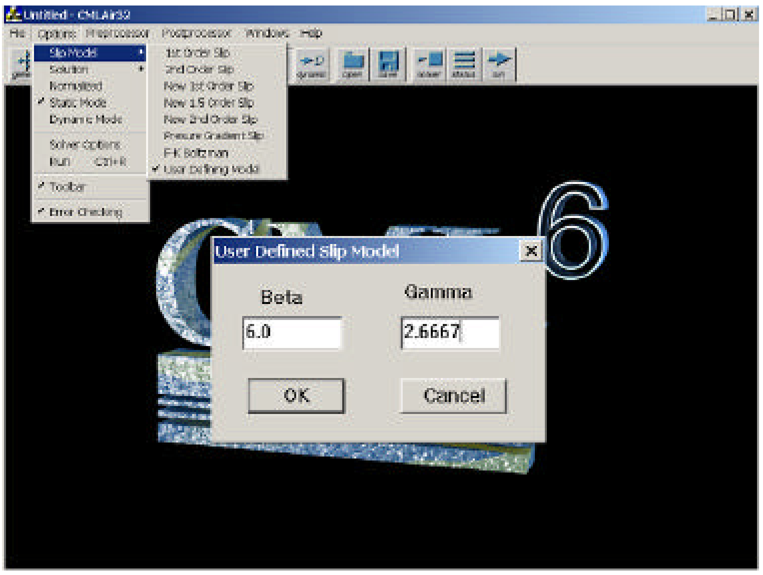
From Fig. 10, however, we see that the different slip models have different correction effects on the Reynolds equation. The final results have significant differences, in flying height the largest is even up to 3 times the smallest. The new slip models predict a larger flying eight and pitch angle, for example around 12 nm and 188 *mrad* for the new 1st order slip model for this slider. On the contrary, the Pressure Gradient model predicts a lower flying height and smaller pitch angle, for example around 4nm and 168 $mrad$ for this slider.

We conducted the same simulations and investigations using several other sliders. The Results show the same trend. Based on our experiences, we currently can give the following suggestion for using these slip models:

1) For sliders with flying height above 7 nm, the F-K model is preferred.
2) For sliders with flying height less than 7nm, the pressure gradient model is believed to be preferred.

5.5 User defined slip models

Even though we have six slip models, there has been a request for users to be able to design their own slip model to match simulation results with measured results for a particular design. To do this we only need to allow the user to choose the two coefficients b and g. By varying either b or g, we get different slip models. For example, when b = 6 and g = 6, we get the old second order slip model; when b = 4 and g = 0, we get the new first order slip model; etc.



Beta: 6.0

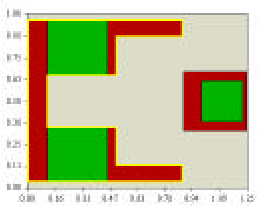
Gamma: 2.6667

Fig. 11 Interface for user defined slip model

The interface for this feature is very convenient, as shown in Fig. 11. Only two input parameters are needed. The user can go to the menu Options and click on the item Slip Model. A sub-menu pops up which allows you to select from different slip models. At the bottom of the Slip Model sub-menu, you find the User Defined Model. When you left click on this item, a small window pops up to enable the user to input b and g values. After entering the desired values you can just click on OK. These two values will then be saved in the program, and will later transfer to the solver for the slip model's calculation. When you use this feature the next time the program will automatically retrieve the b and g values used in the previous calculation, and put them back in the window when it pops up. This is very convenient for repeated computation: the user need not do anything if b and g are the same for next time calculation.

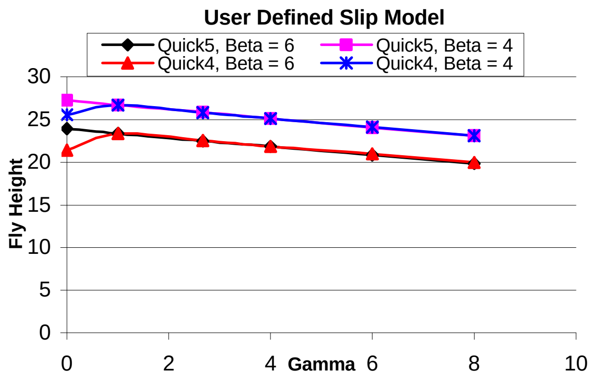
With this feature, we conducted a set of calculations to investigate how the flying attitude varies with the coefficients b and g. First we used the IBM Travelstar slider with a 2.5g force and zero moment pre-load from the suspension. The disk rotation speed was 4500 rpm and the slider flies around 20nm. Fig. 12 a) shows the rail design and the initial conditions for the slider. To our surprise, the calculation result showed a jump in the flying attitude between the first order model (where $g = 0$ and $b = 4$ or $b = 6$) and the other models, as shown in Fig. 12 b). Except for this jump, the results show a linear dependence on g, and both the flying height and the pitch angle decrease with increasing g. We also see a dependence on b: the flying height and the pitch angle decrease with increasing b.

We observe that the jump at $g = 0$ occurs for Quick4, but not for Quick5. It was found that different formulas and code implementation were used for those slip models. Specifically, the difference is in the value of the accommodation factor a in the slip models.

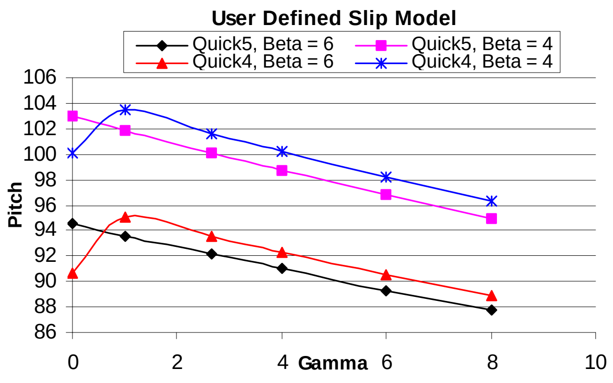


Initial conditions:
TEC: 16 nm
Pitch: 100 *mrad*
Roll: 1 *mrad*

a) Slider and initial conditions



b) Flying height variation with b and g



c) Pitch angle variation with b and g

Fig. 12 Initial calculation for user defined slip model

In Sha Lu's implementation in Quick4, the slip models were given as follows:

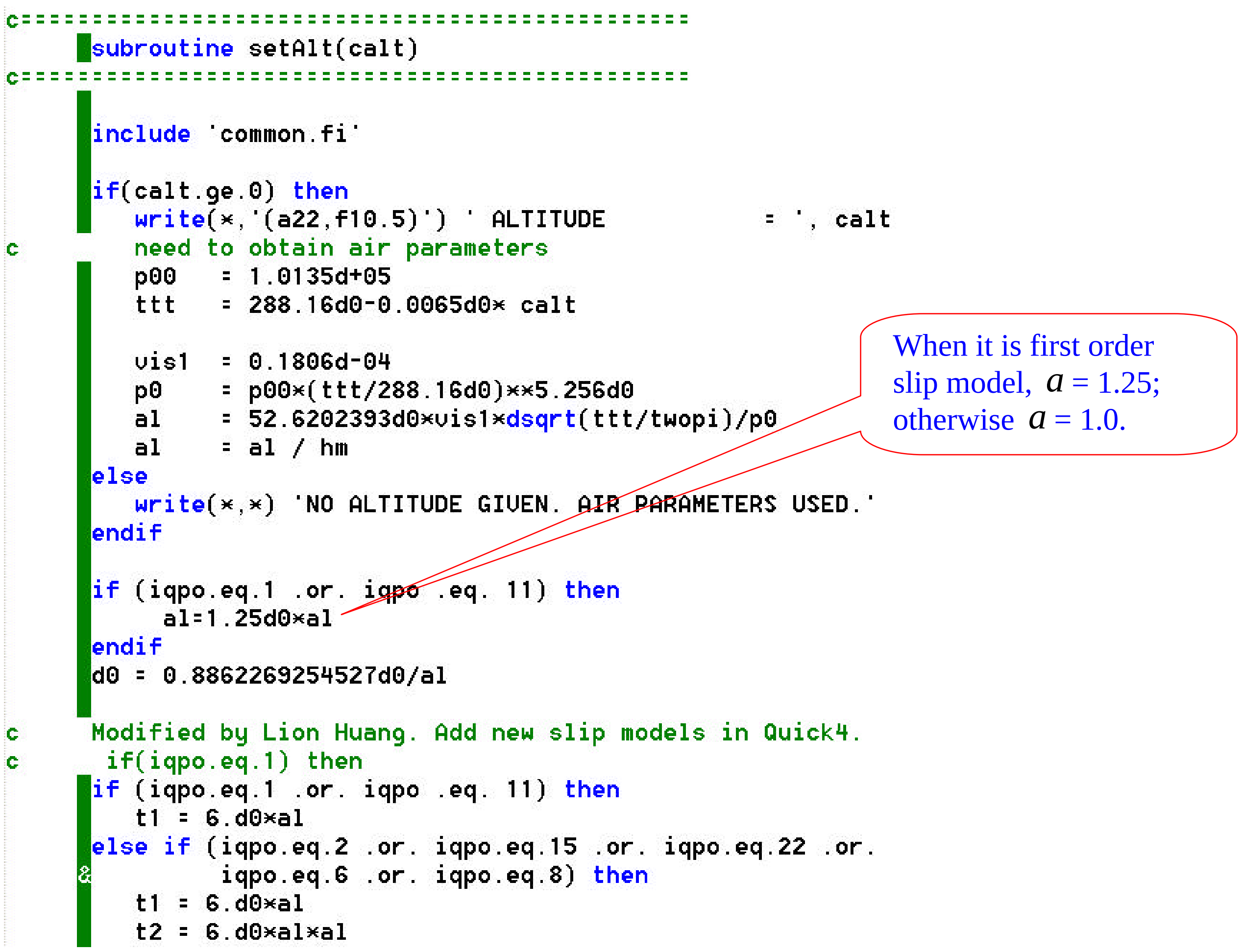
First Order Second Order F-K Model

$$Q = 1 + 6a\frac{K_n}{PH} \qquad Q = 1 + 6\frac{K_n}{PH} + 6\left(\frac{K_n}{PH}\right)^2 \qquad Q = 1 + f\,(\frac{K_n}{PH}) \qquad (3)$$

where $a = \dfrac{2-a}{a}$, a is the surface correction coefficient and a is the accommodation factor.

Sha Lu included the accommodation factor a only in the first order slip model. From the source code of Quick4, which was implemented by Sha Lu based on his Ph.D. thesis, it is seen that the first order slip model is implemented differently from the other models, as shown in Fig. 13. In the first order slip model a is multiplied by 1.25, which means that accommodation factor

a is set to be 0.9. In the other models it is set equal to 1.0. The result is that the prediction from Quick5 at g = 0 is about 1.25 times the value predicted by Quick4


When it is first order slip model, a = 1.25; otherwise a = 1.0.

```
c==============================================
      subroutine setAlt(calt)
c==============================================

      include 'common.fi'

      if(calt.ge.0) then
         write(*,'(a22,f10.5)') ' ALTITUDE            = ', calt
c        need to obtain air parameters
         p00    = 1.0135d+05
         ttt    = 288.16d0-0.0065d0* calt

         vis1   = 0.1806d-04
         p0     = p00*(ttt/288.16d0)**5.256d0
         al     = 52.6202393d0*vis1*dsqrt(ttt/twopi)/p0
         al     = al / hm
      else
         write(*,*) 'NO ALTITUDE GIVEN. AIR PARAMETERS USED.'
      endif

      if (iqpo.eq.1 .or. iqpo .eq. 11) then
           al=1.25d0*al
      endif
      d0 = 0.8862269254527d0/al

c     Modified by Lion Huang. Add new slip models in Quick4.
c      if(iqpo.eq.1) then
      if (iqpo.eq.1 .or. iqpo .eq. 11) then
         t1 = 6.d0*al
      else if (iqpo.eq.2 .or. iqpo.eq.15 .or. iqpo.eq.22 .or.
     &           iqpo.eq.6 .or. iqpo.eq.8) then
         t1 = 6.d0*al
         t2 = 6.d0*al*al
```

Fig. 13 Quick4 code to handle accommodation factor

In Quick5, Lin Wu included the accommodation factor a in all slip models, as shown below. Only in this way can the different slip models have consistent correction effects on Reynolds equation when a is not equal to 1. In Quick5 Lin Wu set a to be 1, which corresponds to no accommodation factor, given for the various slip models.

First Order | Second Order | F-K Model

$$Q = 1 + 6a\frac{K_n}{PH} \qquad Q = 1 + 6a\frac{K_n}{PH} + 6a\left(\frac{K_n}{PH}\right)^2 \qquad Q = 1 + f\left(\frac{K_n}{PH}\right) \qquad (4)$$

Weidong Huang (1998) made a detailed investigation of the accommodation factor a in his Ph.D. thesis. The accommodation coefficient expresses the tendency of the gas to accommodate to the state of the wall. Knudsen introduced the commonly used accommodation coefficient in 1911(Kennard, 1938). In terms of energy, the accommodation coefficient can be defined by:

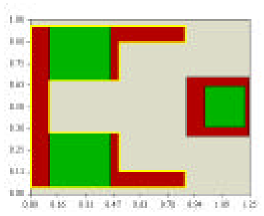
$$a = (E_i - E_r)/(E_i - E_w) \qquad (5)$$

where E_i denotes the energy rate per unit area of the incident molecules, E_r denotes the energy rate per unit area carried away by the reflected molecules and E_w denotes the energy rate per unit area that would be carried away by the reflected molecules in diffuse reflection at the wall temperature.

From the above study, we see that a should be a user input value. It varies with different surface materials, temperatures and surface roughness. For example, sliders and disks may have different accommodation: a is smaller for a smoother surface, so a for a slider is normally greater than that for the disk. For sliders and disks having different accommodation coefficients, Kang et al. (1997) obtained both the Poiseuille flow rate and the Couette flow rate coefficients with different accommodation coefficients ranging from 0.7 to 1.

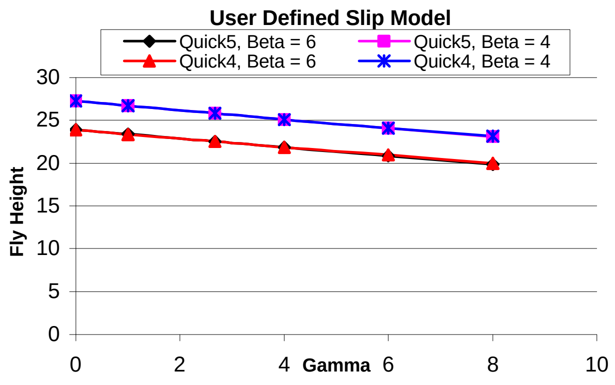
For air bearing problems for the head disk interface, there is very limited published information on the value of the accommodation coefficient to be used. Rettner (1997) obtained the accommodation coefficient $a = 0.95$ for N_2 colliding with a sputtered carbon overcoat, glass disk and an actual 3.5-inch disk surface. Due to the absence of a complete database of accommodation coefficients for sliders and disks in air, most slider designers use the full thermal accommodation, i.e. $a = 1$.

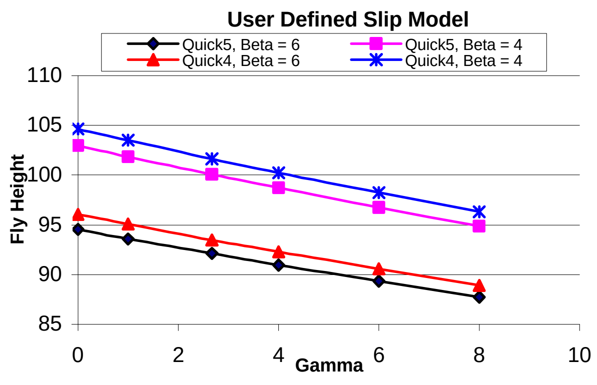
Based on the above studies, we assume that he accommodation coefficient a should appear in every slip model for consistency. It is acceptable to set $a = 1$ for slip models in slider design and simulation. After modifying the code in Quick4, we repeated the flying height simulation for several sliders, and the results are shown in Fig. 14 and Fig. 15. Fig. 14 shows the results for the slider used in the initial calculation. The result agrees with to the previous observation: only by setting a = 1 in first order slip model do we get a linear relationship between the flying attitude and b ,g in Quick4, as well as making the results from Quick4 consistent with those from Quick5. The second slider used was the NSIC slider that flies around 7nm. We again see almost the same relationship between flying attitude and b ,g .



Initial conditions:
TEC: 16 nm
Pitch: 100 urad
Roll: 1 urad

a) Slider and initial conditions

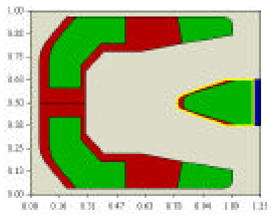




b) Flying height variation with b and g

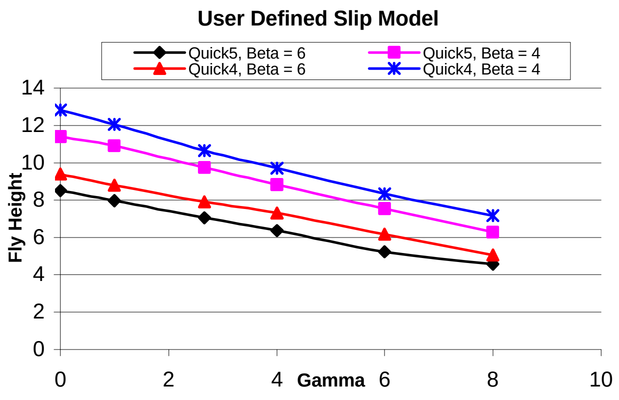
c) Pitch angle variation with b and g

Fig. 14 Result 1 for user defined slip model

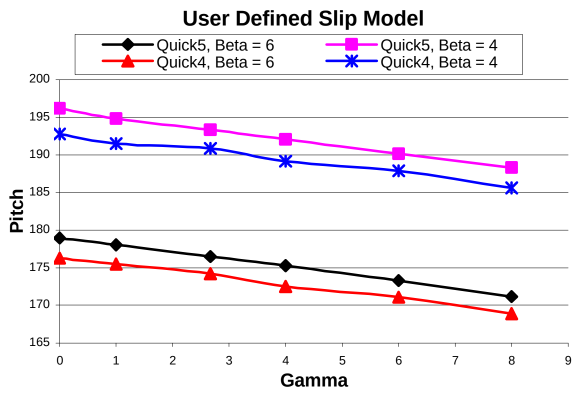


Initial conditions:
TEC: 7nm
Pitch: 150 urad
Roll: 5 urad

a) Slider and initial conditions



b) Flying height variation with *b* and *g*



c) Pitch angle variation with *b* and *g*

Fig. 15 Result 2 for user defined slip model

6. Implementation of intermolecular force

In prior generations of hard disk drives, the sliders flew more than 20nm above the disk. In such situations the flying attitude of the slider was determined by the balance of force and moments contributed from the preload of the suspension and the air bearing pressure and shear force. Most studies of air bearing simulations are concerned with the modifications of the Reynolds lubrication equation in order to get reasonably accurate predictions of the air bearing pressure field under rarefied conditions.

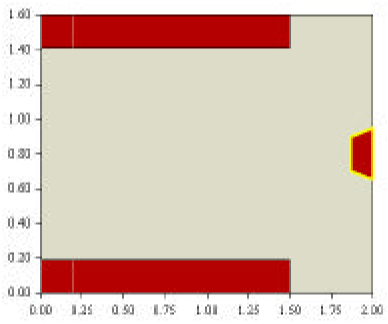
When the flying height of the slider becomes less to 5nm, intermolecular forces become increasingly more important and may no longer be negligible. For a slider with a flying height around 0.4 to 5 nm, the long-range attractive van der Waals force is the dominant one among these forces. It is contributed mostly by the dispersion force between two atoms, which is induced by the fluctuation in their electric dipoles. When two atoms are brought close enough, they start to experience the existence of each other in the form of intermolecular forces. At the beginning, it is an attraction force, and its strength increases with decreasing distance until a maximum point is reached, then it decreases with decreasing distance. When the distance is reduced further, the force becomes repulsive and increasingly stronger. Lin Wu (2000) derived formulas for intermolecular force acting on a triangular surface element as follows:

$$W_v = -\frac{A}{12\pi}\iint_{\Delta}\frac{dxdy}{h(x,y)^2} + \frac{B}{360\pi}\iint_{\Delta}\frac{dxdy}{h(x,y)^8} \tag{6}$$

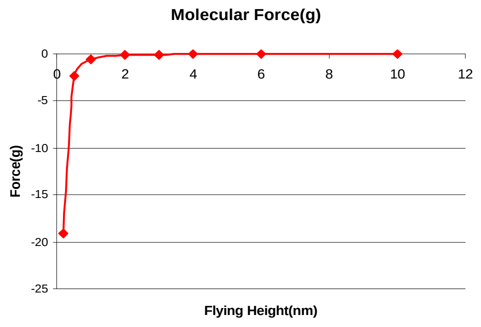
$$F_v = \frac{A}{6\pi}\iint_{\Delta}\frac{dxdy}{h(x,y)^3} - \frac{B}{45\pi}\iint_{\Delta}\frac{dxdy}{h(x,y)^9} \tag{7}$$

where A is the Hamaker constant, with typical value of about $10^{-19}\,J$. The typical value for constant B can be $10^{-76}\,Jm^6$.

We incorporated these forces in the solver in the force and moment balance for the flying attitude calculation. Fig. 16 shows a simulation result for a tripad slider with 3.5g force and zero moment pre-load from the suspension. Fig. 16 b) shows the van der Waals force as a function of the flying height at a fixed pitch angle of 150 $mrad$ and zero roll angle. Fig. 16 c) shows the results for the flying attitude calculation with and without the intermolecular forces included. The flying height is slightly less with theses forces. These results are in agreement with those of Lin Wu in his Ph.D. thesis.



a) Slider used



b) The attractive van der Waals force

	Fly Height	Pitch
W/O Mole. Force	5.82	165.3
W Mole. Force	5.68	165.5

c) The flying attitude w/o and with intermolecular force

Fig. 16 The effect of intermolecular force

7. Gap Extrapolation

Gap Extrapolation was developed to aid the experimental measurement of slider flying attitude.

The next generation hard disk drive with an areal density in the range of $1 TB/in^2$ is under development. For the transducer to read data from and write data at such a high areal density, the slider's flying height (FH) must be sub-5 nm. Head/disk interface stability and head/disk spacing variation are becoming of more concern. In this situation experimental measurement and analysis of the flying attitude is important for the verification of the simulation code.

However, for such a narrow gap of a few nm, it is extremely difficult to measure the flying attitude for sliders, especially the flying height. One possible solution is to select several points (at least three) for measurement on the rails that have large spacing. These points should be easy to measure, even when some points on the slider may be at zero flying height and contact the disk. Based on the flying heights of these measured points, after subtracting the effect of the different recesses and other geometries contributing to the flying heights of these points, we can easily calculate the flying attitude of the slider, and then calculate the flying height for any point on the slider.

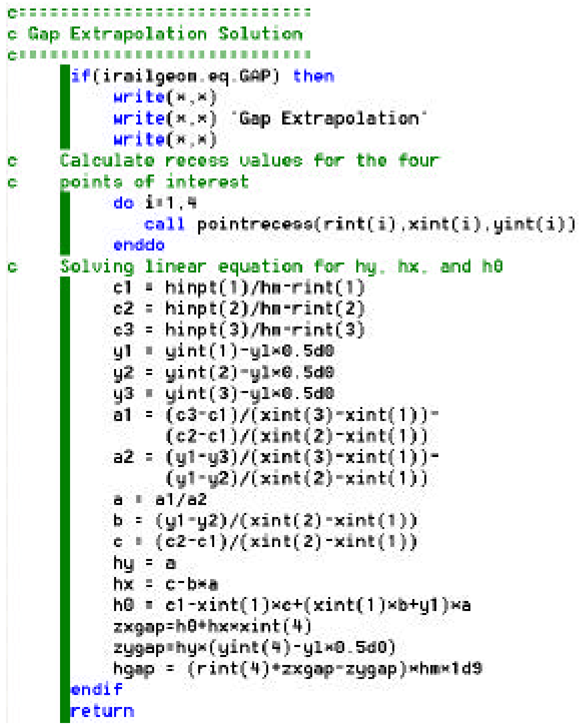
The Gap Extrapolation subroutine is developed to accomplish this. Its algorithm is very simple: just three points determine a plane, and taking into account the recesses and other geometries of theses points on the slider, we can calculate the flying attitude of the slider and the flying height at any other given points, as follows:

$$\text{Plane: } z = hx * x + hy * y + h0 \ \ ===> \ \ \begin{bmatrix} hx \\ hy \\ h0 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix}^{-1} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} \qquad (8)$$

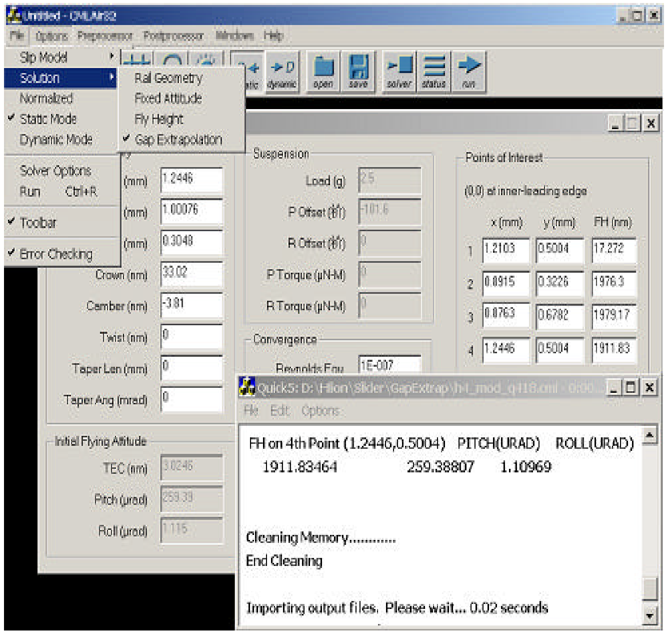
$$\frac{hinpt(i)}{hm} - recess(i) = h0 + hx \times xinpt(i) - hy \times [yinpt(i) - 0.5 \times yl] \qquad (9)$$

Where i = 1,2,3 is the index of the three measured point. h0 is the flying height on at the TEC, hx is the pitch angle of the slider and hy is the roll angle. xinpt(i) and yinpt(i) are the x and y positions of the measured point i. hinpt(i) is the flying height of the measured point i, and recess(i) is the rail recess height for the measured point i. The code implementation of Gap Extrapolation is shown in Fig.17 a).

Fig. 17 b) shows the interface for Gap Extrapolation. From the options/solution menu, you click on “Gap Extrapolation” to activate this feature and put the solver in “Gap Extrapolation” mode. In this mode, when the user clicks the “run” button, the program will only calculate the gap according to the input values. You can go to the general window to define the input/output data. Input values include the x, y positions and the flying heights for three measured points. You also can specify a fourth point, for which the program will calculate and output its flying height. In the calculation the program outputs the flying attitude for this slider and the flying height for any other given point.



```
c::::::::::::::::::::::::::::::
c Gap Extrapolation Solution
c::::::::::::::::::::::::::::::
       if(irailgeom.eq.GAP) then
          write(*,*)
          write(*,*) 'Gap Extrapolation'
          write(*,*)
c      Calculate recess values for the four
c      points of interest
          do i=1,4
             call pointrecess(rint(i),xint(i),yint(i))
          enddo
c      Solving linear equation for hy, hx, and h0
          c1 = hinpt(1)/hm-rint(1)
          c2 = hinpt(2)/hm-rint(2)
          c3 = hinpt(3)/hm-rint(3)
          y1 = yint(1)-yl*0.5d0
          y2 = yint(2)-yl*0.5d0
          y3 = yint(3)-yl*0.5d0
          a1 = (c3-c1)/(xint(3)-xint(1))-
               (c2-c1)/(xint(2)-xint(1))
          a2 = (y1-y3)/(xint(3)-xint(1))-
               (y1-y2)/(xint(2)-xint(1))
          a = a1/a2
          b = (y1-y2)/(xint(2)-xint(1))
          c = (c2-c1)/(xint(2)-xint(1))
          hy = a
          hx = c-b*a
          h0 = c1-xint(1)*c+(xint(1)*b+y1)*a
          zxgap=h0+hx*xint(4)
          zygap=hy*(yint(4)-yl*0.5d0)
          hgap = (rint(4)*zxgap-zygap)*hm*1d9
       endif
       return
```



(mm) 1.2446
(mm) 1.00076
(mm) 0.3048
Crown (nm) 33.02
Camber (nm) -3.81
Twist (nm) 0
Taper Len (mm) 0
Taper Ang (mrad) 0

Initial Flying Altitude
TEC (nm) 3.0246
Pitch (μrad) 259.39
Roll (μrad) 1.115

Suspension
Load (g) 2.5
P Offset (mm) -101.6
R Offset (mm) 0
P Torque (μN-M) 0
R Torque (μN-M) 0

Convergence
Reynolds Equ 1E-007

Points of Interest
(0,0) at inner-leading edge

	x (mm)	y (mm)	FH (nm)
1	1.2103	0.5004	17.272
2	0.8915	0.3226	1976.3
3	0.8763	0.6782	1979.17
4	1.2446	0.5004	1911.83

```
FH on 4th Point (1.2446,0.5004)  PITCH(URAD)  ROLL(URAD)
  1911.83464            259.38807      1.10969


Cleaning Memory............
End Cleaning

Importing output files.  Please wait... 0.02 seconds
```

a) Code implementation b) Interface

Fig. 17 Implementation of Gap Extrapolation

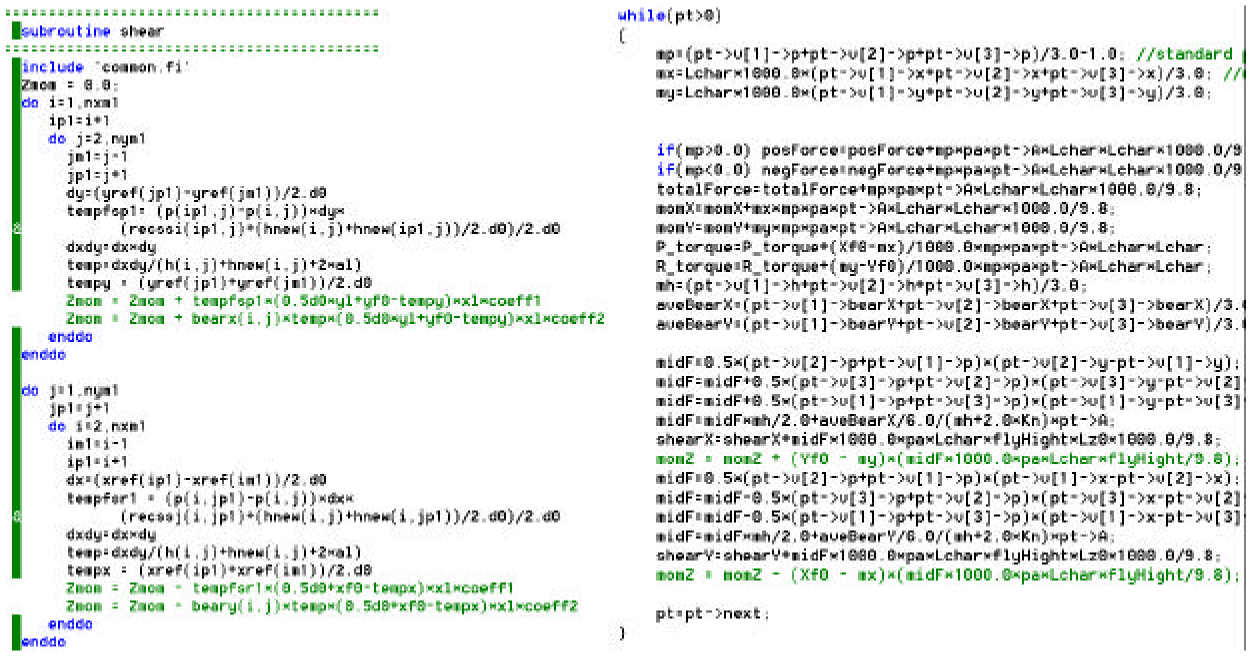
8. Z Moment Calculation

8.1 The effect of Z moment

Currently disk drive servo systems have a single actuator located at the base of the arm. It provides the force and moment to access and follow data tracks. In traditional slider designs the Z moment is ignored in most situations. Some simple methods have been used to roughly estimate the Z moment. As track densities continue to increase several researchers are considering dual stage actuators with the fine adjustment stage provided by a MEMS actuator. Because of the low level of force and moment available in these systems there is more concern about the air bearing shear force and Z moment. So it is important to have an accurate prediction of the Z moment that is produced by shear forces about the center of the slider.

8.2 Code Implementation

The code implementation for Z moment is quite simple. We just need to add several lines of code in the force calculation, as shown in Fig. 18. Z moment is calculated on every grid using shear force multiply by its moment arm distance to the center of the slider. Integrating all of these Z moments on every grid together, we get the total Z moment. The moment is positive if its direction is along the positive Z direction.



```
c=========================================
 subroutine shear
c=========================================
 include 'common.fi'
 Zmom = 0.0;
 do i=1,nxm1
    ip1=i+1
    do j=2,nym1
      jm1=j-1
      jp1=j+1
      dy=(yref(jp1)-yref(jm1))/2.d0
      tempfop1= (p(ip1,j)-p(i,j))*dy*
&           (recossi(ip1,j)*(hnew(i,j)+hnew(ip1,j))/2.d0)/2.d0
      dxdy=dx*dy
      temp=dxdy/(h(i,j)+hnew(i,j)+2*al)
      tempy = (yref(jp1)+yref(jm1))/2.d0
      Zmom = Zmom + tempfop1*(0.5d0*yl+yf0-tempy)*xl*coeff1
      Zmom = Zmom + bearx(i,j)*temp*(0.5d0*yl+yf0-tempy)*xl*coeff2
    enddo
 enddo

 do j=1,nym1
    jp1=j+1
    do i=2,nxm1
      im1=i-1
      ip1=i+1
      dx=(xref(ip1)-xref(im1))/2.d0
      tempfor1 = (p(i,jp1)-p(i,j))*dx*
&           (recosj(i,jp1)*(hnew(i,j)+hnew(i,jp1))/2.d0)/2.d0
      dxdy=dx*dy
      temp=dxdy/(h(i,j)+hnew(i,j)+2*al)
      tempx = (xref(ip1)+xref(im1))/2.d0
      Zmom = Zmom - tempfor1*(0.5d0+xf0-tempx)*xl*coeff1
      Zmom = Zmom - beary(i,j)*temp*(0.5d0+xf0-tempx)*xl*coeff2
    enddo
 enddo
```

```
while(pt>0)
{
    mp=(pt->v[1]->p+pt->v[2]->p+pt->v[3]->p)/3.0-1.0; //standard
    mx=Lchar*1000.0*(pt->v[1]->x+pt->v[2]->x+pt->v[3]->x)/3.0; //
    my=Lchar*1000.0*(pt->v[1]->y+pt->v[2]->y+pt->v[3]->y)/3.0;


    if(mp>0.0) posForce=posForce+mp*pa*pt->A*Lchar*Lchar*1000.0/9
    if(mp<0.0) negForce=negForce+mp*pa*pt->A*Lchar*Lchar*1000.0/9
    totalForce=totalForce+mp*pa*pt->A*Lchar*Lchar*1000.0/9.8;
    momX=momX+mx*mp*pa*pt->A*Lchar*Lchar*1000.0/9.8;
    momY=momY+my*mp*pa*pt->A*Lchar*Lchar*1000.0/9.8;
    P_torque=P_torque+(Xf0-mx)/1000.0*mp*pa*pt->A*Lchar*Lchar;
    R_torque=R_torque+(my-Yf0)/1000.0*mp*pa*pt->A*Lchar*Lchar;
    mh=(pt->v[1]->h+pt->v[2]->h+pt->v[3]->h)/3.0;
    aveBearX=(pt->v[1]->bearX+pt->v[2]->bearX+pt->v[3]->bearX)/3.
    aveBearY=(pt->v[1]->bearY+pt->v[2]->bearY+pt->v[3]->bearY)/3.

    midF=0.5*(pt->v[2]->p+pt->v[1]->p)*(pt->v[2]->y-pt->v[1]->y);
    midF=midF+0.5*(pt->v[3]->p+pt->v[2]->p)*(pt->v[3]->y-pt->v[2]
    midF=midF+0.5*(pt->v[1]->p+pt->v[3]->p)*(pt->v[1]->y-pt->v[3]
    midF=midF*mh/2.0+aveBearX/6.0/(mh+2.0*Kn)*pt->A;
    shearX=shearX+midF*1000.0*pa*Lchar*flyHight*Lz0*1000.0/9.8;
    momZ = momZ + (Yf0 - my)*(midF*1000.0*pa*Lchar*flyHight/9.8);
    midF=0.5*(pt->v[2]->p+pt->v[1]->p)*(pt->v[1]->x-pt->v[2]->x);
    midF=midF-0.5*(pt->v[3]->p+pt->v[2]->p)*(pt->v[3]->x-pt->v[2]
    midF=midF-0.5*(pt->v[1]->p+pt->v[3]->p)*(pt->v[1]->x-pt->v[3]
    midF=midF*mh/2.0+aveBearY/6.0/(mh+2.0*Kn)*pt->A;
    shearY=shearY+midF*1000.0*pa*Lchar*flyHight*Lz0*1000.0/9.8;
    momZ = momZ - (Xf0 - mx)*(midF*1000.0*pa*Lchar*flyHight/9.8);

    pt=pt->next;
}
```

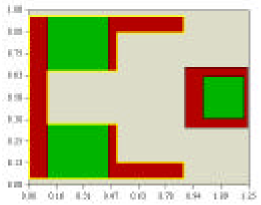
a) Code for Quick4 b) Code for Quick5

Fig. 18 Implementation of Z moment produced by shear force.

8.3 Simulation Result

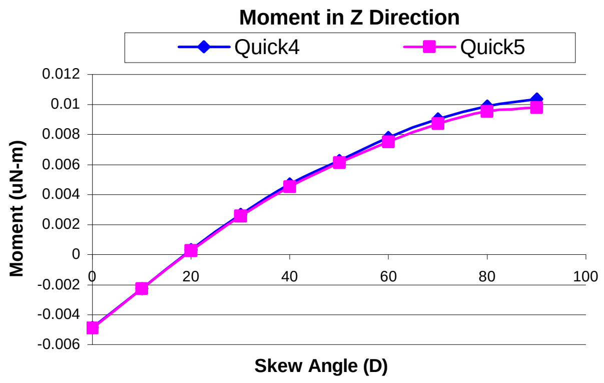
Intuitively, it is believed that the air bearing shear forces may generate a relatively high Z moment when the skew angle is large. That is, the larger the skew angle, the higher the moment generated by shear forces. But this is not what we found in all cases. The relationship between Z moment and skew angle is, in fact, dependent on the design of the slider.

The first slider used to demonstrate the relationship between Z moment and skew angle is the IBM Travelstar-type slider with a 2.5g force and zero moment pre-load from the suspension. The disk rotation speed is 7200 rpm and the slider is flying around 20nm. Fig. 19 shows the slider, its initial conditions and the dependence of the Z moment to the skew angle. We see that from both solvers Quick4 and Quick5 the Z moment increases when the skew angle increases. The direction of the Z moment changes from negative to positive as the skew angle increases.



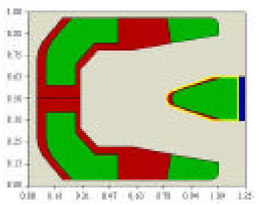
Initial conditions:
TEC: 16 nm
Pitch: 100 urad
Roll: 1 urad

a) Slider and initial conditions



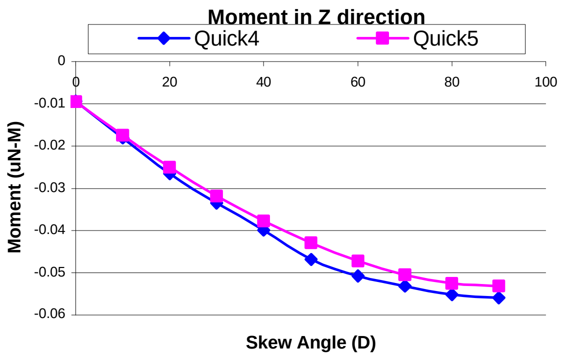
b) Z moment variation with skew angle

Fig. 19 Z moment for IBM tri-pad slider



Initial conditions:
TEC: 7nm
Pitch: 150 urad
Roll: 5 urad

a) Slider and initial conditions



b) Z moment variation with skew angle

Fig. 20 Z moment of NSIC 7nm slider

The second slider used to demonstrate the relationship between Z moment and skew angle is the NSIC 7nm slider with a 1.5g force and zero moment pre-load from the suspension. The disk rotation speed is 7200 rpm and the slider is flying around 7nm. Fig. 20 shows the slider, its initial conditions and the variation of Z moment with the skew angle. Contrary to the result from previous slider, here we see that the Z moment decreases when the skew angle increases. The moment is always in the negative Z direction.

We can draw the following conclusions based on the above analysis and our experiences:

1) More simulation results show that for different sliders, Z moments have different patterns of dependence on skew angle.

2) Z moment is relatively small, and is around several $10^{-2}\, mN \bullet m$. This magnitude of Z moment is one or two orders less than that of the moment in X and Y directions.

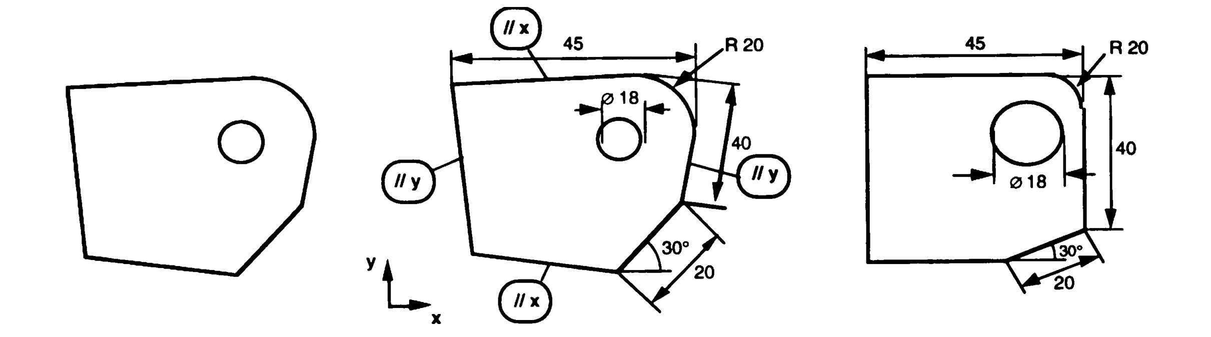
3) The Z moment vs. skew angle curve has a decreasing slope magnitude.

9. Geometric Modeling in Slider Design

In air bearing design, geometric shapes of the sliders play an important role in their static and dynamic performance. To achieve good performance, sliders in modern hard disk drive are becoming more complex in two ways: 1) Many more rails and other components are being added in the ABS. 2) The geometries of the rails and the pads are becoming more complex, and many of them now have streamlined shapes. However, in the current CMLAir32 we still use only lines and planes to define all of the shapes. For rail design, we only can use line segments to approximate its shape. If we have a curved shape in the rail, we have to use many short lines to approximate it. This requires much calculation and data input from the user, making the process error-prone and requiring substantial labor. At the same time, a large number of short line segments in a rail make it confusing for solvers to generate the adaptive mesh based on its geometry. To describe the geometry of the slider, straight lines are not optimal. We also need circles, conic curves, splines, as well as other free form curves to define the rail geometry.

In addition to the geometric shapes, the designer also has to consider a wide variety of technological criteria, such as the flying attitude requirement, structural and dynamic properties, assembling constraints, manufacturing costs and so on, for defining the geometries of the slider. Therefore, the slider and its air bearing are usually designed iteratively, modifying the geometries and evaluating properties from different points of view.

Parametric modeling should improve the process of air bearing design, simulation and optimization. Parametric design is an approach to product modeling, which associates engineering knowledge with geometry in a product design by means of geometric constraints. It has been considered an indispensable tool in many applications such as mechanical part design, tolerance analysis, and knowledge-based design automation. Parametric capabilities for air bearing design enable the description of the geometry of sliders controlled by parameters. They are based on the bi-directional dependencies between dimensional or geometrical entities. These dependencies include both constraints and dimensions. Such constraints are parallelity, perpendicularity, concentricity, etc. With such dependencies, a modification of an entity will cause a successive modification of the associated ones. The modification (variation) of geometry can be performed by an easy (interactive) definition of the parameters and a successive execution of the variation, as shown in Fig. 21.



a) The user opens with a freehand sketch

b) The user specifies dimensions and other geometric constraints

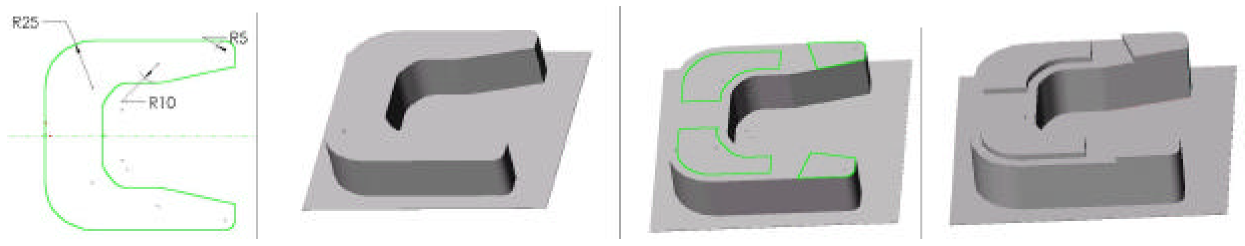
c) The system regenerates the shape in accordance with the specified constraints

Fig. 21 Parametric design for a 2D sketch

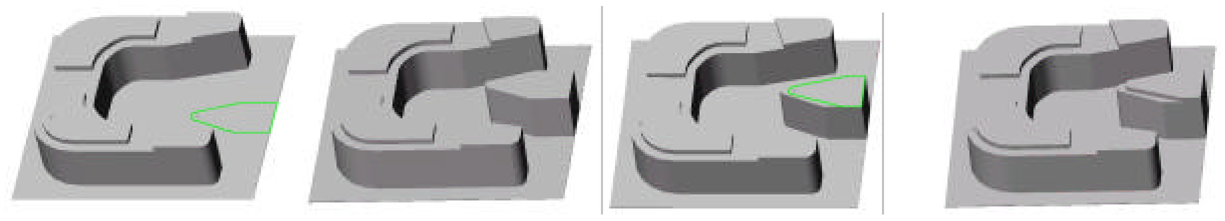
We will demonstrate that the efficiency of air bearing optimization and sensitivity analysis can be greatly improved based on slider parametric modeling. One central thing for air bearing optimization is to vary the geometry of the slider, especially its rails, to generate new samples to see whether a better result can be achieved compared to the target. The optimization procedure can be summarized as follows: based on the initial design, generated some new samples according to the optimization techniques; calculate the flying attitude for those samples in the solver; evaluate the results and determine the new direction of variances to generate new samples according to optimization techniques. This procedure repeats over and over in the optimization space and approaches the final target. For computationally intensive sample evaluations like air bearing flying attitude calculations, the time and CPU power consumed by optimization is very sensitive to the number of parameters to be varied. The more parameters, the more dimensions and the larger the search space for the optimization. The larger the optimization space, the more samples have to be evaluated, and the longer the time is needed for optimization.

Parametric design can simplify the control of the geometry of the slider. For air bearing optimization it can greatly decrease the parameters needed to control the shape of the slider in order to obtain an optimal design. With parametric modeling the overall geometry of the slider, including the base, the rails and the pad, are controlled by several parameters. One needs only a few parameters to control the geometry of each rail. Those parameters can easily be controlled and modified by both the human and the optimization program. By intelligently organizing the constraints and dimensions enforced on the slider, the optimization program can modify fewer parameters to vary the geometry of the slider in a larger search space.

The geometry construction procedure of parametric modeling is briefly shown in Fig. 22. On a base plane, one constructs the 2D sketch for a rail. Then you add dimensions and constraints on the rail, which gives you freedom to modify the sketch. The 2D sketch is extruded into a 3D rail. In addition to strait line extrusion, one can also define complex trajectory curves for the extrusion, creating wall profiles. Additional sketches, and 3D geometries can be constructed on top of this rail, and so on. Finally we get the complete design of the slider.



1) 2D sketch for Rail 1 2) Model of Rail 1 3) 2D sketch for Rail 2, 3, 4, 5 4) Model of Rails



5) 2D sketch for pad 6) Model of pad 7) 2D sketch for Rail on pad 8) Model of completed slider

Fig. 22 Geometric modeling for slider design

Although the geometric modeling module has not yet been incorporated into the CMLAir32 program, we have finished all the system design, and it should be relatively easy to implement it in future CML air bearing design programs.

Reference

Cha, E., 1993, “Numerical Analysis of Head-Disk Assembly Dynamics for Shaped Rail Sliders with Sub-ambient Pressure Regions,” Doctoral Dissertation, Department of Mechanical Engineering, University of California, Berkeley.

Grisso, R. and Bogy, D.B., “The CML Air Bearing Design Program (CMLAir32), Version 5 User Manual”, Technical Report No. 99-024, Computer Mechanics Laboratory, Department of Mechanical Engineering, University of California, Berkeley.

Hoschek, J., 1994, “Parametric and Variational Design”, Edition, B.G. Teubner Stuttgart, Germany.

Huang, W., 1997, “A Study of Rarefied and Contact Air Bearing Modeling for Hard Disk Drives Using the Direct Simulation Monte Carlo Method”, Doctoral Dissertation, Department of Mechanical Engineering, University of California, Berkeley.

Kang, S.-C., 1997, “A Kinetic Theory Description for Molecular Lubrication”, Doctoral Dissertation, Carnegie Mellon University.

Kennard, E. H., 1938, “Kinetic Theory of Gases with an Introduction to Statistical Mechanics”, McGraw-Hill, New York, NY.

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

Rettner, C., 1997, “Determination of Accommodation Coefficients for N_2 at Disk-Drive Air-Bearing Surfaces”, ASME J. of Tribology, Vol. 119, pp. 588-589.